Version 9.6.5 - Queries Release Notes

New Queries:

Language	Group	Name	CWE
Rust	Rust_High_Risk	Plaintext_Password_Storage	256
Rust	Rust_Low_Visibility	Overly_Permissive_Cross_Origin_Resource_Sharing_Policy	942
Rust	Rust_Low_Visibility	Privacy_Violation_in_Files	538
Rust	Rust_Low_Visibility	Privacy_Violation_in_Logs	200
Rust	Rust_Medium_Threat	Open_Redirect	601
Rust	Rust_Medium_Threat	SSL_Verification_Bypass	0

Changed Queries:

Language	Group	Name	CWE	Changed Fields
ASP	ASP_Heuristic	Heuristic_2nd_Order_SQL_Injection	89	**Source** has changed
ASP	ASP_Heuristic	Heuristic_CSRF	352	**Source** has changed
ASP	ASP_Heuristic	Heuristic_DB_Parameter_Tampering	284	**Source** has changed
ASP	ASP_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed
ASP	ASP_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
ASP	ASP_Heuristic	Heuristic_Stored_XSS	79	**Source** has changed
ASP	ASP_High_Risk	UTF7_XSS	79	**Source** has changed
ASP	ASP_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed
ASP	ASP_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171	**Source** has changed
ASP	ASP_Low_Visibility	Improper_Transaction_Handling	460	**Source** has changed
ASP	ASP_Low_Visibility	JavaScript_Hijacking	352	**Source** has changed
ASP	ASP_Low_Visibility	XSS_Evasion_Attack	79	**Source** has changed
ASP	ASP_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
ASP	ASP_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
ASP	ASP_Medium_Threat	Reflected_XSS_Specific_Clients	79	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Async_Future_Method_Inside_Loops	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Bulkify_Apex_Methods_Using_Collections_In_Methods	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Hardcoded_Messages	547	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Hardcoding_Ids	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Hardcoding_Of_Trigger_New	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Hardcoding_Of_Trigger_Old	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Multiple_Trigger_On_same_sObject	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Use_of_Hard_Coded_Cryptographic_Key	321	**Source** has changed
Apex	Apex_Force_com_Critical_Security_Risk	Reflected_XSS	79	**Source** has changed
Apex	Apex_Force_com_Serious_Security_Risk	Frame_Spoofing	79	**Source** has changed
Apex	Apex_Force_com_Serious_Security_Risk	inputText_Ignoring_FLS	0	**Source** has changed
Apex	Apex_Force_com_Serious_Security_Risk	Sharing	472	**Source** has changed
Apex	Apex_ISV_Quality_Rules	Empty_IfStmt	0	**Source** has changed
Apex	Apex_ISV_Quality_Rules	Find_Exposed_Test_Data	0	**Source** has changed
Apex	Apex_ISV_Quality_Rules	SOQL_Dynamic_null_in_Where	0	**Source** has changed
Apex	Apex_ISV_Quality_Rules	SOSL_With_Where_Clause	0	**Source** has changed
Apex	Apex_ISV_Quality_Rules	Warn_About_Viewstate_Size_Limit	0	**Source** has changed
Apex	Apex_Low_Visibility	Escape_False_Warning	0	**Source** has changed
Apex	Apex_Low_Visibility	Hardcoded_Password	259	**Source** has changed

Language	Group	Name	CWE	Changed Fields
Apex	Apex_Low_Visibility	Parameter_Tampering	472	**Source** has changed
Apex	Apex_Low_Visibility	Password_misuse	0	**Source** has changed
Apex	Apex_Low_Visibility	Potential_Frame_Injection	0	**Source** has changed
Apex	Apex_Low_Visibility	Potential_URL_Redirection_Attack	601	**Source** has changed
Apex	Apex_Low_Visibility	Privacy_Violation	359	**Source** has changed
Apex	Apex_Low_Visibility	Second_Order_SOQL_SOSL_Injection	89	**Source** has changed
Apex	Apex_Low_Visibility	Verbose_Error_Reporting	209	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Improper_Index_Access	129	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Overflow_boundcpy_WrongSizeParam	121	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Overflow_LongString	120	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Overflow_Loops	193	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Overflow_Loops_Old	193	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Overflow_Unbounded_Format	120	**Source** has changed
CPP	CPP_Buffer_Overflow	Missing_Precision	120	**Source** has changed
CPP	CPP_Buffer_Overflow	Off_by_One_Error_in_Arrays	193	**Source** has changed
CPP	CPP_Buffer_Overflow	Off_by_One_Error_in_Loops	193	**Source** has changed
CPP	CPP_Buffer_Overflow	Off_by_One_Error_in_Methods	193	**Source** has changed
CPP	CPP_Buffer_Overflow	Potential_Precision_Problem	120	**Source** has changed
CPP	CPP_Buffer_Overflow	String_Termination_cin	170	**Source** has changed
CPP	CPP_Buffer_Overflow	String_Termination_Error	170	**Source** has changed
CPP	CPP_Heuristic	Heuristic_2nd_Order_Buffer_Overflow_malloc	120	**Source** has changed
CPP	CPP_Heuristic	Heuristic_2nd_Order_Buffer_Overflow_read	120	**Source** has changed
CPP	CPP_Heuristic	Heuristic_2nd_Order_SQL_Injection	89	**Source** has changed
CPP	CPP_Heuristic	Heuristic_Buffer_Improper_Index_Access	129	**Source** has changed
CPP	CPP_Heuristic	Heuristic_Buffer_Overflow_malloc	120	**Source** has changed
CPP	CPP_Heuristic	Heuristic_Buffer_Overflow_read	120	**Source** has changed
CPP	CPP_Heuristic	Heuristic_CGI_Stored_XSS	79	**Source** has changed
CPP	CPP_Heuristic	Heuristic_DB_Parameter_Tampering	284	**Source** has changed
CPP	CPP_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed
CPP	CPP_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
CPP	CPP_Heuristic	Heuristic_Unchecked_Return_Value	252	**Source** has changed
CPP	CPP_Heuristic	Potential_Off_by_One_Error_in_Loops	193	**Source** has changed
CPP	CPP_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed
CPP	CPP_Low_Visibility	Creation_of_chroot_Jail_without_Changing_Working_Directory	243	**Source** has changed
CPP	CPP_Low_Visibility	NULL_Pointer_Dereference	476	**Source** has changed
CPP	CPP_Low_Visibility	Potential_Path_Traversal	36	**Source** has changed
CPP	CPP_Low_Visibility	Sizeof_Pointer_Argument	467	**Source** has changed
CPP	CPP_Low_Visibility	Stored_Blind_SQL_Injections	89	**Source** has changed
CPP	CPP_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
CPP	CPP_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
CPP	CPP_Medium_Threat	Divide_By_Zero	369	**Source** has changed
CPP	CPP_Medium_Threat	Inadequate_Pointer_Validation	477	**Source** has changed
CPP	CPP_Medium_Threat	Memory_Leak	401	**Source** has changed
CPP	CPP_MISRA_C	R05_07_Identifier_Name_Reused	0	**Source** has changed
CPP	CPP_MISRA_C	R08_03_Identical_Function_Decl_Def	0	**Source** has changed
CPP	CPP_MISRA_C	R08_07_Block_Scope_Obj_If_Used_By_Single_Function	0	**Source** has changed
CPP	CPP_MISRA_C	R13_06_Loop_Iterator_Modified_In_Loop_Body	0	**Source** has changed

Language	Group	Name	CWE	Changed Fields
CPP	CPP_MISRA_C	R14_08_Not_Compound_Switch_Or_Iteration_Statement	0	**Source** has changed
CPP	CPP_MISRA_C	R14_09_Not_Compound_If_Or_Else	0	**Source** has changed
CPP	CPP_MISRA_C	R14_10_If_Else_If_Not_Ending_With_Else	0	**Source** has changed
CPP	CPP_MISRA_C	R16_03_Function_Prototype_Without_Identifiers	0	**Source** has changed
CPP	CPP_MISRA_C	R16_05_Function_Prototype_Declaration_Without_Parameters	0	**Source** has changed
CPP	CPP_MISRA_C	R16_07_Parameter_Pointer_To_Const_Where_Not_Modified	0	**Source** has changed
CPP	CPP_MISRA_C	R16_08_Non_Explicit_Return_Statement_In_Non_Void_Function	0	**Source** has changed
CPP	CPP_MISRA_CPP	R02_10_02_Identifiers_Hide_Outer_Scope_Identifiers	0	**Source** has changed
CPP	CPP_MISRA_CPP	R02_10_05_Non_Member_Static_Name_Reuse	10751	**Source** has changed
CPP	CPP_MISRA_CPP	R03_02_01_Identical_Function_and_Object_Decl_Def	0	**Source** has changed
CPP	CPP_MISRA_CPP	R03_04_01_Obj_Defined_Outside_Minimal_Scope	0	**Source** has changed
CPP	CPP_MISRA_CPP	R06_03_01_Not_Compound_Switch_Or_Iteration_Statement	0	**Source** has changed
CPP	CPP_MISRA_CPP	R06_04_01_Not_Compound_If_Or_Else	0	**Source** has changed
CPP	CPP_MISRA_CPP	R06_04_02_If_Else_If_Not_Ending_With_Else	0	**Source** has changed
CPP	CPP_MISRA_CPP	R06_05_01_Single_Non_Float_LC	0	**Source** has changed
CPP	CPP_MISRA_CPP	R06_05_02_Loop_Counter_Modify	10755	**Source** has changed
CPP	CPP_MISRA_CPP	R06_05_03_Change_Lc_In_St_And_Cond	10756	**Source** has changed
CPP	CPP_MISRA_CPP	R06_05_04_Incremental_Modified	10757	**Source** has changed
CPP	CPP_MISRA_CPP	R06_05_05_Lcv_Change_In_For_Stmt	10758	**Source** has changed
CPP	CPP_MISRA_CPP	R06_05_06_Bool_Lcv_Change	10759	**Source** has changed
CPP	CPP_MISRA_CPP	R07_01_01_Declare_Const_if_not_Modified	10784	**Source** has changed
CPP	CPP_MISRA_CPP	R07_01_02_Declare_Ref_Const_if_not_Modified	10785	**Source** has changed
CPP	CPP_MISRA_CPP	R07_05_02_Address_Assignment_out_of_Scope	10792	**Source** has changed
CPP	CPP_MISRA_CPP	R08_04_03_Explicit_Return_Throw	0	**Source** has changed
CPP	CPP_MISRA_CPP	R10_03_02_Find_Override_Without_Virtual	10796	**Source** has changed
CPP	CPP_Stored_Vulnerabilities	Stored_DB_Parameter_Tampering	284	**Source** has changed
CPP	CPP_Weak_Cryptography	Use_Of_Weak_Hashing_Primitive	326	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Aptca_Methods_Call_Non_Aptca_Methods	0	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Exposure_of_Resource_to_Wrong_Sphere	493	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	GetLastWin32Error_Is_Not_Called_After_Pinvoke	10018	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Hardcoded_Connection_String	798	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Insufficient_Logging_of_Sensitive_Operations	778	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Leftover_Debug_Code	489	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Pages_Without_Global_Error_Handler	544	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	PersistSecurityInfo_is_True	0	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Unclosed_Objects	459	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Undocumented_API	0	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Unvalidated_Arguments_Of_Public_Methods	0	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Use_of_System_Output_Stream	398	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Using_Of_Index_Instead_Of_Key	398	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Visible_Pointers	0	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_2nd_Order_SQL_Injection	89	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_CSRF	352	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_DB_Parameter_Tampering	284	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_Stored_XSS	79	**Source** has changed

Language	Group	Name	CWE	Changed Fields
CSharp	CSharp_High_Risk	Connection_String_Injection	99	**Source** has changed
CSharp	CSharp_High_Risk	JWT_No_Signature_Verification	287	**Source** has changed
CSharp	CSharp_High_Risk	Reflected_XSS_All_Clients	79	**Source** has changed
CSharp	CSharp_High_Risk	Resource_Injection	99	**Source** has changed
CSharp	CSharp_High_Risk	Unsafe_Reflection	470	**Source** has changed
CSharp	CSharp_High_Risk	UTF7_XSS	79	**Source** has changed
CSharp	CSharp_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed
CSharp	CSharp_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171	**Source** has changed
CSharp	CSharp_Low_Visibility	Client_Side_Only_Validation	602	**Source** has changed
CSharp	CSharp_Low_Visibility	Heap_Inspection	244	**Source** has changed
CSharp	CSharp_Low_Visibility	Impersonation_Issue	520	**Source** has changed
CSharp	CSharp_Low_Visibility	Improper_Resource_Shutdown_or_Release	404	**Source** has changed
CSharp	CSharp_Low_Visibility	Improper_Session_Management	201	**Source** has changed
CSharp	CSharp_Low_Visibility	Improper_Transaction_Handling	460	**Source** has changed
CSharp	CSharp_Low_Visibility	Information_Exposure_Through_an_Error_Message	209	**Source** has changed
CSharp	CSharp_Low_Visibility	Information_Exposure_via_Headers	200	**Source** has changed
CSharp	CSharp_Low_Visibility	Insufficiently_Protected_Credentials	522	**Source** has changed
CSharp	CSharp_Low_Visibility	JavaScript_Hijacking	352	**Source** has changed
CSharp	CSharp_Low_Visibility	JWT_Excessive_Expiration_Time	613	**Source** has changed
CSharp	CSharp_Low_Visibility	JWT_Use_Of_Hardcoded_Secret	798	**Source** has changed
CSharp	CSharp_Low_Visibility	Log_Forging	117	**Source** has changed
CSharp	CSharp_Low_Visibility	Missing_Function_Level_Authorization	862	**Source** has changed
CSharp	CSharp_Low_Visibility	Off_By_One_Error	193	**Source** has changed
CSharp	CSharp_Low_Visibility	Open_Redirect	601	**Source** has changed
CSharp	CSharp_Low_Visibility	Overly_Permissive_Cross_Origin_Resource_Sharing_Policy	346	**Source** has changed
CSharp	CSharp_Low_Visibility	Potential_ReDoS	400	**Source** has changed
CSharp	CSharp_Low_Visibility	Potential_ReDoS_By_Injection	400	**Source** has changed
CSharp	CSharp_Low_Visibility	Potential_ReDoS_In_Code	400	**Source** has changed
CSharp	CSharp_Low_Visibility	Potential_ReDoS_In_Static_Field	400	**Source** has changed
CSharp	CSharp_Low_Visibility	Session_Clearing_Problems	613	**Source** has changed
CSharp	CSharp_Low_Visibility	Stored_Code_Injection	94	**Source** has changed
CSharp	CSharp_Low_Visibility	Stored_Command_Argument_Injection	88	**Source** has changed
CSharp	CSharp_Low_Visibility	Thread_Safety_Issue	567	**Source** has changed
CSharp	CSharp_Low_Visibility	Trust_Boundary_Violation_in_Session_Variables	501	**Source** has changed
CSharp	CSharp_Low_Visibility	Unencrypted_Web_Config_File	312	**Source** has changed
CSharp	CSharp_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
CSharp	CSharp_Low_Visibility	Use_of_Insufficiently_Random_Values	330	**Source** has changed
CSharp	CSharp_Low_Visibility	Use_of_RSA_Algorithm_without_OAEP	780	**Source** has changed
CSharp	CSharp_Low_Visibility	XSS_Evasion_Attack	79	**Source** has changed
CSharp	CSharp_Medium_Threat	Buffer_Overflow	120	**Source** has changed
CSharp	CSharp_Medium_Threat	CGI_XSS	79	**Source** has changed
CSharp	CSharp_Medium_Threat	Cookie_Injection	20	**Source** has changed
CSharp	CSharp_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
CSharp	CSharp_Medium_Threat	Hardcoded_password_in_Connection_String	547	**Source** has changed
CSharp	CSharp_Medium_Threat	HttpOnlyCookies	1004	**Source** has changed
CSharp	CSharp_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
CSharp	CSharp_Medium_Threat	Improper_Locking	667	**Source** has changed

Language	Group	Name	CWE	Changed Fields
CSharp	CSharp_Medium_Threat	Insecure_Cookie	614	**Source** has changed
CSharp	CSharp_Medium_Threat	JWT_Lack_Of_Expiration_Time	613	**Source** has changed
CSharp	CSharp_Medium_Threat	JWT_No_Expiration_Time_Validation	613	**Source** has changed
CSharp	CSharp_Medium_Threat	JWT_Sensitive_Information_Exposure	201	**Source** has changed
CSharp	CSharp_Medium_Threat	Missing_Column_Encryption	311	**Source** has changed
CSharp	CSharp_Medium_Threat	MVC_View_Injection	74	**Source** has changed
CSharp	CSharp_Medium_Threat	No_Request_Validation	20	**Source** has changed
CSharp	CSharp_Medium_Threat	Persistent_Connection_String	257	**Source** has changed
CSharp	CSharp_Medium_Threat	Privacy_Violation	359	**Source** has changed
CSharp	CSharp_Medium_Threat	Race_Condition_within_a_Thread	366	**Source** has changed
CSharp	CSharp_Medium_Threat	Reflected_XSS_Specific_Clients	79	**Source** has changed
CSharp	CSharp_Medium_Threat	Session_Fixation	384	**Source** has changed
CSharp	CSharp_Medium_Threat	SSL_Verification_Bypass	599	**Source** has changed
CSharp	CSharp_Medium_Threat	SSRF	74	**Source** has changed
CSharp	CSharp_Medium_Threat	Stored_LDAP_Injection	90	**Source** has changed
CSharp	CSharp_Medium_Threat	Stored_Path_Traversal	22	**Source** has changed
CSharp	CSharp_Medium_Threat	Stored_XPath_Injection	643	**Source** has changed
CSharp	CSharp_Medium_Threat	Unsafe_Object_Binding	915	**Source** has changed
CSharp	CSharp_Medium_Threat	Use_of_Cryptographically_Weak_PRNG	338	**Source** has changed
CSharp	CSharp_Medium_Threat	Use_of_Hard_coded_Cryptographic_Key	321	**Source** has changed
CSharp	CSharp_Metadata	ActiveMQ_Queues	0	**Source** has changed
CSharp	CSharp_Metadata	AWS_S3_Bucket_Usages	0	**Source** has changed
CSharp	CSharp_Metadata	AWS_SNS_Usages	0	**Source** has changed
CSharp	CSharp_Metadata	AWS_SQS_Usages	0	**Source** has changed
CSharp	CSharp_Metadata	Kafka_Topics	0	**Source** has changed
CSharp	CSharp_Metadata	Memcached_Urls	0	**Source** has changed
CSharp	CSharp_Metadata	RabbitMQ_Queues	0	**Source** has changed
CSharp	CSharp_Metadata	Redis_Urls	0	**Source** has changed
CSharp	CSharp_WebConfig	CookieLess_Session_State	0	**Source** has changed
CSharp	CSharp_WebConfig	CustomError	12	**Source** has changed
CSharp	CSharp_WebConfig	DebugEnabled	11	**Source** has changed
CSharp	CSharp_WebConfig	Directory_Browse	548	**Source** has changed
CSharp	CSharp_WebConfig	Elmah_Enabled	213	**Source** has changed
CSharp	CSharp_WebConfig	HardcodedCredentials	489	**Source** has changed
CSharp	CSharp_WebConfig	Missing_X_Frame_Options	1021	**Source** has changed
CSharp	CSharp_WebConfig	NonUniqueFormName	694	**Source** has changed
CSharp	CSharp_WebConfig	Password_in_Configuration_File	260	**Source** has changed
CSharp	CSharp_WebConfig	RequireSSL	614	**Source** has changed
CSharp	CSharp_WebConfig	SlidingExpiration	613	**Source** has changed
CSharp	CSharp_WebConfig	TraceEnabled	749	**Source** has changed
CSharp	CSharp_Windows_Phone	Client_Side_Injection	89	**Source** has changed
CSharp	CSharp_Windows_Phone	Failure_to_Implement_Least_Privilege	250	**Source** has changed
CSharp	CSharp_Windows_Phone	Hard_Coded_Cryptography_Key	321	**Source** has changed
CSharp	CSharp_Windows_Phone	Insufficient_Application_Layer_Protect	311	**Source** has changed
CSharp	CSharp_Windows_Phone	Poor_Authorization_and_Authentication	287	**Source** has changed
CSharp	CSharp_Windows_Phone	Side_Channel_Data_Leakage	200	**Source** has changed
Dart	Dart_Mobile_Low_Visibility	Parameter_Tampering	472	**Source** has changed

Language	Group	Name	CWE	Changed Fields
Go	Go_High_Risk	Second_Order_SQL_Injection	89	**Source** has changed
Go	Go_Low_Visibility	Use_of_Hardcoded_Password	259	**Source** has changed
Go	Go_Medium_Threat	SSRF	918	**Source** has changed
Groovy	Groovy_Best_Coding_Practice	Getter_Method_Could_Be_Property	398	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_2nd_Order_SQL_Injection	89	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_CGI_Stored_XSS	79	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_CSRF	352	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_DB_Parameter_Tampering	284	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
Groovy	Groovy_Heuristic	Heuristic_Stored_XSS	79	**Source** has changed
Groovy	Groovy_High_Risk	UTF7_XSS	79	**Source** has changed
Groovy	Groovy_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed
Groovy	Groovy_Low_Visibility	Channel_Accessible_by_NonEndpoint	300	**Source** has changed
Groovy	Groovy_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171	**Source** has changed
Groovy	Groovy_Low_Visibility	Improper_Resource_Shutdown_or_Release	404	**Source** has changed
Groovy	Groovy_Low_Visibility	Improper_Transaction_Handling	460	**Source** has changed
Groovy	Groovy_Low_Visibility	Leaving_Temporary_File	376	**Source** has changed
Groovy	Groovy_Low_Visibility	Plaintext_Storage_in_a_Cookie	315	**Source** has changed
Groovy	Groovy_Low_Visibility	Potenial_UTF7_XSS	79	**Source** has changed
Groovy	Groovy_Low_Visibility	Potential_ReDoS	400	**Source** has changed
Groovy	Groovy_Low_Visibility	Potential_ReDoS_By_Injection	400	**Source** has changed
Groovy	Groovy_Low_Visibility	Potential_ReDoS_In_Match	400	**Source** has changed
Groovy	Groovy_Low_Visibility	Potential_ReDoS_In_Replace	400	**Source** has changed
Groovy	Groovy_Low_Visibility	Potential_ReDoS_In_Static_Field	400	**Source** has changed
Groovy	Groovy_Low_Visibility	Reliance_on_Cookies_in_a_Decision	784	**Source** has changed
Groovy	Groovy_Low_Visibility	Use_of_Client_Side_Authentication	603	**Source** has changed
Groovy	Groovy_Low_Visibility	Use_Of_getenv	589	**Source** has changed
Groovy	Groovy_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Groovy	Groovy_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
Groovy	Groovy_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
Groovy	Groovy_Medium_Threat	Multiple_Binds_to_the_Same_Port	605	**Source** has changed
Groovy	Groovy_Medium_Threat	Unnormalize_Input_String	20	**Source** has changed
Groovy	Groovy_Stored	Stored_HTTP_Response_Splitting	113	**Source** has changed
Java	Java_Android	Accessible_Content_Provider	668	**Source** has changed
Java	Java_Android	Exported_Service_Without_Permissions	668	**Source** has changed
Java	Java_Android	Insufficient_Sensitive_Application_Layer	319	**Source** has changed
Java	Java_GWT	JSON_Hijacking	352	**Source** has changed
Java	Java_Heuristic	Heuristic_2nd_Order_SQL_Injection	89	**Source** has changed
Java	Java_Heuristic	Heuristic_CGI_Stored_XSS	79	**Source** has changed
Java	Java_Heuristic	Heuristic_CSRF	352	**Source** has changed
Java	Java_Heuristic	Heuristic_DB_Parameter_Tampering	284	**Source** has changed
Java	Java_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed
Java	Java_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
Java	Java_Heuristic	Heuristic_Stored_XSS	79	**Source** has changed
Java	Java_High_Risk	Reflected_XSS_All_Clients	79	**Source** has changed
Java	Java_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed

Language	Group	Name	CWE	Changed Fields
Java	Java_Low_Visibility	Channel_Accessible_by_NonEndpoint	300	**Source** has changed
Java	Java_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171	**Source** has changed
Java	Java_Low_Visibility	Improper_Resource_Access_Authorization	285	**Source** has changed
Java	Java_Low_Visibility	Improper_Resource_Shutdown_or_Release	404	**Source** has changed
Java	Java_Low_Visibility	Plaintext_Storage_in_a_Cookie	315	**Source** has changed
Java	Java_Low_Visibility	Potential_ReDoS	400	**Source** has changed
Java	Java_Low_Visibility	Potential_ReDoS_By_Injection	400	**Source** has changed
Java	Java_Low_Visibility	Potential_ReDoS_In_Match	400	**Source** has changed
Java	Java_Low_Visibility	Potential_ReDoS_In_Replace	400	**Source** has changed
Java	Java_Low_Visibility	Potential_ReDoS_In_Static_Field	400	**Source** has changed
Java	Java_Low_Visibility	Reliance_on_Cookies_in_a_Decision	784	**Source** has changed
Java	Java_Low_Visibility	Suspected_XSS	79	**Source** has changed
Java	Java_Low_Visibility	Use_of_Client_Side_Authentication	603	**Source** has changed
Java	Java_Low_Visibility	Use_Of_getenv	589	**Source** has changed
Java	Java_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Java	Java_Low_Visibility	UTF7_XSS	79	**Source** has changed
Java	Java_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
Java	Java_Medium_Threat	Hardcoded_password_in_Connection_String	547	**Source** has changed
Java	Java_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
Java	Java_Medium_Threat	Improper_Restriction_of_Stored_XXE_Ref	611	**Source** has changed
Java	Java_Medium_Threat	Missing_HSTS_Header	346	**Source** has changed
Java	Java_Medium_Threat	Multiple_Binds_to_the_Same_Port	605	**Source** has changed
Java	Java_Medium_Threat	Privacy_Violation	359	**Source** has changed
Java	Java_Medium_Threat	SSRF	918	**Source** has changed
Java	Java_Medium_Threat	Unnormalize_Input_String	20	**Source** has changed
Java	Java_Medium_Threat	Unvalidated_Forwards	819	**Source** has changed
Java	Java_Medium_Threat	Unvalidated_SSL_Certificate_Hostname	297	**Source** has changed
Java	Java_Medium_Threat	Use_of_a_One_Way_Hash_with_a_Predictable_Salt	760	**Source** has changed
Java	Java_Medium_Threat	Use_of_Insufficiently_Random_Values	330	**Source** has changed
Java	Java_Potential	Potential_Code_Injection	94	**Source** has changed
Java	Java_Potential	Potential_Command_Injection	77	**Source** has changed
Java	Java_Potential	Potential_Connection_String_Injection	99	**Source** has changed
Java	Java_Potential	Potential_GWT_Reflected_XSS	79	**Source** has changed
Java	Java_Potential	Potential_Hardcoded_password_in_Connection_String	547	**Source** has changed
Java	Java_Potential	Potential_IO_Reflected_XSS_All_Clients	79	**Source** has changed
Java	Java_Potential	Potential_I_Reflected_XSS_All_Clients	79	**Source** has changed
Java	Java_Potential	Potential_LDAP_Injection	90	**Source** has changed
Java	Java_Potential	Potential_O_Reflected_XSS_All_Clients	79	**Source** has changed
Java	Java_Potential	Potential_Parameter_Tampering	472	**Source** has changed
Java	Java_Potential	Potential_Resource_Injection	99	**Source** has changed
Java	Java_Potential	Potential_SQL_Injection	89	**Source** has changed
Java	Java_Potential	Potential_Stored_XSS	79	**Source** has changed
Java	Java_Potential	Potential_Use_of_Hard_coded_Cryptographic_Key	321	**Source** has changed
Java	Java_Potential	Potential_UTF7_XSS	79	**Source** has changed
Java	Java_Potential	Potential_XPath_Injection	643	**Source** has changed
Java	Java_Potential	Potential_XXE_Injection	776	**Source** has changed
Java	Java_Spring	Spring_Missing_Expect_CT_Header	693	**Source** has changed

Language	Group	Name	CWE	Changed Fields
Java	Java_Spring	Spring_Missing_HSTS_Header	346	**Source** has changed
Java	Java_Spring	Spring_Missing_XSS_Protection_Header	693	**Source** has changed
Java	Java_Spring	Spring_Missing_X_Content_Type_Options	693	**Source** has changed
Java	Java_Stored	Stored_HTTP_Response_Splitting	113	**Source** has changed
JavaScript	Javascript_Kony	Kony_Unsecure_iOSBrowser_Configuration	15	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Cross_Session_Contamination	488	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Heuristic_Poor_XSS_Validation	80	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_HTML5_Heuristic_Session_Insecure_Storage	922	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Insufficient_Key_Size	310	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Located_JQuery_Outdated_Lib_File	477	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Potential_Ad_Hoc_Ajax	693	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Potential_ReDoS_In_Match	400	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_Potential_ReDoS_In_Replace	400	**Source** has changed
JavaScript	JavaScript_Medium_Threat	Client_Cross_Frame_Scripting_Attack	79	**Source** has changed
JavaScript	JavaScript_Medium_Threat	Client_Header_Manipulation	113	**Source** has changed
JavaScript	JavaScript_Medium_Threat	Client_HTML5_Store_Sensitive_data_In_Web_Storage	312	**Source** has changed
JavaScript	JavaScript_Medium_Threat	Client_Reflected_File_Download	425	**Source** has changed
JavaScript	JavaScript_SAPUI5	SAPUI5_Hardcoded_UserId_In_Comments	200	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	CSRF	352	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	HTTP_Response_Splitting	113	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Insecure_Direct_Object_References	813	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	JSON_Hijacking	352	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Password_Weak_Encryption	261	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Potentially_Vulnerable_To_CSRF	352	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Reflected_XSS	79	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Security_Misconfiguration	933	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Use_Of_Hardcoded_Password	259	**Source** has changed
JavaScript	JavasScript_Visualforce_Remoting	VF_Remoting_Client_Potential_CSRF	352	**Source** has changed
Kotlin	Kotlin_Android	Accessible_Content_Provider	668	**Source** has changed
Kotlin	Kotlin_Android	Exported_Service_Without_Permissions	668	**Source** has changed
Kotlin	Kotlin_Android	Improper_Certificate_Validation	295	**Source** has changed
Kotlin	Kotlin_Low_Visibility	Use_of_Hardcoded_Password	259	**Source** has changed
Lua	Lua_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
Lua	Lua_Low_Visibility	Missing_Password_Field_Masking	549	Name changed from Missing_Password_Field_Masking_Node to **Missing_Password_Field_Masking**, **Source** has changed
Objc	ObjectiveC_Best_Coding_Practice	Dead_Code	561	**Source** has changed
Objc	ObjectiveC_High_Risk	Deserialization_of_Untrusted_Data	502	**Source** has changed
Objc	ObjectiveC_High_Risk	Universal_XSS	79	**Source** has changed
Objc	ObjectiveC_Low_Visibility	Heap_Inspection	244	**Source** has changed
Objc	ObjectiveC_Low_Visibility	Memory_Leak	401	**Source** has changed
Objc	ObjectiveC_Low_Visibility	Potential_ReDoS	400	**Source** has changed
Objc	ObjectiveC_Low_Visibility	Use_of_Hardcoded_Password	259	**Source** has changed
Objc	ObjectiveC_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
Objc	ObjectiveC_Medium_Threat	Parameter_Tampering	472	**Source** has changed
Objc	ObjectiveC_Medium_Threat	Side_Channel_Data_Leakage	200	**Source** has changed
PHP	PHP_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
PHP	PHP_Medium_Threat	Broken_or_Risky_Encryption_Algorithm	327	**Source** has changed

Language	Group	Name	CWE	Changed Fields
PHP	PHP_Medium_Threat	Broken_or_Risky_Hashing_Function	327	**Source** has changed
PHP	PHP_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
PHP	PHP_Medium_Threat	SSRF	918	**Source** has changed
PLSQL	PLSQL_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
Perl	Perl_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
Perl	Perl_Medium_Threat	Use_Of_Hardcoded_Password	259	**Source** has changed
Python	Python_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Python	Python_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
Python	Python_Medium_Threat	Path_Traversal	22	**Source** has changed
RPG	RPG_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Ruby	Ruby_Best_Coding_Practice	Declaration_Of_Catch_For_Generic_Exception	396	**Source** has changed
Ruby	Ruby_Best_Coding_Practice	Unclosed_Objects	459	**Source** has changed
Ruby	Ruby_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed
Ruby	Ruby_Low_Visibility	Improper_Transaction_Handling	460	**Source** has changed
Ruby	Ruby_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Ruby	Ruby_Low_Visibility	XSS_Evasion_Attack	79	**Source** has changed
Ruby	Ruby_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_JSON_GEM_Remote_Code	20	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_JSON_Remote_Code_Execution	94	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_Bypass_Access_Control	477	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_Cross_Site_Request_Forgery	352	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_DOS_via_ActiveRecord	400	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_SQL_Injection	89	**Source** has changed
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_XSS	79	**Source** has changed
Rust	Rust_High_Risk	Connection_String_Injection	99	**Source** has changed
Rust	Rust_Low_Visibility	JWT_Excessive_Expiration_Time	613	**Source** has changed
Rust	Rust_Low_Visibility	JWT_Lack_of_Expiration_Time	613	**Source** has changed
Rust	Rust_Medium_Threat	DoS_by_Sleep	834	**Source** has changed
Rust	Rust_Medium_Threat	Empty_Password_In_Connection_String	521	**Source** has changed
Rust	Rust_Medium_Threat	Encoding_Used_Instead_of_Encryption	311	**Source** has changed
Rust	Rust_Medium_Threat	Hardcoded_Password_in_Connection_String	547	**Source** has changed
Rust	Rust_Medium_Threat	JWT_Sensitive_Information_Exposure	201	**Source** has changed
Rust	Rust_Medium_Threat	JWT_Use_Of_Hardcoded_Secret	798	**Source** has changed
Scala	Scala_Low_Visibility	Potential_Stored_XSS	79	**Source** has changed
Scala	Scala_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
Scala	Scala_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
Scala	Scala_Medium_Threat	Multiple_Binds_to_the_Same_Port	605	**Source** has changed
Scala	Scala_Stored	Stored_HTTP_Response_Splitting	113	**Source** has changed
Swift	Swift_Low_Visibility	Heap_Inspection	244	**Source** has changed
Swift	Swift_Low_Visibility	Parameter_Tampering	472	**Source** has changed
VB6	VB6_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed
VB6	VB6_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
VbNet	VbNet_Best_Coding_Practice	Unclosed_Objects	459	**Source** has changed
VbNet	VbNet_Heuristic	Heuristic_2nd_Order_SQL_Injection	89	**Source** has changed
VbNet	VbNet_Heuristic	Heuristic_CSRF	352	**Source** has changed
VbNet	VbNet_Heuristic	Heuristic_DB_Parameter_Tampering	284	**Source** has changed
VbNet	VbNet_Heuristic	Heuristic_Parameter_Tampering	472	**Source** has changed

Language	Group	Name	CWE	Changed Fields
VbNet	VbNet_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
VbNet	VbNet_Heuristic	Heuristic_Stored_XSS	79	**Source** has changed
VbNet	VbNet_High_Risk	UTF7_XSS	79	**Source** has changed
VbNet	VbNet_Low_Visibility	Blind_SQL_Injections	89	**Source** has changed
VbNet	VbNet_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171	**Source** has changed
VbNet	VbNet_Low_Visibility	Improper_Transaction_Handling	460	**Source** has changed
VbNet	VbNet_Low_Visibility	JavaScript_Hijacking	352	**Source** has changed
VbNet	VbNet_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
VbNet	VbNet_Low_Visibility	XSS_Evasion_Attack	79	**Source** has changed
VbNet	VbNet_Medium_Threat	DB_Parameter_Tampering	284	**Source** has changed
VbNet	VbNet_Medium_Threat	HTTP_Response_Splitting	113	**Source** has changed
VbNet	VbNet_Medium_Threat	Path_Traversal	22	**Source** has changed
VbNet	VbNet_Medium_Threat	Reflected_XSS_Specific_Clients	79	**Source** has changed
VbNet	VbNet_WebConfig	HttpOnlyCookies_XSS	1004	Severity changed from High to **Medium**, **Source** has changed

Deprecated Queries:

Language	Group	Name	CWE
ASP	ASP_Heuristic	Heuristic_2nd_Order_SQL_Injection	89
ASP	ASP_Heuristic	Heuristic_CSRF	352
ASP	ASP_Heuristic	Heuristic_DB_Parameter_Tampering	284
ASP	ASP_Heuristic	Heuristic_Parameter_Tampering	472
ASP	ASP_Heuristic	Heuristic_SQL_Injection	89
ASP	ASP_Heuristic	Heuristic_Stored_XSS	79
ASP	ASP_High_Risk	UTF7_XSS	79
ASP	ASP_Low_Visibility	Blind_SQL_Injections	89
ASP	ASP_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171
ASP	ASP_Low_Visibility	JavaScript_Hijacking	352
ASP	ASP_Low_Visibility	XSS_Evasion_Attack	79
ASP	ASP_Medium_Threat	DB_Parameter_Tampering	284
ASP	ASP_Medium_Threat	Reflected_XSS_Specific_Clients	79
CPP	CPP_Buffer_Overflow	Buffer_Overflow_Loops_Old	193
CPP	CPP_Heuristic	Heuristic_2nd_Order_Buffer_Overflow_malloc	120
CPP	CPP_Heuristic	Heuristic_2nd_Order_Buffer_Overflow_read	120
CPP	CPP_Heuristic	Heuristic_2nd_Order_SQL_Injection	89
CPP	CPP_Heuristic	Heuristic_Buffer_Improper_Index_Access	129
CPP	CPP_Heuristic	Heuristic_Buffer_Overflow_malloc	120
CPP	CPP_Heuristic	Heuristic_Buffer_Overflow_read	120
CPP	CPP_Heuristic	Heuristic_CGI_Stored_XSS	79
CPP	CPP_Heuristic	Heuristic_DB_Parameter_Tampering	284
CPP	CPP_Heuristic	Heuristic_Parameter_Tampering	472
CPP	CPP_Heuristic	Heuristic_SQL_Injection	89
CPP	CPP_Heuristic	Heuristic_Unchecked_Return_Value	252
CPP	CPP_Low_Visibility	Blind_SQL_Injections	89
CPP	CPP_Low_Visibility	Stored_Blind_SQL_Injections	89
CPP	CPP_Medium_Threat	DB_Parameter_Tampering	284
CPP	CPP_Medium_Threat	Inadequate_Pointer_Validation	477

Language	Group	Name	CWE
CPP	CPP_Stored_Vulnerabilities	Stored_DB_Parameter_Tampering	284
CSharp	CSharp_Heuristic	Heuristic_2nd_Order_SQL_Injection	89
CSharp	CSharp_Heuristic	Heuristic_CSRF	352
CSharp	CSharp_Heuristic	Heuristic_DB_Parameter_Tampering	284
CSharp	CSharp_Heuristic	Heuristic_Parameter_Tampering	472
CSharp	CSharp_Heuristic	Heuristic_SQL_Injection	89
CSharp	CSharp_Heuristic	Heuristic_Stored_XSS	79
CSharp	CSharp_High_Risk	UTF7_XSS	79
CSharp	CSharp_Low_Visibility	Blind_SQL_Injections	89
CSharp	CSharp_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171
CSharp	CSharp_Low_Visibility	JavaScript_Hijacking	352
CSharp	CSharp_Low_Visibility	Potential_ReDoS	400
CSharp	CSharp_Low_Visibility	Potential_ReDoS_By_Injection	400
CSharp	CSharp_Low_Visibility	Potential_ReDoS_In_Code	400
CSharp	CSharp_Low_Visibility	Potential_ReDoS_In_Static_Field	400
CSharp	CSharp_Low_Visibility	XSS_Evasion_Attack	79
CSharp	CSharp_Medium_Threat	DB_Parameter_Tampering	284
CSharp	CSharp_Medium_Threat	Reflected_XSS_Specific_Clients	79
Groovy	Groovy_Heuristic	Heuristic_2nd_Order_SQL_Injection	89
Groovy	Groovy_Heuristic	Heuristic_CGI_Stored_XSS	79
Groovy	Groovy_Heuristic	Heuristic_CSRF	352
Groovy	Groovy_Heuristic	Heuristic_DB_Parameter_Tampering	284
Groovy	Groovy_Heuristic	Heuristic_Parameter_Tampering	472
Groovy	Groovy_Heuristic	Heuristic_SQL_Injection	89
Groovy	Groovy_Heuristic	Heuristic_Stored_XSS	79
Groovy	Groovy_High_Risk	UTF7_XSS	79
Groovy	Groovy_Low_Visibility	Blind_SQL_Injections	89
Groovy	Groovy_Low_Visibility	Channel_Accessible_by_NonEndpoint	300
Groovy	Groovy_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171
Groovy	Groovy_Low_Visibility	Plaintext_Storage_in_a_Cookie	315
Groovy	Groovy_Low_Visibility	Potenial_UTF7_XSS	79
Groovy	Groovy_Low_Visibility	Potential_ReDoS	400
Groovy	Groovy_Low_Visibility	Potential_ReDoS_By_Injection	400
Groovy	Groovy_Low_Visibility	Potential_ReDoS_In_Match	400
Groovy	Groovy_Low_Visibility	Potential_ReDoS_In_Replace	400
Groovy	Groovy_Low_Visibility	Potential_ReDoS_In_Static_Field	400
Groovy	Groovy_Low_Visibility	Reliance_on_Cookies_in_a_Decision	784
Groovy	Groovy_Low_Visibility	Use_of_Client_Side_Authentication	603
Groovy	Groovy_Low_Visibility	Use_Of_getenv	589
Groovy	Groovy_Medium_Threat	DB_Parameter_Tampering	284
Groovy	Groovy_Medium_Threat	HTTP_Response_Splitting	113
Groovy	Groovy_Medium_Threat	Multiple_Binds_to_the_Same_Port	605
Groovy	Groovy_Stored	Stored_HTTP_Response_Splitting	113
Java	Java_GWT	JSON_Hijacking	352
Java	Java_Low_Visibility	Blind_SQL_Injections	89
Java	Java_Low_Visibility	Channel_Accessible_by_NonEndpoint	300
Java	Java_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171

Language	Group	Name	CWE
Java	Java_Low_Visibility	Improper_Resource_Access_Authorization	285
Java	Java_Low_Visibility	Plaintext_Storage_in_a_Cookie	315
Java	Java_Low_Visibility	Potential_ReDoS	400
Java	Java_Low_Visibility	Potential_ReDoS_By_Injection	400
Java	Java_Low_Visibility	Potential_ReDoS_In_Match	400
Java	Java_Low_Visibility	Potential_ReDoS_In_Replace	400
Java	Java_Low_Visibility	Potential_ReDoS_In_Static_Field	400
Java	Java_Low_Visibility	Reliance_on_Cookies_in_a_Decision	784
Java	Java_Low_Visibility	Suspected_XSS	79
Java	Java_Low_Visibility	Use_of_Client_Side_Authentication	603
Java	Java_Low_Visibility	UTF7_XSS	79
Java	Java_Medium_Threat	DB_Parameter_Tampering	284
Java	Java_Medium_Threat	Multiple_Binds_to_the_Same_Port	605
Java	Java_Spring	Spring_Missing_Expect_CT_Header	693
Java	Java_Spring	Spring_Missing_XSS_Protection_Header	693
Java	Java_Spring	Spring_Missing_X_Content_Type_Options	693
Java	Java_Stored	Stored_HTTP_Response_Splitting	113
JavaScript	JavaScript_Low_Visibility	Client_Potential_Ad_Hoc_Ajax	693
JavaScript	JavaScript_Low_Visibility	Client_Potential_ReDoS_In_Match	400
JavaScript	JavaScript_Low_Visibility	Client_Potential_ReDoS_In_Replace	400
JavaScript	JavaScript_Medium_Threat	Client_Cross_Frame_Scripting_Attack	79
JavaScript	JavaScript_Server_Side_Vulnerabilities	JSON_Hijacking	352
JavaScript	JavaScript_Server_Side_Vulnerabilities	Missing_Encryption_of_Sensitive_Data	311
JavaScript	JavaScript_Server_Side_Vulnerabilities	Potentially_Vulnerable_To_CSRF	352
JavaScript	JavaScript_Server_Side_Vulnerabilities	Security_Misconfiguration	933
Lua	Lua_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311
Objc	ObjectiveC_High_Risk	Deserialization_of_Untrusted_Data	502
Objc	ObjectiveC_High_Risk	Universal_XSS	79
Objc	ObjectiveC_Low_Visibility	Heap_Inspection	244
Objc	ObjectiveC_Low_Visibility	Potential_ReDoS	400
Objc	ObjectiveC_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311
Objc	ObjectiveC_Medium_Threat	Parameter_Tampering	472
Objc	ObjectiveC_Medium_Threat	Side_Channel_Data_Leakage	200
Perl	Perl_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311
PHP	PHP_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311
PLSQL	PLSQL_Medium_Threat	HTTP_Response_Splitting	113
Python	Python_Medium_Threat	DB_Parameter_Tampering	284
Ruby	Ruby_Low_Visibility	Blind_SQL_Injections	89
Ruby	Ruby_Low_Visibility	XSS_Evasion_Attack	79
Ruby	Ruby_Medium_Threat	DB_Parameter_Tampering	284
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_JSON_GEM_Remote_Code	20
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_JSON_Remote_Code_Execution	94
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_Bypass_Access_Control	477
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_Cross_Site_Request_Forgery	352
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_DOS_via_ActiveRecord	400
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_SQL_Injection	89
Ruby	Ruby_Vulnerable_Outdated_Versions	Outdated_Rails_Allows_XSS	79

Language	Group	Name	CWE
Scala	Scala_Low_Visibility	Potential_Stored_XSS	79
Scala	Scala_Medium_Threat	DB_Parameter_Tampering	284
Scala	Scala_Medium_Threat	HTTP_Response_Splitting	113
Scala	Scala_Medium_Threat	Multiple_Binds_to_the_Same_Port	605
Scala	Scala_Stored	Stored_HTTP_Response_Splitting	113
Swift	Swift_Low_Visibility	Heap_Inspection	244
Swift	Swift_Low_Visibility	Parameter_Tampering	472
VB6	VB6_Heuristic	Heuristic_Parameter_Tampering	472
VB6	VB6_Heuristic	Heuristic_SQL_Injection	89
VbNet	VbNet_Heuristic	Heuristic_2nd_Order_SQL_Injection	89
VbNet	VbNet_Heuristic	Heuristic_CSRF	352
VbNet	VbNet_Heuristic	Heuristic_DB_Parameter_Tampering	284
VbNet	VbNet_Heuristic	Heuristic_Parameter_Tampering	472
VbNet	VbNet_Heuristic	Heuristic_SQL_Injection	89
VbNet	VbNet_Heuristic	Heuristic_Stored_XSS	79
VbNet	VbNet_High_Risk	UTF7_XSS	79
VbNet	VbNet_Low_Visibility	Blind_SQL_Injections	89
VbNet	VbNet_Low_Visibility	Cleansing_Canonicalization_and_Comparison_Errors	171
VbNet	VbNet_Low_Visibility	JavaScript_Hijacking	352
VbNet	VbNet_Low_Visibility	XSS_Evasion_Attack	79
VbNet	VbNet_Medium_Threat	DB_Parameter_Tampering	284
VbNet	VbNet_Medium_Threat	Reflected_XSS_Specific_Clients	79

Changed Source:

Apex / Apex_Force_com_Code_Quality / Async_Future_Method_Inside_Loops

Code changes

```
---
+++
@@ -1,9 +1,9 @@
 CxList apex = Find_Apex_Files();
 CxList future = apex.FindByCustomAttribute("future");
-future = future.GetAncOfType(typeof(MethodDecl));
+future = future.GetAncOfType<MethodDecl>();

 CxList methods = apex.FindAllReferences(future);
-CxList methodsDecl = methods.GetAncOfType(typeof(MethodDecl));
+CxList methodsDecl = methods.GetAncOfType<MethodDecl>();

 int numMeth = 0;
 for(int i = 0; i < 5 && methods.Count > numMeth; i++)
@@ -16,8 +16,8 @@
 futurePlus.Add(future);
 futurePlus.Add(methods);

-CxList iterations = futurePlus.GetAncOfType(typeof(IterationStmt));
-iterations.Add(futurePlus.GetAncOfType(typeof(ForEachStmt)));
```

```
+CxList iterations = futurePlus.GetAncOfType<IterationStmt>();
+iterations.Add(futurePlus.GetAncOfType<ForEachStmt>());

 CxList callsToFuture = futurePlus.GetByAncs(iterations);
```

Apex / Apex_Force_com_Code_Quality / Bulkify_Apex_Methods_Using_Collections_In_Methods

Code changes

```
---
+++
@@ -6,13 +6,13 @@
 CxList bareDB = All.NewCxList();
 bareDB.Add(db);

-CxList meth = All.FindAllReferences(db.GetAncOfType(typeof(MethodDecl)));
+CxList meth = All.FindAllReferences(db.GetAncOfType<MethodDecl>());

 int numMeth = 0;
 for(int i = 0; i < 5 && meth.Count > numMeth; i++)
 {
    numMeth = meth.Count;
-   meth.Add(All.FindAllReferences(meth.GetAncOfType(typeof(MethodDecl))));
+   meth.Add(All.FindAllReferences(meth.GetAncOfType<MethodDecl>()));
 }
 db.Add(meth);
```

Apex / Apex_Force_com_Code_Quality / Hardcoded_Messages

Code changes

```
---
+++
@@ -1,4 +1,4 @@
 CxList methods = Find_Methods();
 CxList methodAddError = methods.FindByShortName("addError", false);
-CxList errorMsg = All.GetParameters(methodAddError, 0).FindByType(typeof(StringLiteral));
+CxList errorMsg = All.GetParameters(methodAddError, 0).FindByType<StringLiteral>();
 result.Add(errorMsg);
```

Apex / Apex_Force_com_Code_Quality / Hardcoding_Ids

Code changes

```
---
+++
@@ -8,7 +8,7 @@
 CxList idInLSide = id.FindByAssignmentSide(CxList.AssignmentSide.Left);

 // Find id in an initialization operation
-CxList idInDecl = idInLSide.FindByType(typeof(Declarator));
+CxList idInDecl = idInLSide.FindByType<Declarator>();
```

```
 CxList emptyNull = All.NewCxList();
 emptyNull.Add(emptyString);
```

Apex / Apex_Force_com_Code_Quality / Hardcoding_Of_Trigger_New

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList triggers = Find_Triggers_Code();
 CxList triggerNew = triggers.FindByMemberAccess("trigger.New", false);
-CxList triggerIndexerRef = triggerNew.GetFathers().FindByType(typeof(IndexerRef));
+CxList triggerIndexerRef = triggerNew.GetFathers().FindByType<IndexerRef>();

 // Find all indexes of Trigger.New
 CxList triggersIndexes = triggerIndexerRef.CxSelectElements<IndexerRef>(x => x.Indices, 0);
```

Apex / Apex_Force_com_Code_Quality / Hardcoding_Of_Trigger_Old

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList triggers = Find_Triggers_Code();
 CxList triggerNew = triggers.FindByMemberAccess("trigger.Old", false);
-CxList triggerIndexerRef = triggerNew.GetFathers().FindByType(typeof(IndexerRef));
+CxList triggerIndexerRef = triggerNew.GetFathers().FindByType<IndexerRef>();

 // Find all indexes of Trigger.Old
 CxList triggersIndexes = triggerIndexerRef.CxSelectElements<IndexerRef>(x => x.Indices, 0);
```

Apex / Apex_Force_com_Code_Quality / Multiple_Trigger_On_same_sObject

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList triggersCode = Find_Triggers_Code();

-CxList triggers = triggersCode.FindByType(typeof(MethodDecl)).FindByRegex("trigger ").FindByRegex(" on ");
+CxList triggers = triggersCode.FindByType<MethodDecl>().FindByRegex("trigger ").FindByRegex(" on ");
 CxList attrs = triggersCode.FindByFathers(triggers);

 System.Collections.ArrayList triggersOn = new System.Collections.ArrayList();
```

Apex / Apex_Force_com_Code_Quality / Use_of_Hard_Coded_Cryptographic_Key

Code changes

```
---
+++
@@ -9,7 +9,7 @@
```

```
 paramInSecure.Add(All.GetParameters(encryptWithManagedIV, 1));

-CxList sanitizers = base.Find_Same_Value_Sanitizers_Exclude_Sinks_and_Sources(paramInSecure + strings);
+CxList sanitizers = base.Find_Same_Value_Sanitizers_Exclude_Sinks_and_Sources(All.NewCxList(paramInSecure,strings));
 // Remove from the sanitizers all string methods that alter the string in a predictable way.
 string[] stringMethods = new string[]
 {
@@ -36,7 +36,7 @@
    "Blob.valueOf",
    "EncodingUtil.*"
 };
-sanitizers -= Find_Methods().FindByMemberAccesses(stringMethods, false);
+sanitizers -= methods.FindByMemberAccesses(stringMethods, false);

 result = paramInSecure.InfluencedByAndNotSanitized(strings, sanitizers)
    .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Apex / Apex_Force_com_Critical_Security_Risk / Reflected_XSS

Code changes

```
---
+++
@@ -1,8 +1,11 @@
 CxList inputs = Find_Interactive_Inputs();
+
 CxList outputs = Find_Unsafe_Outputs();

 CxList testCode = Find_Test_Code();
+
+CxList sanitized = All.NewCxList();
+
+sanitized.Add(Find_XSS_Sanitize(), testCode, Find_DB());

 result = outputs.InfluencedByAndNotSanitized(inputs, sanitized)
```

Apex / Apex_Force_com_Serious_Security_Risk / Frame_Spoofing

Code changes

```
---
+++
@@ -1,4 +1,5 @@
 CxList vfPages = Find_VF_Pages();
+
 CxList testCode = Find_Test_Code();

 CxList inputs = Find_Interactive_Inputs() - Find_Url_Current_Page();
@@ -7,9 +8,8 @@
    .GetMembersOfTarget().FindByShortName("src");
```

```
 CxList sanitize = Find_NonLeft_Binary(iframe);
-sanitize.Add(Find_Id_Sanitizers());
-sanitize.Add(Find_Integers());
-sanitize.Add(testCode);
+
+sanitize.Add(Find_Id_Sanitizers(),Find_Integers(),testCode);


 result = iframe.InfluencedByAndNotSanitized(inputs, sanitize);
```

Apex / Apex_Force_com_Serious_Security_Risk / inputText_Ignoring_FLS

Code changes

```
---
+++
@@ -26,7 +26,7 @@
 CxList renderedMethod = methods.FindByShortName("rendered");


 CxList sanitized = (All.GetParameters(renderedMethod) * objectTypes)
-    .GetAncOfType((typeof(MethodInvokeExpr))).FindByShortName("rendered");
+    .GetAncOfType<MethodInvokeExpr>().FindByShortName("rendered");


 inputText -= All.GetTargetsWithMembers(sanitized);
```

Apex / Apex_Force_com_Serious_Security_Risk / Sharing

Code changes

```
---
+++
@@ -47,7 +47,7 @@
 {
     CxList inGlob = inGlobals.GetByAncs(glob);
     CxList allReferences = (apexNotForeach - inGlob).FindAllReferences(glob);
-    methodCalls.Add(allReferences.GetMembersOfTarget().FindByType(typeof(MethodInvokeExpr)));
+    methodCalls.Add(allReferences.GetMembersOfTarget().FindByType<MethodInvokeExpr>());
     methodCalls.Add(allReferences.GetByAncs(create));
 }


@@ -60,14 +60,14 @@


 // 2. classes called by VF pages
 CxList vfProblems = VFData.FindAllReferences(noSharing);
-CxList sourceVF = vfProblems.GetAncOfType(typeof(MethodDecl)).FindByShortName("__*");
+CxList sourceVF = vfProblems.GetAncOfType<MethodDecl>().FindByShortName("__*");
 vfProblems -= vfProblems.GetByAncs(sourceVF);


-CxList accessControl = vfProblems.GetAncOfType(typeof(ObjectCreateExpr));
-CxList actions = accessControl.GetAncOfType(typeof(MethodInvokeExpr));
+CxList accessControl = vfProblems.GetAncOfType<ObjectCreateExpr>();
```

```
+CxList actions = accessControl.GetAncOfType<MethodInvokeExpr>();
 accessControl -= accessControl.GetByAncs(actions);

-CxList webServices = apexData.FindByType(typeof(MethodDecl)).FindByFieldAttributes(Modifiers.WebService);
+CxList webServices = apexData.FindByType<MethodDecl>().FindByFieldAttributes(Modifiers.WebService);
 webServices = apexData.GetByAncs(webServices);
 CxList wsProblems = webServices.FindAllReferences(noSharing);
 accessControl.Add(wsProblems);
@@ -83,7 +83,7 @@

 // 3. classes that have "without sharing" but use DB.
 CxList classesWithoutSharing = db.GetByAncs(notWithSharing - alreadyExistingControllers)
-   .GetAncOfType(typeof(ClassDecl)) - Find_With_Sharing();
+   .GetAncOfType<ClassDecl>() - Find_With_Sharing();

 result.Add(classesWithoutSharing);
```

Apex / Apex_ISV_Quality_Rules / Empty_IfStmt

Code changes

```
---
+++
@@ -1,5 +1,3 @@
-CxList ifs = Find_Ifs();
-CxList trueBlock = ifs.FilterByDomProperty<IfStmt>(x => x.TrueStatements.Count == 0);
-result.Add(trueBlock);
+result = Find_Ifs().FilterByDomProperty<IfStmt>(x => x.TrueStatements.Count == 0);
 result -= Find_VF_Pages();
 result -= result.FindByFileName("*.object");
```

Apex / Apex_ISV_Quality_Rules / Find_Exposed_Test_Data

Code changes

```
---
+++
@@ -4,10 +4,10 @@
 //Find isTest(SeeAllData=true)
 CxList isTest = All.FindByCustomAttribute("istest");
 CxList isTestChildren = All.GetByAncs(isTest);
-CxList boolTest = isTestChildren.FindByType(typeof(BooleanLiteral)).FindByShortName("true");
+CxList boolTest = isTestChildren.FindByType<BooleanLiteral>().FindByShortName("true");
 boolTest = boolTest.GetFathers();
 CxList seeAllData = isTestChildren.FindByAssignmentSide(CxList.AssignmentSide.Left).FindByShortName("seealldata");
-result = seeAllData.FindByFathers(boolTest).GetAncOfType(typeof(CustomAttribute));
+result = seeAllData.FindByFathers(boolTest).GetAncOfType<CustomAttribute>();

 isTest -= result;
```

Apex / Apex_ISV_Quality_Rules / SOQL_Dynamic_null_in_Where

Code changes

```
---
+++
@@ -41,7 +41,7 @@
 foreach (CxList curSOQL in soqlStrings)
 {
    whereParameters = curSOQL.ExtractFromSOQL("where");
-   CxList curSOQLParam = curSOQL.GetAncOfType(typeof(Param));
+   CxList curSOQLParam = curSOQL.GetAncOfType<Param>();
    CxList referencesInCurSOQL = unknwnRefs.GetByAncs(curSOQLParam);
    foreach(string inWhere in whereParameters)
    {
```

Apex / Apex_ISV_Quality_Rules / SOSL_With_Where_Clause

Code changes

```
---
+++
@@ -1,12 +1,17 @@
 //WHERE clause must not exist in SOSL
 //We heuristically find SOSL statements that have the "WHERE" keyword following a "RETURNING" keyword.
 CxList methods = Find_Methods();
+
 CxList slStatements = methods.FindByMemberAccess("Cx_VirtualDal.select");
+
 CxList whereSOSL = All.NewCxList();
-whereSOSL.Add(slStatements,
-   Find_Strings());
+
+whereSOSL.Add(slStatements,    Find_Strings());
+
 whereSOSL = whereSOSL.FindByRegex(@"RETURNING\s[^;]*\WWHERE\s", false, true, false);
+
 CxList searchQuery = methods.FindByMemberAccess("search.query");


 result = searchQuery.DataInfluencedBy(whereSOSL);
+
 result.Add(slStatements * whereSOSL);
```

Apex / Apex_ISV_Quality_Rules / Warn_About_Viewstate_Size_Limit

Code changes

```
---
+++
@@ -1,16 +1,16 @@
 //Warn if any non-transient variable is found in a controller or extension
 CxList apexFiles = Find_Apex_Files();
-CxList nonTransientFields = apexFiles.FindByType(typeof(FieldDecl));
-nonTransientFields.Add(apexFiles.FindByType(typeof(PropertyDecl)));
```

```
+CxList nonTransientFields = apexFiles.FindByType<FieldDecl>();
+nonTransientFields.Add(apexFiles.FindByType<PropertyDecl>());

 CxList attTransientSttaic = nonTransientFields.FindByFieldAttributes(Modifiers.Transient);
 attTransientSttaic.Add(nonTransientFields.FindByFieldAttributes(Modifiers.Static));

 nonTransientFields -= attTransientSttaic;

-CxList nonTransientClasses = nonTransientFields.GetAncOfType(typeof(ClassDecl));
+CxList nonTransientClasses = nonTransientFields.GetAncOfType<ClassDecl>();

-CxList extensionNames = Find_VF_Pages().FindByType(typeof(TypeRef));
+CxList extensionNames = Find_VF_Pages().FindByType<TypeRef>();
 nonTransientClasses = nonTransientClasses.FindByShortName(extensionNames);
 foreach (CxList cls in nonTransientClasses)
 {  //Find the pages that have cls as a controller or extension
```

Apex / Apex_Low_Visibility / Escape_False_Warning

Code changes

```
---
+++
@@ -1,12 +1,14 @@
+CxList unknownReferences = Find_UnknownReference();
+
 CxList methodsOutput = All.FindByMemberAccess("apex.output*", false).GetMembersOfTarget().FindByShortName("value");
-CxList escapeFalseOutput = Find_UnknownReference().GetParameters(methodsOutput)
-    .FindByShortName("escape_false");

-result = escapeFalseOutput;
+CxList escapeFalseOutput = unknownReferences.GetParameters(methodsOutput).FindByShortName("escape_false");
+
+result.Add(escapeFalseOutput);

 CxList methodsInput = All.FindByMemberAccess("apex.input*", false).GetMembersOfTarget().FindByShortName("value");
-CxList escapeFalseInput = Find_UnknownReference().GetParameters(methodsInput)
-    .FindByShortName("escape_false");
+
+CxList escapeFalseInput = unknownReferences.GetParameters(methodsInput).FindByShortName("escape_false");

 result.Add(escapeFalseInput);
```

Apex / Apex_Low_Visibility / Hardcoded_Password

Code changes

```
---
+++
@@ -5,11 +5,10 @@
```

```
 // Find password in an initialization operation
 CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList psw_in_lSide_decl = psw_in_lSide.FindByType(typeof(Declarator));

-CxList emptyStrNull = All.NewCxList();
-emptyStrNull.Add(emptyString);
-emptyStrNull.Add(NULL);
+CxList psw_in_lSide_decl = psw_in_lSide.FindByType<Declarator>();
+
+CxList emptyStrNull = All.NewCxList(emptyString,NULL);

 CxList strLiterals = Find_Strings() - emptyStrNull;

@@ -26,7 +25,7 @@
 eq *= strLiterals.GetMembersOfTarget();
 equalsPassword.Add(psw.GetByAncs(eq));

-CxList assignPassword = psw_in_lSide.GetAncOfType(typeof(AssignExpr));
+CxList assignPassword = psw_in_lSide.GetAncOfType<AssignExpr>();
 assignPassword = lit_in_rSide.GetByAncs(assignPassword);

 result = initializedPassword;
@@ -37,8 +36,8 @@
 CxList setheader = methods.FindByMemberAccess("httprequest.setheader");
 CxList param1 = All.GetParameters(setheader, 0).FindByShortName("authorization");
 CxList secondParam = All.GetParameters(All.FindByParameters(param1));
-CxList paramAffectedByPass = secondParam.DataInfluencedBy(Find_Passwords())
-       .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
+CxList paramAffectedByPass = secondParam.DataInfluencedBy(psw)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);

 result.Add(paramAffectedByPass);
```

Apex / Apex_Low_Visibility / Parameter_Tampering

Code changes

```
---
+++
@@ -3,6 +3,10 @@
 CxList methods = Find_Methods();
 CxList testCode = Find_Test_Code();

+CxList withSharingMethods = methods.GetByAncs(
+   Find_ClassDecl().GetClass(Find_CustomAttribute().FindByCustomAttribute("with sharing")));
+
+CxList sanitizers = All.NewCxList(testCode, withSharingMethods);
 List<string> dbMethodsNames = new List<string> {"insert","insertAsync",
        "update","updateAsync",
```

```
        "delete","deleteAsync",
@@ -23,8 +27,7 @@
 db1 -= db1.DataInfluencedBy(And);
 db = db1 + (Select * Where - And).GetByAncs(All.GetParameters(db));

-result = db.InfluencedByAndNotSanitized(input, testCode);
+result = db.InfluencedByAndNotSanitized(input, sanitizers);

 result -= testCode;
-
 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Apex / Apex_Low_Visibility / Password_misuse

Code changes

```
---
+++
@@ -1,12 +1,15 @@
 CxList psw  = Find_Passwords();
+
 psw -= Find_Methods();

 CxList DB = Find_DB_Input();
+
 CxList testCode = Find_Test_Code();
 result = DB.InfluencingOnAndNotSanitized(psw, testCode);

 result -= testCode;
+
 result -= result.DataInfluencedBy(result);

 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Apex / Apex_Low_Visibility / Potential_Frame_Injection

Code changes

```
---
+++
@@ -6,6 +6,6 @@
 CxList nameAndId = All.NewCxList();
 nameAndId.Add(methods.FindByName("*iframe.id"), methods.FindByName("*iframe.name"));

-result = vfPages.GetParameters(nameAndId).FindByType(typeof(Expression));
+result = vfPages.GetParameters(nameAndId).FindByType<Expression>();

 result -= Find_Test_Code();
```

Apex / Apex_Low_Visibility / Potential_URL_Redirection_Attack

Code changes

```
---
+++
@@ -10,23 +10,21 @@
 referencePath.Add(reference);

 referencePath = All.GetByAncs(referencePath);
-referencePath.Add(referencePath.GetAncOfType(typeof(AssignExpr)));
+referencePath.Add(referencePath.GetAncOfType<AssignExpr>());

 CxList inputs = All * Find_Url_Current_Page() * Find_Interactive_Inputs();

+CxList nonLeftBinary = Find_NonLeft_Binary();

-CxList sanitize = Find_NonLeft_Binary(referencePath);
-sanitize.Add(Find_Integers());
-sanitize.Add(Find_Id_Sanitizers());
-sanitize.Add(testCode);
+CxList sanitize = All.NewCxList(nonLeftBinary, Find_Integers(), Find_Id_Sanitizers(), testCode);

 reference -= reference.GetByAncs(Find_Boolean_Conditions());

 CxList refInputs = inputs * reference - sanitize;
 sanitize -= reference;

-CxList refsBinaries = Find_NonLeft_Binary(reference);
-refsBinaries.Add(reference.FindByType(typeof(BinaryExpr)));
+CxList refsBinaries = All.NewCxList(nonLeftBinary);
+refsBinaries.Add(reference.FindByType<BinaryExpr>());

 reference -= refsBinaries;
```

Apex / Apex_Low_Visibility / Privacy_Violation

Code changes

```
---
+++
@@ -26,7 +26,7 @@
 personalInfo -= upperCase;

 // 2) exclude constants that are assigned a literal
-CxList constants = personalInfo.FindByType(typeof(ConstantDecl));
+CxList constants = personalInfo.FindByType<ConstantDecl>();
 CxList allConstRef = personalInfo.FindAllReferences(constants);



@@ -39,13 +39,13 @@
 intStringLiteral.Add(strings);
 intStringLiteral.Add(integerLiteral);
```

```
-CxList ConstAssignedL = intStringLiteral.FindByFathers(allConstRef.FindByType(typeof(Declarator)));
+CxList ConstAssignedL = intStringLiteral.FindByFathers(allConstRef.FindByType<Declarator>());

 // remove assignments of constants to string or integer literals
 allConstRef -= personalInfo.FindAllReferences(ConstAssignedL.GetFathers());

 // remove assignments of constants to null literals
-CxList declWithNull = allConstRef * nullLiteral.GetFathers().FindByType(typeof(Declarator));
+CxList declWithNull = allConstRef * nullLiteral.GetFathers().FindByType<Declarator>();
 // constants are assigned null value by default if the real assignment is not in the declaration line, and so the implicit assignment to null is irrelevant.
 // eg. final x; is parsed as final x=null; although x can be assigned later
 CxList allReferences = allConstRef.FindAllReferences(declWithNull);
@@ -110,7 +110,7 @@
 // find declarators of constants and variables so they can be removed - declarators are not a part of the flow from input to output
 // eg. string x = ___ is parsed as: (Declarator) string (UnknownReference) x (AssignExpr) = (value / expression / literal)___
 // the real flow is from the UnknownReference and not the Declarator
-CxList declarator = personalInfo.FindByType(typeof(Declarator));
+CxList declarator = personalInfo.FindByType<Declarator>();

 // remove the declaration from the references of the variables and constants
 variableRef -= declarator;
```

Apex / Apex_Low_Visibility / Second_Order_SOQL_SOSL_Injection

Code changes

```
---
+++
@@ -1,8 +1,11 @@
 CxList db = Find_DB_Output();
+
 CxList inputs = Find_DB_Input();
+
 CxList testCode = Find_Test_Code();

 CxList sanitized = Sanitize();
+
 sanitized.Add(testCode);

 // The inputs removal from sanitizers, remove outputs sanitizers too, so first we remove sanitized outputs
@@ -10,6 +13,7 @@

 // Remove the inputs (db_out) from the sanitization, otherwise there are not results at all
 CxList sanitizedInputs = sanitized * inputs;
+
 sanitized = sanitized - sanitizedInputs;

 result = inputs.InfluencingOnAndNotSanitized(db, sanitized).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Apex / Apex_Low_Visibility / Verbose_Error_Reporting

Code changes

```
---
+++
@@ -1,8 +1,11 @@
 CxList ctch = Find_Catch();
+
 CxList classOfCatch = All.GetClass(ctch);

 ctch = ctch.GetByAncs(classOfCatch);
+
 CxList outputs = Find_Interactive_Outputs();
+
 CxList exc = All.FindAllReferences(ctch) - ctch;

 CxList testCode = Find_Test_Code();
```

ASP / ASP_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,12 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList db = Find_SQL_DB_In();
-   CxList sanitize = Find_SQL_Sanitize();
-   result = All.FindSQLInjections(Find_Read() + possible_db, possible_db, sanitize);
-   if (result.Count > 0)
-   {
-       result = All.FindSQLInjections(Find_Read() + Find_DB_Out(), db, sanitize);
-   }
-}
+//This query is deprecated.
```

ASP / ASP_Heuristic / Heuristic_CSRF

Code changes

```
---
+++
@@ -1,17 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList requests = Find_Interactive_Inputs();
-   CxList strings = Find_Strings();
-   CxList write = strings.FindByNames(new string[] {"*update*", "*delete*", "*insert*"}
```

```
-       , StringComparison.OrdinalIgnoreCase);
-
-   result = possible_db.DataInfluencedBy(write).DataInfluencedBy(requests);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB();
-       result -= db.DataInfluencedBy(write).DataInfluencedBy(requests);
-   }
-}
+//This query is deprecated.
```

ASP / ASP_Heuristic / Heuristic_DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,32 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList tables = All.FindByName("*orders*", false) +
-       All.FindByName("*credit*", false) +
-       All.FindByName("*invoice*", false) +
-       All.FindByName("*booking*", false) +
-       All.FindByName("*bill*", false) +
-       All.FindByName("*payment*", false) +
-       All.FindByName("*account*", false) +
-       All.FindByName("*cash*", false) +
-       All.FindByName("*customer*", false);
-
-   CxList inputs = Find_Interactive_Inputs();
-
-   CxList user = All.FindByName("*user*", false) +
-       All.FindByName("*cust*", false) +
-       All.FindByName("*member*", false);
-
-   possible_db = possible_db.DataInfluencedBy(tables);
-   possible_db -= possible_db.DataInfluencedBy(user);
-   result = inputs.DataInfluencingOn(possible_db);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB();
-       db = db.DataInfluencedBy(tables);
-       db -= db.DataInfluencedBy(user);
-       result -= inputs.DataInfluencingOn(db);
-   }
```

```
-}
+//This query is deprecated.
```

ASP / ASP_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---
+++
@@ -1,25 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList input = Find_Interactive_Inputs();
-
-   CxList strings = Find_Strings();
-   CxList Select = strings.FindByName("*select*", false);
-   CxList Where = strings.FindByName("*where*", false);
-   CxList And = strings.FindByName("*and *", false) +
-       strings.FindByName("* and*", false);
-
-   possible_db = possible_db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-   possible_db -= possible_db.DataInfluencedBy(And);
-
-   result = possible_db.DataInfluencedBy(input);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB();
-       db = db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-       db -= db.DataInfluencedBy(And);
-       result -= db.DataInfluencedBy(input);
-   }
-}
+//This query is deprecated.
```

ASP / ASP_Heuristic / Heuristic_SQL_Injection

Code changes

```
---
+++
@@ -1,15 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList inputs = Find_Interactive_Inputs();
-   CxList sanitized = Find_SQL_Sanitize();
-
-   result = inputs.InfluencingOnAndNotSanitized(possible_db, sanitized);
```

```
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_SQL_DB_In();
-       result -= inputs.InfluencingOnAndNotSanitized(db, sanitized);
-   }
-}
+//This query is deprecated.
```

ASP / ASP_Heuristic / Heuristic_Stored_XSS

Code changes

```
---
+++
@@ -1,15 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList outputs = Find_XSS_Outputs();
-   CxList sanitize = Find_XSS_Sanitize() - Find_LDAP();
-
-   result = All.FindXSS(possible_db + Find_IO(), outputs, sanitize);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB_Out();
-       result -= All.FindXSS(db + Find_IO(), outputs, sanitize);
-   }
-}
+//This query is deprecated.
```

ASP / ASP_High_Risk / UTF7_XSS

Code changes

```
---
+++
@@ -1,9 +1 @@
-CxList UTF7 = Find_Strings().FindByName("utf-7");
-CxList response = All.FindByName("response.charset");
-
-UTF7 = response.DataInfluencedBy(UTF7);
-
-if(UTF7.Count > 0)
-{
-   result = Find_XSS_Outputs().DataInfluencedBy(Find_Interactive_Inputs());
-}
+//This query is deprecated.
```

ASP / ASP_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList db = Find_SQL_DB_In();
-CxList db_not_in_try = Improper_Exception_Handling(db);
-CxList db_in_try = db - db_not_in_try;
-
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitized = Find_SQL_Sanitize();
-
-result = All.FindSQLInjections(inputs, db_in_try, sanitized);
+//This query is deprecated.
```

ASP / ASP_Low_Visibility / Cleansing_Canonicalization_and_Comparison_Errors

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList inputs = Find_Interactive_Inputs();
-CxList obj = All.FindByType(typeof(UnknownReference)) + All.FindByType(typeof(Declarator));
-CxList files =     obj.FindByType("*filestream") +
-               obj.FindByType("*fileinfo") +
-               All.FindByName("*.file.*");
-
-CxList sanitize = All.FindByName("*server.mappath") + All.FindByName("*request.mappath");
-result = files.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

ASP / ASP_Low_Visibility / Improper_Transaction_Handling

Code changes

```
---
+++
@@ -1,25 +1,23 @@
 CxList Commit = All.FindByName("*.commit");
 CxList Rollback = All.FindByName("*.rollback");


-CxList TryBlock = Commit.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList TryBlock = Commit.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList cml in TryBlock)
 {
     TryCatchFinallyStmt TryGraph = cml.TryGetCSharpGraph<TryCatchFinallyStmt>();
-
+
     CxList curTry = All.FindById(TryGraph.Try.NodeId);


     CxList curCatch = All.NewCxList();
```

```
    if(TryGraph.CatchClauses != null && TryGraph.CatchClauses.Count > 0)
-   {
        curCatch = All.FindById(TryGraph.CatchClauses[0].NodeId);
-   }

    CxList CommitInTry = Commit.GetByAncs(curTry);
    CxList RollbackInCatch = Rollback.GetByAncs(curCatch);



-   if( (RollbackInCatch.GetAncOfType(typeof(TryCatchFinallyStmt)) *
-       CommitInTry.GetAncOfType(typeof(TryCatchFinallyStmt))).Count == 0)
+   if( (RollbackInCatch.GetAncOfType<TryCatchFinallyStmt>() *
+       CommitInTry.GetAncOfType<TryCatchFinallyStmt>()).Count == 0)
    {
        result.Add(cml);
    }
```

ASP / ASP_Low_Visibility / JavaScript_Hijacking

Code changes

```
---
+++
@@ -1,12 +1 @@
-//DWR framework prevents javascript hijacking
-CxList dwrFramework= All.FindByName("*dwr.util*",true);
-
-CxList CleanAJAXFramework=dwrFramework;// we'll add other frameworks that take care of javascript hijacking to this lis
-
-if (CleanAJAXFramework.Count==0)
-{
-   CxList db=Find_DB().DataInfluencedBy(All.FindByName("*select*",false)+All.FindByName("*exec*",false));
-   CxList jason=All.FindByName("*json*",false);
-   jason=jason.DataInfluencedBy(db);
-   result= jason;
-}
+//This query is deprecated.
```

ASP / ASP_Low_Visibility / XSS_Evasion_Attack

Code changes

```
---
+++
@@ -1,5 +1 @@
-CxList decode = All.FindByName("*server.htmldecode");
-CxList sanitize = Find_XSS_Sanitize();
-CxList output = Find_Interactive_Outputs();
-
-result = output.InfluencedByAndNotSanitized(decode, sanitize);
+//This query is deprecated.
```

ASP / ASP_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,20 +1 @@
-CxList tables = All.FindByShortName("*orders*", false);
-tables.Add(All.FindByShortName("*credit*", false));
-tables.Add(All.FindByShortName("*invoice*", false));
-tables.Add(All.FindByShortName("*booking*", false));
-tables.Add(All.FindByShortName("*bill*", false));
-tables.Add(All.FindByShortName("*payment*", false));
-tables.Add(All.FindByShortName("*account*", false));
-tables.Add(All.FindByShortName("*cash*", false));
-tables.Add(All.FindByShortName("*customer*", false));
-
-CxList inputs = Find_Interactive_Inputs();
-CxList db = Find_DB();
-
-CxList user = All.FindByShortName("*user*", false);
-user.Add(All.FindByShortName("*cust*", false));
-user.Add(All.FindByShortName("*member*", false));
-
-db = db.DataInfluencedBy(tables);
-db = db - db.DataInfluencedBy(user);
-result = inputs.DataInfluencingOn(db);
+//This query is deprecated.
```

ASP / ASP_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query HTTP_Response_Splitting is deprecated");
```

ASP / ASP_Medium_Threat / Reflected_XSS_Specific_Clients

Code changes

```
---
+++
@@ -1,6 +1 @@
-CxList inputs = Find_Interactive_Inputs();
-CxList outputs = Find_Interactive_Outputs()-Find_XSS_Outputs();
-
-CxList sanitized = Find_XSS_Sanitize();
-
-result = All.FindXSS(inputs, outputs, sanitized);
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Buffer_Improper_Index_Access

Code changes

```
---
+++
@@ -27,7 +27,7 @@


 // Remove nodes where length is checked in if statement
 CxList sizeComp = ints.GetAncOfType<BinaryExpr>() * strlenCalls.GetAncOfType<BinaryExpr>();
-CxList sizeCompInIf  = sizeComp.GetByAncs(ifStmts);
+CxList sizeCompInIf = sizeComp.GetByAncs(ifStmts);


 foreach(CxList sizeComparison in sizeCompInIf)
 {
@@ -121,8 +121,28 @@
 indices = indices.InfluencedByAndNotSanitized(inputs, idxSanitizers).GetLastNodesInPath();
 // paths from inputs to conditions
 CxList inputToCondition = inputs.DataInfluencingOn(conditions);
+
 // those referencess that are used in a condition
 CxList unkRefIndices = unkRefs.GetByAncs(inputToCondition.GetLastNodesInPath());
+
 // remove those indices that were used in a condition
-indices -= indices.FindAllReferences(allDeclarators.FindDefinition(unkRefIndices));
+indices -= indices.GetByAncs(inputToCondition.GetLastNodesInPath().GetFathers());
+
+foreach (CxList indice in indices) {
+   CxList currIfVar = unkRefs.FindAllReferences(indice) * unkRefIndices;
+
+   if (currIfVar.Count > 0) {
+       CSharpGraph ifVarGraph = currIfVar.GetFirstGraph();
+       CSharpGraph divVarGraph = indice.GetFirstGraph();
+
+       if (ifVarGraph.ShortName == divVarGraph.ShortName &&
+           ifVarGraph != divVarGraph &&
+           ifVarGraph.LinePragma.FileName == divVarGraph.LinePragma.FileName &&
+           ifVarGraph.LinePragma.Line < divVarGraph.LinePragma.Line)
+       {
+           indices -= indice;
+       }
+   }
+}
+
 result.Add(indices);
```

CPP / CPP_Buffer_Overflow / Buffer_Overflow_boundcpy_WrongSizeParam

Code changes

```
---
+++
```

```
@@ -1 +1 @@
-//This query is deprecated. Please consider Buffer_Overflow_Wrong_Buffer_Size instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Buffer_Overflow_LongString

Code changes

```
---
+++
@@ -9,6 +9,8 @@
 CxList integers = Find_Integer_Literals();
 CxList ur = Find_UnknownReference();
 CxList declarators = Find_Declarators();
+CxList methods = Find_Methods();
+CxList parameters = Find_Parameters().CxSelectDomProperty<Param>(x => x.Value);
 CxList arraryInitializer = Find_ArrayInitializer();
 CxList charDecls = declarators.FindByType("char");
 CxList variableDeclarations = declarators.GetAncOfType<VariableDeclStmt>();
@@ -21,9 +23,19 @@
 CxList sanitizers = Find_Sanitize();

 // remove results from methods that return pointers not related to their parameters
-sanitizers.Add(stringLiteral.GetParameters(Find_Methods()
+sanitizers.Add(stringLiteral.GetParameters(methods
     .FindByShortNames("fopen", "fdopen", "popen", "getenv")).GetAncOfType<MethodInvokeExpr>());

+// Sanitize char* in snprintf methods, where the second paremeters matches the destination var.
+CxList snprintfCalls = parameters.FindByShortName("sizeof").GetFathers().GetAncOfType<MethodInvokeExpr>()
+                                                                .FindByShortName("snprintf");
+foreach (CxList snprintf in snprintfCalls)
+{
+   CxList destVar = parameters.GetParameters(snprintf, 0);
+   CxList sizeofFunc = parameters.GetParameters(snprintf, 1).FindByShortName("sizeof");
+   CxList sizeVar = parameters.GetParameters(sizeofFunc, 0);
+   sanitizers.Add(destVar.FindAllReferences(sizeVar));
+}

 CxList relevant = varsOfTypeCharPointer.FindByFathers(
                   All.FindByAssignmentSide(CxList.AssignmentSide.Left));
@@ -47,7 +59,7 @@
     CxList stringInFault = elementInFlow.GetFirstNodesInPath();
     int stringSize = stringInFault.GetName().Length;

-    CxList arrayCreation = arrayCreateExpr.GetByAncs(declOfSink);
+    CxList arrayCreation = arrayCreateExpr.GetByAncs(declOfSink);
     foreach(CxList arr in arrayCreation)
     {
         try{
```

CPP / CPP_Buffer_Overflow / Buffer_Overflow_Loops

Code changes

```
---

+++

@@ -1 +1 @@

-//This query is deprecated. Please consider Buffer_Improper_Index_Access query instead.

+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Buffer_Overflow_Loops_Old

Code changes

```
---

+++

@@ -1,90 +1 @@

-// Buffer_Overflow - loops

-//  -----------------

-//  Find all the loops that have "<=" instead of "<", thus may create

-//  Buffer Overflow problem.

-//////////////////////////////////////////////////////////////////////

-

-// Find all iterations/loops

-CxList iterations = Find_IterationStmt();

-// Keep all iterations members (for performance, not to use All anymore)

-CxList allIterationsMembers = All.GetByAncs(iterations);

-

-CxList arrayCreate = Find_ArrayCreateExpr();

-CxList arraySize = Find_Integer_Literals().GetByAncs(arrayCreate);

-

-// Run on every loop and look for the vulnerability

-foreach (CxList iteration in iterations)

-{

-   // Check if we are dealing with a potential problem ("<=")

-   IterationStmt iter = iteration.TryGetCSharpGraph<IterationStmt>();

-

-   if ((iter.Test != null) && (iter.Test.ShortName.Equals("<=")))

-   {

-       // Keep all the members of the current iteration (performance)

-       CxList iterationMembers = allIterationsMembers.GetByAncs(iteration);

-

-       // Find all initializing expression of the iteration

-       CxList initExpr = All.NewCxList();

-       foreach (Statement stmt in iter.Init)

-       {

-           initExpr.Add(iterationMembers.FindById(stmt.NodeId));

-       }

-

-       CxList condition = All.FindById(iter.Test.NodeId);

-       CxList indexUses = iterationMembers.FindAllReferences(iterationMembers.GetByAncs(initExpr).FindByAssignmentSide(CxList.AssignmentSide.Left));

-       CxList arrays = iterationMembers.FindByType(typeof(IndexerRef));
```

```
-       CxList sizes = arraySize.GetByAncs(All.FindDefinition(arrays));
-       CxList conditionParts = All.GetByAncs(condition) - condition - indexUses;
-
-       // Sanitize False Positive where Loop Condition < Array Size
-       bool initLikeCondition = true;
-       foreach (CxList size in sizes)
-       {
-           IntegerLiteral intSize = size.TryGetCSharpGraph<IntegerLiteral>();
-           long intArraySize = intSize.Value;
-
-           long intLoopCond = 0;
-           if (conditionParts.FindByType(typeof(IntegerLiteral)).Count >= 1)
-           {
-               IntegerLiteral intCond = conditionParts.TryGetCSharpGraph<IntegerLiteral>();
-               if (intCond != null)
-               {
-                   intLoopCond = intCond.Value;
-               }
-           }
-
-           if (intLoopCond < intArraySize)
-           {
-               initLikeCondition = false;
-           }
-       }
-
-       indexUses = indexUses.FindByFathers(arrays);
-       // Find the values of initialization
-       CxList val = iterationMembers.GetByAncs(initExpr).FindByAssignmentSide(CxList.AssignmentSide.Right);
-       // If exists a value 0, then it's probably a problematic loop.
-       if (initLikeCondition && (val.FindByShortName("0").Count > 0) && (indexUses.Count > 0))
-       {
-           CxList iterationValue = iterationMembers.FindById(iter.Test.NodeId);
-           CxList sizeMethod = conditionParts.FindByType(typeof(MethodInvokeExpr)).FindByShortName("size");
-           if ((sizeMethod - iterationMembers.FindById(iter.Test.NodeId)).Count > 0)
-           {
-               CxList binary = conditionParts.FindByType(typeof(BinaryExpr));
-               if (iterationMembers.GetByAncs(binary).Count > 0)
-               {
-                   iterationValue -= iterationValue;
-               }
-           }
-           result.Add(iterationValue);
-       }
-
-       // Sanitize if minus in interation test
-       CxList TestNode = iterationMembers.FindById(iter.Test.NodeId);
-       BinaryExpr b = conditionParts.TryGetCSharpGraph<BinaryExpr>();
```

```
-       if (b != null && b.Operator == BinaryOperator.Subtract)
-       {
-           result -= TestNode;
-       }
-
-   }
-}
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Buffer_Overflow_Unbounded_Format

Code changes

```
---
+++
@@ -166,8 +166,11 @@
                        // There is flow from input to the source param (sscanf and sprintf only)
                        if (inputToSourceParam.Count > 0)
                        {
-                           CxList fromInputBufferOverflow = inputToSourceParam.ConcatenatePath(destinationParam, false);
-                           result.Add(fromInputBufferOverflow);
+                           // Cases in which there is more than one flow
+                           if (inputToSourceParam.Count > 1)
+                               result.Add(inputToSourceParam.GetCxListByPath().First().ConcatenatePath(destinationParam, false));
+                           else
+                               result.Add(inputToSourceParam.ConcatenatePath(destinationParam, false));
                        }


                    }
```

CPP / CPP_Buffer_Overflow / Missing_Precision

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please consider Buffer_Overflow_Unbounded_Format instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Off_by_One_Error_in_Arrays

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please refer to Off_By_One_Error instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Off_by_One_Error_in_Loops

Code changes

```
---
+++
```

```
@@ -1 +1 @@
-//This query is deprecated. Please refer to Off_By_One_Error instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Off_by_One_Error_in_Methods

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please refer to Off_By_One_Error instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / Potential_Precision_Problem

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please consider Buffer_Overflow_Unbounded_Format instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / String_Termination_cin

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please refer to Improper_Null_Termination instead.
+//This query is deprecated.
```

CPP / CPP_Buffer_Overflow / String_Termination_Error

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please refer to Improper_Null_Termination instead.
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_2nd_Order_Buffer_Overflow_malloc

Code changes

```
---
+++
@@ -1,15 +1 @@
-// The correct sanitization should be added
-
-CxList malloc = Find_Methods().FindByShortName("malloc");
-
-// Find input influence on size of copy
-CxList sizeParam = All.GetByAncs(malloc) - malloc;
-CxList db = All.NewCxList();
```

```
-db.Add(Find_DB(), Find_Read());
-CxList sanitize = Find_All_Strlen();
-
-//get strlen parameters that are "outputs" and remove them
-CxList outputStrlenParams = sizeParam.GetParameters(sanitize);
-sizeParam -= outputStrlenParams;
-
-result = sizeParam.InfluencedByAndNotSanitized(db, sanitize);
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_2nd_Order_Buffer_Overflow_read

Code changes

```
---
+++
@@ -1,13 +1 @@
-// The correct sanitization shoud be added
-
-CxList read = Find_Methods().FindByShortName("read");
-read.Add(Find_Methods().FindByShortName("pread"));
-read.Add(Find_Methods().FindByShortName("pread64"));
-
-// Find input influence on size of copy
-CxList sizeParam = All.GetParameters(read, 2);
-sizeParam = All.GetByAncs(sizeParam);
-CxList db = Find_Read() + Find_DB();
-CxList sanitize = All.GetByAncs(Find_Methods().FindByShortName("sizeof"));
-
-result = sizeParam.InfluencedByAndNotSanitized(db, sanitize);
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,14 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList db = Find_DB();
-   CxList sanitize = Find_Sanitize_SQL_Injection();
-
-   result = (Find_Read() + possible_db).InfluencingOnAndNotSanitized(possible_db, sanitize);
-
-   if (result.Count > 0)
-   {
-       result -= (Find_Read() + db).InfluencingOnAndNotSanitized(db, sanitize);
-   }
```

```
-}
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_Buffer_Improper_Index_Access

Code changes

```
---
+++
@@ -1,126 +1 @@
-CxList arrayCreateList = Find_ArrayCreateExpr();
-CxList declarators = Find_Declarators();
-CxList unknRefs = Find_Unknown_References();
-CxList indexes = Find_IndexerRefs();
-
-CxList inputs = Find_Inputs();
-CxList allRefsInputs = All.FindAllReferences(inputs);
-CxList indexesUnkRef = unknRefs.GetByAncs(indexes);
-CxList indexesUnkAllRefs = All.FindAllReferences(indexesUnkRef);
-CxList indexesNoInput = indexesUnkRef - allRefsInputs;
-CxList indexesInfluenced = indexesNoInput.DataInfluencedBy(inputs);
-
-CxList conditions = Find_Conditions();
-CxList binaryExpressions = Find_BinaryExpressions();
-List<string> relevantBinary = new List<string>() {"<", "<=", ">", ">="};
-CxList relevantBinaryList = binaryExpressions.FindByShortNames(relevantBinary);
-CxList binarysInIfs = relevantBinaryList.GetByAncs(conditions);
-CxList inConditions = indexesUnkAllRefs.GetByAncs(conditions);
-CxList sanitizedRefsNodes = binarysInIfs.DataInfluencedBy(inConditions).GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly);
-CxList sanitizedRefs = All.FindAllReferences(sanitizedRefsNodes).GetAncOfType(typeof(IndexerRef));
-CxList sanitizedIndexesUnkRef = unknRefs.GetByAncs(sanitizedRefs);
-
-CxList indexesInfluencedAndNotSanitized = indexesInfluenced - sanitizedIndexesUnkRef;
-
-CxList nodes = All.NewCxList();
-
-//The purpose of this lambda is to not duplicate code inside the foreach
-//This lambda receives the current node in the foreach and returns its IntegerIntervalAbstractValue
-Func <CxList, IntegerIntervalAbstractValue> calcAbsValue = arrayReference => {
-   IntegerIntervalAbstractValue arrayReferenceValue = null;
-   CxList definition = declarators.FindDefinition(arrayReference);
-   CxList arrayCreate = arrayCreateList.FindByFathers(definition);
-   ArrayCreateExpr arrayDef = arrayCreate.TryGetCSharpGraph<ArrayCreateExpr>();
-   if(arrayDef != null){
-       if(arrayDef.Sizes.Count != 0){
-           if (arrayDef.Sizes[0] != null && arrayDef.Sizes is ExpressionCollection) {
-               Expression col = arrayDef.Sizes[0] as Expression;
-               if(col is IntegerLiteral) {
-                   IntegerLiteral integer = col as IntegerLiteral;
-                   arrayReferenceValue = new IntegerIntervalAbstractValue(0, integer.Value);
```

```
-              }
-          }
-      }
-      else if(arrayDef.Initializer != null && arrayDef.Initializer is AbstractCollectionInitializer) {
-          AbstractCollectionInitializer values = arrayDef.Initializer as AbstractCollectionInitializer;
-          if(values.InitialValues != null) {
-              ExpressionCollection col = values.InitialValues as ExpressionCollection;
-              arrayReferenceValue = new IntegerIntervalAbstractValue(col.Count, col.Count);
-          }
-      }
-  }
-  return arrayReferenceValue;
-  };
-
-foreach(CxList arrayRef in indexes) {
-  try {
-      IndexerRef indexer = arrayRef.TryGetCSharpGraph<IndexerRef>();
-      Expression arrayIndexer = indexer.Indices[0] as Expression;
-      if (arrayIndexer is IntegerLiteral) {
-          continue;
-      }
-      CxList arrayReference = unknRefs.FindByFathers(arrayRef);
-
-      UnknownReference array = arrayReference.TryGetCSharpGraph<UnknownReference>();
-      if(arrayIndexer != null && array != null){
-          IntegerIntervalAbstractValue arrayReferenceValue = null;
-          long arrayValueMax = 0;
-          long arrayValueMin = 0;
-          if(array.AbsValue is ObjectAbstractValue) {
-              ObjectAbstractValue arrayAbstractValue = array.AbsValue as ObjectAbstractValue;
-              if(arrayAbstractValue != null) {
-                  //char msg[13] = {'h','l'};
-                  if(arrayAbstractValue.AllocatedSize != null) {
-                      arrayValueMax = arrayAbstractValue.AllocatedSize.UpperIntervalBound.GetValueOrDefault();
-                      arrayValueMin = arrayAbstractValue.AllocatedSize.LowerIntervalBound.GetValueOrDefault();
-                      arrayReferenceValue = new IntegerIntervalAbstractValue(arrayValueMin, arrayValueMax);
-                  }
-                      //char msg[] = {'h','l'};
-                  else{
-                      arrayReferenceValue = calcAbsValue(arrayReference);
-                  }
-              }
-          }
-              //Loops
-          else if (array.AbsValue is AnyAbstractValue){
-              arrayReferenceValue = calcAbsValue(arrayReference);
-          }
-          if(arrayReferenceValue == null) {
```

```
-              if(!(arrayIndexer.AbsValue is IntegerIntervalAbstractValue)) {
-                  CxList indexesUnkRefIn = unknRefs.FindByFathers(arrayRef);
-                  CxList common = indexesUnkRefIn * indexesInfluencedAndNotSanitized;
-                  if(common.Count > 0) {
-                      nodes.Add(arrayRef);
-                  }
-              }
-          }
-          else if(arrayReferenceValue != null) {
-              if(arrayIndexer.AbsValue != null && !(arrayIndexer.AbsValue is IntegerIntervalAbstractValue)) {
-                  CxList indexesUnkRefIn = unknRefs.FindByFathers(arrayRef);
-                  CxList common = indexesUnkRefIn * indexesInfluencedAndNotSanitized;
-                  if(common.Count > 0) {
-                      nodes.Add(arrayRef);
-                  }
-              }
-          }
-      }
-  }
-  catch(Exception e){
-      cxLog.WriteDebugMessage(e.Message);
-  }
-}
-
-//Creates flow with indexerRefs that are influenced by user input
-CxList unkRefsIndexer = unknRefs.FindByFathers(nodes);
-//Check if indexerRefs and Unknown refs from nodes have been initialized to exclude them
-CxList declsInit = declarators.FilterByDomProperty<Declarator>(x => x.InitExpression != null);
-CxList refsInit = unkRefsIndexer.FindAllReferences(declsInit);
-unkRefsIndexer -= refsInit;
-CxList nodesInit = nodes.FindAllReferences(declsInit);
-nodes -= nodesInit;
-CxList inputsNodes = unkRefsIndexer.DataInfluencedBy(inputs);
-CxList inputNodesUnkRefs = inputsNodes.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-CxList inputNodeIndexerRef = inputNodesUnkRefs.GetAncOfType(typeof(IndexerRef));
-nodes -= inputNodeIndexerRef;
-nodes.Add(inputsNodes);
-result = nodes;
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_Buffer_Overflow_malloc

Code changes

```
---
+++
@@ -1,13 +1 @@
-// The purpose of the query to protect the system from very big input malloc size
-// for example maloc(100000000) can stuck the system
-
```

```
-// The correct sanitization shoud be added
-
-CxList malloc = Find_Methods().FindByShortName("malloc");
-
-// Find input influence on size of copy
-CxList sizeParam = All.GetByAncs(malloc) - malloc;
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitize = Find_All_Strlen();
-
-result = sizeParam.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_Buffer_Overflow_read

Code changes

```
---
+++
@@ -1,12 +1 @@
-// The correct sanitization shoud be added
-
-CxList read = Find_Methods().FindByShortName("read");
-read.Add(Find_Methods().FindByShortName("pread"));
-read.Add(Find_Methods().FindByShortName("pread64"));
-
-// Find input influence on size of copy
-CxList sizeParam = All.GetParameters(read,2);
-sizeParam = All.GetByAncs(sizeParam);
-CxList inputs = Find_Interactive_Inputs();
-
-result = sizeParam.DataInfluencedBy(inputs);
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_CGI_Stored_XSS

Code changes

```
---
+++
@@ -1,18 +1 @@
-if (CGI().Count > 0) //web application (CGI)
-{
-   CxList possible_db = Find_DB_Heuristic();
-
-   if (possible_db.Count > 0)
-   {
-       CxList sanitize = Find_Sanitize() + All.FindByShortName("encode", false);
-       CxList stored = possible_db + Find_Read();
-       result = stored.InfluencingOnAndNotSanitized(Find_Outputs(), sanitize);
-
-       if (result.Count > 0)
-       {
```

```
-           CxList db = Find_DB();
-           stored = db + Find_Read();
-           result -= stored.InfluencingOnAndNotSanitized(Find_Outputs(), sanitize);
-       }
-   }
-}
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,32 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList tables = All.FindByName("*orders*", false) +
-       All.FindByName("*credit*", false) +
-       All.FindByName("*invoice*", false) +
-       All.FindByName("*booking*", false) +
-       All.FindByName("*bill*", false) +
-       All.FindByName("*payment*", false) +
-       All.FindByName("*account*", false) +
-       All.FindByName("*cash*", false) +
-       All.FindByName("*customer*", false);
-
-   CxList inputs = Find_Interactive_Inputs();
-
-   CxList user = All.FindByName("*user*", false) +
-       All.FindByName("*cust*", false) +
-       All.FindByName("*member*", false);
-
-   possible_db = possible_db.DataInfluencedBy(tables);
-   possible_db -= possible_db.DataInfluencedBy(user);
-   result = inputs.DataInfluencingOn(possible_db);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB();
-       db = db.DataInfluencedBy(tables);
-       db -= db.DataInfluencedBy(user);
-       result -= inputs.DataInfluencingOn(db);
-   }
-}
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---

+++

@@ -1,25 +1 @@

-CxList possible_db = Find_DB_Heuristic();

-

-if (possible_db.Count > 0)

-{

-   CxList input = Find_Interactive_Inputs();

-

-   CxList strings = Find_Strings();

-   CxList Select = strings.FindByName("*select*", false);

-   CxList Where = strings.FindByName("*where*", false);

-   CxList And = strings.FindByName("*And *", false) +

-       strings.FindByName("* And*", false);

-

-   possible_db = possible_db.DataInfluencedBy(Select).DataInfluencedBy(Where);

-   possible_db -= possible_db.DataInfluencedBy(And);

-

-   result = possible_db.DataInfluencedBy(input);

-

-   if (result.Count > 0)

-   {

-       CxList db = Find_DB();

-       db = db.DataInfluencedBy(Select).DataInfluencedBy(Where);

-       db -= db.DataInfluencedBy(And);

-       result -= db.DataInfluencedBy(input);

-   }

-}

+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_SQL_Injection

Code changes

```
---

+++

@@ -1,15 +1 @@

-CxList possible_db = Find_DB_Heuristic();

-

-if (possible_db.Count > 0)

-{

-   CxList inputs = Find_Interactive_Inputs();

-   CxList sanitized = Find_Sanitize_SQL_Injection();

-

-   result = inputs.InfluencingOnAndNotSanitized(possible_db, sanitized);

-

-   if (result.Count > 0)

-   {
```

```
-       CxList db = Find_DB();
-       result -= inputs.InfluencingOnAndNotSanitized(db, sanitized);
-   }
-}
+//This query is deprecated.
```

CPP / CPP_Heuristic / Heuristic_Unchecked_Return_Value

Code changes

```
---
+++
@@ -1,63 +1 @@
-// Use of uninitialized pointer
-//  ----------------------------
-//  In this query we look for NULL-initialized pointers, that the user
-//  tries to use although they have no actual value in this address.
-////////////////////////////////////////////////////////////////////////
-
-// Find undefined methods
-CxList empty = Find_Empty_Methods();
-CxList defs = All.FindDefinition(Find_Methods()) - empty;
-CxList undefinedMethods = Find_Methods() - All.FindAllReferences(defs);
-undefinedMethods -= undefinedMethods.FindByShortName("sizeof");
-undefinedMethods -= undefinedMethods.FindByShortName("strlen");
-
-// Find undefined methods in the right side of an assign expression
-CxList rightSide = All.FindByAssignmentSide(CxList.AssignmentSide.Right);
-CxList rightSideMethods = undefinedMethods * rightSide;
-
-// Which variables we want to check
-CxList testVars = All.FindByFathers(rightSideMethods.GetAncOfType(typeof(AssignExpr))).FindByAssignmentSide(CxList.AssignmentSide.Left)
-   + rightSideMethods.GetAncOfType(typeof(Declarator));
-
-// If statements conditions - to sanitize the check
-CxList conditions = Get_Conditions();
-
-// If statements that contain the vaviables to check
-CxList IfStmts = conditions.DataInfluencedBy(testVars).GetAncOfType(typeof(IfStmt));
-CxList IfStmtsInside = All.GetByAncs(IfStmts);
-
-// Assert is a sanitizer too
-CxList assert = Find_Methods().FindByName("assert");
-CxList assertParam = All.GetByAncs(assert);
-assertParam -= assertParam.FindByShortName("assert");
-
-// What we want to check for influence - the references of the variables
-// and all parameters of functions
-CxList toCheck = All.FindAllReferences(testVars) + All.GetParameters(All);
-
```

```
-// Remove all atomic types
-CxList builtinTypes = Find_Builtin_Types();
-toCheck -= toCheck * builtinTypes;
-
-CxList binary = Find_BinaryExpr();
-CxList boolOrIntExpr =
-   binary.FindByShortName("*") +
-   binary.FindByShortName("/") +
-   binary.FindByShortName("+") +
-   binary.FindByShortName("-") +
-   binary.FindByShortName("<") +
-   binary.FindByShortName(">") +
-   binary.FindByShortName("==") +
-   binary.FindByShortName("!=") +
-   binary.FindByShortName("<>") +
-   binary.FindByShortName("<=") +
-   binary.FindByShortName(">=") +
-   binary.FindByShortName("||") +
-   binary.FindByShortName("&&") +
-   Find_Unarys().FindByShortName("Not");
-
-toCheck -= boolOrIntExpr;
-
-// Find influence of relevant undefined methods on the things we want to check
-result = toCheck.InfluencedByAndNotSanitized(undefinedMethods, IfStmtsInside + assertParam + All.FindByType(typeof (MethodRef)));
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

CPP / CPP_Heuristic / Potential_Off_by_One_Error_in_Loops

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated. Please refer to Off_By_One_Error instead.
+//This query is deprecated.
```

CPP / CPP_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList db = Find_DB();
-CxList db_not_in_try = Improper_Exception_Handling(db);
-CxList db_in_try = db - db_not_in_try;
-
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitized = Find_Sanitize_SQL_Injection();
-
```

```
-result = inputs.InfluencingOnAndNotSanitized(db_in_try, sanitized);
+//This query is deprecated.
```

CPP / CPP_Low_Visibility / Creation_of_chroot_Jail_without_Changing_Working_Directory

Code changes

```
---
+++
@@ -5,55 +5,48 @@



 //1.serach for chroot and check if have chdir in the same scope
-
-CxList usingOfChroot = Find_Methods().FindByName("chroot");
-usingOfChroot.Add(Find_Methods().FindByName("fchroot"));
-CxList unsafeChroot = usingOfChroot;
+CxList methods = Find_Methods();
+CxList methodDecls = base.Find_MethodDecls();
+CxList usingOfChroot = methods.FindByShortNames("chroot", "fchroot");
+CxList unsafeChroot = All.NewCxList(usingOfChroot);

 //check each chroot scope to see if has chdir
 foreach (CxList curChroot in usingOfChroot)
 {
-   CxList statementCollection = curChroot.GetAncOfType(typeof(StatementCollection));
+   CxList statementCollection = curChroot.GetAncOfType<StatementCollection>();
    CxList allItemsInScope = All.GetByAncs(statementCollection);

    //find all methods that are called and check if they have the chdir
-   CxList methodInvokedInScope = allItemsInScope.FindByType(typeof(MethodInvokeExpr));
-   CxList methodInScopeDecl = All.FindAllReferences(methodInvokedInScope).FindByType(typeof(MethodDecl));
+   CxList methodInvokedInScope = allItemsInScope.FindByType<MethodInvokeExpr>();
+   CxList methodInScopeDecl = methodDecls.FindAllReferences(methodInvokedInScope);
    CxList allMethodsScope = All.GetByAncs(methodInScopeDecl);
    allItemsInScope.Add(allMethodsScope);

-   CxList scopeDir = allItemsInScope.FindByType(typeof(MethodInvokeExpr)).FindByName("chdir");
-   scopeDir.Add(allItemsInScope.FindByType(typeof(MethodInvokeExpr)).FindByName("fchdir"));
+   CxList scopeDir = methodInvokedInScope.FindByShortNames("chdir", "fchdir");
    if (scopeDir.Count > 0)
-   {
        unsafeChroot -= curChroot;
-   }
 }



 //2.serach for chdir and check if have chroot in the same scope

-CxList allChdir = Find_Methods().FindByName("chdir");
```

```
-allChdir.Add(Find_Methods().FindByName("fchdir"));
+CxList allChdir = methods.FindByShortNames("chdir", "fchdir");

 //check each chdir scope to see if has chroot
 foreach (CxList curChdir in allChdir)
 {
-   CxList statementCollection = curChdir.GetAncOfType(typeof(StatementCollection));
+   CxList statementCollection = curChdir.GetAncOfType<StatementCollection>();
    CxList allItemsInScope = All.GetByAncs(statementCollection);

    //find all methods that are called and check if they have the chroot
-   CxList methodInvoedInScope = allItemsInScope.FindByType(typeof(MethodInvokeExpr));
-   CxList methodInScopeDecl = All.FindAllReferences(methodInvoedInScope).FindByType(typeof(MethodDecl));
+   CxList methodInvoedInScope = allItemsInScope.FindByType<MethodInvokeExpr>();
+   CxList methodInScopeDecl = methodDecls.FindAllReferences(methodInvoedInScope);
    CxList allMethodsScope = All.GetByAncs(methodInScopeDecl);
    allItemsInScope.Add(allMethodsScope);

-   CxList scopeChroot = allItemsInScope.FindByType(typeof(MethodInvokeExpr)).FindByName("chroot");
-   scopeChroot.Add(allItemsInScope.FindByType(typeof(MethodInvokeExpr)).FindByName("fchroot"));
+   CxList scopeChroot = methodInvoedInScope.FindByShortNames("chroot", "fchroot");
    if (scopeChroot.Count > 0)
-   {
        unsafeChroot -= scopeChroot;
-   }

 }
```

CPP / CPP_Low_Visibility / NULL_Pointer_Dereference

Code changes

```
---
+++
@@ -1,20 +1,62 @@
 CxList pointers = Find_Pointers();
 CxList unary = Find_Unarys();
+CxList binaryExpr = Find_BinaryExpr();
+CxList methods = Find_Methods();
+CxList parameters = Find_Param();
+CxList unknownReferences = Find_Unknown_References();
+CxList conditions = Find_Conditions();
+CxList ifs = Find_Ifs();
 CxList pointerTotarget = pointers.GetByAncs(All.NewCxList(unary.FindByShortName("Pointer"),
                                                           pointers.FindByType<IndexerRef>()));
-CxList unknownReferences = Find_Unknown_References();
+CxList equalsNullCond = Find_Ifs_NullCondition(true);
+CxList notEqualsNullCond = Find_Ifs_NullCondition(false);
```

```
 // Remove pointers that check pointer condition
-CxList findConditions = Find_Conditions();
-CxList pointerCondition = pointers.GetByAncs(findConditions);
-CxList ifs = pointerCondition.GetFathers().FindByType<IfStmt>();
-pointerTotarget -= pointerTotarget.GetByAncs(ifs);
+CxList pointerCondition = pointers.GetByAncs(conditions);
+CxList pointerIfs = pointerCondition.GetFathers().FindByType<IfStmt>();
+pointerTotarget -= pointerTotarget.GetByAncs(pointerIfs);
+
+// Remove pointer accesses that are guarenteed to be safe via condition check.
+CxList stmtCollectionNullSafe = All.NewCxList(equalsNullCond.CxSelectDomProperty<IfStmt>(x => x.FalseStatements),
+                                              equalsNullCond.CxSelectDomProperty<TernaryExpr>(x => x.False),
+                                           notEqualsNullCond.CxSelectDomProperty<IfStmt>(x => x.TrueStatements),
+                                           notEqualsNullCond.CxSelectDomProperty<TernaryExpr>(x => x.True),
+                                           notEqualsNullCond.CxSelectDomProperty<IterationStmt>(x => x.Statements));
+CxList safePointerAccess = All.NewCxList();
+CxList pointersToTest = All.NewCxList(pointerTotarget, pointers.GetTargetsWithMembers());
+foreach (CxList ppt in pointersToTest)
+{
+   if (ppt.GetByAncs(stmtCollectionNullSafe).Count > 0 ||
+       (ppt.GetByAncs(conditions).GetAncOfType<IfStmt>() * stmtCollectionNullSafe.GetFathers()).Count > 0)
+   {
+       CxList pptCond = ppt.GetAncOfType(typeof(IfStmt), typeof(TernaryExpr), typeof(IterationStmt));
+       CxList varInCond = unknownReferences.GetByAncs(conditions.FindByFathers(pptCond));
+       varInCond -= varInCond.GetByAncs(varInCond.GetMembersOfTarget());
+       varInCond -= ppt;
+       varInCond = varInCond.FilterByDomProperty<UnknownReference>(x => x.ShortName == ppt.GetFirstGraph().ShortName);
+       if (varInCond.Count > 0)
+       {
+           safePointerAccess.Add(ppt);
+       }
+   }
+}
+pointerTotarget -= safePointerAccess;



 ////////////////////////// Null Values ////////////////////////
 CxList nullValues = All.NewCxList(Find_NullLiteral(),
    Find_CharLiteral().FindByShortName("\0"),
    Find_IntegerLiterals().FindByShortName("0"));
+
+// SANITIZER CALCULATION BEFORE "nullValues" CLEANUP: ASSERT Operations
+CxList identityBin = All.NewCxList(binaryExpr.GetByBinaryOperator(BinaryOperator.IdentityInequality),
+                                   binaryExpr.GetByBinaryOperator(BinaryOperator.IdentityEquality));
+identityBin = nullValues.FindByFathers(identityBin).GetFathers();
+CxList notUnaryOpr = unary.FindByShortName("Not");
+CxList sanitizerFatherOfRef = All.NewCxList(identityBin, notUnaryOpr,parameters);
+CxList unknownRefsWithNoMembers = unknownReferences - unknownReferences.GetMembersOfTarget().GetTargetOfMembers();
```

```
+CxList assertMethods = methods.FindByShortName("assert", false);
+CxList sanitizedByAssert = unknownRefsWithNoMembers.FindByFathers(sanitizerFatherOfRef).GetByAncs(assertMethods);


 //Remove Null Values Assigned to IndexerRefs that the base type is not a pointer.
 CxList nullAssignee = nullValues.GetAssignee();
@@ -48,20 +90,20 @@
 ////////////////////////// Influencing ////////////////////////
 // Remove the 0 or 1  ( return 0  or return 1 )
 CxList returnStmts = Find_ReturnStmt();
-CxList removeZeroInReturn = All.NewCxList();
-removeZeroInReturn.Add(nullValues.GetByAncs(returnStmts));
+CxList removeZeroInReturn = nullValues.GetByAncs(returnStmts);
 removeZeroInReturn -= returnStmts;
 CxList influencing = nullValues - removeZeroInReturn;


 // Include declarations such as: shared_ptr<T> x (new T); where T is a built-in type.
 string[] builtInTypes = {"int", "long", "short", "float", "double", "bool"};
-CxList uninitBuiltIn = Find_ObjectCreations().FindByShortNames(builtInTypes).FilterByDomProperty<ObjectCreateExpr>(obj => 0 == obj.Parameters?.Count);
+CxList uninitBuiltIn = Find_ObjectCreations().FindByShortNames(builtInTypes)
+                                             .FilterByDomProperty<ObjectCreateExpr>(obj => 0 == obj.Parameters?.Count);
 CxList shared_ptrs = Find_Uninitialized_Pointer_Decl().GetByAncs(Find_Smart_Pointer_Declarators());
 CxList uninitSharedPtrs = uninitBuiltIn.GetByAncs(shared_ptrs).GetAncOfType<Declarator>();
 influencing.Add(uninitSharedPtrs);

 // Remove the 0 that are in conditions (if, while, for, etc...)
-influencing -= influencing.GetByAncs(findConditions);
+influencing -= influencing.GetByAncs(conditions);


 // Remove the 0 that are in an IndexerRef
 CxList zeroIndexerRef = nullValues.GetAncOfType<IndexerRef>();
@@ -69,10 +111,7 @@




 ////////////////////////// Sanitizers ////////////////////////
-CxList sanitizers = All.NewCxList();
-
-// Sanitize Conditions
-sanitizers.Add(pointerCondition);
+CxList sanitizers = All.NewCxList(safePointerAccess,sanitizedByAssert,pointerCondition);


 // Sanitize &*
 CxList address = unary.FindByShortName("Address");
@@ -81,24 +120,20 @@
 sanitizers.Add(pointers.FindByFathers(addressChild));


 // Sanitize address passed as parameter
-CxList parameters = Find_Param();
 sanitizers.Add(pointers.FindByFathers(address.FindByFathers(parameters)));
```

```
 // Sanitize others
-CxList binaryExpr = Find_BinaryExpr();
 sanitizers.Add(binaryExpr);

 // Remove pointer dereferences appearing in if blocks that are not executed (ie if (x != null)// when x is null)
-sanitizers.Add(pointerTotarget.GetByAncs(Find_Ifs_NullCondition()));
 CxList pointerToTargetPlusInfluencing = All.NewCxList();
 pointerToTargetPlusInfluencing.Add(new List<CxList>{pointerTotarget, influencing});
 sanitizers.Add(base.Find_Same_Value_Sanitizers_Exclude_Sinks_and_Sources(pointerToTargetPlusInfluencing));

 //Removes pointer dereferences sanitized by "if(x==null)return;"
+CxList ifNullWithReturnStmts = returnStmts.GetAncOfType<IfStmt>() * equalsNullCond;
 CxList equalityOperators = binaryExpr.GetByBinaryOperator(BinaryOperator.IdentityEquality);
-CxList equalsNull = nullValues.GetFathers() * equalityOperators;
-CxList ifNullWithReturnStmts = returnStmts.GetAncOfType<IfStmt>() * equalsNull.GetFathers();
-CxList binaryOperatorParts = unknownReferences.FindByFathers(equalsNull.FindByFathers(ifNullWithReturnStmts));
+CxList binaryOperatorParts = unknownReferences.FindByFathers(equalityOperators.FindByFathers(ifNullWithReturnStmts));
 CxList sanitizedReferences = unknownReferences.FindAllReferences(binaryOperatorParts);
 sanitizers.Add(sanitizedReferences);

@@ -119,8 +154,20 @@
 CxList pointerToInfluencing = pointerTotarget.InfluencedByAndNotSanitized(influencing, sanitizers);
 pointerToInfluencing = pointerToInfluencing.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+// Clear flows where the dereference originated directly from the right hand side.
+// (Ie: 'arr' in arr[i] = elem, when elem == NULL)
+CxList lastNode = pointerToInfluencing.GetLastNodesInPath();
+CxList lastAssigners = All.NewCxList(lastNode.GetAssigner(), lastNode.GetFathers().GetAssigner());
+// Remove Impossible flows, where an allegdly null pointer completes a Member Access Operation before dereference.
+CxList refsWithMembers = unknownReferences.GetMembersOfTarget().GetTargetOfMembers();
+refsWithMembers -= lastNode;
+// Intersection Procedure
+CxList nodesToIntersect = All.NewCxList(lastAssigners, refsWithMembers);
+pointerToInfluencing -= pointerToInfluencing.IntersectWithNodes(nodesToIntersect);
+
 // For Flows inside IterationStmt, remove Flows that cross scopes.
-foreach (CxList pti in pointerToInfluencing.GetCxListByPath()){
+foreach (CxList pti in pointerToInfluencing.GetCxListByPath())
+{
     if(pti.GetAncOfType<IterationStmt>().Count == 1){
         CxList nullAssignement = pti.GetFirstNodesInPath();
         CxList nullDereference = pti.GetLastNodesInPath();
@@ -132,21 +179,41 @@
     }
 }

-//Sanitize dereferences that are previously initialized inside a (x==NULL) condition.
```

```
+
+//------- Sanitize dereferences that are previously initialized inside conditions. -------
 CxList refsOfTarget = unknownReferences.FindAllReferences(pointerTotarget);
 CxList assignNotNull = refsOfTarget.GetAncOfType<AssignExpr>() - nullValues.GetAncOfType<AssignExpr>();
-CxList ifNullWithSafeAssign = equalsNull.GetAncOfType<IfStmt>() * assignNotNull.GetAncOfType<IfStmt>();
-CxList safeIfBinaryExpr = binaryExpr.GetByAncs(ifNullWithSafeAssign);
-CxList safeRefs = refsOfTarget.GetByAncs(safeIfBinaryExpr);
-foreach (CxList pti in pointerToInfluencing.GetCxListByPath()){
+CxList allNullConds = All.NewCxList(equalsNullCond, notEqualsNullCond);
+CxList stmtCollectionNullUnsafe = All.NewCxList(equalsNullCond.CxSelectDomProperty<IfStmt>(x => x.TrueStatements),
+                                                equalsNullCond.CxSelectDomProperty<TernaryExpr>(x => x.True),
+                                                equalsNullCond.CxSelectDomProperty<IterationStmt>(x => x.Statements),
+                                                notEqualsNullCond.CxSelectDomProperty<IfStmt>(x => x.FalseStatements),
+                                                notEqualsNullCond.CxSelectDomProperty<TernaryExpr>(x => x.False));
+CxList safeRefs = All.NewCxList();
+foreach (CxList nullCond in allNullConds)
+{
+   CxList assignNotNullInCond = assignNotNull.GetByAncs(nullCond).GetByAncs(stmtCollectionNullUnsafe);
+   if (assignNotNullInCond.Count > 0){
+       CxList varsInCond = unknownReferences.GetByAncs(conditions.FindByFathers(nullCond));
+       varsInCond -= varsInCond.GetByAncs(varsInCond.GetMembersOfTarget());
+       varsInCond = varsInCond.FilterByDomProperty<UnknownReference>(varInCond =>
+                                 assignNotNullInCond.FilterByDomProperty<AssignExpr>(assign =>
+                                                    assign.Left.ShortName == varInCond.ShortName).Count > 0);
+       safeRefs.Add(varsInCond);
+   }
+}
+
+//Check if condition is between Null Assignement and Access.
+foreach (CxList pti in pointerToInfluencing.GetCxListByPath())
+{
     CSharpGraph nullAssignement = pti.GetFirstNodesInPath().GetFirstGraph();
     CSharpGraph nullDereference = pti.GetLastNodesInPath().GetFirstGraph();
     CxList ifInBetween = safeRefs.FilterByDomProperty<UnknownReference>(x =>
         x.VariableName == nullDereference.ShortName &&
         x.LinePragma.FileName == nullAssignement.LinePragma.FileName &&
         x.LinePragma.FileName == nullDereference.LinePragma.FileName &&
-        x.LinePragma.Line > nullAssignement.LinePragma.Line &&
-        x.LinePragma.Line < nullDereference.LinePragma.Line);
+        x.LinePragma.Line >= nullAssignement.LinePragma.Line &&
+        x.LinePragma.Line <= nullDereference.LinePragma.Line);
     if (ifInBetween.Count > 0)
     {
         pointerToInfluencing -= pti;
@@ -155,23 +222,26 @@


 ////////////////////////// Pointer member access without memory allocation ////////////////////////
 CxList pointerMemberAccess = pointers.GetTargetsWithMembers();
```

```
+pointerMemberAccess -= safePointerAccess;
 CxList binaryExprEqual = binaryExpr.FindByShortName("==");
-CxList methods = Find_Methods();
 CxList forEach = Find_ForEachStmt();
 CxList parametersAddress = address.GetByAncs(parameters);
 CxList parametersAddressPointer = pointers.GetByAncs(parametersAddress);
-
-foreach (CxList memberAccess in pointerMemberAccess) {
+foreach (CxList memberAccess in pointerMemberAccess)
+{
     CxList pointerDefinition = pointers.FindDefinition(memberAccess).FindByType<Declarator>();
     pointerDefinition -= pointerDefinition.GetByAncs(forEach);
     CxList definitionScope = methods.GetMethod(pointerDefinition);
+   if (definitionScope.Count == 0) continue;
     CxList memberAccessScope = methods.GetMethod(memberAccess);
+   if (memberAccessScope.Count == 0) continue;
     CxList pointerReferences = pointers.FindAllReferences(pointerDefinition) - pointerMemberAccess;
     CxList pointerReferencesScope = methods.GetMethod(pointerReferences);
     CxList scope = memberAccessScope * definitionScope * pointerReferencesScope;

     // All in same scope?
-   if (scope.Count == 1) {
+   if (scope.Count == 1)
+   {
        // get declaration initialization
        CxList pointerAssign = All.FindInitialization(pointerDefinition);
        // get all references assignees
@@ -192,7 +262,8 @@
        // Example: if (pointer == NULL) return ;
        CxList allUnderDefinitionScope = binaryExpr.GetByAncs(definitionScope);
        CxList comparesInScope = binaryExprEqual * allUnderDefinitionScope;
-       foreach (CxList compare in comparesInScope) {
+       foreach (CxList compare in comparesInScope)
+       {
            CxList left = pointerReferences.FindByFathers(compare);
            CxList right = nullValues.FindByFathers(compare);
            CxList childReturnStmts = returnStmts.FindByFathers(compare);
@@ -205,7 +276,8 @@
        CxList possibleNotNullBinary = binaryExpr.GetByBinaryOperator(BinaryOperator.IdentityInequality);
        possibleNotNullBinary = possibleNotNullBinary * allUnderDefinitionScope;
        CxList toRemove = All.NewCxList();
-       foreach(CxList val in possibleNotNullBinary) {
+       foreach(CxList val in possibleNotNullBinary)
+       {
            CxList left = pointerReferences.FindByFathers(val);
            CxList right = nullValues.FindByFathers(val);
            if (left.Count >= 1 && right.Count >= 1)
@@ -213,13 +285,15 @@
```

```
        }


        // Gather cases where "!= null" is implicit (if (x) x->a)
-       CxList impNull = pointerReferences.GetFathers() * ifs;
+       CxList impNull = pointerReferences.GetFathers() * pointerIfs;
        CxList notNullAndPointerChecks = toRemove.Clone();
-       CxList ifNotNullChecks = (notNullAndPointerChecks.GetFathers() * Find_Ifs());
+       CxList ifNotNullChecks = (notNullAndPointerChecks.GetFathers() * ifs);
        ifNotNullChecks.Add(impNull);
        CxList unsanMembers = memberAccess - memberAccess.GetByAncs(ifNotNullChecks.GetBlocksOfIfStatements(true));
-
-       if (pointerAssign.FindInScope(pointerDefinition, unsanMembers).Count == 0 && unsanMembers.Count > 0) {
+       CxList assertCheck = sanitizedByAssert.FindByShortName(pointerDefinition.GetFirstGraph().ShortName);
+       CxList safePointerRef = All.NewCxList(pointerAssign,assertCheck);
+       if (safePointerRef.FindInScope(pointerDefinition, unsanMembers).Count == 0 && unsanMembers.Count > 0)
+       {
            result.Add(pointerDefinition.Concatenate(unsanMembers));
        }
    }
```

CPP / CPP_Low_Visibility / Potential_Path_Traversal

Code changes

```
---
+++
@@ -1,2 +1 @@
-//This query is deprecated. Please consider Path_Traversal instead.
-cxLog.WriteDebugMessage("The query Potential_Path_Traversal is deprecated.");
+//This query is deprecated.
```

CPP / CPP_Low_Visibility / Sizeof_Pointer_Argument

Code changes

```
---
+++
@@ -1,4 +1 @@
-////////////////////////////////////////////////////////////////////////
-//  This query is deprecated.
-//  Please use Use_of_Sizeof_On_a_Pointer_Type instead.
-////////////////////////////////////////////////////////////////////////
+//This query is deprecated.
```

CPP / CPP_Low_Visibility / Stored_Blind_SQL_Injections

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList db = Find_DB();
-CxList db_not_in_try = Improper_Exception_Handling(db);
-CxList db_in_try = db - db_not_in_try;
```

```
-
-CxList inputs = Find_Read()+Find_DB();
-CxList sanitized = Find_Sanitize_SQL_Injection();
-
-result = inputs.InfluencingOnAndNotSanitized(db_in_try, sanitized);
+//This query is deprecated.
```

CPP / CPP_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,49 +1,20 @@
 CxList psw = Find_Passwords();
-CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList psw_in_lSide_decl = psw_in_lSide.FindByType<Declarator>();
-CxList strLiterals = Find_Strings();
-CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
-//when the hardcoded string includes a space or dot we believe
-//it is not a password string
-lit_in_rSide -= lit_in_rSide.FindByName("* *");
-lit_in_rSide -= lit_in_rSide.FindByName("*.*");
-lit_in_rSide -= lit_in_rSide.FindByName("*/*");
-lit_in_rSide -= lit_in_rSide.FindByName("*\\*");
+CxList methods = Find_Methods();
+CxList strLiterals = Find_Strings().FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);
+CxList useOfHardcodedPasswords = Common_Low_Visibility.Use_Of_Hardcoded_Password();


-//empty string is OK
-lit_in_rSide -= Find_Empty_Strings();
-
-// Password in declaration
-CxList PasswordInDecl = psw_in_lSide_decl.FindByInitialization(lit_in_rSide);
-
-//remove passwords with equal name and contant ==> currPassword = "currPassword";
-CxList notHdPass = All.NewCxList();
-char[] trimChars = new char[2] { '\'', '"'};
-foreach(CxList currPass in PasswordInDecl)
-{
-   CxList currStrInLeft = lit_in_rSide.FindInitialization(currPass);
-   string strName = currStrInLeft.GetName().Trim(trimChars);
-   string passName = currPass.GetName();
-   if (passName.Equals(strName))
-   {
-       notHdPass.Add(currPass);
-   }
-}
-PasswordInDecl -= notHdPass;
-
```

```
-CxList methods = Find_Methods();
 CxList strcmp = methods.FindByShortNames("strcmp", "strncmp", "bcmp", "strcoll");
 CxList strcmpParam1 = All.GetParameters(strcmp, 0);
 CxList strcmpParam2 = All.GetParameters(strcmp, 1);

-   //strcmp(password, "myPass")
-   //strcnmp(password, "myPass", length)
-   //bcmp(password, "myPass", cnt)
+//strcmp(password, "myPass")
+//strcnmp(password, "myPass", length)
+//bcmp(password, "myPass", cnt)
 CxList hPassInStrcmp = All.FindByParameters(strcmpParam1 * psw).FindByParameters(strcmpParam2 * strLiterals);

-   //strcmp("myPass", password)
-   //strcnmp("myPass", password, length)
-   //bcmp("myPass", password, cnt)
+//strcmp("myPass", password)
+//strcnmp("myPass", password, length)
+//bcmp("myPass", password, cnt)
 hPassInStrcmp.Add(All.FindByParameters(strcmpParam2 * psw).FindByParameters(strcmpParam1 * strLiterals));

 // Find password in an "compare" operation
@@ -53,29 +24,11 @@
 eq *= strLiterals.GetMembersOfTarget();
 equalsPassword.Add(psw.GetByAncs(eq));

-// Find password in a '==' operator
-CxList EqualBinaryExpr = psw.GetFathers().FindByType<BinaryExpr>().
-   GetByBinaryOperator(Checkmarx.Dom.BinaryOperator.IdentityEquality);
-CxList EqualOperatorStrings = All.NewCxList();
-foreach(CxList bin in EqualBinaryExpr)
-{
-   CxList password = psw.FindByFathers(bin);
-   CxList stringLit = strLiterals.FindByFathers(bin);
-
-   if(password.Count > 0 && stringLit.Count > 0)
-   {
-       EqualOperatorStrings.Add(stringLit);
-   }
-}
 //equals of type "string".compare(password);
 CxList strEQ = strLiterals.GetMembersOfTarget().FindByShortName("compare");
 strEQ = psw.GetByAncs(strEQ);
 equalsPassword.Add(strEQ);

-// Password in simple assignment
-CxList assignPassword = psw_in_lSide.GetAncOfType<AssignExpr>();
-assignPassword = lit_in_rSide.GetByAncs(assignPassword);
```

```
-
 CxList macroPassword = strLiterals.GetParameters(methods.FindByShortName("CxPw"));


-result.Add(PasswordInDecl, hPassInStrcmp, equalsPassword, assignPassword, macroPassword, EqualOperatorStrings);
+result.Add(useOfHardcodedPasswords, hPassInStrcmp, equalsPassword, macroPassword);
```

CPP / CPP_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,21 +1 @@
-CxList tables = All.FindByShortName("*orders*", false);
-tables.Add(All.FindByShortName("*credit*", false));
-tables.Add(All.FindByShortName("*invoice*", false));
-tables.Add(All.FindByShortName("*booking*", false));
-tables.Add(All.FindByShortName("*bill*", false));
-tables.Add(All.FindByShortName("*payment*", false));
-tables.Add(All.FindByShortName("*account*", false));
-tables.Add(All.FindByShortName("*cash*", false));
-tables.Add(All.FindByShortName("*customer*", false));
-
-CxList inputs = Find_Interactive_Inputs();
-CxList db = Find_DB();
-
-CxList user = All.FindByShortName("*user*", false);
-user.Add(All.FindByShortName("*cust*", false));
-user.Add(All.FindByShortName("*member*", false));
-
-db = db.DataInfluencedBy(tables);
-db -= db.DataInfluencedBy(user);
-
-result = inputs.DataInfluencingOn(db);
+//This query is deprecated.
```

CPP / CPP_Medium_Threat / Divide_By_Zero

Code changes

```
---
+++
@@ -71,7 +71,8 @@
 foreach(CxList binaryExpr in binaryExprs)
 {
     BinaryExpr binaryExprDOM = binaryExpr.TryGetCSharpGraph<BinaryExpr>();
-    if (binaryExprDOM != null && binaryExprDOM.Operator == BinaryOperator.Divide || binaryExprDOM.Operator == BinaryOperator.Modulus)
+    if (binaryExprDOM != null && binaryExprDOM.Operator == BinaryOperator.Divide
+        || binaryExprDOM.Operator == BinaryOperator.Modulus)
     {
         Expression right = binaryExprDOM.Right;
         if (right != null)
```

```
@@ -82,7 +83,7 @@
 }



-// Remove divs preceded by an if with return statement and
+// Remove divs preceded by an if with return or break (for iterations) statement
 // that checks if the divisor is null

 CxList ifs = Find_Ifs();
@@ -103,9 +104,20 @@
     relevZeros.GetFathers().CxSelectDomProperty<BinaryExpr>(x => x.Left),
     relevNots.CxSelectDomProperty<UnaryExpr>(x => x.Right));

+CxList iterations = Find_IterationStmt();
+CxList breakStmts = Find_BreakStmt();
+CxList breakIfs = breakStmts.GetByAncs(ifs.GetByAncs(iterations)).GetAncOfType<IfStmt>();
+
+CxList relevBreakIfs = zero.GetByAncs(
+   binaryExprs.GetByBinaryOperator(BinaryOperator.IdentityEquality).FindByFathers(breakIfs).GetFathers())
+   .GetFathers().CxSelectDomProperty<BinaryExpr>(x => x.Right.GraphType == GraphTypes.IntegerLiteral ? x.Left : x.Right);
+
 //Check if the items in rightToDiv are preceded by those ifs, if so they're safe
 foreach (CxList varRef in rightToDiv) {
-   CxList currIfVar = unknown.FindAllReferences(varRef) * ifVars;
+   CxList currIfVar = unknown.FindAllReferences(varRef) * ifVars;
+
+   if (currIfVar.Count == 0 && varRef.GetByAncs(iterations).Count > 0)
+       currIfVar = unknown.FindAllReferences(varRef) * relevBreakIfs;

     if (currIfVar.Count > 0) {

@@ -120,9 +132,8 @@

             CxList stmts = currIfVar.GetFathers().GetFathers().CxSelectDomProperty<IfStmt>(x => x.TrueStatements);

-            if (varRef.GetByAncs(stmts).Count == 0){
+            if (varRef.GetByAncs(stmts).Count == 0)
                 rightToDiv -= varRef;
-            }
         }
     }
 }
@@ -132,11 +143,12 @@
 CxList zeroAbsVal = rightToDiv.FindByAbstractValue(
     _ => _ is IntegerIntervalAbstractValue && _.Contains(zeroAbstractValue));
 rightToDiv -= zeroAbsVal;
+zeroAbsVal -= zeroAbsVal.NotInfluencedBy(zero).InfluencedBy(sanitize).FindByType<UnknownReference>();
```

```
 result.Add(zeroAbsVal);



-//--------------------------
+////////////////////////////
 //Numeric variables whose abstract value is AnyAbstractValues may contain 0, so they should be part of the result
 CxList decimals = Find_Integers() * unknown;
 decimals.Add(unknown.FindByTypes("float","double"));
@@ -147,7 +159,7 @@
 //Find possible zeros that are in the context of an IF/ELSE/Loop where it appears also in the condition
 CxList possibleZerosInConditions = conditions * possibleZeros;
 CxList conditionsIfsLoops = All.NewCxList();
-conditionsIfsLoops.Add(ifs, Find_IterationStmt());
+conditionsIfsLoops.Add(ifs, iterations);
 CxList possibleZerosInConditionalStmts = possibleZeros.GetByAncs(conditionsIfsLoops);
 foreach(CxList possibleZero in possibleZerosInConditions){
    CxList condStmt = possibleZero.GetAncOfType<IfStmt>();
@@ -167,7 +179,7 @@

 possibleZeros -= possibleZerosToRemove;
 result.Add(rightToDiv * possibleZeros);
-//----------------------
+//////////////////////-

 // Add real literal zeros (0.0) on the right side of divide operations to the results.
 CxList literalZeros = rightToDiv * zero;
```

CPP / CPP_Medium_Threat / Inadequate_Pointer_Validation

Code changes

```
---
+++
@@ -1,11 +1 @@
-// Inadequate Pointer Validation
-// ----------------------------
-// The functions are obsolete and cannot guarantee that a pointer is valid or referenced memory is safe to use.
-
-
-CxList methods = Find_Methods();
-
-result =  methods.FindByShortName("IsBadWritePtr") +
-         methods.FindByShortName("IsBadCodePtr")  +
-         methods.FindByShortName("IsBadReadPtr")  +
-         methods.FindByShortName("IsBadStringPtr");
+//This query is deprecated.
```

CPP / CPP_Medium_Threat / Memory_Leak

Code changes

```
---
+++
@@ -11,6 +11,7 @@
 CxList declarators = Find_Declarators();
 CxList methods = Find_Methods();
 CxList parms = Find_Parameters().CxSelectDomProperty<Param>(x => x.Value);
+CxList objCreateExpr = Find_ObjectCreations();
 CxList fieldDeclaration = declarators.GetByAncs(fieldDecls);
 CxList references = All.NewCxList(unknownRefs, Find_IndexerRefs());


@@ -63,7 +64,8 @@
 }


 // 2. Remove allocations that influence on a memory deallocation
-memoryAllocation -= memoryAllocation.DataInfluencingOn(nonCatchMemoryDeallocation).GetFirstNodesInPath();
+CxList sanitizers = memoryAllocation.GetByAncs(parms).GetByAncs(objCreateExpr);
+memoryAllocation -= memoryAllocation.InfluencingOnAndNotSanitized(nonCatchMemoryDeallocation,sanitizers).GetFirstNodesInPath();


 CxList memmoryAllocVar = memoryAllocation.GetAncOfType<AssignExpr>().CxSelectDomProperty<AssignExpr>(assign => assign.Left);
 // remove if they are used inside a function call
```

CPP / CPP_MISRA_C / R05_07_Identifier_Name_Reused

Code changes

```
---
+++
@@ -16,8 +16,8 @@
 identifiers -= identifiers.GetByAncs(All.FindByFieldAttributes(Dom.Modifiers.Extern));


 // remove method declaration if we have the definition
-CxList methodDeclAndDefs = identifiers.FindByType(typeof(MethodDecl));
-CxList methodDefs = methodDeclAndDefs * All.FindByType(typeof(StatementCollection)).GetFathers();
+CxList methodDeclAndDefs = identifiers.FindByType<MethodDecl>();
+CxList methodDefs = methodDeclAndDefs * Find_StatementCollection().GetFathers();
 CxList methodDecls = methodDeclAndDefs - methodDefs;
 CxList doubleMethods = identifiers.GetByAncs(methodDecls.FindByShortName(methodDefs));
 identifiers -= doubleMethods;
```

CPP / CPP_MISRA_C / R08_03_Identical_Function_Decl_Def

Code changes

```
---
+++
@@ -13,14 +13,15 @@


 */


-CxList methDecls = All.FindByType(typeof(MethodDecl));
-CxList typeRef = All.FindByType(typeof(TypeRef));
+CxList methDecls = Find_MethodDecls();
```

```
+CxList typeRef = Find_TypeRef();
+CxList paramDecls = Find_ParamDecl();
 //get all method definitions:
-CxList methDef = methDecls - All.FindByType(typeof(StatementCollection)).FindByFathers(methDecls).GetFathers();
+CxList methDef = methDecls - Find_StatementCollection().FindByFathers(methDecls).GetFathers();

-CxList allMethodsWithSameName = All.FindByType(typeof(MethodDecl)).FindByShortName(methDef);
-CxList allMdParams = All.FindByType(typeof(ParamDecl)).GetParameters(methDef);
-CxList allMwsnParams = All.FindByType(typeof(ParamDecl)).GetParameters(allMethodsWithSameName);
+CxList allMethodsWithSameName = methDecls.FindByShortName(methDef);
+CxList allMdParams = paramDecls.GetParameters(methDef);
+CxList allMwsnParams = paramDecls.GetParameters(allMethodsWithSameName);
 CxList allMdReturnType = typeRef.FindByFathers(methDef);
 CxList allMwsnReturnType = typeRef.FindByFathers(allMethodsWithSameName);
 CxList mwsnTypeRef = typeRef.GetByAncs(allMwsnParams);
@@ -41,19 +42,20 @@

         if(rtoo.ShortName != rt.ShortName)
         {
-            result.Add(mwsn + md);
+            result.Add(mwsn, md);
             nonCompliant.Add(mwsn);
-            allMethodsWithSameName -= mwsn + md;
+            allMethodsWithSameName -= All.NewCxList(mwsn, md);
         }
     }
     methodsWithSameName -= nonCompliant;
     foreach(CxList mwsn in  methodsWithSameName)
     {
         CxList otherParamTypes = mwsnTypeRef.GetByAncs(allMwsnParams.GetParameters(mwsn));
+
         if(otherParamTypes.Count != ParamTypes.Count)
         {
-            result.Add(mwsn + md);
-            allMethodsWithSameName -= mwsn + md;
+            result.Add(mwsn, md);
+            allMethodsWithSameName -= All.NewCxList(mwsn, md);
             continue;
         }
         if(otherParamTypes.Count == 0 && ParamTypes.Count == 0)
@@ -67,16 +69,15 @@
             continue;
         }

-       CxList defParamColl = ParamTypes.GetAncOfType(typeof(ParamDeclCollection));
-       CxList declParamColl = otherParamTypes.GetAncOfType(typeof(ParamDeclCollection));
+       CxList defParamColl = ParamTypes.GetAncOfType<ParamDeclCollection>();
        for (int i = 0; i < defParamColl.Count; i++)
```

```
        {
-           CSharpGraph def = ParamTypes.data.GetByIndex(i) as CSharpGraph;
-           CSharpGraph decl = otherParamTypes.data.GetByIndex(i) as CSharpGraph;
+           CSharpGraph def = ParamTypes.ElementAt(i).TryGetCSharpGraph<CSharpGraph>();
+           CSharpGraph decl = otherParamTypes.ElementAt(i).TryGetCSharpGraph<CSharpGraph>();
            if(def.ShortName != decl.ShortName)
            {
-               result.Add(mwsn + md);
-               allMethodsWithSameName -= mwsn + md;
+               result.Add(mwsn, md);
+               allMethodsWithSameName -= All.NewCxList(mwsn, md);
            }
        }
    }
```

CPP / CPP_MISRA_C / R08_07_Block_Scope_Obj_If_Used_By_Single_Function

Code changes

```
---
+++
@@ -17,19 +17,20 @@
 */

 CxList vars = Find_All_Declarators();
-vars -= All.FindByType(typeof(StringLiteral)).FindByName("CX_TYPEDEF").GetFathers();
-CxList classes = All.FindByType(typeof(ClassDecl));
+vars -= Find_String_Literal().FindByShortName("CX_TYPEDEF").GetFathers();
+CxList classes = Find_ClassDecl();
 classes -= classes.FindByShortName("checkmarx_default_classname*");
-CxList atBlockScope = vars.GetByAncs(All.FindByType(typeof(StatementCollection))
-   + classes + All.FindByType(typeof(StructDecl)));
+CxList atBlockScope = All.NewCxList(
+   vars.GetByAncs(Find_StatementCollection()),
+   classes, Find_StructDecl());
 CxList globalVars = vars - atBlockScope;
-CxList globalVarUses = All.FindAllReferences(globalVars).FindByType(typeof(UnknownReference));
+CxList globalVarUses = Find_UnknownReference().FindAllReferences(globalVars);

 foreach(CxList cur in globalVars){
    CxList curVarUses = globalVarUses.FindAllReferences(cur);

    // Check there are uses in only one (or less) function
-   if (curVarUses.GetAncOfType(typeof(MethodDecl)).Count <= 1){
+   if (curVarUses.GetAncOfType<MethodDecl>().Count <= 1){
        result.Add(cur);
    }
 }
```

CPP / CPP_MISRA_C / R13_06_Loop_Iterator_Modified_In_Loop_Body

Code changes

```
---
+++
@@ -14,7 +14,7 @@


 // find all for loops
-CxList forIterations = All.FindByType(typeof(IterationStmt));
+CxList forIterations = Find_IterationStmt();
 foreach (CxList cur in forIterations){
    IterationStmt curIter = cur.TryGetCSharpGraph<IterationStmt>();
    if(curIter != null && curIter.IterationType != null)
@@ -24,13 +24,12 @@
        }
    }
 }
-CxList forIterators = All.NewCxList();
-CxList unknownRefs = All.FindByType(typeof(UnknownReference));
+CxList unknownRefs = Find_UnknownReference();
 CxList allForsDescendants = All.GetByAncs(forIterations);

 // remove boolean instances
 // first find and remove instances of a type that is typedefd bool
-CxList typedefBools = All.FindByType(typeof(StringLiteral)).FindByName("CX_TYPEDEF").GetFathers().FindByShortName("*bool*");
+CxList typedefBools = Find_String_Literal().FindByShortName("CX_TYPEDEF").GetFathers().FindByShortName("*bool*");
 ArrayList boolTypes = new ArrayList();

 foreach(CxList cur in typedefBools){
@@ -48,10 +47,10 @@
 unknownRefs -= unknownRefs.FindAllReferences(boolDecls);

 // now find operators that receive boolean, and remove their direct inputs
-CxList binaryExprs = All.FindByType(typeof(BinaryExpr));
-CxList unaryExprs = All.FindByType(typeof(UnaryExpr));
-CxList recBoolean = binaryExprs.GetByBinaryOperator(BinaryOperator.BooleanOr) +
-   binaryExprs.GetByBinaryOperator(BinaryOperator.BooleanAnd);
+CxList binaryExprs = Find_BinaryExpr();
+CxList unaryExprs = Find_Unarys();
+CxList recBoolean = All.NewCxList(binaryExprs.GetByBinaryOperator(BinaryOperator.BooleanOr),
+   binaryExprs.GetByBinaryOperator(BinaryOperator.BooleanAnd));
 foreach (CxList cur in unaryExprs){
    UnaryOperator curOp = cur.TryGetCSharpGraph<UnaryExpr>().Operator;
    if(curOp != null)
@@ -78,7 +77,7 @@
 }

 // add postfix increment and decrement to the checked assignments
```

```
-CxList postfixExprs = All.FindByType(typeof(PostfixExpr));
+CxList postfixExprs = Find_PostfixExpr();
 CxList postIncDec = All.NewCxList();
 foreach (CxList cur in postfixExprs){


@@ -108,29 +107,22 @@
        CxList test = All.NewCxList();
        Expression t = curIteration.Test;
        if(t != null)
-       {
            test = All.FindById(t.NodeId);
-       }

        // build the set of current iteration iterators
        foreach(Statement bodyPart in bodyCol){
            if(bodyPart != null)
-           {
                body.Add(All.FindById(bodyPart.NodeId));
-           }
        }

        foreach(Statement init in initCol){
            if(init != null)
-           {
                inits.Add(All.FindById(init.NodeId));
-           }
        }
+
        foreach(Statement inc in incCol){
            if(inc != null)
-           {
                incs.Add(All.FindById(inc.NodeId));
-           }
        }

        inits = unknownRefs.GetByAncs(inits);
@@ -148,11 +140,10 @@
        // direct assigments
        result.Add(bodyRefs.FindByAssignmentSide(CxList.AssignmentSide.Left));
        // postfix/prefix increment/decrement
-       result.Add(bodyRefs * (preIncDec + postIncDec));
+       result.Add(bodyRefs * All.NewCxList(preIncDec, postIncDec));
        // passed by ref to function
-       CxList adressObjs = body.FindByType(typeof(UnaryExpr)).FindByShortName("Address");
-       result.Add(bodyRefs.FindByFathers(adressObjs.FindByFathers(body.FindByType(typeof(Param)))));
-
+       CxList adressObjs = body.FindByType<UnaryExpr>().FindByShortName("Address");
+       result.Add(bodyRefs.FindByFathers(adressObjs.FindByFathers(body.FindByType<Param>())));
```

```
    }


 }
```

CPP / CPP_MISRA_C / R14_08_Not_Compound_Switch_Or_Iteration_Statement

Code changes

```
---
+++
@@ -10,10 +10,10 @@


 */


-CxList potentials = All.FindByType(typeof(SwitchStmt)) + All.FindByType(typeof(IterationStmt));
+CxList potentials = All.NewCxList(Find_SwitchStmt(), Find_IterationStmt());


 // Remove iterations followed by compound statements
-CxList directSC = All.FindByType(typeof(StatementCollection)).FindByFathers(potentials);
+CxList directSC = Find_StatementCollection().FindByFathers(potentials);
 CxList allCompound = directSC.FindByRegex("{");


 allCompound -= (allCompound - directSC).GetByAncs(directSC);
```

CPP / CPP_MISRA_C / R14_09_Not_Compound_If_Or_Else

Code changes

```
---
+++
@@ -12,7 +12,7 @@
 */


 //find all if-else statements
-CxList ifs = All.FindByType(typeof(IfStmt));
+CxList ifs = Find_Ifs();
 CxList elseStatements = All.NewCxList();


 // find else statements with no statement collection and are not ifelse statements
@@ -21,7 +21,7 @@
         StatementCollection falseStmts = cur.TryGetCSharpGraph<IfStmt>().FalseStatements;
         if (falseStmts != null && falseStmts.Count == 1){
             CxList tempNode = All.FindById(falseStmts[0].NodeId);
-            if (falseStmts.Father.ToString().Equals("N/A") && tempNode.FindByType(typeof(IfStmt)).Count == 0 )
+            if (falseStmts.Father.ToString().Equals("N/A") && tempNode.FindByType<IfStmt>().Count == 0 )
             {
                 elseStatements.Add(tempNode);
             }
@@ -34,7 +34,7 @@



 // remove ifs with a non empty compound statements
```

```
-ifs -= All.FindByType(typeof(StatementCollection)).GetFathers();
+ifs -= Find_StatementCollection().GetFathers();

 // remove ifs with an empty compound statement
 foreach(CxList cur in ifs){
@@ -50,4 +50,4 @@
     }
 }

-result = ifs + elseStatements;
+result.Add(ifs, elseStatements);
```

CPP / CPP_MISRA_C / R14_10_If_Else_If_Not_Ending_With_Else

Code changes

```
---
+++
@@ -15,7 +15,7 @@

 */

-CxList ifs = All.FindByType(typeof(IfStmt));
+CxList ifs = Find_Ifs();
 CxList elseIfs = All.NewCxList();

 // find else if statements
@@ -28,7 +28,7 @@
         }
     }
 }
-elseIfs -= elseIfs.FindByFathers(All.FindByType(typeof(StatementCollection)));
+elseIfs -= elseIfs.FindByFathers(Find_StatementCollection());

 // remove else ifs that have an else with a non comment statement
 foreach(CxList cur in elseIfs){
```

CPP / CPP_MISRA_C / R16_03_Function_Prototype_Without_Identifiers

Code changes

```
---
+++
@@ -9,14 +9,14 @@

 */

-CxList methodDefs = All.FindByType(typeof(StatementCollection)).GetFathers().FindByType(typeof(MethodDecl));
-CxList methodPrototypes = All.FindByType(typeof(MethodDecl)) - methodDefs;
-CxList typeRef = All.FindByType(typeof(TypeRef));
-CxList emptyTypeParamaters = typeRef.FindByShortName("").GetAncOfType(typeof(ParamDecl));
-CxList voidTypeParamaters = typeRef.FindByShortName("void").GetAncOfType(typeof(ParamDecl));
```

```
+CxList methodDefs = Find_StatementCollection().GetFathers().FindByType<MethodDecl>();
+CxList methodPrototypes = Find_MethodDecls() - methodDefs;
+CxList typeRef = Find_TypeRef();
+CxList emptyTypeParamaters = typeRef.FindByShortName("").GetAncOfType<ParamDecl>();
+CxList voidTypeParamaters = typeRef.FindByShortName("void").GetAncOfType<ParamDecl>();

 // we only concerned with prototype (not definition) paramaters
-CxList emptyIdentParamaters = All.FindByType(typeof(ParamDecl)).FindByShortName("").
+CxList emptyIdentParamaters = Find_ParamDecl().FindByShortName("").
    GetByAncs(methodPrototypes);

 // Almost all paramaters with empty name are empty identifiers and should be addes
```

CPP / CPP_MISRA_C / R16_05_Function_Prototype_Declaration_Without_Parameters

Code changes

```
---
+++
@@ -14,13 +14,13 @@
 */

 // Find all function declarations
-CxList methodDeclsAndDefs = All.FindByType(typeof(MethodDecl));
-CxList methodDefs = All.FindByType(typeof(StatementCollection)).GetFathers().FindByType(typeof(MethodDecl));
+CxList methodDeclsAndDefs = Find_MethodDecls();
+CxList methodDefs = Find_StatementCollection().GetFathers().FindByType<MethodDecl>();
 CxList methodDecls = methodDeclsAndDefs - methodDefs;

 // Find those without parameter
 CxList parameters = All.GetParameters(methodDecls);
-CxList methodDeclsWithParams = parameters.GetAncOfType(typeof(MethodDecl));
+CxList methodDeclsWithParams = parameters.GetAncOfType<MethodDecl>();
 CxList methodDeclsWithoutParams = methodDecls - methodDeclsWithParams;

 /*
```

CPP / CPP_MISRA_C / R16_07_Parameter_Pointer_To_Const_Where_Not_Modified

Code changes

```
---
+++
@@ -13,12 +13,14 @@
 }

 */
+CxList unarys = Find_Unarys();
+CxList parameters = Find_Param();

 // ignore function declarations and main
-CxList functionDefinitions = All.FindByType(typeof(StatementCollection)).GetFathers().FindByType(typeof(MethodDecl));
```

```
+CxList functionDefinitions = Find_StatementCollection().GetFathers().FindByType<MethodDecl>();
 functionDefinitions -= functionDefinitions.FindByShortName("main");
 // start with all pointer paramaters
-CxList definitionParams = All.FindByType(typeof(ParamDecl)).FindByFathers(All.FindByFathers(functionDefinitions));
+CxList definitionParams = Find_ParamDecl().FindByFathers(All.FindByFathers(functionDefinitions));
 CxList pointerParams = definitionParams.FindByRegex(@"\w\s*?\*",false,false,false);

 // remove pointers to const
@@ -26,21 +28,22 @@

 // now only keep those whose data is changed
 CxList changedPointerDataRef = All.NewCxList();
-CxList pointerParamDataRef = All.FindAllReferences(pointerParams).GetFathers().FindByType(typeof(UnaryExpr)).FindByName("Pointer");
+CxList pointerParamDataRef = All.FindAllReferences(pointerParams).GetFathers()
+   .FindByType<UnaryExpr>().FindByShortName("Pointer");

 // data ref direct assigments
 changedPointerDataRef.Add(pointerParamDataRef.FindByAssignmentSide(CxList.AssignmentSide.Left));

 // data ref postfix/prefix increment/decrement
 CxList preIncDec = All.NewCxList();
-foreach (CxList cur in All.FindByType(typeof(UnaryExpr))){
+foreach (CxList cur in unarys){
    UnaryOperator curOp = cur.TryGetCSharpGraph<UnaryExpr>().Operator;
    if ((curOp == UnaryOperator.Increment) || (curOp == UnaryOperator.Decrement)){
        preIncDec.Add(All.FindById(cur.TryGetCSharpGraph<UnaryExpr>().Right.NodeId));
    }
 }
 CxList postIncDec = All.NewCxList();
-foreach (CxList cur in All.FindByType(typeof(PostfixExpr))){
+foreach (CxList cur in Find_PostfixExpr()){
    PostfixOperator curOp = cur.TryGetCSharpGraph<PostfixExpr>().Operator;
    if ((curOp == PostfixOperator.Increment) || (curOp == PostfixOperator.Decrement)){
        if (cur.TryGetCSharpGraph<PostfixExpr>().Left != null){
@@ -49,22 +52,22 @@
    }
 }

-// note cahnged paramaters
-changedPointerDataRef.Add(pointerParamDataRef * (preIncDec + postIncDec));
-changedPointerDataRef.Add(pointerParamDataRef * (preIncDec + postIncDec).GetFathers());
+// note changed paramaters
+CxList helper = pointerParamDataRef * All.NewCxList(preIncDec, postIncDec);
+changedPointerDataRef.Add(helper, helper.GetFathers());

 // data ref passed by ref to function
-CxList adressObjs = All.FindByType(typeof(UnaryExpr)).FindByShortName("Address");
-changedPointerDataRef.Add(pointerParamDataRef.FindByFathers(adressObjs.FindByFathers(All.FindByType(typeof(Param)))));
```

```
+CxList adressObjs = unarys.FindByShortName("Address");
+changedPointerDataRef.Add(pointerParamDataRef.FindByFathers(adressObjs.FindByFathers(parameters)));

 // remove all param decls who have an instance changed
 pointerParams -= pointerParams.FindDefinition(All.FindByFathers(changedPointerDataRef));

 // param itself passed by value to function
-pointerParams -= pointerParams.FindDefinition(All.FindAllReferences(pointerParams).FindByFathers(All.FindByType(typeof(Param))));
+pointerParams -= pointerParams.FindDefinition(All.FindAllReferences(pointerParams).FindByFathers(parameters));

 // remove changes through array access
-CxList arrays = All.FindByType(typeof(IndexerRef)).FindAllReferences(pointerParams);
+CxList arrays = Find_IndexerRefs().FindAllReferences(pointerParams);
 arrays = arrays.FindByAssignmentSide(CxList.AssignmentSide.Left);
 arrays.Add(All.FindByName(arrays).FindByFathers(arrays));
 pointerParams -= pointerParams.FindDefinition(arrays);
```

CPP / CPP_MISRA_C / R16_08_Non_Explicit_Return_Statement_In_Non_Void_Function

Code changes

```
---
+++
@@ -16,20 +16,21 @@
 */
 //finds all methods that have non void return type but return void.
-CxList returnStmt = All.FindByType(typeof(ReturnStmt));
+CxList returnStmt = Find_ReturnStmt();
 CxList emptyReturn = returnStmt - All.FindByFathers(returnStmt).GetFathers();
-CxList returnType = All.FindByType(typeof(TypeRef)).FindByFathers(emptyReturn.GetAncOfType(typeof(MethodDecl)));
+CxList typeRefs = Find_TypeRef();
+CxList returnType = typeRefs.FindByFathers(emptyReturn.GetAncOfType<MethodDecl>());
 CxList voidRT = returnType.FindByShortName("void");
 CxList nonVoidRT = returnType - voidRT;
 result = nonVoidRT;



 //finds methods that have non void return type but don't have return statements:
-CxList allMethodDecl = All.FindByType(typeof(MethodDecl));
+CxList allMethodDecl = Find_MethodDecls();
 //remove all definitions
-CxList declared = All.FindByFathers(allMethodDecl).FindByType(typeof(StatementCollection)).GetFathers();
+CxList declared = All.FindByFathers(allMethodDecl).FindByType<StatementCollection>().GetFathers();
 //get non those who have non void return types:
-returnType = All.FindByType(typeof(TypeRef)).FindByFathers(declared);
+returnType = typeRefs.FindByFathers(declared);
 nonVoidRT = returnType - returnType.FindByShortName("void");
 CxList backToMeth = nonVoidRT.GetFathers();
```

```
@@ -37,8 +38,6 @@
 {
     CxList foundRetStmt = returnStmt.GetByAncs(method);
     if(foundRetStmt.Count == 0)
-    {
         result.Add(method);
-    }
 }
 result -= allMethodDecl.FindByShortName("main");
```

CPP / CPP_MISRA_CPP / R02_10_02_Identifiers_Hide_Outer_Scope_Identifiers

Code changes

```
---
+++
@@ -16,16 +16,16 @@

 CxList identifiers = Find_Identifiers();

-// remove extern delcarations since by definition they do not "declare" an identifier, just repeat it
+// remove extern declarations since by definition they do not "declare" an identifier, just repeat it
 identifiers -= identifiers.GetByAncs(All.FindByFieldAttributes(Dom.Modifiers.Extern));
 // remove gotos
 CxList gotos = All.FindByFathers(Find_All_Declarators());
-gotos = gotos.FindByName("goto").GetFathers();
+gotos = gotos.FindByShortName("goto").GetFathers();
 identifiers -= gotos;

 // remove method declaration if we have the definition
-CxList methodDeclAndDefs = identifiers.FindByType(typeof(MethodDecl));
-CxList methodDefs = methodDeclAndDefs * All.FindByType(typeof(StatementCollection)).GetFathers();
+CxList methodDeclAndDefs = identifiers.FindByType<MethodDecl>();
+CxList methodDefs = methodDeclAndDefs * Find_StatementCollection().GetFathers();
 CxList methodDecls = methodDeclAndDefs - methodDefs;
 CxList doubleMethods = identifiers.GetByAncs(methodDecls.FindByShortName(methodDefs));
 identifiers -= doubleMethods;
@@ -54,18 +54,18 @@
     if(IdentAppearances.Count > 0)
     {
         // an object is in larger or equal scope of another object if and only if its direct scope contains the other object
-        CxList curScope = curIdentifier.GetAncOfType(typeof(StatementCollection));
+        CxList curScope = curIdentifier.GetAncOfType<StatementCollection>();
         if (curScope.Count == 0){
-            curScope = curIdentifier.GetAncOfType(typeof(ParamDeclCollection));
+            curScope = curIdentifier.GetAncOfType<ParamDeclCollection>();
         }
         if (curScope.Count == 0){
-            curScope = curIdentifier.GetAncOfType(typeof(MemberDeclCollection));
+            curScope = curIdentifier.GetAncOfType<MemberDeclCollection>();
```

```
        }
        if (curScope.Count == 0){
-           curScope = curIdentifier.GetAncOfType(typeof(MethodDecl));
+           curScope = curIdentifier.GetAncOfType<MethodDecl>();
        }
        if (curScope.Count == 0){
-           curScope = curIdentifier.GetAncOfType(typeof(NamespaceDecl));
+           curScope = curIdentifier.GetAncOfType<NamespaceDecl>();
        }

        // find same name identifiers that have inner/equal scope
@@ -73,18 +73,18 @@

        // only add those in strictly inner scope
        foreach(CxList cur in identAppearancesInCurScope){
-           CxList curTestScope = cur.GetAncOfType(typeof(StatementCollection));
+           CxList curTestScope = cur.GetAncOfType<StatementCollection>();
            if (curTestScope.Count == 0){
-               curTestScope = cur.GetAncOfType(typeof(ParamDeclCollection));
+               curTestScope = cur.GetAncOfType<ParamDeclCollection>();
            }
            if (curTestScope.Count == 0){
-               curTestScope = cur.GetAncOfType(typeof(MemberDeclCollection));
+               curTestScope = cur.GetAncOfType<MemberDeclCollection>();
            }
            if (curTestScope.Count == 0){
-               curTestScope = cur.GetAncOfType(typeof(MethodDecl));
+               curTestScope = cur.GetAncOfType<MethodDecl>();
            }
            if (curTestScope.Count == 0){
-               curTestScope = cur.GetAncOfType(typeof(NamespaceDecl));
+               curTestScope = cur.GetAncOfType<NamespaceDecl>();
            }

            if (curScope != curTestScope)
```

CPP / CPP_MISRA_CPP / R02_10_05_Non_Member_Static_Name_Reuse

Code changes

```
---
+++
@@ -23,19 +23,18 @@
 */



+CxList allMethods = Find_MethodDecls();
+CxList allDecl = Find_All_Declarators();

-//all declarations
```

```
-CxList allDecl = Find_All_Declarators();
 //get all static declarations
 Modifiers mod = new Modifiers();
 mod = Dom.Modifiers.Static;
-CxList fd = All.FindByType(typeof(FieldDecl));
-CxList dcltr = Find_All_Declarators();
+CxList fd = Find_FieldDecls();

-CxList staticFields = All.FindByFieldAttributes(mod) - All.FindByType(typeof(MethodDecl));
-staticFields = (dcltr + fd).GetByAncs(staticFields);
+CxList staticFields = All.FindByFieldAttributes(mod) - allMethods;
+staticFields = All.NewCxList(allDecl, fd).GetByAncs(staticFields);

-CxList classDecl = All.FindByType(typeof(ClassDecl));
+CxList classDecl = Find_ClassDecl();
 //get non- memebers declarations
 foreach(CxList cur in staticFields)
 {
@@ -46,7 +45,6 @@
        staticFields -= cur;
    }
 }
-
```

```
 //compare names of all declarations

@@ -64,11 +62,8 @@
    }
 }

-
-//for all non- members functions
-CxList allMethods = All.FindByType(typeof(MethodDecl));
 //get all non-members
-CxList stmtCol = All.FindByType(typeof(StatementCollection));
+CxList stmtCol = Find_StatementCollection();
 foreach(CxList cur in allMethods)
 {
    CxList myClass = classDecl.GetClass(cur);
@@ -79,7 +74,7 @@
    }
 }
 //get all static method declarations
-CxList staticMethods = allMethods.FindByType(typeof(MethodDecl));
+CxList staticMethods = allMethods.FindByType<MethodDecl>();
 staticMethods = staticMethods.FindByFieldAttributes(mod);

 //compare their names
```

CPP / CPP_MISRA_CPP / R03_02_01_Identical_Function_and_Object_Decl_Def

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 /*
 MISRA CPP RULE 3-2-1
------------------------------
+//////////////////////////////
 This query searches for decleration/definitions of functions which differ in the return type or paramater types
 and of other objects.

@@ -14,21 +14,22 @@

 */

-CxList methDecls = All.FindByType(typeof(MethodDecl));
-CxList typeRef = All.FindByType(typeof(TypeRef));
-CxList parameters = All.FindByType(typeof(ParamDecl));
+CxList methDecls = Find_MethodDecls();
+CxList allTypeRefs = Find_TypeRef();
+CxList parameters = Find_ParamDecl();
 CxList voidType = All.FindByShortName("void");
 //get all method definitions:
-CxList methDef = methDecls - All.FindByType(typeof(StatementCollection)).FindByFathers(methDecls).GetFathers();
+CxList methDef = methDecls - Find_StatementCollection().FindByFathers(methDecls).GetFathers();
+
 CxList allMethodsWithSameName = methDecls.FindByShortName(methDef);
 CxList allMdParams = parameters.GetParameters(methDef);

 CxList allMwsnParams = allMdParams.GetParameters(allMethodsWithSameName);
-CxList allMdReturnType = typeRef.FindByFathers(methDef);
-CxList allMwsnReturnType = typeRef.FindByFathers(allMethodsWithSameName);
-CxList mwsnTypeRef = typeRef.FindByFathers(allMwsnParams);
+CxList allMdReturnType = allTypeRefs.FindByFathers(methDef);
+CxList allMwsnReturnType = allTypeRefs.FindByFathers(allMethodsWithSameName);
+CxList mwsnTypeRef = allTypeRefs.FindByFathers(allMwsnParams);

-typeRef = typeRef.FindByFathers(allMdParams);
+CxList typeRef = allTypeRefs.FindByFathers(allMdParams);

 CxList updatedTR = typeRef.FindByFathers(allMdParams.GetParameters(methDef));
 CxList rtype = allMdReturnType.FindByFathers(methDef);
@@ -50,9 +51,9 @@

         if(rtoo.ShortName != rt.ShortName)
         {
-            result.Add(mwsn + md);
```

```
+           result.Add(mwsn, md);
            nonCompliant.Add(mwsn);
-           allMethodsWithSameName -= mwsn + md;
+           allMethodsWithSameName -= All.NewCxList(mwsn, md);
        }
    }
    methodsWithSameName -= nonCompliant;
@@ -61,8 +62,8 @@
        CxList otherParamTypes = mwsnTypeRef.GetByAncs(allMwsnParams.GetParameters(mwsn));
        if(otherParamTypes.Count != ParamTypes.Count)
        {
-           result.Add(mwsn + md);
-           allMethodsWithSameName -= mwsn + md;
+           result.Add(mwsn, md);
+           allMethodsWithSameName -= All.NewCxList(mwsn, md);
            continue;
        }
        if(otherParamTypes.Count == 0 && ParamTypes.Count == 0)
@@ -76,23 +77,22 @@
            continue;
        }

-       CxList defParamColl = ParamTypes.GetAncOfType(typeof(ParamDeclCollection));
-       CxList declParamColl = otherParamTypes.GetAncOfType(typeof(ParamDeclCollection));
+       CxList defParamColl = ParamTypes.GetAncOfType<ParamDeclCollection>();
        for (int i = 0; i < defParamColl.Count; i++)
        {
-           CSharpGraph def = ParamTypes.data.GetByIndex(i) as CSharpGraph;
-           CSharpGraph decl = otherParamTypes.data.GetByIndex(i) as CSharpGraph;
+           CSharpGraph def = ParamTypes.ElementAt(i).TryGetCSharpGraph<CSharpGraph>();
+           CSharpGraph decl = otherParamTypes.ElementAt(i).TryGetCSharpGraph<CSharpGraph>();
            if(def.ShortName != decl.ShortName)
            {
-               result.Add(mwsn + md);
-               allMethodsWithSameName -= mwsn + md;
+               result.Add(mwsn, md);
+               allMethodsWithSameName -= All.NewCxList(mwsn, md);
            }
        }
    }
 }

 CxList decls = Find_All_Declarators();
-typeRef = All.FindByType(typeof(TypeRef)).GetByAncs(decls);
+typeRef = allTypeRefs.GetByAncs(decls);
 CxList cltrs = typeRef.GetByAncs(decls);
 // go over all declarators with a name that appears twice
 foreach (CxList decl in decls) {
```

```
@@ -103,8 +103,8 @@
    foreach(CxList compDecl in sameNameDecls) {
        CxList compType = sameTypeRef.GetByAncs(compDecl);
        if (compType.FindByName(curType).Count == 0) {
-           result.Add(sameNameDecls + decl);
-           decls -= sameNameDecls + decl;
+           result.Add(sameNameDecls, decl);
+           decls -= All.NewCxList(sameNameDecls, decl);
            typeRef -= sameTypeRef;
            break;
        }
```

CPP / CPP_MISRA_CPP / R03_04_01_Obj_Defined_Outside_Minimal_Scope

Code changes

```
---
+++
@@ -16,18 +16,18 @@
 */


 CxList decls = Find_All_Declarators();
-CxList unknownRefs = All.FindByType(typeof(UnknownReference)).FindAllReferences(decls);
+CxList unknownRefs = Find_UnknownReference().FindAllReferences(decls);
 decls = All.FindDefinition(unknownRefs);


 string oldfile = "";
 CxList oldUnknown = All.NewCxList();


 foreach (CxList decl in decls) {
-   CxList scope = decl.GetAncOfType(typeof(StatementCollection));
+   CxList scope = decl.GetAncOfType<StatementCollection>();
    if (scope.Count == 0){
-       scope = decl.GetAncOfType(typeof(ClassDecl));
+       scope = decl.GetAncOfType<ClassDecl>();
        if (scope.Count == 0){
-           scope = decl.GetAncOfType(typeof(StructDecl));
+           scope = decl.GetAncOfType<StructDecl>();
        }
    }
    string declFile = decl.GetFirstGraph().LinePragma.FileName;
@@ -44,18 +44,17 @@
    CxList firstRef = All.NewCxList();
    CxList refScope = All.NewCxList();
    while (refScope.Count == 0 && counter < refs.Count){
-       firstRef = All.FindById(((CSharpGraph) refs.data.GetByIndex(counter)).NodeId);
+       firstRef = All.FindById(refs.ElementAt(counter).TryGetCSharpGraph<CSharpGraph>().NodeId);
        counter++;
-       refScope = firstRef.GetAncOfType(typeof(StatementCollection));
+       refScope = firstRef.GetAncOfType<StatementCollection>();
```

```
        if (refScope.Count == 0){
-           refScope = firstRef.GetAncOfType(typeof(ClassDecl));
+           refScope = firstRef.GetAncOfType<ClassDecl>();
            if (refScope.Count == 0){
-               refScope = firstRef.GetAncOfType(typeof(StructDecl));
+               refScope = firstRef.GetAncOfType<StructDecl>();
            }
        }
    }
    while (scope != refScope && scope.FindByShortName(refScope).Count == 0 && refScope.Count != 0 ) {
-       CxList oldref = refScope;
        refs -= refs.GetByAncs(refScope);//Remove all refs under current refScope
        if (refs.Count == 0) {
            result.Add(decl);
@@ -63,11 +62,11 @@
        }
        //Find refScope's scope
        firstRef = refScope.GetFathers();
-       refScope = firstRef.GetAncOfType(typeof(StatementCollection));
+       refScope = firstRef.GetAncOfType<StatementCollection>();
        if (refScope.Count == 0){
-           refScope = firstRef.GetAncOfType(typeof(ClassDecl));
+           refScope = firstRef.GetAncOfType<ClassDecl>();
            if (refScope.Count == 0){
-               refScope = firstRef.GetAncOfType(typeof(StructDecl));
+               refScope = firstRef.GetAncOfType<StructDecl>();
            }
        }
```

CPP / CPP_MISRA_CPP / R06_03_01_Not_Compound_Switch_Or_Iteration_Statement

Code changes

```
---
+++
@@ -10,10 +10,11 @@

 */

-CxList potentials = All.FindByType(typeof(SwitchStmt)) + All.FindByType(typeof(IterationStmt));
+CxList potentials = All.NewCxList(
+   Find_SwitchStmt(), Find_IterationStmt());

 // Remove iterations followed by compound statements
-CxList directSC = All.FindByType(typeof(StatementCollection)).FindByFathers(potentials);
+CxList directSC = Find_StatementCollection().FindByFathers(potentials);

 CxList allCompound = directSC.FindByRegex("{");
 allCompound -= (allCompound - directSC).GetByAncs(directSC);
```

CPP / CPP_MISRA_CPP / R06_04_01_Not_Compound_If_Or_Else

Code changes

```
---
+++
@@ -18,7 +18,7 @@
 */


 //find all if-else statements
-CxList ifs = All.FindByType(typeof(IfStmt));
+CxList ifs = Find_Ifs();
 CxList elseStatements = All.NewCxList();


 // find else statements with no statement collection and are not ifelse statements
@@ -27,7 +27,7 @@
        StatementCollection falseStmts = cur.TryGetCSharpGraph<IfStmt>().FalseStatements;
        if (falseStmts != null && falseStmts.Count == 1){
            CxList tempNode = All.FindById(falseStmts[0].NodeId);
-           if (falseStmts.Father.ToString().Equals("N/A") && tempNode.FindByType(typeof(IfStmt)).Count == 0 )
+           if (falseStmts.Father.ToString().Equals("N/A") && tempNode.FindByType<IfStmt>().Count == 0 )
            {
                elseStatements.Add(tempNode);
            }
@@ -40,7 +40,7 @@


 // remove ifs with a non empty compound statements
-ifs -= All.FindByType(typeof(StatementCollection)).GetFathers();
+ifs -= Find_StatementCollection().GetFathers();


 // remove ifs with an empty compound statement
 foreach(CxList cur in ifs){
@@ -56,4 +56,4 @@
    }
 }


-result =  ifs + elseStatements;
+result.Add(ifs, elseStatements);
```

CPP / CPP_MISRA_CPP / R06_04_02_If_Else_If_Not_Ending_With_Else

Code changes

```
---
+++
@@ -15,7 +15,7 @@


 */


-CxList ifs = All.FindByType(typeof(IfStmt));
+CxList ifs = Find_Ifs();
```

```
 CxList elseIfs = All.NewCxList();


 // find else if statements
@@ -28,7 +28,7 @@
        }
    }
 }
-elseIfs -= elseIfs.FindByFathers(All.FindByType(typeof(StatementCollection)));
+elseIfs -= elseIfs.FindByFathers(Find_StatementCollection());


 // remove else ifs that have an else with a non comment statement
 foreach(CxList cur in elseIfs){
```

CPP / CPP_MISRA_CPP / R06_05_01_Single_Non_Float_LC

Code changes

```
---
+++
@@ -10,39 +10,23 @@
 */



+CxList fieldDecls = Find_FieldDecls();
+CxList allFors = Find_ForStatement();

-CxList allFors = All.FindByType(typeof(IterationStmt));
-
-CxList helper = allFors;
-foreach(CxList allf in allFors)
-{
-   IterationStmt i = allf.TryGetCSharpGraph<IterationStmt>();
-   if(i != null)
-   {
-       IterationType it = i.IterationType;
-       if(!it.ToString().Equals("For"))
-           helper -= allf;
-   }
-}
-allFors = helper;
-
-//CxList totalUnknownR = All.FindByType(typeof(UnknownReference)) + Find_All_Declarators() + All.FindByType(typeof(ParamDecl));
-CxList unrf = All.FindByType(typeof(UnknownReference)).GetByAncs(allFors);
-CxList totalDecl = Find_All_Declarators() + All.FindByType(typeof(FieldDecl)) + All.FindByType(typeof(ParamDecl));
+CxList unrf = Find_UnknownReference().GetByAncs(allFors);
+CxList totalDecl = All.NewCxList(Find_All_Declarators(), fieldDecls, Find_ParamDecl());
 CxList declarators = totalDecl.GetByAncs(allFors);
 CxList leftSd = All.FindByAssignmentSide(CxList.AssignmentSide.Left);


 CxList notRef = unrf.GetMembersOfTarget().GetTargetOfMembers();
```

```
 unrf -= notRef;


-

-

 //find float typedefs

-CxList typedefs = All.FindByName("CX_TYPEDEF").FindByType(typeof(StringLiteral));

-typedefs = typedefs.GetAncOfType(typeof(VariableDeclStmt))

-   + typedefs.GetAncOfType(typeof(FieldDecl));

-CxList tpr = All.FindByType(typeof(TypeRef));

-typedefs.Add(totalDecl.GetByAncs(typedefs) * All.FindByType(typeof(FieldDecl)));

+CxList typedefs = All.FindByShortName("CX_TYPEDEF").FindByType<StringLiteral>();

+typedefs = All.NewCxList(typedefs.GetAncOfType<VariableDeclStmt>(),

+    typedefs.GetAncOfType<FieldDecl>());

+CxList tpr = Find_TypeRef();

+typedefs.Add(totalDecl.GetByAncs(typedefs) * fieldDecls);



 CxList alternativeFloats = All.NewCxList();

@@ -62,12 +46,7 @@

 alternativeFloats = totalDecl.GetByAncs(alternativeFloats.GetFathers());



-

-

-

-

 //find the init part of the for statement

-

 foreach(CxList cur in allFors)

 {

    CxList init = All.NewCxList();

@@ -82,11 +61,10 @@

    }

    CxList unknownRef = unrf.GetByAncs(init);

    //retrieves the loop counters

-   CxList loopCounter = unknownRef.FindByFathers(init.FindByType(typeof(ExprStmt)));

+   CxList loopCounter = unknownRef.FindByFathers(init.FindByType<ExprStmt>());


    CxList leftAsn = unknownRef * leftSd;

-   loopCounter.Add(leftAsn + declarators.GetByAncs(init));

-

+   loopCounter.Add(leftAsn, declarators.GetByAncs(init));


    CxList increment = All.NewCxList();

    StatementCollection incrementColl = iterA.Increment;

@@ -108,7 +86,7 @@


    CxList testUn = unrf.GetByAncs(test);
```

```
    CxList incrUn = unrf.GetByAncs(increment);
-   CxList additionalLC = incrUn.FindAllReferences(testUn) + testUn.FindAllReferences(incrUn);
+   CxList additionalLC = All.NewCxList(incrUn.FindAllReferences(testUn), testUn.FindAllReferences(incrUn));
    CxList temp = loopCounter.FindAllReferences(additionalLC);
    additionalLC -= additionalLC.FindAllReferences(temp);
```

CPP / CPP_MISRA_CPP / R06_05_02_Loop_Counter_Modify

Code changes

```
---
+++
@@ -12,39 +12,12 @@
 */

 //finds all for statements
-CxList allFors = All.FindByType(typeof(IterationStmt));
-CxList helper = allFors;
-foreach(CxList allf in allFors)
-{
-
-   IterationStmt i = allf.TryGetCSharpGraph<IterationStmt>();
-   if(i != null)
-   {
-       IterationType it = i.IterationType;
-       if(!it.ToString().Equals("For"))
-       {
-           helper -= allf;
-       }
-   }
-
-}
-allFors = helper;
-
-CxList totalUr = All.FindByType(typeof(UnknownReference));
-CxList unrf = totalUr.GetByAncs(allFors);
-CxList totalUn = All.FindByType(typeof(UnaryExpr));
-CxList unaryEx = totalUn.GetByAncs(allFors);
-CxList totalPf = All.FindByType(typeof(PostfixExpr));
-CxList totalEx = totalPf.GetByAncs(allFors);
-CxList totalExpr = All.FindByType(typeof(ExprStmt));
-CxList expr = totalExpr.GetByAncs(allFors);
-CxList assignEx = All.FindByType(typeof(AssignExpr)).GetByAncs(allFors);
-CxList methInvEx = All.FindByType(typeof(MethodInvokeExpr)).GetByAncs(allFors);
-CxList bo = All.GetByBinaryOperator(BinaryOperator.IdentityEquality) +
-   All.GetByBinaryOperator(BinaryOperator.IdentityInequality);
-CxList leftSide = All.FindByAssignmentSide(CxList.AssignmentSide.Left);
-
-CxList declarators = Find_All_Declarators().GetByAncs(allFors);
```

```
+CxList allFors = Find_ForStatement();
+CxList unrf = Find_UnknownReference().GetByAncs(allFors);
+CxList assignEx = Find_AssignExpr().GetByAncs(allFors);
+CxList methInvEx = Find_Methods().GetByAncs(allFors);
+CxList bo = All.NewCxList(All.GetByBinaryOperator(BinaryOperator.IdentityEquality),
+   All.GetByBinaryOperator(BinaryOperator.IdentityInequality));

 //finds all increment statements inside the for
 foreach(CxList cur in allFors)
@@ -59,11 +32,12 @@
             increment.Add(All.FindById(incrementColl[i].NodeId));
         }
     }
-   CxList illegalExpr = assignEx.GetByAncs(increment) + methInvEx.GetByAncs(increment);
+   CxList illegalExpr = All.NewCxList(assignEx.GetByAncs(increment),
+       methInvEx.GetByAncs(increment));
     CxList refInIncr = unrf.GetByAncs(illegalExpr);
     CxList left = refInIncr.FindByAssignmentSide(CxList.AssignmentSide.Left);
     CxList prm = refInIncr.GetParameters(methInvEx);
-   CxList toCheck = prm + left;
+   CxList illegRef = All.NewCxList(prm, left);

     //get's for this "for" statement its "test" statement
     Expression exp = iterS.Test;
@@ -72,17 +46,19 @@
         test.Add(All.FindById(exp.NodeId));
     }
     //checks the operators
-   CxList binary = test.FindByType(typeof(BinaryExpr));
-   CxList legal = binary.GetByBinaryOperator(BinaryOperator.LessThan) +
-       binary.GetByBinaryOperator(BinaryOperator.GreaterThan) +
-       binary.GetByBinaryOperator(BinaryOperator.LessThanOrEqual) +
-       binary.GetByBinaryOperator(BinaryOperator.GreaterThanOrEqual);
+   CxList binary = test.FindByType<BinaryExpr>();
+   CxList legal = All.NewCxList(
+       binary.GetByBinaryOperator(BinaryOperator.LessThan),
+       binary.GetByBinaryOperator(BinaryOperator.GreaterThan),
+       binary.GetByBinaryOperator(BinaryOperator.LessThanOrEqual),
+       binary.GetByBinaryOperator(BinaryOperator.GreaterThanOrEqual));
     CxList foundIllegal = binary - legal;
     //handles the case that there is or or and in the test
-   CxList special = foundIllegal.GetByBinaryOperator(BinaryOperator.BooleanOr) +
-       foundIllegal.GetByBinaryOperator(BinaryOperator.BooleanAnd) +
-       foundIllegal.GetByBinaryOperator(BinaryOperator.BitwiseOr) +
-       foundIllegal.GetByBinaryOperator(BinaryOperator.BitwiseAnd);
+   CxList special = All.NewCxList(
+       foundIllegal.GetByBinaryOperator(BinaryOperator.BooleanOr),
+       foundIllegal.GetByBinaryOperator(BinaryOperator.BooleanAnd),
```

```
+       foundIllegal.GetByBinaryOperator(BinaryOperator.BitwiseOr),
+       foundIllegal.GetByBinaryOperator(BinaryOperator.BitwiseAnd));
    CxList illegal = foundIllegal - special;
    //retrieves the operators from the complex expression
    CxList specialAnc = bo.GetByAncs(special);
@@ -93,28 +69,6 @@
    CxList testRef = unrf.GetByAncs(illegal);
    testRef.Add(methInvEx.GetByAncs(test).GetTargetOfMembers());
    testRef.Add(unrf.GetByAncs(methInvEx.GetByAncs(test)));
-   CxList illegRef = toCheck;
-   CxList isLC = All.NewCxList();

-   isLC = testRef.FindAllReferences(illegRef);
-
-   CxList init = All.NewCxList();
-   StatementCollection initColl = iterS.Init;
-   for(int i = 0; initColl != null && i < initColl.Count; i++)
-   {
-       if(initColl[i] != null)
-       {
-           init.Add(All.FindById(initColl[i].NodeId));
-       }
-   }
-
-   //retrieves the loop counters
-   CxList loopCounter = unrf.FindByFathers(init.FindByType(typeof(ExprStmt)));
-
-   CxList unknownRef = unrf.GetByAncs(init);
-
-   CxList leftAsn = unknownRef * leftSide;
-
-   loopCounter.Add(leftAsn + declarators.GetByAncs(init));
-   result.Add(isLC);
+   result.Add(testRef.FindAllReferences(illegRef));
 }
```

CPP / CPP_MISRA_CPP / R06_05_03_Change_Lc_In_St_And_Cond

Code changes

```
---
+++
@@ -15,30 +15,14 @@
     }
 */

-   CxList allFors = All.FindByType(typeof(IterationStmt));
-   CxList helper = allFors;
-   foreach(CxList allf in allFors)
-   {
```

```
-       IterationStmt i = allf.TryGetCSharpGraph<IterationStmt>();
-       if(i != null)
-       {
-           IterationType it = i.IterationType;
-           if(!it.ToString().Equals("For"))
-           {
-               helper -= allf;
-           }
-       }
-   }
-   allFors = helper;
-
-   CxList postfix = All.FindByType(typeof(PostfixExpr)).GetByAncs(allFors);
-   CxList unrf = All.FindByType(typeof(UnknownReference)).GetByAncs(allFors);
-   CxList declarators = Find_All_Declarators().GetByAncs(allFors);
-   CxList leftSide=All.FindByAssignmentSide(CxList.AssignmentSide.Left);
+CxList allFors = Find_ForStatement();
+CxList postfix = Find_PostfixExpr().GetByAncs(allFors);
+CxList unrf = Find_UnknownReference().GetByAncs(allFors);
+CxList declarators = Find_All_Declarators().GetByAncs(allFors);
+CxList leftSide = All.FindByAssignmentSide(CxList.AssignmentSide.Left);

 foreach(CxList cur in allFors)
 {
-
     CxList init = All.NewCxList();
     IterationStmt iterA = cur.TryGetCSharpGraph<IterationStmt>();
     StatementCollection initColl = iterA.Init;
@@ -51,11 +35,11 @@
         }
     }

-   CxList loopCounter = unrf.FindByFathers(init.FindByType(typeof(ExprStmt)));
+   CxList loopCounter = unrf.FindByFathers(init.FindByType<ExprStmt>());

     CxList unknownRef = unrf.GetByAncs(init);
     CxList leftAsn = unknownRef * leftSide;
-   loopCounter.Add(leftAsn + declarators.GetByAncs(init));
+   loopCounter.Add(leftAsn, declarators.GetByAncs(init));

     CxList allReferencesToLC = All.NewCxList();

@@ -80,35 +64,37 @@

     CxList testUn = unrf.GetByAncs(test);
     CxList incrUn = unrf.GetByAncs(increment);
-   CxList additionalLC = incrUn.FindAllReferences(testUn) + testUn.FindAllReferences(incrUn);
+   CxList additionalLC = All.NewCxList(incrUn.FindAllReferences(testUn),
```

```
+       testUn.FindAllReferences(incrUn));
    loopCounter.Add(additionalLC * testUn);

-   CxList curExpr = unrf.GetByAncs(loopCounter.GetAncOfType(typeof(IterationStmt)));
+   CxList curExpr = unrf.GetByAncs(loopCounter.GetAncOfType<IterationStmt>());
    allReferencesToLC.Add(curExpr.FindAllReferences(loopCounter));
    CxList lCinCondition = allReferencesToLC.GetByAncs(test);

    //bar(&j)
-   CxList asMethodParam = lCinCondition.GetByAncs(test).GetFathers().FindByName("Address");
-   CxList theMethod = asMethodParam.GetAncOfType(typeof(MethodInvokeExpr));
+   CxList asMethodParam = lCinCondition.GetByAncs(test).GetFathers().FindByShortName("Address");
+   CxList theMethod = asMethodParam.GetAncOfType<MethodInvokeExpr>();
    result.Add(theMethod);
-   CxList allLCRefInsideStmt = allReferencesToLC
-       - allReferencesToLC.GetByAncs(init)
-       - allReferencesToLC.GetByAncs(test)
-       - allReferencesToLC.GetByAncs(increment);
+
+   CxList toRemove = All.NewCxList( allReferencesToLC.GetByAncs(init),
+       allReferencesToLC.GetByAncs(test),
+       allReferencesToLC.GetByAncs(increment));
+   CxList allLCRefInsideStmt = allReferencesToLC - toRemove;

-   allLCRefInsideStmt = allLCRefInsideStmt - allLCRefInsideStmt.FindByType(typeof(Declarator));
+   allLCRefInsideStmt = allLCRefInsideStmt - allLCRefInsideStmt.FindByType<Declarator>();
    CxList operation = allLCRefInsideStmt.GetFathers();

    CxList assignment = allLCRefInsideStmt * leftSide;

-   assignment.Add(allLCRefInsideStmt.FindByFathers(postfix.GetByAncs(operation).GetAncOfType(typeof(AssignExpr))));
-
-   CxList change = postfix.FindByFathers(operation.FindByType(typeof(ExprStmt))) + //i++
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Increment") + //++i
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Decrement")  //--i
-       +assignment;
+   assignment.Add(allLCRefInsideStmt.FindByFathers(postfix.GetByAncs(operation).GetAncOfType<AssignExpr>()));

-   change.Add(operation.FindByType(typeof(UnaryExpr)).FindByName("Address").GetAncOfType(typeof(MethodInvokeExpr)));
+   CxList opUnaryExpr = operation.FindByType<UnaryExpr>();
+   CxList change = All.NewCxList(
+       postfix.FindByFathers(operation.FindByType<ExprStmt>()), //i++
+       opUnaryExpr.FindByShortNames("Increment", "Decrement"),  //++i --i
+       assignment);
+
+   change.Add(opUnaryExpr.FindByShortName("Address").GetAncOfType<MethodInvokeExpr>());
    result.Add(change);
-
```

```
 }
```

CPP / CPP_MISRA_CPP / R06_05_04_Incremental_Modified

Code changes

```
---
+++
@@ -22,28 +22,13 @@


 */


-CxList allFors = All.FindByType(typeof(IterationStmt));
-
-CxList helper = allFors;
-foreach(CxList allf in allFors)
-{
-   IterationStmt i = allf.TryGetCSharpGraph<IterationStmt>();
-   if(i != null){
-       IterationType it = i.IterationType;
-       if(!it.ToString().Equals("For")){
-           helper -= allf;
-       }
-   }
-}
-allFors = helper;
-
-
-CxList unrf = All.FindByType(typeof(UnknownReference)).GetByAncs(allFors);
+CxList allFors = Find_ForStatement();
+CxList unrf = Find_UnknownReference().GetByAncs(allFors);
 CxList declarators = Find_All_Declarators().GetByAncs(allFors);
 CxList leftSd = unrf.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList binEx = All.FindByType(typeof(BinaryExpr)).GetByAncs(allFors);
-CxList methInv = All.FindByType(typeof(MethodInvokeExpr)).GetByAncs(allFors);
-CxList unaryEx = All.FindByType(typeof(UnaryExpr)).GetByAncs(allFors);
+CxList binEx = Find_BinaryExpr().GetByAncs(allFors);
+CxList methInv = Find_Methods().GetByAncs(allFors);
+CxList unaryEx = Find_Unarys().GetByAncs(allFors);
 CxList rightSd = All.FindByAssignmentSide(CxList.AssignmentSide.Right);


 //find the init part of the for statement
@@ -66,10 +51,10 @@



     //retrieves the loop counters
-    CxList loopCounter = unknownRef.FindByFathers(init.FindByType(typeof(ExprStmt)));
+    CxList loopCounter = unknownRef.FindByFathers(init.FindByType<ExprStmt>());


     CxList leftAsn = unknownRef * leftSd;
```

```
-   loopCounter.Add(leftAsn + declarators.GetByAncs(init));
+   loopCounter.Add(leftAsn, declarators.GetByAncs(init));

    //retrieves the increment part of the loop
    CxList increment = All.NewCxList();
@@ -92,16 +77,15 @@

    CxList testUn = thisRef.GetByAncs(test);
    CxList incrUn = thisRef.GetByAncs(increment);
-   CxList additionalLC = incrUn.FindAllReferences(testUn) + testUn.FindAllReferences(incrUn);
-   loopCounter.Add(additionalLC);
+   loopCounter.Add(incrUn.FindAllReferences(testUn), testUn.FindAllReferences(incrUn));

    //find all illegal for expressions
    CxList lcInIncrement = incrUn.FindAllReferences(loopCounter);
-   result.Add(lcInIncrement.GetFathers().FindByType(typeof(BinaryExpr)).GetAncOfType(typeof(IterationStmt)));
+   result.Add(lcInIncrement.GetFathers().FindByType<BinaryExpr>().GetAncOfType<IterationStmt>());

-   CxList oper = lcInIncrement.GetFathers() - binEx.FindByType(typeof(BinaryExpr));
+   CxList oper = lcInIncrement.GetFathers() - binEx.FindByType<BinaryExpr>();

-   CxList asnInIncrement = oper.FindByType(typeof(AssignExpr));
+   CxList asnInIncrement = oper.FindByType<AssignExpr>();

    CxList lc = thisRef.FindAllReferences(loopCounter);

@@ -124,20 +108,17 @@
            }
        }
    }
-   result.Add(totalResult.GetAncOfType(typeof(IterationStmt)));
-   asnInIncrement = assign;
+   result.Add(totalResult.GetAncOfType<IterationStmt>());

    CxList temp = methInv.GetByAncs(increment);
    CxList methInvInIncr = temp * rightSd;

-   result.Add(methInvInIncr.GetAncOfType(typeof(IterationStmt)));
-   asnInIncrement -= methInvInIncr.GetAncOfType(typeof(AssignExpr));
-   result.Add((temp - methInvInIncr).GetAncOfType(typeof(IterationStmt)));
+   result.Add(methInvInIncr.GetAncOfType<IterationStmt>(),
+       (temp - methInvInIncr).GetAncOfType<IterationStmt>());

-   //asnInIncrement now has all expressions of the increment part that are of the form -= +=
    //find all right side of the -= +=
    CxList allRefsInIncrement = thisRef.GetByAncs(assign);
-   CxList pointer = unaryEx.FindByName("Pointer");
+   CxList pointer = unaryEx.FindByShortName("Pointer");
```

```
    CxList rightSide = allRefsInIncrement * rightSd;

    CxList rtPtr = pointer * rightSd;
@@ -145,39 +126,39 @@
    CxList pointerRefInIncr = allRefsInIncrement.FindByFathers(rtPtr);

    CxList allRefToRs = All.NewCxList();
-   CxList change = All.NewCxList();

-   CxList curExpr = thisRef.GetByAncs(rightSide.GetAncOfType(typeof(IterationStmt)));
+   CxList curExpr = thisRef.GetByAncs(rightSide.GetAncOfType<IterationStmt>());

    allRefToRs = curExpr.FindAllReferences(rightSide) - rightSide;//+=
-   CxList allRefsInStmt = allRefToRs - thisRef.GetByAncs(increment) - thisRef.GetByAncs(init);
+   CxList allRefsToRemove = All.NewCxList(thisRef.GetByAncs(increment), thisRef.GetByAncs(init));
+   CxList allRefsInStmt = allRefToRs - allRefsToRemove;
    CxList operation = allRefsInStmt.GetFathers();
-   CxList edit = operation.FindByType(typeof(ExprStmt)) +
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Increment") +
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Decrement");
-   edit.Add(allRefsInStmt.FindByAssignmentSide(CxList.AssignmentSide.Left));
-   edit.Add(operation.FindByType(typeof(UnaryExpr)).FindByName("Address").GetAncOfType(typeof(MethodInvokeExpr)));
-
+   CxList opUnaryExpr = operation.FindByType<UnaryExpr>();
+   CxList edit = All.NewCxList(operation.FindByType<ExprStmt>(),
+       opUnaryExpr.FindByShortNames("Increment", "Decrement"),
+       allRefsInStmt.FindByAssignmentSide(CxList.AssignmentSide.Left),
+       opUnaryExpr.FindByShortName("Address").GetAncOfType<MethodInvokeExpr>());

    if(edit.Count > 0)
    {
        result.Add(cur);
    }
-   CxList thisExpr = thisRef.GetByAncs(pointerRefInIncr.GetAncOfType(typeof(IterationStmt)));
+   CxList thisExpr = thisRef.GetByAncs(pointerRefInIncr.GetAncOfType<IterationStmt>());
    allRefToRs = thisExpr.FindAllReferences(pointerRefInIncr) - pointerRefInIncr;
-   CxList allRefsStmt = allRefToRs - thisRef.GetByAncs(increment) - thisRef.GetByAncs(init);
-   CxList prm = allRefsStmt.GetFathers().FindByType(typeof(Param));
+   CxList allRefsToRem = All.NewCxList(thisRef.GetByAncs(increment), thisRef.GetByAncs(init));
+   CxList allRefsStmt = allRefToRs - allRefsToRem;
+   CxList prm = allRefsStmt.GetFathers().FindByType<Param>();
    if(prm.Count > 0)
    {
        result.Add(cur);
    }
-   CxList ptrToRf = allRefsStmt.GetFathers().FindByType(typeof(UnaryExpr)).FindByName("Pointer");
+   CxList ptrToRf = allRefsStmt.GetFathers().FindByType<UnaryExpr>().FindByShortName("Pointer");
    CxList ris = thisRef.FindByFathers(ptrToRf).GetFathers();
```

```
    operation = ris.GetFathers();
-   edit = operation.FindByType(typeof(ExprStmt)) +
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Increment") +
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Decrement");
-   edit.Add(ris.FindByAssignmentSide(CxList.AssignmentSide.Left));
+   opUnaryExpr = operation.FindByType<UnaryExpr>();
+   edit = All.NewCxList(operation.FindByType<ExprStmt>(),
+       opUnaryExpr.FindByShortNames("Increment", "Decrement"),
+       ris.FindByAssignmentSide(CxList.AssignmentSide.Left));

    if(edit.Count > 0)
    {
```

CPP / CPP_MISRA_CPP / R06_05_05_Lcv_Change_In_For_Stmt

Code changes

```
---
+++
@@ -18,30 +18,18 @@

 //gets all for statements

-CxList allFors = All.FindByType(typeof(IterationStmt));
-CxList helper = allFors;
-foreach(CxList allf in allFors)
-{
-   IterationStmt i = allf.TryGetCSharpGraph<IterationStmt>();
-   if(i != null)
-   {
-       IterationType it = i.IterationType;
-       if(!it.ToString().Equals("For"))
-           helper -= allf;
-   }
-}
-allFors = helper;
-
-   CxList rf = All.FindByType(typeof(UnknownReference)) + Find_All_Declarators();
-   CxList unrf = rf.GetByAncs(allFors);
-   CxList totalAsns = All.FindByType(typeof(AssignExpr));
-   CxList asns = totalAsns.GetByAncs(allFors);
-   CxList totalUn = All.FindByType(typeof(UnaryExpr));
-   CxList unaryEx = totalUn.GetByAncs(allFors);
-   CxList totalPf = All.FindByType(typeof(PostfixExpr));
-   CxList postfixEx = totalPf.GetByAncs(allFors);
-   CxList declarators = Find_All_Declarators().GetByAncs(allFors);
-   CxList leftSd = All.FindByAssignmentSide(CxList.AssignmentSide.Left);
+CxList allFors = Find_ForStatement();
+CxList decls = Find_All_Declarators();
+CxList rf = All.NewCxList(Find_UnknownReference(), decls);
```

```
+CxList unrf = rf.GetByAncs(allFors);
+CxList totalAsns = Find_AssignExpr();
+CxList asns = totalAsns.GetByAncs(allFors);
+CxList totalUn = Find_Unarys();
+CxList unaryEx = totalUn.GetByAncs(allFors);
+CxList totalPf = Find_PostfixExpr();
+CxList postfixEx = totalPf.GetByAncs(allFors);
+CxList declarators = decls.GetByAncs(allFors);
+CxList leftSd = All.FindByAssignmentSide(CxList.AssignmentSide.Left);


 foreach(CxList cur in allFors)
 {
@@ -55,43 +43,32 @@
     {
         Expression expr = iterA.Test;
         if(expr != null)
-        {
             testExpr.Add(All.FindById(expr.NodeId));
-        }
+
         //finds the increment element
         StatementCollection incrementColl = iterA.Increment;
+
         for(int i = 0; incrementColl != null && i < incrementColl.Count; i++)
-        {
-            if(incrementColl[i] != null)
-            {
-                incrExpr.Add(All.FindById(incrementColl[i].NodeId));
-            }
-        }
+            incrExpr.Add(All.FindById(incrementColl[i].NodeId));

         StatementCollection initColl = iterA.Init;
         for(int i = 0; initColl != null && i < initColl.Count; i++)
-        {
-            if(initColl[i] != null)
-            {
-                init.Add(All.FindById(initColl[i].NodeId));
-            }
-        }
+            init.Add(All.FindById(initColl[i].NodeId));
     }

-    CxList loopCounter = unrf.FindByFathers(init.FindByType(typeof(ExprStmt)));
+    CxList loopCounter = unrf.FindByFathers(init.FindByType<ExprStmt>());

     CxList unknownRef = unrf.GetByAncs(init);
     CxList leftAsn = unknownRef * leftSd;
```

```
-   loopCounter.Add(leftAsn + declarators.GetByAncs(init));
-   CxList lcv = loopCounter;
+   loopCounter.Add(leftAsn, declarators.GetByAncs(init));

    //all references in statement

    CxList incrUn = unrf.GetByAncs(incrExpr);
    CxList testUn = unrf.GetByAncs(testExpr);
-   CxList testRef = incrUn + testUn;
+   CxList testRef = All.NewCxList(incrUn, testUn);

-   CxList tmp = lcv.FindAllReferences(testRef) - testRef;
+   CxList tmp = loopCounter.FindAllReferences(testRef) - testRef;

    testRef -= testRef.FindAllReferences(tmp) - tmp;

@@ -100,33 +77,31 @@

    CxList rmv = testUn.FindAllReferences(incrUn);
    CxList rmv2 = incrUn.FindAllReferences(testUn);
-   testRef -= (rmv + rmv2);
+   testRef -= All.NewCxList(rmv, rmv2);
    if(testRef.Count == 0)
    {
        continue;
    }

    //finds bar(&n) in the test element
-   CxList test = testRef.GetFathers().FindByType(typeof(UnaryExpr)).FindByShortName("Address");
-   test = test.GetAncOfType(typeof(MethodInvokeExpr));
+   CxList test = testRef.GetFathers().FindByType<UnaryExpr>().FindByShortName("Address");
+   test = test.GetAncOfType<MethodInvokeExpr>();
    CxList reference = testRef.GetByAncs(test);

    //for+= =
    CxList testLeftAsn = All.NewCxList();
-   CxList assn = asns.GetByAncs(testExpr)
-       + asns.GetByAncs(incrExpr);
+   CxList assn = All.NewCxList(asns.GetByAncs(testExpr),
+         asns.GetByAncs(incrExpr));

    testLeftAsn.Add(testRef.FindByFathers(assn) * leftSd);
    reference.Add(testLeftAsn);
    //for ++n,--n
-   CxList unary = unaryEx.GetByAncs(testExpr) +
-       unaryEx.GetByAncs(incrExpr);
-   unary = unary.FindByShortName("Decrement") +
-       unary.FindByShortName("Increment");
```

```
-   reference.Add(testRef.GetByAncs(unary).FindByType(typeof(UnknownReference)));
+   CxList unary = All.NewCxList(unaryEx.GetByAncs(testExpr),
+       unaryEx.GetByAncs(incrExpr));
+   unary = unary.FindByShortNames("Decrement", "Increment");
+   reference.Add(testRef.GetByAncs(unary).FindByType<UnknownReference>());
    //for n++,n--
-   CxList pE = postfixEx.GetByAncs(testExpr)
-       + postfixEx.GetByAncs(incrExpr);
+   CxList pE = All.NewCxList(postfixEx.GetByAncs(testExpr), postfixEx.GetByAncs(incrExpr));
    CxList postRef = All.NewCxList();
    foreach(CxList w in pE)
    {
@@ -144,6 +119,6 @@


    //checks if the found illegal loop control variable  is not the loop counter
    //and in case it's not adds it to the result.
-   result.Add(reference - reference.FindAllReferences(lcv));
+   result.Add(reference - reference.FindAllReferences(loopCounter));


 }
```

CPP / CPP_MISRA_CPP / R06_05_06_Bool_Lcv_Change

Code changes

```
---
+++
@@ -30,36 +30,19 @@
 */


 //finds all for statements
-CxList allFors = All.FindByType(typeof(IterationStmt));
-
-CxList helper = allFors;
-foreach(CxList allf in allFors)
-{
-   IterationStmt i = allf.TryGetCSharpGraph<IterationStmt>();
-   if(i != null)
-   {
-       IterationType it = i.IterationType;
-       if(!it.ToString().Equals("For"))
-       {
-           helper -= allf;
-       }
-   }
-}
-allFors = helper;
-
-CxList totalUnknownR = All.FindByType(typeof(UnknownReference)) + Find_All_Declarators() + All.FindByType(typeof(FieldDecl));
+CxList allFors = Find_ForStatement();
```

```
+CxList decls = Find_All_Declarators();
+CxList totalUnknownR = All.NewCxList(Find_UnknownReference(), decls, Find_FieldDecls());
 CxList unrf = totalUnknownR.GetByAncs(allFors);
-CxList unrfWithMA = unrf + All.FindByType(typeof(MemberAccess)).GetByAncs(allFors);
-CxList declarators = Find_All_Declarators().GetByAncs(allFors);
-CxList postFix = All.FindByType(typeof(PostfixExpr)).GetByAncs(allFors);
-CxList typeR = All.FindByType(typeof(TypeRef));
+CxList unrfWithMA = All.NewCxList(unrf, Find_MemberAccesses().GetByAncs(allFors));
+CxList declarators = decls.GetByAncs(allFors);
+CxList postFix = Find_PostfixExpr().GetByAncs(allFors);
 CxList leftSd = All.FindByAssignmentSide(CxList.AssignmentSide.Left);

 //first we find all loop control variables (appear in condition and iteration) that are modified inside statement
 foreach(CxList cur in allFors)
 {
    CxList eis = unrfWithMA.GetByAncs(cur);
-   CxList results = All.NewCxList();
    CxList testExpr = All.NewCxList();
    CxList incrExpr = All.NewCxList();
    CxList init = All.NewCxList();
@@ -77,40 +60,28 @@

        StatementCollection incrementColl = iterA.Increment;
        for(int i = 0;incrementColl != null && i < incrementColl.Count; i++)
-       {
-           if(incrementColl[i] != null)
-           {
-               incrExpr.Add(All.FindById(incrementColl[i].NodeId));
-           }
-       }
+           incrExpr.Add(All.FindById(incrementColl[i].NodeId));
+
        //finds all init elements
-
        StatementCollection initColl = iterA.Init;
        for(int i = 0;initColl != null && i < initColl.Count; i++)
-       {
-           if(initColl[i] != null)
-           {
-               init.Add(All.FindById(initColl[i].NodeId));
-           }
-       }
-
+           init.Add(All.FindById(initColl[i].NodeId));
    }

-   CxList loopCounter = eis.FindByFathers(init.FindByType(typeof(ExprStmt)));
+   CxList loopCounter = eis.FindByFathers(init.FindByType<ExprStmt>());
```

```
    CxList unknownRef = eis.GetByAncs(init);

    CxList leftAsn = unknownRef * leftSd;

-   loopCounter.Add(leftAsn + declarators.GetByAncs(init));
-   CxList lcv = loopCounter;
+   loopCounter.Add(leftAsn, declarators.GetByAncs(init));

    //finds all references inside the statement
    CxList incrUn = eis.GetByAncs(incrExpr);
    CxList testUn = eis.GetByAncs(testExpr);
-   CxList allRefsInFor = incrUn + testUn;
+   CxList allRefsInFor = All.NewCxList(incrUn, testUn);

-   CxList tmp = lcv.FindAllReferences(allRefsInFor) - allRefsInFor;
+   CxList tmp = loopCounter.FindAllReferences(allRefsInFor) - allRefsInFor;
    allRefsInFor -= allRefsInFor.FindAllReferences(tmp) - tmp;

    incrUn = allRefsInFor.GetByAncs(incrExpr);
@@ -118,34 +89,31 @@

    CxList rmv = testUn.FindAllReferences(incrUn);
    CxList rmv2 = incrUn.FindAllReferences(testUn);
-   allRefsInFor -= (rmv + rmv2);
+   allRefsInFor -= All.NewCxList(rmv, rmv2);
    if(allRefsInFor.Count == 0)
-   {
        continue;
-   }

    allRefsInFor = eis.FindAllReferences(allRefsInFor) -
-       (eis.GetByAncs(init) + eis.GetByAncs(incrExpr) + eis.GetByAncs(testExpr));
+       All.NewCxList(eis.GetByAncs(init), eis.GetByAncs(incrExpr), eis.GetByAncs(testExpr));
    //checks if they are modified inside the statement
    CxList backToAsn = All.NewCxList();

    allRefsInFor -= allRefsInFor.GetMembersOfTarget().GetTargetOfMembers();
    CxList operation = allRefsInFor.GetFathers();
+   CxList opUnaryExpr = operation.FindByType<UnaryExpr>();

-   CxList change = postFix.FindByFathers(allRefsInFor.GetAncOfType(typeof(ExprStmt)));
+   CxList change = postFix.FindByFathers(allRefsInFor.GetAncOfType<ExprStmt>());
+   change = allRefsInFor.GetByAncs(change.GetAncOfType<ExprStmt>());
+   change.Add(opUnaryExpr.FindByShortNames("Increment", "Decrement"));
+   change.Add(opUnaryExpr.FindByShortName("Address").GetAncOfType<MethodInvokeExpr>());

-   change = allRefsInFor.GetByAncs(change.GetAncOfType(typeof(ExprStmt)));
```

```
-   change.Add(operation.FindByType(typeof(UnaryExpr)).FindByName("Increment") +
-       operation.FindByType(typeof(UnaryExpr)).FindByName("Decrement"));
    backToAsn.Add(allRefsInFor.FindByAssignmentSide(CxList.AssignmentSide.Left));
-   change.Add(operation.FindByType(typeof(UnaryExpr)).FindByName("Address").GetAncOfType(typeof(MethodInvokeExpr)));
-   backToAsn.Add(allRefsInFor.GetByAncs(change.GetAncOfType(typeof(ExprStmt))));
+   backToAsn.Add(allRefsInFor.GetByAncs(change.GetAncOfType<ExprStmt>()));

    //checks if the found illegal loop control variable  is not the loop counter
    //and in case it's not adds it to the result.

-   results = backToAsn;
-   CxList temp = totalUnknownR.FindDefinition(results);
+   CxList temp = totalUnknownR.FindDefinition(backToAsn);
    foreach(CxList t in temp)
    {
        CSharpGraph g = t.GetFirstGraph();
```

CPP / CPP_MISRA_CPP / R07_01_01_Declare_Const_if_not_Modified

Code changes

```
---
+++
@@ -16,28 +16,33 @@

 */
-CxList decls = Find_All_Declarators() + All.FindByType(typeof(ParamDecl));
+CxList methodDecls = Find_MethodDecls();
+CxList allDecls = Find_All_Declarators();
+CxList decls = All.NewCxList(allDecls, Find_ParamDecl());
 decls -= decls.FindByShortName("");
-decls -= decls.GetByAncs(All.FindByType(typeof(StructDecl)));
+decls -= decls.GetByAncs(Find_StructDecl());
+CxList typeRefs = Find_TypeRef();
 //Remove params of prototype functions
-CxList protoMethods = All.FindByType(typeof(MethodDecl));
-protoMethods -= All.FindByFathers(protoMethods).FindByType(typeof(StatementCollection)).GetFathers();
+CxList protoMethods = methodDecls -
+   Find_StatementCollection().FindByFathers(methodDecls).GetFathers();
 decls -= decls.GetParameters(protoMethods);
 //Find const declarations on variables - especially pointer.
-CxList consts = All.FindByType(typeof(TypeRef)).FindByRegex(@"const\s+(\w+::)?\w+\s*&", false, false, false);
-consts = consts.GetAncOfType(typeof(ParamDecl)) + decls.GetByAncs(consts.GetFathers().FindByType(typeof(VariableDeclStmt)));
-consts.Add(decls.GetByAncs((All - All.FindByType(typeof(MethodDecl))).FindByFieldAttributes(Modifiers.Readonly)));
+CxList consts = typeRefs.FindByRegex(@"const\s+(\w+::)?\w+\s*&", false, false, false);
+consts = All.NewCxList(consts.GetAncOfType<ParamDecl>(),
+   decls.GetByAncs(consts.GetFathers().FindByType<VariableDeclStmt>()),
+   decls.GetByAncs((All - methodDecls).FindByFieldAttributes(Modifiers.Readonly)));
 CxList pointers = decls.FindByRegex(@"\w+\s*\*",false,false,false);
```

```
-decls -= consts + pointers;
+decls -= All.NewCxList(consts, pointers);

-CxList potPoint = All.FindByType(typeof(TypeRef)).FindByRegex(@"\w+\s*\*\s*const\W", false, false, false);
-pointers -= Find_All_Declarators().GetByAncs(potPoint.GetFathers())
-   + potPoint.GetAncOfType(typeof(ParamDecl));
+CxList potPoint = typeRefs.FindByRegex(@"\w+\s*\*\s*const\W", false, false, false);
+pointers -= All.NewCxList(allDecls.GetByAncs(potPoint.GetFathers()),
+   potPoint.GetAncOfType<ParamDecl>());

 //Find unmodified
-CxList refs = All.FindAllReferences(decls + pointers) - decls - pointers;
-refs = (decls + pointers).FindDefinition(refs.FindByAssignmentSide(CxList.AssignmentSide.Left));
-result.Add(decls + pointers - refs);
+CxList declsPointers = All.NewCxList(decls, pointers);
+CxList refs = All.FindAllReferences(declsPointers) - declsPointers;
+refs = declsPointers.FindDefinition(refs.FindByAssignmentSide(CxList.AssignmentSide.Left));
+result.Add(declsPointers - refs);

 //Add modification with << operator and usage of object's functions.
 CxList doubleLeft = All.GetByBinaryOperator(BinaryOperator.ShiftLeft);
@@ -46,19 +51,20 @@
    CSharpGraph left = curr.TryGetCSharpGraph<BinaryExpr>().Left;
    sanitizeLeft.Add(All.FindById(left.NodeId));
 }
-doubleLeft = sanitizeLeft.FindByType(typeof(UnknownReference));
+doubleLeft = sanitizeLeft.FindByType<UnknownReference>();
 CxList members = All.FindByMemberAccess(".*").GetTargetOfMembers();
-members.Add(All.FindByFathers(members.FindByType(typeof(IndexerRef))));
-doubleLeft.Add(members.FindByType(typeof(UnknownReference)));
-result -= (pointers + decls).FindDefinition(doubleLeft);
+members.Add(All.FindByFathers(members.FindByType<IndexerRef>()));
+doubleLeft.Add(members.FindByType<UnknownReference>());
+result -= All.NewCxList(pointers, decls).FindDefinition(doubleLeft);

 //Remove typedefs
-CxList typedefs = All.FindByType(typeof(StringLiteral)).FindByName("CX_TYPEDEF");
-typedefs = typedefs.GetAncOfType(typeof(VariableDeclStmt))
-   + typedefs.GetAncOfType(typeof(FieldDecl));
-typedefs = All.GetByAncs(typedefs).FindByType(typeof(Declarator));
+CxList typedefs = Find_String_Literal().FindByShortName("CX_TYPEDEF");
+typedefs = All.NewCxList(typedefs.GetAncOfType<VariableDeclStmt>(),
+   typedefs.GetAncOfType<FieldDecl>());
+typedefs = All.GetByAncs(typedefs).FindByType<Declarator>();
 //Remove gotos
-CxList gotos = All.FindByFathers(Find_All_Declarators());
-gotos = gotos.FindByName("goto").GetFathers();
+CxList gotos = All.FindByFathers(allDecls);
```

```
+gotos = gotos.FindByShortName("goto").GetFathers();
 //Remove (void) parameters
-result -= typedefs + gotos + All.FindByFathers(result).FindByShortName("void").GetFathers();;
+result -= typedefs;
+result.Add(gotos, All.FindByFathers(result).FindByShortName("void").GetFathers());
```

CPP / CPP_MISRA_CPP / R07_01_02_Declare_Ref_Const_if_not_Modified

Code changes

```
---
+++
@@ -16,24 +16,26 @@
 */

+CxList methodDecls = Find_MethodDecls();
+
 //Find pointer parameters.
-CxList typerefs = All.FindByType(typeof(TypeRef));
+CxList typerefs = Find_TypeRef();
 CxList pointers = typerefs.FindByRegex(@"\w+\s*\*", false, false, false);
-CxList potPoint = typerefs.FindByRegex(@"\w+\s+const\s*\*",false,false,false) +
-   typerefs.FindByRegex(@"(?<=const\s+)(\w+::)?\w+\s*\*",false,false,false);
+CxList potPoint = All.NewCxList(typerefs.FindByRegex(@"\w+\s+const\s*\*",false,false,false),
+   typerefs.FindByRegex(@"(?<=const\s+)(\w+::)?\w+\s*\*",false,false,false));
 pointers = pointers - potPoint;
-pointers = pointers.GetAncOfType(typeof(ParamDecl));
+pointers = pointers.GetAncOfType<ParamDecl>();
 pointers -= pointers.FindByShortName("");

 //Get prototype parameters
-CxList protoMethods = All.FindByType(typeof(MethodDecl));
-protoMethods -= All.FindByFathers(protoMethods).FindByType(typeof(StatementCollection)).GetFathers();
+CxList protoMethods =
+   methodDecls - Find_StatementCollection().FindByFathers(methodDecls).GetFathers();
 CxList protoParams = All.GetParameters(protoMethods);
 pointers -= protoParams;

 //Find modified pointer objects.
 CxList modified = All.FindAllReferences(pointers) - pointers;
-CxList unarys = modified.GetFathers().FindByType(typeof(UnaryExpr));
+CxList unarys = modified.GetFathers().FindByType<UnaryExpr>();
 unarys = unarys.FindByAssignmentSide(CxList.AssignmentSide.Left);
 modified = pointers.FindDefinition(All.FindByFathers(unarys));

@@ -44,63 +46,64 @@
    CSharpGraph left = curr.TryGetCSharpGraph<BinaryExpr>().Left;
    sanitizeLeft.Add(All.FindById(left.NodeId));
 }
```

```
-doubleLeft = sanitizeLeft.FindByType(typeof(UnknownReference));
+doubleLeft = sanitizeLeft.FindByType<UnknownReference>();
 CxList members = All.FindByMemberAccess(".*").GetTargetOfMembers();
-members.Add(All.FindByFathers(members.FindByType(typeof(IndexerRef))));
-doubleLeft.Add(members.FindByType(typeof(UnknownReference)));
+members.Add(All.FindByFathers(members.FindByType<IndexerRef>()));
+doubleLeft.Add(members.FindByType<UnknownReference>());

 //Find referenced parameters.
-CxList refs = typerefs.FindByRegex(@"\w+\s*&", false, false,false).GetAncOfType(typeof(ParamDecl));
+CxList refs = typerefs.FindByRegex(@"\w+\s*&", false, false,false).GetAncOfType<ParamDecl>();
 refs -= refs.FindByFieldAttributes(Modifiers.Readonly);
-refs -= typerefs.FindByRegex(@"const\s+(\w+::)?\w+\s*&", false, false,false).GetAncOfType(typeof(ParamDecl));
+refs -= typerefs.FindByRegex(@"const\s+(\w+::)?\w+\s*&", false, false,false).GetAncOfType<ParamDecl>();
 refs -= protoParams;
 refs -= refs.FindByShortName("");
 //Find modified ref'd parameters.
 CxList modRefs = All.FindAllReferences(refs) - refs;
 modRefs = refs.FindDefinition(modRefs.FindByAssignmentSide(CxList.AssignmentSide.Left));
-modRefs.Add((pointers + refs).FindDefinition(doubleLeft));
+modRefs.Add(All.NewCxList(pointers,refs).FindDefinition(doubleLeft));
 modified.Add(modRefs);

 //Find methods that are overriden by modified.
-CxList modMethods = modified.GetAncOfType(typeof(MethodDecl));
-CxList relMethods = All.FindByType(typeof(MethodDecl)).FindByShortName(modMethods) - modMethods - protoMethods;
-CxList allParams = All.FindByType(typeof(ParamDecl));
-CxList relParams = allParams.FindByShortName(modified) - modified - protoParams;
+CxList modMethods = modified.GetAncOfType<MethodDecl>();
+CxList relMethods = methodDecls.FindByShortName(modMethods) - All.NewCxList(modMethods, protoMethods);
+CxList allParams = Find_ParamDecl();
+CxList relParams = allParams.FindByShortName(modified) - All.NewCxList(modified, protoParams);
 CxList relClasses = All.GetClass(relMethods);
 CxList modClasses = All.GetClass(modMethods);
-typerefs = typerefs.GetByAncs(relMethods.GetParameters(relParams.GetAncOfType(typeof(MethodDecl))));
+typerefs = typerefs.GetByAncs(relMethods.GetParameters(relParams.GetAncOfType<MethodDecl>()));
 bool isOverride = true;
 foreach(CxList curr in relParams) {
    CxList sons = modClasses.InheritsFrom(relClasses.GetClass(curr));
    CxList others = modified.FindByShortName(curr);
-   CxList currMethod = curr.GetAncOfType(typeof(MethodDecl));
+   CxList currMethod = curr.GetAncOfType<MethodDecl>();
    if (currMethod.Count == 0) {
        continue;
    }
    CxList otherMethods = relMethods.FindByShortName(currMethod);
    others = others.GetByAncs(otherMethods);
    String currMethodString = currMethod.TryGetCSharpGraph<MethodDecl>().Name;
```

```
-   CxList currParams = relMethods.GetParameters(curr.GetAncOfType(typeof(MethodDecl)));
+   CxList currParams = relMethods.GetParameters(curr.GetAncOfType<MethodDecl>());
    currParams = typerefs.GetByAncs(currParams);
    foreach(CxList other in others) {
-       if (curr.GetAncOfType(typeof(MethodDecl)).Count == 0) {
+       if (curr.GetAncOfType<MethodDecl>().Count == 0) {
            continue;
        }
-       String otherMethod = other.GetAncOfType(typeof(MethodDecl)).TryGetCSharpGraph<MethodDecl>().Name;
-       CxList otherParams = allParams.GetParameters(other.GetAncOfType(typeof(MethodDecl)));
+       String otherMethod = other.GetAncOfType<MethodDecl>().TryGetCSharpGraph<MethodDecl>().Name;
+       CxList otherParams = allParams.GetParameters(other.GetAncOfType<MethodDecl>());
        otherParams = typerefs.GetByAncs(otherParams);
        //Check if otherMethod overrides currMethod.
-       if( sons.FindByName(modClasses.GetClass(other)).Count == 1 && //other's class inherits curr's class.
-       otherMethod.Equals(currMethodString) && currParams.Count == otherParams.Count) {
+       if(sons.FindByName(modClasses.GetClass(other)).Count == 1 && //other's class inherits curr's class.
+       otherMethod.Equals(currMethodString) && currParams.Count == otherParams.Count)
+       {
            for(int i = 0; i < currParams.Count; i++) {
-               string cName = ((ParamDecl) currParams.data.GetByIndex(i)).Name;
-               string oName = ((ParamDecl) otherParams.data.GetByIndex(i)).Name;
+               string cName = currParams.ElementAt(i).TryGetCSharpGraph<ParamDecl>().Name;
+               string oName = otherParams.ElementAt(i).TryGetCSharpGraph<ParamDecl>().Name;
                if(!cName.Equals(oName)) {
                    isOverride = false;
                    break;
                }
-           }//end for
-       }//end if
+           }
+       }
        else {
            isOverride = false;
        }
@@ -112,4 +115,5 @@
    }//end foreach on others
 }//end foreach on relParams

-result = refs + pointers - modified;
+result.Add(refs, pointers);
+result -= modified;
```

CPP / CPP_MISRA_CPP / R07_05_02_Address_Assignment_out_of_Scope

Code changes

```
---
+++
@@ -1,6 +1,6 @@
```

```
 /*
  MISRA CPP RULE 7-5-2
- ------------------------------
+ //////////////////////////////
  This query searches for address assignments of auto-storage variables outside of their scope.

  The Example below shows code with vulnerability:
@@ -16,32 +16,32 @@

 //Find auto-storage variables.
 CxList decs = Find_All_Declarators();
-CxList vars = decs;
-CxList nonAuto = (All - All.FindByType(typeof(MethodDecl)));
-nonAuto = nonAuto.FindByFieldAttributes(Modifiers.Extern) + nonAuto.FindByFieldAttributes(Modifiers.Static);
+CxList nonAuto = (All - Find_MethodDecls());
+nonAuto.Add(nonAuto.FindByFieldAttributes(Modifiers.Extern),
+   nonAuto.FindByFieldAttributes(Modifiers.Static));
 nonAuto = decs.GetByAncs(nonAuto);
-vars -= decs;
-CxList refs = All.FindByType(typeof(Reference));
+CxList vars = decs - nonAuto;
+CxList refs = Find_Reference();
 CxList addresses = All.NewCxList();
 //Getting address assignments.
-CxList unarys = All.FindByType(typeof(UnaryExpr));
+CxList unarys = Find_Unarys();
 foreach (CxList unary in unarys) {
    if(unary.TryGetCSharpGraph<UnaryExpr>().Operator == UnaryOperator.Address &&
-   unary.GetAncOfType(typeof(Param)).Count == 0) {
+   unary.GetAncOfType<Param>().Count == 0) {
        addresses.Add(All.FindById(unary.TryGetCSharpGraph<UnaryExpr>().Right.NodeId));
    }
 }
 vars = All.FindDefinition(addresses.FindAllReferences(vars));
 refs = refs.DataInfluencedBy(vars);
 decs = decs.FindDefinition(refs);
-CxList classes = refs.FindByType(typeof(ClassDecl));
+CxList classes = refs.FindByType<ClassDecl>();
 foreach(CxList auto in vars) {
    //Get auto's scope
-   CxList scope = auto.GetAncOfType(typeof(StatementCollection));
-   if (scope.Count == 0){
-       scope = auto.GetAncOfType(typeof(ClassDecl));
+   CxList scope = auto.GetAncOfType<StatementCollection>();
+   if (scope != null && scope.Count == 0){
+       scope = auto.GetAncOfType<ClassDecl>();
        if (scope.Count == 0){
-           scope = auto.GetAncOfType(typeof(StructDecl));
```

```
+           scope = auto.GetAncOfType<StructDecl>();
        }
        else if (scope.TryGetCSharpGraph<ClassDecl>().Name.Contains("checkmarx_default_classname")){
            continue;
@@ -60,20 +60,20 @@

        //Get call's scope, check if within auto's scope.
        bool isIn = false;
-       CxList callScope = dec;
-       CxList prevScope = dec;
+       CxList callScope = All.NewCxList(dec);
+       CxList prevScope = All.NewCxList(dec);
        do {
-           prevScope = callScope;
+           prevScope = All.NewCxList(callScope);
            CxList check = scope * callScope;
            if (check.Count > 0) {
                isIn = true;
            }
-           CxList oldScope = callScope;
-           callScope = oldScope.GetAncOfType(typeof(StatementCollection));
+           CxList oldScope = All.NewCxList(callScope);
+           callScope = oldScope.GetAncOfType<StatementCollection>();
            if (callScope.Count == 0){
-               callScope = oldScope.GetAncOfType(typeof(ClassDecl));
+               callScope = oldScope.GetAncOfType<ClassDecl>();
                if (callScope.Count == 0){
-                   callScope = oldScope.GetAncOfType(typeof(StructDecl));
+                   callScope = oldScope.GetAncOfType<StructDecl>();
                }
            }
```

CPP / CPP_MISRA_CPP / R08_04_03_Explicit_Return_Throw

Code changes

```
---
+++
@@ -16,22 +16,22 @@

 */

-
+CxList typeRefs = Find_TypeRef();
 //finds all methods that have non void return type but return void.
-CxList returnStmt = All.FindByType(typeof(ReturnStmt));
+CxList returnStmt = Find_ReturnStmt();
 CxList emptyReturn = returnStmt - All.FindByFathers(returnStmt).GetFathers();
-CxList returnType = All.FindByType(typeof(TypeRef)).FindByFathers(emptyReturn.GetAncOfType(typeof(MethodDecl)));
```

```
+CxList returnType = typeRefs.FindByFathers(emptyReturn.GetAncOfType<MethodDecl>());
 CxList voidRT = returnType.FindByShortName("void");
 CxList nonVoidRT = returnType - voidRT;
 result = nonVoidRT;



 //finds methods that have non void return type but don't have return statements:
-CxList allMethodDecl = All.FindByType(typeof(MethodDecl));
+CxList allMethodDecl = Find_MethodDecls();
 //remove all definitions
-CxList declared = All.FindByFathers(allMethodDecl).FindByType(typeof(StatementCollection)).GetFathers();
+CxList declared = All.FindByFathers(allMethodDecl).FindByType<StatementCollection>().GetFathers();
 //get non those who have non void return types:
-returnType = All.FindByType(typeof(TypeRef)).FindByFathers(declared);
+returnType = typeRefs.FindByFathers(declared);
 nonVoidRT = returnType - returnType.FindByShortName("void");
 CxList backToMeth = nonVoidRT.GetFathers();
```

CPP / CPP_MISRA_CPP / R10_03_02_Find_Override_Without_Virtual

Code changes

```
---
+++
@@ -19,29 +19,30 @@
 */

+CxList methodDecls = Find_MethodDecls();
+CxList classDecls = Find_ClassDecl();
+CxList typeRef = Find_TypeRef();
+
 //Find virtual methods.
-CxList virtMethod = All.FindByType(typeof(MethodDecl));
-virtMethod = virtMethod.FindByFieldAttributes(Modifiers.Virtual);
+CxList virtMethod = methodDecls.FindByFieldAttributes(Modifiers.Virtual);

 //Find methods that may be problematic.
-CxList protoMethods = All.FindByType(typeof(MethodDecl));
-protoMethods -= All.FindByFathers(protoMethods).FindByType(typeof(StatementCollection)).GetFathers();
+CxList protoMethods = methodDecls -
+   Find_StatementCollection().FindByFathers(methodDecls).GetFathers();
 protoMethods -= virtMethod;
 protoMethods = protoMethods.FindByShortName(virtMethod);

-CxList virtClasses = All.FindByType(typeof(ClassDecl)).GetClass(virtMethod);
-CxList protoClasses = All.FindByType(typeof(ClassDecl)).InheritsFrom(virtMethod);
-protoClasses = All.FindByType(typeof(ClassDecl));//protoClasses.GetClass(protoMethods);
+CxList virtClasses = classDecls.GetClass(virtMethod);
```

```
-CxList virtParams = All.FindByType(typeof(ParamDecl));
+CxList virtParams = Find_ParamDecl();
 CxList protoParams = virtParams.GetParameters(protoMethods);
 virtParams = virtParams.GetParameters(virtMethod);

-CxList virtTyperefs = All.FindByType(typeof(TypeRef)).GetByAncs(virtParams);
-CxList protoTyperefs = All.FindByType(typeof(TypeRef)).GetByAncs(protoParams);
+CxList virtTyperefs = typeRef.GetByAncs(virtParams);
+CxList protoTyperefs = typeRef.GetByAncs(protoParams);

 foreach (CxList curr in virtMethod) {
-   CxList sons = protoClasses.InheritsFrom(virtClasses.GetClass(curr));//Get classes that inherent from curr's class.
+   CxList sons = classDecls.InheritsFrom(virtClasses.GetClass(curr));//Get classes that inherent from curr's class.
    //Get methods with same name that aren't virtual.
    CxList others = protoMethods.FindByShortName(curr);
    others = others.GetByClass(sons);
@@ -53,13 +54,13 @@
        CxList otherTypeRefs = protoParams.GetParameters(other);
        otherTypeRefs = protoTyperefs.GetByAncs(otherTypeRefs);
        //Check if otherMethod overrides currMethod.
-       if(sons.FindByShortName(protoClasses.GetClass(other)).Count == 1 &&
+       if(sons.FindByShortName(classDecls.GetClass(other)).Count == 1 &&
            currTyperefs.Count == otherTypeRefs.Count)
        {
            for(int i = 0; i < currTyperefs.Count; i++)
            {
-               string cName = ((TypeRef) currTyperefs.data.GetByIndex(i)).TypeName;
-               string oName = ((TypeRef) otherTypeRefs.data.GetByIndex(i)).TypeName;
+               string cName = currTyperefs.ElementAt(i).TryGetCSharpGraph<TypeRef>().TypeName;
+               string oName = otherTypeRefs.ElementAt(i).TryGetCSharpGraph<TypeRef>().TypeName;
                if(!cName.Equals(oName))
                {
                    isOverride = false;
```

CPP / CPP_Stored_Vulnerabilities / Stored_DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,20 +1 @@
-CxList tables =   All.FindByName("*orders*", false) +
-           All.FindByName("*credit*", false) +
-           All.FindByName("*invoice*", false) +
-           All.FindByName("*booking*", false) +
-           All.FindByName("*bill*", false) +
-           All.FindByName("*payment*", false) +
-           All.FindByName("*account*", false) +
-           All.FindByName("*cash*", false) +
```

```
-           All.FindByName("*customer*", false);
-
-CxList inputs = Find_Read()+Find_DB();
-CxList db = Find_DB();
-
-CxList user = All.FindByName("*user*", false) +
-         All.FindByName("*cust*", false) +
-         All.FindByName("*member*", false);
-
-db = db.DataInfluencedBy(tables);
-db = db - db.DataInfluencedBy(user);
-result = inputs.DataInfluencingOn(db);
+//This query is deprecated.
```

CPP / CPP_Weak_Cryptography / Use_Of_Weak_Hashing_Primitive

Code changes

```
---
+++
@@ -7,6 +7,8 @@
    - SHA-1
 */
 CxList methods = Find_Methods();
+CxList parameters = Find_Parameters().CxSelectDomProperty<Param>(p => p.Value);
+CxList objCreateExpr = Find_ObjectCreations();

 ////////////////Crypto++////////////////
 string[] weakHashAlgorithms = { "*MD2", "*MD4", "*MD5", "*RIPEMD160", "*SHA1"};
@@ -32,7 +34,7 @@
 List < string > weakdigests = new List<string> {
        "SHA1", "SHA1", "RIPEMD160", "MD2", "MD5" };

-CxList opensslweakdigests = All.GetParameters(methods.FindByShortName("EVP_get_digestbyname"))
+CxList opensslweakdigests = parameters.GetParameters(methods.FindByShortName("EVP_get_digestbyname"))
    .FindByShortNames(weakdigests);

 result.Add(opensslweakdigests);
@@ -46,10 +48,12 @@
 result.Add(botanWeak);

 // When a HashFunction is created using a weak algorithm as parameter
-CxList createHashes = All.FindByType("HashFunction").GetMembersOfTarget().FindByShortNames(
+CxList hashFunctions = All.FindByType("HashFunction");
+CxList createHashes = hashFunctions.GetMembersOfTarget().FindByShortNames(
    new List<string> { "create", "create_or_throw", "copy_state" });
+createHashes.Add(parameters.GetParameters(objCreateExpr.FindByFathers(hashFunctions)));

-CxList paramsCreate = All.GetParameters(createHashes).FindByShortNames(
+CxList paramsCreate = parameters.GetParameters(createHashes).FindByShortNames(
```

```
    new List<string> { "SHA-1", "SHA-160", "SHA1", "RIPEMD-160", "MD5", "MD4" });


 result.Add(paramsCreate);
@@ -60,12 +64,11 @@
 // similar to MD5 or Sha - 1 wich are vulnerable
 CxList weakMethod = methods.FindByShortName("crypto_generichash");


-CxList nullKey = All.GetParameters(weakMethod, 4)
+CxList nullKey = parameters.GetParameters(weakMethod, 4)
     .FindByAbstractValue(abstractValue => abstractValue is NullAbstractValue);


 IAbstractValue zero = new IntegerIntervalAbstractValue(0);
-CxList keyLengthZero = All.GetParameters(weakMethod, 5)
+CxList keyLengthZero = parameters.GetParameters(weakMethod, 5)
     .FindByAbstractValue(abstractValue => zero.IncludedIn(abstractValue));


-result.Add(nullKey);
-result.Add(keyLengthZero);
+result.Add(nullKey,keyLengthZero);
```

CSharp / CSharp_Best_Coding_Practice / Aptca_Methods_Call_Non_Aptca_Methods

Code changes

```
---
+++
@@ -1,7 +1,6 @@
 CxList allMethods = Find_MethodDecls();
-CxList aptcaMethods = All.FindByCustomAttribute("AllowPartiallyTrustedCallersAttribute").GetAncOfType(typeof(MethodDecl));
-CxList nonAptcaMethods = allMethods - aptcaMethods;
-CxList allMethodsCalls = All.FindAllReferences(allMethods) - allMethods;
+CxList aptcaMethods = Find_CustomAttribute().FindByCustomAttribute("AllowPartiallyTrustedCallersAttribute")
+   .GetAncOfType<MethodDecl>();
 CxList allNonAptcaMethods = allMethods - aptcaMethods;
 CxList allNonAptcaMethodsCalls = All.FindAllReferences(allNonAptcaMethods) - allNonAptcaMethods;
 result = allNonAptcaMethodsCalls.GetByAncs(aptcaMethods);
```

CSharp / CSharp_Best_Coding_Practice / Exposure_of_Resource_to_Wrong_Sphere

Code changes

```
---
+++
@@ -1,5 +1,5 @@
-CxList allFields = All.FindByType(typeof(FieldDecl));
-allFields -= All.FindByCustomAttribute("event").GetAncOfType<FieldDecl>();
+CxList allFields = Find_FieldDecls();
+allFields -= Find_CustomAttribute().FindByCustomAttribute("event").GetAncOfType<FieldDecl>();
 CxList allPublicFields = allFields.FindByFieldAttributes(Modifiers.Public);
 CxList allConstFields = allFields.FindByFieldAttributes(Modifiers.Readonly);
 result = allPublicFields - allConstFields;
```

CSharp / CSharp_Best_Coding_Practice / GetLastWin32Error_Is_Not_Called_After_Pinvoke

Code changes

```
---
+++
@@ -1,8 +1,8 @@
-CxList importedMethods = All.FindByCustomAttribute("DllImport").GetAncOfType(typeof(MethodDecl));
+CxList importedMethods = Find_CustomAttribute().FindByCustomAttribute("DllImport").GetAncOfType<MethodDecl>();
 CxList allImportedMethodsCalls = All.FindAllReferences(importedMethods) - importedMethods;
 CxList allGLWECalls = All.FindAllReferences(All.FindByName("*GetLastWin32Error"));
-CxList allMethodsThatCallGLWE = allGLWECalls.GetAncOfType(typeof(MethodDecl));
-CxList allMethodsThatCallImportedMethods = allImportedMethodsCalls.GetAncOfType(typeof(MethodDecl));
+CxList allMethodsThatCallGLWE = allGLWECalls.GetAncOfType<MethodDecl>();
+CxList allMethodsThatCallImportedMethods = allImportedMethodsCalls.GetAncOfType<MethodDecl>();

 result = allImportedMethodsCalls.GetByAncs(allMethodsThatCallImportedMethods - allMethodsThatCallGLWE);
-result = result.FindByType(typeof(MethodInvokeExpr));
+result = result.FindByType<MethodInvokeExpr>();
```

CSharp / CSharp_Best_Coding_Practice / Hardcoded_Connection_String

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList conStr = Find_Strings().FindByName("*Provider*", false);
 CxList openConnection = Find_ObjectCreations().FindByShortName("*Connection");
-CxList openConParam = All.GetParameters(openConnection);
+CxList openConParam = Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(openConnection);

 result = conStr.DataInfluencingOn(openConParam);
```

CSharp / CSharp_Best_Coding_Practice / Insufficient_Logging_of_Sensitive_Operations

Code changes

```
---
+++
@@ -4,7 +4,8 @@
 CxList sensitiveOperationsNotLogged = Common_Best_Coding_Practice.Insufficient_Logging_of_Sensitive_Operations();

 // Test Tags
-CxList testTags = attributes.FindByShortNames(new List<string>{"Test", "TestCase", "TestCaseSource", "TestFixture", "TestFixtureSource"});
+CxList testTags = attributes.FindByShortNames(
+   new []{"Test", "TestCase", "TestCaseSource", "TestFixture", "TestFixtureSource"});

 // Test Classes / Operations
 CxList testClasses = testTags.GetAncOfType<ClassDecl>();
```

CSharp / CSharp_Best_Coding_Practice / Leftover_Debug_Code

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 if(All.isWebApplication)
 {
-   result = All.FindByName("*.Main").FindByType(typeof(MethodDecl))
+   result = All.FindByName("*.Main").FindByType<MethodDecl>()
         .FindByFieldAttributes(Modifiers.Public | Modifiers.Static);
 }
```

CSharp / CSharp_Best_Coding_Practice / Pages_Without_Global_Error_Handler

Code changes

```
---
+++
@@ -9,6 +9,6 @@
 {

    CxList errorHandledPages = All.GetClass(All.FindByName("*Page.ErrorPage"));
-   CxList AllPages = All.GetClass(All.FindByName("*Page_Load*").FindByType(typeof(MethodDecl)));
+   CxList AllPages = All.GetClass(All.FindByName("*Page_Load*").FindByType<MethodDecl>());
    result = AllPages - errorHandledPages;
 }
```

CSharp / CSharp_Best_Coding_Practice / PersistSecurityInfo_is_True

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList PersistSecurityInfo = All.FindByRegex(@"Persist Security Info(\s)*=(\s)*(True|Yes)");

 CxList openConnection = Find_ObjectCreations().FindByShortName("*Connection");
-CxList openConParam = All.GetParameters(openConnection);
+CxList openConParam = Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(openConnection);

-result = openConParam.DataInfluencedBy(PersistSecurityInfo.FindByType(typeof(StringLiteral)));
+result = openConParam.DataInfluencedBy(PersistSecurityInfo.FindByType<StringLiteral>());
```

CSharp / CSharp_Best_Coding_Practice / Unclosed_Objects

Code changes

```
---
+++
@@ -1,12 +1,12 @@
 CxList close = Find_Methods().FindByName("*.close", false);
 CxList closeTarget = close.GetTargetOfMembers();
 CxList definitions = All.FindDefinition(closeTarget);
-CxList methods = All.GetMethod(closeTarget);
+CxList methods = Find_MethodDecls().GetMethod(closeTarget);
 foreach (CxList target in closeTarget)
```

```
 {
     CxList method = methods.GetMethod(target);
     CxList targetDefinition = definitions.FindDefinition(target);
-    CxList targetInForEach = targetDefinition.GetAncOfType(typeof(ForEachStmt));
+    CxList targetInForEach = targetDefinition.GetAncOfType<ForEachStmt>();
     CxList methodDef = method.GetMethod(targetDefinition);
     if (targetInForEach.Count > 0 || methodDef.Count == 0)
     {
@@ -14,27 +14,19 @@
     }
 }

-CxList AllTrys = All.GetAncOfType(typeof(TryCatchFinallyStmt));
-CxList fin = All.NewCxList();
-
-foreach(CxList oneTry in AllTrys)
-{
-    TryCatchFinallyStmt t = oneTry.TryGetCSharpGraph<TryCatchFinallyStmt>();
-    fin.Add(All.FindById(t.Finally.NodeId));
-}
+CxList fin = Find_TryCatchFinallyStmt().CxSelectDomProperty<TryCatchFinallyStmt>(x => x.Finally);
 fin = All.GetByAncs(fin);

-CxList Try = close.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList Try = close.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList oneTry in Try)
 {
-    TryCatchFinallyStmt TryGraph = oneTry.TryGetCSharpGraph<TryCatchFinallyStmt>();
-    CxList curTry = All.FindById(TryGraph.Try.NodeId);
+    CxList curTry = oneTry.CxSelectDomProperty<TryCatchFinallyStmt>(x => x.Try);
     CxList TryClose = close.GetByAncs(curTry);
     CxList AllClose = close.GetByAncs(oneTry);

     if( (AllClose - TryClose).Count == 0)
     {
-        if (TryClose.GetAncOfType(typeof(UsingStmt)).Count == 0)
+        if (TryClose.GetAncOfType<UsingStmt>().Count == 0)
         {
             result.Add(TryClose);
         }
```

CSharp / CSharp_Best_Coding_Practice / Undocumented_API

Code changes

```
---
+++
@@ -7,7 +7,7 @@

 // See if a Swagger tool is enabled
```

```
-CxList swaggerUse = methods.FindByShortNames(new List<string>() {
+CxList swaggerUse = methods.FindByShortNames(new [] {
    // Swashbuckle.AspNetCore
    "UseSwagger", "UseSwaggerUI", "AddSwaggerGen", "SwaggerDoc",
    // Swashbuckle
```

CSharp / CSharp_Best_Coding_Practice / Unvalidated_Arguments_Of_Public_Methods

Code changes

```
---
+++
@@ -3,14 +3,13 @@
 allUnkRef.Add(paramDecl);

 CxList memberAccess = Find_MemberAccesses().GetTargetOfMembers().FindByType<UnknownReference>();
-CxList paramDeclMemberAccess = memberAccess.Clone();
-paramDeclMemberAccess.Add(paramDecl);
+CxList paramDeclMemberAccess = All.NewCxList(memberAccess, paramDecl);

 CxList arrayTypes = Find_ArrayTypes();

 CxList paramDeclsArrayTypes = arrayTypes.GetAncOfType<ParamDecl>() * paramDecl;

-CxList publicMethods = All.FindByFieldAttributes(Modifiers.Public).FindByType<MethodDecl>();
+CxList publicMethods = Find_MethodDecls().FindByFieldAttributes(Modifiers.Public);

 CxList allIfStmt = Find_Ifs();
 CxList allTryCatchFinallyStmt = Find_TryCatchFinallyStmt();
@@ -35,9 +34,9 @@
 CxList paramsInPublicMethods = paramDecl.GetByAncs(publicMethodsWithParams);
 CxList nullInIfStmt = Find_NullLiteral().GetByAncs(allIfStmt);

-CxList checkedParamsInPublicMethods = paramsInPublicMethods.Clone();
+CxList checkedParamsInPublicMethods = All.NewCxList(paramsInPublicMethods);
 CxList IsNull = All.FindByShortName("IsNull*");
-CxList isNullParameters = All.GetParameters(IsNull);
+CxList isNullParameters = Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(IsNull);

 // Caluculate CxLists that will be used inside the loop (performance optimization)
 CxList params0 = paramDecl.GetByAncs(publicMethodsWithParams);
@@ -58,8 +57,9 @@
         CxList paramMemberAccessInMethodReferences = paramsInMethodReferences.FindAllReferences(paramInMethod);
         CxList paramInMethodReferences = paramMemberAccessReferences.FindAllReferences(paramInMethod);

-        CxList paramsToRemove = All.NewCxList();
-        paramsToRemove.Add(paramInMethod, paramMemberAccessInMethodReferences.GetByAncs(allMethodInvokeExpr),
+        CxList paramsToRemove = All.NewCxList(
+            paramInMethod,
```

```
+           paramMemberAccessInMethodReferences.GetByAncs(allMethodInvokeExpr),
            paramMemberAccessInMethodReferences.GetByAncs(allTryCatchFinallyStmt));


        paramMemberAccessInMethodReferences -= paramsToRemove;
```

CSharp / CSharp_Best_Coding_Practice / Use_of_System_Output_Stream

Code changes

```
---
+++
@@ -1,5 +1,4 @@
 if(All.isWebApplication)
 {
-   CxList methods = Find_Methods();
-   result = methods.FindByName("*Out.Write*") + methods.FindByName("*Error.Write*");
+   result = Find_Methods().FindByNames(new []{"*Out.Write*", "*Error.Write*"});
 }
```

CSharp / CSharp_Best_Coding_Practice / Using_Of_Index_Instead_Of_Key

Code changes

```
---
+++
@@ -1,5 +1,4 @@
 // using of index as key in SortedList or Dictionary
 string regexString = @"\[[^\]]+\.Count[^\]]*\]";
-result.Add(All.FindByType("SortedList").FindByRegex(regexString));
-result.Add(All.FindByType("Dictionary").FindByRegex(regexString));
+result.Add(All.FindByTypes(new []{"SortedList", "Dictionary"}).FindByRegex(regexString));
```

CSharp / CSharp_Best_Coding_Practice / Visible_Pointers

Code changes

```
---
+++
@@ -4,13 +4,13 @@
 CxList allProtectedFields = allFields.FindByFieldAttributes(Modifiers.Protected);
 CxList allInternalFields = allFields.FindByFieldAttributes(Modifiers.Internal);
 CxList allReadonlyFields = allFields.FindByFieldAttributes(Modifiers.Readonly);
-CxList allExposedFields = ((allPublicFields + allProtectedFields) -
+CxList allExposedFields = (All.NewCxList(allPublicFields, allProtectedFields) -
     allInternalFields) - allReadonlyFields;

-CxList allIntPtr = All.FindByType("IntPtr").GetAncOfType(typeof(FieldDecl));
+CxList allIntPtr = All.FindByType("IntPtr").GetAncOfType<FieldDecl>();
 allIntPtr.Add(allFields.FindByType("nint"));

-CxList allUIntPtr = All.FindByType("UIntPtr").GetAncOfType(typeof(FieldDecl));
+CxList allUIntPtr = All.FindByType("UIntPtr").GetAncOfType<FieldDecl>();
 allUIntPtr.Add(allFields.FindByType("nuint"));
```

```
-result = (allIntPtr + allUIntPtr) * allExposedFields;
+result = All.NewCxList(allIntPtr, allUIntPtr) * allExposedFields;
```

CSharp / CSharp_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,11 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-possible_db -= possible_db.DataInfluencedBy(possible_db);
-
-if (possible_db.Count > 0)
-{
-   CxList db = Find_DB_Base() + Find_Read();
-   CxList dbParams = All.GetParameters(db);
-   CxList sanitize = Find_SQL_Sanitize();
-   result = possible_db.InfluencingOnAndNotSanitized(possible_db + dbParams, sanitize);
-   result.Add(db.InfluencingOnAndNotSanitized(possible_db, sanitize));
-}
+//This query is deprecated.
```

CSharp / CSharp_Heuristic / Heuristic_CSRF

Code changes

```
---
+++
@@ -1,19 +1 @@
-if (All.isWebApplication)
-{
-   CxList possible_db = Find_DB_Heuristic();
-
-   // Exclude cache related methods
-   possible_db -= possible_db.FindByNames(new string []{"System.Web.HttpRuntime.Cache.*", "*Cache.GetCacheItem"});
-
-   possible_db -= possible_db.DataInfluencedBy(possible_db);
-
-   if (possible_db.Count > 0)
-   {
-       CxList requests = Find_Interactive_Inputs();
-       requests.Add(All.FindByName("*Request.QueryString*"));
-       CxList strings = Find_Strings();
-       CxList write = strings.FindByNames(new string [] {"update*", "delete*", "insert*"}
-           , StringComparison.OrdinalIgnoreCase);
-       result = possible_db.DataInfluencedBy(write).DataInfluencedBy(requests);
-   }
-}
+//This query is deprecated.
```

CSharp / CSharp_Heuristic / Heuristic_DB_Parameter_Tampering

Code changes

```
---

+++

@@ -1 +1 @@

-result = Find_Heuristic_DB_Parameter_Tampering();

+//This query is deprecated.
```

CSharp / CSharp_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---

+++

@@ -1,18 +1 @@

-CxList possible_db = Find_DB_Heuristic();

-possible_db -= possible_db.DataInfluencedBy(possible_db);

-

-if (possible_db.Count > 0)

-{

-   CxList input = Find_Interactive_Inputs();

-

-   CxList strings = Find_Strings();

-   CxList Select = strings.FindByName("*select*", false);

-   CxList Where = strings.FindByName("*where*", false);

-   CxList And = strings.FindByName("*And *", false) +

-       strings.FindByName("* And*", false);

-

-   possible_db = possible_db.DataInfluencedBy(Select).DataInfluencedBy(Where);

-   possible_db -= possible_db.DataInfluencedBy(And);

-

-   result = possible_db.DataInfluencedBy(input);

-}

+//This query is deprecated.
```

CSharp / CSharp_Heuristic / Heuristic_SQL_Injection

Code changes

```
---

+++

@@ -1,10 +1 @@

-CxList possible_db = Find_DB_Heuristic();

-possible_db -= possible_db.DataInfluencedBy(possible_db);

-

-if (possible_db.Count > 0)

-{

-   CxList inputs = Find_Interactive_Inputs();

-   CxList sanitized = Find_SQL_Sanitize();

-

-   result = inputs.InfluencingOnAndNotSanitized(possible_db, sanitized);

-}
```

```
+//This query is deprecated.
```

CSharp / CSharp_Heuristic / Heuristic_Stored_XSS

Code changes

```
---
+++
@@ -1,14 +1 @@
-if(All.isWebApplication || Check_Web_Application().Any())
-{
-   CxList possible_db = Find_DB_Heuristic();
-   possible_db -= possible_db.DataInfluencedBy(possible_db);
-   possible_db -= Find_Read();
-
-   if (possible_db.Count > 0)
-   {
-       CxList outputs = Find_XSS_Outputs();
-       CxList sanitize = Find_XSS_Sanitize();
-
-       result = (possible_db).InfluencingOnAndNotSanitized(outputs, sanitize);
-   }
-}
+//This query is deprecated.
```

CSharp / CSharp_High_Risk / Connection_String_Injection

Code changes

```
---
+++
@@ -2,5 +2,4 @@
 CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList sanitize = Find_Connection_String_Sanitize();

-result = con.InfluencedByAndNotSanitized(inputs, sanitize);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result = con.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

CSharp / CSharp_High_Risk / JWT_No_Signature_Verification

Code changes

```
---
+++
@@ -1,11 +1,9 @@
 CxList tokenValidation = JWT_TokenValidationParameters();

-CxList listToValidate = All.NewCxList();
-CxList memberAccess = tokenValidation.FindByType(typeof(MemberAccess)).FindByShortName("RequireSignedTokens");
-CxList fieldDecl = tokenValidation.FindByType(typeof(FieldDecl)).FindByShortName("RequireSignedTokens");
+CxList memberAccess = tokenValidation.FindByType<MemberAccess>().FindByShortName("RequireSignedTokens");
+CxList fieldDecl = tokenValidation.FindByType<FieldDecl>().FindByShortName("RequireSignedTokens");
```

```
-listToValidate.Add(memberAccess);
-listToValidate.Add(fieldDecl);
+CxList listToValidate = All.NewCxList(memberAccess, fieldDecl);

 CxList vulnerable = JWT_TokenParameter_Validation(listToValidate);
```

CSharp / CSharp_High_Risk / Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -11,9 +11,8 @@
    CxList sanitized = Find_XSS_Sanitize();

    //Add LINQ with HttpUtility.HtmlEncode as sanitizer
-   CxList linqWhereAndSelect = methods.FindByShortNames(new List<string> {"where", "select"}, false);
-   CxList lambdaExpr = Find_LambdaExpr();
-   CxList lambdaMethods = methods.GetByAncs(lambdaExpr);
+   CxList linqWhereAndSelect = methods.FindByShortNames(new [] {"where", "select"}, false);
+   CxList lambdaMethods = methods.GetByAncs(Find_LambdaExpr());
    CxList isHtmlEncode = lambdaMethods.FindByMemberAccess("HttpUtility.HtmlEncode");
    CxList linqSanitizers = linqWhereAndSelect.FindByParameters(
        isHtmlEncode.GetByAncs(parameters.GetParameters(linqWhereAndSelect)).GetAncOfType<LambdaExpr>()
```

CSharp / CSharp_High_Risk / Resource_Injection

Code changes

```
---
+++
@@ -1,10 +1,7 @@
-CxList createExpr = Find_ObjectCreations();
-
 CxList inputs = Find_Interactive_Inputs();

-CxList outputs = All.GetByAncs(createExpr.FindByShortNames(new List<string>{"TcpListener","UdpClient"}));
+CxList outputs = All.GetByAncs(Find_ObjectCreations().FindByShortNames(new []{"TcpListener","UdpClient"}));

 CxList sanitizers = Find_Resource_Injection_Sanitizers();

-result = inputs.InfluencingOnAndNotSanitized(outputs, sanitizers)
-   .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
+result = inputs.InfluencingOnAndNotSanitized(outputs, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

CSharp / CSharp_High_Risk / Unsafe_Reflection

Code changes

```
---
+++
@@ -1,4 +1,5 @@
 CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Interactive_Inputs());
```

```
+CxList parameters = Find_Param();
 CxList comCreations = Find_ObjectCreations().FindByTypes(new String[]{
    "CodeSnippetExpression",
    "CodeEntryPointMethod",
@@ -10,14 +11,13 @@
    "CodeMethodReturnStatement",
    "CodeMemberMethod"});

-CxList unknownList = All.NewCxList();
-unknownList.Add(
+CxList unknownList = All.NewCxList(
    Find_UnknownReference(),
-   Find_Param(),
+   parameters,
    Find_TypeRef());

-CxList auxInput = All.NewCxList();
-auxInput.Add(
+
+CxList auxInput = All.NewCxList(
    inputs,
    unknownList.FindAllReferences(inputs.GetAssignee()));

@@ -39,10 +39,10 @@
 notInfluencedChecks.Add(All.FindAllReferences(notInfluencedChecks.GetAssignee()));

 CxList notInfluencedComCreations = comCreations - comCreations.FindByParameters(allReferencesOfifParams);
-CxList objsC = notInfluencedComCreations.FindByParameters(All.GetParameters(notInfluencedComCreations) * auxInput);
+CxList objsC = notInfluencedComCreations.FindByParameters(
+   parameters.CxSelectDomProperty<Param>(p => p.Value).GetParameters(notInfluencedComCreations) * auxInput);

-CxList sinks = All.NewCxList();
-sinks.Add(notInfluencedChecks, objsC);
+CxList sinks = All.NewCxList(notInfluencedChecks, objsC);

 CxList sanitizers = Find_Integers();
```

CSharp / CSharp_High_Risk / UTF7_XSS

Code changes

```
---
+++
@@ -1,42 +1 @@
-if(All.isWebApplication || Check_Web_Application().Any())
-{
-   CxList UTF7 = Find_Strings().FindByName("UTF-7");
-   CxList response = All.FindByName("*Response.Charset");
-
```

```
-   UTF7 = response.DataInfluencedBy(UTF7);
-
-   // get last node of the path (this node is part of response CxList
-   CxList temp = UTF7.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-
-   // take from response only nodes that influenced by UTF7
-   response = temp;
-
-   if(UTF7.Count > 0)
-   {
-       CxList outputs = Find_XSS_Outputs();
-       CxList inputs = Find_Interactive_Inputs();
-       CxList sanitize = Find_UTF7_XSS_Sanitize();
-       CxList tempInputs = All.NewCxList();
-
-       //limit to inputs in the same class as "UTF-7"
-       foreach (CxList r in response)
-       {
-           CxList responseClass = r.GetAncOfType(typeof(ClassDecl));
-
-           foreach (CxList i in inputs)
-           {
-               CxList inputsClass = i.GetAncOfType(typeof(ClassDecl));
-               CxList sameClass = responseClass * inputsClass;
-
-               if (sameClass.Count > 0)
-               {
-                   tempInputs.Add(i);
-               }
-           }
-       }
-
-       inputs = tempInputs;
-
-       result = outputs.InfluencedByAndNotSanitized(inputs, sanitize);
-   }
-}
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_Blind_SQL_Injections();
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Cleansing_Canonicalization_and_Comparison_Errors

Code changes

```
---
+++
@@ -1,18 +1 @@
-//This query finds attempts to open file while using not normalized file name
-CxList inputs = Find_Interactive_Inputs();
-CxList obj = Find_Unknown_References();
-obj.Add(All.FindByType(typeof(Declarator)));
-
-CxList files = obj.FindByTypes(new String[] {"FileStream", "FileInfo",
-   "*.Filestream", "*.FileInfo"});
-//files.Add(All.FindByName("*File.*"));
-files.Add(All.FindByName("*.File.*"));
-
-CxList filesMethods = files.GetMembersOfTarget();;
-filesMethods = filesMethods.FindByShortName("Close") + filesMethods.FindByShortName("Dispose");
-files -= filesMethods.GetTargetOfMembers();
-
-CxList sanitize = All.FindByName("*Server.MapPath") + All.FindByName("*Request.MapPath");
-
-result = files.InfluencedByAndNotSanitized(inputs, sanitize);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Client_Side_Only_Validation

Code changes

```
---
+++
@@ -2,7 +2,7 @@
 CxList methods = Find_Methods();
 CxList memberAccess = Find_MemberAccesses();

-var validatorsList = new String[] {
+string[] validatorsList = new string[] {
     "RequiredFieldValidator",
     "RangeValidator",
     "RegularExpressionValidator",
@@ -11,16 +11,17 @@

 CxList ClientValidators = All.FindByTypes(validatorsList);

-CxList ClientValidatorsDeclarations = classDecls.FindByShortNames(validatorsList.ToList());
+CxList ClientValidatorsDeclarations = classDecls.FindByShortNames(validatorsList);

 CxList PagesWithClientValidators = classDecls.GetClass(ClientValidators) - ClientValidatorsDeclarations;

-CxList ServerValidators = memberAccess.FindByMemberAccess("Page.IsValid");
```

```
-ServerValidators.Add(memberAccess.FindByMemberAccess("Page.Validators.IsValid"));
-ServerValidators.Add(methods.FindByMemberAccess("Page.Validate"));
+CxList ServerValidators = All.NewCxList(
+   memberAccess.FindByMemberAccesses(new [] {"Page.IsValid", "Page.Validators.IsValid"}),
+   methods.FindByMemberAccess("Page.Validate"));
+
 CxList PagesWithServerValidators = classDecls.GetClass(ServerValidators);

-CxList relevantPages = PagesWithServerValidators.Clone();
+CxList relevantPages = All.NewCxList(PagesWithServerValidators);

 foreach(CxList curPagesWithServerValidators in relevantPages)
 {
```

CSharp / CSharp_Low_Visibility / Heap_Inspection

Code changes

```
---
+++
@@ -14,23 +14,14 @@
 */
 // Remove passwords inside designer files
 passwords = passwords.FindByFileName("*.cs");
-passwords -= passwords.FindByFileName("*.designer.cs");
-passwords -= passwords.FindByFileName("*.Designer.cs");
+passwords -= passwords.FindByFileNames(new []{"*.designer.cs", "*.Designer.cs"});

 //Remove potential passwords which name contains Encrypt
 passwords -= passwords.FindByShortName("*Encrypt*");

 //Exclude variables that are all uppercase - usually describes the pattern of the data,
 //such as PASSWORDPATTERN, PASSWORDTYPE...
-CxList upperCase = All.NewCxList();
-foreach (CxList res in passwords)
-{
-   string name = res.GetName();
-   if (name.ToUpper().Equals(name))
-   {
-       upperCase.Add(res);
-   }
-}
+CxList upperCase = passwords.Filter(x => x.ShortName.ToUpper().Equals(x.ShortName));
 passwords -= upperCase;

 //Get string and char[] objetcs
@@ -39,15 +30,14 @@

 CxList pwdStrings = passwords.FindByTypes(primitivesString);
 pwdStrings.Add(passwords * All.FindByMemberAccess("*.ToString").GetAssignee());
```

```
-CxList passwordsValid = pwdStrings.FindByType(typeof(Declarator));
-passwordsValid.Add(pwdStrings.FindByType(typeof(PropertyDecl)));
+CxList passwordsValid = pwdStrings.FindByTypes(typeof(Declarator), typeof(PropertyDecl));

 //char[]
 CxList pwdCharArr = passwords.FindByTypes("char[]", "Char[]");
 pwdCharArr.Add(Find_ArrayTypes().FindByTypes("char", "Char")
-   .GetByAncs(passwords.GetAncOfType(typeof(VariableDeclStmt))));
+   .GetByAncs(passwords.GetAncOfType<VariableDeclStmt>()));

-passwordsValid.Add(pwdCharArr.GetAncOfType(typeof(VariableDeclStmt)));
+passwordsValid.Add(pwdCharArr.GetAncOfType<VariableDeclStmt>());

 /*
    Sanitizers
@@ -55,13 +45,13 @@
 CxList influencedByEncrypt = passwordsValid.InfluencedBy(encrypt);
 CxList influencingOnDecrypt = passwordsValid.InfluencingOn(decrypt);

-CxList sanitizers = influencedByEncrypt.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-sanitizers.Add(influencingOnDecrypt.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly));
+CxList sanitizers = influencedByEncrypt.GetLastNodesInPath();
+sanitizers.Add(influencingOnDecrypt.GetFirstNodesInPath());
 passwordsValid -= sanitizers;

 //net core
-CxList sanitizerProtector = encrypt.FindByType(typeof(MethodInvokeExpr)).FindByShortName("Protect");
+CxList sanitizerProtector = encrypt.FindByType<MethodInvokeExpr>().FindByShortName("Protect");
 CxList sanitizeObj = sanitizerProtector.GetAssignee();
 sanitizeObj.Add(All.FindAllReferences(sanitizeObj));
 passwordsValid -= sanitizeObj;
@@ -74,10 +64,10 @@

 //Remove PropertyDecls with accessors' implementation
 CxList autoImplementedPropertiesRefs = unknownList.FindAllReferences(cxImplementedDecl);
-CxList autoImplementedProperties = autoImplementedPropertiesRefs.GetAncOfType(typeof(PropertyDecl));
-CxList accessorDecls = All.FindByType(typeof(AccessorDecl));
+CxList autoImplementedProperties = autoImplementedPropertiesRefs.GetAncOfType<PropertyDecl>();
+CxList accessorDecls = Find_AccessorDecls();
 CxList statementsInsideAccessorDecls = statements.GetByAncs(accessorDecls);
-CxList propertyDeclsWithStatements = statementsInsideAccessorDecls.GetAncOfType(typeof(PropertyDecl)) - autoImplementedProperties;
+CxList propertyDeclsWithStatements = statementsInsideAccessorDecls.GetAncOfType<PropertyDecl>() - autoImplementedProperties;
 passwordsValid -= propertyDeclsWithStatements;

 //Remove password as property name
@@ -88,11 +78,11 @@
 CxList arrayClear = methodsList.FindByName("*Array.Clear");
```

```
 CxList firstParamClear = unknownList.GetParameters(arrayClear, 0);
 CxList firstParamDeclarators = declaratorsList.FindDefinition(firstParamClear);
-passwordsValid -= firstParamDeclarators.GetAncOfType(typeof(VariableDeclStmt));
+passwordsValid -= firstParamDeclarators.GetAncOfType<VariableDeclStmt>();

 //second case - array create expr
 CxList assignVar = arrayCreateList.GetAssignee();
 CxList assignVarDeclarators = declaratorsList.FindDefinition(assignVar);
-passwordsValid -= assignVarDeclarators.GetAncOfType(typeof(VariableDeclStmt));
+passwordsValid -= assignVarDeclarators.GetAncOfType<VariableDeclStmt>();

 result = passwordsValid;
```

CSharp / CSharp_Low_Visibility / Impersonation_Issue

Code changes

```
---
+++
@@ -1,21 +1,22 @@
 CxList Inputs = Find_Inputs();
 CxList methods = Find_Methods();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
 CxList Logon = methods.FindByShortName("LogonUser");

 CxList DuplicateToken = methods.FindByShortName("DuplicateToken");
-CxList Impersonate = All.FindByMemberAccess("WindowsIdentity.Impersonate");
+CxList Impersonate = methods.FindByMemberAccess("WindowsIdentity.Impersonate");
 Impersonate.Add(All.FindByShortName("ImpersonateLoggedOnUser"));

 //only parameters of logonuser methods that are influenced by inputs
-CxList LogonParams = All.GetParameters(Logon);     //all parameters of LogonUser method
+CxList LogonParams = paramValue.GetParameters(Logon);      //all parameters of LogonUser method
 CxList LogonParamsIn = LogonParams.DataInfluencedBy(Inputs);   //parameters of LogonUser method that influenced by input
-LogonParamsIn = LogonParamsIn.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
+LogonParamsIn = LogonParamsIn.GetLastNodesInPath();

 //only parameters of duplicatetoken methods that are influenced by logonuser methods parameters
-CxList DuplicateTokenParams = All.GetParameters(DuplicateToken);   //all parameters of DuplicateToken method
+CxList DuplicateTokenParams = paramValue.GetParameters(DuplicateToken);    //all parameters of DuplicateToken method
 CxList DuplicateTokenParamsLP = DuplicateTokenParams.DataInfluencedBy(LogonParams); //parameters of DuplicateToken method that influenced by input
-DuplicateTokenParams = DuplicateTokenParams.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
+DuplicateTokenParams = DuplicateTokenParams.GetLastNodesInPath();

 CxList prms = All.NewCxList();
```

CSharp / CSharp_Low_Visibility / Improper_Resource_Shutdown_or_Release

Code changes

```
---
+++
@@ -21,7 +21,7 @@
        };

 // Static methods
-List<string> ObjectStaticMethods = new List<string>()
+string[] ObjectStaticMethods = new string[]
    {
        "Create","OpenText","CreateText",
        "CreateCommand","ExecuteReader",
@@ -30,7 +30,7 @@
        };



-List<string> ResourceCloseMethods = new List<string>() {"Close","Dispose","DisposeAsync","DisposeAsyncCore"};
+string[] ResourceCloseMethods = new string[] {"Close","Dispose","DisposeAsync","DisposeAsyncCore"};

 List<string> ResourceTypeNames = new List<string>();
 ResourceTypeNames.AddRange(dbResourceTypes);
@@ -53,9 +53,6 @@
 }

 /* Collect all the instances of allocated resources */
-CxList objectQueryInherits = All.NewCxList();
-objectQueryInherits.Add(ClassInheritsFrom);
-
 CxList typeOfObjectQuery = All.FindByTypes(ResourceTypeNames.ToArray());

 typeOfObjectQuery.Add(TypeRef.FindAllReferences(ClassInheritsFrom).GetFathers());
@@ -132,23 +129,7 @@
 /* Collect the Try statements Block */
 CxList TryEnds = ResourceAllocationInstances.GetLastNodesInPath();

-CxList TryBlocks = All.NewCxList();
-foreach(CxList tryCatch in Trys){
-   try
-   {
-       TryCatchFinallyStmt tryGraph = tryCatch.TryGetCSharpGraph<TryCatchFinallyStmt>();
-       if(tryGraph.Try != null)
-       {
-           TryBlocks.Add(tryGraph.Try.NodeId, tryGraph.Try);
-       }
-   }
-   catch(Exception ex)
-   {
-       cxLog.WriteDebugMessage(ex);
-   }
```

```
-}
-
-
+CxList TryBlocks = Trys.CxSelectDomProperty<TryCatchFinallyStmt>(x => x.Try);

 CxList TryBlock = All.NewCxList(ResourceAllocationInstances.GetFirstNodesInPath(), wrappingObjects).GetByAncs(TryBlocks);
```

CSharp / CSharp_Low_Visibility / Improper_Session_Management

Code changes

```
---
+++
@@ -1,5 +1,4 @@
-CxList sid = All.FindByName("*Request.QueryString_SID*", false);
-sid.Add(All.FindByName("*Request.QueryString_Session*", false));
+CxList sid = All.FindByNames(new []{"*Request.QueryString_SID*", "*Request.QueryString_Session*"}, false);

 CxList queryString = All.FindByMemberAccess("HttpRequest.QueryString_*");
 queryString.Add(All.FindByName("*Request.QueryString_*"));
```

CSharp / CSharp_Low_Visibility / Improper_Transaction_Handling

Code changes

```
---
+++
@@ -1,24 +1,20 @@
 CxList Commit = All.FindByName("*.Commit");
 CxList Rollback = All.FindByName("*.Rollback");

-CxList TryBlock = Commit.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList TryBlock = Commit.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList cml in TryBlock)
 {
     TryCatchFinallyStmt TryGraph = cml.TryGetCSharpGraph<TryCatchFinallyStmt>();
-
+
     CxList curTry = All.FindById(TryGraph.Try.NodeId);

     CxList curCatch = All.NewCxList();
     if(TryGraph.CatchClauses != null && TryGraph.CatchClauses.Count > 0)
-    {
         curCatch = All.FindById(TryGraph.CatchClauses[0].NodeId);
-    }

     CxList curFinally = All.NewCxList();
-    if(TryGraph.CatchClauses != null && TryGraph.Finally.Count > 0)
-    {
+    if(TryGraph.CatchClauses != null && TryGraph.Finally != null && TryGraph.Finally.Count > 0)
         curFinally = All.FindById(TryGraph.Finally.NodeId);
```

```
-   }


    CxList CommitInTry = Commit.GetByAncs(curTry);
    CxList RollbackInCatch = Rollback.GetByAncs(curCatch);
@@ -26,11 +22,11 @@


    if(
        (
-           (
-               RollbackInCatch.GetAncOfType(typeof(TryCatchFinallyStmt)) +
-               RollbackInFinally.GetAncOfType(typeof(TryCatchFinallyStmt))
+           All.NewCxList(
+               RollbackInCatch.GetAncOfType<TryCatchFinallyStmt>(),
+               RollbackInFinally.GetAncOfType<TryCatchFinallyStmt>()
            ) *
-               CommitInTry.GetAncOfType(typeof(TryCatchFinallyStmt))
+               CommitInTry.GetAncOfType<TryCatchFinallyStmt>()
                                                                    ).Count == 0
        )
    {
```

CSharp / CSharp_Low_Visibility / Information_Exposure_Through_an_Error_Message

Code changes

```
---
+++
@@ -1,6 +1,7 @@
-CxList outputs = Find_Outputs() - Find_Console_Outputs() - Find_Log_Outputs();
+CxList outputs = Find_Outputs();
+outputs -= All.NewCxList(Find_Console_Outputs(), Find_Log_Outputs());
 CxList unknownRef = Find_Unknown_References();
-CxList exc = All.FindByType("*Exception").FindByType(typeof(Declarator));
+CxList exc = All.FindByType("*Exception").FindByType<Declarator>();
 exc = (All - exc).FindAllReferences(exc);


 if(!All.isWebApplication)
@@ -13,21 +14,21 @@


    CxList classes_with_main = All.GetClass(main_decl);


-   CxList ctch = All.FindByType(typeof(Catch));
+   CxList ctch = Find_Catch();
    CxList class_of_ctch_not_with_main = (All - classes_with_main).GetClass(ctch);
    ctch = ctch.GetByAncs(class_of_ctch_not_with_main);


    CxList class_not_with_main = Find_ClassDecl() - classes_with_main;
    class_not_with_main = All.GetByAncs(class_not_with_main);


-   exc.Add(class_not_with_main.FindByName("*Server.GetLastError*"));
```

```
-   exc.Add(class_not_with_main.FindByName("*InnerException*"));
+   exc.Add(class_not_with_main.FindByNames(new []{"*Server.GetLastError*", "*InnerException*"}));
 }
 result = outputs.DataInfluencedBy(exc).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);


-CxList methodsToFilter = unknownRef.FindByTypes(new String[]{"IApplicationBuilder","IHostingEnvironment", "IWebHostEnvironment"}).GetMembersOfTarget();
+CxList methodsToFilter = unknownRef.FindByTypes(new []{
+       "IApplicationBuilder","IHostingEnvironment", "IWebHostEnvironment"}).GetMembersOfTarget();
 CxList isDelopment = methodsToFilter.FindByShortName("IsDevelopment");
-CxList elementIf = All.GetByAncs(isDelopment.GetAncOfType(typeof(IfStmt)));
+CxList elementIf = All.GetByAncs(isDelopment.GetAncOfType<IfStmt>());
 CxList methodUseDeveloperExceptionPage = methodsToFilter.FindByShortName("UseDeveloperExceptionPage");
 CxList sanitizeHandlerException = methodUseDeveloperExceptionPage * elementIf;
```

CSharp / CSharp_Low_Visibility / Information_Exposure_via_Headers

Code changes

```
---
+++
@@ -13,21 +13,21 @@
    CxList httpRunTime = (webConfig * unknownList).FindByName("httpRuntime", false);
    CxList enableVersionList = (webConfig * memberList).FindByShortName("enableVersionHeader", false);
    CxList enableVersionHeaderSanitizer = (enableVersionList.GetAssigner() * stringList.FindByName("false"));
-   CxList ifSanitized = enableVersionHeaderSanitizer.GetAncOfType(typeof(IfStmt));
+   CxList ifSanitized = enableVersionHeaderSanitizer.GetAncOfType<IfStmt>();
    result.Add(httpRunTime - httpRunTime.FindByFathers(ifSanitized));
    //X-AspNetMvc-Version
    if ( All.FindByFileNames(@"*.cshtml", @"*.aspx").Count > 0 &&
    All.FindByFileNames(@"*controllers*", @"*views*").Count > 0){
        CxList sanitizerMvc = memberList.FindByName("MvcHandler.DisableMvcResponseHeader") * trueList;
        CxList applicationStart = globals.FindByShortName("Application_Start");
-       result.Add(applicationStart - sanitizerMvc.GetAncOfType(typeof(MethodDecl)));
+       result.Add(applicationStart - sanitizerMvc.GetAncOfType<MethodDecl>());
    }
    //server
    CxList PreSendRequestHeaders = globals.FindByShortName("Application_PreSendRequestHeaders");

    CxList sanitizerServer = methodsList.FindByName("HttpContext.Current.Response.Headers.Remove", false)
        .FindByParameterValue(0, "Server", BinaryOperator.IdentityEquality);
-   result.Add(PreSendRequestHeaders - sanitizerServer.GetAncOfType(typeof(MethodDecl)));
+   result.Add(PreSendRequestHeaders - sanitizerServer.GetAncOfType<MethodDecl>());

    //X-Powered-By
    CxList httpProtocolList = (memberList* webConfig).FindByName("system.webServer",false);
@@ -45,5 +45,5 @@
 if(kestrelMethod.Count > 0 ){
    CxList falseList = Find_False_Abstract_Value();
    CxList sanitizedHeader = memberList.FindByShortName("AddServerHeader") * falseList;
```

```
-   result.Add(kestrelMethod- sanitizedHeader.GetAncOfType(typeof(MethodInvokeExpr)));
+   result.Add(kestrelMethod- sanitizedHeader.GetAncOfType<MethodInvokeExpr>());
 }
```

CSharp / CSharp_Low_Visibility / Insufficiently_Protected_Credentials

Code changes

```
---
+++
@@ -1,12 +1,8 @@
-CxList psw  = Find_Passwords();
-
-psw = psw - Find_Methods();
+CxList psw  = Find_Passwords() - Find_Methods();

 CxList DB = All.FindByName("*db*");
 DB.Add(Find_DB_Out());

 CxList sanitize = Find_Decrypt();

-result = psw.InfluencedByAndNotSanitized(DB,sanitize);
-
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result = psw.InfluencedByAndNotSanitized(DB,sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

CSharp / CSharp_Low_Visibility / JavaScript_Hijacking

Code changes

```
---
+++
@@ -1,6 +1 @@
-// DWR is not relevant for C#, therefore it was removed
-
-CxList db  =  Find_DB_Out().DataInfluencedBy(All.FindByName("*select*",  false)  +  All.FindByName("*exec*",  false));
-CxList jason = All.FindByName("*JSON*", false);
-jason = jason.DataInfluencedBy(db);
-result = jason;
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / JWT_Excessive_Expiration_Time

Code changes

```
---
+++
@@ -2,6 +2,7 @@
 CxList decls = Find_Declarators();
 CxList methods = Find_Methods();
 CxList members = Find_MemberAccesses();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 CxList secTokenDesc = objCreates.FindByShortName("SecurityTokenDescriptor");
```

```
 CxList expiresField = decls.GetByAncs(secTokenDesc).FindByShortName("Expires");
@@ -17,7 +18,7 @@
 // Add TimeSpan methods, as those can be used with DateTime
 addMethods.Add(methods.FindByMemberAccess("TimeSpan.From*"));

-CxList addParams = All.GetParameters(addMethods);
+CxList addParams = paramValue.GetParameters(addMethods);

 // We flag durations bigger than 24 hours (86400 seconds)
 IAbstractValue hoursAbs = new IntegerIntervalAbstractValue(24, null);
@@ -52,7 +53,7 @@
 }

 // Add SecurityTokenDescriptor creations with no Expires definition.
-CxList secTokenDescWithExpire = expiresField.GetAncOfType(typeof(ObjectCreateExpr));
+CxList secTokenDescWithExpire = expiresField.GetAncOfType<ObjectCreateExpr>();
 result.Add(secTokenDesc - secTokenDescWithExpire);
 // Add excessive values used in Expires field.
 result.Add(expiresField.DataInfluencedBy(excessiveValues));
```

CSharp / CSharp_Low_Visibility / JWT_Use_Of_Hardcoded_Secret

Code changes

```
---
+++
@@ -1,18 +1,12 @@
 CxList objectCreations = Find_ObjectCreations();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
 CxList objectCreate = objectCreations.FindByShortName("SymmetricSecurityKey");
-
-List<string> objt = new List<string> {"SecurityTokenDescriptor", "TokenValidationParameters"};
-
-CxList isSymmetricSecurityKey = objectCreate
-   .GetByAncs(objectCreations.FindByShortNames(objt));

 CxList inputs = Find_Strings();

-CxList sanitized = All.FindByMemberAccess("Configuration.*");
-sanitized.Add(All.FindByMemberAccess("ConfigurationManager.*"));
-sanitized.Add(All.FindByMemberAccess("AppSettingsSection.*"));
+CxList sanitized = All.FindByMemberAccesses(new []{"Configuration.*", "ConfigurationManager.*", "AppSettingsSection.*"});

-CxList paramsReferences = All.GetParameters(objectCreate, 0);
+CxList paramsReferences = paramValue.GetParameters(objectCreate, 0);

 CxList finalResult = All.NewCxList();
```

CSharp / CSharp_Low_Visibility / Log_Forging

Code changes

```
---
+++
@@ -29,6 +29,11 @@


 CxList Log = Find_Log_Outputs();


+// Removing Hardcoded strings being logged, they cannot be forged
+CxList oneParamLogMethods = Log.FilterByDomProperty<MethodInvokeExpr>(x => x.Parameters.Count == 1);
+CxList relevantStringParams = Find_String_Literal().GetParameters(oneParamLogMethods);
+Log -= Log.FindByParameters(relevantStringParams);
+
 CxList sanitize = Find_Log_Sanitizers();


 result = Log.InfluencedByAndNotSanitized(Inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

CSharp / CSharp_Low_Visibility / Missing_Function_Level_Authorization

Code changes

```
---
+++
@@ -11,34 +11,35 @@
 // Find [Authorize] attributes with Roles or Policy defined
 CxList rolePoliciesParams = customAttributes.FindCustomAttributeParameterByKey("Authorize", "Roles");
 rolePoliciesParams.Add(customAttributes.FindCustomAttributeParameterByKey("Authorize", "Policy"));
-CxList authAttrWithRolePolicy = rolePoliciesParams.GetAncOfType(typeof(CustomAttribute));
+CxList authAttrWithRolePolicy = rolePoliciesParams.GetAncOfType<CustomAttribute>();


-CxList authAttributeInClasses = Find_ClassDecl().CxSelectElements<ClassDecl>(x => x.CustomAttributes) * authorizeAtt;
+CxList authAttributeInClasses = classDecls.CxSelectElements<ClassDecl>(x => x.CustomAttributes) * authorizeAtt;
 CxList classesWithAuth = authAttributeInClasses.GetAncOfType<ClassDecl>();
 CxList classesWithoutAuth = classDecls - classesWithAuth;


-CxList methodsWithAuth = authorizeAtt.GetAncOfType(typeof(MethodDecl));
+CxList methodsWithAuth = authorizeAtt.GetAncOfType<MethodDecl>();
 CxList methodsWithoutAuth = methodDecls - methodsWithAuth;
 CxList methodsWithoutAuthFromClassesWithoutAuth = methodsWithoutAuth.GetByAncs(classesWithoutAuth);


-CxList safeClasses = authAttrWithRolePolicy.GetByAncs(classDecls).GetAncOfType(typeof(ClassDecl));
-CxList safeMethods = authAttrWithRolePolicy.GetByAncs(methodDecls).GetAncOfType(typeof(MethodDecl));
-safeMethods.Add(methodDecls.GetByAncs(safeClasses));
-safeMethods.Add(customAttributes.FindByShortName("AllowAnonymous").GetAncOfType(typeof(MethodDecl)));
-safeMethods.Add(methodDecls.FindAllReferences(Find_LambdaExpr()));
-safeMethods.Add(methodsWithoutAuthFromClassesWithoutAuth);
-
-safeMethods.Add(methodDecls.GetByAncs(interfaceDecls));
-safeMethods.Add(methodDecls.GetByAncs(structDecls));
-safeMethods.Add(methodDecls.GetByAncs(Find_NamespaceDecl().FindByShortName("Page")));
```

```
+CxList safeClasses = authAttrWithRolePolicy.GetByAncs(classDecls).GetAncOfType<ClassDecl>();
+CxList safeMethods = authAttrWithRolePolicy.GetByAncs(methodDecls).GetAncOfType<MethodDecl>();
+safeMethods.Add(
+   methodDecls.GetByAncs(safeClasses),
+   customAttributes.FindByShortName("AllowAnonymous").GetAncOfType<MethodDecl>(),
+   methodDecls.FindAllReferences(Find_LambdaExpr()),
+   methodsWithoutAuthFromClassesWithoutAuth,
+   methodDecls.GetByAncs(interfaceDecls),
+   methodDecls.GetByAncs(structDecls),
+   methodDecls.GetByAncs(Find_NamespaceDecl().FindByShortName("Page")));

 CxList isInRole = methodInvokes.FindByMemberAccess("*.IsInRole");
-safeMethods.Add(isInRole.GetAncOfType(typeof(MethodDecl)));
+safeMethods.Add(isInRole.GetAncOfType<MethodDecl>());

 CxList userValidation = All.FindByName("User.Identity.Name");
-safeMethods.Add(Find_BinaryExpr().DataInfluencedBy(userValidation).GetAncOfType(typeof(MethodDecl)));
+safeMethods.Add(Find_BinaryExpr().DataInfluencedBy(userValidation).GetAncOfType<MethodDecl>());

-CxList methodsInfluencedByUserValidation = unkRefs.FindAllReferences(Find_ParamDecl()).DataInfluencedBy(userValidation).GetAncOfType(typeof(MethodDecl));
+CxList methodsInfluencedByUserValidation = unkRefs.FindAllReferences(
+   Find_ParamDecl()).DataInfluencedBy(userValidation).GetAncOfType<MethodDecl>();
 safeMethods.Add(methodsInfluencedByUserValidation);

 result = methodDecls - safeMethods;
```

CSharp / CSharp_Low_Visibility / Off_By_One_Error

Code changes

```
---
+++
@@ -1,14 +1,13 @@
 //find all indexer access using counf or length
-CxList members = Find_MemberAccesses();
-CxList counts = members.FindByShortNames(new List <string> {"Count",  "Length"});
+CxList counts = Find_MemberAccesses().FindByShortNames(new []{"Count",  "Length"});

-CxList listsUsingCount = counts.GetFathers().FindByType(typeof(IndexerRef));
+CxList listsUsingCount = counts.GetFathers().FindByType<IndexerRef>();
 counts = counts.GetByAncs(listsUsingCount);

 foreach (CxList count in counts)
 {
    CxList listUsingCount = count.GetFathers();
-   listUsingCount = listUsingCount.FindByType(typeof(IndexerRef));
+   listUsingCount = listUsingCount.FindByType<IndexerRef>();
    CxList usedCount = count.GetTargetOfMembers();
    result.Add(listUsingCount.FindByShortName(usedCount));
 }
```

CSharp / CSharp_Low_Visibility / Open_Redirect

Code changes

```
---
+++
@@ -6,72 +6,33 @@
 CxList memberAccesses = Find_MemberAccesses();


 // Outputs
-CxList redirect = methods.FindByMemberAccess("HttpResponse.Redirect");
-redirect.Add(methods.FindByName("*Response.Redirect"));
-redirect.Add(methods.FindByShortNames(new List<string>(){"Redirect", "RedirectPermanent"}));
-redirect.Add(constructors.FindByShortName("RedirectResult"));
+CxList redirect = All.NewCxList(
+   methods.FindByMemberAccess("HttpResponse.Redirect"),
+   methods.FindByName("*Response.Redirect"),
+   methods.FindByShortNames(new []{"Redirect", "RedirectPermanent"}),
+   constructors.FindByShortName("RedirectResult"));

-CxList addHeader = methods.FindByName("*Response.AddHeader");
-addHeader.Add(methods.FindByName("*Response.Headers.Add"));
-CxList redirectHeader = addHeader.FindByParameters(strings.FindByShortName("Location"));
+CxList addHeader = methods.FindByNames(new []{"*Response.AddHeader", "*Response.Headers.Add"});
+CxList redirectHeader = All.NewCxList(
+   addHeader.FindByParameters(strings.FindByShortName("Location")));
+
 redirect.Add(All.GetParameters(redirectHeader, 1));


 // Inputs
-CxList inputs = memberAccesses.FindByName("*Request.QueryString_*");
-inputs.Add(methods.FindByName("*Request.QueryString"));
-inputs.Add(unknownRefs.FindByShortName("Request").FindByFathers(indexerRefs));
-inputs.Add(memberAccesses.FindByName("*Request.QueryString").FindByFathers(indexerRefs));
-inputs.Add(indexerRefs.FindByShortName("QueryString"));
+CxList inputs = All.NewCxList(
+   memberAccesses.FindByName("*Request.QueryString_*"),
+   methods.FindByName("*Request.QueryString"),
+   unknownRefs.FindByShortName("Request").FindByFathers(indexerRefs),
+   memberAccesses.FindByName("*Request.QueryString").FindByFathers(indexerRefs),
+   indexerRefs.FindByShortName("QueryString"));


 // Sanitizers
-CxList sanitize = Find_Sanitize();
-CxList urlHelper = unknownRefs.FindByTypes(new string[]{"UrlHelper", "Url"});
-sanitize.Add(urlHelper.GetMembersOfTarget().FindByShortNames(new List<string>(){"Action", "HttpRouteUrl", "RouteUrl"}));
-CxList islocalUrlSanitizer = methods.FindByShortNames(new List<string>(){"IsLocalUrl", "IsUrlLocalToHost"});
-CxList references = Find_Unknown_References();
-CxList sanitizedParams = references.GetParameters(islocalUrlSanitizer);
-// We're assuming that if users test an input string with "Url.IsLocalUrl(input)" every reference of that input is sanitized.
```

```
-CxList paramReferences = references.FindAllReferences(sanitizedParams);
-foreach(CxList isLocalUrl in islocalUrlSanitizer){
-   isLocalUrl.Add(references.FindAllReferences(isLocalUrl.GetAssignee()));
-   CxList ifParent = isLocalUrl.GetFathers().FindByType(typeof(IfStmt));
-   if(ifParent.Count > 0){
-       CxList trueBlock = ifParent.GetBlocksOfIfStatements(true);
-       sanitize.Add(paramReferences.GetByAncs(trueBlock));
-       continue;
-   }
-   CxList notParentThenIf = isLocalUrl.GetFathers().FindByType(typeof(UnaryExpr))
-       .FindByShortName("Not").GetFathers().FindByType(typeof(IfStmt));
-   if(notParentThenIf.Count > 0){
-       CxList falseBlock = notParentThenIf.GetBlocksOfIfStatements(false);
-       sanitize.Add(paramReferences.GetByAncs(falseBlock));
-       continue;
-   }
-   CxList ternaryExpr = isLocalUrl.GetFathers().FindByType(typeof(TernaryExpr));
-   if(ternaryExpr.Count > 0){
-       TernaryExpr ternaryExprDom = ternaryExpr.TryGetCSharpGraph<TernaryExpr>();
-
-       CxList trueExpression = All.NewCxList();
-       trueExpression.Add(ternaryExprDom.True.NodeId, ternaryExprDom.True);
-       sanitize.Add(paramReferences.GetByAncs(trueExpression));
-       continue;
-   }
-   CxList ternaryExprFalse = isLocalUrl.GetFathers().FindByType(typeof(UnaryExpr))
-       .FindByShortName("Not").GetFathers().FindByType(typeof(TernaryExpr));
-   if(ternaryExprFalse.Count > 0){
-       TernaryExpr ternaryExprFalseDom = ternaryExprFalse.TryGetCSharpGraph<TernaryExpr>();
-
-       CxList falseExpression = All.NewCxList();
-       falseExpression.Add(ternaryExprFalseDom.False.NodeId, ternaryExprFalseDom.False);
-       sanitize.Add(paramReferences.GetByAncs(falseExpression));
-       continue;
-   }
-}
+CxList sanitize = Find_Open_Redirect_Sanitizers();

 result = redirect.InfluencedByAndNotSanitized(inputs, sanitize);

 CxList aspNetControllers = Find_Classes().InheritsFrom("Controller");
-CxList controllerRedirects = methods.FindByShortNames(new List<string>{"Redirect", "RedirectToAction", "RedirectToRoute"})
+CxList controllerRedirects = methods.FindByShortNames(new []{"Redirect", "RedirectToAction", "RedirectToRoute"})
     .GetByAncs(aspNetControllers);

 result.Add(controllerRedirects.InfluencedByAndNotSanitized(Find_Interactive_Inputs(), sanitize));
```

CSharp / CSharp_Low_Visibility / Overly_Permissive_Cross_Origin_Resource_Sharing_Policy

Code changes

```
---
+++
@@ -11,10 +11,10 @@
    "IServiceCollection", "IApplicationBuilder" });
 CxList corsList = unknownService
    .GetMembersOfTarget()
-   .FindByShortNames(new List<string>{ "AddCors", "UseCors" });
+   .FindByShortNames(new []{ "AddCors", "UseCors" });
 CxList lambdaCors = lambdaExpr.GetParameters(corsList);
 CxList methodsLamdba = methods.GetByAncs(lambdaCors);
-CxList methodsCors = methodsLamdba.GetAncOfType(typeof(MethodInvokeExpr));
+CxList methodsCors = methodsLamdba.GetAncOfType<MethodInvokeExpr>();
 CxList methodsCorsDef = methodDecl.FindDefinition(methodsCors);
 methodsCors.Add(methods.GetByAncs(methodsCorsDef));
 CxList allowAnyOrigin = methodsCors.FindByShortName("AllowAnyOrigin");
@@ -23,7 +23,7 @@
 IAbstractValue asteriskAbsValue = new StringAbstractValue("*");
 CxList asteriskStrings = strings.FindByAbstractValue(abstractValue =>
    asteriskAbsValue.IncludedIn(abstractValue));
-CxList asteriskStringsDecl = asteriskStrings.GetAncOfType(typeof(Declarator));
+CxList asteriskStringsDecl = asteriskStrings.GetAncOfType<Declarator>();
 asteriskStrings.Add(unknown.FindAllReferences(asteriskStringsDecl));

 CxList withOrigins = methodsCors.FindByShortName("WithOrigins");
```

CSharp / CSharp_Low_Visibility / Potential_ReDoS

Code changes

```
---
+++
@@ -1,7 +1 @@
-CxList evil = Find_Evil_Strings();
-
-CxList filter = Potential_ReDoS_In_Code();
-filter.Add(Potential_ReDoS_In_Static_Field());
-filter.Add(All.FindByType("ValidationExpression"));
-
-result = evil - filter - evil.DataInfluencingOn(filter);
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Potential_ReDoS_By_Injection

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_ReDoS(Find_Inputs(), true);
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Potential_ReDoS_In_Code

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_ReDoS(Find_Evil_Strings(), true);
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Potential_ReDoS_In_Static_Field

Code changes

```
---
+++
@@ -1,20 +1 @@
-CxList evilStrings = Find_Evil_Strings();
-CxList regex = Find_Regex();
-
-// Static regex declarations (these do not influence their references, so needed in addition)
-CxList types = regex.GetAncOfType<FieldDecl>();
-types.Add(regex.GetAncOfType<ConstantDecl>());
-
-CxList stat = types.FindByFieldAttributes(Modifiers.Static);
-
-// Regex commands that descends from a static regex declaration
-CxList inputsRegex = Find_Inputs();
-inputsRegex.Add(regex);
-
-CxList commands = inputsRegex.GetByAncs(stat);
-
-// Sanitization
-CxList sanitize = Find_Sanitize();
-
-result = evilStrings.InfluencingOnAndNotSanitized(commands, sanitize)
-   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

CSharp / CSharp_Low_Visibility / Session_Clearing_Problems

Code changes

```
---
+++
@@ -1,21 +1,21 @@
-CxList session = All.FindByShortNames(new List<string> {"*Session_User*",
+CxList session = All.FindByShortNames(new []{"*Session_User*",
         "*Session_Cust*",
         "*Session_Id*"}, false);


 //The code below takes just one line from 3 possible for each Session Open
-session = session.FindByType(typeof(IndexerRef));
+session = session.FindByType<IndexerRef>();
```

```
 if(session.Count > 0)
 {
    CxList emptyString = Find_Empty_Strings();
    CxList zero = All.FindByName("0");
-   CxList clear = All.FindByName("*Session.Clear") +
-       All.FindByName("*Session.Abandon") +
-       All.FindByMemberAccess("*FormsAuthentication.SignOut");
+   CxList clear = All.NewCxList(
+       All.FindByNames(new []{"*Session.Clear", "*Session.Abandon"}),
+       All.FindByMemberAccess("*FormsAuthentication.SignOut"));
    CxList a = All.FindByAssignmentSide(CxList.AssignmentSide.Left) * session;
-   CxList b = All.FindByAssignmentSide(CxList.AssignmentSide.Right) * (emptyString + zero);
+   CxList b = All.FindByAssignmentSide(CxList.AssignmentSide.Right) * (All.NewCxList(emptyString, zero));

    CxList c = a.GetFathers() * b.GetFathers();
-   if((c + clear).Count == 0 && session.data.Count > 0)
+   if(All.NewCxList(c, clear).Count == 0 && session.data.Count > 0)
    {
        // From all places where session is opened, take just first one to present
        // Build first as DOM element
```

CSharp / CSharp_Low_Visibility / Stored_Code_Injection

Code changes

```
---
+++
@@ -1,14 +1,12 @@
 CxList memberAccesses = Find_MemberAccesses();

 //find all code provider
-CxList CodeProvider = memberAccesses.FindByMemberAccess("CSharpCodeProvider.*");
-CodeProvider.Add(memberAccesses.FindByMemberAccess("VBCodeProvider.*"));
-CodeProvider.Add(All.FindByMemberAccess("MethodInfo.*"));
-CodeProvider.Add(memberAccesses.FindByMemberAccess("JScriptCodeProvider.*"));
-CodeProvider.Add(All.FindByMemberAccess("CodeDomProvider.*"));
+CxList CodeProvider = memberAccesses.FindByMemberAccesses(new [] {"CSharpCodeProvider.*",
+       "VBCodeProvider.*", "JScriptCodeProvider.*"});
+CodeProvider.Add(All.FindByMemberAccesses(new []{"MethodInfo.*", "CodeDomProvider.*"}));

 //find only codeCompilers (clean irrelevant methods)
-List < string > methCodeCompilers = new List<string> {"CompileAssemblyFrom*", "Parse", "Invoke"};
+string[] methCodeCompilers = new [] {"CompileAssemblyFrom*", "Parse", "Invoke"};
 CxList codeCompilers = CodeProvider.FindByShortNames(methCodeCompilers, false);

 CxList inputs = Find_Read();
```

CSharp / CSharp_Low_Visibility / Stored_Command_Argument_Injection

Code changes

```
---
+++
@@ -1,11 +1,12 @@
 CxList memberlst =  Find_MemberAccesses();
 CxList dbOutputs = Find_DB_Out();

-CxList exec = All.NewCxList();
-exec.Add(Find_String_Command_Execution(true), Find_String_Command_Execution(false));
+CxList exec = All.NewCxList(Find_String_Command_Execution(true), Find_String_Command_Execution(false));

-CxList inputs = All.NewCxList();
-inputs.Add(Find_Read(), dbOutputs ,memberlst.FindByMemberAccess("Process.GetProcesses"));
+CxList inputs = All.NewCxList(
+   Find_Read(),
+   dbOutputs,
+   memberlst.FindByMemberAccess("Process.GetProcesses"));

 CxList inputsDb = Find_UnknownReference().FindAllReferences(dbOutputs.GetAssignee());
 inputs.Add(inputsDb.GetMembersOfTarget().FindByShortName("Get*"));
@@ -15,7 +16,7 @@
 CxList processWriteLines = memberlst.FindByMemberAccess("Process.StandardInput")
     .GetMembersOfTarget()
     .FindByShortName("Write*")
-   .FindByType(typeof(MethodInvokeExpr));
+   .FindByType<MethodInvokeExpr>();

 exec.Add(processWriteLines);
```

CSharp / CSharp_Low_Visibility / Thread_Safety_Issue

Code changes

```
---
+++
@@ -1,32 +1,32 @@
 if (All.isWebApplication || Check_Web_Application().Any())
 {
-   CxList logs = All.FindByName("*Log*", false) +
-       All.FindByType("Logger");
+   CxList logs = All.NewCxList(All.FindByName("*Log*", false), All.FindByType("Logger"));

    CxList no_logs = All - logs;

    CxList statics = no_logs.FindAllReferences(no_logs.FindByFieldAttributes(Modifiers.Static) -
        no_logs.FindByFieldAttributes(Modifiers.Readonly));

-   statics = statics - no_logs.FindByType(typeof(MethodInvokeExpr));
+   statics = statics - no_logs.FindByType<MethodInvokeExpr>();
```

```
    CxList EventArgs = All.FindByType("*CommandEventArgs");

    CxList request = Find_Request();
-   CxList request_classes = request.GetAncOfType(typeof(ClassDecl)).InheritsFrom("Page");
+   CxList request_classes = request.GetAncOfType<ClassDecl>().InheritsFrom("Page");

     CxList inputs = Find_Interactive_Inputs() - request.GetByAncs(request_classes);

    CxList methods = Find_Methods();

    // Ignore flows through the predicate of IEnumerable and ICollection filter methods:
-   CxList filterMethodsFirst = methods.FindByShortNames(new List<string> {
+   CxList filterMethodsFirst = methods.FindByShortNames(new []{
            "All", "Any", "Count", "LongCount", "First", "FirstOrDefault","Last", "LastOrDefault",
            "Single", "SingleOrDefault", "TakeWhile","SkipWhile", "Where",
            "Find", "FindAll", "FindLastIndex", "FindIndex", "RemoveAll", "TrueForAll"});
-   CxList filterMethodsSecond = methods.FindByShortNames(new List<string> { "Contains" });
+   CxList filterMethodsSecond = methods.FindByShortNames(new []{ "Contains" });

-   CxList filterMethodDefinitions = All.FindDefinition(filterMethodsFirst + filterMethodsSecond);
+   CxList filterMethodsFirstAndSec = All.NewCxList(filterMethodsFirst, filterMethodsSecond);
+   CxList filterMethodDefinitions = All.FindDefinition(filterMethodsFirstAndSec);
    CxList implementedFilterMethods = methods.FindAllReferences(filterMethodDefinitions);
    filterMethodsFirst -= implementedFilterMethods;
    filterMethodsSecond -= implementedFilterMethods;
@@ -37,5 +37,6 @@
    parameters.Add(lambda.GetParameters(filterMethodsSecond, 1));
    CxList sanitizers = All.FindByFathers(returns.GetByAncs(parameters));

-   result = (EventArgs + inputs).InfluencingOnAndNotSanitized(statics, sanitizers);
+   CxList source = All.NewCxList(EventArgs, inputs);
+   result = source.InfluencingOnAndNotSanitized(statics, sanitizers);
 }
```

CSharp / CSharp_Low_Visibility / Trust_Boundary_Violation_in_Session_Variables

Code changes

```
---
+++
@@ -22,10 +22,10 @@
 CxList sessions = httpTypes.GetFathers().FindByShortName("session*", false);
 sessions.Add(unknownRef.FindByShortName("Session"));

-List<string> sessionMembersThatCanStoreData = new List<string> { "LCID", "Item",
+string[] sessionMembersThatCanStoreData = new [] { "LCID", "Item",
        "CodePage", "Timeout", "Add" };

-List<string> sessionOptions = new List<string> { "SetInt32", "SetString" };
+string[] sessionOptions = new [] { "SetInt32", "SetString" };
```

```
 CxList parameters = All.GetParameters(sessions.GetMembersOfTarget().FindByShortNames(sessionOptions), 1) - AllParameters;
 CxList unknowRefsInsideParameters = unknownRef.GetByAncs(parameters);
@@ -33,7 +33,7 @@
 CxList sinks = sessions.GetMembersOfTarget().FindByShortNames(sessionMembersThatCanStoreData);
 sinks.Add(unknowRefsInsideParameters);

-sinks.Add(memberAccesses.FindByShortNames(new List<string> {"Session_*", "Session"})
+sinks.Add(memberAccesses.FindByShortNames(new [] {"Session_*", "Session"})
     .FindByFathers(indexerRefs.FindByAssignmentSide(CxList.AssignmentSide.Left)));

 result = sinks.InfluencedByAndNotSanitized(inputs, sanitizers);
```

CSharp / CSharp_Low_Visibility / Unencrypted_Web_Config_File

Code changes

```
---
+++
@@ -6,7 +6,7 @@

 // encrypted values and keys
 CxList xmlTokens = webConfigs.FindByAssignmentSide(CxList.AssignmentSide.Left).GetTargetOfMembers();
-xmlTokens -= xmlTokens.FindByShortNames(new List<string>{
+xmlTokens -= xmlTokens.FindByShortNames(new []{
         "keyinfo","keyname","encryptionmethod", "encryptedkey", "encrypteddata","cipherdata","ciphervalue"}, false);
 string[] targetConfigSections = new string[]{
```

CSharp / CSharp_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,20 +1,20 @@
 CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null");
+CxList nullStrings = All.FindByName("null");
 CxList psw = Find_Passwords();
 CxList allParams = Find_Param();
 CxList allMethods = Find_Methods();
 CxList fields = Find_FieldDecls();
+CxList paramValue = allParams.CxSelectDomProperty<Param>(p => p.Value);
+CxList equalsPassword = All.NewCxList();

 // Find password in an initialization operation
 CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
 CxList psw_in_lSide_decl = psw_in_lSide.FindByType<Declarator>();

-CxList allStrLiterals = Find_Strings();
-
```

```
-CxList strToRemove = All.NewCxList();
-strToRemove.Add(emptyString, NULL);
-
-CxList strLiterals = allStrLiterals - strToRemove;
+CxList strLiterals = Find_Strings();
+CxList toRemove = All.NewCxList(emptyString, nullStrings, strLiterals.FindByShortNames("* *", "*.*"));
+strLiterals -= toRemove;
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);

 CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);

@@ -35,24 +35,21 @@
 }
 initializedPassword -= notHdPass;

-
 CxList paswInParams = psw.GetFathers() * allParams;
 CxList methodsWithPswInParam = allMethods.FindByParameters(paswInParams);

-CxList equalsPassword = All.NewCxList();
-
 //Find in string.Equals("mypass", psw)
-CxList equalsMemberAccess = All.FindByMemberAccesses(new []{ "String.Equals", "string.Equals", "string.Compare*", "String.Compare*"});
+CxList equalsMemberAccess = allMethods.FindByMemberAccesses(new []
+   {"String.Equals", "string.Equals", "string.Compare*", "String.Compare*"});
 CxList pswParamsInCompareMethods = methodsWithPswInParam * equalsMemberAccess;
-equalsPassword.Add(strLiterals * All.GetParameters(pswParamsInCompareMethods));
-
+equalsPassword.Add(strLiterals.GetParameters(pswParamsInCompareMethods));

 //Find in psw == "mypass"
 CxList pswUsedInEqualExpr = psw.GetFathers() * Find_BinaryExpr().FindByShortName("==");
 equalsPassword.Add(strLiterals.GetByAncs(pswUsedInEqualExpr));

 //Find in "mypass".Equals(psw)
-CxList compareMethods = allMethods.FindByShortNames(new List<string>{ "Equals", "CompareTo"});
+CxList compareMethods = allMethods.FindByShortNames("Equals", "CompareTo");
 CxList targetWhenPswInParams = strLiterals * (methodsWithPswInParam * compareMethods).GetTargetOfMembers();
 equalsPassword.Add(strLiterals * targetWhenPswInParams);

@@ -67,14 +64,15 @@
 assignPassword = lit_in_rSide.GetAssigner().GetByAncs(assignPassword);
 // Remove fields from this object, as it will be treated later
 assignPassword -= assignPassword.GetByAncs(networkCredentialObjs);
-result.Add(initializedPassword, equalsPassword, assignPassword.GetAssignee());
-

 // Find hardcoded parameter of new NetworkCredential(user, pass)
```

```
-CxList netCredsPassword = All.GetParameters(networkCredentialObjs, 1);
+CxList netCredsPassword = paramValue.GetParameters(networkCredentialObjs, 1);
 // when object is created like new NetworkCredential { Password = ""}
 netCredsPassword.Add(fields.FindByShortName("Password")
     .GetByAncs(networkCredentialObjs).GetAssigner());
 CxList credPwds = netCredsPassword.FindByAbstractValue(abs => abs is StringAbstractValue);

-result.Add(credPwds);
+result.Add(initializedPassword,
+    equalsPassword,
+    assignPassword.GetAssignee(),
+    credPwds);
```

CSharp / CSharp_Low_Visibility / Use_of_Insufficiently_Random_Values

Code changes

```
---
+++
@@ -8,7 +8,7 @@
     "Random.NextDouble"});

 /* Find calls to {Next,NextByte,NextDouble} inside a class definition that extends Random */
-CxList relevantMethods = methodsList.FindByShortNames(new List<string> {"Next", "NextByte", "NextDouble"});
+CxList relevantMethods = methodsList.FindByShortNames(new []{"Next", "NextByte", "NextDouble"});
 relevantMethods -= memberAccessList;
 CxList definitionRelevantMethods = methods.FindDefinition(relevantMethods);
```

CSharp / CSharp_Low_Visibility / Use_of_RSA_Algorithm_without_OAEP

Code changes

```
---
+++
@@ -1,6 +1,8 @@
-CxList rsa = All.FindByMemberAccess("RSACryptoServiceProvider.Encrypt");
-CxList rsaParam = All.GetParameters(rsa, 1);
+CxList rsa = Find_Methods().FindByMemberAccess("RSACryptoServiceProvider.Encrypt");
+CxList rsaParam = Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(rsa, 1);
 CxList inputs = Find_Inputs();
-CxList negative = inputs + All.FindByType(typeof(BooleanLiteral)).FindByShortName("false");
+CxList negative = All.NewCxList(inputs, Find_BooleanLiteral().FindByShortName("false"));

-result = rsaParam * negative + rsaParam.DataInfluencedBy(negative);
+result = All.NewCxList(
+    rsaParam * negative,
+    rsaParam.DataInfluencedBy(negative));
```

CSharp / CSharp_Low_Visibility / XSS_Evasion_Attack

Code changes

```
---
+++
@@ -1,8 +1 @@
-if(All.isWebApplication || Check_Web_Application().Any())
-{
-   CxList decode = All.FindByName("*Server.HtmlDecode");
-   CxList sanitize = Find_XSS_Sanitize();
-   CxList output = Find_XSS_Outputs();
-
-   result = output.InfluencedByAndNotSanitized(decode, sanitize);
-}
+//This query is deprecated.
```

CSharp / CSharp_Medium_Threat / Buffer_Overflow

Code changes

```
---
+++
@@ -1,21 +1,19 @@
 CxList Inputs = Find_Interactive_Inputs();
 CxList declarators = Find_Declarators();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 //Find all external Methods
 CxList DllImport = All.FindByName("*DllImport*").FindByType<CustomAttribute>();
 CxList ExternalMethod = DllImport.GetFathers().GetFathers().FindByType<MethodDecl>();
 ExternalMethod = All.FindAllReferences(ExternalMethod);
-CxList ExternalMethodParams = All.GetParameters(ExternalMethod);
+CxList ExternalMethodParams = paramValue.GetParameters(ExternalMethod);
 CxList Unsafe = All.GetByMethod(All.FindByFieldAttributes(Modifiers.Unsafe));

 //All pointers in a method with Unsafe modifier
 CxList pointers = declarators.FindByRegex(@"[^(\s]\s*\*\s*\w", CxList.CxRegexOptions.None, RegexOptions.None, null);
-pointers = All.GetByAncs(pointers).FindByType<Declarator>();
+pointers = declarators.FindDescendantsOfType<Declarator>(pointers);
 pointers = Unsafe.FindAllReferences(pointers);

-CxList pointersAndMethods = All.NewCxList();
-pointersAndMethods.Add(pointers);
-pointersAndMethods.Add(ExternalMethodParams);
+CxList pointersAndMethods = All.NewCxList(pointers, ExternalMethodParams);

-result = pointersAndMethods.InfluencedBy(Inputs);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result = pointersAndMethods.InfluencedBy(Inputs).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

CSharp / CSharp_Medium_Threat / CGI_XSS

Code changes

```
---
+++
@@ -1,10 +1,8 @@
-CxList unkRefsDecls = All.NewCxList();
-unkRefsDecls.Add(Find_Unknown_References(), Find_Declarators());
+CxList unkRefsDecls = All.NewCxList(Find_Unknown_References(), Find_Declarators());
 CxList strings = Find_Strings();
 CxList methods = Find_Methods();
 CxList getPropertyMethods = methods.FindByMemberAccess("Environment.GetEnvironmentVariable");
-CxList queryString = strings.GetParameters(getPropertyMethods)
-   .FindByShortNames(new List <string> {"cgi.*","QUERY_STRING"});
+CxList queryString = strings.GetParameters(getPropertyMethods).FindByShortNames(new []{"cgi.*","QUERY_STRING"});

 CxList contentType = strings.FindByShortName("Content-type*", false);
 CxList parameterOfOutputMethods = Find_Param().GetParameters(Find_Interactive_Outputs());
@@ -14,8 +12,7 @@
     .FindByParameters(strings.FindByShortName("system.webServer/cgi"))
     .InfluencedBy(unkRefsDecls.FindByType("ServerManager"));

-CxList cgiVerification = All.NewCxList();
-cgiVerification.Add(queryString, contentType, webServerCgi);
+CxList cgiVerification = All.NewCxList(queryString, contentType, webServerCgi);

 if(!All.isWebApplication && cgiVerification.Count > 0){
```

```
    //Get Inputs
@@ -48,10 +45,11 @@
    //Add the results that are both input and output but are not sanitized
    foreach (CxList output in outputs.GetCxListByPath())
    {   //Get the start node that is also an input
-       CxList pathStart = output.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly) * inputs;
-       CxList pathEnd = output.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly); //Get end node
+       CxList pathStart = output.GetFirstNodesInPath() * inputs;
+       CxList pathEnd = output.GetLastNodesInPath(); //Get end node
        //If the start node and end node are not a sanitizer
-       if ((pathStart + pathEnd - sanitized).Count == 2)
+       CxList path = All.NewCxList(pathStart, pathEnd);
+       if ((path - sanitized).Count == 2)
        {   //If the path is not sanitized, then add to result
            result.Add(output);
        }
```

CSharp / CSharp_Medium_Threat / Cookie_Injection

Code changes

```
---
+++
@@ -15,7 +15,7 @@
    );
```

```
 CxList http_cookies = All.FindByType("HttpCookie");
-CxList add_methods = All.FindByMemberAccess("Cookies.Add");
+CxList add_methods = Find_Methods().FindByMemberAccess("Cookies.Add");

 CxList cookies_in_response = All.FindByName("Response.Cookies*");
 cookies_in_response.Add(http_cookies.InfluencingOn(add_methods));
```

CSharp / CSharp_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_DB_Parameter_Tampering();
+//This query is deprecated.
```

CSharp / CSharp_Medium_Threat / Hardcoded_password_in_Connection_String

Code changes

```
---
+++
@@ -1,17 +1,18 @@
 CxList methodsList = Find_Methods();
 CxList unknownList = Find_Unknown_References();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
 // Find the string literals containig "password"
 CxList psw = Find_Password_In_Connection_String();

 // Find all the connections
 CxList createExpressions = base.Find_ObjectCreations();

-CxList openConnection = createExpressions.FindByShortNames(new List<string> {
-       "*Connection", "*DataContext"});
+CxList openConnection = createExpressions.FindByShortNames(new []{"*Connection", "*DataContext"});

 openConnection.Add(All.FindByMemberAccess("Connection.ConnectionString"));

-CxList sanitizers = Find_Same_Value_Sanitizers_For_Hardcoded_Password(openConnection + psw);
+CxList openConnAndPsw = All.NewCxList(openConnection, psw);
+CxList sanitizers = Find_Same_Value_Sanitizers_For_Hardcoded_Password(openConnAndPsw);

 // Return the connections influenced by "passworded" strings.
 // Notice that since it's an ObjectCreateExpr ("new()"), only its parameters
@@ -20,16 +21,18 @@

 // Add connection strings that contain a password in their initialization
 CxList getConnection = methodsList.FindByName("DriverManager.getConnection");
-CxList connectionStrings = getConnection.DataInfluencedBy(All.GetParameters(getConnection, 2).FindByAbstractValue(x => x is StringAbstractValue));
+CxList connectionStrings = getConnection.DataInfluencedBy(
+   paramValue.GetParameters(getConnection, 2).FindByAbstractValue(x => x is StringAbstractValue));
```

```
 //Add Entity Framework
 CxList dbContextList = unknownList.FindByType("IServiceCollection").GetMembersOfTarget().FindByShortName("AddDbContext");
 CxList possibleMethods = methodsList.GetByAncs(dbContextList);
 CxList optionBuilderList = unknownList.FindByType("DbContextOptionsBuilder");
 possibleMethods.Add(optionBuilderList.GetMembersOfTarget());
-possibleMethods = possibleMethods.FindByShortNames(new List<string> {"UseSqlServer", "UseSqlite"});
+possibleMethods = possibleMethods.FindByShortNames(new []{"UseSqlServer", "UseSqlite"});

-sanitizers = Find_Same_Value_Sanitizers_For_Hardcoded_Password(possibleMethods + psw);
+CxList possibleMethodsAndPsw = All.NewCxList(possibleMethods, psw);
+sanitizers = Find_Same_Value_Sanitizers_For_Hardcoded_Password(possibleMethodsAndPsw);

 result.Add(possibleMethods.InfluencedByAndNotSanitized(psw, sanitizers));
```

CSharp / CSharp_Medium_Threat / HttpOnlyCookies

Code changes

```
---
+++
@@ -20,10 +20,10 @@
    // result
    result = aspNetCoreCookiePolicyOptions;

-   CxList toRemove = All.NewCxList();
-   toRemove.Add(
+   CxList toRemove = All.NewCxList(
        Find_ClassDecl(),
-       assignedOptionField.GetAncOfType(typeof(MethodInvokeExpr)));
+       assignedOptionField.GetAncOfType<MethodInvokeExpr>());
+
    result -= toRemove;
 }

@@ -40,23 +40,19 @@
    CxList booleanLiterals = Find_BooleanLiteral();
    CxList objectCreations = Find_ObjectCreations();

-   CxList possibleCookiesOptions = All.NewCxList();
-   possibleCookiesOptions.Add(
+   CxList possibleCookiesOptions = All.NewCxList(
        methods,
        unknownRef,
        objectCreations);

-   CxList httpCookiesOrigins = All.NewCxList();
-   httpCookiesOrigins.Add(
+   CxList httpCookiesOrigins = All.NewCxList(
```

```
        objectCreations.FindByType("HttpCookie"),
        methods.FindByMemberAccess("*Cookies.Get"),
-       memberAccesses.FindByShortName("*Cookies*").GetAncOfType(typeof(IndexerRef)));
+       memberAccesses.FindByShortName("*Cookies*").GetAncOfType<IndexerRef>());
    httpCookiesOrigins -= memberAccesses.FindByShortName("Value").GetTargetOfMembers();
-
-   CxList cookiesFathers = All.NewCxList();
-   cookiesFathers.Add(parameters, returns);
-   CxList httpCookies = All.NewCxList();
-   httpCookies.Add(
+
+   CxList cookiesFathers = All.NewCxList(parameters, returns);
+   CxList httpCookies = All.NewCxList(
        httpCookiesOrigins.GetAssignee(),
        httpCookiesOrigins.FindByFathers(cookiesFathers));

@@ -64,36 +60,35 @@
    CxList cookieSecurePolicyAlways = memberAccesses.FindByMemberAccess("CookieSecurePolicy.Always");

    CxList httpOnlyFlag = trueLiterals.GetAssignee().FindByShortName("HttpOnly");
-   httpOnlyFlag.Add(httpOnlyFlag.GetByAncs(fieldDecls).GetAncOfType(typeof(ObjectCreateExpr)));
+   httpOnlyFlag.Add(httpOnlyFlag.GetByAncs(fieldDecls).GetAncOfType<ObjectCreateExpr>());
    httpOnlyFlag.Add(httpOnlyFlag.GetAssignee());

-   CxList secureFlag = All.NewCxList();
-   secureFlag.Add(
+   CxList secureFlag = All.NewCxList(
        trueLiterals.GetAssignee().FindByShortName("Secure"),
        cookieSecurePolicyAlways.GetAssignee().FindByShortName("SecurePolicy"));
-   secureFlag.Add(secureFlag.GetByAncs(fieldDecls).GetAncOfType(typeof(ObjectCreateExpr)));
+   secureFlag.Add(secureFlag.GetByAncs(fieldDecls).GetAncOfType<ObjectCreateExpr>());
    secureFlag.Add(secureFlag.GetAssignee());

    CxList cookiesAppend = methods.FindByMemberAccess("*Cookies.Append");

-   CxList cookies = All.NewCxList();
-   cookies.Add(
+   CxList cookies = All.NewCxList(
        httpCookies,
        possibleCookiesOptions.GetParameters(cookiesAppend, 2));

-   CxList safeCookies = cookies.DataInfluencedBy(httpOnlyFlag)
-       .GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-   safeCookies = safeCookies.DataInfluencedBy(secureFlag)
-       .GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-   safeCookies.Add(httpCookies.FindDefinition(httpOnlyFlag.GetLeftmostTarget())
-       .FindDefinition(secureFlag.GetLeftmostTarget()));
+   CxList safeCookies = cookies.DataInfluencedBy(httpOnlyFlag).GetLastNodesInPath();
```

```
+   safeCookies = safeCookies.DataInfluencedBy(secureFlag).GetLastNodesInPath();
+   safeCookies.Add(httpCookies.FindDefinition(httpOnlyFlag.GetLeftmostTarget()));
    safeCookies.Add(cookies * httpOnlyFlag * secureFlag);

-   CxList unsafeHttpCookies = httpCookies.Clone();
+   // HttpContext.Current just retrieves the values of existing cookies
+   CxList httpCurrent = memberAccesses.FindByMemberAccess("HttpContext.Current").GetAncOfType<Declarator>();
+   safeCookies.Add(cookies * httpCurrent);
+
+   CxList unsafeHttpCookies = All.NewCxList(httpCookies);
    unsafeHttpCookies -= safeCookies;

-   CxList safeCookiesAppend = safeCookies.GetAncOfType(typeof(Param)).GetAncOfType(typeof(MethodInvokeExpr));
-   CxList unsafeCookiesAppend = cookiesAppend.Clone();
+   CxList safeCookiesAppend = safeCookies.GetAncOfType<Param>().GetAncOfType<MethodInvokeExpr>();
+   CxList unsafeCookiesAppend = All.NewCxList(cookiesAppend);
    unsafeCookiesAppend -= safeCookiesAppend;

    // result
```

CSharp / CSharp_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query HTTP_Response_Splitting is deprecated");
```

CSharp / CSharp_Medium_Threat / Improper_Locking

Code changes

```
---
+++
@@ -2,11 +2,9 @@
 CxList usingStmtDeclarators = Find_Using_Declarators();
 CxList mutexList = typeMutex.GetMembersOfTarget();

-CxList mutexClose = mutexList.FindByName("*.Close");
-mutexClose.Add(mutexList.FindByName("*.Dispose"));
-mutexClose.Add(mutexList.FindByName("*.ReleaseMutex"));
+CxList mutexClose = mutexList.FindByNames(new []{"*.Close", "*.Dispose", "*.ReleaseMutex"});

-CxList mutexCons = typeMutex.FindByType(typeof(ObjectCreateExpr));
+CxList mutexCons = typeMutex.FindByType<ObjectCreateExpr>();
 CxList usingMutexCons = mutexCons.FindByFathers(usingStmtDeclarators);
 mutexCons -= usingMutexCons;

@@ -14,6 +12,6 @@
```

```
 mutexOpen = typeMutex.FindDefinition(mutexOpen.GetTargetOfMembers());
 mutexOpen.Add(mutexCons);
-mutexOpen -= mutexOpen.FindByType(typeof(Declarator));
+mutexOpen -= mutexOpen.FindByType<Declarator>();

 result = mutexOpen - mutexOpen.DataInfluencingOn(mutexClose);
```

CSharp / CSharp_Medium_Threat / Insecure_Cookie

Code changes

```
---
+++
@@ -27,37 +27,34 @@
    CxList fieldDecls = Find_FieldDecls();
    CxList objectCreations = Find_ObjectCreations();

-   CxList possibleCookiesOptions = All.NewCxList();
-   possibleCookiesOptions.Add(methods, unknownRef, objectCreations);
+   CxList possibleCookiesOptions = All.NewCxList(methods, unknownRef, objectCreations);

-   CxList httpCookiesOrigins = All.NewCxList();
-   httpCookiesOrigins.Add(
+   CxList httpCookiesOrigins = All.NewCxList(
        objectCreations.FindByType("HttpCookie"),
        methods.FindByMemberAccess("*Cookies.Get"),
        memberAccesses.FindByShortName("*Cookies*").GetFathers().FindByType<IndexerRef>());
+
    httpCookiesOrigins -= memberAccesses.FindByShortName("Value").GetTargetOfMembers();

-   CxList cookiesFathers = All.NewCxList();
-   cookiesFathers.Add(parameters, returns);
-   CxList httpCookies = All.NewCxList();
-   httpCookies.Add(
+   CxList cookiesFathers = All.NewCxList(parameters, returns);
+
+   CxList httpCookies = All.NewCxList(
        httpCookiesOrigins.GetAssignee(),
        httpCookiesOrigins.FindByFathers(cookiesFathers));

    CxList trueLiterals = booleanLiterals.FindByShortName("true", false);
    CxList cookieSecurePolicyAlways = memberAccesses.FindByMemberAccess("CookieSecurePolicy.Always");

-   CxList secureFlag = All.NewCxList();
-   secureFlag.Add(
+   CxList secureFlag = All.NewCxList(
        trueLiterals.GetAssignee().FindByShortName("Secure"),
        cookieSecurePolicyAlways.GetAssignee().FindByShortName("SecurePolicy"));
+
    secureFlag.Add(secureFlag.GetByAncs(fieldDecls).GetAncOfType<ObjectCreateExpr>());
```

```
    secureFlag.Add(secureFlag.GetAssignee());


    CxList cookiesAppend = methods.FindByMemberAccess("*Cookies.Append");


-   CxList cookies = All.NewCxList();
-   cookies.Add(
+   CxList cookies = All.NewCxList(
        httpCookies,
        possibleCookiesOptions.GetParameters(cookiesAppend, 2));


@@ -76,7 +73,8 @@
 else if(ASPNetCoreCookiePolicyOptions.Count > 0){


    CxList SafeCookieSecurityPolicy =
-       memberAccesses.FindByMemberAccesses(new string[] {"CookieSecurePolicy.Always", "CookieSecurePolicy.SameAsRequest"});
+       memberAccesses.FindByMemberAccesses(new string[]
+       {"CookieSecurePolicy.Always", "CookieSecurePolicy.SameAsRequest"});


    /*
    The purpose of this section is to identify misconfigured Cookie policy options such as:
```

CSharp / CSharp_Medium_Threat / JWT_Lack_Of_Expiration_Time

Code changes

```
---
+++
@@ -1,12 +1,6 @@
 CxList tokenValidation = JWT_TokenValidationParameters();


-CxList listToValidate = All.NewCxList();
-CxList memberAccess = tokenValidation.FindByType(typeof(MemberAccess)).FindByShortName("RequireExpirationTime");
-CxList fieldDecl = tokenValidation.FindByType(typeof(FieldDecl)).FindByShortName("RequireExpirationTime");
+CxList listToValidate = tokenValidation.FindByTypes(typeof(MemberAccess), typeof(FieldDecl))
+   .FindByShortName("RequireExpirationTime");


-listToValidate.Add(memberAccess);
-listToValidate.Add(fieldDecl);
-
-CxList vulnerable = JWT_TokenParameter_Validation(listToValidate);
-
-result = vulnerable;
+result = JWT_TokenParameter_Validation(listToValidate);
```

CSharp / CSharp_Medium_Threat / JWT_No_Expiration_Time_Validation

Code changes

```
---
+++
@@ -1,12 +1,6 @@
 CxList tokenValidation = JWT_TokenValidationParameters();
```

```
-CxList listToValidate = All.NewCxList();
-CxList memberAccess = tokenValidation.FindByType(typeof(MemberAccess)).FindByShortName("ValidateLifetime");
-CxList fieldDecl = tokenValidation.FindByType(typeof(FieldDecl)).FindByShortName("ValidateLifetime");
+CxList listToValidate = tokenValidation.FindByTypes(typeof(MemberAccess), typeof(FieldDecl))
+   .FindByShortName("ValidateLifetime");

-listToValidate.Add(memberAccess);
-listToValidate.Add(fieldDecl);
-
-CxList vulnerable = JWT_TokenParameter_Validation(listToValidate);
-
-result = vulnerable;
+result = JWT_TokenParameter_Validation(listToValidate);
```

CSharp / CSharp_Medium_Threat / JWT_Sensitive_Information_Exposure

Code changes

```
---
+++
@@ -1,4 +1,3 @@
-CxList unkRefs = Find_UnknownReference();
 CxList sensitiveInfo = Find_Personal_Info();
 sensitiveInfo.Add(Find_All_Passwords());
@@ -10,14 +9,13 @@
 CxList allClaimsParams = All.GetParameters(claimsCreation);
 CxList firstAndSecParams = allClaimsParams.GetParameters(claimsCreation, 0);
 firstAndSecParams.Add(allClaimsParams.GetParameters(claimsCreation, 1));
-CxList sanitize = allClaimsParams - firstAndSecParams;

 CxList sensitiveParams = (firstAndSecParams * sensitiveInfo);
 // Add results from claims creation for SecurityTokenDescriptor
 foreach(CxList secToken in secTokenDesc) {
    CxList itemsToConsider = sensitiveParams.GetByAncs(secToken);
    foreach (CxList item in itemsToConsider) {
-       CxList claimObj = item.GetAncOfType(typeof(ObjectCreateExpr));
+       CxList claimObj = item.GetAncOfType<ObjectCreateExpr>();
        result.Add(item.ConcatenatePath(claimObj, false).ConcatenatePath(secToken, false));
    }
 }
```

CSharp / CSharp_Medium_Threat / Missing_Column_Encryption

Code changes

```
---
+++
@@ -1,14 +1,20 @@
-CxList columnEncryption = Find_Strings()
+CxList strings = Find_Strings();
```

```
+CxList methods = Find_Methods();
+CxList indexerRefs = Find_IndexerRefs();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
+
+CxList columnEncryption = strings
    .FindByShortName("Column Encryption Setting")
    .GetFathers();

 CxList enables = All.FindByMemberAccess("SqlConnectionColumnEncryptionSetting.Enabled");
-enables.Add(Find_Strings().FindByShortName("enabled", false));
+enables.Add(strings.FindByShortName("enabled", false));

 CxList abstractStrings = All.FindByAbstractValue(_ => _ is StringAbstractValue);
 IAbstractValue validEnabledUpper = new StringAbstractValue("Enabled");
 IAbstractValue validEnabledLower = new StringAbstractValue("enabled");
-CxList abstractEnables = abstractStrings.FindByAbstractValue(_ => _.Contains(validEnabledUpper) || _.Contains(validEnabledLower));
+CxList abstractEnables = abstractStrings.FindByAbstractValue(_ => _.Contains(validEnabledUpper)
+   || _.Contains(validEnabledLower));
 enables.Add(abstractEnables);

 // Get all the builder.ColumnEncryptionSetting = SqlConnectionColumnEncryptionSetting.Enabled
@@ -18,38 +24,40 @@
 CxList buildersEnabledRefs = All.FindAllReferences(buildersEnabled);

 // Get all builder["Column Encryption Setting"] = "Enabled"
-CxList indexers = All.FindByType("SqlConnectionStringBuilder").FindByType(typeof(IndexerRef));
+CxList indexers = indexerRefs.FindByType("SqlConnectionStringBuilder");

 CxList indexersEnabled = (columnEncryption * indexers).GetAssigner(enables);
-indexersEnabled.Add(All.FindAllReferences(indexersEnabled.GetAssignee()).GetByAncs(Find_IndexerRefs()));
+indexersEnabled.Add(All.FindAllReferences(indexersEnabled.GetAssignee()).GetByAncs(indexerRefs));

 // Get All builder.Add("Column Encryption Setting", "Enabled")
-CxList buildersAdd = All.FindByMemberAccess("SqlConnectionStringBuilder.Add");
+CxList buildersAdd = methods.FindByMemberAccess("SqlConnectionStringBuilder.Add");

 CxList buildersAddEnabled = buildersAdd.FindByParameters(columnEncryption)
    .FindByParameters(enables);

 //Safe ConnectionStrings
-CxList connStringSafe = All.FindByType(typeof(StringLiteral))
+CxList connStringSafe = strings
    .FindByRegex(@"Column Encryption Setting\s*=\s*[Ee]nabled");
 connStringSafe.Add(connStringSafe.GetAssignee());

-CxList safes = All.NewCxList();
-safes.Add(buildersEnabled);
-safes.Add(buildersEnabledRefs);
```

```
-safes.Add(indexersEnabled);
-safes.Add(connStringSafe);
+CxList safes = All.NewCxList(
+   buildersEnabled,
+   buildersEnabledRefs,
+   indexersEnabled,
+   connStringSafe,
+   buildersAddEnabled);

 //Find all methods connection.open
 CxList sqlConnections = All.FindByType("SqlConnection");
-CxList sqlCommandConnections = All.FindByMemberAccess("SqlCommand.Connection");
-CxList connections = sqlConnections;
-connections.Add(All.FindByMemberAccess("SqlCommand.Connection"));
+CxList connections = All.NewCxList(
+   sqlConnections,
+   All.FindByMemberAccess("SqlCommand.Connection"));
 CxList opens = connections.GetMembersOfTarget().FindByShortName("Open");
 //Find all first parameters of SqlConnection
-CxList paramsConnection = All.GetParameters(sqlConnections, 0);
-paramsConnection.Add(Find_Connection_String_Concat_Value());
-paramsConnection.Add(Find_Connection_String_Value());
+CxList paramsConnection = All.NewCxList(
+   paramValue.GetParameters(sqlConnections, 0),
+   Find_Connection_String_Concat_Value(),
+   Find_Connection_String_Value());

 //Remove all opens that are influenced by safes
 opens -= opens.InfluencedBy(safes);
```

CSharp / CSharp_Medium_Threat / MVC_View_Injection

Code changes

```
---
+++
@@ -9,7 +9,7 @@
 inputs.Add(Find_ASP_MVC_Inputs());

 // Outputs
-CxList outputs = methods.FindByShortNames(new List<string>(){"View", "PartialView"});
+CxList outputs = methods.FindByShortNames(new []{"View", "PartialView"});
 outputs = strings.GetParameters(outputs, 0);

 // Sanitizers
```

CSharp / CSharp_Medium_Threat / No_Request_Validation

Code changes

```
---
+++
```

```
@@ -1,6 +1,5 @@
 CxList allClasses = Find_Classes();
 CxList allAssign = Find_AssignExpr();
-CxList allUnknownRefs = Find_UnknownReference();
 CxList allCustomAttributes = Find_CustomAttribute();
 CxList allProperties = Find_PropertyDecl();
```

CSharp / CSharp_Medium_Threat / Persistent_Connection_String

Code changes

```
---
+++
@@ -17,7 +17,7 @@
    "OleDbConnectionStringBuilder",
    "OracleConnectionStringBuilder"};

-buildersPersistSecurity.Add(persistSecurity.FindByTypes(connectionStrings).FindByType(typeof(IndexerRef)));
+buildersPersistSecurity.Add(persistSecurity.FindByTypes(connectionStrings).FindByType<IndexerRef>());

 CxList AssingedToPersistSecurity = buildersPersistSecurity.GetAssigner(All);
```

CSharp / CSharp_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -5,6 +5,7 @@
 CxList integerLiteral = Find_IntegerLiterals();
 CxList nullLiteral = Find_NullLiteral();
 CxList methods = Find_Methods();
+CxList objectCreations = Find_ObjectCreations();
 CxList literals = All.NewCxList();
 literals.Add(strings, integerLiteral);

@@ -19,8 +20,7 @@
 // 2) Exclude constants that are assigned a literal
 CxList constants = personal_info.FindByType<ConstantDecl>();
 CxList allConstRef = personal_info.FindAllReferences(constants);
-CxList allConstRefOrigin = All.NewCxList();
-allConstRefOrigin.Add(allConstRef);
+CxList allConstRefOrigin = All.NewCxList(allConstRef);

 // Find all assignments of string or integer literals
 CxList constAssignedL = literals.FindByFathers(allConstRef.FindByType<Declarator>());
@@ -59,7 +59,7 @@
 personal_info.Add(personal_info * inputs);

 // 3) Add exceptions (that could be thrown) to outputs.
```

```
-CxList exceptions = Find_ObjectCreations().FindByName("*Exception");
+CxList exceptions = objectCreations.FindByName("*Exception");
 CxList exceptionsCtors = Find_ConstructorDecl().FindByName("*Exception");

 // Handle the case where the super (base) constructor of the exception is used to create a new throwable exception
@@ -78,15 +78,12 @@
 sanitize.Add(Find_Encrypt(), Find_Booleans());

 // Add methods that return information about the object - and not the object itself
-sanitize.Add(methods.FindByShortNames(new List<string> {"nameof", "typeof"}));
-sanitize.Add(All.FindByShortNames(new List<string> {
-       "GetType",
-       "GetHashCode",
-       "Equals"}));
+sanitize.Add(methods.FindByShortNames(new [] {"nameof", "typeof"}));
+sanitize.Add(All.FindByShortNames(new [] {"GetType", "GetHashCode", "Equals"}));

 // Add additional "integer" sanitizers
 CxList memberAccessesList = base.Find_MemberAccesses();
-sanitize.Add(memberAccessesList.FindByShortNames(new List<string> {"*Length*","*Index*","*Contains*", "*Count*"}, true));
+sanitize.Add(memberAccessesList.FindByShortNames(new [] {"*Length*","*Index*","*Contains*", "*Count*"}, true));

 // Add second argument of "Regex.Replace(input, pattern, ...)"
 sanitize.Add(All.GetParameters(methods.FindByMemberAccess("Regex.Replace"), 1));
@@ -94,6 +91,10 @@
 // Add response from web services: var response = MyWebService.Update(input)
 CxList webServiceResponses = Find_Web_Services().GetMembersOfTarget().FindByType<MethodInvokeExpr>().GetAssignee();
 sanitize.Add(webServiceResponses);
+
+// Add Gui objects to sanitizers
+CxList guidObjects = objectCreations.FindByShortName("Guid");
+sanitize.Add(guidObjects);

 // Split personal_info into variables and constants
 CxList variableRef = personal_info - allConstRef;
```

CSharp / CSharp_Medium_Threat / Race_Condition_within_a_Thread

Code changes

```
---
+++
@@ -1,13 +1,14 @@
 CxList thread = All.FindByType("Thread*");
 CxList decls = Find_Declarators();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 if(thread.Count > 0)
 {
    //find the thread initiation point
```

```
-   CxList delegatePassedToThread = All.GetParameters(thread);
+   CxList delegatePassedToThread = paramValue.GetParameters(thread);
    CxList methods = Find_Methods();
-   delegatePassedToThread.Add(All.GetParameters(delegatePassedToThread.FindByType<ObjectCreateExpr>()));
-   delegatePassedToThread.Add(All.GetParameters(thread.GetMembersOfTarget(), 0));
+   delegatePassedToThread.Add(paramValue.GetParameters(delegatePassedToThread.FindByType<ObjectCreateExpr>()));
+   delegatePassedToThread.Add(paramValue.GetParameters(thread.GetMembersOfTarget(), 0));
    delegatePassedToThread.Add(methods.GetByAncs(delegatePassedToThread.FindByType<LambdaExpr>()));

    CxList threadMethod = All.FindDefinition(delegatePassedToThread);
@@ -15,9 +16,9 @@

    stat = decls.GetByAncs(stat.FindByType<FieldDecl>());

-   CxList safe = All.NewCxList();
-   safe.Add(stat.FindByTypes(new string[]{"SynchronizedCollection", "BlockingCollection", "Concurrent*"}));
-   safe.Add(All.FindAllReferences(stat).GetAssigner().FindByShortName("Synchronized").GetAssignee());
+   CxList safe = All.NewCxList(
+       stat.FindByTypes(new string[]{"SynchronizedCollection", "BlockingCollection", "Concurrent*"}),
+       All.FindAllReferences(stat).GetAssigner().FindByShortName("Synchronized").GetAssignee());

    stat -= safe;

@@ -35,14 +36,13 @@
    CxList leftOfAssign = All.FindByAssignmentSide(CxList.AssignmentSide.Left);
    CxList potentialRC = All.NewCxList();
    check.Add(containsCheck.GetTargetOfMembers());
-   CxList modifiers = All.NewCxList();
-   modifiers.Add(leftOfAssign);
-   modifiers.Add(collectionModification);
+   CxList modifiers = All.NewCxList(leftOfAssign, collectionModification);
    foreach(CxList indexerUnderCheck in check)
    {
-       CxList parent = indexerUnderCheck.GetAncOfType<IfStmt>();
-       parent.Add(indexerUnderCheck.GetAncOfType<TernaryExpr>());
-       parent.Add(indexerUnderCheck.GetAncOfType<IterationStmt>());
+       CxList parent = All.NewCxList(
+           indexerUnderCheck.GetAncOfType<IfStmt>(),
+           indexerUnderCheck.GetAncOfType<TernaryExpr>(),
+           indexerUnderCheck.GetAncOfType<IterationStmt>());
        CxList valid = All.NewCxList();

        foreach(CxList pStmt in parent)
@@ -63,7 +63,7 @@

    foreach(CxList marker in potentialRC)
    {
-       CxList curMarker = marker.Clone();
```

```
+       CxList curMarker = All.NewCxList(marker);
        // Maximum search depth in the call graph
        int depth = 6;
        while(depth > 0)
```

CSharp / CSharp_Medium_Threat / Reflected_XSS_Specific_Clients

Code changes

```
---
+++
@@ -1,9 +1 @@
-if(All.isWebApplication || Check_Web_Application().Any())
-{
-   CxList inputs = Find_Interactive_Inputs();
-   CxList outputs = Find_Web_Outputs() - Find_XSS_Outputs() - Find_Safe_Response();
-
-   CxList sanitized = Find_XSS_Sanitize();
-
-   result = inputs.InfluencingOnAndNotSanitized(outputs, sanitized);
-}
+//This query is deprecated.
```

CSharp / CSharp_Medium_Threat / Session_Fixation

Code changes

```
---
+++
@@ -7,13 +7,13 @@
 CxList sessionAssigned = Find_Session_Create();

 CxList methods = sessionAssigned.GetMembersOfTarget();
-CxList indexerRefs = sessionAssigned.FindByType(typeof(IndexerRef));
+CxList indexerRefs = sessionAssigned.FindByType<IndexerRef>();
 CxList sessionIdIndexerRefs = indexerRefs.FindByShortName("*SessionID");
 CxList sessionIdMethods = methods.FindByShortName("SessionID");

 CxList inputs = Find_Inputs();

-CxList sessionIds = sessionIdMethods + sessionIdIndexerRefs.GetFathers();
+CxList sessionIds = All.NewCxList(sessionIdMethods, sessionIdIndexerRefs.GetFathers());
 CxList sessionIdsInfluenced = sessionIds.InfluencedBy(inputs);

 result = sessionIdsInfluenced;
```

CSharp / CSharp_Medium_Threat / SSL_Verification_Bypass

Code changes

```
---
+++
@@ -10,7 +10,7 @@
 CxList booleansTrue = Find_BooleanLiteral().FindByShortName("true", false);
```

```
 // outputs
-List<string> validCBNames = new List<string>{
+string[] validCBNames = new string[]{
        "ServerCertificateCustomValidationCallback", "ServerCertificateValidationCallback",
        "RemoteCertificateValidationCallback"};
 CxList validCallbacks = membAccs.FindByShortNames(validCBNames, false);
@@ -23,7 +23,7 @@

 // sanitizers
 // if boolean "true" inside an if condition with validation methods
-CxList certVerifyMethods = methods.FindByShortNames(new List<string>{
+CxList certVerifyMethods = methods.FindByShortNames(new []{
        "GetCertHash","GetCertHashString","GetHashCode","Verify"}, false);
 certVerifyMethods.Add(membAccs.FindByMemberAccesses(new string[] {
    "SslPolicyErrors.*"}, false));
@@ -70,8 +70,7 @@
 CxList vulnerableCtors = objCreations.GetParameters(objCreations.FindByShortName("SslStream"))
    .FindByShortName("RemoteCertificateValidationCallback");

-CxList delegateTargets = All.NewCxList();
-delegateTargets.Add(vulnerableCallbacks, vulnerableCtors);
+CxList delegateTargets = All.NewCxList(vulnerableCallbacks, vulnerableCtors);
```

```
 result.Add(callbackResults,
    x509Certs - safeCertFlows,
```

CSharp / CSharp_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -9,6 +9,7 @@
 CxList binaryExpr = Find_BinaryExpr();
 CxList memberAccess = Find_MemberAccesses();
 CxList allCastExpr = Find_CastExpr();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 CxList sinks = All.NewCxList();

@@ -72,10 +73,9 @@
 CxList webClientMethods = unknRefs.FindAllReferences(webClient).GetMembersOfTarget()
    .FindByShortNames(webClientMethodsNames);

-CxList clientMethods = All.NewCxList();
-clientMethods.Add(httpClientMethods, webRequestMethods, webClientMethods);
+CxList clientMethods = All.NewCxList(httpClientMethods, webRequestMethods, webClientMethods);

-CxList clientSinks = All.GetParameters(clientMethods, 0);
```

```
+CxList clientSinks = paramValue.GetParameters(clientMethods, 0);
 sinks.Add(clientSinks);

 //Sanitizers
@@ -111,7 +111,7 @@
 CxList sanitizedBaseAddress = sinks.Filter(x => x.ShortName == "BaseAddress").InfluencedBy(validPrefixs)
     .GetLastNodesInPath();
 CxList methodsInflBySanitizedBaseAddress = sinks.GetAncOfType<MethodInvokeExpr>().InfluencedBy(sanitizedBaseAddress);
-CxList sinksInflBySanitizedBaseAddress = All.GetParameters(methodsInflBySanitizedBaseAddress, 0);
+CxList sinksInflBySanitizedBaseAddress = paramValue.GetParameters(methodsInflBySanitizedBaseAddress, 0);

 CxList binaryInflByBaseAddress = sinksInflBySanitizedBaseAddress.FindByType<BinaryExpr>();
 binaryInflByBaseAddress.Add(sinksInflBySanitizedBaseAddress.FindByType<UnknownReference>()
@@ -156,10 +156,11 @@

 sanitizers.Add(sinks.GetByAncs(trueStmts));
 /*General sanitizers
-e.g. hashing, encrypting, casting to int/bool/float
+e.g. hashing, encrypting, integers and casting to int/bool/float
 */
 CxList generalSanitizers = Find_Hashing_Functions();
 generalSanitizers.Add(Find_Encrypt());
+generalSanitizers.Add(Find_Integers());

 //Casts to int/bool/float
 List<string> strType = new List<string> {"int", "Int16", "Int32", "Int64", "UInt64",
@@ -168,22 +169,9 @@
         "System.UInt64", "System.UInt32", "System.UInt16", "System.Decimal", "System.Single",
         "System.Boolean", "System.Double", "System.IntPtr","System.UIntPtr" };

-CxList castInts = All.NewCxList();
-foreach(CxList ace in allCastExpr)
-{
-   CastExpr ce = ace.TryGetCSharpGraph<CastExpr>();
-   if(ce != null)
-   {
-       string typeName = ce.TargetType.TypeName;
-       if(strType.Contains(typeName))
-       {
-           castInts.Add(ace);
-       }
-   }
-}
+CxList castInts = allCastExpr.FilterByDomProperty<CastExpr>(x => strType.Contains(x.TargetType.TypeName));

-CxList casts = All.NewCxList();
-casts.Add(unknRefs, methods, memberAccess);
+CxList casts = All.NewCxList(unknRefs, methods, memberAccess);
```

```
 castInts = casts.FindByFathers(castInts);
 castInts -= castInts.GetMembersOfTarget().GetTargetOfMembers();


@@ -195,7 +183,7 @@
 //Sockets
 CxList socket = objCreations.FindByShortName("Socket").GetAssignee();
 CxList socketMethods = unknRefs.FindAllReferences(socket).GetMembersOfTarget().FindByShortName("Connect");
-CxList socketSinks = All.GetParameters(socketMethods, 0);
+CxList socketSinks = paramValue.GetParameters(socketMethods, 0);


 CxList hostEntry = methods.FindByMemberAccess("Dns.GetHostEntry");
 CxList socketPaths = inputs.InfluencingOnAndNotSanitized(socketSinks, sanitizers);
```

CSharp / CSharp_Medium_Threat / Stored_LDAP_Injection

Code changes

```
---
+++
@@ -14,7 +14,6 @@
 ldap -= DNParams;
 DNParams = DNParams.FindByType("String");


-
     /*Detect input files*/
     CxList fileIO = All.NewCxList();
 {
@@ -24,13 +23,14 @@
         "FileStream",           "*.FileStream",
         "StreamReader",         "*.StreamReader"
         };
-    List<string> relevantClasses = new List<string> {
+    string[] relevantClasses = new string[] {
             "Directory",
             "File"
             };
     CxList objects = All.NewCxList(Find_Unknown_References(), Find_Declarators());
-    fileIO.Add(objects.FindByTypes(relevantTypes));
-    fileIO.Add(All.FindByShortNames(relevantClasses));
+    fileIO.Add(
+        objects.FindByTypes(relevantTypes),
+        All.FindByShortNames(relevantClasses));
     fileIO = fileIO.GetMembersOfTarget(); // We want the methods, not the objects
     fileIO.Add(Find_ASP_MVC_Controller_File_Result());
 }
```

CSharp / CSharp_Medium_Threat / Stored_Path_Traversal

Code changes

```
---
+++
```

```
@@ -17,8 +17,10 @@
    The output is then all the I/O methods whose pathname is an unsanitized
    result from Find_Read() + Find_DB_Out();.
 */
-CxList inputs = All.NewCxList();
-inputs.Add(Find_Read(), Find_DB_Out(), Find_Cloud_Storage_Inputs());
+CxList inputs = All.NewCxList(
+   Find_Read(),
+   Find_DB_Out(),
+   Find_Cloud_Storage_Inputs());

 string[] irrelevantInputs =
    new string[]{"FileInfo.Name", "FileInfo.Fullname", "FileInfo.Directory", "FileInfo.DirectoryName",
```

CSharp / CSharp_Medium_Threat / Stored_XPath_Injection

Code changes

```
---
+++
@@ -1,7 +1,9 @@
 CxList XPath = Find_XPath_Output();

-CxList inputs = All.NewCxList(Find_Read(), Find_Cloud_Storage_Inputs());
-inputs.Add(Find_DB_Out());
+CxList inputs = All.NewCxList(
+   Find_Read(),
+   Find_Cloud_Storage_Inputs(),
+   Find_DB_Out());

 CxList sanitized = Find_XPath_Injection_Sanitizers();
```

CSharp / CSharp_Medium_Threat / Unsafe_Object_Binding

Code changes

```
---
+++
@@ -8,6 +8,7 @@
    CxList parameters = Find_Param();
    CxList customAttributes = Find_CustomAttribute();
    CxList arrayCreation = Find_ArrayCreateExpr();
+   CxList paramValue = parameters.CxSelectDomProperty<Param>(p => p.Value);

    /*
        GET INPUTS
@@ -25,23 +26,21 @@
    CxList publicMethodsParameters = paramDecl.GetParameters(publicMethods);
    CxList publicMethodsParametersPrimitive = paramDecl.GetParameters(publicMethods).FindByTypes(primitives);
    CxList publicMethodsParametersBind = customAttributes.FindByShortName("Bind").GetAncOfType<ParamDecl>();
-   CxList safePublicMethodsParameters = All.NewCxList();
```

```
-   safePublicMethodsParameters.Add(publicMethodsParametersBind, publicMethodsParametersPrimitive);
+   CxList safePublicMethodsParameters = All.NewCxList(publicMethodsParametersBind, publicMethodsParametersPrimitive);
    CxList vulnerableParameters = publicMethodsParameters - safePublicMethodsParameters;


    inputs.Add(vulnerableParameters);


    //UpdateModels and TryUpdateModels 1st parameter
-   List<string> updateNames = new List<string>(){"TryUpdateModel", "TryUpdateModelAsync","UpdateModel", "UpdateModelAsync"};
+   string[] updateNames = new string[]{"TryUpdateModel", "TryUpdateModelAsync","UpdateModel", "UpdateModelAsync"};
    CxList updateMethods = methodsControllers.FindByShortNames(updateNames);
-   inputs.Add(All.FindDefinition(All.GetParameters(updateMethods, 0)));
+   inputs.Add(All.FindDefinition(paramValue.GetParameters(updateMethods, 0)));


    /*
        SaveChanges
    */
-   CxList saveChanges = methodsControllers.FindByShortNames(new List<string> {"SaveChanges", "SaveChangesAsync"}, true);
+   CxList saveChanges = methodsControllers.FindByShortNames(new []{"SaveChanges", "SaveChangesAsync"}, true);
    CxList dbSaveChanges = saveChanges * methodsControllers;
-


    /*
        updateMethods - Sanitize
```

CSharp / CSharp_Medium_Threat / Use_of_Cryptographically_Weak_PRNG

Code changes

```
---
+++
@@ -1,42 +1,36 @@
-CxList inputs = All.NewCxList();
-CxList outputs = All.NewCxList();
 CxList methods = Find_MethodDecls();
 CxList methodsList = Find_Methods();
 CxList paramList = Find_ParamDecl();
 CxList classDeclarations = Find_ClassDecl();
 CxList unknownReferences = Find_UnknownReference();
 CxList classInFrRandom = classDeclarations.InheritsFrom("Random");
-List<string> randomMethods = new List<string> {"Next", "NextDouble"};
-
-outputs.Add(Find_Encrypt());
-outputs.Add(Find_Decrypt());
-
-inputs = methodsList.FindByMemberAccesses(new string[] {"Random.Next", "Random.NextDouble", "Random.NextByte*"});
-
-inputs.Add(unknownReferences.GetParameters(methodsList.FindByMemberAccess("Random.NextByte*"))); // The NextBytes method returns by parameters
+string[] randomMethods = new string[] {"Next", "NextDouble"};


 /* Find Methods in classes that Inherits From Random */
```

```
 CxList methodsInFrRandom = methodsList.FindByShortNames(randomMethods).GetByAncs(classInFrRandom);
 methodsInFrRandom -= methodsInFrRandom.GetMembersWithTargets();

-inputs.Add(methodsInFrRandom);
-
 CxList nextBytesMethodInFrRandom = methodsList.FindByShortNames("NextByte*").GetByAncs(classInFrRandom);
 nextBytesMethodInFrRandom -= nextBytesMethodInFrRandom.GetMembersWithTargets();
-
-inputs.Add(unknownReferences.GetParameters(nextBytesMethodInFrRandom)); // The NextBytes method returns by parameters

 /* Find Methods From classes that extends Random */
 CxList methodsExtendRandom = methods.GetByAncs(classInFrRandom);
 CxList dangerousRandom = methodsExtendRandom.FindByShortNames(randomMethods);
 CxList refRandom = methodsList.FindAllReferences(dangerousRandom);

-inputs.Add(refRandom);
-
 CxList dangerousNextBytesRandom = methodsExtendRandom.FindByShortNames("NextByte*");
 CxList refNextBytesRandom = methodsList.FindAllReferences(dangerousNextBytesRandom);

-inputs.Add(unknownReferences.GetParameters(refNextBytesRandom)); // The NextBytes method returns by parameters
+CxList inputs = All.NewCxList(
+   methodsList.FindByMemberAccesses(new string[] {"Random.Next", "Random.NextDouble", "Random.NextByte*"}),
+   // The NextBytes method returns by parameters
+   unknownReferences.GetParameters(methodsList.FindByMemberAccess("Random.NextByte*")),
+   methodsInFrRandom,
+   // The NextBytes method returns by parameters
+   unknownReferences.GetParameters(nextBytesMethodInFrRandom),
+   refRandom,
+   // The NextBytes method returns by parameters
+   unknownReferences.GetParameters(refNextBytesRandom));

 /* Find Methods from Classes that extends HashAlgorithm */
 CxList methodsExtendHashAlgorithm = methods.GetByAncs(classDeclarations.InheritsFrom("HashAlgorithm"));
@@ -46,8 +40,11 @@
 CxList paramHashCore = paramList.GetParameters(dangerousHashCore, 0);
 CxList refComputeHash = methodsList.FindAllReferences(methods.FindByParameters(paramComputeHash));
 CxList refHashCore = methodsList.FindAllReferences(methods.FindByParameters(paramHashCore));
+CxList refCompHashAndHashCore = All.NewCxList(refComputeHash, refHashCore);

-outputs.Add(unknownReferences.GetParameters(refComputeHash));
-outputs.Add(unknownReferences.GetParameters(refHashCore));
+CxList outputs = All.NewCxList(
+   Find_Encrypt(),
+   Find_Decrypt(),
+   unknownReferences.GetParameters(refCompHashAndHashCore));

 result = inputs.InfluencingOn(outputs);
```

CSharp / CSharp_Medium_Threat / Use_of_Hard_coded_Cryptographic_Key

Code changes

```
---
+++
@@ -1,22 +1,23 @@
 /*
    Inputs
 */
+CxList unkRefs = Find_UnknownReference();
 CxList booleans = Find_BooleanLiteral();
-CxList emptyString = Find_Empty_Strings();
 CxList strLiterals = Find_String_Literal();
 CxList fieldsDecls = Find_FieldDecls().FindByType("byte[]");
-fieldsDecls.Add(Find_ArrayTypes().FindByType("byte").GetAncOfType(typeof(FieldDecl)));
+fieldsDecls.Add(Find_ArrayTypes().FindByType("byte").GetAncOfType<FieldDecl>());
 CxList bytes = fieldsDecls.GetAssigner().CxSelectElements<ArrayInitializer>(x => x.InitialValues);
 strLiterals.Add(bytes);
 CxList methods = Find_Methods();
 CxList NULL = All.FindByName("null");
 CxList OIDs = strLiterals.FindByRegex("([1-9][0-9]{0,3}|0)(\\.([1-9][0-9]{0,3}|0)){5,13}");
+CxList toRemove = All.NewCxList(methods, NULL, OIDs, booleans);

-CxList inputsAux = strLiterals.FindByType(typeof(PrimitiveExpr)) - emptyString - NULL - booleans - OIDs;
+CxList inputsAux = strLiterals.FindByType<PrimitiveExpr>() - toRemove;
 CxList inputs = All.NewCxList();

 foreach(CxList input in inputsAux){
-   if (input.FindByType(typeof(IntegerLiteral)).Count > 0){
+   if (input.FindByType<IntegerLiteral>().Count > 0){
        inputs.Add(input);
    } else {
        int length = input.GetName().Length;
@@ -52,7 +53,6 @@
    listSinks.Add(asymAlgName, new List<string>(){"FromXmlString"});
 }

-
 foreach(string className in new List<string>(listSinks.Keys)){
    foreach(string methodName in listSinks[className]){
        invoke.Add(methods.FindByMemberAccess(className, methodName));
@@ -66,14 +66,25 @@

 //SymetricAlgoritms that can influence CreateEncryptor by flow
 CxList createEncryptorMethod = methods.FindByShortName("CreateEncryptor");
-CxList symetricAlgoritmsDecls = Find_Declarators().GetAssigner().FindByShortNames(new List<string>{"AesCryptoServiceProvider","RijndaelManaged","DESCryptoServiceProvider","RC2CryptoServiceProvider", "TripleDESCryptoServiceProvider"}, false);
-symetricAlgoritmsDecls.Add(methods.FindByMemberAccesses(new string[]{"Aes.Create", "Rijndael.Create", "DES.Create", "RC2.Create", "TripleDes.Create"}));
+CxList symetricAlgoritmsDecls = Find_Declarators().GetAssigner()
+   .FindByShortNames(new List<string>{
```

```
+       "AesCryptoServiceProvider",
+       "RijndaelManaged",
+       "DESCryptoServiceProvider",
+       "RC2CryptoServiceProvider",
+       "TripleDESCryptoServiceProvider"}, false);
+symetricAlgoritmsDecls.Add(methods.FindByMemberAccesses(new string[]{
+   "Aes.Create",
+   "Rijndael.Create",
+   "DES.Create",
+   "RC2.Create",
+   "TripleDes.Create"}));
 CxList symetricCreators = createEncryptorMethod.InfluencedBy(symetricAlgoritmsDecls).GetLastNodesInPath();
 sinks.Add(symetricCreators);

 //For algoritms declared like DES des = new DESCryptoServiceProvider();
 CxList symetricAlgoritmsAssignee = symetricAlgoritmsDecls.GetAssignee();
-CxList symetricAlgoritmsAllRefs = Find_UnknownReference().FindAllReferences(symetricAlgoritmsAssignee);
+CxList symetricAlgoritmsAllRefs = unkRefs.FindAllReferences(symetricAlgoritmsAssignee);
 CxList symetricCreatorsWithUnknwon = createEncryptorMethod.InfluencedBy(symetricAlgoritmsAllRefs).GetLastNodesInPath();
 sinks.Add(symetricCreatorsWithUnknwon);

@@ -93,7 +104,7 @@
 */
 CxList sanitizers = Find_CollectionAccesses();
-sanitizers.Add(Find_Same_Value_Sanitizers_For_Hardcoded_Password(sinks + inputs));
+sanitizers.Add(Find_Same_Value_Sanitizers_For_Hardcoded_Password(All.NewCxList(sinks, inputs)));

 result = sinks.InfluencedByAndNotSanitized(inputs, sanitizers);

@@ -101,6 +112,39 @@
 CxList objAesGcmCcm = Find_ObjectCreations().FindByTypes(new String[]{"AesGcm","AesCcm"});
 CxList paramObjKey = All.GetParameters(objAesGcmCcm);
 CxList stringsToTest = Find_Strings();
-sanitizers = Find_Same_Value_Sanitizers_For_Hardcoded_Password(paramObjKey + stringsToTest);
+sanitizers = Find_Same_Value_Sanitizers_For_Hardcoded_Password(All.NewCxList(paramObjKey, stringsToTest));

 result.Add(paramObjKey.InfluencedByAndNotSanitized(stringsToTest, sanitizers));
+
+//Iterate over config files to connect with configured keys there.
+CxList configFlows = Find_IndexerRefs().FindByShortName("AppSettings").InfluencingOn(sinks);
+
+Dictionary<string, CxList> keyAndFlowDic = new Dictionary<string, CxList>();
+foreach(CxList configFlow in configFlows.GetCxListByPath())
+{
+   CxList thisKey = strLiterals.FindByFathers(configFlow.GetFirstNodesInPath());
+   keyAndFlowDic.Add(thisKey.GetName(), configFlow);
+}
```

```
+
+List<CxXmlDoc> configs = cxXPath.GetXmlFiles("Web.config").ToList();
+configs.AddRange(cxXPath.GetXmlFiles("App.config").ToList());
+
+foreach(CxXmlDoc doc in configs){
+   string docPath = doc.FileName;
+   CxList keysAndValues = strLiterals.FindByFileName(docPath);
+   XPathNavigator nav = doc.CreateNavigator();
+
+   foreach(string thisKey in keyAndFlowDic.Keys)
+   {
+
+       XPathNodeIterator nodeIterator = nav.Select(
+           string.Concat("/configuration/appSettings/add[@key=\"", thisKey, "\"]/@value"));
+
+       //for evey hardcoded key in the config file we connect it the the usages in the code
+       while(nodeIterator.MoveNext())
+       {
+           CxList thisNode = keysAndValues.FindByShortName(nodeIterator.Current.ToString());
+           result.Add(thisNode.ConcatenatePath(keyAndFlowDic[thisKey]));
+       }
+   }
+}
```

CSharp / CSharp_Metadata / ActiveMQ_Queues

Code changes

```
---
+++
@@ -1,13 +1,13 @@
 CxList methods = Find_Methods();
 CxList strings = Find_Strings();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 // using of ActiveMQ.Artemis nugget
 CxList addActiveMq = methods.FindByShortName("AddActiveMq");
-CxList activeMqNames = All.GetParameters(addActiveMq, 0).FindByType(typeof(StringLiteral));
+CxList activeMqNames = strings.GetParameters(addActiveMq, 0);
 CxList writeMethods = methods.FindByShortName("*Producer");
-CxList readMethods  = methods.FindByShortName("*Consumer");
-CxList readWriteMethods = All.NewCxList();
-readWriteMethods.Add(readMethods, writeMethods);
+CxList readMethods = methods.FindByShortName("*Consumer");
+CxList readWriteMethods = All.NewCxList(readMethods, writeMethods);

 foreach (CxList activeMqName in activeMqNames.FindByAbstractValue(s => s is StringAbstractValue))
 {
@@ -40,17 +40,10 @@
 // using of Apache.NMS.ActiveMQ nugget
```

```
 string[] sessions = new [] { "ISession" };


-CxList readSinks = All.NewCxList();
-CxList writeSinks = All.NewCxList();
-foreach (string session in sessions)
-{
-   writeSinks.Add(
-       All.GetParameters(methods.FindByMemberAccess(session, "CreateProducer"), 0));
-
-   readSinks.Add(
-       All.GetParameters(methods.FindByMemberAccess(session, "CreateConsumer"), 0),
-       All.GetParameters(methods.FindByMemberAccess(session, "CreateDurableConsumer"), 0));
-}
+CxList readSinks = All.NewCxList(
+   paramValue.GetParameters(methods.FindByMemberAccesses(sessions, new []{"CreateConsumer"}), 0),
+   paramValue.GetParameters(methods.FindByMemberAccesses(sessions, new []{"CreateDurableConsumer"}), 0));
+CxList writeSinks = paramValue.GetParameters(methods.FindByMemberAccesses(sessions, new []{"CreateProducer"}), 0);

 CxList sanitizers = All.FindByTypes(sessions);


@@ -59,9 +52,9 @@
    Add_Action_Type_to_Flow(strings.InfluencingOnAndNotSanitized(readSinks, sanitizers), "Read"));

 // using of MassTransit.ActiveMqTransport nugget
-CxList usingMassTransit = methods.FindByShortNames(new List<string>() {"CreateUsingActiveMq", "UsingActiveMq"});
+CxList usingMassTransit = methods.FindByShortNames(new [] {"CreateUsingActiveMq", "UsingActiveMq"});
 if (usingMassTransit.Count > 0)
 {
-   CxList receiveEndPoint = methods.FindByShortNames(new List<string>() {"ReceiveEndpoint"});
+   CxList receiveEndPoint = methods.FindByShortNames(new [] {"ReceiveEndpoint"});
    result.Add(Add_Action_Type_to_Flow(strings.InfluencingOn(receiveEndPoint), "Unknown"));
 }
```

CSharp / CSharp_Metadata / AWS_S3_Bucket_Usages

Code changes

```
---
+++
@@ -7,9 +7,6 @@
 * 2) We use a heuristic to find all variables that seem to be holding bucket names.
 */
 CxList methods = Find_Methods();
-CxList unkRef = Find_Unknown_References();
-CxList memberAcc = Find_MemberAccesses();
-CxList strings = Find_Strings();
 CxList objectCreateExpressions = base.Find_ObjectCreations();
 Func<string, bool> canBeBucketName = s => new Regex("^([0-9a-z.-]){3,63}$").IsMatch(s);


@@ -49,11 +46,10 @@
```

```
        };


 CxList amazonAcc = methods.FindByMemberAccesses(amazonS3Sdks);
-CxList unknownUsages = amazonAcc;
+CxList unknownUsages = All.NewCxList(amazonAcc);
 CxList readUsages = amazonAcc.FindByShortNames(readMthds);
 CxList writeUsages = amazonAcc.FindByShortNames(writeMthds);
-CxList knownUsages = All.NewCxList();
-knownUsages.Add(readUsages, writeUsages);
+CxList knownUsages = All.NewCxList(readUsages, writeUsages);


 unknownUsages -= knownUsages;
 unknownUsages -= unknownUsages.FindByShortNames(new [] {"getRegion", "getRegionName", "getS3AccountOwner",
@@ -77,9 +73,10 @@


 string[] requestSetNameMethods = new [] { "bucket", "withBucketName", "setBucketName" };


-result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(readUsages, requestSetNameMethods, "Name", "Read"));
-result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(writeUsages, requestSetNameMethods, "Name", "Write"));
-result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(unknownUsages, requestSetNameMethods, "Name", "Unknown"));
+result.Add(
+   Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(readUsages, requestSetNameMethods, "Name", "Read"),
+   Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(writeUsages, requestSetNameMethods, "Name", "Write"),
+   Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(unknownUsages, requestSetNameMethods, "Name", "Unknown"));


 /*
 * Part 2: A heuristic to find all variables that seem to be holding bucket names
```

CSharp / CSharp_Metadata / AWS_SNS_Usages

Code changes

```
---
+++
@@ -4,9 +4,7 @@
         "AmazonSimpleNotificationServiceClient",
         "SimpleNotificationService", "IAmazonSimpleNotificationService", "AmazonSimpleNotificationService"};


-CxList usages = All.NewCxList();
-foreach (var sdk in amazonSnsSdks)
-   usages.Add(methods.FindByMemberAccess(sdk, "*"));
+CxList usages = All.NewCxList(methods.FindByMemberAccesses(amazonSnsSdks, new []{"*"}));


 string[] requestSetNameMethods = new [] { "setSubscriptionArn", "withSubscriptionArn", "subscriptionArn", "withTopicArn",
         "setTopicArn", "topicArn"};
```

CSharp / CSharp_Metadata / AWS_SQS_Usages

Code changes

```
---
+++
```

```
@@ -1,28 +1,16 @@
 CxList methods = Find_Methods();

 string[] amazonSqsSdks = new [] { "Amazon.SQS", "AmazonSQSAsync", "IAmazonSQS", "AmazonSQSClient",
-//     "AmazonSQSAsyncClient", "AmazonSQSBufferedAsyncClient", "AbstractAmazonSQS", "SqsClient"
+       //      "AmazonSQSAsyncClient", "AmazonSQSBufferedAsyncClient", "AbstractAmazonSQS", "SqsClient"
        };
 string[] requestSetNameMethods = new [] { "withQueueName", "setQueueName", "withQueueUrl", "setQueueUrl", "withTopicArn",
        "setTopicArn", "queueName"};

-CxList readUsages = All.NewCxList();
-foreach (var sdk in amazonSqsSdks)
-{
-   readUsages.Add(methods.FindByMemberAccess(sdk, "ReceiveMessage*"));
-   readUsages.Add(methods.FindByMemberAccess(sdk, "GetQueueAttributesAsync"));
-}
+CxList readUsages = methods.FindByMemberAccesses(amazonSqsSdks, new []{"ReceiveMessage*", "GetQueueAttributesAsync"});

 result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(readUsages, requestSetNameMethods, "QueueUrl", "Read"));

-CxList writeUsages = All.NewCxList();
-foreach (var sdk in amazonSqsSdks)
-{
-   writeUsages.Add(methods.FindByMemberAccess(sdk, "SendMessage*"));
-   writeUsages.Add(methods.FindByMemberAccess(sdk, "DeleteMessageAsync"));
-   writeUsages.Add(methods.FindByMemberAccess(sdk, "PurgeQueueAsync"));
-   writeUsages.Add(methods.FindByMemberAccess(sdk, "DeleteMessageAsync"));
-   writeUsages.Add(methods.FindByMemberAccess(sdk, "DeleteQueueAsync"));
-}
+CxList writeUsages = methods.FindByMemberAccesses(amazonSqsSdks, new []{"SendMessage*", "DeleteMessageAsync",
+       "PurgeQueueAsync", "DeleteMessageAsync", "DeleteQueueAsync"});

 result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(writeUsages, requestSetNameMethods, "QueueUrl", "Write"));
```

CSharp / CSharp_Metadata / Kafka_Topics

Code changes

```
---
+++
@@ -2,14 +2,9 @@
 CxList methods = Find_Methods();
 CxList strings = Find_Strings();
 CxList members = Find_MemberAccesses();
-CxList decls = Find_Declarators();
 CxList unknownRefs = Find_Unknown_References();

-CxList varScope = All.NewCxList();
-varScope.Add(unknownRefs, decls);
-
```

```
-CxList topicScope = All.NewCxList();
-topicScope.Add(unknownRefs, strings, members);
+CxList topicScope = All.NewCxList(unknownRefs, strings, members);

 /*   KNet   */

@@ -56,8 +51,7 @@
 CxList confluentProducers = methods.GetMembersWithTargets(producerBuilders, 10).FindByShortName("Build").GetAssignee();
 confluentProducers = unknownRefs.FindAllReferences(confluentProducers);

-CxList confluentProduces = All.NewCxList();
-confluentProduces.Add(
+CxList confluentProduces = All.NewCxList(
     confluentProducers.GetMembersOfTarget().FindByShortNames(new string[] {"Produce", "ProduceAsync"}),
     methods.FindByMemberAccesses(new string[]{"IProducer.Produce", "IProducer.ProduceAsync"}));

@@ -82,23 +76,17 @@
 confluentConsumers = unknownRefs.FindAllReferences(confluentConsumers);
 CxList confluentConsumerMembers = confluentConsumers.GetMembersOfTarget();

-CxList confluentConsumes = All.NewCxList();
-confluentConsumes.Add(
+CxList confluentConsumes = All.NewCxList(
     confluentConsumerMembers.FindByShortName("Consume"),
-    methods.FindByMemberAccess("IConsumer.Consume")
-);
+    methods.FindByMemberAccess("IConsumer.Consume"));

-CxList confluentSubscribes = All.NewCxList();
-confluentSubscribes.Add(
+CxList confluentSubscribes = All.NewCxList(
     confluentConsumerMembers.FindByShortName("Subscribe"),
-    methods.FindByMemberAccess("IConsumer.Subscribe")
-);
+    methods.FindByMemberAccess("IConsumer.Subscribe"));

-CxList confluentAssigns = All.NewCxList();
-confluentAssigns.Add(
+CxList confluentAssigns = All.NewCxList(
     confluentConsumerMembers.FindByShortName("Assign"),
-    methods.FindByMemberAccess("IConsumer.Assign")
-);
+    methods.FindByMemberAccess("IConsumer.Assign"));

 foreach (CxList consume in confluentConsumes)
 {
@@ -108,11 +96,10 @@
```

```
    CxList topicParts = constructors.FindByShortName("TopicPartitionOffset").DataInfluencingOn(assigns);


-   CxList topics = All.NewCxList();
-   topics.Add(
+   CxList topics = All.NewCxList(
        topicScope.GetParameters(subs, 0),
-       topicScope.GetParameters(topicParts.GetFirstNodesInPath(),0));
-
+       topicScope.GetParameters(topicParts.GetFirstNodesInPath(), 0));
+
    foreach (CxList topic in topics.FindByAbstractValue(s => s is StringAbstractValue))
    {
        string absVal = topic.CxSelectElementValues<Expression, string>(exp => exp.AbsValue.ToString())[0];
```

CSharp / CSharp_Metadata / Memcached_Urls

Code changes

```
---
+++
@@ -4,5 +4,6 @@

 CxList strings = Find_Strings();
 CxList flow = strings.InfluencingOn(memcachedClients);
-result.Add(Add_Action_Type_to_Flow(flow, "Read"));
-result.Add(Add_Action_Type_to_Flow(flow, "Write"));
+result.Add(
+   Add_Action_Type_to_Flow(flow, "Read"),
+   Add_Action_Type_to_Flow(flow, "Write"));
```

CSharp / CSharp_Metadata / RabbitMQ_Queues

Code changes

```
---
+++
@@ -1,9 +1,11 @@
 CxList methods = Find_Methods();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
+CxList strings = Find_Strings();

-CxList writeDef = methods.FindByShortNames(new List<string>() { "BasicPublish"});
-CxList readDef  = methods.FindByShortNames(new List<string>() { "BasicConsume"});
+CxList writeDef = methods.FindByShortNames(new [] { "BasicPublish"});
+CxList readDef  = methods.FindByShortNames(new [] { "BasicConsume"});

-CxList methodsFirstParameter = All.GetParameters(methods, 0);
+CxList methodsFirstParameter = paramValue.GetParameters(methods, 0);
 CxList readParameters = methodsFirstParameter.GetParameters(readDef, 0);
 foreach (CxList readParameter in readParameters.FindByAbstractValue(s => s is StringAbstractValue))
 {
@@ -17,7 +19,7 @@
```

```
 // For now we only support publishing to the default exchange - where the queue is the same as the routingKey.
 // Publishing to other exchanges will be ignored.
-CxList writeParameters = All.GetParameters(writeDef);
+CxList writeParameters = paramValue.GetParameters(writeDef);
 foreach (CxList writeUse in writeDef)
 {
    CxList exchangeName = writeParameters.GetParameters(writeUse, 0);
@@ -36,10 +38,10 @@
 }

 // using of MassTransit.ActiveMqTransport nugget
-CxList usingMassTransit = methods.FindByShortNames(new List<string>() {"CreateUsingRabbitMq", "UsingRabbitMq"});
+CxList usingMassTransit = methods.FindByShortNames(new [] {"CreateUsingRabbitMq", "UsingRabbitMq"});
 if (usingMassTransit.Count > 0)
 {
-   CxList receiveEndPoints = methods.FindByShortNames(new List<string>() {"ReceiveEndpoint"});
+   CxList receiveEndPoints = methods.FindByShortNames(new [] {"ReceiveEndpoint"});
    CxList receiveEndPointsParam = methodsFirstParameter.GetParameters(receiveEndPoints, 0);

    foreach (CxList receiveEndPoint in receiveEndPoints)
@@ -75,8 +77,7 @@
 // add support for read
 CxList endpointsConfigurationCreation = Find_ObjectCreations().FindByShortName("EndpointConfiguration");
-CxList endpointsConfigurationCreationFirstParam = All.GetParameters(endpointsConfigurationCreation, 0).
-                                                  FindByType<StringLiteral>();
+CxList endpointsConfigurationCreationFirstParam = strings.GetParameters(endpointsConfigurationCreation, 0);

 CxList startWork = methods.FindByMemberAccess("Endpoint.Start");
```

CSharp / CSharp_Metadata / Redis_Urls

Code changes

```
---
+++
@@ -1,20 +1,20 @@
 // StackExchange.Redis
 CxList methods = Find_Methods();
 CxList stackExchangesRedis = methods.FindByMemberAccess("ConnectionMultiplexer.Connect");
-//CxList paramList = Find_Param();

 string[] requestSetNameMethods = new [] { "Connect"};
-
-result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(stackExchangesRedis, requestSetNameMethods, "", "Write"));
-result.Add(Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(stackExchangesRedis, requestSetNameMethods, "", "Read"));

 //SignalR
```

```
 string[] useRedis = new [] { "UseStackExchangeRedis", "UseRedis"};
 CxList useRedisMethods = methods.FindByShortNames(useRedis);
-result.Add(AddAccordingToRequestParam(useRedisMethods, 3, "Read"));
-result.Add(AddAccordingToRequestParam(useRedisMethods, 3, "Write"));


 string[] addRedis = new [] { "AddStackExchangeRedis", "AddRedis"};
 CxList addRedisMethods = methods.FindByShortNames(addRedis);
-result.Add(AddAccordingToRequestParam(addRedisMethods, 1, "Read"));
-result.Add(AddAccordingToRequestParam(addRedisMethods, 1, "Write"));
+
+result = All.NewCxList(
+   Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(stackExchangesRedis, requestSetNameMethods, "", "Write"),
+   Add_Name_For_AWS_Cloud_Sdk_Recourse_Methods(stackExchangesRedis, requestSetNameMethods, "", "Read"),
+   AddAccordingToRequestParam(useRedisMethods, 3, "Read"),
+   AddAccordingToRequestParam(useRedisMethods, 3, "Write"),
+   AddAccordingToRequestParam(addRedisMethods, 1, "Read"),
+   AddAccordingToRequestParam(addRedisMethods, 1, "Write"));
```

CSharp / CSharp_WebConfig / CookieLess_Session_State

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList webConfig = Find_Web_Config();
-CxList value_UseUri = webConfig.FindByName("UseUri").FindByType(typeof(StringLiteral));
+CxList value_UseUri = webConfig.FindByName("UseUri").FindByType<StringLiteral>();
 CxList sessionState_cookieless = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.SESSIONSTATE.COOKIELESS");


 result = value_UseUri * value_UseUri.DataInfluencingOn(sessionState_cookieless);
```

CSharp / CSharp_WebConfig / CustomError

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList webConfig = Find_Web_Config();
-CxList value_Off = webConfig.FindByName("Off").FindByType(typeof(StringLiteral));
+CxList value_Off = webConfig.FindByName("Off").FindByType<StringLiteral>();
 CxList customers_mode = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.CUSTOMERRORS.MODE");


 result = value_Off * value_Off.DataInfluencingOn(customers_mode);
```

CSharp / CSharp_WebConfig / DebugEnabled

Code changes

```
---
+++
@@ -1,4 +1,4 @@
 CxList webConfig = Find_Web_Config();
```

```
-CxList value_true = webConfig.FindByName("true").FindByType(typeof(StringLiteral));
+CxList value_true = webConfig.FindByName("true").FindByType<StringLiteral>();
 CxList compilation_Debug = webConfig.GetMembersOfTarget().GetMembersOfTarget().FindByShortName("debug", false);
 result = value_true * value_true.DataInfluencingOn(compilation_Debug);
```

CSharp / CSharp_WebConfig / Directory_Browse

Code changes

```
---
+++
@@ -5,13 +5,13 @@

 */
 CxList webConfig = Find_Web_Config();
-CxList ma = webConfig.FindByType(typeof(MemberAccess));
-CxList enabled =ma.FindByShortName("WEBSERVER").GetByAncs(webConfig.FindByShortName("DIRECTORYBROWSE"));
+CxList ma = webConfig.FindByType<MemberAccess>();
+CxList enabled = ma.FindByShortName("WEBSERVER").GetByAncs(webConfig.FindByShortName("DIRECTORYBROWSE"));

-CxList ae=enabled.GetAncOfType(typeof(AssignExpr));
+CxList ae = enabled.GetAncOfType<AssignExpr>();

 CxList curAss = ma.FindByShortName("ENABLED").FindByFathers(ae);
-ae=curAss.GetAncOfType(typeof(AssignExpr));
+ae = curAss.GetAncOfType<AssignExpr>();
-CxList sl = webConfig.FindByType(typeof(StringLiteral)).FindByFathers(ae).FindByShortName("true");
-result= sl.FindByAssignmentSide(CxList.AssignmentSide.Right);
+CxList sl = webConfig.FindByType<StringLiteral>().FindByFathers(ae).FindByShortName("true");
+result = sl.FindByAssignmentSide(CxList.AssignmentSide.Right);
```

CSharp / CSharp_WebConfig / Elmah_Enabled

Code changes

```
---
+++
@@ -31,34 +31,34 @@
    }
 }

-var unknownRef = Find_UnknownReference();
-var appConfReferences = unknownRef.FindByTypes("IApplicationBuilder", "IServiceCollection");
-var appConfMembers = appConfReferences.GetMembersOfTarget();
-var getAddElmah = appConfMembers.FindByShortName("AddElmah");
+CxList unknownRef = Find_UnknownReference();
+CxList appConfReferences = unknownRef.FindByTypes("IApplicationBuilder", "IServiceCollection");
+CxList appConfMembers = appConfReferences.GetMembersOfTarget();
+CxList getAddElmah = appConfMembers.FindByShortName("AddElmah");

-var useElmah = appConfMembers.FindByShortNames("UseElmah");
```

```
+CxList useElmah = appConfMembers.FindByShortNames("UseElmah");


 if(getAddElmah.Count > 0){

    //Issue in netcore Elmah plugin, auth only works if added before Elmah
-   var getAddElmahApp = getAddElmah.GetTargetOfMembers();
-   var getAllInfluencinguseElmah = unknownRef.DataInfluencingOn(getAddElmahApp)
+   CxList getAddElmahApp = getAddElmah.GetTargetOfMembers();
+   CxList getAllInfluencinguseElmah = unknownRef.DataInfluencingOn(getAddElmahApp)
        .GetStartAndEndNodes(CxList.GetStartEndNodesType.AllNodes);

-   var getMembersofInfluencingElmah = getAllInfluencinguseElmah.GetMembersOfTarget();
+   CxList getMembersofInfluencingElmah = getAllInfluencinguseElmah.GetMembersOfTarget();

-   var useAuthentication = getMembersofInfluencingElmah.FindByShortNames("UseAuthentication");
-   var useAuthorization = getMembersofInfluencingElmah.FindByShortNames("UseAuthorization");
+   CxList useAuthentication = getMembersofInfluencingElmah.FindByShortNames("UseAuthentication");
+   CxList useAuthorization = getMembersofInfluencingElmah.FindByShortNames("UseAuthorization");

-   if(useAuthentication.Count <= 0 || useAuthentication.Count <= 0){
+   if(useAuthentication.Count <= 0 || useAuthorization.Count <= 0){

        result.Add(useElmah);
    }

-   var addAuth = appConfReferences.GetMembersOfTarget().FindByShortNames("AddAuthorization");
+   CxList addAuth = appConfReferences.GetMembersOfTarget().FindByShortNames("AddAuthorization");

-   var requireRoles = Find_Strings().GetByAncs(addAuth).DataInfluencingOn(Find_Methods().FindByShortName("RequireRole"))
-       .GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly);
+   CxList requireRoles = Find_Strings().GetByAncs(addAuth).DataInfluencingOn(Find_Methods()
+       .FindByShortName("RequireRole")).GetFirstNodesInPath();

    if( requireRoles.Count > 0 && requireRoles.GetName().Equals("*")){
        result.Add(addAuth);
```

CSharp / CSharp_WebConfig / HardcodedCredentials

Code changes

```
---
+++
@@ -2,11 +2,13 @@
 CxList userName_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.CREDENTIALS.USER.NAME");
 CxList password_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.CREDENTIALS.USER.PASSWORD");

-if ((userName_exist + password_exist).Count == 1)
+CxList usernameAndPassExist = All.NewCxList(userName_exist, password_exist);
+
+if (usernameAndPassExist.Count == 1)
```

```
 {
-   result = userName_exist + password_exist;
+   result = usernameAndPassExist;
 }
-if ((userName_exist + password_exist).Count > 1)
+if (usernameAndPassExist.Count > 1)
 {
    result = userName_exist;
 }
```

CSharp / CSharp_WebConfig / Missing_X_Frame_Options

Code changes

```
---
+++
@@ -16,7 +16,7 @@
        CxList webConfig = Find_Web_Config().FindByFileName("*web.config");

        // Find the XML config class of the web.config (first line)
-       result = webConfig.FindByName("CxXmlConfigClass*", false).FindByType(typeof(ClassDecl));
+       result = webConfig.FindByName("CxXmlConfigClass*", false).FindByType<ClassDecl>();

        // If web.config is absent, just find first position in first file.
        if(result.Count == 0)
```

CSharp / CSharp_WebConfig / NonUniqueFormName

Code changes

```
---
+++
@@ -1,11 +1,10 @@
 CxList webConfig = Find_Web_Config();
-CxList value_Forms = webConfig.FindByName("Forms").FindByType(typeof(StringLiteral));
-CxList value_ASPXAUTH = webConfig.FindByName(".ASPXAUTH").FindByType(typeof(StringLiteral));
+CxList value_Forms = webConfig.FindByName("Forms").FindByType<StringLiteral>();
+CxList value_ASPXAUTH = webConfig.FindByName(".ASPXAUTH").FindByType<StringLiteral>();

 CxList formName_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.NAME");
 CxList forms_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS");
 CxList forms_childs = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.*");
-CxList configuration = webConfig.FindByName("CONFIGURATION");

 CxList mode_forms = value_Forms.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.MODE"));
 CxList formName = value_ASPXAUTH.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.NAME"));
@@ -19,7 +18,7 @@
 }
 else
 {
-   if ((mode_forms + formName).Count >= 2)
+   if (All.NewCxList(mode_forms, formName).Count >= 2)
```

```
    {
        result = value_ASPXAUTH * formName;
    }
```

CSharp / CSharp_WebConfig / Password_in_Configuration_File

Code changes

```
---
+++
@@ -16,7 +16,7 @@
 CxList config = Find_Web_Config();
 config.Add(All.FindByFileName("*app.config"));
 // Get all fils in config files
-CxList strings = config.FindByType(typeof(StringLiteral));
+CxList strings = config.FindByType<StringLiteral>();
 // Find all conditions in a config file
 CxList conditions = config * Find_Conditions();
 // Find all password string in config files
@@ -31,22 +31,22 @@
 CxList passwordKey = passwordString.FindByFathers(key.GetFathers());

 // b. A value is set to an element named "Password"
-CxList configMembers = config.FindByType(typeof(MemberAccess));
+CxList configMembers = config.FindByType<MemberAccess>();

-CxList passwordName = All.NewCxList();
-passwordName.Add(passwordInConfig);
-passwordName.Add(configMembers.FindByShortName("*PASS"));
+CxList passwordName = All.NewCxList(
+   passwordInConfig,
+   configMembers.FindByShortName("*PASS"));

 passwordName = passwordName.DataInfluencedBy(strings);

 // Get all results
-result.Add(passwordInConfig);
-result.Add(passwordKey);
-result.Add(passwordName);
-
-result.Add(Find_Password_In_AppSettings());
+result.Add(
+   passwordInConfig,
+   passwordKey,
+   passwordName,
+   Find_Password_In_AppSettings());

 // Remove results under SecureAppSettings
 CxList secureAppSettings = conditions.FindByShortName("SECUREAPPSETTINGS");
-secureAppSettings = secureAppSettings.GetAncOfType(typeof(IfStmt));
```

```
+secureAppSettings = secureAppSettings.GetAncOfType<IfStmt>();
 result -= result.GetByAncs(secureAppSettings);
```

CSharp / CSharp_WebConfig / RequireSSL

Code changes

```
---
+++
@@ -1,11 +1,10 @@
 CxList webConfig = Find_Web_Config();
-CxList value_Forms = webConfig.FindByName("Forms").FindByType(typeof(StringLiteral));
-CxList value_false = webConfig.FindByName("false").FindByType(typeof(StringLiteral));
+CxList value_Forms = webConfig.FindByName("Forms").FindByType<StringLiteral>();
+CxList value_false = webConfig.FindByName("false").FindByType<StringLiteral>();

 CxList requireSSL_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.REQUIRESSL");
 CxList forms_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS");
 CxList forms_childs = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.*");
-CxList configuration = webConfig.FindByName("CONFIGURATION");

 CxList mode_forms = value_Forms.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.MODE"));
 CxList requireSSL = value_false.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.REQUIRESSL"));
@@ -19,7 +18,7 @@
 }
 else
 {
-   if ((mode_forms + requireSSL).Count >= 2)
+   if (All.NewCxList(mode_forms, requireSSL).Count >= 2)
    {
        result = value_false * requireSSL;
    }
```

CSharp / CSharp_WebConfig / SlidingExpiration

Code changes

```
---
+++
@@ -1,11 +1,10 @@
 CxList webConfig = Find_Web_Config();
-CxList value_Forms = webConfig.FindByName("Forms").FindByType(typeof(StringLiteral));
-CxList value_true = webConfig.FindByName("true").FindByType(typeof(StringLiteral));
+CxList value_Forms = webConfig.FindByName("Forms").FindByType<StringLiteral>();
+CxList value_true = webConfig.FindByName("true").FindByType<StringLiteral>();

 CxList slidingExpiration_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.SLIDINGEXPIRATION");
 CxList forms_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS");
 CxList forms_childs = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.*");
-CxList configuration = webConfig.FindByName("CONFIGURATION");

 CxList mode_forms = value_Forms.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.MODE"));
```

```
 CxList slidingExpiration = value_true.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.AUTHENTICATION.FORMS.SLIDINGEXPIRATION"));
@@ -19,7 +18,7 @@
 }
 else
 {
-   if ((mode_forms + slidingExpiration).Count >= 2)
+   if (All.NewCxList(mode_forms, slidingExpiration).Count >= 2)
     {
         result = value_true * slidingExpiration;
     }
```

CSharp / CSharp_WebConfig / TraceEnabled

Code changes

```
---
+++
@@ -1,15 +1,17 @@
 CxList webConfig = Find_Web_Config();
-CxList value_false = webConfig.FindByName("false").FindByType(typeof(StringLiteral));
-CxList value_true = webConfig.FindByName("true").FindByType(typeof(StringLiteral));
+CxList value_false = webConfig.FindByName("false").FindByType<StringLiteral>();
+CxList value_true = webConfig.FindByName("true").FindByType<StringLiteral>();

 CxList enabledTrue = value_true.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.TRACE.ENABLED"));
 CxList localOnlyFalse = value_false.DataInfluencingOn(webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.TRACE.LOCALONLY"));

-if ((enabledTrue + localOnlyFalse).Count > 1)
+CxList enabTrueAndLocalOnlyFalse = All.NewCxList(enabledTrue, localOnlyFalse);
+if (enabTrueAndLocalOnlyFalse.Count > 1)
 {
     result = value_true * enabledTrue;
-   }
-if ((enabledTrue + localOnlyFalse).Count == 1)
+}
+if (enabTrueAndLocalOnlyFalse.Count == 1)
 {
-   result = (value_true * enabledTrue) + (value_false * localOnlyFalse);
+   result = value_true * enabledTrue;
+   result.Add(value_false * localOnlyFalse);
 }
```

CSharp / CSharp_Windows_Phone / Client_Side_Injection

Code changes

```
---
+++
@@ -7,10 +7,10 @@

 // Find URL inputs (for protocol handlers)
 CxList mapUriMethod = All.FindByShortName("MapUri");
```

```
-CxList inputUrls = All.GetParameters(mapUriMethod, 0);
+CxList inputUrls = Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(mapUriMethod, 0);


 // Find external inputs (SMS, protocol handlers)
-CxList inputs = All.FindByMemberAccess("ISmsTextMessage.Body") + inputUrls;
+CxList inputs = All.NewCxList(All.FindByMemberAccess("ISmsTextMessage.Body"), inputUrls);


 CxList sanitize = Find_Sanitize();
```

CSharp / CSharp_Windows_Phone / Failure_to_Implement_Least_Privilege

Code changes

```
---
+++
@@ -17,8 +17,8 @@
 CxList mdeiaAudioCap = idCap.FindByShortName("ID_CAP_MEDIALIB_AUDIO");
 CxList mediaPlayCap = idCap.FindByShortName("ID_CAP_MEDIALIB_PLAYBACK");

-CxList caps = All.NewCxList();
-caps.Add(networkCap,
+CxList caps = All.NewCxList(
+   networkCap,
    locationCap,
    contactsCap,
    storageCap,
@@ -46,7 +46,7 @@
 CxList walletSecureElementPermission = permissions * wSecureCap;
 CxList dialerPermission = permissions * phoneCap;
 CxList photoPermission = permissions * mediaPhotoCap;
-CxList audioPermission = permissions * (mdeiaAudioCap + mediaPlayCap);
+CxList audioPermission = permissions * All.NewCxList(mdeiaAudioCap, mediaPlayCap);


 //find all uses of capabilities
@@ -95,15 +95,10 @@

 CxList usingDialer = All.FindByType("PhoneCallTask");

-CxList usingPhoto = All.FindByMemberAccess("MediaLibrary.Pictures");
-usingPhoto.Add(All.FindByMemberAccess("MediaLibrary.RootPictureAlbum"));
-usingPhoto.Add(All.FindByMemberAccess("MediaLibrary.SavedPictures"));
-usingPhoto.Add(All.FindByMemberAccess("MediaLibrary.GetPictureFromToken"));
-usingPhoto.Add(All.FindByMemberAccess("MediaLibrary.SavePicture*"));
+CxList usingPhoto = All.FindByMemberAccesses("MediaLibrary",
+   new[]{"Pictures", "RootPictureAlbum", "SavedPictures", "GetPictureFromToken", "SavePicture*"});

-CxList usingAudio = All.FindByMemberAccess("MediaLibrary.Albums");
-usingAudio.Add(All.FindByMemberAccess("MediaLibrary.Artists"));
```

```
-usingAudio.Add(All.FindByMemberAccess("MediaLibrary.Genres"));
+CxList usingAudio = All.FindByMemberAccesses("MediaLibrary", new []{"Albums", "Artists", "Genres"});


 // Application Required capability access but not uses it
 if ((networkPermission.Count > 0) && (usingNetwork.Count == 0))
```

CSharp / CSharp_Windows_Phone / Hard_Coded_Cryptography_Key

Code changes

```
---
+++
@@ -7,26 +7,17 @@
 CxList decls = Find_Declarators();
 CxList integers = Find_IntegerLiterals();
 CxList unknown = Find_UnknownReference();
-CxList combinedTypes = methods + unknown;
+CxList combinedTypes = All.NewCxList(methods, unknown);

-CxList hardCoded = strings + integers;
+CxList hardCoded = All.NewCxList(strings, integers);

+CxList keys = All.FindByMemberAccesses(new []{"Rijndael*.Key", "Aes*.Key", "DES*.Key", "HMAC*.Key",
+       "TripleDES*.Key", "ECDiffieHellman*.Key", "AsymmetricKey*.SetKey", "AsymmetricSignature*.SetKey"});

-CxList keys = All.FindByMemberAccess("Rijndael*.Key");
-keys.Add(All.FindByMemberAccess("Aes*.Key"));
-keys.Add(All.FindByMemberAccess("DES*.Key"));
-keys.Add(All.FindByMemberAccess("HMAC*.Key"));
-keys.Add(All.FindByMemberAccess("TripleDES*.Key"));
-keys.Add(All.FindByMemberAccess("ECDiffieHellman*.Key"));
-keys.Add(All.FindByMemberAccess("AsymmetricKey*.SetKey"));
-keys.Add(All.FindByMemberAccess("AsymmetricSignature*.SetKey"));
-
-CxList dpapiEnc = All.FindByMemberAccess("ProtectedData.Protect") +
-                  All.FindByMemberAccess("ProtectedData.Unprotect");
-CxList dpapiKey =  All.GetParameters(dpapiEnc, 1);
+CxList dpapiEnc = All.FindByMemberAccesses(new []{"ProtectedData.Protect", "ProtectedData.Unprotect"});
+CxList dpapiKey =  Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(dpapiEnc, 1);

 keys.Add(dpapiKey);
-

 //search for all references from PasswordDeriveBytes
 CxList typeRefsPassDeriveBytes = Find_TypeRef().FindByShortName("PasswordDeriveBytes");
@@ -47,8 +38,7 @@
 CxList edgeCase = getByAncsMembers.InfluencedBy(hardCoded).GetFirstNodesInPath();

 //remove sanitized results
-CxList notHardcoded = All.NewCxList();
```

```
-notHardcoded.Add(sanitizedHardCoded, initValues, edgeCase);
+CxList notHardcoded = All.NewCxList(sanitizedHardCoded, initValues, edgeCase);
 hardCoded -= notHardcoded;

 result = hardCoded.DataInfluencingOn(keys);
```

CSharp / CSharp_Windows_Phone / Insufficient_Application_Layer_Protect

Code changes

```
---
+++
@@ -15,20 +15,18 @@
 CxList contacts = All.FindByType("Contact");

 CxList pureHttp = Find_Pure_http();
-CxList httpWrite = All.FindByMemberAccess("*Stream.Write").DataInfluencedBy(pureHttp);
-httpWrite.Add(All.FindByMemberAccess("*HttpClient.PostAsync").DataInfluencedBy(pureHttp));
-httpWrite.Add(All.FindByMemberAccess("*HttpClient.PutAsync").DataInfluencedBy(pureHttp));
+CxList httpWrite = All.FindByMemberAccesses(new []{"*Stream.Write", "*HttpClient.PostAsync",
+       "*HttpClient.PutAsync"}).DataInfluencedBy(pureHttp);

-CxList personal = All.NewCxList();
-personal.Add(Find_All_Passwords());
-personal.Add(Find_Personal_Info());
-personal.Add(gpsData);
-personal.Add(contacts);
+CxList personal = All.NewCxList(
+   Find_All_Passwords(),
+   Find_Personal_Info(),
+   gpsData,
+   contacts);

 CxList encrypt = Find_Encrypt();

 result = httpWrite.InfluencedByAndNotSanitized(personal, encrypt);
-

 //handle https with filters

@@ -36,9 +34,7 @@
 CxList client = All.FindByType("HttpClient");
 CxList postAsyncCall = client.GetMembersOfTarget().FindByShortName("PostAsync").GetTargetOfMembers();
 CxList putAsyncCall = client.GetMembersOfTarget().FindByShortName("PutAsync").GetTargetOfMembers();
-CxList calls = All.NewCxList();
-calls.Add(postAsyncCall);
-calls.Add(putAsyncCall);
+CxList calls = All.NewCxList(postAsyncCall, putAsyncCall);
 CxList clientWithIgnoredFilter = calls.DataInfluencedBy(iscAdd);
 clientWithIgnoredFilter = clientWithIgnoredFilter.GetMembersOfTarget();
```

```
 clientWithIgnoredFilter = clientWithIgnoredFilter.DataInfluencedBy(All.FindByShortName("https:*"));
```

CSharp / CSharp_Windows_Phone / Poor_Authorization_and_Authentication

Code changes

```
---
+++
@@ -7,13 +7,14 @@
 CxList deviceExtendedProperties = All.FindByMemberAccess("DeviceExtendedProperties.TryGetValue");
 CxList deviceIdPropertyName = All.FindByShortName("DeviceUniqueId");
 CxList deviceIdFunc = deviceExtendedProperties.FindByParameters(deviceIdPropertyName);
-CxList deviceId = All.GetParameters(deviceIdFunc, 1);
+CxList deviceId = Find_Param().CxSelectDomProperty<Param>(p => p.Value).GetParameters(deviceIdFunc, 1);
 // Another method to get device ID
 CxList deviceId2 = All.FindByMemberAccess("DeviceExtendedProperties.GetValue");
 deviceId.Add(deviceId2);

 // Send DeviceID info over network
 CxList http = All.FindByName(@"*http*");
-CxList outInfluencedByHttp = Find_Write() * Find_Write().DataInfluencedBy(http);
+CxList write = Find_Write();
+CxList outInfluencedByHttp = write * write.DataInfluencedBy(http);

 result = deviceId.DataInfluencingOn(outInfluencedByHttp);
```

CSharp / CSharp_Windows_Phone / Side_Channel_Data_Leakage

Code changes

```
---
+++
@@ -3,7 +3,7 @@
 ////////////////////////////////////////////////////

 CxList logOutputs = Find_Log_Outputs();
-CxList personal = Find_All_Passwords() + Find_Personal_Info();
+CxList personal = All.NewCxList(Find_All_Passwords(), Find_Personal_Info());
 CxList sanitize = Find_Encrypt();

 result = personal.InfluencingOnAndNotSanitized(logOutputs, sanitize);
```

Dart / Dart_Mobile_Low_Visibility / Parameter_Tampering

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query Parameter_Tampering is deprecated");
```

Go / Go_High_Risk / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,12 +1,12 @@
 // Sources
 //  - data retrieval from DB
 //  - reading from a file
-CxList inputs = All.NewCxList(Find_DB_Out(), Find_Read(), Find_DB_Out_GORM());
+CxList inputs = All.NewCxList(Find_DB_Out(), Find_Read());

 // Sanitizers
 CxList sanitizers = All.NewCxList(Find_DB_Sanitize(), Find_DB_Sanitize_GORM());

 // Sinks for Second Order SQL Injection are SQL statements
-CxList outputs = Find_DB_In();
+CxList outputs = All.NewCxList(Find_DB_In(), Find_DB_Out_GORM());

 result = inputs.InfluencingOnAndNotSanitized(outputs, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Go / Go_Low_Visibility / Use_of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,13 +1,15 @@
 CxList psw = Find_Passwords();
 CxList pswInLeftSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList pswInLeftSideDecl = pswInLeftSide.FindByType(typeof(Declarator));
+CxList pswInLeftSideDecl = pswInLeftSide.FindByType<Declarator>();

 CxList strLiterals = Find_Strings();
 strLiterals -= strLiterals.FindByShortName("");

 //when the hardcoded string includes a space or dot we believe
 //it is not a password string
-strLiterals -= strLiterals.FindByShortNames(new List<string>{"* *","*.*","*/*","*\\*"});
+strLiterals -= strLiterals.FindByShortNames("* *","*.*","*/*","*\\*");
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);
+
 CxList litInRightSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);

 // Password in declaration
@@ -29,14 +31,12 @@
 PasswordInDecl -= notHdPass;

 // Find password in an compare operations
-List < string > compareMethodsNames = new List<string>{"Compare", "EqualFold"};
-CxList compareMethods =
-   psw.GetAncOfType(typeof(MethodInvokeExpr))
-   .FindByShortNames(compareMethodsNames);
```

```
+string[] compareMethodsNames = new string[]{"Compare", "EqualFold"};
+CxList compareMethods = psw.GetAncOfType<MethodInvokeExpr>().FindByShortNames(compareMethodsNames);
 compareMethods = strLiterals.GetByAncs(compareMethods);


 // Find password in a '==' operator
-CxList EqualBinaryExpr = psw.GetFathers().FindByType(typeof(BinaryExpr)).
+CxList EqualBinaryExpr = psw.GetFathers().FindByType<BinaryExpr>().
    GetByBinaryOperator(Checkmarx.Dom.BinaryOperator.IdentityEquality);
 CxList EqualOperatorStrings = All.NewCxList();
 foreach(CxList bin in EqualBinaryExpr)
@@ -51,8 +51,7 @@
 }


 // Password in simple assignment
-CxList assignPassword = pswInLeftSide.GetAncOfType(typeof(AssignExpr));
+CxList assignPassword = pswInLeftSide.GetAncOfType<AssignExpr>();
 assignPassword = litInRightSide.GetByAncs(assignPassword);


-result = PasswordInDecl;
-result.Add(compareMethods, assignPassword, EqualOperatorStrings);
+result.Add(PasswordInDecl, compareMethods, assignPassword, EqualOperatorStrings);
```

Go / Go_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -4,20 +4,18 @@
    the input that comes from an outside request
 */
 CxList methods = Find_Methods();
-CxList inputs = All.NewCxList();
-CxList outputs = All.NewCxList();
-CxList sanitizers = All.NewCxList();
 CxList whiteListSanitizers = Find_WhiteListSanitizers();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

-inputs.Add(Find_Interactive_Inputs(), Find_Web_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Web_Inputs());

 CxList dialMethods = methods.FindByMemberAccess("Dialer.Dial", false);
 CxList dialContextMethods = methods.FindByMemberAccess("Dialer.DialContext", false);

-outputs.Add(
+CxList outputs = All.NewCxList(
    Find_Remote_Requests(),
-   All.GetParameters(dialMethods, 1),
-   All.GetParameters(dialContextMethods, 2));
+   paramValue.GetParameters(dialMethods, 1),
```

```
+   paramValue.GetParameters(dialContextMethods, 2));


 outputs -= Find_HTTP_Request_URL_Query_Methods();
 outputs -= Find_Exec_Outputs();
@@ -25,7 +23,7 @@
 CxList generalSanitizers = Find_General_Sanitize();
 CxList dbSanitizers = Find_DB_Sanitize() - whiteListSanitizers;
 CxList xssSanitizers = Find_XSS_Sanitize() - whiteListSanitizers;
-sanitizers.Add(generalSanitizers, dbSanitizers, xssSanitizers);
+CxList sanitizers = All.NewCxList(generalSanitizers, dbSanitizers, xssSanitizers);


 CxList ssrf = outputs.InfluencedByAndNotSanitized(inputs, sanitizers);
-result = ssrf.ReduceFlowByPragma();
+result = ssrf.ReduceFlowByPragma().ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Groovy / Groovy_Best_Coding_Practice / Getter_Method_Could_Be_Property

Code changes

```
---
+++
@@ -1,21 +1,24 @@
 // If a class defines a public method that follows the Java getter notation and
 // that returns a constant, then it is cleaner to provide a Groovy property for
 // the value rather than a Groovy method.
-CxList methods = All.FindByType(typeof(MethodDecl)).FindByShortName("get*");
+CxList methods = Find_MethodDecls().FindByShortName("get*");


 CxList list = All.NewCxList();
-
+CxList stmtCollection = Find_StatementCollection();
+CxList fieldDecl = Find_FieldDecls();
+
 foreach(CxList m in methods)
 {
-   CxList stmtCollect = All.FindByType(typeof(StatementCollection)).FindByFathers(m);
+   CxList stmtCollect = stmtCollection.FindByFathers(m);

-   if (m.GetAncOfType(typeof(ClassDecl)).Count > 0 // the father is a class
+   if (m.GetAncOfType<ClassDecl>().Count > 0 // the father is a class
        && stmtCollect.Count == 1) // single return
    {
        try
        {
-           ClassDecl c = m.GetAncOfType(typeof(ClassDecl)).TryGetCSharpGraph<ClassDecl>();
+           ClassDecl c = m.GetAncOfType<ClassDecl>().TryGetCSharpGraph<ClassDecl>();
            StatementCollection stmtCol = stmtCollect.TryGetCSharpGraph<StatementCollection>();
+           if(stmtCol == null) continue;
            ReturnStmt ret = stmtCol[0] as ReturnStmt;
            if (ret != null)
```

```
            {
@@ -32,7 +35,7 @@
                else if(e.GetType() == typeof(UnknownReference))
                {
                    UnknownReference id = e as UnknownReference;
-                   CxList vars = All.FindByType(typeof(FieldDecl)).
+                   CxList vars = fieldDecl.
                        FindByFieldAttributes(Modifiers.Static).
                        FindByName(c.Name + "." + id.VariableName);
                    if (vars.Count == 1)
```

Groovy / Groovy_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,15 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList dbOut = Find_DB_Out();
-   CxList dbIn = Find_SQL_DB_In();
-   CxList sanitize = Find_SQL_Sanitize();
-
-   result = All.FindSQLInjections(Find_Read() + possible_db, possible_db - Find_DAL_DB(), sanitize);
-
-   if (result.Count > 0)
-   {
-       result -= All.FindSQLInjections(Find_Read() + dbOut, dbIn, sanitize);
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_Heuristic / Heuristic_CGI_Stored_XSS

Code changes

```
---
+++
@@ -1,15 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList outputs = Find_Console_Outputs();
-   CxList sanitize = Find_XSS_Sanitize();
-
-   result = All.FindXSS(possible_db + Find_Read(), outputs, sanitize - Find_Read());
-
-   if (result.Count > 0)
```

```
-   {
-       CxList db = Find_DB_Out();
-       result -= All.FindXSS(db + Find_Read(), outputs, sanitize - Find_Read());
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_Heuristic / Heuristic_CSRF

Code changes

```
---
+++
@@ -1,21 +1 @@
-if (All.isWebApplication)
-{
-   CxList possible_db = Find_DB_Heuristic();
-
-   if (possible_db.Count > 0)
-   {
-       CxList requests = Find_Interactive_Inputs();
-       requests.Add(All.FindByName("*Request.QueryString*"));
-       CxList strings = Find_Strings();
-       CxList write = strings.FindByNames(new string [] {"*update*", "*delete*", "*insert*"}
-           , StringComparison.OrdinalIgnoreCase);
-
-       result = possible_db.DataInfluencedBy(write).DataInfluencedBy(requests);
-
-       if (result.Count > 0)
-       {
-           CxList db = Find_DB_In();
-           result -= db.DataInfluencedBy(write).DataInfluencedBy(requests);
-       }
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_Heuristic / Heuristic_DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,32 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList tables = All.FindByName("*orders*", false) +
-       All.FindByName("*credit*", false) +
-       All.FindByName("*invoice*", false) +
-       All.FindByName("*booking*", false) +
```

```
-       All.FindByName("*bill*", false) +
-       All.FindByName("*payment*", false) +
-       All.FindByName("*account*", false) +
-       All.FindByName("*cash*", false) +
-       All.FindByName("*customer*", false);
-
-   CxList inputs = Find_Interactive_Inputs();
-
-   CxList user = All.FindByName("*user*", false) +
-       All.FindByName("*cust*", false) +
-       All.FindByName("*member*", false);
-
-   possible_db = possible_db.DataInfluencedBy(tables);
-   possible_db -= possible_db.DataInfluencedBy(user);
-   result = inputs.DataInfluencingOn(possible_db);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB_In();
-       db = db.DataInfluencedBy(tables);
-       db -= db.DataInfluencedBy(user);
-       result -= inputs.DataInfluencingOn(db);
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---
+++
@@ -1,25 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList input = Find_Interactive_Inputs();
-
-   CxList strings = Find_Strings();
-   CxList Select = strings.FindByName("*select*", false);
-   CxList Where = strings.FindByName("*where*", false);
-   CxList And = strings.FindByName("*And *", false) +
-       strings.FindByName("* And*", false);
-
-   possible_db = possible_db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-   possible_db -= possible_db.DataInfluencedBy(And);
-
-   result = possible_db.DataInfluencedBy(input);
-
```

```
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB_In();
-       db = db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-       db -= db.DataInfluencedBy(And);
-       result -= db.DataInfluencedBy(input);
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_Heuristic / Heuristic_SQL_Injection

Code changes

```
---
+++
@@ -1,15 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList inputs = Find_Interactive_Inputs();
-   CxList sanitized = Find_SQL_Sanitize();
-
-   result = inputs.InfluencingOnAndNotSanitized(possible_db - Find_DAL_DB(), sanitized);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_SQL_DB_In();
-       result -= inputs.InfluencingOnAndNotSanitized(db, sanitized);
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_Heuristic / Heuristic_Stored_XSS

Code changes

```
---
+++
@@ -1,19 +1 @@
-if(All.isWebApplication)
-{
-   CxList possible_db = Find_DB_Heuristic();
-
-   if (possible_db.Count > 0)
-   {
-       CxList outputs = Find_XSS_Outputs();
-       CxList sanitize = Find_XSS_Sanitize();
-
-       result = (possible_db + Find_IO()).InfluencingOnAndNotSanitized(outputs, sanitize)
-           .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

```
-
-       if (result.Count > 0)
-       {
-           CxList db = Find_DB_Out();
-           result -= (db + Find_IO()).InfluencingOnAndNotSanitized(outputs, sanitize);
-       }
-   }
-}
+//This query is deprecated.
```

Groovy / Groovy_High_Risk / UTF7_XSS

Code changes

```
---
+++
@@ -1,14 +1 @@
-CxList UTF7 = Find_Strings().FindByName("UTF-7");
-CxList response =
-   All.FindByName("*Response.setCharacterEncoding") +
-   All.FindByName("*response.setCharacterEncoding") +
-   All.FindByMemberAccess("HttpServletResponse.setCharacterEncoding");
-
-UTF7 = response.DataInfluencedBy(UTF7);
-
-if (UTF7.Count > 0)
-{
-   CxList outputs = Find_XSS_Outputs();
-   CxList inputs = Find_Interactive_Inputs();
-   result = outputs.DataInfluencedBy(inputs);
-}
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList db = Find_SQL_DB_In();
-CxList db_not_in_try = Improper_Exception_Handling(db);
-CxList db_in_try = db - db_not_in_try;
-
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitized = Find_SQL_Sanitize();
-
-result = inputs.InfluencingOnAndNotSanitized(db_in_try, sanitized);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Channel_Accessible_by_NonEndpoint

Code changes

```
---
+++
@@ -1,23 +1 @@
-// Find all outputs that have "print*" or "write*"
-CxList outputs = Find_Interactive_Outputs();
-outputs =
-   outputs.FindByShortName("print*") +
-   outputs.FindByShortName("write*");
-
-/*
-Bypassing:
-// When the output is response, it will be checked response.write will be checked if it is not secured
-CxList response = All.FindByType("HttpServletResponse").FindByType(typeof(ParamDecl));
-CxList isSecure = Find_Conditions().FindByMemberAccess("HttpServletRequest.isSecure");
-CxList secureIf = isSecure.GetFathers();
-CxList outputsResponse = outputs - outputs.GetByAncs(secureIf);
-// Find insecured response, influenced by inputs
-outputsResponse = response.DataInfluencingOn(outputsResponse).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
-*/
-
-// When the output is Socket, it is not secured (not SSLSocket)
-CxList Socket = All.FindByMemberAccess("Socket.getOut*").GetTargetOfMembers();
-// Find Socket influenced by inputs
-CxList outputsSocket = Socket.DataInfluencingOn(outputs);
-
-result = outputsSocket.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);// +outputsResponse;
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Cleansing_Canonicalization_and_Comparison_Errors

Code changes

```
---
+++
@@ -1,14 +1 @@
-CxList inputs = All.FindByMemberAccess("*HttpServletRequest.getRequestURI") +
-   All.FindByMemberAccess("*HttpServletRequest.getRequestURL") +
-   All.FindByMemberAccess("*HttpServletRequest.getServletPath");
-
-CxList sanitize =
-   All.FindByMemberAccess("URLDecoder.decode") +
-   All.FindByMemberAccess("Encoder.decodeForURL") +
-   Get_ESAPI().FindByMemberAccess("Encoder.canonicalize");
-
-CxList binaryExpr = All.FindByType(typeof(BinaryExpr));
-
-CxList bin = binaryExpr.InfluencedByAndNotSanitized(inputs, sanitize);
-
-result = bin.ControlInfluencingOn(All);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Improper_Resource_Shutdown_or_Release

Code changes

```
---
+++
@@ -1,13 +1,14 @@
 CxList methods = Find_Methods();
 CxList returns = Find_ReturnStmt();
-CxList cattr = All.FindByType(typeof(CustomAttribute));
+CxList cattr = Find_CustomAttribute();
 CxList cathrows = cattr.FindByShortName("CxThrows");
-CxList myMethods = cathrows.GetAncOfType(typeof(MethodDecl));
+CxList myMethods = cathrows.GetAncOfType<MethodDecl>();
 CxList objects = Find_Object_Create();
-CxList allRefsInProj = All.FindByType(typeof(TypeRef));
-CxList allConstructorDeclInProj = All.FindByType(typeof(ConstructorDecl));
-CxList allStatementCollection = All.FindByType(typeof(StatementCollection));
-CxList Trys = All.FindByType(typeof(TryCatchFinallyStmt));
+CxList allRefsInProj = Find_TypeRef();
+CxList allConstructorDeclInProj = Find_ConstructorDecl();
+CxList allStatementCollection = Find_StatementCollection();
+CxList Trys = Find_TryCatchFinallyStmt();
+CxList castExpr = Find_CastExpr();
```

```
 List<string> relevantTypes = new List<string> {
@@ -236,8 +237,7 @@
         "ZipOutputStream"};

 // Get all classes inherits
-CxList ClassInheritsFrom = All.NewCxList();
-CxList ClassDeclList = All.FindByType(typeof(ClassDecl));
+CxList ClassDeclList = Find_ClassDecl();

 List<string> inheriting = new List<string>();
 foreach( string c in relevantTypes){
@@ -250,58 +250,51 @@

 CxList allResourcesInProj = (objects.FindByShortNames(relevantTypes));

-CxList wrappingObjects = All.FindAllReferences(All.GetClass(allResourcesInProj.GetAncOfType(typeof(ConstructorDecl))))
-    .GetFathers().FindByType(typeof(ObjectCreateExpr));
+CxList wrappingObjects = All.FindAllReferences(All.GetClass(allResourcesInProj.GetAncOfType<ConstructorDecl>()))
+    .GetFathers().FindByType<ObjectCreateExpr>();

 //methods where is opened resources
-CxList auxOpenedMethods = All.FindAllReferences(myMethods.GetMethod(allResourcesInProj.FindByFathers(returns))).FindByType(typeof(MethodInvokeExpr));
+CxList auxOpenedMethods = All.FindAllReferences(myMethods.GetMethod(allResourcesInProj.FindByFathers(returns))).FindByType<MethodInvokeExpr>();
 allResourcesInProj.Add(auxOpenedMethods);
```

```
 allResourcesInProj.Add(allResourcesInProj.GetByAncs(myMethods.GetMethod(allResourcesInProj.FindByFathers(returns))));


-CxList allOpen = (methods.FindByShortNames(new List<string> {
-       "getConnection",
-       "open",
-       "getOutputStream",
-       "getInputStream",
-       "getWriter",
-       "getReader",
-       "getResourceAsStream",
-       "openOutputStream",
-       "openFileOutput",
-       "makeConnection",
-       "getErrorStream",
-       "streamContent",
-       "createInputStream"
-       }));
-
-CxList unAccountable = allOpen.FindByMemberAccess("HttpServletRequest.*");
-unAccountable.Add(allOpen.FindByMemberAccess("HttpServletResponse.*"));
+CxList allOpen = methods.FindByShortNames(
+   "getConnection",
+   "open",
+   "getOutputStream",
+   "getInputStream",
+   "getWriter",
+   "getReader",
+   "getResourceAsStream",
+   "openOutputStream",
+   "openFileOutput",
+   "makeConnection",
+   "getErrorStream",
+   "streamContent",
+   "createInputStream");
+
+CxList unAccountable = allOpen.FindByMemberAccesses(new string [] {
+   "HttpServletRequest.*", "HttpServletResponse.*"});
 allOpen -= unAccountable;


-allResourcesInProj.Add(allOpen);
-
-CxList mimeUtility = methods.FindByMemberAccess("MimeUtility.encode");
-mimeUtility.Add(methods.FindByMemberAccess("MimeUtility.decode"));
-allResourcesInProj.Add(mimeUtility);
-
-CxList sesionConnection = methods.FindByMemberAccess("Session.connection");
-allResourcesInProj.Add(sesionConnection);
-
```

```
-CxList urlStream = methods.FindByMemberAccess("URL.openStream");
-allResourcesInProj.Add(urlStream);
-CxList transtaction = methods.FindByMemberAccess("Transaction.begin");
-allResourcesInProj.Add(transtaction);
+allResourcesInProj.Add(allOpen,
+   methods.FindByMemberAccesses(new string [] {
+   "MimeUtility.encode", "MimeUtility.decode",
+   "Session.connection",
+   "URL.openStream",
+   "Transaction.begin"
+   }));

 CxList returningConnection = All.FindAllReferences(allOpen.GetAssignee()).FindByFathers(returns);
 returningConnection.Add(allOpen.FindByFathers(returns));

-allResourcesInProj.Add(All.FindAllReferences(myMethods.GetMethod(returningConnection)).FindByType(typeof(MethodInvokeExpr)));
+allResourcesInProj.Add(All.FindAllReferences(myMethods.GetMethod(returningConnection)).FindByType<MethodInvokeExpr>());
 // avoid encapsulating objects (considering the resource is closed by the wrapping object)
 allResourcesInProj -= allResourcesInProj.GetParameters(allResourcesInProj.FindByParameters(allResourcesInProj));



 CxList ThrowingReferences = All.FindAllReferences(allResourcesInProj.GetAssignee());
-ThrowingReferences.Add(All.FindAllReferences(All.FindByType(typeof(TypeRef)).FindByFathers(All.FindByType(typeof(CastExpr)))
+ThrowingReferences.Add(All.FindAllReferences(allRefsInProj.FindByFathers(castExpr)
    .FindByTypes(relevantTypes.ToArray()).GetFathers().GetAssignee()));

 ThrowingReferences.Add(ThrowingReferences.GetMembersOfTarget());
@@ -331,35 +324,34 @@

 allClose -= Find_Dead_Code();
 //search for all invocations of methods which close the resource
-CxList auxClosedMethods = All.FindAllReferences(myMethods.GetMethod(allClose)).FindByType(typeof(MethodInvokeExpr));
+CxList auxClosedMethods = All.FindAllReferences(myMethods.GetMethod(allClose)).FindByType<MethodInvokeExpr>();

 CxList closes = TryEnds * allClose;
 //methods close transaction from Apache Transaction
-closes.Add(methods.FindByMemberAccess("Transaction.safeRollback"));
-closes.Add(methods.FindByMemberAccess("Transaction.rollback"));
-closes.Add(methods.FindByMemberAccess("Transaction.commit"));
+closes.Add(methods.FindByMemberAccesses(new string [] {
+   "Transaction.safeRollback", "Transaction.rollback", "Transaction.commit"}));

 CxList closedResources = All.NewCxList();
 foreach(CxList myTry in TryBlock){
-   CxList curFinally = allStatementCollection.GetFinallyClause(myTry.GetAncOfType(typeof(TryCatchFinallyStmt)));
+   CxList curFinally = allStatementCollection.GetFinallyClause(myTry.GetAncOfType<TryCatchFinallyStmt>());
    try{
        //we test if we have a try statement
```

```
        if(curFinally.TryGetCSharpGraph<StatementCollection>().Count > 0){
            // remove resource that close on the finally block
            CxList myList = All.FindAllReferences(closes.GetByAncs(curFinally).GetTargetOfMembers()).GetAssigner();
-           myList.Add(All.FindByFathers(myList.FindByType(typeof(CastExpr))));
+           myList.Add(All.FindByFathers(myList.FindByType<CastExpr>()));
            closedResources.Add(myList.GetByAncs(myTry));

            // remove resources that are wrapped and are closed along recurring to auxiliar method
-           CxList openedResources = All.FindByFathers(All.FindAllReferences(auxClosedMethods.GetByAncs(curFinally).GetTargetOfMembers()).GetByAncs(TryBlocks).GetAncOfType(typeof(AssignExpr))).GetAssignee();
-           CxList resourcesWrapperClass = All.FindDefinition(allRefsInProj.FindByFathers(openedResources.GetAssigner().FindByType(typeof(ObjectCreateExpr))));
+           CxList openedResources = All.FindByFathers(All.FindAllReferences(auxClosedMethods.GetByAncs(curFinally).GetTargetOfMembers()).GetByAncs(TryBlocks).GetAncOfType<AssignExpr>()).GetAssignee();
+           CxList resourcesWrapperClass = All.FindDefinition(allRefsInProj.FindByFathers(openedResources.GetAssigner().FindByType<ObjectCreateExpr>()));
            CxList ResourcesAllocatedOnWrapperConstructor = allResourcesInProj.GetByAncs(allConstructorDeclInProj.GetByAncs(resourcesWrapperClass));
            closedResources.Add(ResourcesAllocatedOnWrapperConstructor);
            CxList resource_closer = All.GetParameters(auxClosedMethods).GetByAncs(curFinally);
            closedResources.Add(All.FindAllReferences(resource_closer).FindByAssignmentSide(CxList.AssignmentSide.Left).GetAssigner());

            // remove resources that are wrapped, opened through a method and are close along recurring to auxiliar method
-           CxList methodsReturningResources = All.FindDefinition(openedResources.GetAssigner().FindByType(typeof(MethodInvokeExpr)));
+           CxList methodsReturningResources = All.FindDefinition(openedResources.GetAssigner().FindByType<MethodInvokeExpr>());
            closedResources.Add(allResourcesInProj.GetByAncs(allResourcesInProj.GetByAncs(methodsReturningResources)));

        }
@@ -370,18 +362,18 @@
 /*
 remove all resources that are closed at some "finally" even if not located under try.
 */
-CxList allFinallies = allStatementCollection.GetFinallyClause(All.FindByType(typeof(TryCatchFinallyStmt)));
+CxList allFinallies = allStatementCollection.GetFinallyClause(Trys);
 CxList closable = methods.FindByShortName("close").GetByAncs(allFinallies);
 closedResources.Add(closable);
 CxList resourceReference = closable.GetTargetOfMembers();
 closedResources.Add(All.FindAllReferences(resourceReference));

-CxList TryBlockMethods = TryBlock.FindByType(typeof(MethodInvokeExpr)) * auxOpenedMethods;
+CxList TryBlockMethods = TryBlock.FindByType<MethodInvokeExpr>() * auxOpenedMethods;

 CxList mDefinitions = All.FindDefinition(TryBlockMethods);
 //list with the resource open in auxiliar methods
 //this methods are invoke inside a try block (mDefinitions)
-CxList auxResourcesOpen = All.GetByAncs(mDefinitions).FindByType(typeof(ObjectCreateExpr));
+CxList auxResourcesOpen = All.GetByAncs(mDefinitions).FindByType<ObjectCreateExpr>();
 auxResourcesOpen.Add(closedResources);

 CxList tempResult = allResourcesInProj - auxResourcesOpen;
@@ -403,7 +395,7 @@
 }
```

```
 CxList paramOfBuffered = All.GetParameters(All.FindByShortNames(autocloseable));
-CxList closedByBuffered = tempResult.InfluencingOn(paramOfBuffered) + paramOfBuffered;
+CxList closedByBuffered = All.NewCxList(tempResult.InfluencingOn(paramOfBuffered), paramOfBuffered);
 tempResult -= closedByBuffered;


 CxList addResult = All.NewCxList();
```

Groovy / Groovy_Low_Visibility / Improper_Transaction_Handling

Code changes

```
---
+++
@@ -1,36 +1,31 @@
 CxList Commit = All.FindByName("*.commit");
 CxList Rollback = All.FindByName("*.rollback");


-CxList TryBlock = Commit.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList TryBlock = Commit.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList cml in TryBlock)
 {
    TryCatchFinallyStmt TryGraph = cml.TryGetCSharpGraph<TryCatchFinallyStmt>();


    CxList curTry = All.FindById(TryGraph.Try.NodeId);
-
    CxList curCatch = All.NewCxList();
    if(TryGraph.CatchClauses != null && TryGraph.CatchClauses.Count > 0)
-   {
        curCatch = All.FindById(TryGraph.CatchClauses[0].NodeId);
-   }


    CxList curFinally = All.NewCxList();
-   if(TryGraph.CatchClauses != null && TryGraph.Finally.Count > 0)
-   {
+   if(TryGraph.CatchClauses != null && TryGraph.Finally != null && TryGraph.Finally.Count > 0)
        curFinally = All.FindById(TryGraph.Finally.NodeId);
-   }


    CxList CommitInTry = Commit.GetByAncs(curTry);


    CxList RollbackInCatch = Rollback.GetByAncs(curCatch);
    CxList RollbackInFinally = Rollback.GetByAncs(curFinally);
-   CxList goodTryStmt =
-       RollbackInCatch.GetAncOfType(typeof(TryCatchFinallyStmt)) +
-       RollbackInFinally.GetAncOfType(typeof(TryCatchFinallyStmt));
-   goodTryStmt = goodTryStmt.GetFathers().GetAncOfType(typeof(TryCatchFinallyStmt));
+   CxList goodTryStmt = All.NewCxList(
+       RollbackInCatch.GetAncOfType<TryCatchFinallyStmt>(),
+       RollbackInFinally.GetAncOfType<TryCatchFinallyStmt>());
```

```
+   goodTryStmt = goodTryStmt.GetFathers().GetAncOfType<TryCatchFinallyStmt>();


-   CxList testedTry = CommitInTry.GetAncOfType(typeof(TryCatchFinallyStmt));
+   CxList testedTry = CommitInTry.GetAncOfType<TryCatchFinallyStmt>();


    if((goodTryStmt * testedTry).Count == 0)
    {
```

Groovy / Groovy_Low_Visibility / Leaving_Temporary_File

Code changes

```
---
+++
@@ -7,21 +7,22 @@
    if(curTmpFile.DataInfluencingOn(delete).Count == 0)
    { // if no delete found
        // find the parameter that holds the filename, and where it has delete
-       CxList fileParamName = All.GetByAncs(curTmpFile.GetAncOfType(typeof(AssignExpr))).FindByAssignmentSide(CxList.AssignmentSide.Left);
+       CxList fileParamName = All.GetByAncs(curTmpFile.GetAncOfType<AssignExpr>())
+           .FindByAssignmentSide(CxList.AssignmentSide.Left);
        fileParamName = All.FindByName(fileParamName);
        fileParamName = delete.GetTargetOfMembers() * fileParamName;


        // find the relevant finally block.
-       CxList Try = curTmpFile.GetAncOfType(typeof(TryCatchFinallyStmt));
+       CxList Try = curTmpFile.GetAncOfType<TryCatchFinallyStmt>();
        if (Try != null && Try.data.Count > 0)
        {
            TryCatchFinallyStmt TryGraph = Try.TryGetCSharpGraph<TryCatchFinallyStmt>();
+           if(TryGraph == null) continue;
+
            CxList curFinally = All.FindById(TryGraph.Finally.NodeId);
            // See if there is no delete in the relevant block
            if (fileParamName.GetByAncs(curFinally).Count == 0)
-           {
                result.data.AddRange(curTmpFile.data);
-           }
        }
        else // If no try block at all - add to result
        {
```

Groovy / Groovy_Low_Visibility / Plaintext_Storage_in_a_Cookie

Code changes

```
---
+++
@@ -1,13 +1 @@
-CxList addCookie =
-   All.FindByMemberAccess("HttpServletRequest.addCookie") +
-   All.FindByName("*response.addCookie") +
```

```
-   All.FindByName("*Response.addCookie") +
-   All.FindByMemberAccess("HTTPUtilities.safeAddCookie"); // ESAPI
-CxList cookies = All.FindByType("Cookie");
-CxList cookieParam = All.GetParameters(cookies, 1).FindByType(typeof(UnknownReference));
-
-CxList plainText = Find_Strings();
-
-CxList sanitize = Find_General_Sanitize();
-
-result = plainText.InfluencingOnAndNotSanitized(cookieParam, sanitize);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Potenial_UTF7_XSS

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList decode = All.FindByName("*.htmlDecode") +
-   All.FindByMemberAccess("HtmlDecoder.decode") +
-   All.FindByName("*HtmlDecoder.decode") +
-   All.FindByMemberAccess("ServletResponse.decode*");
-CxList sanitize = Find_XSS_Sanitize() + Find_DB_In();
-CxList output = Find_Interactive_Outputs();
-
-result = output.InfluencedByAndNotSanitized(decode, sanitize);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Potential_ReDoS

Code changes

```
---
+++
@@ -1,5 +1 @@
-CxList filter = Potential_ReDoS_In_Match() +
-   Potential_ReDoS_In_Replace() +
-   Potential_ReDoS_In_Static_Field();
-
-result = Find_Evil_Strings() - filter - Find_Evil_Strings().DataInfluencingOn(filter);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Potential_ReDoS_By_Injection

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_ReDoS(Find_Inputs(), Find_Match() + Find_Replace_Regex(), true);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Potential_ReDoS_In_Match

Code changes

```
---

+++

@@ -1 +1 @@

-result = Find_ReDoS(Find_Evil_Strings(), Find_Match(), true);

+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Potential_ReDoS_In_Replace

Code changes

```
---

+++

@@ -1 +1 @@

-result = Find_ReDoS(Find_Evil_Strings_For_Replace(), Find_Replace_Regex(), true);

+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Potential_ReDoS_In_Static_Field

Code changes

```
---

+++

@@ -1,14 +1 @@

-// Find all evil strings

-CxList evilStrings = Find_Evil_Strings();

-

-// Sanitization

-CxList sanitize = Find_Integers();

-

-// Find all regex commands

-CxList regex = All.FindByMemberAccess("Pattern.compile");

-

-// Add static regexes (these do not influence their references, so needed here)

-CxList staticFields = All.FindByFieldAttributes(Modifiers.Static);

-staticFields = staticFields.FindByType(typeof(ConstantDecl)) + staticFields.FindByType(typeof(FieldDecl));

-

-result = evilStrings.InfluencingOnAndNotSanitized(regex.GetByAncs(staticFields), sanitize);

+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Reliance_on_Cookies_in_a_Decision

Code changes

```
---

+++

@@ -1,6 +1 @@

-CxList cookies =

-   All.FindByMemberAccess("request.getCookies");

-

-CxList cond = Find_Conditions();

-
```

```
-result = cond.DataInfluencedBy(cookies);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Use_of_Client_Side_Authentication

Code changes

```
---
+++
@@ -1,10 +1 @@
-CxList htmlOutput = Find_Web_Outputs() + Find_Html_Outputs();
-
-CxList strings = Find_Strings();
-CxList htmlRemarks = strings.FindByName("*<script*") + strings.FindByName("*</script>*");
-
-htmlOutput = htmlOutput.DataInfluencedBy(htmlRemarks);
-
-CxList pass = Find_Password_Strings();
-
-result = htmlOutput.DataInfluencedBy(pass);
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Use_Of_getenv

Code changes

```
---
+++
@@ -1,3 +1 @@
-CxList method = Find_Methods();
-
-result = method.FindByShortName("getenv");
+//This query is deprecated.
```

Groovy / Groovy_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,37 +1,13 @@
-CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null");
 CxList psw = Find_Passwords();
+CxList useOfHardcodedPasswords = Common_Low_Visibility.Use_Of_Hardcoded_Password();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
+CxList methods = Find_Methods();

-// Find password in an initialization operation
-CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList psw_in_lSide_decl = psw_in_lSide.FindByType(typeof(Declarator));
-
-CxList strLiterals = Find_Strings() - emptyString - NULL;
-
```

```
-//when the hardcoded string includes a space or dot we believe
-//it is not a password string
-strLiterals -= strLiterals.FindByName("* *");
-strLiterals -= strLiterals.FindByName("*.*");
-strLiterals -= strLiterals.FindByName("*/*");
-strLiterals -= strLiterals.FindByName("*\\*");
-
-CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
-CxList initializedPassword = psw_in_lSide_decl.FindByInitialization(lit_in_rSide);
-
-//remove passwords with equal name and contant ==> currPassword = "currPassword";
-CxList notHdPass = All.NewCxList();
-char[] trimChars = new char[2] { '\'', '"'};
-foreach(CxList currPass in initializedPassword)
-{
-   CxList currStrInLeft = lit_in_rSide.FindInitialization(currPass);
-   string strName = currStrInLeft.GetName().Trim(trimChars);
-   string passName = currPass.GetName();
-   if (passName.Equals(strName))
-   {
-       notHdPass.Add(currPass);
-   }
-}
-initializedPassword -= notHdPass;
+CxList strLiterals = Find_Strings();
+strLiterals -= Find_Empty_Strings();
+strLiterals -= All.FindByName("null");
+strLiterals -= strLiterals.FindByNames("* *", "*.*", "*/*", "*\\*");
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);


 // Find password in an "equals" operation
 CxList eq = All.FindByMemberAccess("String.equals");
@@ -45,63 +21,53 @@
 strEQ = psw.GetByAncs(strEQ);
 equalsPassword.Add(strEQ);


-// Find password in assignments
-CxList assignPassword = psw_in_lSide.GetAncOfType(typeof(AssignExpr));
-assignPassword = lit_in_rSide.GetByAncs(assignPassword);
-
-CxList methods = Find_Methods();
 CxList connection = methods.FindByShortName("getConnection");
 CxList sqlNewInstance = methods.FindByName("*Sql.newInstance");
-CxList connetionParam2 = All.GetParameters(connection, 2) + All.GetParameters(sqlNewInstance, 2);
+CxList connetionParam2 = All.NewCxList(
+   paramValue.GetParameters(connection, 2),
+   paramValue.GetParameters(sqlNewInstance, 2));
```

```
-CxList connetionParam0 = All.GetParameters(connection, 0);
-CxList pwdInFirstConnectionParam = connetionParam0.FindByType(typeof(StringLiteral)).FindByName("*PWD=*") +
-   connetionParam0.FindByType(typeof(StringLiteral)).FindByName("*PWD =*");
-
-
+CxList connetionParam0 = paramValue.GetParameters(connection, 0);
+CxList pwdInFirstConnectionParam = All.NewCxList(
+   connetionParam0.FindByType<StringLiteral>().FindByNames("*PWD=*", "*PWD =*"));

 // Add also KerberosKey's second parameter as a potentially vulnerable hardcoded parameter
-CxList KerberosKey = All.FindByType("KerberosKey");
-KerberosKey =
-   KerberosKey.FindByType(typeof(ObjectCreateExpr)) +
-   KerberosKey.FindByType(typeof(Declarator));
+CxList KerberosKey = All.FindByType("KerberosKey").FindByTypes(typeof(ObjectCreateExpr), typeof(Declarator));

 // Get second parameter
-CxList KerberosKeyParam1 = All.GetParameters(KerberosKey, 1);
+CxList KerberosKeyParam1 = paramValue.GetParameters(KerberosKey, 1);

 // Add also KerberosPrincipal's second parameter as a potentially vulnerable hardcoded parameter
 CxList PasswordAuthentication = All.FindByType("PasswordAuthentication");
-PasswordAuthentication =
-   PasswordAuthentication.FindByType(typeof(ObjectCreateExpr)) +
-   PasswordAuthentication.FindByType(typeof(Declarator));
+PasswordAuthentication.Add(PasswordAuthentication.FindByTypes(typeof(ObjectCreateExpr), typeof(Declarator)));

 // Get second parameter
-CxList PasswordAuthenticationParam1 = All.GetParameters(PasswordAuthentication, 1);
+CxList PasswordAuthenticationParam1 = paramValue.GetParameters(PasswordAuthentication, 1);

-CxList relevantParams = KerberosKeyParam1 + connetionParam2 + PasswordAuthenticationParam1;
+CxList relevantParams = All.NewCxList(KerberosKeyParam1, connetionParam2, PasswordAuthenticationParam1);

 // Sanitize by binaries such as "+" and by concatenate - could be concatenated with a non hard-coded key,
 // which is OK
-CxList bin = All.FindByType(typeof(BinaryExpr));
-bin = bin.FindByShortName("");
+CxList bin = Find_BinaryExpr().FindByShortName("");
 CxList concat = All.FindByShortName("concatenate", false);

-CxList sanitize = bin + concat;
+CxList sanitize = All.NewCxList(bin, concat);
+
 CxList undefinedMethods = methods - methods.FindAllReferences(All.FindDefinition(methods));
 sanitize.Add(undefinedMethods);

 // Add the parameter itself, or whatever is influencing it
```

```
-CxList paramsAffectedByString = relevantParams * strLiterals +
-   relevantParams.InfluencedByAndNotSanitized(strLiterals, sanitize);
+CxList paramsAffectedByString = All.NewCxList(
+   relevantParams * strLiterals,
+   relevantParams.InfluencedByAndNotSanitized(strLiterals, sanitize));

-CxList setPasswordMethod = Find_Methods().FindByShortName("setPassword", false);
-CxList passwordParams = All.GetParameters(setPasswordMethod).FindByType(typeof(StringLiteral));
+CxList setPasswordMethod = methods.FindByShortName("setPassword", false);
+CxList passwordParams = strLiterals.GetParameters(setPasswordMethod);
 CxList hardcodedPasswordInMethod = setPasswordMethod.DataInfluencedBy(passwordParams);

 // All
-result = initializedPassword +
-   equalsPassword +
-   assignPassword +
-   paramsAffectedByString +
-   hardcodedPasswordInMethod +
-   pwdInFirstConnectionParam;
+result.Add(useOfHardcodedPasswords,
+       equalsPassword,
+       pwdInFirstConnectionParam,
+       paramsAffectedByString,
+       hardcodedPasswordInMethod);
```

Groovy / Groovy_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,22 +1 @@
-CxList tables = All.FindByShortName("*orders*", false);
-tables.Add(All.FindByShortName("*credit*", false));
-tables.Add(All.FindByShortName("*invoice*", false));
-tables.Add(All.FindByShortName("*booking*", false));
-tables.Add(All.FindByShortName("*bill*", false));
-tables.Add(All.FindByShortName("*payment*", false));
-tables.Add(All.FindByShortName("*account*", false));
-tables.Add(All.FindByShortName("*cash*", false));
-tables.Add(All.FindByShortName("*customer*", false));
-
-CxList inputs = Find_Interactive_Inputs();
-CxList db = Find_DB_In();
-
-CxList user = All.FindByShortName("*user*", false);
-user.Add(All.FindByShortName("*cust*", false));
-user.Add(All.FindByShortName("*member*", false));
-
-db = db.DataInfluencedBy(tables);
```

```
-db -= db.DataInfluencedBy(user);
-CxList sanitize = Find_Parameter_Tampering_Sanitize();
-
-result = inputs.InfluencingOnAndNotSanitized(db, sanitize);
+//This query is deprecated.
```

Groovy / Groovy_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,9 +1 @@
-CxList header_inputs = Find_Headers();
-
-CxList sanitize = Find_Splitting_Sanitizer();
-
-CxList inputs = Find_Interactive_Inputs() - header_inputs;
-
-CxList outputs = Find_Header_Outputs();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

Groovy / Groovy_Medium_Threat / Multiple_Binds_to_the_Same_Port

Code changes

```
---
+++
@@ -1,27 +1 @@
-CxList bind = All.FindByMemberAccess("ServerSocket.bind");
-CxList socketAddress = All.FindByType("InetSocketAddress");
-CxList ports = All.GetParameters(socketAddress).FindByType(typeof(IntegerLiteral));
-CxList portsMethods = All.GetMethod(ports);
-
-foreach (CxList method in portsMethods)
-{
-   CxList methodPorts = ports.GetByAncs(method);
-   SortedList portNumbers = new SortedList();
-   foreach(CxList port in methodPorts)
-   {
-       IntegerLiteral g = port.TryGetCSharpGraph<IntegerLiteral>();
-       int portNumber = -1;
-       if (int.TryParse(g.ShortName, out portNumber))
-       {
-           if (portNumbers.ContainsValue(portNumber))
-           {
-               result.Add(g.NodeId, g);
-               break;
-           }
-           else
```

```
-          {
-              portNumbers[portNumber] = portNumber;
-          }
-      }
-  }
-}
+//This query is deprecated.
```

Groovy / Groovy_Medium_Threat / Unnormalize_Input_String

Code changes

```
---
+++
@@ -1 +1 @@
-//Deprecated query: see Input_Path_Not_Canonicalized
+//This query is deprecated.
```

Groovy / Groovy_Stored / Stored_HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,9 +1 @@
-CxList header_inputs = Find_Headers();
-
-CxList sanitize = Find_Splitting_Sanitizer();
-
-CxList inputs = Find_Read() + Find_DB_Out() - header_inputs;
-
-CxList outputs = Find_Header_Outputs();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

Java / Java_Android / Accessible_Content_Provider

Code changes

```
---
+++
@@ -1 +1 @@
-// This query is deprecated in favour of Exported_Content_Provider_Without_Protective_Permissions
+//This query is deprecated.
```

Java / Java_Android / Exported_Service_Without_Permissions

Code changes

```
---
+++
@@ -1 +1 @@
-// This query is deprecated in favour of Exported_Service_Without_Protective_Permissions
+//This query is deprecated.
```

Java / Java_Android / Insufficient_Sensitive_Application_Layer

Code changes

```
---
+++
@@ -20,6 +20,14 @@

    // Find outputs that performed over HTTP
    CxList write = All.NewCxList(Find_Write(), Find_Request());
+   // File output operations don't imply remote connections
+   CxList sanitizerWrite = All.FindByMemberAccesses(new string[] {
+       "FileWriter.write*",
+       "PrintWriter.*",
+       "FileOutputStream.*",
+       "XMLStreamWriter.*",
+       });
+   write -= sanitizerWrite;

    CxList outInfluencedByHttp = write * write.DataInfluencedBy(pureHTTP);
```

Java / Java_GWT / JSON_Hijacking

Code changes

```
---
+++
@@ -1,19 +1 @@
-//DWR framework prevents javascript hijacking
-CxList dwrFramework = All.FindByName("*dwr.util*", true);
-
-// we'll add other frameworks that take care of javascript hijacking to this list
-CxList CleanAJAXFramework = All.NewCxList(dwrFramework);
-
-if (CleanAJAXFramework.Count == 0)
-{
-   CxList unkRefs = Find_UnknownReference();
-   unkRefs.Add(Find_ObjectCreations());
-   CxList json = unkRefs.FindByTypes(new string[]{"JsonValue", "JsonArray", "JsonParser"}, false);
-
-   if (json.Count > 0)
-   {
-       CxList names = All.NewCxList(All.FindByNames(new string [] {"*select*", "*exec*"}, false));
-       CxList db = Find_DB_Out().DataInfluencedBy(names);
-       result = json.DataInfluencedBy(db).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
-   }
-}
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_2nd_Order_SQL_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_CGI_Stored_XSS

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_CGI_Stored_XSS is deprecated");
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_CSRF

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_CSRF is deprecated");
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_DB_Parameter_Tampering is deprecated");
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_Parameter_Tampering is deprecated");
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_SQL_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
```

```
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_SQL_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Heuristic / Heuristic_Stored_XSS

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Heuristic_Stored_XSS is deprecated");
+//This query is deprecated.
```

Java / Java_High_Risk / Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -29,7 +29,7 @@

 CxList conditions = Find_Ifs().CxSelectDomProperty<IfStmt>(x => x.Condition);
 CxList varDeclsRefs = unkRefs.FindAllReferences(varDeclsInForEach);
-varDeclsRefs-= varDeclsRefs.GetByAncs(conditions);
+varDeclsRefs -= varDeclsRefs.GetByAncs(conditions);
 outputs.Add(varDeclsRefs.GetMembersOfTarget());

@@ -57,6 +57,11 @@
 getsToRemove.Add(cookieGetValue.NotInfluencedBy(cookieSetValue));
 sanitized.Add(getsToRemove);

+// Remove outputs inside of esapi encode sanitized tags
+CxList esapiEncode = sanitized.FindByMemberAccesses(new string[]{"esapi.encodeForHTML*", "esapi.encodeForXML*",
+   "esapi.encodeForURL", "esapi.encodeForCSS", "esapi.encodeForJavaScript"}, false);
+outputs -= outputs.GetByAncs(esapiEncode);
+
 result = inputs.InfluencingOnAndNotSanitized(outputs, sanitized)
     .ReduceFlowByPragma()
     .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Java / Java_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_Blind_SQL_Injections();
+//This query is deprecated.
```

Java / Java_Low_Visibility / Channel_Accessible_by_NonEndpoint

Code changes

```
---
+++
@@ -1,30 +1 @@
-CxList methods = Find_Methods();
-
-// Find all outputs that have "print*" or "write*"
-CxList outputsPrintWrites = Find_Outputs();
-CxList outputs = outputsPrintWrites.FindByShortNames(new string [] {"print*", "write*"});
-/*
-Bypassing:
-// When the output is response, it will be checked response.write will be checked if it is not secured
-CxList response = All.FindByType("HttpServletResponse").FindByType(typeof(ParamDecl));
-CxList isSecure = Find_Conditions().FindByMemberAccess("HttpServletRequest.isSecure");
-CxList secureIf = isSecure.GetFathers();
-CxList outputsResponse = outputs - outputs.GetByAncs(secureIf);
-// Find insecured response, influenced by inputs
-outputsResponse = response.DataInfluencingOn(outputsResponse).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
-*/
-
-// When the output is Socket, it is not secured (not SSLSocket)
-CxList Socket = methods.FindByMemberAccess("Socket.getOut*").GetTargetOfMembers();
-//Socket.Add(All.FindByMemberAccess("SocketChannel.open"));
-
-//Secure
-CxList wrapSSL = methods.FindByMemberAccess("SSLEngine.wrap");
-//Parameters that are secure
-CxList wrap_param = All.FindAllReferences(All.GetParameters(wrapSSL, 1));//Get output from wrap(passed by reference)
-//Outputs that use secure parameters
-CxList sanitized_outputs = wrap_param.DataInfluencingOn(outputs).GetLastNodesInPath();
-//Sockets that influce outputs that dont have secure parameters
-CxList outputsSocket = Socket.DataInfluencingOn(outputs - sanitized_outputs);
-
-result = outputsSocket.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);// +outputsResponse;
+//This query is deprecated.
```

Java / Java_Low_Visibility / Cleansing_Canonicalization_and_Comparison_Errors

Code changes

```
---
+++
@@ -1,27 +1 @@
-CxList methods = Find_Methods();
-
-CxList inputs = methods.FindByMemberAccesses(new string[]{
-   "*HttpServletRequest.getRequestURI",
-   "*HttpServletRequest.getRequestURL",
-   "*HttpServletRequest.getServletPath"});
```

```
-
-CxList sanitize = methods.FindByMemberAccesses(new string[]{
-   "URLDecoder.decode",
-   "Encoder.decodeForURL"});
-
-sanitize.Add(Get_ESAPI().FindByMemberAccess("Encoder.canonicalize"));
-
-CxList emptyStringFathers = Find_Empty_Strings().GetFathers();
-CxList emptyStringBinaryExpr = emptyStringFathers.GetByBinaryOperator(BinaryOperator.IdentityEquality);
-emptyStringBinaryExpr.Add(emptyStringFathers.GetByBinaryOperator(BinaryOperator.IdentityInequality));
-
-
-CxList binaryExpr = Find_BinaryExpr();
-
-CxList nullFathers = Find_NullLiteral().GetFathers() * binaryExpr;
-
-CxList toRemove = All.NewCxList(emptyStringBinaryExpr, nullFathers);
-
-binaryExpr -= toRemove;
-
-result = binaryExpr.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Improper_Resource_Access_Authorization

Code changes

```
---
+++
@@ -1,206 +1 @@
-CxList inputs = Find_Inputs();
-CxList read = Find_Read_NonDB();
-CxList fileReads = Find_FileSystem_Read();
-CxList methods = Find_Methods();
-CxList methodDecls = Find_MethodDecls();
-CxList unkRefs = Find_UnknownReference();
-CxList memberAccesses = Find_MemberAccess();
-CxList declarators = Find_Declarators();
-CxList paramss = Find_Params();
-
-// Find database and file accesses
-CxList db = Find_DB();
-CxList dataAccess = All.NewCxList(db, Find_IO(), fileReads, Find_FileSystem_Write());
-
-// Only consider methods
-dataAccess = dataAccess.FindByType<MethodInvokeExpr>();
-
-// Remove results in tests
-CxList testAndOtherItems = Find_CustomAttribute().FindByShortName("Test").GetAncOfType<MethodDecl>();
-// H2 TestDB
```

```
-CxList h2TestClasses = Find_TypeRef().FindByType("TestDb").GetFathers().GetFathers().FindByType<ClassDecl>();
-testAndOtherItems.Add(h2TestClasses);
-// Generic *Test(s) classes
-CxList compUnits = Find_Classes().FindByFileNames(new string[] {"*Tests.java", "*Test.java"});
-testAndOtherItems.Add(compUnits);
-
-// Remove data from main methods and static blocks
-CxList mainMethods = methodDecls.FindByShortName("main")
-   .FindByFieldAttributes(Modifiers.Static | Modifiers.Public);
-testAndOtherItems.Add(mainMethods, methodDecls.FindByShortName("CxStaticBlock1"));
-dataAccess -= dataAccess.GetByAncs(testAndOtherItems);
-
-CxList suspectConditionParams = All.NewCxList(unkRefs, memberAccesses, methods);
-
-CxList toRemove = All.NewCxList();
-
-string[] toRemoveNames = new string[] {
-   "System.getProperty",
-   "CallableStatement.get*",
-   "CallableStatement.set*",
-   "HttpServletRequest.get*",
-   "SQLResultSetReader.get*",
-   "ResultSet.setAutoClose",
-   "ResultSet.close",
-   "ResultSet.clearWarnings",
-   "ResultSet.is*",
-   "ResultSet.wasNull",
-   "ResultSet.absolute",
-   "ResultSet.next",
-   "ResultSet.get*",
-   "JSPWriter.print*",
-   "File.length",
-   "File.lastModified",
-   "File.can*"
-   };
-toRemove.Add(dataAccess.FindByMemberAccesses(toRemoveNames), dataAccess.FindByShortName("getenv"));
-
-CxList searchSpace = All.NewCxList(unkRefs, declarators);
-
-toRemove.Add(searchSpace.InfluencedBy(db));
-
-dataAccess -= toRemove;
-
-//remove non-db methods influenced by HttpServletResponse because they are not considered a sensitive resource (e.g. println)
-CxList httpServletResponse = unkRefs.FindByTypes(new string[]{"HttpServletResponse","ServletResponse"});
-
-CxList dataAccessParameters = All.GetParameters(dataAccess);
-
```

```
-dataAccess -= dataAccessParameters;
-
-
-CxList dataAccessInfluencedByServlet = All.NewCxList();
-
-// Remove generated refs
-CxList generatedItems = httpServletResponse.GetByAncs(methodDecls.FindByShortName("CxHttpServletExtension"));
-httpServletResponse = httpServletResponse - generatedItems;
-
-// Heuristic: remove args from printStackTrace
-CxList printTraceMethods = methods.FindByShortName("printStackTrace");
-httpServletResponse -= httpServletResponse.GetByAncs(printTraceMethods);
-
-int limit = 30000;
-
-// check the size of sink and source of influencing calculation
-if ((dataAccess.Count > limit) && (httpServletResponse.Count > limit))
-{
-   // select only sink that appears in source file and vice versa
-   CxList tempDataAccess = dataAccess.FindByFiles(httpServletResponse);
-   CxList temphttpServletResponse = dataAccess.FindByFiles(tempDataAccess);
-
-   // select size of the loop
-   CxList small = All.NewCxList();
-   CxList big = All.NewCxList();
-   bool forward = true;
-   if (tempDataAccess.Count > temphttpServletResponse.Count)
-   {
-       small.Add(temphttpServletResponse);
-       big.Add(tempDataAccess);
-       forward = false;
-   }
-   else
-   {
-       small.Add(tempDataAccess);
-       big.Add(temphttpServletResponse);
-       forward = true;
-   }
-
-   int count = 0;
-
-   // create dictionary that includes source or sink sorted by file name (value). Key is fileId of this file
-   Dictionary<int,CxList> smallDic = new Dictionary<int,CxList>();
-
-   foreach (CxList tmp in small)
-   {
-       CSharpGraph sg = tmp.GetFirstGraph();
-       int fileId = sg.LinePragma.FileId;
```

```
-       CxList cxList;
-       if (!smallDic.TryGetValue(fileId, out cxList))
-       {
-           cxList = All.NewCxList();
-           smallDic[fileId] = cxList;
-       }
-       cxList.Add(tmp);
-       smallDic[fileId] = cxList;
-   }
-
-
-   // create dictionary that includes source or sink sorted by file name (value). Key is fileId of this file
-   Dictionary<int,CxList> bigDic = new Dictionary<int,CxList>();
-   foreach (CxList tmp1 in big)
-   {
-       CSharpGraph sg1 = tmp1.GetFirstGraph();
-       int fileId1 = sg1.LinePragma.FileId;
-       CxList cxList1;
-       if (!bigDic.TryGetValue(fileId1, out cxList1))
-       {
-           cxList1 = All.NewCxList();
-           bigDic[fileId1] = cxList1;
-       }
-       cxList1.Add(tmp1);
-       bigDic[fileId1] = cxList1;
-   }
-
-   // implement searching of influencing algorithm for each file separatly
-   foreach (int keyValue in smallDic.Keys)
-   {
-       CxList tempSmall = smallDic[keyValue];
-       CxList tempBig;
-
-       if (!bigDic.TryGetValue(keyValue, out tempBig))
-       {
-           continue;
-       }
-
-       CxList tempRes;
-       if (forward)
-           tempRes = tempSmall.InfluencedByAndNotSanitized(tempBig, dataAccessParameters);
-       else
-           tempRes = tempBig.InfluencedByAndNotSanitized(tempSmall, dataAccessParameters);
-
-       dataAccessInfluencedByServlet.Add(tempRes.GetLastNodesInPath());
-       count++;
-   }
-}
```

```
-else
-{
-   dataAccessInfluencedByServlet = dataAccess.InfluencedByAndNotSanitized(httpServletResponse, dataAccessParameters);
-   dataAccessInfluencedByServlet = dataAccessInfluencedByServlet.GetLastNodesInPath();
-}
-
-dataAccess -= dataAccessInfluencedByServlet;
-
-
-//Find unauthenticated authorization of requests from http security from spring framework
-CxList authorizeRequests = methods.FindByMemberAccess("HttpSecurity.authorizeRequests");
-
-CxList authenticated = authorizeRequests.GetRightmostMember().FindByShortName("authenticated");
-CxList anyRequestAuthenticated = authenticated.GetTargetOfMembers().FindByShortName("anyRequest");
-authorizeRequests -= anyRequestAuthenticated.GetLeftmostTarget().GetMembersOfTarget();
-dataAccess.Add(authorizeRequests);
-
-// Find conditions that make use of *auth* and *admin* words //- heuristics
-CxList conditions = suspectConditionParams.GetByAncs(Find_Conditions());
-string[] possibleStr = new string[]{
-       "*admin*", "*allow",
-       "*allowed", "*allows", "*deny", "*denies", "*denied",
-       "*authoriz*", "*permission*"
-       };
-CxList potentialCheck = conditions.FindByShortNames(possibleStr, false);
-CxList ifStmtCheck = potentialCheck.GetAncOfType<IfStmt>();
-
-// All data accesses that contain permissions to do the access
-CxList dataAccessCheck = dataAccess.GetByAncs(ifStmtCheck);
-CxList dataAccessUnsafe = dataAccess - dataAccessCheck;
-dataAccessUnsafe -= Find_Properties_Files();
-
-// To show only params from methods (except reads) that are affected by inputs
-read.Add(fileReads);
-CxList unsafeReads = dataAccessUnsafe * read;
-CxList unsafeWriteMethods = dataAccessUnsafe - unsafeReads;
-CxList unsafeParams = All.NewCxList(unkRefs, methods, memberAccesses, paramss);
-CxList flowInputs = inputs.InfluencingOn(unsafeParams.GetParameters(unsafeWriteMethods));
-
-result = unsafeReads;
-result.Add(dataAccessUnsafe.FindByParameters(flowInputs.GetLastNodesInPath()),
-   dataAccessUnsafe * authorizeRequests);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Improper_Resource_Shutdown_or_Release

Code changes

```
---
+++
```

```
@@ -13,7 +13,7 @@


 List<string> relevantTypes = new List<string> {
         "AbstractInterruptibleChannel", "AbstractSelectableChannel", "AbstractSelector", "AsynchronousFileChannel",
-        "AsynchronousServerSocketChannel", "AsynchronousSocketChannel", "AudioInputStream", "AutoCloseable",
+        "AsynchronousServerSocketChannel", "AsynchronousSocketChannel", "AudioInputStream", "AutoCloseable","BufferedReader",
         "CharArrayReader", "CharArrayWriter", "CheckedInputStream", "CheckedOutputStream", "CipherInputStream",
         "CipherOutputStream", "Closeable","CloseableHttpClient","CloseableHttpResponse", "DatagramChannel",
         "DatagramSocket", "DataInputStream", "DataOutputStream", "Deflater", "DeflaterInputStream",
```

Java / Java_Low_Visibility / Plaintext_Storage_in_a_Cookie

Code changes

```
---
+++
@@ -1,31 +1 @@
-CxList methods = Find_Methods();
-
-CxList addCookie = methods.FindByMemberAccesses(new string [] {"HttpServletRequest.addCookie",
-   "HTTPUtilities.safeAddCookie"});// ESAPI
-addCookie.Add(methods.FindByName("*Response.addCookie", false));
-
-CxList cookies = All.FindByType("Cookie");
-CxList cookieParam = All.GetParameters(cookies, 1).FindByType<UnknownReference>();
-
-CxList strings = Find_Strings();
-
-CxList toRemove = Find_Empty_Strings();
-toRemove.Add(strings.FindByShortNames(new string [] {"\"-\"", "\";\"", "';", "\":\"", "\"_\"", "\"/\"", "\"\\\""})); //dividers
-
-CxList plainText = strings - toRemove;
-
-CxList binaryExprs = plainText.GetAncOfType<BinaryExpr>()
-   .FilterByDomProperty<BinaryExpr>(_ => _.Operator == BinaryOperator.Add);
-CxList descendantBinary = All.NewCxList();
-foreach(CxList binaryExpr in binaryExprs){
-   descendantBinary.Add(binaryExprs.GetByAncs(binaryExpr) - binaryExpr);
-}
-plainText -= plainText.GetByAncs(descendantBinary);
-
-CxList sanitize = Find_General_Sanitize();
-CxList javaSqlMethods = methods.FindByMemberAccess("DriverManager.getConnection");
-CxList connections = Find_UnknownReference().FindAllReferences(javaSqlMethods.GetAssignee());
-javaSqlMethods.Add(connections.GetMembersOfTarget().FindByShortName("createStatement"));
-sanitize.Add(javaSqlMethods);
-
-result = plainText.InfluencingOnAndNotSanitized(cookieParam, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Potential_ReDoS

Code changes

```
---

+++

@@ -1,8 +1 @@

-CxList filter = Potential_ReDoS_In_Match();

-filter.Add(Potential_ReDoS_In_Replace(), Potential_ReDoS_In_Static_Field());

-

-CxList evilString = Find_Evil_Strings();

-

-CxList toRemove = All.NewCxList(filter, evilString.DataInfluencingOn(filter));

-

-result = evilString - toRemove;

+//This query is deprecated.
```

Java / Java_Low_Visibility / Potential_ReDoS_By_Injection

Code changes

```
---

+++

@@ -1,8 +1 @@

-CxList inputs = Find_Interactive_Inputs();

-CxList regexInFirstParam = Find_Match();

-

-regexInFirstParam.Add(Find_Replace_Regex_In_First_Param());

-result = Find_ReDoS(inputs, regexInFirstParam, 0, true, false);

-

-CxList regexInSecondParam = Find_Replace_Regex_In_Second_Param();

-result.Add(Find_ReDoS(inputs, regexInSecondParam, 1, false));

+//This query is deprecated.
```

Java / Java_Low_Visibility / Potential_ReDoS_In_Match

Code changes

```
---

+++

@@ -1 +1 @@

-result = Find_ReDoS(Find_Evil_Strings(), Find_Match(), 0, true, false);

+//This query is deprecated.
```

Java / Java_Low_Visibility / Potential_ReDoS_In_Replace

Code changes

```
---

+++

@@ -1,5 +1 @@

-CxList evilStringsForReplace = Find_Evil_Strings_For_Replace();

-

-result = Find_ReDoS(evilStringsForReplace, Find_Replace_Regex_In_First_Param(), 0, true, false);

-
```

```
-result.Add(Find_ReDoS(evilStringsForReplace, Find_Replace_Regex_In_Second_Param(), 1, true, false));
+//This query is deprecated.
```

Java / Java_Low_Visibility / Potential_ReDoS_In_Static_Field

Code changes

```
---
+++
@@ -1,14 +1 @@
-// Find all evil strings
-CxList evilStrings = Find_Evil_Strings();
-
-// Sanitization
-CxList sanitize = Find_Integers();
-
-// Find all regex commands
-CxList regex = Find_Methods().FindByMemberAccess("Pattern.compile");
-
-// Add static regexes (these do not influence their references, so needed here)
-CxList staticFields = All.FindByFieldAttributes(Modifiers.Static);
-staticFields = staticFields.FindByType<ConstantDecl>() + (staticFields * Find_Field_Decl());
-
-result = evilStrings.InfluencingOnAndNotSanitized(regex.GetByAncs(staticFields), sanitize);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Reliance_on_Cookies_in_a_Decision

Code changes

```
---
+++
@@ -1,12 +1 @@
-CxList cookies =
-   Find_Methods().FindByMemberAccess("request.getCookies");
-cookies.Add(Find_CookieValue_Annotation());
-
-CxList cond = Find_Conditions();
-
-// Ignore conditions like if (cookie != null)
-CxList sanitizers = Find_NullLiteral();
-sanitizers.Add(All.FindByMemberAccesses(new string [] {"*.length", "*.Count"}, false));
-sanitizers = sanitizers.GetFathers();
-
-result = cond.InfluencedByAndNotSanitized(cookies, sanitizers);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Suspected_XSS

Code changes

```
---
+++
@@ -1,55 +1 @@
```

```
-////////////////////////////////////////////////////
-// Query Suspected_XSS
-//
-// This query finds suspected XSS
-// Every session.getAttribute is defined as suspected XSS input
-// We try find targets for every non active output (without flow)
-////////////////////////////////////////////////////
-
-// Every session attribute is suspected as input
-CxList getAttribs = Find_Jsp_Code().FindByMemberAccess("HttpSession.getAttribute");
-
-CxList assignGetAttribs = getAttribs.GetAncOfType<AssignExpr>();
-CxList declStmtGetAttribs = getAttribs.GetAncOfType<VariableDeclStmt>();
-
-CxList filteredAttributes = All.NewCxList();
-filteredAttributes.Add(
-   assignGetAttribs,
-   declStmtGetAttribs);
-
-CxList inputs = All.GetByAncs(filteredAttributes).FindByAssignmentSide(CxList.AssignmentSide.Left);
-
-// Refine Sanitizer
-CxList methodReturnInt = All.FindByReturnType("int");
-CxList returnStatements = Find_ReturnStmt();
-CxList methodInvSanitizers = returnStatements.GetByMethod(methodReturnInt);
-
-CxList sanitized = Find_XSS_Sanitize();
-sanitized.Add(
-   Find_DB_In(),
-   methodInvSanitizers);
-
-// End Refine Sanitizer
-
-CxList outputs = Find_XSS_Outputs();
-
-// Refine outputs
-// Get non active nodes - Output nodes with no flow from input
-CxList activeNodes = inputs.DataInfluencingOn(outputs).GetLastNodesInPath();
-CxList nonActiveNodes = (outputs - activeNodes).FindByType<MethodInvokeExpr>();
-
-// Try to find flow to the targets of the non-active nodes
-CxList unknownRefs = nonActiveNodes.GetTargetOfMembers();
-unknownRefs = unknownRefs - methodInvSanitizers.GetTargetOfMembers();
-
-outputs.Add(unknownRefs);
-
-CxList binExprOutputs = outputs.FindByType<BinaryExpr>();
-CxList membersInOutputs = All.FindDescendantsOfType<MethodInvokeExpr>(binExprOutputs);
```

```
-
-outputs -= binExprOutputs;
-outputs.Add(membersInOutputs);
-// end Refine outputs
-
-result = inputs.InfluencingOnAndNotSanitized(outputs, sanitized);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Use_of_Client_Side_Authentication

Code changes

```
---
+++
@@ -1,11 +1 @@
-CxList htmlOutput = Find_Web_Outputs();
-htmlOutput.Add(Find_Html_Outputs());
-
-CxList strings = Find_Strings();
-CxList htmlRemarks = strings.FindByNames(new string [] {"*<script*", "*</script>*"});
-
-htmlOutput = htmlOutput.DataInfluencedBy(htmlRemarks);
-
-CxList pass = Find_Password_Strings();
-
-result = htmlOutput.DataInfluencedBy(pass);
+//This query is deprecated.
```

Java / Java_Low_Visibility / Use_Of_getenv

Code changes

```
---
+++
@@ -1,2 +1 @@
-//This query is deprecated
-cxLog.WriteDebugMessage("The query Use_Of_getenv is deprecated");
+//This query is deprecated.
```

Java / Java_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,10 +1,7 @@
-CxList emptyString = Find_Empty_Strings();
-CxList nullsString = All.FindByName("null");
 CxList psw = Find_All_Passwords();
 CxList stringLiterals = Find_Strings();
 CxList passwordString = Find_Password_Strings();
 CxList methods = Find_Methods();
-
```

```
 CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
 CxList passAndStrings = All.NewCxList(passwordString, psw);


@@ -13,13 +10,10 @@
 CxList pswInLSideDecl = pswInLSide.FindByType<Declarator>();


 CxList strLiterals = All.NewCxList(stringLiterals);
-strLiterals -= emptyString;
-strLiterals -= nullsString;
-
-//when the hardcoded string includes a space or dot we believe
-//it is not a password string
+strLiterals -= Find_Empty_Strings();
+strLiterals -= All.FindByName("null");
 strLiterals -= strLiterals.FindByNames("* *","*.*","*/*","*\\*");
-
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);


 CxList litInRSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
 CxList initializedPassword = pswInLSideDecl.FindByInitialization(litInRSide);
@@ -118,14 +112,13 @@
 CxList passwordInPreferences = passwordValueInPreference.InfluencedBy(stringLiterals);
 passwordInPreferences.Add(hardcodedPasswordString);


-result = initializedPassword;
-
 CxList setProp = methods.FindByMemberAccess("ContextResource.setProperty");
 CxList setPropWithPass = passAndStrings.GetByAncs(setProp);
 result.Add(stringLiterals.GetParameters(setPropWithPass.GetAncOfType<MethodInvokeExpr>(), 1));


 // All
-result.Add(equalsPassword,
+result.Add(initializedPassword,
+           equalsPassword,
            mapPassword,
            assignPassword,
            paramsAffectedByString,
```

Java / Java_Low_Visibility / UTF7_XSS

Code changes

```
---
+++
@@ -1,13 +1 @@
-CxList UTF7 = Find_Strings().FindByName("UTF-7");
-
-CxList response = All.FindByName("*Response.setCharacterEncoding", false);
-response.Add(Find_Methods().FindByMemberAccess("HttpServletResponse.setCharacterEncoding"));
-
```

```
-UTF7 = response.DataInfluencedBy(UTF7);
-
-if (UTF7.Count > 0)
-{
-   CxList outputs = Find_XSS_Outputs();
-   CxList inputs = Find_Interactive_Inputs();
-   result = outputs.DataInfluencedBy(inputs);
-}
+//This query is deprecated.
```

Java / Java_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1 +1 @@
-result = Find_DB_Parameter_Tampering();
+//This query is deprecated.
```

Java / Java_Medium_Threat / Hardcoded_password_in_Connection_String

Code changes

```
---
+++
@@ -12,7 +12,7 @@
 sanitizers.Add(hardcodedStringSanitizers.FindByShortNames(safeMethods));
 // Only first parameters of get/setProperty methods are safe
-CxList propertyMethods = methods.FindByMemberAccesses("Properties", new string[]{ "getProperty", "setProperty"});
+CxList propertyMethods = methods.FindByMemberAccesses(new [] {"Properties", "System"}, new string[]{ "getProperty", "setProperty"});
 sanitizers.Add(All.GetParameters(propertyMethods, 0));

 // Find creation of connections or connection strings, influenced by password
@@ -72,7 +72,7 @@
 CxList notPswParamInMethod = All.GetParameters(notPswInMethod);
 notPswParamInMethod -= pswParamInMethod;
 sanitizers.Add(notPswParamInMethod,
-           Find_Hardcoded_Key_Sanitizers());
+   Find_Hardcoded_Key_Sanitizers());

 CxList propDeclBuilders = objects.FindByShortName("Builder").GetMembersOfTarget();
 CxList nameStrings = Find_Strings().GetParameters(methods.FindByShortName("name"));
```

Java / Java_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query HTTP_Response_Splitting is deprecated");
```

Java / Java_Medium_Threat / Improper_Restriction_of_Stored_XXE_Ref

Code changes

```
---
+++
@@ -1,9 +1,11 @@
 // Find Stored_XXE (XML External Entity vulnerability) in Java
 CxList ifs = Find_Ifs();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

-CxList inputs = Find_Read_NonDB();
-inputs.Add(Find_DB_Out());
-inputs.Add(Find_ObjectCreations().FindByShortName("File*"));
+CxList inputs = All.NewCxList(
+   Find_Read_NonDB(),
+   Find_DB_Out(),
+   Find_ObjectCreations().FindByShortName("File*"));

 CxList xxe = Find_XXE_Requests();

@@ -42,4 +44,10 @@
 sanitizers -= All.FindByType("Boolean");
 sanitizers -= Find_Bytes();

-result = inputs.InfluencingOnAndNotSanitized(xxe, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result = xxe.InfluencedByAndNotSanitized(inputs, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+//Ensure transform method flows pass through first parameter
+CxList transformMethods = result.GetLastNodesInPath().FindByMemberAccess("Transformer.transform");
+CxList tfMethodSecParam = paramValue.GetParameters(transformMethods, 1);
+CxList invalidTransformFlow = result.IntersectWithNodes(tfMethodSecParam);
+result -= invalidTransformFlow;
```

Java / Java_Medium_Threat / Missing_HSTS_Header

Code changes

```
---
+++
@@ -40,6 +40,12 @@
     result.Add(HeaderInJSP);

     CxList writeResponse = Find_HTTP_Response_Write();
+
+   // Absence of an HSTS header in a client-side code (HttpClient.java) doesn't imply vulnerability to 'missing HSTS Header';
+   CxList httpClient = writeResponse.GetAncOfType<ClassDecl>().FindByShortName("HttpClient");
+   httpClient = writeResponse.FindDescendantsOfType<MethodInvokeExpr>(httpClient);
+   writeResponse -= httpClient;
+
     if(writeResponse.Count > 0 || HeaderInJSP.Count > 0){
         CxList aux = writeResponse.GetLeftmostTarget();
```

```
        if(aux.Count == 0){
```

Java / Java_Medium_Threat / Multiple_Binds_to_the_Same_Port

Code changes

```
---
+++
@@ -1,57 +1 @@
-/*
-   This query looks for methods in which server socket bindings to variable interfaces
-   with the same port number happen.
-
-   EX: public void bindSocket(Socket socket,Interface interface){
-           socket.bind(new InetSocketAddress(interface, 2000));
-       }
-*/
-CxList inputs = Find_Interactive_Inputs();
-CxList integerLiterals = Find_IntegerLiterals();
-CxList zeros = integerLiterals.FindByShortName("0");
-CxList integers = Find_Integers();
-integers.Add(integerLiterals,
-           integers.DataInfluencedBy(inputs)); // Note that integerLiterals.DataInfluencedBy(inputs)
-                                               // is not empty (as could be initially expected).
-
-CxList allSocketAddress = All.FindByType("InetSocketAddress");
-CxList portNumberAsSecondParameter = integers.GetParameters(allSocketAddress, 1);
-
-/*
-   Passing port 0 to the InetSocketAddress constructor instructs it to bind
-   to a random available temporary port, which is safe and therefore not a result
-*/
-
-zeros.Add(portNumberAsSecondParameter.DataInfluencedBy(zeros));
-
-portNumberAsSecondParameter -= zeros;
-
-/*
-   Only constructors of InetSocketAddress with two parameters matter since initializing
-   it with a port number only (one parameter), binds it with the wildcard interface
-   (0.0.0.0) which can't be binded twice.
-*/
-CxList socketAddressWithTwoParameters = allSocketAddress.FindByParameters(portNumberAsSecondParameter);
-CxList interfaceValue = All.GetParameters(socketAddressWithTwoParameters, 0);
-interfaceValue -= interfaceValue.FindByType<Param>();
-
-/*
-   Remove results binding to the same interface. i.e. using a string literal as interface.
-*/
-CxList strings = Find_Strings();
```

```
-CxList safeInterfaceValue = interfaceValue * strings;
-safeInterfaceValue.Add(interfaceValue.DataInfluencedBy(strings) - interfaceValue.DataInfluencedBy(inputs));
-
-
-/*
-   Remove references to localhost name since they are constant although possibly not strings.
-*/
-safeInterfaceValue.Add(interfaceValue.FindByShortName("*localhost*", false));
-/*
-   Return parent method encapsulating unsafe bindings since calling them multiple times can lead
-   to the vulnerability.
-*/
-CxList methods = Find_Methods();
-CxList bindings = methods.FindByShortName("bind", false);
-CxList unsafeSocketAddresses = socketAddressWithTwoParameters.FindByParameters(interfaceValue - safeInterfaceValue);
-result = methods.GetMethod(bindings.DataInfluencedBy(unsafeSocketAddresses));
+//This query is deprecated.
```

Java / Java_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -92,17 +92,20 @@
    );

 // Define sanitize
-CxList sanitize = All.NewCxList(Find_DB(), Find_Encrypt(), Find_UnitTest_Code(), Find_HashSanitize());
+CxList sanitize = All.NewCxList(Find_DB(), Find_Encrypt(), Find_UnitTest_Code(), Find_HashSanitize(), Find_DataSource_Sanitizers());

 // Add additional "integer" sanitizers
 sanitize.Add(All.FindByShortNames(new string[] {"size", "length", "Index*", "indexOf"}, false),
    All.FindByName("*boolean.class.cast", StringComparison.OrdinalIgnoreCase),
-   methods.FindByMemberAccess("Boolean.parse*"), methods.FindByReturnType("bool"), Find_BooleanLiteral());
+   methods.FindByMemberAccess("Boolean.parse*"), methods.FindByReturnType("bool"), Find_BooleanLiteral(),
+   methods.FindByMemberAccess("Process.waitFor"));

 CxList javaSqlMethods = methods.FindByMemberAccess("DriverManager.getConnection");
 CxList connections = unknRefs.FindAllReferences(javaSqlMethods.GetAssignee());
 javaSqlMethods.Add(connections.GetMembersOfTarget().FindByShortName("createStatement"));
 sanitize.Add(javaSqlMethods);
+
+sanitize.Add(All.FindByShortNames(new string[]{"*clientname*", "*username*"}, false));

 // Split personal_info into variables and constants
 CxList variableRef = personal_info - allConstRef;
@@ -131,6 +134,7 @@
        constInfluencedByInput.Add(elem);
    }
```

```
 }
+
 // find all variables that are influencing an output
 CxList variableRefPath = outputs.InfluencedByAndNotSanitized(variableRef, sanitize);
 variableRefPath.Add(variableRef * outputs - sanitize);
```

Java / Java_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -1,19 +1,49 @@
+CxList methods = Find_Methods();
+CxList unkRefs = Find_UnknownReference();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
+
 CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Queue_Inputs());

 // Remove argc/argv from inputs
 CxList mainDeclarations = All.FindByShortName("*main*").FindByType<MethodDecl>();
-CxList argsInputs = All.GetParameters(mainDeclarations);
+CxList argsInputs = paramValue.GetParameters(mainDeclarations);
 inputs -= argsInputs;

 CxList sanitizers = Find_Remote_Requests_Sanitize();
 CxList requests = Find_Remote_Requests();

-// Exclude javax email methods from SSRF checks; they don't trigger network requests,
-// a prerequisite for SSRF vulnerabilities.
-string[] nonSSRFMethods = new string [] {"message.setText", "message.setSubject"};
-CxList nonSSRFMethodCalls = Find_Methods().FindByMemberAccesses(nonSSRFMethods);
-sanitizers.Add(nonSSRFMethodCalls);
+//Transport.send must be influenced by a Properties object with key "mail.smtp.host" influenced by user input
+CxList sendMethods = methods.FindByMemberAccess("Transport.send");
+CxList relevantKey = Find_Strings().FindByShortName("mail.smtp.host");
+CxList propertySetMethods = methods.FindByMemberAccesses("Properties",
+   new[]{"setProperty", "put", "putAll", "putIfAbsent"});

-result = requests.InfluencedByAndNotSanitized(inputs, sanitizers).ReduceFlowByPragma();
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+IEnumerable<IAbstractValue> keyAbsValue = relevantKey.CxSelectElementValue<Expression, IAbstractValue>(e => e.AbsValue);
+CxList relevantKeyProperties = paramValue.GetParameters(propertySetMethods).FindByAbstractValue(
+   absValue => absValue.IncludedIn(keyAbsValue.FirstOrDefault()));
+
+requests -= paramValue.GetParameters(sendMethods.NotInfluencedBy(relevantKeyProperties));
+
+// remove 2nd and 3rd params from DriverManager.getConnection since only the first one can be valid
+CxList driverManagerGetConnections = methods.FindByMemberAccess("DriverManager.getConnection");
```

```
+requests -= requests.GetParameters(driverManagerGetConnections, 1);
+requests -= requests.GetParameters(driverManagerGetConnections, 2);
+
+result = requests.InfluencedByAndNotSanitized(inputs, sanitizers)
+   .ReduceFlowByPragma()
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+//Removing openStream results on previously Opened Connection
+CxList resultLastNodes = result.GetLastNodesInPath();
+CxList openConnection = resultLastNodes.FindByShortName("openConnection");
+CxList openStream = resultLastNodes.FindByShortName("openStream");
+CxList toRemove = All.NewCxList();
+foreach(CxList opConn in openConnection){
+   CxList openConnectionVar = opConn.GetTargetOfMembers();
+   CxList subsequentVarRefs = unkRefs.FindAllReferences(openConnectionVar)
+       .Filter(x => x.Line > openConnectionVar.GetDOMPropertiesOfFirst().Line);
+   CxList openStreamSameVar = subsequentVarRefs.GetMembersOfTarget() * openStream;
+   toRemove.Add(openStreamSameVar.InfluencedByAndNotSanitized(inputs, sanitizers));
+}
+result -= toRemove;
```

Java / Java_Medium_Threat / Unnormalize_Input_String

Code changes

```
---
+++
@@ -1 +1 @@
-//Deprecated query: see Input_Path_Not_Canonicalized
+//This query is deprecated.
```

Java / Java_Medium_Threat / Unvalidated_Forwards

Code changes

```
---
+++
@@ -1,13 +1,15 @@
-//
 CxList inputs = Find_Interactive_Inputs();

 CxList requestDispatcher = Find_Request_Dispatcher_Unvalidated_Forwards();
+CxList unkRefStrMethd = All.NewCxList(Find_UnknownReference(), Find_Strings(), Find_Methods());
+CxList requestDispatcherParams = unkRefStrMethd.GetParameters(requestDispatcher);

-CxList unkRefStrMethd = Find_UnknownReference();
-unkRefStrMethd.Add(Find_Strings(), Find_Methods());
+CxList unvalidatedForwards = Find_Forward_Unvalidated_Forwards();

-CxList rdParams = unkRefStrMethd.GetParameters(requestDispatcher);
+CxList sanitizers = All.NewCxList(
+   Find_General_Sanitize(),
```

```
+   // Flow should not flow past forward
+   unkRefStrMethd.GetParameters(unkRefStrMethd.FindByShortName("forward")));

-CxList forward = Find_Forward_Unvalidated_Forwards();
-
-result= inputs.DataInfluencingOn(rdParams).DataInfluencingOn(forward);
+result = inputs.InfluencingOnAndNotSanitized(requestDispatcherParams, sanitizers)
+   .InfluencingOnAndNotSanitized(unvalidatedForwards, sanitizers);
```

Java / Java_Medium_Threat / Unvalidated_SSL_Certificate_Hostname

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query has been deprecated
-cxLog.WriteDebugMessage("The query Unvalidated_SSL_Certificate_Hostname has been deprecated.");
+//This query is deprecated.
```

Java / Java_Medium_Threat / Use_of_a_One_Way_Hash_with_a_Predictable_Salt

Code changes

```
---
+++
@@ -19,4 +19,14 @@
 CxList sanitizers = Find_Password_Hash_Sanitize();

+// Results that use a CSPRNG to generate Salt // `Security.getRandomToken` are not a predictable value.
+CxList getRandom = methods.FindByMemberAccess("Security.getRandomToken").GetFathers();
+CxList getRandomRefs = unkRefs.FindAllReferences(getRandom);
+CxList getRandomSafeSalts = All.NewCxList(
+   // If getRandomToken references salts is concatenated with a tainted unknowReference(the tainted value) it sanitizes it.
+   getRandomRefs.GetFathers().CxSelectDomProperty<BinaryExpr>(x => x.Left),
+   getRandomRefs);
+
 result = sinks.InfluencedByAndNotSanitized(sources, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+result -= result.DataInfluencedBy(getRandomSafeSalts);
```

Java / Java_Medium_Threat / Use_of_Insufficiently_Random_Values

Code changes

```
---
+++
@@ -1,2 +1 @@
-//This query is deprecated. Please consider Use_of_Non_Cryptographic_Random instead.
-cxLog.WriteDebugMessage("The query Use_of_Insufficiently_Random_Values is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_Code_Injection

Code changes

```
---

+++

@@ -1,2 +1 @@

-// This query is deprecated

-cxLog.WriteDebugMessage("The query Potential_Code_Injection is deprecated");

+//This query is deprecated.
```

Java / Java_Potential / Potential_Command_Injection

Code changes

```
---

+++

@@ -1,2 +1 @@

-// This query is deprecated

-cxLog.WriteDebugMessage("The query Potential_Command_Injection is deprecated");

+//This query is deprecated.
```

Java / Java_Potential / Potential_Connection_String_Injection

Code changes

```
---

+++

@@ -1,2 +1 @@

-// This query is deprecated

-cxLog.WriteDebugMessage("The query Potential_Connection_String_Injection is deprecated");

+//This query is deprecated.
```

Java / Java_Potential / Potential_GWT_Reflected_XSS

Code changes

```
---

+++

@@ -1,2 +1 @@

-// This query is deprecated

-cxLog.WriteDebugMessage("The query Potential_GWT_Reflected_XSS is deprecated");

+//This query is deprecated.
```

Java / Java_Potential / Potential_Hardcoded_password_in_Connection_String

Code changes

```
---

+++

@@ -1,2 +1 @@

-// This query is deprecated

-cxLog.WriteDebugMessage("The query Potential_Hardcoded_password_in_Connection_String is deprecated");

+//This query is deprecated.
```

Java / Java_Potential / Potential_IO_Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_IO_Reflected_XSS_All_Clients is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_I_Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_I_Reflected_XSS_All_Clients is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_LDAP_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_LDAP_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_O_Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_O_Reflected_XSS_All_Clients is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_Parameter_Tampering

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_Parameter_Tampering is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_Resource_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
```

```
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_Resource_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_SQL_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_SQL_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_Stored_XSS

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_Stored_XSS is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_Use_of_Hard_coded_Cryptographic_Key

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_Use_of_Hard_coded_Cryptographic_Key is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_UTF7_XSS

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_UTF7_XSS is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_XPath_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_XPath_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Potential / Potential_XXE_Injection

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Potential_XXE_Injection is deprecated");
+//This query is deprecated.
```

Java / Java_Spring / Spring_Missing_Expect_CT_Header

Code changes

```
---
+++
@@ -1,9 +1 @@
-CxList expectCtHeader = Find_Spring_Security_Headers("Expect-CT");
-if (expectCtHeader.Count == 0)
-{
-   CxList springImports = Find_Spring_Imports();
-   CxList firstNode = springImports.GetCxListByPath().FirstOrDefault();
-   if(firstNode != null)
-       result.Add(firstNode);
-}
-result.Add(Find_Spring_DisabledDefaultHeaders());
+//This query is deprecated.
```

Java / Java_Spring / Spring_Missing_HSTS_Header

Code changes

```
---
+++
@@ -35,9 +35,16 @@
 hstsHttpSecurity -= toRemove;

 CxList httpSecurityMaxAge = methods.FindByShortName("maxAgeInSeconds").FindByParameters(
-           Find_IntegerLiterals().FilterByDomProperty<IntegerLiteral>(_ => _.Value >= 31536000));
-CxList invalidHstsHttpSecurity = hstsHttpSecurity - hstsHttpSecurity.GetByAncs(httpSecurityMaxAge);
-hstsHeaders.Add(invalidHstsHttpSecurity);
+   Find_IntegerLiterals().FilterByDomProperty<IntegerLiteral>(_ => _.Value >= 31536000));
+CxList httpIncludeSubDomains = methods.FindByShortName("includeSubDomains").FindByParameters(Find_True_Abstract_Value());
+CxList secureHstsHttpSecurityMaxAge = hstsHttpSecurity.GetByAncs(httpSecurityMaxAge);
+CxList secureHstsHttpSecuritySubDomains = hstsHttpSecurity.GetByAncs(httpIncludeSubDomains);
+
+CxList secureHeaders = secureHstsHttpSecurityMaxAge.
+   GetByAncs(secureHstsHttpSecuritySubDomains.GetAncOfType<NamespaceDecl>());
+
+if(secureHeaders.Count == 0 || secureHstsHttpSecuritySubDomains.Count == 0)
+   hstsHeaders.Add(hstsHttpSecurity);

 //HttpSecurity disabled
```

```
 CxList httpSecurityDisabled = hstsHttpSecurity.GetByAncs(methods.FindByShortName("disable"));
```

Java / Java_Spring / Spring_Missing_XSS_Protection_Header

Code changes

```
---
+++
@@ -1,6 +1 @@
-//XML Files
-CxList headersDisabledInXML = cxXPath.FindXmlNodesByLocalName("*.xml", 2, "xss-protection", true, "disabled", "true", false, true);
-
-CxList disabled = Find_Methods().FindByMemberAccess("xssProtection.disable");
-
-result.Add(headersDisabledInXML, disabled, Find_Spring_DisabledDefaultHeaders());
+//This query is deprecated.
```

Java / Java_Spring / Spring_Missing_X_Content_Type_Options

Code changes

```
---
+++
@@ -1,6 +1 @@
-//XML Files
-CxList headersDisabledInXML = cxXPath.FindXmlNodesByLocalName("*.xml", 2, "content-type-options", true, "disabled", "true", false, true);
-
-CxList disabled = Find_Methods().FindByMemberAccess("contentTypeOptions.disable");
-
-result.Add(headersDisabledInXML, disabled, Find_Spring_DisabledDefaultHeaders());
+//This query is deprecated.
```

Java / Java_Stored / Stored_HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,9 +1 @@
-CxList inputs = Find_FileStreams();
-inputs.Add(Find_DB_Out());
-inputs -= Find_Headers();
-
-CxList sanitize = Find_Splitting_Sanitizer();
-
-CxList outputs = Find_Header_Outputs();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

JavaScript / Javascript_Kony / Kony_Unsecure_iOSBrowser_Configuration

Code changes

```
---
+++
```

```
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query Kony_Unsecure_iOSBrowser_Configuration is deprecated");
```

JavaScript / JavaScript_Low_Visibility / Client_Cross_Session_Contamination

Code changes

```
---
+++
@@ -7,4 +7,4 @@
 CxList sessionStorage = storage.GetTargetOfMembers().FindByShortName("sessionStorage").GetMembersOfTarget();
 CxList inputs = storage - sessionStorage;

-result = inputs.DataInfluencedBy(session);
+result = inputs.DataInfluencedBy(session).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

JavaScript / JavaScript_Low_Visibility / Client_Heuristic_Poor_XSS_Validation

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Client_Heuristic_Poor_XSS_Validation is deprecated");
+//This query is deprecated.
```

JavaScript / JavaScript_Low_Visibility / Client_HTML5_Heuristic_Session_Insecure_Storage

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
-cxLog.WriteDebugMessage("The query Client_HTML5_Heuristic_Session_Insecure_Storage is deprecated");
+//This query is deprecated.
```

JavaScript / JavaScript_Low_Visibility / Client_Insufficient_Key_Size

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query Client_Insufficient_Key_Size is deprecated");
```

JavaScript / JavaScript_Low_Visibility / Client_Located_JQuery_Outdated_Lib_File

Code changes

```
---
+++
@@ -1,2 +1 @@
-// This query is deprecated
```

```
-cxLog.WriteDebugMessage("The query Client_Located_JQuery_Outdated_Lib_File is deprecated");
+//This query is deprecated.
```

JavaScript / JavaScript_Low_Visibility / Client_Potential_Ad_Hoc_Ajax

Code changes

```
---
+++
@@ -1,6 +1 @@
-// Query Potential_Ad_Hoc_Ajax
-// The purpose of the query is to detect heuristic Ad Hoc Ajax vulnerability
-
-CxList methodsEval = Find_Methods().FindByShortName("eval");
-CxList xhrResponse = Find_XHR_Response();
-result = methodsEval.DataInfluencedBy(xhrResponse);
+//This query is deprecated.
```

JavaScript / JavaScript_Low_Visibility / Client_Potential_ReDoS_In_Match

Code changes

```
---
+++
@@ -1,31 +1 @@
-CxList evilStrings = Find_Evil_Strings();
-
-// Find all regex commands
-CxList regex = Find_Match();
-
-// Find regex commands that are influenced by evil strings
-CxList activeEvilRegexes = evilStrings.DataInfluencingOn(regex);
-
-regex -= regex.DataInfluencedBy(Find_Inputs());
-
-// Find relevant matches
-CxList result1 = activeEvilRegexes.DataInfluencingOn(regex);
-
-// Add static regexes (these do not influence their references, so needed in addition)
-CxList staticFields = regex.FindByType<FieldDecl>().FindByFieldAttributes(Modifiers.Static);
-CxList result2 = regex.GetByAncs(staticFields).DataInfluencedBy(evilStrings);
-
-// If only one type is found, no need to combine CxLists, because combining might lose the path
-if (result1.Count > 0 && result2.Count > 0)
-{
-   result = result1;
-   result.Add(result2);
-}
-else if (result1.Count > 0)
-{
-   result = result1;
-}
```

```
-else
-{
-   result = result2;
-}
+//This query is deprecated.
```

JavaScript / JavaScript_Low_Visibility / Client_Potential_ReDoS_In_Replace

Code changes

```
---
+++
@@ -1,19 +1 @@
-CxList evilStrings = Find_Evil_Strings_For_Replace();
-
-// Find all regex commands
-CxList regex = Find_Replace_Param();
-
-// Find regex commands that are influenced by evil strings
-CxList activeEvilRegexes = evilStrings.DataInfluencingOn(regex);
-
-regex -= regex.DataInfluencedBy(Find_Inputs());
-
-// Find relevant matches
-CxList result1 = activeEvilRegexes.DataInfluencingOn(regex);
-
-// Add static regexes (these do not influence their references, so needed in addition)
-CxList staticFields = regex.FindByType<FieldDecl>().FindByFieldAttributes(Modifiers.Static);
-CxList result2 = regex.GetByAncs(staticFields).DataInfluencedBy(evilStrings);
-
-// Add the results
-result.Add(result1, result2);
+//This query is deprecated.
```

JavaScript / JavaScript_Medium_Threat / Client_Cross_Frame_Scripting_Attack

Code changes

```
---
+++
@@ -1,2 +1 @@
-/* This query was integrated into query Client_Insufficient_ClickJacking_Protection */
-cxLog.WriteDebugMessage("The query Client_Cross_Frame_Scripting_Attack is deprecated");
+//This query is deprecated.
```

JavaScript / JavaScript_Medium_Threat / Client_Header_Manipulation

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query Client_Header_Manipulation is deprecated");
```

JavaScript / JavaScript_Medium_Threat / Client_HTML5_Store_Sensitive_data_In_Web_Storage

Code changes

```
---
+++
@@ -1,13 +1,29 @@
-//finds all sensitive data  saved into storage
-//this is done by finding the flow from any indicator of sensitive data to the first parameter of stoarage setItem
+// Finds all sensitive data saved in storage.
+// This is done by finding the flow from any indicator of sensitive data to the second argument of storage.setItem
 CxList personalInfo = Find_Personal_Info();
 CxList storageIn = Find_Storage_Inputs();
 CxList parameters = Find_Parameters();
-CxList irrelevant = All.NewCxList();
-irrelevant.Add(parameters, Find_Binarys());
-CxList valueToStore = (All - irrelevant).GetByAncs(parameters.GetParameters(storageIn));
-CxList target = (valueToStore * personalInfo);
-CxList sanitizer = Find_Encrypt();
+CxList relevant = All - All.NewCxList(parameters, Find_Binarys());

-result = valueToStore.InfluencedByAndNotSanitized(personalInfo - target, sanitizer);
-result.Add(target);
+CxList storageSetItem = storageIn.FindByShortName("setItem");
+CxList setItemKeyArgs = parameters.GetParameters(storageSetItem, 0);
+
+CxList storageArgs = parameters.GetParameters(storageIn);
+storageArgs -= setItemKeyArgs;
+
+CxList outputs = relevant.GetByAncs(storageArgs);
+CxList personalInfoValues = personalInfo * outputs;
+CxList inputs = personalInfo - personalInfoValues;
+CxList sanitizers = Find_Encrypt();
+
+// Gets the value of setItem when the personal info is in the first argument (the key of storage)
+CxList storageKeys = setItemKeyArgs * personalInfo;
+CxList relevantStorageIn = storageIn.FindByParameters(storageKeys);
+CxList storageValues = parameters.GetParameters(relevantStorageIn, 1);
+storageValues.Add(personalInfoValues.GetAncOfType<Param>() * storageArgs);
+
+CxList flows = outputs.InfluencedByAndNotSanitized(inputs, sanitizers);
+storageValues -= flows.GetLastNodesInPath().GetAncOfType<Param>(); // exclude duplicate results
+
+result = flows.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result.Add(storageValues);
```

JavaScript / JavaScript_Medium_Threat / Client_Reflected_File_Download

Code changes

```
---
+++
```

```
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query Client_Reflected_File_Download is deprecated");
```

JavaScript / JavaScript_SAPUI5 / SAPUI5_Hardcoded_UserId_In_Comments

Code changes

```
---
+++
@@ -1 +1 @@
-//This query is deprecated
+//This query is deprecated.
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / CSRF

Code changes

```
---
+++
@@ -32,11 +32,10 @@
 CxList expressAppUnkRef = unkRefs.FindAllReferences(expressAppDefs);


 /* Finds all occurrences of methods named .csrf() targetting objects returned by
-   require('express') or by require('csurf') invocations. */
-CxList expressAndCsurf = Find_Require("express", 2);
-expressAndCsurf.Add(Find_Require("csurf", 2));
+   require('express') invocations. */
+CxList express = Find_Require("express", 2);


-CxList csrfMethodInv = expressAndCsurf.GetMembersOfTarget().FindByShortName("csrf");
+CxList csrfMethodInv = express.GetMembersOfTarget().FindByShortName("csrf");


 /* Finds all occurrences of x.use(...), for <x> in expressAppUnkRef. */
 CxList expressAppUses = expressAppUnkRef.GetMembersOfTarget().FindByShortName("use").FindByType<MethodInvokeExpr>();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query HTTP_Response_Splitting is deprecated");
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Insecure_Direct_Object_References

Code changes

```
---
+++
@@ -1 +1 @@
-// This query has been deprecated.
+//This query is deprecated.
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / JSON_Hijacking

Code changes

```
---
+++
@@ -1,21 +1 @@
-// This query finds JSON strings returned to the server output
-// where there's no prefix or mitigation to avoid JS interpretation
-CxList methods = Find_Methods();
-CxList outputs = NodeJS_Find_Interactive_Outputs();
-CxList objs = Find_ObjectCreations();
-
-// Arrays are not safe
-CxList unsafeObjs = objs.FindByShortName("Array");
-objs -= unsafeObjs;
-
-// to Json converters
-CxList stringify = methods.FindByName("*JSON.stringify");
-stringify.Add(methods.FindByShortName("getJSON"));
-stringify -= stringify.InfluencedByAndNotSanitized(objs,unsafeObjs);
-stringify.Add(stringify.GetFathers().FindByType<Param>());
-
-// consider prefix/suffixed JSON as sanitized
-CxList sanitize = Find_Binarys().FindByShortName("+");
-
-result = stringify.InfluencingOn(outputs);
-result.Add((stringify * outputs) - sanitize);
+//This query is deprecated.
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -1,37 +1 @@
-CxList vulnSeqQueryMethods = Find_SensitiveData_Vuln_Sequelize_Methods();
-
-CxList nodeJsDbIn = NodeJS_Find_DB_IN();
-CxList storage = All.NewCxList(
-   nodeJsDbIn,
-   NodeJS_Find_Write(),
-   Find_Cloud_Outputs());
-
-// When dealing with Sequelize.query methods,
-// only vulnerable if flow passes trough 1st parameter and query options (2nd param) are set to either INSERT or UPDATE
-CxList sequelizeMethods = nodeJsDbIn.FindByMemberAccess("Sequelize.query");
-storage -= All.NewCxList(sequelizeMethods, storage.GetByAncs(sequelizeMethods));
-
-// If there are no sinks, there is no need to calculate personal info (sources)
-if (vulnSeqQueryMethods.Count == 0 && storage.Count == 0)
```

```
-   return All.NewCxList();
-
-CxList personalInfo = Find_Personal_Info() - Find_String_Literal();
-CxList sanitizers = NodeJS_Find_Encrypt();
-sanitizers.Add(All.GetParameters(Find_ObjectCreations().FindByShortName("Sequelize")),
-   Find_Integers());
-
-// Remove cases like: (password : hash), where hash was generated by encryption method.
-CxList rightSideValue = All.FindByAssignmentSide(CxList.AssignmentSide.Right).GetByAncs(personalInfo.GetFathers());
-CxList toRemove = sanitizers.DataInfluencingOn(rightSideValue).GetLastNodesInPath();
-toRemove = All.FindByAssignmentSide(CxList.AssignmentSide.Left).GetByAncs(toRemove.GetFathers());
-toRemove.Add(All.FindDefinition(toRemove), All_Passwords());
-personalInfo -= toRemove;
-
-CxList parameters = Find_Parameters().CxSelectDomProperty<Param>(_ => _.Value);
-CxList seqFirstParam = parameters.GetParameters(vulnSeqQueryMethods, 0);
-
-result = personalInfo.InfluencingOnAndNotSanitized(vulnSeqQueryMethods, sanitizers).IntersectWithNodes(seqFirstParam);
-
-result.Add(personalInfo.InfluencingOnAndNotSanitized(storage, sanitizers)
-   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
-   .ReduceFlowByPragma());
+//This query is deprecated.
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Password_Weak_Encryption

Code changes

```
---
+++
@@ -20,9 +20,8 @@


 CxList sqlQuery = db.FindByShortName("query", false);
 db -= sqlQuery;
-CxList sqlQueryParams = All.GetParameters(sqlQuery);
-CxList validFlowResults = candidates.DataInfluencingOn(sqlQuery, CxList.InfluenceAlgorithmCalculation.NewAlgorithm)
-   .IntersectWithNodes(sqlQueryParams);
+CxList sqlQueryParams = Find_Expressions().GetParameters(sqlQuery);


-result.Add(candidates.DataInfluencingOn(db, CxList.InfluenceAlgorithmCalculation.NewAlgorithm),
-   validFlowResults);
+db.Add(sqlQueryParams);
+result = candidates.DataInfluencingOn(db, CxList.InfluenceAlgorithmCalculation.NewAlgorithm)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Potentially_Vulnerable_To_CSRF

Code changes

```
---
+++
@@ -1,90 +1 @@
```

```
-/// <summary>
-/// This query searches for app and http.server that are not configured with the express.csrf or csurf protection for XSRF.
-/// </summary>
-CxList declarators = Find_Declarators();
-CxList unknown = Find_UnknownReference();
-CxList vulnerable = All.NewCxList();
-CxList vulnerableOld = All.NewCxList();
-CxList vulnerableNew = All.NewCxList();
-CxList methods = Find_Methods();
-
-CxList parameters = Find_Parameters();
-
-// Find references of express objects/function: var express = require("express");
-CxList express = Find_Require("express", 1);
-CxList expressMembers = (unknown * express).GetMembersOfTarget();
-
-CxList expressCSRF = expressMembers.FindByMemberAccess("*.csrf");
-
-// Find referemces of app objects: var express = require("express"); var app = express();
-CxList app = Find_Require("express", 2);
-CxList appRef = unknown * app;
-// Find var csrf = require("csurf"); and var csrfProtection = csrf();
-CxList csrfProtection = Find_Require("csurf", 2);
-// Find app.use(express.csrf()); and var csrf = require("csurf"); app.use(csrf);
-CxList appUse = appRef.GetMembersOfTarget().FindByMemberAccess("*.use");
-
-CxList useExpressCSRF = expressCSRF.GetParameters(appUse);
-CxList appUseCSRF = appUse.FindByParameters(useExpressCSRF).GetTargetOfMembers();
-appUseCSRF.Add(appUse.FindByParameters(useExpressCSRF.GetFathers() * parameters).GetTargetOfMembers());
-CxList safeAppUse = appUse.FindByParameters(csrfProtection);
-safeAppUse.Add(appUse.FindByParameters(csrfProtection.GetFathers() * parameters));
-appUseCSRF.Add(safeAppUse.GetTargetOfMembers());
-vulnerableOld.Add(All.FindDefinition((appRef - appRef.FindAllReferences(appUseCSRF)) - express));
-
-// Find cases where the router is initialized before the csrf protection
-//   which can prevents validating the document_csrf token correctly
-CxList expressRouter = expressMembers.FindByMemberAccess("*.Router");
-CxList useExpressRouter = expressRouter.GetParameters(appUse);
-useExpressRouter.Add(appRef.GetMembersOfTarget().FindByMemberAccess("*.router"));
-useExpressRouter.Add(unknown.FindAllReferences(expressRouter.GetAssignee()));
-CxList appUseRouter = appUse.FindByParameters(useExpressRouter).GetTargetOfMembers();
-appUseRouter.Add(appUse.FindByParameters(useExpressRouter.GetFathers() * parameters).GetTargetOfMembers());
-CxList influenced = appUseRouter.DataInfluencingOn(appUseCSRF);
-// Remove router from the results (we are interessed in its app instead)
-vulnerableOld -= declarators.FindDefinition(useExpressRouter);
-vulnerableOld.Add(declarators.FindDefinition(influenced));
-
-// Find app methods recieving the csurf object as a parameter, so it could be used to craate and validate tokens
```

```
-string[] HTTPMethods = {"all", "checkout", "connect", "copy", "delete", "get", "head", "lock", "merge", "mkactivity",
-   "mkcol", "move", "m - search", "notify", "options", "patch", "post", "propfind", "proppatch", "purge",
-   "put", "report", "search", "subscribe", "trace", "unlock", "unsubscribe"};
-CxList appsMethods = appRef.GetMembersOfTarget().FindByShortNames(new List<string>(HTTPMethods));
-CxList safeMethods = appsMethods.FindByParameters(csrfProtection);
-safeMethods.Add(appsMethods.FindByParameters(csrfProtection.GetFathers() * parameters));
-// If all methods of the app get the csurf object as parameter - the app is using csurf.
-CxList safeAppRef = appRef.FindAllReferences(appUseCSRF);
-safeAppRef.Add((appsMethods - safeMethods).GetTargetOfMembers());
-safeAppRef.Add(express);
-CxList vul = appRef - safeAppRef;
-
-vulnerableNew.Add(All.FindDefinition(vul));
-
-// Find router usages, and and add the respective app
-CxList useWithRouters = appUse.FindByParameters(unknown.FindAllReferences(vulnerableNew));
-if (useWithRouters.Count > 0) {
-   vulnerableNew.Add(All.FindDefinition(useWithRouters.GetTargetOfMembers()));
-}
-
-// Return only app(s) that do not use csrf or csurf
-vulnerable.Add(vulnerableOld * vulnerableNew);
-result.Add(vulnerable);
-
-// Find http(s) server that is created using an Express app without csrf/csurf protection
-CxList vulnerableApp = vulnerable.GetLastNodesInPath();
-CxList vulnerableAppRef = appRef.FindAllReferences(vulnerableApp);
-
-CxList http = Find_Require("http", 1);
-http.Add(Find_Require("https", 1));
-CxList httpMembers = http.GetMembersOfTarget();
-List<string> httpMethods = new List<string>{"Server", "createServer"};
-CxList server = httpMembers.FindByShortNames(httpMethods);
-
-//Find Server or createServer in var x = require("http").[Server, createServer]
-CxList httpRequireMembers = methods.GetMembersOfTarget();
-server.Add(httpRequireMembers.FindByShortNames(httpMethods));
-
-CxList serverUsingExpress = server.FindByParameters(appRef - vulnerableAppRef);
-// Find: http.Server() and http.CreateServer(app) where app is not using csrf/csurf
-CxList vulnerableServer = server - serverUsingExpress;
-result.Add(vulnerableServer);
+//This query is deprecated.
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Reflected_XSS

Code changes

```
---
+++
```

```
@@ -4,31 +4,64 @@

 CxList fieldDecls = Find_FieldDecls();

 CxList declarators = Find_Declarators();

 CxList paramDecls = Find_ParamDecl();

-CxList outputs = NodeJS_Find_Interactive_Outputs();

-outputs.Add(Find_Outputs_XSS());

+CxList indexerRefs = Find_IndexerRefs();

+CxList catchs = All.NewCxList(Find_Catch(),    methods.FindByShortName("catch", false));

+CxList decls = All.NewCxList(declarators, paramDecls);


-CxList react = Find_Require("React").GetMembersOfTarget();

-CxList createElement = react.FindByShortName("createElement");

+//==== Inputs ====

+CxList inputs = All.NewCxList(

+   NodeJS_Find_Interactive_Inputs(),

+   AngularJS_Find_Inputs(),

+   Find_Cloud_Interactive_Inputs());


-CxList jsonParse = unknRefs.FindAllReferences(methods.FindByMemberAccess("JSON.parse").GetAssignee());

+//Removing duplicate inputs when dealing with references of same input in and outside lambda

+CxList returnsInfByInput = Find_ReturnStmt().GetByAncs(Find_LambdaExpr()).DataInfluencedBy(inputs).GetLastNodesInPath();

+foreach(CxList ret in returnsInfByInput){

+   CxList relatedLambda = ret.GetAncOfType<LambdaExpr>();

+   CxList inputsNearLambda = inputs.Filter(x => x.Line == relatedLambda.GetDOMPropertiesOfFirst().Line);

+   CxList inputRefs = indexerRefs.FindAllReferences(inputsNearLambda);

+   CxList inputInLambda = inputRefs.FindInScope(relatedLambda, ret);

+   CxList relevantInputInLambda = inputInLambda - inputInLambda.FindByAssignmentSide(CxList.AssignmentSide.Left);

+   if(relevantInputInLambda.Count > 0){

+       CxList potentialRemove = inputsNearLambda.FindAllReferences(relevantInputInLambda);

+       inputs -= potentialRemove;

+   }

+}


-//if a user parses JSON to send to a React element we consider that data an output

-outputs.Add(jsonParse.GetByAncs(createElement));

+//================

+

+//==== Sanitizers ====

+CxList sanitize = All.NewCxList(

+   NodeJS_Find_XSS_Sanitize(),

+   Find_XSS_Sanitize());

+

+// Angular IO default sanitizers

+CxList angularDefaultSanitizers = methods.FindByShortName("CxDefaultSanitizer");

+sanitize.Add(angularDefaultSanitizers - angularDefaultSanitizers.DataInfluencedBy(Find_Angular_Sanitizers_Bypass()));


 CxList res = unknRefs.FindByShortNames(new string[] {"res", "response"});

 CxList renders = res.GetMembersOfTarget().FindByShortName("render");
```

```
 CxList rendersSanitizedParams = parameters.GetParameters(renders, 0);


-CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), AngularJS_Find_Inputs(), Find_Cloud_Interactive_Inputs());
+sanitize.Add(
+   rendersSanitizedParams,
+   All.GetByAncs(rendersSanitizedParams),
+   decls.GetByAncs(catchs).FindByShortNames(new string [] {"e", "err", "error"}, false),
+   Find_ExpressValidator_Sanitized());


-CxList sanitize = NodeJS_Find_XSS_Sanitize();
-sanitize.Add(Find_XSS_Sanitize());
-// Angular IO default sanitizers
-CxList angularDefaultSanitizers = methods.FindByShortName("CxDefaultSanitizer");
-sanitize.Add(angularDefaultSanitizers - angularDefaultSanitizers.DataInfluencedBy(Find_Angular_Sanitizers_Bypass()));
+//================


-sanitize.Add(rendersSanitizedParams);
-sanitize.Add(All.GetByAncs(rendersSanitizedParams));
+//==== Outputs ====
+
+CxList outputs = All.NewCxList(
+   NodeJS_Find_Interactive_Outputs(),
+   Find_Outputs_XSS());
+
+CxList react = Find_Require("React").GetMembersOfTarget();
+CxList createElement = react.FindByShortName("createElement");
+CxList jsonParse = unknRefs.FindAllReferences(methods.FindByMemberAccess("JSON.parse").GetAssignee());
+//if a user parses JSON to send to a React element we consider that data an output
+outputs.Add(jsonParse.GetByAncs(createElement));


 // remove results from React.createElement with children field as the props are not vulnerable to XSS
 CxList elementFields = fieldDecls.GetByAncs(createElement);
@@ -44,16 +77,9 @@
 {
    outputs -= NodeJS_Find_Swig_Interactive_Outputs();
 }
+//================


-CxList decls = All.NewCxList(declarators);
-decls.Add(paramDecls);
-
-CxList catchs = Find_Catch();
-catchs.Add(methods.FindByShortName("catch", false));
-
-sanitize.Add(decls.GetByAncs(catchs).FindByShortNames(new string [] {"e", "err", "error"}, false));
-
-sanitize.Add(Find_ExpressValidator_Sanitized());
+//==== Results ====
```

```
 result = outputs.InfluencedByAndNotSanitized(inputs, sanitize);


@@ -67,4 +93,4 @@


 result.Add(inputs * outputs - sanitize);


-result = result.ReduceFlow(Checkmarx.DataCollections.CxQueryProvidersInterface.CxList.ReduceFlowType.ReduceBigFlow);
+result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Security_Misconfiguration

Code changes

```
---
+++
@@ -1 +1 @@
-//This query has ben deprecated and replaced by query Insecure_Storage_of_Sensitive_Data
+//This query is deprecated.
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -26,7 +26,7 @@
 // Remove string framework templates from hardcoded passwords
 // e.g. var password_template = "<input type='password' value='password'>"
-stringLiterals -= stringLiterals.FindByShortName("<*").FindByShortName("*>");
+stringLiterals -= stringLiterals.FindByShortNames("<*", "*>", "* *", "*.*");


 // remove hardcoded strings in complex associative array with array creation
 CxList arrayCreateInAssociativeArrays = Find_ArrayCreateExpr().GetByAncs(Find_AssociativeArrayExpr());
@@ -109,6 +109,7 @@
    }
 }


+stringLiterals = stringLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);
 // Result
 result.Add(
    stringLiterals.FindByFathers(passParamDecl),
```

JavaScript / JavasScript_Visualforce_Remoting / VF_Remoting_Client_Potential_CSRF

Code changes

```
---
+++
@@ -1,2 +1 @@
 //This query is deprecated.
-cxLog.WriteDebugMessage("The query VF_Remoting_Client_Potential_CSRF is deprecated");
```

Kotlin / Kotlin_Android / Accessible_Content_Provider

Code changes

```
---

+++

@@ -1 +1 @@

-// This query is deprecated in favour of Exported_Content_Provider_Without_Protective_Permissions

+//This query is deprecated.
```

Kotlin / Kotlin_Android / Exported_Service_Without_Permissions

Code changes

```
---

+++

@@ -1 +1 @@

-// This query is deprecated in favour of Exported_Service_Without_Protective_Permissions

+//This query is deprecated.
```

Kotlin / Kotlin_Android / Improper_Certificate_Validation

Code changes

```
---

+++

@@ -25,7 +25,7 @@

 // overloads of checkClientTrusted & checkServerTrusted methods

 CxList trustManagerCheckMethods = methodDecls

    .FindByShortNames(new List<string> {"checkClientTrusted", "checkServerTrusted"});

-CxList vulnerableMethods = emptyReturnStmts.GetAncOfType(typeof(MethodDecl)) * trustManagerCheckMethods;

+CxList vulnerableMethods = emptyReturnStmts.GetAncOfType<MethodDecl>() * trustManagerCheckMethods;

 //Ensure checkClient & checkServer methods return and nothing else.

 foreach(CxList vulnerableMethod in vulnerableMethods){

    MethodDecl md = vulnerableMethod.TryGetCSharpGraph<MethodDecl>();

@@ -36,13 +36,12 @@

 }

 CxList vulnerableTrsutManagerImpl = trustManagerImpl.GetClass(vulnerableMethods);

 CxList vulnerableTrustManagerCreation = typeReferences.FindAllReferences(vulnerableTrsutManagerImpl)

-   .GetAncOfType(typeof(ObjectCreateExpr));

+   .GetAncOfType<ObjectCreateExpr>();


 CxList flowSinks = All.NewCxList();

-flowSinks.Add(vulnerableTrustManagerCreation);

-

-flowSinks.Add(vulnerableTrustManagerCreation.GetAncOfType(typeof(AssignExpr)).GetAssignee());

-flowSinks.Add(vulnerableTrustManagerCreation.GetAncOfType(typeof(Declarator)));

+flowSinks.Add(vulnerableTrustManagerCreation,

+   vulnerableTrustManagerCreation.GetAncOfType<AssignExpr>().GetAssignee(),

+   vulnerableTrustManagerCreation.GetAncOfType<Declarator>());


 CxList sslContextsInits = methods.FindByMemberAccess("SSLContext.init");


@@ -56,7 +55,7 @@
```

```
 CxList returnTrueStatements = trueAbsValue.GetFathers() * returnStmts;
 CxList hostNameVerifierMethods = methodDecls.FindByShortName("verify");
-CxList vulnerableHostVerMethods = returnTrueStatements.GetAncOfType(typeof(MethodDecl)) * hostNameVerifierMethods;
+CxList vulnerableHostVerMethods = returnTrueStatements.GetAncOfType<MethodDecl>() * hostNameVerifierMethods;
 //Ensure verification methods returns true and nothing else.
 foreach(CxList vulnerableMethod in vulnerableHostVerMethods){
    if(returnStmts.GetByAncs(vulnerableMethod).Count > 1)
@@ -64,12 +63,11 @@
 }
 CxList vulnerableHostnameVerImpl = hostNameVerifierImpl.GetClass(vulnerableHostVerMethods);
 CxList vulnerableHostnameVerCreation = typeReferences.FindAllReferences(vulnerableHostnameVerImpl)
-   .GetAncOfType(typeof(ObjectCreateExpr));
+   .GetAncOfType<ObjectCreateExpr>();

-flowSinks = All.NewCxList();
-flowSinks.Add(vulnerableHostnameVerCreation);
-flowSinks.Add(vulnerableHostnameVerCreation.GetAncOfType(typeof(AssignExpr)).GetAssignee());
-flowSinks.Add(vulnerableHostnameVerCreation.GetAncOfType(typeof(Declarator)));
+flowSinks = All.NewCxList(vulnerableHostnameVerCreation,
+   vulnerableHostnameVerCreation.GetAncOfType<AssignExpr>().GetAssignee(),
+   vulnerableHostnameVerCreation.GetAncOfType<Declarator>());

 CxList httpURLConnDefaultHost = methods.FindByMemberAccess("HttpsURLConnection.setDefaultHostnameVerifier");
 CxList setDefaultHNResults = flowSinks.DataInfluencingOn(httpURLConnDefaultHost);
```

Kotlin / Kotlin_Low_Visibility / Use_of_Hardcoded_Password

Code changes

```
---
+++
@@ -6,7 +6,7 @@
  */

 CxList passwordVariables = Find_All_Passwords();
-CxList stringLiterals = Find_Strings();
+CxList stringLiterals = Find_Strings().FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);

 CxList notPasswordStrings = All.NewCxList();
 notPasswordStrings.Add(
```

Lua / Lua_Medium_Threat / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -1,7 +1 @@
-CxList sensitiveInfo = Find_Sensitive_Information();
-
-CxList sanitizers = Find_Encryption();
```

```
-
-CxList outputs = All.NewCxList(Find_Stored_Outputs_Files(), Find_Stored_Outputs_DB(), Find_Stored_Outputs_Caches());
-
-result = outputs.InfluencedByAndNotSanitized(sensitiveInfo, sanitizers);
+//This query is deprecated.
```

Objc / ObjectiveC_Best_Coding_Practice / Dead_Code

Code changes

```
---
+++
@@ -7,12 +7,12 @@
    try
    {
        CxList cond = t.GetFathers();
-       if (cond.FindByType(typeof(IfStmt)).Count > 0)
+       if (cond.FindByType<IfStmt>().Count > 0)
        {
            IfStmt ifStmt = cond.TryGetCSharpGraph<IfStmt>();
            result.Add(ifStmt.TrueStatements.NodeId, ifStmt.TrueStatements);
        }
-       else if (cond.FindByType(typeof(IterationStmt)).Count > 0)
+       else if (cond.FindByType<IterationStmt>().Count > 0)
        {
            IterationStmt iter = cond.TryGetCSharpGraph<IterationStmt>();
            result.Add(iter.Statements.NodeId, iter.Statements);
```

Objc / ObjectiveC_High_Risk / Deserialization_of_Untrusted_Data

Code changes

```
---
+++
@@ -1,3 +1 @@
-CxList inputs = Find_Inputs();
-CxList deserializers = Find_Unsafe_Deserializers();
-result.Add(inputs.InfluencingOn(deserializers));
+//This query is deprecated.
```

Objc / ObjectiveC_High_Risk / Universal_XSS

Code changes

```
---
+++
@@ -1,5 +1 @@
-CxList inputs = Find_Interactive_Inputs() - Find_Interactive_Inputs_User();
-CxList outputs = Find_XSS_Outputs_Webview();
-CxList sanitized = Find_XSS_Sanitize();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitized);
+//This query is deprecated.
```

Objc / ObjectiveC_Low_Visibility / Heap_Inspection

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList methods = Find_Methods();
-
-List<string> methodsNames = new List<string> {"realloc","fork","vfork"};
-CxList badForSecurity = methods.FindByShortNames(methodsNames);
-
-CxList personal = Find_Personal_Info();
-
-result = badForSecurity.DataInfluencedBy(personal);
+//This query is deprecated.
```

Objc / ObjectiveC_Low_Visibility / Memory_Leak

Code changes

```
---
+++
@@ -25,16 +25,16 @@
 CxList inReturnStmt = All.GetByAncs(ReturnStmt);
 CxList inParam = All.GetByAncs(Param);
 CxList inParamDecl = All.GetByAncs(ParamDecl);
-inParamDecl -= inParamDecl.FindByType(typeof(TypeRef));
+inParamDecl -= inParamDecl.FindByType<TypeRef>();

 // All the x's in the case of "x = y;"
 CxList unknownRefLeftSide = UnknownReference.GetByAncs(All.FindByAssignmentSide(CxList.AssignmentSide.Left));

 // Find allocated variables that are on the left assignment side side of "x = malloc(..)"
-CxList allAllocatedVariables = unknownRefLeftSide.GetByAncs(memAllocations.GetAncOfType(typeof(AssignExpr)));
+CxList allAllocatedVariables = unknownRefLeftSide.GetByAncs(memAllocations.GetAncOfType<AssignExpr>());

 // Add objects that were allocated at their declaration: "int *x = malloc(..)"
-allAllocatedVariables.Add(memAllocations.GetAncOfType(typeof(Declarator)));
+allAllocatedVariables.Add(memAllocations.GetAncOfType<Declarator>());

 // See if there are any deallocations for every allocation parameter
 CxList deallocParams = All.GetParameters(dealloc);
@@ -63,7 +63,7 @@
 // Between all the variables that were never deallocated, leave only results that are not assigned to a parameter.
 // This parameter is most likely passed by ref.
 CxList inParamRefs = All.FindAllReferences(inParamDecl);
-CxList assignOfUnknown = (unknownRefLeftSide - onlyAllocatedVariables).GetAncOfType(typeof(AssignExpr));
+CxList assignOfUnknown = (unknownRefLeftSide - onlyAllocatedVariables).GetAncOfType<AssignExpr>();
 CxList allocatedVariablesRef = All.FindAllReferences(onlyAllocatedVariables);

 foreach (CxList onlyAllocatedVariable in onlyAllocatedVariables)
```

```
@@ -85,15 +85,15 @@

 // Find if statements starting with "not" and containing null. Both are relevant to sanitize Memory Leak
 CxList conditions = Find_Condition();
-CxList negativeIf = conditions.FindByShortName("Not").GetFathers().FindByType(typeof(IfStmt));
+CxList negativeIf = conditions.FindByShortName("Not").GetFathers().FindByType<IfStmt>();
 CxList nullLiterals = Find_Null_Literals();
-CxList nullIf = nullLiterals.GetByAncs(conditions).GetAncOfType(typeof(IfStmt));
+CxList nullIf = nullLiterals.GetByAncs(conditions).GetAncOfType<IfStmt>();

 CxList safeIfStatement = All.NewCxList();
 safeIfStatement.Add(nullIf, negativeIf);

 // Remove return statement the contain "this", because returning "this" sanitizing Memory Leak
-CxList thisReturn = ThisRef.GetAncOfType(typeof(ReturnStmt));
+CxList thisReturn = ThisRef.GetAncOfType<ReturnStmt>();
 ReturnStmt -= thisReturn;

 // Performance optimization
@@ -118,16 +118,16 @@
 // Look at all the memory allocations and return the ones that are not deallocated
 foreach (CxList allocation in memAllocations)
 {
-   CxList method = allocation.GetAncOfType(typeof(MethodDecl));
+   CxList method = allocation.GetAncOfType<MethodDecl>();

    // Get the variables with memory allocations in an assign expression
-   CxList allocationInAssign = inAssign.FindByFathers(allocation.GetAncOfType(typeof(AssignExpr)));
+   CxList allocationInAssign = inAssign.FindByFathers(allocation.GetAncOfType<AssignExpr>());
    allocationInAssign = allocationInAssign.FindByAssignmentSide(CxList.AssignmentSide.Left);
    // We need the following line in case of a pointer: "*out = malloc(...)"
    // where only the pointer's assignment side is left
    allocationInAssign = inAssign.GetByAncs(allocationInAssign);
    // Get the variables with memory allocation in a declaration
-   CxList decl = allocation.GetAncOfType(typeof(Declarator));
+   CxList decl = allocation.GetAncOfType<Declarator>();
    CxList allocatedVariables = All.NewCxList();
    allocatedVariables.Add(decl, allocationInAssign);

@@ -144,10 +144,10 @@
    }

    // See if the allocated variable is assigned to some other variable, if so - false alarm
-   CxList leftParam = unknownRefLeftSide.GetByAncs(method).GetAncOfType(typeof(AssignExpr));
+   CxList leftParam = unknownRefLeftSide.GetByAncs(method).GetAncOfType<AssignExpr>();
    CxList relevantParams = (allocatedVariables * inParamRefs).FindAllReferences(inParamDecl.GetByAncs(method));
    relevantParams = relevantParams.GetByAncs(leftParam);
-   if (allocation.GetByAncs(relevantParams.GetAncOfType(typeof(AssignExpr))).Count > 0)
```

```
+   if (allocation.GetByAncs(relevantParams.GetAncOfType<AssignExpr>()).Count > 0)
    {
        continue;
    }
@@ -180,11 +180,11 @@

    // Remove the return statements that contain the allocated variables
    CxList allocatedVariablesInReturn = inReturnStmt.GetByAncs(returnInMethod).FindAllReferences(allocatedVariables);
-   returnInMethod -= allocatedVariablesInReturn.GetAncOfType(typeof(ReturnStmt));
+   returnInMethod -= allocatedVariablesInReturn.GetAncOfType<ReturnStmt>();

    // Remove the if statements that contain the allocated variables
    CxList allocatedVariablesInIf = allocatedVariablesInCondition.FindAllReferences(allocatedVariables);
-   CxList ifWithTarget = allocatedVariablesInIf.GetAncOfType(typeof(IfStmt));
+   CxList ifWithTarget = allocatedVariablesInIf.GetAncOfType<IfStmt>();
    returnInMethod -= returnInMethod.GetByAncs(ifWithTarget * safeIfStatement);

    // All deallocated variables in deallocation functions
@@ -196,13 +196,13 @@
    {
        // retStmt is filled with the statement collection of this return statement,
        // and the containing statement collection
-       CxList retStmt = r.GetAncOfType(typeof(StatementCollection));
-       retStmt.Add(retStmt.GetFathers().GetAncOfType(typeof(StatementCollection)));
+       CxList retStmt = r.GetAncOfType<StatementCollection>();
+       retStmt.Add(retStmt.GetFathers().GetAncOfType<StatementCollection>());
        // See if we have a deallocation in this statement collection (or its predecessor)
        CxList correctDeallocation = deallocatedVariables.GetByAncs(retStmt);
        // leave only the dealloc statements that are in the same level as the return statement.
        // dealloc statements inside an interior statements are not needed
-       correctDeallocation = correctDeallocation.GetByAncs(retStmt * correctDeallocation.GetAncOfType(typeof(StatementCollection)));
+       correctDeallocation = correctDeallocation.GetByAncs(retStmt * correctDeallocation.GetAncOfType<StatementCollection>());
        // No deallocation was done in this block
        if (correctDeallocation.Count == 0)
        {
```

Objc / ObjectiveC_Low_Visibility / Potential_ReDoS

Code changes

```
---
+++
@@ -1,4 +1 @@
-CxList evilStringsInputs = All.NewCxList();
-evilStringsInputs.Add(Find_Evil_Strings(), Find_Inputs());
-
-result = Find_ReDoS(evilStringsInputs, true);
+//This query is deprecated.
```

Objc / ObjectiveC_Low_Visibility / Use_of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,24 +1,22 @@
+CxList methods = Find_Methods();
+CxList arrays = Find_ArrayInitializer();
 CxList pswInStr = Find_Password_Strings();
 CxList psw = Find_Passwords() - pswInStr;
 psw -= psw.FindByShortName(@"""*"); // Remove Param objects that are of strings

 CxList pswInLeftSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList pswInLeftSideDecl = pswInLeftSide.FindByType(typeof(Declarator));
+CxList pswInLeftSideDecl = pswInLeftSide.FindByType<Declarator>();

 CxList allUnknownRef = Find_UnknownReference();
 CxList allParams = Find_Param();
 CxList arrayIndexes = Find_IndexerRefs();

-CxList strLiterals = Find_Strings() - Find_Empty_Strings();
+CxList strLiterals = Find_Strings().FilterByDomProperty<StringLiteral>(x => x.Text.Length > 4);// Length > 4 due to "@"
+strLiterals -= Find_Empty_Strings();

 //when the hardcoded string includes a space or dot we believe
 //it is not a password string
-
-CxList stringToRemove = All.NewCxList();
-stringToRemove.Add(strLiterals.FindByName("* *"), strLiterals.FindByName("*.*"),
-                   strLiterals.FindByName("*/*"), strLiterals.FindByName("*\\*"));
-
-strLiterals -= stringToRemove;
+strLiterals -= strLiterals.FindByNames("* *", "*.*", "*/*", "*\\*");

 // remove also strings that are used as indexers of an array
 strLiterals -= strLiterals.GetByAncs(arrayIndexes);
@@ -28,19 +26,15 @@
 // Password in declaration
 CxList stringsOfPass = strLiterals.GetByAncs(psw);
 CxList PasswordInDecl = pswInLeftSideDecl.FindByInitialization(litInRightSide);
-PasswordInDecl.Add(stringsOfPass.GetAncOfType(typeof(Declarator)));
-PasswordInDecl.Add(stringsOfPass.GetAncOfType(typeof(UnknownReference)));
+PasswordInDecl.Add(stringsOfPass.GetAncOfType<Declarator>(), stringsOfPass.GetAncOfType<UnknownReference>());

 // remove strings that are indexes in dictionaries
 CxList forKey = All.FindByParameters(stringsOfPass.GetByAncs(PasswordInDecl)).FindByShortName("*ForKey:");
-PasswordInDecl -= stringsOfPass.GetByAncs(forKey).GetAncOfType(typeof(Declarator));
+PasswordInDecl -= stringsOfPass.GetByAncs(forKey).GetAncOfType<Declarator>();
```

```
-CxList methods = Find_Methods();
+CxList strcmp = methods.FindByShortNames(new string[]{"strcmp","strncmp","bcmp"});

-List<string> strcmpMethods = new List<string>{"strcmp","strncmp","bcmp"};
-CxList strcmp = methods.FindByShortNames(strcmpMethods);
-
-CxList objCEqual = methods.FindByShortNames(new List<string>{"isEqualToString:","isEqual:"});
+CxList objCEqual = methods.FindByShortNames(new string[]{"isEqualToString:","isEqual:"});

 CxList allParamsStrLiterals = All.NewCxList();
 allParamsStrLiterals.Add(allParams, strLiterals);
@@ -50,14 +44,13 @@
 //get fathers of "isEqualToString:" then find ancs of type unknown reference
 CxList strcmpParam2 = All.GetParameters(strcmp, 1);

-CxList allParamsStr = All.NewCxList();
-allParamsStr.Add(allParams, strLiterals);
+CxList allParamsStr = All.NewCxList(allParams, strLiterals);

 CxList objCParam1 = allParamsStr.GetParameters(objCEqual, 0);
 CxList objCParam2 = allUnknownRef.GetByAncs(objCEqual.GetFathers()) * psw; //-2-

 //results of type [@"hello" isEqualToString:password]
-CxList isEq = strLiterals.GetAncOfType(typeof(MethodInvokeExpr)) * objCEqual;
+CxList isEq = strLiterals.GetAncOfType<MethodInvokeExpr>() * objCEqual;
 CxList passParams = psw.GetParameters(isEq, 0);
 CxList passInIsEq = isEq.FindByParameters(passParams);

@@ -69,16 +62,12 @@
 //strcmp("myPass", password)
 //strcnmp("myPass", password, length)
 //bcmp("myPass", password, cnt)
-
-
-CxList strcmpParam2Psw = All.NewCxList();
-strcmpParam2Psw.Add(strcmpParam2, psw);
+CxList strcmpParam2Psw = All.NewCxList(strcmpParam2, psw);

 CxList parametersFathers = All.FindByParameters(strcmpParam2Psw);
 parametersFathers.Add(objCEqual.GetByAncs((objCParam2 * psw).GetFathers()));

-CxList strcmpParam1ObjCParam = All.NewCxList();
-strcmpParam1ObjCParam.Add(strcmpParam1, objCParam1);
+CxList strcmpParam1ObjCParam = All.NewCxList(strcmpParam1, objCParam1);

 CxList parametersLiterals = All.FindByParameters(strcmpParam1ObjCParam * strLiterals);

@@ -92,15 +81,13 @@
```

```
 equalsPassword.Add(psw.GetByAncs(eq));


 // Password in simple assignment
-CxList assignPassword = pswInLeftSide.GetAncOfType(typeof(AssignExpr));
+CxList assignPassword = pswInLeftSide.GetAncOfType<AssignExpr>();
 assignPassword = litInRightSide.GetByAncs(assignPassword);
-


 //Password in Dictionary/array initialization
 CxList passInDictionary = All.NewCxList();
-CxList arrays = Find_ArrayInitializer();
 CxList pswInArrayInit = pswInStr.GetByAncs(arrays);
-CxList arrayInitFatherOfPsw = pswInStr.GetAncOfType(typeof(ArrayInitializer));
+CxList arrayInitFatherOfPsw = pswInStr.GetAncOfType<ArrayInitializer>();
 //Look for dictionaries that have a literal password as Key
 //If the value of that key is a string literal, we are facing a case of hardcoded password
 //Example: let x = [ "Password" : "test" ]
@@ -124,7 +111,7 @@
 //Password in dictionary/array access
 //Example: x["password"] = "test"
 //         y["password"]["user"] = "pswd"
-CxList strLiteralsInAssign = strLiterals.GetAncOfType(typeof(AssignExpr));
+CxList strLiteralsInAssign = strLiterals.GetAncOfType<AssignExpr>();
 CxList arraysAssignedToLiterals = arrayIndexes.FindByFathers(strLiteralsInAssign);
 passInDictionary.Add(pswInStr.GetByAncs(arraysAssignedToLiterals));
```

Objc / ObjectiveC_Medium_Threat / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -1,10 +1 @@
-// Missing_Encryption_Sensitive_Data
-// --------------------------------
-// The purpose of the query is as to find applications that allow the following:
-//     Use non encrypted files while writing protected data.
-
-// All insecure write, not including Log Write, because as insecure as Log is, we won't be encrypting the log output,
-// so any data written to Log is found in a different query (not Encryption missing, but data Leakage).
-CxList notRelevantOutput = Find_Log_Outputs();
-
-result = Find_Insecure_Data_Storage(Find_Personal_Info(), notRelevantOutput);
+//This query is deprecated.
```

Objc / ObjectiveC_Medium_Threat / Parameter_Tampering

Code changes

```
---
+++
```

```
@@ -1,15 +1 @@
-////////////////////////////////////////////////////////////////
-//                Parameter_Tampering
-// This query checks that each input from the user,
-// that is used to get information from the database,
-// is sanitized by:
-// 1.In the "Select","Insert" and "Update" statements there is also a "And" part in the "Where".
-// 2.The input is being checked by an if condition somewhere in the program.
-////////////////////////////////////////////////////////////////
-
-CxList input = Find_Interactive_Inputs();
-CxList db = Find_DB_For_Parameter_Tampering();
-CxList sanitize = Find_Parameter_Tampering_Sanitize();
-
-result = db.InfluencedByAndNotSanitized(input, sanitize);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

Objc / ObjectiveC_Medium_Threat / Side_Channel_Data_Leakage

Code changes

```
---
+++
@@ -1,45 +1 @@
-// Side Channel Data Leakage (a part of Privacy_Violation)
-// ----------------------
-// The following cases will be classified as Privacy Violation
-//
-// 1) Personal information kept in log file
-// 2) Keystroke logging of sensitive information
-// 3) Enabled iOS screenshot capture for sensitive information
-//
-
-CxList sensitive = Find_UI_Widgets_With_Sensitive_Data();
-
-// 1) Personal information kept in log file
-CxList personalInfo = Find_Personal_Info();
-CxList logOutput = Find_Log_Outputs();
-CxList sanitize = Find_General_Sanitize();
-
-CxList inputs = Find_Inputs();
-inputs.Add(Find_DB_Out());
-
-CxList personalInfoInputs = personalInfo * inputs;
-
-personalInfo = personalInfo.DataInfluencedBy(inputs).GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-personalInfo.Add(personalInfoInputs);
-
-CxList personalInfoResult = personalInfo.InfluencingOnAndNotSanitized(logOutput, sanitize);
```

```
-
-CxList interactiveOutputsLogOutput = Find_Interactive_Outputs();
-interactiveOutputsLogOutput.Add(logOutput);
-
-result.Add(personalInfoResult.InfluencingOnAndNotSanitized(interactiveOutputsLogOutput, sanitize));
-result.Add(personalInfoResult.InfluencedByAndNotSanitized(Find_Interactive_Inputs_User(), sanitize));
-
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
-
-// 2) Keystroke logging of sensitive information - add the UI objects that are not defined as secured
-//    and the autocorrection is not turned off.
-// Keystroke logging is rellevant only to UITextField and UITextView objects
-CxList sensitiveText = sensitive.FindByTypes(new string[]{"UITextField","UITextView"});
-
-CxList AutoCorrectRemoved = Find_Autocorrection_Off(sensitiveText);
-result.Add(All.FindDefinition(sensitiveText - sensitiveText.FindByShortName(AutoCorrectRemoved)));
-
-// 3) Enabled IOS screenshot capture for sensitive information - add the UI objects that are not hidden defined as secured
-//    when the application goes to background, and so their content can be captured by the app-switcher
-result -= All.FindDefinition(Find_Screen_Caching(sensitive));
+//This query is deprecated.
```

Perl / Perl_Medium_Threat / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -1,8 +1 @@
-// Find all personal data that is sent to the output without encoding
-
-CxList personal = Find_Personal_Info();
-CxList outputs = Find_Write() + Find_DB_In();
-CxList sanitize = Find_Methods().FindByShortName("encrypt");
-result = personal.InfluencingOnAndNotSanitized(outputs, sanitize);
-
-result -= result.DataInfluencedBy(result);
+//This query is deprecated.
```

Perl / Perl_Medium_Threat / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,43 +1,43 @@
 CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null");
+CxList nullStrings = All.FindByName("null");
 CxList psw = Find_Passwords();
+CxList methods = Find_Methods();
```

```
 // dbi password
-CxList dbi_conn = Find_Methods().FindByMemberAccess("DBI", "connect") + Find_Methods().FindByMemberAccess("DBI", "connect_cached");
+CxList dbi_conn = All.NewCxList(
+   methods.FindByMemberAccess("DBI", "connect"),
+   methods.FindByMemberAccess("DBI", "connect_cached"));
 psw.Add(All.GetParameters(dbi_conn, 3));

 // oracle password
-CxList oracle_conn = Find_Methods().FindByShortName("ora_login");
+CxList oracle_conn = methods.FindByShortName("ora_login");
 psw.Add(All.GetParameters(oracle_conn, 3));

 // mysql password
-CxList mysql_conn = Find_Methods().FindByMemberAccess("Mysql", "connect");
+CxList mysql_conn = methods.FindByMemberAccess("Mysql", "connect");
 psw.Add(All.GetParameters(mysql_conn, 2));

 // Lists preperation
 CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
+CxList strLiterals = Find_Strings().FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);
+strLiterals -= emptyString;
+strLiterals -= nullStrings;
+//     (when the hardcoded string includes a space or dot we believe it is not a password string)
+strLiterals -= strLiterals.FindByNames("* *", "*.*", "*/*", "*\\*");
-CxList strLiterals = Find_Strings() - emptyString - NULL;
-//     (when the hardcoded string includes a space or dot we believe it is not a password string)
-strLiterals -= strLiterals.FindByName("* *");
-strLiterals -= strLiterals.FindByName("*.*");
-strLiterals -= strLiterals.FindByName("*/*");
-strLiterals -= strLiterals.FindByName("*\\*");
 CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
 CxList passNoString = psw - strLiterals;

 // Find password in an initialization operation
-CxList eq = All.FindByShortName("==");
-eq.Add(All.FindByShortName("eq"));
-eq.Add(All.FindByShortName("ne"));
-eq.Add(All.FindByShortName("!="));
+CxList eq = All.FindByShortNames("==", "eq", "ne", "!=");

 CxList equalsPassword = passNoString.GetByAncs(eq);
 eq = equalsPassword.GetFathers() * eq;
 equalsPassword = strLiterals.GetByAncs(eq);

 // Find password in as assignment
-CxList assignPassword = psw_in_lSide.GetAncOfType(typeof(AssignExpr));
+CxList assignPassword = psw_in_lSide.GetAncOfType<AssignExpr>();
```

```
+assignPassword.Add(psw_in_lSide.GetAncOfType<Declarator>());
 assignPassword = lit_in_rSide.GetByAncs(assignPassword);

 //// Add hardcoded password in post login
@@ -53,4 +53,4 @@

 assignPassword.Add(All.FindByRegex(@"password\s*=>\s*'\w"));

-result = assignPassword + equalsPassword;
+result.Add(assignPassword, equalsPassword);
```

PHP / PHP_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,21 +1,16 @@
 CxList binary_expr = Find_BinaryExpr();
 CxList emptyString = Find_Empty_Strings();
-CxList NULL = Find_NullLiteral();
+CxList nullStrings = Find_NullLiteral();
 CxList psw = Find_Passwords();
-CxList strings = Find_Strings();
+CxList strings = Find_Strings().FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);
 CxList methods = Find_Methods();
 CxList parameters = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

-CxList strLiterals = All.NewCxList();
-strLiterals.Add(strings);
+CxList strLiterals = All.NewCxList(strings);
 strLiterals -= emptyString;
-strLiterals -= NULL;
+strLiterals -= nullStrings;

-CxList allStrings = All.NewCxList();
-allStrings.Add(strLiterals);
-
-CxList allPsw = All.NewCxList();
-allPsw.Add(psw, Find_Password_Strings());
+CxList allPsw = All.NewCxList(psw, Find_Password_Strings());

 //     (when the hardcoded string includes a space or dot we believe it is not a password string)
 strLiterals -= strLiterals.FindByNames("*\t*", "* *", "*.*", "*/*", "*\\*");
@@ -28,8 +23,7 @@
 CxList fathers = lit_in_rSide.GetFathers() * psw_in_lSide.GetFathers();
 lit_in_rSide = lit_in_rSide.FindByFathers(fathers);
 //Add hardcoded passwords from assignments
-CxList assignPassword = All.NewCxList();
-assignPassword.Add(lit_in_rSide);
```

```
+CxList assignPassword = All.NewCxList(lit_in_rSide);


 //remove passwords with equal name and contant ==> currPassword = "currPassword";
 CxList notHdPass = All.NewCxList();
```

PHP / PHP_Medium_Threat / Broken_or_Risky_Encryption_Algorithm

Code changes

```
---
+++
@@ -1,13 +1,13 @@
 CxList methods = Find_Methods();
-CxList srtings = Find_Strings();
 CxList unkRefs = Find_UnknownReference();


-CxList mcryptInvokes = methods.FindByShortNames(new List<string>(){"mcrypt_encrypt","mcrypt_decrypt"}, false);
+CxList mcryptInvokes = methods.FindByShortNames(new List<string>(){
+       "mcrypt_encrypt", "mcrypt_decrypt", "mcrypt_generic", "mdecrypt_generic"}, false);
 CxList opensslEncrypts = methods.FindByShortName("openssl_encrypt*", false);
 CxList cipherParams = All.GetParameters(opensslEncrypts, 1);
 CxList possibleVulnStrings = unkRefs.FindAllReferences(cipherParams).GetAssigner();


-List <string> whiteList = new List<string>(){
+string[] whiteList = new string[]{
        "AES128",
        "AES192",
        "AES256",
@@ -57,8 +57,11 @@
        "id-aes256-wrap",
        "id-aes256-wrap-pad"
        };
+
 possibleVulnStrings.Add(cipherParams.FindByType<StringLiteral>());
 possibleVulnStrings -= possibleVulnStrings.FindByShortNames(whiteList, false);


 result.Add(possibleVulnStrings.InfluencingOn(cipherParams));
 result.Add(mcryptInvokes);
+
+result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

PHP / PHP_Medium_Threat / Broken_or_Risky_Hashing_Function

Code changes

```
---
+++
@@ -1,5 +1,4 @@
 CxList methods = Find_Methods();
-CxList srtings = Find_Strings();
 CxList unkRefs = Find_UnknownReference();
```

```
 CxList dedicatedHash = methods.FindByShortNames(new List<string>(){"ezmlm_hash","md5","sha1","md5_file","sha1_file"}, false);
@@ -8,7 +7,7 @@
 CxList hashParams = All.GetParameters(dynamicHash, 0);
 CxList possibleVulnStrings = unkRefs.FindAllReferences(hashParams).GetAssigner();

-List <string> whiteList = new List<string>(){
+string[] whiteList = new string[]{
        "sha224",
        "sha256",
        "sha384",
@@ -21,10 +20,11 @@
        "sha3-512",
        "whirlpool"
        };
-possibleVulnStrings.Add(hashParams.FindByType(typeof(StringLiteral)));
+possibleVulnStrings.Add(hashParams.FindByType<StringLiteral>());
 possibleVulnStrings -= possibleVulnStrings.FindByShortNames(whiteList, false);

 result.Add(possibleVulnStrings.InfluencingOn(hashParams));
+result.Add(dynamicHash.InfluencedBy(possibleVulnStrings).GetLastNodesInPath());
 result.Add(dedicatedHash);

 CxList dynamicCrypt = methods.FindByShortName("crypt", false);
```

PHP / PHP_Medium_Threat / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -1,22 +1 @@
-CxList outputs = All.NewCxList(
-   Find_DB_In() - Find_Raw_SQL_Selects(),
-   Find_File_Write_Outputs());
-
-CxList sanitizers = All.NewCxList(
-   Find_Hashing_Functions(),
-   Find_Encryption_Sanitize());
-
-string[] heuristics = new []{
-       "*Account*","*credentials*","*Credit*","*secret*",
-       "*SocialSecurity*","*SSN","SSN*","dob", "username",
-       "auth*", "*address*", "*mobile*","*telephone*", "pwd", "*password", "user_login"};
-
-CxList possiblePii = All.NewCxList(
-   // indexer refs whose key matches he heuristic
-   Find_Strings().FindByShortNames(heuristics, false).FindByFathers(Find_IndexerRefs()).GetFathers(),
-   // any unk ref that matches the heuristics
-   Find_UnknownReference().FindByShortNames(heuristics, false));
-
```

```
-result = Find_Interactive_Inputs().InfluencingOnAndNotSanitized(outputs, sanitizers)
-   .IntersectWithNodes(possiblePii)
-   .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
+//This query is deprecated.
```

PHP / PHP_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -80,6 +80,9 @@
 whitelistHttp = whitelistHttp.FindByAssignmentSide(CxList.AssignmentSide.Left);
 sanitizers.Add(whitelistHttp);


+//'tmp_name' field is not user-controllable. It is a random temporary filename generated by PHP
+sanitizers.Add(strs.FindByShortName("tmp_name").GetFathers().FindByType<IndexerRef>());
+
 // Remove unwanted isset sanitizer
 CxList toRemove = sanitizers.FindByShortName("isset");
 toRemove.Add(toRemove.GetFathers());
```

PLSQL / PLSQL_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList header_outputs =
-   Find_Header_Outputs();
-
-CxList sanitize = Find_XSS_Sanitize();
-
-CxList inputs = Find_Interactive_Inputs();
-
-result = header_outputs.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

Python / Python_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -3,15 +3,15 @@
 CxList psw = Find_Passwords();
 CxList unkRefs = Find_UnknownReference();
 CxList arrayInit = Find_ArrayInitializer();
+CxList strings = Find_Strings();


-CxList emptyStringNull = All.NewCxList();
-emptyStringNull.Add(emptyString, nullLiteral);
-
```

```
-CxList strings = Find_Strings();
-CxList strLiterals = strings - emptyStringNull;
-
+CxList strLiterals = All.NewCxList(strings);
+strLiterals -= emptyString;
+strLiterals -= nullLiteral;
+
 //when the hardcoded string includes a space or dot we believe it is not a password string
 strLiterals -= strLiterals.FindByNames("* *", "*.*");
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);

 // Find password in an initialization operation (declaration or assignment)
 CxList initializedPassword = psw.GetAssigner() * strLiterals;
@@ -19,7 +19,7 @@
 // Find password in an "equals" operation
 CxList bin = Find_BinaryExpr();

-CxList eq = bin.FindByShortNames(new List<string> {"==", "!="});
+CxList eq = bin.FindByShortNames("==", "!=");

 CxList equalsPassword = psw.GetFathers() * eq;
 equalsPassword = strLiterals.FindByFathers(equalsPassword);
@@ -32,8 +32,7 @@
 CxList connetionParam2 = All.GetParameters(connection, 2);
 CxList connetionParam1 = All.GetParameters(connection, 1);

-CxList connetParams = All.NewCxList();
-connetParams.Add(connetionParam1, connetionParam2);
+CxList connetParams = All.NewCxList(connetionParam1, connetionParam2);

 CxList pwdInConnectioParam = strLiterals.GetByAncs(connetParams * psw);
 CxList ancsPsw = unkRefs.GetByAncs(arrayInit.GetParameters(connection)) * psw;
```

Python / Python_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,15 +1 @@
-List<string> names = new List<string>{
-       "*orders*","*credit*","*invoice*","*booking*","*bill*","*payment*","*account*","*cash*","*customer*" };
-
-CxList tables = All.FindByShortNames(names, false);
-
-CxList inputs = Find_Inputs();
-CxList db = Find_DB();
-
-List<string> usersNames = new List<string>{"*user*","*cust*","*member*" };
-
```

```
-CxList user = All.FindByShortNames(usersNames, false);
-
-db = db.DataInfluencedBy(tables);
-db = db - db.DataInfluencedBy(user);
-result = inputs.DataInfluencingOn(db);
+//This query is deprecated.
```

Python / Python_Medium_Threat / Path_Traversal

Code changes

```
---
+++
@@ -21,7 +21,7 @@
    Find_Methods_By_Import("fileinput", new string[]{"input", "FileInput"}),
    Find_Methods_By_Import("linecache", new string[]{"getline"}),
    //Directory access
-   Find_Methods_By_Import("os.path", new string[]{"join", "walk"}),
+   Find_Methods_By_Import("os.path", new string[]{"walk"}),
    Find_Methods_By_Import("macpath", new string[]{"walks"}),
    Find_Methods_By_Import("dircache", new string[]{"listdir", "opendir"}),
    Find_Methods_By_Import("glob", new string[]{"glob", "iglob"}),
```

RPG / RPG_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1 +1,61 @@
-result = Common_Low_Visibility.Use_Of_Hardcoded_Password();
+CxList psw = Find_Passwords();
+
+// Find MOVE and MOVEL operations
+CxList moves = Find_Methods().FindByShortNames(new string[] {"MOVE", "MOVEL"});
+
+CxList pswInLSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
+CxList pswInLSideDecl = pswInLSide.FindByType<Declarator>();
+CxList strLiterals = Find_Strings();
+CxList litInRSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
+litInRSide.Add(strLiterals.GetParameters(moves, 0));
+//when the hardcoded string includes a space or dot we believe
+//it is not a password string
+litInRSide -= litInRSide.FindByNames(new []{"* *","*.*","*/*","*\\*"});
+
+//empty string is OK
+litInRSide -= Find_Empty_Strings();
+
+// Password in declaration
+CxList PasswordInDecl = pswInLSideDecl.FindByInitialization(litInRSide);
+
+//remove passwords with equal name and contant ==> currPassword = "currPassword";
```

```
+CxList notHdPass = All.NewCxList();
+char[] trimChars = new char[2] { '\'', '"'};
+foreach(CxList currPass in PasswordInDecl)
+{
+   CxList currStrInLeft = litInRSide.FindInitialization(currPass);
+   string strName = currStrInLeft.GetName().Trim(trimChars);
+   string passName = currPass.GetName();
+   if (passName.Equals(strName))
+   {
+       notHdPass.Add(currPass);
+   }
+}
+PasswordInDecl -= notHdPass;
+
+
+// Find password in a '==' operator
+CxList EqualBinaryExpr = psw.GetFathers().FindByType<BinaryExpr>().
+   GetByBinaryOperator(Checkmarx.Dom.BinaryOperator.IdentityEquality);
+CxList EqualOperatorStrings = All.NewCxList();
+foreach(CxList bin in EqualBinaryExpr)
+{
+   CxList password = psw.FindByFathers(bin);
+   CxList stringLit = strLiterals.FindByFathers(bin);
+
+   if(password.Count > 0 && stringLit.Count > 0)
+   {
+       EqualOperatorStrings.Add(stringLit);
+   }
+}
+
+// Password in simple assignment
+CxList assignPassword = pswInLSide.GetAncOfType<AssignExpr>();
+assignPassword = litInRSide.GetByAncs(assignPassword);
+
+// Password in MOVE and MOVEL operations
+CxList passInMoves = psw.GetParameters(moves, 1).GetAncOfType<MethodInvokeExpr>();
+passInMoves = litInRSide.GetParameters(passInMoves, 0);
+
+result = PasswordInDecl;
+result.Add(assignPassword, EqualOperatorStrings,passInMoves);
```

Ruby / Ruby_Best_Coding_Practice / Declaration_Of_Catch_For_Generic_Exception

Code changes

```
---
+++
@@ -1,8 +1,6 @@
-CxList Try = All.FindByType(typeof(TryCatchFinallyStmt));
-CxList Catch = All.FindByType(typeof(Catch));
```

```
-CxList generalException = All.FindByName("Exception").GetFathers().FindByType(typeof(Catch));
-
-CxList genExc = All.FindAllReferences(Catch); // an exception type was found
+CxList Try = Find_TryCatchFinallyStmt();
+CxList Catch = Find_Catch();
+CxList generalException = All.FindByShortName("Exception").GetFathers().FindByType<Catch>();


 generalException = Catch - All.FindAllReferences(Catch);
```

Ruby / Ruby_Best_Coding_Practice / Unclosed_Objects

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList close = Find_Methods().FindByName("*.close", false);
-CxList AllTrys = All.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList AllTrys = All.GetAncOfType<TryCatchFinallyStmt>();
 CxList fin = All.NewCxList();

 foreach(CxList oneTry in AllTrys)
@@ -9,7 +9,7 @@
 }
 fin = All.GetByAncs(fin);

-CxList Try = close.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList Try = close.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList oneTry in Try)
 {
    TryCatchFinallyStmt TryGraph = oneTry.TryGetCSharpGraph<TryCatchFinallyStmt>();
@@ -17,13 +17,9 @@
    CxList TryClose = close.GetByAncs(curTry);
    CxList AllClose = close.GetByAncs(oneTry);

-   if( (AllClose - TryClose).Count == 0)
-   {
-       if (TryClose.GetAncOfType(typeof(UsingStmt)).Count == 0)
-       {
-           result.Add(TryClose);
-       }
-   }
+   if((AllClose - TryClose).Count == 0 && TryClose.GetAncOfType<UsingStmt>().Count == 0)
+       result.Add(TryClose);
+
 }

 result -= result.FindByFathers(fin);
```

Ruby / Ruby_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList db = Find_SQL_DB_In();
-CxList db_not_in_try = Improper_Exception_Handling(db);
-CxList db_in_try = db - db_not_in_try;
-
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitized = Find_SQL_Sanitize();
-
-result = All.FindSQLInjections(inputs, db_in_try, sanitized);
+//This query is deprecated.
```

Ruby / Ruby_Low_Visibility / Improper_Transaction_Handling

Code changes

```
---
+++
@@ -1,16 +1,16 @@
 CxList notStrings = All - Find_Strings();
-notStrings -= notStrings.FindByType(typeof(MethodDecl));
+notStrings -= notStrings.FindByType<MethodDecl>();
 CxList Commit = notStrings.FindByShortName("commit*", false);
-Commit -= Commit.FindByType(typeof(Param));
-Commit -= Commit.FindByType(typeof(UnknownReference));
+Commit -= Commit.FindByType<Param>();
+Commit -= Commit.FindByType<UnknownReference>();
 CxList Rollback = notStrings.FindByShortName("rollback*", false);
-Rollback -= Rollback.FindByType(typeof(Param));
-Rollback -= Rollback.FindByType(typeof(UnknownReference));
+Rollback -= Rollback.FindByType<Param>();
+Rollback -= Rollback.FindByType<UnknownReference>();

-CxList TryBlock = Commit.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList TryBlock = Commit.GetAncOfType<TryCatchFinallyStmt>();

 result = Commit - Commit.GetByAncs(TryBlock);
-result -= result.FindByType(typeof(MethodRef));
+result -= result.FindByType<MethodRef>();

 foreach(CxList cml in TryBlock)
 {
@@ -27,8 +27,8 @@
    CxList RollbackInCatch = Rollback.GetByAncs(curCatch);



-   if( (RollbackInCatch.GetAncOfType(typeof(TryCatchFinallyStmt)) *
```

```
-       CommitInTry.GetAncOfType(typeof(TryCatchFinallyStmt))).Count == 0)
+   if( (RollbackInCatch.GetAncOfType<TryCatchFinallyStmt>() *
+       CommitInTry.GetAncOfType<TryCatchFinallyStmt>()).Count == 0)
    {
        result.Add(cml);
    }
```

Ruby / Ruby_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,64 +1,60 @@
 CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null");
+CxList nullStrings = All.FindByName("null");
 CxList psw = Find_Passwords();
+CxList methods = Find_Methods();

 // Lists preperation
 CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList strLiterals = Find_Strings() - emptyString - NULL;
+CxList strLiterals = Find_Strings();
+strLiterals -= emptyString;
+strLiterals -= nullStrings ;
 //     (when the hardcoded string includes a space or dot we believe it is not a password string)
-strLiterals -= strLiterals.FindByName("* *");
-strLiterals -= strLiterals.FindByName("*.*");
-strLiterals -= strLiterals.FindByName("*/*");
-strLiterals -= strLiterals.FindByName("*\\*");
+strLiterals -= strLiterals.FindByNames("* *", "*.*", "*/*", "*\\*");
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);
+
 CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);

 // Find password in an initialization operation
-CxList eq = All.FindByShortName("==");
-eq.Add(All.FindByShortName("==="));
-eq.Add(All.FindByShortName("!="));
-eq.Add(All.FindByShortName("!=="));
-eq.Add(All.FindByShortName("|"));
-eq.Add(All.FindByShortName("&"));
-eq.Add(All.FindByShortName("^"));
-eq = eq.GetAncOfType(typeof(BinaryExpr));
+CxList eq = All.FindByShortNames("==", "===", "!=", "!==", "|", "&", "^").GetAncOfType<BinaryExpr>();

-CxList strPass = psw.FindByType(typeof (StringLiteral));
-CxList paramPass = psw.FindByType(typeof (Param));
+CxList strPass = psw.FindByType<StringLiteral>();
```

```
+CxList paramPass = psw.FindByType<Param>();


 // Find all comparisons of type ==> "hello" == password
 eq = strLiterals.GetFathers() * eq;
 CxList equalsPassword = (psw - strPass).FindByFathers(eq);


 // Find password in as assignment
-CxList assignPassword = psw_in_lSide.GetAncOfType(typeof(AssignExpr));
+CxList assignPassword = psw_in_lSide.GetAncOfType<AssignExpr>();
 assignPassword = lit_in_rSide.GetByAncs(assignPassword);


-CxList assignPasswords = psw - strPass - paramPass.FindByShortName(strPass);
+CxList assignPasswords = All.NewCxList(psw);
+assignPasswords -= strPass;
+assignPasswords -= paramPass.FindByShortName(strPass);
 assignPassword.Add(strLiterals.GetFathers() * assignPasswords);    //assignment of type ==> password = "This_string"
 ///assignment of type ==>  pass = String.new("This is my string")
-assignPassword.Add(strLiterals.GetAncOfType(typeof (ObjectCreateExpr)).GetFathers() * assignPasswords);
+assignPassword.Add(strLiterals.GetAncOfType<ObjectCreateExpr>().GetFathers() * assignPasswords);


 // Find password inside define methods
-CxList defineMethods = Find_Methods().FindByShortName("define");
-CxList pswParameter = All.GetParameters(defineMethods, 0).FindByType(typeof(StringLiteral));
-CxList pswLiteralParameter = All.GetParameters(defineMethods, 1).FindByType(typeof(StringLiteral));
+CxList defineMethods = methods.FindByShortName("define");
+CxList pswParameter = strLiterals.GetParameters(defineMethods, 0);
+CxList pswLiteralParameter = strLiterals.GetParameters(defineMethods, 1);


 pswParameter = pswParameter * psw;
 pswLiteralParameter = pswLiteralParameter * strLiterals;
 CxList definePasswords = defineMethods.FindByParameters(pswParameter).FindByParameters(pswLiteralParameter);


 //Find passwords in "equal?", "eql?", "casecmp"
-CxList allMethods = All.FindByType(typeof (MethodInvokeExpr));
-CxList equalMethods = allMethods.FindByShortName("eql?") + allMethods.FindByShortName("equal?")
-   + allMethods.FindByShortName("casecmp");
+CxList equalMethods = methods.FindByShortNames("eql?", "equal?", "casecmp");
 //find equal methods ==> "hello".eql? pwd  ==> "hello".equal? pwd
 CxList strEqualMeth = strLiterals.GetFathers();
-strEqualMeth = allMethods.GetByAncs(strEqualMeth);
-CxList inEql = psw.GetParameters(strEqualMeth).FindByType(typeof (Param));
+strEqualMeth = methods.GetByAncs(strEqualMeth);
+CxList inEql = psw.GetParameters(strEqualMeth).FindByType<Param>();
+
 //find equal methods ==> pwd.eql? "hello"  ==> pwd.equal? "hello"
 strEqualMeth = equalMethods.FindByParameters(strLiterals);
-CxList equalOfPsd = allMethods.FindByParameters(psw);
+CxList equalOfPsd = methods.FindByParameters(psw);
```

```
 strEqualMeth *= equalOfPsd;
 inEql.Add(strLiterals.GetByAncs(strEqualMeth));


-result = definePasswords + equalsPassword + assignPassword + inEql;
+result.Add(definePasswords, equalsPassword, assignPassword, inEql);
```

Ruby / Ruby_Low_Visibility / XSS_Evasion_Attack

Code changes

```
---
+++
@@ -1,5 +1 @@
-CxList decode = All.FindByName("*decode*", false);
-CxList sanitize = Find_XSS_Sanitize();
-CxList output = Find_Interactive_Outputs();
-
-result = output.InfluencedByAndNotSanitized(decode, sanitize);
+//This query is deprecated.
```

Ruby / Ruby_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,13 +1 @@
-List<string> names = new List<string> {"*orders*", "*credit*", "*invoice*", "*booking*",
-       "*bill*","*payment*","*account*","*cash*","*customer*"};
-
-CxList tables = All.FindByShortNames(names, false);
-
-CxList inputs = Find_Interactive_Inputs();
-CxList db = Find_DB();
-
-CxList user = All.FindByNames(new string[] {"*user*","*cust*","*member*"}, false);
-
-db = db.DataInfluencedBy(tables);
-db = db - db.DataInfluencedBy(user);
-result = inputs.DataInfluencingOn(db);
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_JSON_GEM_Remote_Code

Code changes

```
---
+++
@@ -1,5 +1 @@
-// CVE-2013-0269  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0269
-// The JSON gem before 1.5.5, 1.6.x before 1.6.8, and 1.7.x before 1.7.7 for Ruby allows remote attackers ...
-// Corresponds to CWE-20 http://cwe.mitre.org/data/definitions/20.html
-
-result = Find_Packages_Not_Satisfying_Version("json", "1.7.7");
```

```
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_JSON_Remote_Code_Execution

Code changes

```
---
+++
@@ -1,8 +1 @@
-// CVE-2013-0333 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0333
-// Rails<3.0.20 : lib/active_support/json/backends/yaml.rb in Ruby on Rails 2.3.x before 2.3.16 and 3.0.x
-//    before 3.0.20 does not properly convert JSON data to YAML data[...] allows remote attackers to execute arbitrary code
-// Corresponds to CWE 94, http://cwe.mitre.org/data/definitions/94.html
-
-
-CxList ver = Find_Rails_Version();
-result = Find_Gemlocks_Not_Satisfying_Version(ver, "3.0.20", "yes");
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_Rails_Allows_Bypass_Access_Control

Code changes

```
---
+++
@@ -1,12 +1 @@
-// CVE_2012_2660 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2660
-// Rails<3.2.4 : A remote attacker could send specially-crafted SQL statements using an unspecified parameter,
-//   which could allow the attacker to view, add, modify or delete information in the back-end database
-// Corresponds CWE-264 to http://cwe.mitre.org/data/definitions/264.html
-CxList ver = Find_Rails_Version();
-result = Find_Gemlocks_Not_Satisfying_Version(ver, "3.2.4", "yes");
-
-// CVE-2013-0155
-// Ruby on Rails 3.0.x before 3.0.19, 3.1.x before 3.1.10, and 3.2.x before 3.2.11 does not properly consider
-//  differences in parameter handling between the Active Record component and the JSON implementation
-
-result.Add(Find_Gemlocks_Not_Satisfying_Version(ver, "3.2.11", "yes"));
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_Rails_Allows_Cross_Site_Request_Forgery

Code changes

```
---
+++
@@ -1,5 +1 @@
-// CVE-2011-0447 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-0447
-// Rails<2.3.11 : Cross-Site-Request-Forgery , CVE-2011-0447
-// Corresponds to CWE-352 http://cwe.mitre.org/data/definitions/352.html
-CxList ver = Find_Rails_Version();
-result = Find_Gemlocks_Not_Satisfying_Version(ver, "2.3.11", "yes");
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_Rails_Allows_DOS_via_ActiveRecord

Code changes

```
---
+++
@@ -1,5 +1 @@
-// CVE-2013-1854 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-1854
-// Rails<3.2.13: Denial of Service via ActiveRecord
-// Corresponds to CWE-400 http://cwe.mitre.org/data/definitions/400.html
-CxList ver = Find_Rails_Version();
-result = Find_Gemlocks_Not_Satisfying_Version(ver, "3.2.13", "yes");
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_Rails_Allows_SQL_Injection

Code changes

```
---
+++
@@ -1,17 +1 @@
-// CVE-2011-2930 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2930
-// Multiple SQL injection vulnerabilities in the quote_table_name method in the ActiveRecord adapters in activerecord/lib/active_record/connection_adapters/ in Ruby on Rails before 2.3.13, 3.0.x before 3.0.10, and 3.1.x before 3.1.0.rc5 allow remote attackers to execute arbitrary SQL commands via a crafted column nam
-// Corresponds to CWE-89 http://cwe.mitre.org/data/definitions/89.html
-CxList ver = Find_Rails_Version();
-result = Find_Gemlocks_Not_Satisfying_Version(ver, "3.1.0.rc5", "yes");
-
-// CVE-2012_2695 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2695
-// The Active Record component in Ruby on Rails before 3.0.14, 3.1.x before 3.1.6, and 3.2.x before 3.2.6 does not properly implement the passing of request data to a where method in an ActiveRecord class, which allows remote attackers to conduct certain SQL injection attacks via nested query parameters that leverage
-// Rails<3.2.6 : SQL Injection: CVE-2012-2695
-// Corresponds to CWE-89 http://cwe.mitre.org/data/definitions/89.html
-result.Add(Find_Gemlocks_Not_Satisfying_Version(ver, "3.2.6", "yes"));
-
-// CVE-2012-6496 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6496
-// Rails<3.2.10 : SQL Injection: CVE-2012-6496
-// SQL injection vulnerability in the Active Record component in Ruby on Rails before 3.0.18, 3.1.x before 3.1.9, and 3.2.x before 3.2.10 allows remote attackers to execute arbitrary SQL commands via a crafted request that leverages incorrect behavior of dynamic finders in applications that can use unexpected data ty
-
-result.Add(Find_Gemlocks_Not_Satisfying_Version(ver, "3.2.10", "yes"));
+//This query is deprecated.
```

Ruby / Ruby_Vulnerable_Outdated_Versions / Outdated_Rails_Allows_XSS

Code changes

```
---
+++
@@ -1,14 +1 @@
-// CVE-2012-3464 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-3464
-// Cross-site scripting (XSS) vulnerability in in Ruby on Rails before 3.0.17, 3.1.x before 3.1.8, and 3.2.x before 3.2.8 might
-//   allow remote attackers to inject arbitrary web script or HTML via vectors involving a ' (quote) character.
-// Corresponds to CWE 79, http://cwe.mitre.org/data/definitions/79.html
-
-// Not escaping single quotes
-CxList ver = Find_Rails_Version();
```

```
-result = Find_Gemlocks_Not_Satisfying_Version(ver, "3.2.8", "yes");
-
-// Rails<3.0.11 : XSS- vulnerability in the translate helper keys may allow
-//   an attacker to insert arbitrary code into a page
-// http://groups.google.com/group/rubyonrails-security/browse_thread/thread/2b61d70fb73c7cc5
-
-result.Add(Find_Gemlocks_Not_Satisfying_Version(ver, "3.0.11", "yes"));
+//This query is deprecated.
```

Rust / Rust_High_Risk / Connection_String_Injection

Code changes

```
---
+++
@@ -7,7 +7,7 @@
        "Client",
        "ClientOptions",
        "ConnectionString").GetMembersOfTarget();
-CxList mongoConnectionMethods = mongoConnectionMembers.FindByShortNames("with_uri_str*", "parse", "from_str");
+CxList mongoConnectionMethods = mongoConnectionMembers.FindByShortNames("with_uri_str", "parse*", "from_str");

 // Sqlx
 CxList sqlxConnectionMembers = importRefs.FindByShortNames(
@@ -26,8 +26,7 @@
        "PgConnectOptions",
        "MySqlConnectOptions",
        "AnyConnectOptions",
-       "SqliteConnectOptions"
-       ).GetMembersOfTarget();
+       "SqliteConnectOptions").GetMembersOfTarget();
 sqlxConnectMethods.Add(sqlxConnectOptionsMembers.FindByShortName("from_url"));

 CxList sqlxNewConnectionOptionsMethods = sqlxConnectOptionsMembers.FindByShortName("new");
@@ -37,7 +36,12 @@
    methods.FindByShortName("set_connect_options"));

 // Diesel
-CxList dieselEstablishMethods = methods.FindByMemberAccess("PgConnection.establish");
+CxList dieselConnectionImplementorsMembers = importRefs.FindByShortNames(
+   "MysqlConnection",
+   "PgConnection",
+   "SqliteConnection",
+   "PooledConnection").GetMembersOfTarget();
+CxList dieselEstablishMethods = dieselConnectionImplementorsMembers.FindByShortName("establish");

 CxList sinks = All.NewCxList(mongoConnectionMethods, dieselEstablishMethods, sqlxConnectMethods);
 CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Stored_Inputs());
```

Rust / Rust_Low_Visibility / JWT_Excessive_Expiration_Time

Code changes

```
---
+++
@@ -5,20 +5,18 @@

 IAbstractValue unsafeValuesRange = new IntegerIntervalAbstractValue(86400, null);

-CxList claimsObjects = objectCreations.FindByShortName("Claims");
+CxList claimType = unkRefs.FindByType("Claims");
+objectCreations.Add(claimType.GetAncOfType<Declarator>());

-CxList claimType = unkRefs.FindByType("Claims");
-claimsObjects.Add(claimType.GetAncOfType<Declarator>());
+objectCreations.Add(unkRefs.FindAllReferences(objectCreations));

-claimsObjects.Add(unkRefs.FindAllReferences(claimsObjects));
-
-CxList parameters = Find_Param().FindParameterByName("exp").GetByAncs(claimsObjects);
-parameters.Add(expressions.GetParameters(claimsObjects.FindByNumberOfParameters(1), 0));
+CxList parameters = Find_Param().FindParameterByName("exp").GetByAncs(objectCreations);
+parameters.Add(expressions.GetParameters(objectCreations.FindByNumberOfParameters(1), 0));

 CxList excessiveExp = expressions.FindByAbstractValue(abstractValue => abstractValue.IncludedIn(unsafeValuesRange));

 CxList expField = expressions.FindByFathers(parameters);
-expField.Add(claimsObjects.GetMembersOfTarget().FindByShortName("exp"));
+expField.Add(objectCreations.GetMembersOfTarget().FindByShortName("exp"));

 excessiveExp = excessiveExp.InfluencingOn(expField).GetFirstNodesInPath();
```

Rust / Rust_Low_Visibility / JWT_Lack_of_Expiration_Time

Code changes

```
---
+++
@@ -2,11 +2,9 @@
 CxList methods = Find_Methods();
 CxList expressions = Find_Expressions();

-CxList claimsObjects = objectCreations.FindByShortName("Claims");
-
-CxList sanitizers = claimsObjects.FindByParameterName("exp");
-sanitizers.Add(claimsObjects.FindByNumberOfParameters(1));
+CxList sanitizers = objectCreations.FindByParameterName("exp");
+sanitizers.Add(objectCreations.FindByNumberOfParameters(1));

 CxList sinks = methods.FindByShortName("encode");
```

```
-result = expressions.GetParameters(sinks, 1).InfluencedByAndNotSanitized(claimsObjects, sanitizers);
+result = expressions.GetParameters(sinks, 1).InfluencedByAndNotSanitized(objectCreations, sanitizers);
```

Rust / Rust_Medium_Threat / DoS_by_Sleep

Code changes

```
---
+++
@@ -1,8 +1,27 @@
 CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Stored_Inputs());
-CxList methods = Find_Methods();

-CxList outputs = methods.FindByShortNames("sleep*", "interval*");
+CxList outputs = Find_Methods().FindByShortNames("sleep*", "interval*");

-CxList sanitizer = Find_UnknownReference().FindByShortName("max_value");
+// Sanitizers - verifying the input does not exceed an allowed maximum value
+CxList integers = Find_IntegerLiterals();
+CxList intsAndUnkRefs = All.NewCxList(integers, Find_UnknownReference());

-result = inputs.InfluencingOnAndNotSanitized(outputs, sanitizer);
+CxList intAndRealAbsValue = All.NewCxList(integers, Find_RealLiterals());
+CxList intAbsValue = intsAndUnkRefs.FindByAbstractValues(intAndRealAbsValue);
+
+CxList conditions = Find_BinaryExpr().InfluencedBy(intAbsValue);
+
+CxList lessThan = conditions.FilterByDomProperty<BinaryExpr>(_ => _?.Operator ==
+   BinaryOperator.LessThan || _?.Operator == BinaryOperator.LessThanOrEqual);
+CxList greaterThan = conditions.FilterByDomProperty<BinaryExpr>(_ => _?.Operator ==
+   BinaryOperator.GreaterThan || _?.Operator == BinaryOperator.GreaterThanOrEqual);
+
+CxList inputsLeft = lessThan.CxSelectDomProperty<BinaryExpr>(_ => _.Left)
+   .InfluencedBy(inputs).GetFirstNodesInPath();
+CxList inputsRight = greaterThan.CxSelectDomProperty<BinaryExpr>(_ => _.Right)
+   .InfluencedBy(inputs).GetFirstNodesInPath();
+
+CxList sanitizers = All.NewCxList(inputsLeft, inputsRight);
+sanitizers.Add(outputs.FindByParameters(inputs.FindAllReferences(sanitizers)));
+
+result = inputs.InfluencingOnAndNotSanitized(outputs, sanitizers);
```

Rust / Rust_Medium_Threat / Empty_Password_In_Connection_String

Code changes

```
---
+++
@@ -1,14 +1,18 @@
 CxList emptyStrings = Find_Empty_Strings();
 emptyStrings.Add(Find_ConnectionStrings(true));
```

```
+CxList vars = Find_Declarators();
 CxList expressions = Find_Expressions();
 CxList members = Find_MemberAccesses();
 CxList methods = Find_Methods();
+


 Func < string[], CxList > GetImports = (importsNames) =>
    {
    CxList imports = members.FindByNames(importsNames);
    imports.Add(expressions.FindAllReferences(imports.GetAssignee()));
+   imports.Add(vars.FindByNames(importsNames));
+   imports.Add(expressions.FindAllReferences(vars));
    return imports;
    };
```

Rust / Rust_Medium_Threat / Encoding_Used_Instead_of_Encryption

Code changes

```
---
+++
@@ -8,7 +8,9 @@
    Find_Encrypt(),
    Find_Hash());
-CxList outputs = Find_Stored_Outputs();
+CxList outputs = All.NewCxList(
+   Find_Stored_Outputs(),
+   Find_Remote_Outputs());


 result = outputs.InfluencedByAndNotSanitized(encodedSensitiveInfo, sanitizers)
    .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Rust / Rust_Medium_Threat / Hardcoded_Password_in_Connection_String

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList passwords = Find_ConnectionStrings(false);
-
+CxList vars = Find_Declarators();
 CxList expressions = Find_Expressions();
 CxList members = Find_MemberAccesses();


@@ -7,6 +7,8 @@
    {
    CxList imports = members.FindByNames(importsNames);
    imports.Add(expressions.FindAllReferences(imports.GetAssignee()));
```

```
+   imports.Add(vars.FindByNames(importsNames));
+   imports.Add(expressions.FindAllReferences(vars));
    return imports;
    };


@@ -18,7 +20,7 @@
    };


 //Mongo
-string[] mongoImports = new[]{"mongodb.sync.Client", "mongodb.Client"};
+string[] mongoImports = new[]{"mongodb.sync.Client", "mongodb.Client", "Client"};
 string[] mongoMethods = new[]{"with_uri_str", "with_options"};
 CxList mongoSinks = GetMembers(mongoImports, mongoMethods);
```

Rust / Rust_Medium_Threat / JWT_Sensitive_Information_Exposure

Code changes

```
---
+++
@@ -2,11 +2,15 @@

 CxList sanitize = All.NewCxList();
 sanitize.Add(
-   Find_DB_In(),
    Find_Hash(),
    Find_Encrypt());

-CxList outputs = Find_Methods().FindByShortName("encode");
+CxList jsonwebtokenImports = Find_Import().FindByShortName("jsonwebtoken");
+IEnumerable<string> filesWithJSONIMport = jsonwebtokenImports.CxSelectElementValue<Import, string>(x => x.LinePragma.FileName);

-result = inputs.InfluencingOnAndNotSanitized(outputs, sanitize)
-   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+if(jsonwebtokenImports.Count > 0)
+{
+   CxList outputs = Find_Methods().FindByShortName("encode").FindByFileNames(filesWithJSONIMport.ToArray());
+   result = inputs.InfluencingOnAndNotSanitized(outputs, sanitize)
+       .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+}
```

Rust / Rust_Medium_Threat / JWT_Use_Of_Hardcoded_Secret

Code changes

```
---
+++
@@ -1,8 +1,13 @@
-CxList inputs = Find_Strings();
-CxList methods = Find_Methods();
+CxList sinks = Find_Methods().FindByShortName("from_secret");
```

```
-CxList sanitizers = methods.FindByShortNames(new[]{"var", "var_os", "vars", "vars_os", "env"});
+CxList unknownRefs = Find_UnknownReference();
+CxList possibleParameters = All.NewCxList(Find_String_Literal(), unknownRefs);

-CxList sinks = methods.FindByShortName("from_secret");
+CxList possibleInputs = Find_Strings();
+CxList possibleInputsDeclarators = possibleInputs.GetFathers().FindByType<Declarator>();
+possibleInputs.Add(unknownRefs.FindAllReferences(possibleInputsDeclarators));

-result = sinks.InfluencedByAndNotSanitized(inputs, sanitizers);
+CxList taintedParameters = possibleParameters.GetParameters(sinks, 0) * possibleInputs;
+
+result = sinks.InfluencedBy(possibleInputs).IntersectWithNodes(taintedParameters)
+    .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Scala / Scala_Low_Visibility / Potential_Stored_XSS

Code changes

```
---
+++
@@ -1,6 +1 @@
-CxList db = Find_DB_Out();
-CxList read = Find_Read_NonDB();
-CxList outputs = Find_Potential_Outputs() - Find_Header_Outputs();
-CxList sanitize = Find_XSS_Sanitize();
-
-result = (db + read).InfluencingOnAndNotSanitized(outputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

Scala / Scala_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,22 +1 @@
-CxList tables = All.FindByShortName("*orders*", false);
-tables.Add(All.FindByShortName("*credit*", false));
-tables.Add(All.FindByShortName("*invoice*", false));
-tables.Add(All.FindByShortName("*booking*", false));
-tables.Add(All.FindByShortName("*bill*", false));
-tables.Add(All.FindByShortName("*payment*", false));
-tables.Add(All.FindByShortName("*account*", false));
-tables.Add(All.FindByShortName("*cash*", false));
-tables.Add(All.FindByShortName("*customer*", false));
-
-CxList inputs = Find_Interactive_Inputs();
-CxList db = Find_DB_In();
-
-CxList user = All.FindByShortName("*user*", false);
```

```
-user.Add(All.FindByShortName("*cust*", false));
-user.Add(All.FindByShortName("*member*", false));
-
-db = db.DataInfluencedBy(tables);
-db -= db.DataInfluencedBy(user);
-CxList sanitize = Find_Parameter_Tampering_Sanitize();
-
-result = inputs.InfluencingOnAndNotSanitized(db, sanitize);
+//This query is deprecated.
```

Scala / Scala_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,5 +1 @@
-CxList sanitize = Find_Splitting_Sanitizer();
-CxList inputs = Find_Interactive_Inputs() - Find_Headers();
-CxList outputs = Find_Header_Outputs();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

Scala / Scala_Medium_Threat / Multiple_Binds_to_the_Same_Port

Code changes

```
---
+++
@@ -1,48 +1 @@
-/*
-   This query looks for methods in which server socket bindings to variable interfaces
-   with the same port number happen.
-*/
-CxList integers = Find_Integers();
-CxList integerLiterals = Find_IntegerLiterals();
-CxList zeros = integerLiterals.FindByShortName("0");
-integers.Add(integerLiterals);
-CxList allSocketAddress = All.FindByType("InetSocketAddress");
-CxList portNumberAsSecondParameter = integers.GetParameters(allSocketAddress, 1);
-
-/*
-   Passing port 0 to the InetSocketAddress constructor instructs it to bind
-   to a random available temporary port, which is safe and therefore not a result
-*/
-
-zeros.Add(portNumberAsSecondParameter.DataInfluencedBy(zeros));
-portNumberAsSecondParameter -= zeros;
-
-/*
-   Only constructors of InetSocketAddress with two parameters matter since initializing
```

```
-   it with a port number only (one parameter), binds it with the wildcard interface
-   (0.0.0.0) which can't be binded twice.
-*/
-CxList socketAddressWithTwoParameters = allSocketAddress.FindByParameters(portNumberAsSecondParameter);
-CxList interfaceValue = All.GetParameters(socketAddressWithTwoParameters, 0);
-interfaceValue -= interfaceValue.FindByType(typeof(Param));
-
-/*
-   Remove results binding to the same interface. i.e. using a string literal as interface.
-*/
-CxList strings = Find_Strings();
-CxList safeInterfaceValue = (interfaceValue * strings);
-safeInterfaceValue.Add(interfaceValue.DataInfluencedBy(strings));
-
-/*
-   Remove references to localhost name since they are constant although possibly not strings.
-*/
-safeInterfaceValue.Add(interfaceValue.FindByShortName("*localhost*", false));
-
-/*
-   Return parent method encapsulating unsafe bindings since calling them multiple times can lead
-   to the vulnerability.
-*/
-CxList methods = Find_Methods();
-CxList bindings = methods.FindByShortName("bind", false);
-CxList unsafeSocketAddresses = socketAddressWithTwoParameters.FindByParameters(interfaceValue - safeInterfaceValue);
-result = methods.GetMethod(bindings.DataInfluencedBy(unsafeSocketAddresses));
+//This query is deprecated.
```

Scala / Scala_Stored / Stored_HTTP_Response_Splitting

Code changes

```
---
+++
@@ -1,5 +1 @@
-CxList sanitize = Find_Splitting_Sanitizer();
-CxList inputs = Find_Read() + Find_DB_Out() - Find_Headers();
-CxList outputs = Find_Header_Outputs();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+//This query is deprecated.
```

Swift / Swift_Low_Visibility / Heap_Inspection

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList methods = Find_Methods();
-
```

```
-List<string> methodsNames = new List<string> {"realloc","fork","vfork"};
-CxList badForSecurity = methods.FindByShortNames(methodsNames);
-
-CxList personal = Find_Personal_Info();
-
-result = badForSecurity.DataInfluencedBy(personal);
+//This query is deprecated.
```

Swift / Swift_Low_Visibility / Parameter_Tampering

Code changes

```
---
+++
@@ -1,14 +1 @@
-CxList declarators = Find_Declarators();
-CxList objectCreations = Find_ObjectCreations();
-CxList unknownRefs = Find_UnknownReference();
-
-CxList inputs = Find_Interactive_Inputs_Web() * Find_ParamDecl();
-
-// UUID acts as sanitizer
-CxList uuid = declarators.DataInfluencedBy(objectCreations.FindByType("UUID"));
-CxList uuidDecl = uuid.GetLastNodesInPath();
-CxList sanitizers = unknownRefs.FindAllReferences(uuidDecl);
-
-CxList outputs = Find_DB_For_Parameter_Tampering();
-
-result = outputs.InfluencedByAndNotSanitized(inputs, sanitizers);
+//This query is deprecated.
```

VB6 / VB6_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---
+++
@@ -1,25 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList input = Find_Interactive_Inputs();
-
-   CxList strings = Find_Strings();
-   CxList Select = strings.FindByName("*select*", false);
-   CxList Where = strings.FindByName("*where*", false);
-   CxList And = strings.FindByName("*and *", false) +
-       strings.FindByName("* and*", false);
-
-   possible_db = possible_db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-   possible_db -= possible_db.DataInfluencedBy(And);
```

```
-
-   result = possible_db.DataInfluencedBy(input);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_DB();
-       db = db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-       db -= db.DataInfluencedBy(And);
-       result -= db.DataInfluencedBy(input);
-   }
-}
+//This query is deprecated.
```

VB6 / VB6_Heuristic / Heuristic_SQL_Injection

Code changes

```
---
+++
@@ -1,15 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList inputs = Find_Interactive_Inputs();
-   CxList sanitized = Find_SQL_Sanitize();
-
-   result = inputs.InfluencingOnAndNotSanitized(possible_db, sanitized);
-
-   if (result.Count > 0)
-   {
-       CxList db = Find_SQL_DB_In();
-       result -= inputs.InfluencingOnAndNotSanitized(db, sanitized);
-   }
-}
+//This query is deprecated.
```

VbNet / VbNet_Best_Coding_Practice / Unclosed_Objects

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList close = Find_Methods().FindByName("*.Close", false);
-CxList AllTrys = All.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList AllTrys = All.GetAncOfType<TryCatchFinallyStmt>();
 CxList fin = All.NewCxList();

 foreach(CxList oneTry in AllTrys)
@@ -14,7 +14,7 @@
 }
```

```
 fin = All.GetByAncs(fin);


-CxList Try = close.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList Try = close.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList oneTry in Try)
 {
    try{
@@ -25,7 +25,7 @@


        if( (AllClose - TryClose).Count == 0)
        {
-           if (TryClose.GetAncOfType(typeof(UsingStmt)).Count == 0)
+           if (TryClose.GetAncOfType<UsingStmt>().Count == 0)
            {
                result.Add(TryClose);
            }
```

VbNet / VbNet_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,13 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList db = Find_DB_Base() + Find_Read() - All.FindByShortName("ExecuteNonQuery", false);
-   CxList dbParams = All.GetParameters(db);
-   CxList sanitize = Find_SQL_Sanitize();
-
-   // Find flows: possible_db -> db, possible_db -> possible_db
-   result = All.FindSQLInjections(possible_db, possible_db + dbParams, sanitize);
-   // Find flows: db -> possible db
-   result.Add(All.FindSQLInjections(db, possible_db, sanitize));
-}
+//This query is deprecated.
```

VbNet / VbNet_Heuristic / Heuristic_CSRF

Code changes

```
---
+++
@@ -1,14 +1 @@
-if (All.isWebApplication)
-{
-   CxList possible_db = Find_DB_Heuristic();
-
-   if (possible_db.Count > 0)
-   {
```

```
-       CxList requests = Find_Interactive_Inputs();
-       requests.Add(All.FindByName("*request.QueryString*", false));
-       CxList strings = Find_Strings();
-       CxList write = strings.FindByNames(new string [] {"*update*", "*delete*", "*insert*"}, StringComparison.OrdinalIgnoreCase);
-
-       result = possible_db.DataInfluencedBy(write).DataInfluencedBy(requests);
-   }
-}
+//This query is deprecated.
```

VbNet / VbNet_Heuristic / Heuristic_DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,25 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList tables = All.FindByName("*orders*", false) +
-       All.FindByName("*credit*", false) +
-       All.FindByName("*invoice*", false) +
-       All.FindByName("*booking*", false) +
-       All.FindByName("*bill*", false) +
-       All.FindByName("*payment*", false) +
-       All.FindByName("*account*", false) +
-       All.FindByName("*cash*", false) +
-       All.FindByName("*customer*", false);
-
-   CxList inputs = Find_Interactive_Inputs();
-
-   CxList user = All.FindByName("*user*", false) +
-       All.FindByName("*cust*", false) +
-       All.FindByName("*member*", false);
-
-   possible_db = possible_db.DataInfluencedBy(tables);
-   possible_db -= possible_db.DataInfluencedBy(user);
-
-   result = inputs.InfluencingOnAndNotSanitized(possible_db, Find_Parameters());
-}
+//This query is deprecated.
```

VbNet / VbNet_Heuristic / Heuristic_Parameter_Tampering

Code changes

```
---
+++
@@ -1,17 +1 @@
-CxList possible_db = Find_DB_Heuristic();
```

```
-
-if (possible_db.Count > 0)
-{
-   CxList input = Find_Interactive_Inputs();
-
-   CxList strings = Find_Strings();
-   CxList Select = strings.FindByName("*select*", false);
-   CxList Where = strings.FindByName("*where*", false);
-   CxList And = strings.FindByName("*and *", false) +
-       strings.FindByName("* and*", false);
-
-   possible_db = possible_db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-   possible_db -= possible_db.DataInfluencedBy(And);
-
-   result = possible_db.DataInfluencedBy(input);
-}
+//This query is deprecated.
```

VbNet / VbNet_Heuristic / Heuristic_SQL_Injection

Code changes

```
---
+++
@@ -1,9 +1 @@
-CxList possible_db = Find_DB_Heuristic();
-
-if (possible_db.Count > 0)
-{
-   CxList inputs = Find_Interactive_Inputs();
-   CxList sanitized = Find_SQL_Sanitize();
-
-   result = inputs.InfluencingOnAndNotSanitized(possible_db, sanitized);
-}
+//This query is deprecated.
```

VbNet / VbNet_Heuristic / Heuristic_Stored_XSS

Code changes

```
---
+++
@@ -1,13 +1 @@
-if(All.isWebApplication)
-{
-   CxList possible_db = Find_DB_Heuristic() - Find_IO();
-
-   if (possible_db.Count > 0)
-   {
-       CxList outputs = Find_XSS_Outputs();
-       CxList sanitize = Find_XSS_Sanitize();
-
```

```
-       result = All.FindXSS(possible_db, outputs, sanitize);
-
-   }
-}
+//This query is deprecated.
```

VbNet / VbNet_High_Risk / UTF7_XSS

Code changes

```
---
+++
@@ -1,12 +1 @@
-if(All.isWebApplication)
-{
-   CxList UTF7 = Find_Strings().FindByName("UTF-7");
-   CxList response = All.FindByName("Response.Charset");
-
-   UTF7 = response.DataInfluencedBy(UTF7);
-
-   if(UTF7.Count > 0)
-   {
-       result = Find_XSS_Outputs().DataInfluencedBy(Find_Interactive_Inputs());
-   }
-}
+//This query is deprecated.
```

VbNet / VbNet_Low_Visibility / Blind_SQL_Injections

Code changes

```
---
+++
@@ -1,8 +1 @@
-CxList db = Find_SQL_DB_In();
-CxList db_not_in_try = Improper_Exception_Handling(db);
-CxList db_in_try = db - db_not_in_try;
-
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitized = Find_Sanitize();
-
-result = All.FindSQLInjections(inputs, db_in_try, sanitized);
+//This query is deprecated.
```

VbNet / VbNet_Low_Visibility / Cleansing_Canonicalization_and_Comparison_Errors

Code changes

```
---
+++
@@ -1,22 +1 @@
-CxList inputs = Find_Interactive_Inputs();
-CxList obj = All.FindByType(typeof(UnknownReference));
-obj.Add(All.FindByType(typeof(Declarator)));
```

```
-CxList files = obj.FindByType("FileStream", false);
-files.Add(obj.FindByType("FileInfo", false));
-CxList file = All.FindByName("*File.*", false);
-CxList whitelist = All.FindByType(typeof(MemberAccess));
-//words that contain file in it
-CxList whitelistTarget = All.FindByShortNames(new List<string>{"Profile","Defilements","Defilement","Alfilerias"
-       ,"Filefishes","Interfiled","Interfiles","Alfileria","Filenames","Interfile","Profilers","Undefiled","Defilers","Fileable",
-"Filefish","Filename","Fileting","Misfiled","Misfiles","Prefiled","Prefiles","Profiled","Profiler","Profiles","Subfiles",
-"Defiled","Defiler","Defiles","Filemot","Fileted","Misfile","Prefile","Refiled","Refiles","Subfile","Defile","Filers",
-       "Filets","Refile","Filed","Filer","Filet"}, false);
-
-//Heuristical number 3 to get up to 3 members of a target
-//ex: Profile.Address.Street.Number
-file -= whitelist.GetMembersWithTargets(whitelistTarget, 3);
-files.Add(file);
-
-CxList sanitize = All.FindByName("*Server.MapPath", false);
-sanitize.Add(All.FindByName("*Request.MapPath", false));
-result = files.InfluencedByAndNotSanitized(inputs, sanitize);
+//This query is deprecated.
```

VbNet / VbNet_Low_Visibility / Improper_Transaction_Handling

Code changes

```
---
+++
@@ -1,29 +1,24 @@
 CxList Commit = All.FindByName("*.Commit", false);
 CxList Rollback = All.FindByName("*.Rollback", false);

-CxList TryBlock = Commit.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList TryBlock = Commit.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList cml in TryBlock)
 {
    try{
        TryCatchFinallyStmt TryGraph = cml.TryGetCSharpGraph<TryCatchFinallyStmt>();
-
+
        CxList curTry = All.FindById(TryGraph.Try.NodeId);

        CxList curCatch = All.NewCxList();
        if(TryGraph.CatchClauses != null && TryGraph.CatchClauses.Count > 0)
-       {
            curCatch = All.FindById(TryGraph.CatchClauses[0].NodeId);
-       }

        CxList CommitInTry = Commit.GetByAncs(curTry);
        CxList RollbackInCatch = Rollback.GetByAncs(curCatch);
```

```
-
-       if( (RollbackInCatch.GetAncOfType(typeof(TryCatchFinallyStmt)) *
-           CommitInTry.GetAncOfType(typeof(TryCatchFinallyStmt))).Count == 0)
-       {
+       if( (RollbackInCatch.GetAncOfType<TryCatchFinallyStmt>() *
+           CommitInTry.GetAncOfType<TryCatchFinallyStmt>()).Count == 0)
            result.Add(cml);
-       }
    }
    catch (Exception ex){
        cxLog.WriteDebugMessage(ex);
```

VbNet / VbNet_Low_Visibility / JavaScript_Hijacking

Code changes

```
---
+++
@@ -1,12 +1 @@
-//DWR framework prevents javascript hijacking
-CxList dwrFramework= All.FindByName("*dwr.util*",true);
-
-CxList CleanAJAXFramework=dwrFramework;// we'll add other frameworks that take care of javascript hijacking to this lis
-
-if (CleanAJAXFramework.Count==0)
-{
-   CxList db=Find_DB_Out().DataInfluencedBy(All.FindByName("*select*",false)+All.FindByName("*exec*",false));
-   CxList jason=All.FindByName("*json*",false);
-   jason=jason.DataInfluencedBy(db);
-   result= jason;
-}
+//This query is deprecated.
```

VbNet / VbNet_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,38 +1,37 @@
 CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null", false);
+CxList nullStrings = All.FindByName("null", false);
 CxList psw = Find_Passwords();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 // Find password in an initialization operation
 CxList psw_in_lSide = psw.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList psw_in_lSide_decl = psw_in_lSide.FindByType(typeof(Declarator));
+CxList psw_in_lSide_decl = psw_in_lSide.FindByType<Declarator>();

-CxList strLiterals = Find_Strings() - emptyString - NULL;
```

```
+CxList strLiterals = Find_Strings();
+strLiterals -= emptyString;
+strLiterals -= nullStrings;

 //when the hardcoded string includes a space or dot we believe
 //it is not a password string
-CxList strToREmove = strLiterals.FindByName("* *");
-strToREmove.Add(strLiterals.FindByName("*.*"));
-strToREmove.Add(strLiterals.FindByName("*/*"));
-strToREmove.Add(strLiterals.FindByName("*\\*"));
-strLiterals -= strToREmove;
+strLiterals -= strLiterals.FindByNames("* *", "*.*", "*/*", "*\\*");;
+strLiterals = strLiterals.FilterByDomProperty<StringLiteral>(x => x.Text.Length > 3);

 //strings as assignment in initialization   ==> Dim pass As String =  "goodbye"
 CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
 //add string parametes of StringBuilder to initialization ==> Dim password As New StringBuilder("abcd")
-CxList StringBuilders = All.FindByType(typeof (ObjectCreateExpr)).FindByShortName("StringBuilder", false);
-CxList allParams = All.FindByType(typeof (Param));
-CxList allStringsParams = strLiterals.GetFathers() * allParams.GetParameters(StringBuilders);
+CxList StringBuilders = Find_ObjectCreations().FindByShortName("StringBuilder", false);
+CxList allStringsParams = strLiterals.GetFathers() * paramValue.GetParameters(StringBuilders);
 lit_in_rSide.Add(allStringsParams);

 CxList initializePassword = psw_in_lSide_decl.FindByInitialization(lit_in_rSide);

 // Find password in an "equals" operation
 CxList eq = All.FindByMemberAccess("String.Equals", false);
-CxList eqWithTwoParams = eq;
+CxList eqWithTwoParams = All.NewCxList(eq);
 eq *= psw.GetMembersOfTarget();
 CxList equalsPassword = strLiterals.GetByAncs(eq);

-eq = eqWithTwoParams;
+eq = All.NewCxList(eqWithTwoParams);
 eq *= strLiterals.GetMembersOfTarget();
 equalsPassword.Add(psw.GetByAncs(eq));

@@ -53,10 +52,7 @@
 }

 // Find a password in a simple assignment
-CxList assignPassword = psw_in_lSide.GetAncOfType(typeof(AssignExpr));
+CxList assignPassword = psw_in_lSide.GetAncOfType<AssignExpr>();
 assignPassword = lit_in_rSide.GetByAncs(assignPassword);

-result = initializePassword;
-result.Add(equalsPassword);
```

```
-result.Add(assignPassword);
-result.Add(hPassInEq2Params);
+result.Add(initializePassword, equalsPassword, assignPassword, hPassInEq2Params);
```

VbNet / VbNet_Low_Visibility / XSS_Evasion_Attack

Code changes

```
---
+++
@@ -1,8 +1 @@
-if(All.isWebApplication)
-{
-   CxList decode = All.FindByName("*Server.HtmlDecode", false);
-   CxList sanitize = Find_XSS_Sanitize();
-   CxList output = Find_Interactive_Outputs();
-
-   result = output.InfluencedByAndNotSanitized(decode, sanitize);
-}
+//This query is deprecated.
```

VbNet / VbNet_Medium_Threat / DB_Parameter_Tampering

Code changes

```
---
+++
@@ -1,21 +1 @@
-CxList tables = All.FindByShortName("*orders*", false);
-tables.Add(All.FindByShortName("*credit*", false));
-tables.Add(All.FindByShortName("*invoice*", false));
-tables.Add(All.FindByShortName("*booking*", false));
-tables.Add(All.FindByShortName("*bill*", false));
-tables.Add(All.FindByShortName("*payment*", false));
-tables.Add(All.FindByShortName("*account*", false));
-tables.Add(All.FindByShortName("*cash*", false));
-tables.Add(All.FindByShortName("*customer*", false));
-
-CxList inputs = Find_Interactive_Inputs();
-CxList db = Find_DB_Base();
-
-CxList user = All.FindByShortName("*user*", false);
-user.Add(All.FindByShortName("*cust*", false));
-user.Add(All.FindByShortName("*member*", false));
-
-db = db.DataInfluencedBy(tables);
-db = db - db.DataInfluencedBy(user);
-
-result = inputs.InfluencingOnAndNotSanitized(db, Find_Parameters());
+//This query is deprecated.
```

VbNet / VbNet_Medium_Threat / HTTP_Response_Splitting

Code changes

```
---

+++

@@ -1,2 +1 @@

 //This query is deprecated.

-cxLog.WriteDebugMessage("The query HTTP_Response_Splitting is deprecated");
```

VbNet / VbNet_Medium_Threat / Path_Traversal

Code changes

```
---

+++

@@ -1,19 +1,20 @@

-CxList Inputs = Find_Interactive_Inputs();

-CxList Methods = Find_Methods();

+CxList inputs = Find_Interactive_Inputs();

+CxList methods = Find_Methods();

+CxList unkRefs = Find_UnknownReference();

+CxList obj = All.NewCxList(

+   unkRefs,

+   Find_Declarators(),

+   methods);


-CxList obj = All.FindByType(typeof(UnknownReference));

-obj.Add(All.FindByType(typeof(Declarator)));

-obj.Add(Methods);

+CxList files = All.NewCxList(

+   obj.FindByTypes(new [] {"*StreamReader", "*FileStream", "*FileInfo", "*DirectoryInfo"}, false),

+   obj.FindByMemberAccesses(new [] {"*File.*", "*Directory.*"}, false));

+files -= files.FindByMemberAccess("Profile.*", false);


-CxList files = obj.FindByType("*StreamReader", false);

-files.Add(obj.FindByType("*FileStream", false));

-files.Add(obj.FindByType("*FileInfo", false));

-files.Add(obj.FindByType("*DirectoryInfo", false));

-files.Add(obj.FindByMemberAccess("*File.*", false));

-files -= files.FindByMemberAccess("Profile.*", false);

-files.Add(obj.FindByMemberAccess("*Directory.*", false));

+CxList sanitized = All.NewCxList(

+   Find_Sanitize(),

+   methods.FindByMemberAccesses(new [] {"Path.GetFileName", "Path.GetFileNameWithoutExtension"}, false),

+   unkRefs.FindByShortName("Path", false).GetMembersOfTarget().FindByShortNames(

+   new [] {"GetFileName", "GetFileNameWithoutExtension"}, false));


-CxList sanitized = Find_Sanitize();

-sanitized.Add(Methods.FindByShortName("getfilename"));

-

-result = files.InfluencedByAndNotSanitized(Inputs, sanitized);
```

```
+result = files.InfluencedByAndNotSanitized(inputs, sanitized).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

VbNet / VbNet_Medium_Threat / Reflected_XSS_Specific_Clients

Code changes

```
---
+++
@@ -1,16 +1 @@
-if(All.isWebApplication)
-{
-   CxList inputs = Find_Interactive_Inputs();
-   CxList outputs = Find_Interactive_Outputs() - Find_XSS_Outputs();
-
-   outputs -= All.FindByMemberAccess("Textbox.Text", false);
-
-   CxList outpParam = All.GetParameters(outputs);
-   CxList sanitized = Find_XSS_Sanitize();
-   result = outpParam.InfluencedByAndNotSanitized(inputs, sanitized);
-   result = outputs.DataInfluencedBy(result, CxList.InfluenceAlgorithmCalculation.NewAlgorithm);
-
-   CxList outpMemAcs = outputs.FindByType(typeof(MemberAccess));
-   CxList outputsRes = outpMemAcs.InfluencedByAndNotSanitized(inputs, sanitized);
-   result.Add(outputsRes);
-}
+//This query is deprecated.
```

VbNet / VbNet_WebConfig / HttpOnlyCookies_XSS

Code changes

```
---
+++
@@ -5,7 +5,6 @@
 CxList httpCookies_exist = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.HTTPCOOKIES", false);
 CxList httpCookies_childs = webConfig.FindByName("CONFIGURATION.SYSTEM.WEB.HTTPCOOKIES.*", false);
 CxList httpOnlCookieysFalse = value_false.DataInfluencingOn(httpOnlCookies_exist);
-CxList configuration = webConfig.FindByName("CONFIGURATION", false);


 if (httpOnlCookies_exist.Count == 0)
 {
```

Lua / Lua_Low_Visibility / Missing_Password_Field_Masking

Code changes