Version 9.6.0 - Queries Release Notes

New Queries:

Language	Group	Name	CWE
CSharp	CSharp_APISecurity	CSharp_WebApi_GetApiList	0
Java	Java_APISecurity	Java_WebApi_GetApiList	0
JavaScript	JavaScript_APISecurity	NodeJS_Express_WebApi_GetApiList	0
Kotlin	Kotlin_Low_Visibility	Use_of_Unsafe_JNI	111
Kotlin	Kotlin_Medium_Threat	Cleartext_Submission_of_Sensitive_Information	319
Kotlin	Kotlin_Medium_Threat	DoS_by_Sleep	834
Kotlin	Kotlin_Medium_Threat	Excessive_Data_Exposure	201
Kotlin	Kotlin_Medium_Threat	Frameable_Login_Page	829
Kotlin	Kotlin_Medium_Threat	Hardcoded_password_in_Connection_String	547
Kotlin	Kotlin_Medium_Threat	Improper_Locking	667
Kotlin	Kotlin_Medium_Threat	Missing_HSTS_Header	346
Kotlin	Kotlin_Medium_Threat	Missing_Secure_In_Code	614
Kotlin	Kotlin_Medium_Threat	Parameter_Tampering	472
Kotlin	Kotlin_Medium_Threat	Privacy_Violation	359
Kotlin	Kotlin_Medium_Threat	Reliance_on_Cookies_without_Validation	565
Kotlin	Kotlin_Medium_Threat	SSRF	918
Kotlin	Kotlin_Medium_Threat	Unsafe_Object_Binding	915
Kotlin	Kotlin_Spring	Spring_ModelView_Injection	74
Lua	Lua_High_Risk	Arbitrary_File_Write	669
Lua	Lua_High_Risk	Code_Injection	74
Lua	Lua_High_Risk	Command_Injection	77
Lua	Lua_High_Risk	Connection_String_Injection	99
Lua	Lua_High_Risk	Dangerous_File_Inclusion	829
Lua	Lua_High_Risk	Insufficiently_Secure_Password_Storage_Algorithm_Parameters	522
Lua	Lua_High_Risk	JWT_No_Signature_Verification	287
Lua	Lua_High_Risk	Reflected_XSS_All_Clients	79
Lua	Lua_High_Risk	Resource_Injection	99
Lua	Lua_High_Risk	Second_Order_SQL_Injection	89
Lua	Lua_High_Risk	SQL_Injection	89
Lua	Lua_High_Risk	Stored_Code_Injection	74
Lua	Lua_High_Risk	Stored_Command_Injection	77
Lua	Lua_High_Risk	Stored_XSS	79
Lua	Lua_Low_Visibility	Command_Argument_Injection	78
Lua	Lua_Low_Visibility	Cookie_Overly_Broad_Path	539
Lua	Lua_Low_Visibility	Dangerous_File_Extension	434
Lua	Lua_Low_Visibility	Empty_password_in_Connection_String	521
Lua	Lua_Low_Visibility	Exposure_of_System_Data	497
Lua	Lua_Low_Visibility	Hardcoded_AWS_Credentials	798
Lua	Lua_Low_Visibility	Improper_Exception_Handling	248
Lua	Lua_Low_Visibility	Improper_Resource_Shutdown_or_Release	404
Lua	Lua_Low_Visibility	Information_Exposure_Through_Server_Log	359
Lua	Lua_Low_Visibility	Insufficient_Session_Expiration	613
Lua	Lua_Low_Visibility	JWT_Excessive_Expiration_Time	613
Lua	Lua_Low_Visibility	JWT_No_Expiration_Time_Validation	613

Language	Group	Name	CWE
Lua	Lua_Low_Visibility	JWT_No_NotBefore_Validation	304
Lua	Lua_Low_Visibility	Leaving_Temporary_File	377
Lua	Lua_Low_Visibility	Missing_Framing_Policy	1021
Lua	Lua_Low_Visibility	Missing_Password_Field_Masking_Node	549
Lua	Lua_Low_Visibility	Null_Pointer_Dereference	457
Lua	Lua_Low_Visibility	Overly_Permissive_Cross_Origin_Resource_Sharing_Policy	284
Lua	Lua_Low_Visibility	Password_In_Comment	615
Lua	Lua_Low_Visibility	Path_Traversal_Evasion_Attack_via_Replace	36
Lua	Lua_Low_Visibility	Reliance_on_DNS_Lookups_in_a_Decision	350
Lua	Lua_Low_Visibility	Stored_Command_Argument_Injection	78
Lua	Lua_Low_Visibility	Trust_Boundary_Violation_in_Session_Variables	501
Lua	Lua_Low_Visibility	Uncontrolled_Format_String	74
Lua	Lua_Low_Visibility	Unrestricted_Read_S3	639
Lua	Lua_Low_Visibility	Use_Of_Hardcoded_Password	259
Lua	Lua_Low_Visibility	Use_of_Non_Cryptographic_Random	338
Lua	Lua_Low_Visibility	Using_Referer_Field_for_Authentication	287
Lua	Lua_Low_Visibility	XSS_Evasion_Attack_via_Replace	79
Lua	Lua_Medium_Threat	Absolute_Path_Traversal	36
Lua	Lua_Medium_Threat	Broken_or_Risky_Encryption_Algorithm	327
Lua	Lua_Medium_Threat	Broken_or_Risky_Hashing_Function	327
Lua	Lua_Medium_Threat	DoS_by_Sleep	834
Lua	Lua_Medium_Threat	DoS_from_Evil_Regex	400
Lua	Lua_Medium_Threat	DoS_from_RegEx_Injection	400
Lua	Lua_Medium_Threat	Encoding_Used_Instead_of_Encryption	311
Lua	Lua_Medium_Threat	Hardcoded_Cryptographic_IV	326
Lua	Lua_Medium_Threat	Hardcoded_Cryptographic_Key	321
Lua	Lua_Medium_Threat	Hardcoded_password_in_Connection_String	547
Lua	Lua_Medium_Threat	Hardcoded_Salt	760
Lua	Lua_Medium_Threat	HttpOnly_Cookie_Flag_Not_Set	1004
Lua	Lua_Medium_Threat	Insecure_Asymmetric_Cryptographic_Algorithm_Parameters	326
Lua	Lua_Medium_Threat	Insecure_Value_of_the_SameSite_Cookie_Attribute	1275
Lua	Lua_Medium_Threat	JWT_Lack_of_Expiration_Time	613
Lua	Lua_Medium_Threat	JWT_Sensitive_Information_Exposure	359
Lua	Lua_Medium_Threat	JWT_Use_Of_Hardcoded_Secret	798
Lua	Lua_Medium_Threat	Misconfigured_HSTS_Header	346
Lua	Lua_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311
Lua	Lua_Medium_Threat	Open_Redirect	601
Lua	Lua_Medium_Threat	Parameter_Tampering	472
Lua	Lua_Medium_Threat	Plaintext_Storage_of_a_Password	256
Lua	Lua_Medium_Threat	Privacy_Violation	359
Lua	Lua_Medium_Threat	Race_Condition	366
Lua	Lua_Medium_Threat	Relative_Path_Traversal	23
Lua	Lua_Medium_Threat	Secure_Cookie_Flag_Not_Set	614
Lua	Lua_Medium_Threat	Sensitive_Information_Exposure_in_Cleartext_Channel	319
Lua	Lua_Medium_Threat	Session_Fixation	384
Lua	Lua_Medium_Threat	SSL_Verification_Bypass	599
Lua	Lua_Medium_Threat	SSRF	74

Language	Group	Name	CWE
Lua	Lua_Medium_Threat	Stored_Absolute_Path_Traversal	36
Lua	Lua_Medium_Threat	Stored_Dangerous_File_Inclusion	829
Lua	Lua_Medium_Threat	Stored_Relative_Path_Traversal	23
Lua	Lua_Medium_Threat	Unchecked_Input_for_Loop_Condition	606
Lua	Lua_Medium_Threat	Uncontrolled_Memory_Allocation	789
Lua	Lua_Medium_Threat	Use_of_Cryptographically_Weak_PRNG	329
PHP	PHP_High_Risk	Absolute_Path_Traversal	36
PHP	PHP_High_Risk	Dangerous_File_Inclusion	829
PHP	PHP_High_Risk	Dangerous_File_Upload	434
PHP	PHP_High_Risk	Insufficiently_Secure_Password_Storage_Algorithm_Parameters	522
PHP	PHP_High_Risk	JWT_No_Signature_Verification	347
PHP	PHP_High_Risk	JWT_Use_Of_None_Algorithm	287
PHP	PHP_High_Risk	MongoDB_NoSQL_Injection	74
PHP	PHP_High_Risk	Server_Side_Template_Injection	1336
PHP	PHP_High_Risk	Stored_Absolute_Path_Traversal	36
PHP	PHP_Low_Visibility	Hardcoded_Public_Key	321
PHP	PHP_Low_Visibility	JWT_Excessive_Expiration_Time	613
PHP	PHP_Low_Visibility	JWT_No_Expiration_Time_Validation	613
PHP	PHP_Low_Visibility	JWT_No_NotBefore_Validation	304
PHP	PHP_Medium_Threat	Broken_or_Risky_Encryption_Algorithm	327
PHP	PHP_Medium_Threat	Broken_or_Risky_Hashing_Function	327
PHP	PHP_Medium_Threat	DoS_from_Evil_Regex	400
PHP	PHP_Medium_Threat	DoS_from_Regex_Injection	74
PHP	PHP_Medium_Threat	Encoding_Used_Instead_of_Encryption	311
PHP	PHP_Medium_Threat	Hardcoded_Salt	760
PHP	PHP_Medium_Threat	Hashing_Length_Extension_Attack	310
PHP	PHP_Medium_Threat	HttpOnly_Cookie_Flag_Not_Set	1004
PHP	PHP_Medium_Threat	HttpOnly_Cookie_Flag_Not_Set_In_Config	1004
PHP	PHP_Medium_Threat	Insecure_Asymmetric_Cryptographic_Algorithm_Parameters	326
PHP	PHP_Medium_Threat	Insecure_Value_of_the_SameSite_Cookie_Attribute_In_Code	1275
PHP	PHP_Medium_Threat	Insecure_Value_of_the_SameSite_Cookie_Attribute_In_Config	1275
PHP	PHP_Medium_Threat	Insecure_WebSocket_Connection	1385
PHP	PHP_Medium_Threat	JWT_Lack_Of_Expiration_Time	613
PHP	PHP_Medium_Threat	JWT_Sensitive_Information_Exposure	201
PHP	PHP_Medium_Threat	JWT_Use_Of_Hardcoded_Secret	798
PHP	PHP_Medium_Threat	Loose_Comparison	480
PHP	PHP_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311
PHP	PHP_Medium_Threat	Plaintext_Storage_of_a_Password	256
PHP	PHP_Medium_Threat	Regex_Filter_Bypass	358
PHP	PHP_Medium_Threat	Relative_Path_Traversal	23
PHP	PHP_Medium_Threat	Secure_Cookie_Flag_Not_Set	614
PHP	PHP_Medium_Threat	Secure_Cookie_Flag_Not_Set_In_Config	614
PHP	PHP_Medium_Threat	SSRF	918
PHP	PHP_Medium_Threat	SSTI_Twig	1336
PHP	PHP_Medium_Threat	Storage_Controlled_Dynamic_Variable	914
PHP	PHP_Medium_Threat	Stored_Dangerous_File_Inclusion	829
PHP	PHP_Medium_Threat	Stored_DoS_from_Evil_Regex	400

Language	Group	Name	CWE
PHP	PHP_Medium_Threat	Stored_DoS_from_Regex_Injection	74
PHP	PHP_Medium_Threat	Stored_Relative_Path_Traversal	23
PHP	PHP_Medium_Threat	Unchecked_Input_for_Loop_Condition	606
PHP	PHP_Medium_Threat	Use_of_Cryptographically_Weak_PRNG	338
PHP	PHP_Medium_Threat	Use_of_Hardcoded_Cryptographic_IV_in_Server	547
PHP	PHP_Medium_Threat	Value_Shadowing	233
Python	Python_APISecurity	Python_Django_WebApi_GetApiList	0
Python	Python_APISecurity	Python_Flask_WebApi_GetApiList	0
Scala	Scala_High_Risk	Expression_Language_Injection_MVEL	917
Scala	Scala_High_Risk	Expression_Language_Injection_SPEL	917
Scala	Scala_Low_Visibility	JWT_Excessive_Expiration_Time	613
Scala	Scala_Low_Visibility	JWT_Use_Of_None_Algorithm	287
Scala	Scala_Low_Visibility	Use_of_Unsafe_JNI	111
Scala	Scala_Medium_Threat	Excessive_Data_Exposure	201
Scala	Scala_Medium_Threat	JWT_Lack_Of_Expiration_Time	613
Scala	Scala_Medium_Threat	JWT_No_Signature_Verification	287
Scala	Scala_Medium_Threat	JWT_Sensitive_Information_Exposure	201
Scala	Scala_Medium_Threat	JWT_Use_Of_Hardcoded_Secret	321
Scala	Scala_Medium_Threat	Spring_ModelView_Injection	74
Scala	Scala_Medium_Threat	Stored_Command_Injection	77
Scala	Scala_Medium_Threat	Unvalidated_Forwards	601

Changed Queries:

Language	Group	Name	CWE	Changed Fields
CPP	CPP_Buffer_Overflow	Improper_Null_Termination	170	**Source** has changed
CPP	CPP_Medium_Threat	Wrong_Memory_Allocation	131	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Aptca_Methods_Call_Non_Aptca_Methods	0	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Dynamic_SQL_Queries	89	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Exposure_of_Resource_to_Wrong_Sphere	493	**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	Missing_XML_Validation	112	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Pages_Without_Global_Error_Handler	544	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Routed_Deprecated_Code	477	**Source** has changed
CSharp	CSharp_Best_Coding_Practice	Unchecked_Error_Condition	391	**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	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_Parameter_Tampering	472	**Source** has changed
CSharp	CSharp_Heuristic	Heuristic_SQL_Injection	89	**Source** has changed
CSharp	CSharp_High_Risk	Connection_String_Injection	99	**Source** has changed
CSharp	CSharp_High_Risk	Deserialization_of_Untrusted_Data_MSMQ	502	**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

Language	Group	Name	CWE	Changed Fields
CSharp	CSharp_High_Risk	Resource_Injection	99	**Source** has changed
CSharp	CSharp_High_Risk	Second_Order_SQL_Injection	89	**Source** has changed
CSharp	CSharp_High_Risk	Stored_XSS	79	**Source** has changed
CSharp	CSharp_High_Risk	Unsafe_Reflection	470	**Source** has changed
CSharp	CSharp_Low_Visibility	Cross_Site_History_Manipulation	203	**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	Information_Exposure_Through_an_Error_Message	209	**Source** has changed
CSharp	CSharp_Low_Visibility	Information_Leak_Through_Persistent_Cookies	539	**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	Missing_Function_Level_Authorization	862	**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	Password_In_Comment	615	**Source** has changed
CSharp	CSharp_Low_Visibility	Potential_ReDoS_In_Static_Field	400	**Source** has changed
CSharp	CSharp_Low_Visibility	Reliance_on_DNS_Lookups_in_a_Decision	350	**Source** has changed
CSharp	CSharp_Low_Visibility	Stored_Command_Argument_Injection	88	**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_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	Data_Filter_Injection	943	**Source** has changed
CSharp	CSharp_Medium_Threat	DoS_by_Sleep	834	**Source** has changed
CSharp	CSharp_Medium_Threat	Excessive_Data_Exposure	201	**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
CSharp	CSharp_Medium_Threat	Improper_Restriction_of_XXE_Ref	611	**Source** has changed
CSharp	CSharp_Medium_Threat	Insecure_Cookie	614	**Source** has changed
CSharp	CSharp_Medium_Threat	Insufficient_Connection_String_Encryption	522	**Source** has changed
CSharp	CSharp_Medium_Threat	JWT_Sensitive_Information_Exposure	201	**Source** has changed
CSharp	CSharp_Medium_Threat	Missing_Object_Level_Authorization	862	**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	Parameter_Tampering	472	**Source** has changed
CSharp	CSharp_Medium_Threat	Path_Traversal	22	**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	ReDoS_By_Regex_Injection	400	**Source** has changed

Language	Group	Name	CWE	Changed Fields
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_Command_Injection	77	**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_WebConfig	CookieLess_Authentication	642	**Source** has changed
CSharp	CSharp_WebConfig	DebugEnabled	11	**Source** has changed
CSharp	CSharp_WebConfig	Elmah_Enabled	213	**Source** has changed
CSharp	CSharp_WebConfig	HardcodedCredentials	489	**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
Dart	Dart_Mobile_Low_Visibility	Use_of_Non_Cryptographic_Random	330	**Source** has changed
Dart	Dart_Mobile_Medium_Threat	Use_of_Cryptographically_Weak_PRNG	338	**Source** has changed
Go	Go_Medium_Threat	Missing_HttpOnly_Cookie	1004	**Source** has changed
Go	Go_Medium_Threat	Missing_Secure_Cookie	614	**Source** has changed
Java	Java_High_Risk	Code_Injection	94	**Source** has changed
Java	Java_High_Risk	Connection_String_Injection	99	**Source** has changed
Java	Java_High_Risk	JSF_Local_File_Inclusion	98	**Source** has changed
Java	Java_High_Risk	Second_Order_SQL_Injection	89	**Source** has changed
Java	Java_High_Risk	Stored_XSS	79	**Source** has changed
Java	Java_High_Risk	Unsafe_Reflection	470	**Source** has changed
Java	Java_Low_Visibility	Leaving_Temporary_File	376	DescriptionId changed from 342 to **4145**
Java	Java_Medium_Threat	CGI_Stored_XSS	79	**Source** has changed
Java	Java_Medium_Threat	Dangerous_File_Inclusion	829	**Source** has changed
Java	Java_Medium_Threat	Improper_Restriction_of_XXE_Ref	611	**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	Stored_Command_Injection	77	**Source** has changed
Java	Java_Medium_Threat	Stored_LDAP_Injection	90	**Source** has changed
Java	Java_Medium_Threat	XQuery_Injection	652	**Source** has changed
JavaScript	JavaScript_High_Risk	Client_DOM_Code_Injection	94	**Source** has changed
JavaScript	JavaScript_High_Risk	Client_DOM_Stored_Code_Injection	94	**Source** has changed
JavaScript	JavaScript_High_Risk	Prototype_Pollution	1321	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Unsafe_Use_Of_Target_blank	1022	**Source** has changed
JavaScript	JavaScript_Medium_Threat	CSV_Injection	74	**Source** has changed
JavaScript	JavaScript_Medium_Threat	XML_External_Entities_XXE	611	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Absolute_Path_Traversal	36	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Cleartext_Storage_Of_Sensitive_Information	312	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Code_Injection	94	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Command_Injection	77	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Cookie_Poisoning	472	**Source** has changed

Language	Group	Name	CWE	Changed Fields
JavaScript	JavaScript_Server_Side_Vulnerabilities	Excessive_Data_Exposure	201	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Insecure_Storage_of_Sensitive_Data	933	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	MongoDB_NoSQL_Injection	89	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Parameter_Tampering	472	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Plaintext_Storage_of_a_Password	256	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Privacy_Violation	359	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Reflected_XSS	79	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Second_Order_SQL_Injection	89	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	SQL_Injection	89	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Stored_Code_Injection	94	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Stored_Path_Traversal	22	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Stored_XSS	79	**Source** has changed
Kotlin	Kotlin_High_Risk	Deserialization_of_Untrusted_Data	502	**Source** has changed
Kotlin	Kotlin_Medium_Threat	Use_of_Hardcoded_Cryptographic_Key	321	**Source** has changed
PHP	PHP_Best_Coding_Practice	Declaration_Of_Catch_For_Generic_Exception	396	Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Detection_of_Error_Condition_Without_Action	390	Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Exposure_of_Resource_to_Wrong_Sphere	493	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Hardcoded_Absolute_Path	426	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Possible_Global_Variable_Overwrite	0	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Unchecked_Error_Condition	391	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Unclosed_Objects	459	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Use_Of_Namespace	398	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Use_Of_Private_Static_Variable	398	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_Best_Coding_Practice	Use_Of_Super_GLOBALS	766	**Source** has changed, Group changed from Php_Best_Coding_Practice to **PHP_Best_Coding_Practice**
PHP	PHP_High_Risk	Code_Injection	94	**Source** has changed
PHP	PHP_High_Risk	LDAP_Injection	90	**Source** has changed
PHP	PHP_High_Risk	Second_Order_SQL_Injection	89	**Source** has changed
PHP	PHP_High_Risk	SQL_Injection	89	**Source** has changed
PHP	PHP_High_Risk	Stored_XSS	79	**Source** has changed
PHP	PHP_High_Risk	XPath_Injection	643	**Source** has changed
PHP	PHP_Low_Visibility	Deprecated_Functions	477	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Improper_Exception_Handling	248	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Improper_Transaction_Handling	460	DescriptionId changed from 336 to **4165**, **Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Information_Exposure_Through_an_Error_Message	209	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Information_Leak_Through_Persistent_Cookies	539	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Log_Forging	117	DescriptionId changed from 18 to **3058**, **Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Reliance_on_DNS_Lookups_in_a_Decision	350	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Trust_Boundary_Violation_in_Session_Variables	501	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Unsafe_Use_Of_Target_Blank	1022	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed, Group changed from Php_Low_Visibility to **PHP_Low_Visibility**
PHP	PHP_Medium_Threat	CSRF	352	**Source** has changed
PHP	PHP_Medium_Threat	DoS_by_Sleep	834	**Source** has changed
PHP	PHP_Medium_Threat	Header_Injection	113	**Source** has changed
PHP	PHP_Medium_Threat	Improper_Restriction_of_Stored_XXE_Ref	611	**Source** has changed
PHP	PHP_Medium_Threat	Improper_Restriction_of_XXE_Ref	611	**Source** has changed
PHP	PHP_Medium_Threat	Missing_HSTS_Header	346	**Source** has changed

Language	Group	Name	CWE	Changed Fields
PHP	PHP_Medium_Threat	Open_Redirect	601	**Source** has changed
PHP	PHP_Medium_Threat	Parameter_Tampering	472	**Source** has changed
PHP	PHP_Medium_Threat	Privacy_Violation	359	**Source** has changed
PHP	PHP_Medium_Threat	Session_Fixation	384	**Source** has changed
PHP	PHP_Medium_Threat	SSL_Verification_Bypass	599	DescriptionId changed from 125 to **3086**, **Source** has changed
PHP	PHP_Medium_Threat	Stored_Code_Injection	94	**Source** has changed
PHP	PHP_Medium_Threat	Stored_LDAP_Injection	90	**Source** has changed
PHP	PHP_Medium_Threat	Use_of_Hard_coded_Cryptographic_Key	321	**Source** has changed
PHP	PHP_High_Risk	Deserialization_of_Untrusted_Data	502	Severity changed from Medium to **High**, **Source** has changed, Group changed from PHP_Medium_Threat to **PHP_High_Risk**
PHP	PHP_High_Risk	Reflected_XSS	79	Name changed from Reflected_XSS_All_Clients to **Reflected_XSS**, **Source** has changed
PHP	PHP_High_Risk	Stored_XPath_Injection	643	Severity changed from Medium to **High**, **Source** has changed, Group changed from PHP_Medium_Threat to **PHP_High_Risk**
PHP	PHP_High_Risk	Unsafe_Reflection	470	Name changed from Reflection_Injection to **Unsafe_Reflection**, **Source** has changed
PHP	PHP_Low_Visibility	Insufficient_Sanitization_for_XSS	838	Name changed from Inappropriate_Encoding_for_Output_Context to **Insufficient_Sanitization_for_XSS**, Severity changed from Medium to **Low**, **Source** has changed, Group changed from PHP_Medium_Threat to **PHP_Low_Visibility**
PHP	PHP_Low_Visibility	Use_of_Non_Cryptographic_Random	330	Name changed from Insecure_Randomness to **Use_of_Non_Cryptographic_Random**, Severity changed from Medium to **Low**, **Source** has changed, Group changed from PHP_Medium_Threat to **PHP_Low_Visibility**
PHP	PHP_Medium_Threat	Stored_Unsafe_Reflection	470	Name changed from Stored_Reflection_Injection to **Stored_Unsafe_Reflection**, **Source** has changed
PHP	PHP_Medium_Threat	User_Controlled_Dynamic_Variable	914	Name changed from Improper_Control_of_Dynamically_Identified_Variables to **User_Controlled_Dynamic_Variable**, **Source** has changed
Python	Python_Low_Visibility	Log_Forging	117	DescriptionId changed from 18 to **2247**
Ruby	Ruby_Best_Coding_Practice	Dynamic_Render_Path	10714	**Source** has changed
Scala	Scala_Medium_Threat	Stored_External_XML_Entities_XXE	611	**Source** has changed
Scala	Scala_Medium_Threat	Use_of_Hardcoded_Cryptographic_Key	321	Name changed from Use_of_Hard_coded_Cryptographic_Key to **Use_of_Hardcoded_Cryptographic_Key**, **Source** has changed
Swift	Swift_Low_Visibility	Null_Password	252	**Source** has changed
Swift	Swift_Medium_Threat	Pasteboard_Leakage	200	**Source** has changed
Swift	Swift_Medium_Threat	ReDoS	1333	**Source** has changed

Deleted Queries:

Language	Group	Name	CWE
Lua	Lua_Best_Coding_Practice	Empty_Methods	398
PHP	Php_Best_Coding_Practice	Dynamic_SQL_Queries	89
PHP	PHP_High_Risk	File_Disclosure	538
PHP	PHP_High_Risk	File_Inclusion	98
PHP	PHP_High_Risk	File_Manipulation	552
PHP	PHP_High_Risk	Remote_File_Inclusion	829
PHP	Php_Low_Visibility	Blind_SQL_Injections	89
PHP	Php_Low_Visibility	Cross_Site_History_Manipulation	203
PHP	Php_Low_Visibility	ESAPI_Same_Password_Repeats_Twice	521
PHP	Php_Low_Visibility	Incorrect_Implementation_of_Authentication_Algorithm	303
PHP	Php_Low_Visibility	Insufficiently_Protected_Credentials	522
PHP	Php_Low_Visibility	Possible_Flow_Control	691
PHP	Php_Low_Visibility	Reliance_on_Cookies_in_a_Decision	784
PHP	Php_Low_Visibility	Use_of_Broken_or_Risky_Cryptographic_Algorithm	327
PHP	Php_Low_Visibility	XSS_Evasion_Attack	79
PHP	PHP_Medium_Threat	DB_Parameter_Tampering	284
PHP	PHP_Medium_Threat	HttpOnlyCookies	1004
PHP	PHP_Medium_Threat	HTTP_Response_Splitting	113
PHP	PHP_Medium_Threat	Improper_Neutralization_of_SQL_Command	89
PHP	PHP_Medium_Threat	Object_Injection	502

Language	Group	Name	CWE
PHP	PHP_Medium_Threat	Path_Traversal	22
PHP	PHP_Medium_Threat	Reflected_File_Download	425
PHP	PHP_Medium_Threat	Stored_File_Inclusion	98
PHP	PHP_Medium_Threat	Stored_File_Manipulation	552
PHP	PHP_Medium_Threat	Stored_Remote_File_Inclusion	98

Changed Source:

CPP / CPP_Buffer_Overflow / Improper_Null_Termination

Code changes

```
---
+++
@@ -4,43 +4,51 @@
 CxList unkRefs = Find_Unknown_References();
 CxList declarators = Find_Declarators();
 CxList allArrayInitializer = Find_ArrayInitializer();
-CxList allRelevantTypes = Find_Builtin_Char_Types();
-allRelevantTypes.Add(Find_Builtin_Char_Microsoft_Types());

-CxList declsParams = All.NewCxList();
-declsParams.Add(declarators, Find_ParamDecl());
+CxList allRelevantTypes = All.NewCxList(
+   Find_Builtin_Char_Types(),
+   Find_Builtin_Char_Microsoft_Types()
+   );

-CxList supportedParamTypes = declsParams.FindByTypes(new string[] {"char", "png_bytep", "OLECHAR", "wchar_t", "TCHAR", "void"});
-CxList supportedParamTypesRefs = unkRefs.FindAllReferences(supportedParamTypes);
-
+CxList declsParams = All.NewCxList(
+   declarators,
+   Find_ParamDecl()
+   );
+
 // Common functions that do not copy the null terminator.
 List<string> methodWithoutNullTermStr = new List<string>{
-       "strncpy","wcsncpy","fread","mbstowcs", "mbsrtowcs","wcstombs",
-       "wcsrtombs","wmemcpy","wmemmove","_memccpy",
-       "_strncpy_l","_tcsncpy","_tcsncpy_l","_tcsncat","_tcsncat_l"
+       "strncpy","wcsncpy","mbstowcs", "mbsrtowcs","wcstombs",
+       "wcsrtombs", "_strncpy_l","_tcsncpy","_tcsncpy_l","_tcsncat","_tcsncat_l"
        };
+
 CxList methodWithoutNullTerm = methods.FindByShortNames(methodWithoutNullTermStr);
-
 CxList declWithNullTerm = allArrayInitializer.GetAncOfType<Declarator>();
```

```
 // Add other types of parameters references
-supportedParamTypesRefs.Add(Find_CastExpr(), Find_Unarys(), Find_IndexerRefs(),
-   Find_MemberAccesses(), Find_BinaryExpr());
+CxList supportedParamTypes = declsParams.FindByTypes(new string[] {"char", "png_bytep", "OLECHAR", "wchar_t", "TCHAR", "void"});
+CxList supportedParamTypesRefs = All.NewCxList(
+   unkRefs.FindAllReferences(supportedParamTypes),
+   Find_CastExpr(),
+   Find_Unarys(),
+   Find_IndexerRefs(),
+   Find_MemberAccesses(),
+   Find_BinaryExpr());

 CxList charToCheckReqNullTerm = supportedParamTypesRefs.GetParameters(methodWithoutNullTerm, 0);

-CxList declUnk = All.NewCxList();
-declUnk.Add(unkRefs, declarators);
+CxList declUnk = All.NewCxList(unkRefs, declarators);
 allRelevantTypes.Add(declUnk.FindByPointerTypes(new string[] {"char*", "BYTE*", "ValueType.Ch" }));

 CxList relevantParams = unkRefs.FindAllReferences(allRelevantTypes);

-CxList methodsToCheck = methods.FindByShortNames("memcpy", "memmove", "fread");
+CxList methodsToCheck = methods.FindByShortNames("fread");
 CxList memCpyParams = relevantParams.GetParameters(methodsToCheck, 0);
-CxList funcsNames = methodsToCheck.GetAncOfType<MethodDecl>();
-funcsNames.Add(methodWithoutNullTerm.GetAncOfType<MethodDecl>());
-funcsNames.Add(methodWithoutNullTerm.GetAncOfType<ConstructorDecl>());
+CxList funcsNames = All.NewCxList(
+   methodsToCheck.GetAncOfType<MethodDecl>(),
+   methodWithoutNullTerm.GetAncOfType<MethodDecl>(),
+   methodWithoutNullTerm.GetAncOfType<ConstructorDecl>()
+   );

 // Sanitizers
 /*
@@ -93,6 +101,14 @@
 sanitizerRefs.Add(methods.FindByShortName("calloc").GetAssignee());
 sanitizerRefs = allRelevantParams.InfluencedBy(sanitizerRefs).GetLastNodesInPath();
 sanitizers.Add(allRelevantParams.FindAllReferences(sanitizerRefs));
+//All the char parameters that are declared as unsined should be removed (they are, probably, declared as unsigned to be used as numbers not chars of string)
+sanitizers.Add(allRelevantParams.FindByType("char").FindByTypeModifiers(TypeSignednessModifiers.Unsigned));
+
+// strncpy sanitizer => strnctyp(ret, sanitized), sprintf(sanitized, "value")
+CxList sanitizedParam = unkRefs.GetParameters(methods.FindByShortName("sprintf"), 0);
+CxList sanitizedStrncpy = unkRefs.FindAllReferences(sanitizedParam).GetParameters(methods.FindByShortName("strncpy"), 1).
+   GetAncOfType<MethodInvokeExpr>();
+sanitizers.Add(allRefsOfRelevantParams.GetParameters(sanitizedStrncpy, 0));
```

```
 // Flow and outputs
 result = charToCheckReqNullTerm;
```

CPP / CPP_Medium_Threat / Wrong_Memory_Allocation

Code changes

```
---
+++
@@ -1,43 +1,38 @@
-// Wrong Memory Allocation
-//  -----------------------
-//  In this query we look for calls to malloc with sizeof, where there
-//  is no correct multiplication of the parameter by ist size.
-////////////////////////////////////////////////////////////////////////
+//Wrong Memory Allocation
+//---------------------------
+//In this query we look for calls to malloc with sizeof, where there
+//is no correct multiplication of the parameter by ist size.
+//////////////////////////////////////////////////////////////////////


-// Find all mallocs
+//Find all mallocs
 CxList mallocs = Find_Methods().FindByShortName("malloc");
-// Find the sizeof functions that are direct parameters of malloc (no sizeof*somthing)
+//Find the sizeof functions that are direct parameters of malloc (no sizeof*somthing)
 CxList sizeofInMalloc = Find_Methods().GetParameters(mallocs).FindByShortName("sizeof");


-// Get the sizeof's parameters
-CxList sizeofParams = All.GetParameters(sizeofInMalloc);
+//Get the sizeofs parameters
+CxList sizeofParams = Find_Parameters().GetParameters(sizeofInMalloc).CxSelectDomProperty<Param>(p=>p.Value);


-// Filter all the primitives (sizeof(int) is OK) and the pointers (sizeof(*p) is
-// usually OK)
-sizeofParams -= sizeofParams.FindByShortName("bool");
-sizeofParams -= sizeofParams.FindByShortName("short");
-sizeofParams -= sizeofParams.FindByShortName("char");
-sizeofParams -= sizeofParams.FindByShortName("int");
-sizeofParams -= sizeofParams.FindByShortName("long");
-sizeofParams -= sizeofParams.FindByShortName("float");
-sizeofParams -= sizeofParams.FindByShortName("double");
-sizeofParams -= sizeofParams.FindByShortName("Pointer");
+//Filter all the primitives (sizeof(int) is OK) and the pointers (sizeof(*p) is usually OK)
+sizeofParams -= sizeofParams.FindByShortNames("bool", "short", "char", "int", "long", "float", "double", "Pointer");


-// Filter any types
+//Filter UnknownReferences that are TypeAliasDecl
+sizeofParams -= sizeofParams.FindAllReferences(Find_TypeAliasDecl());
```

```
+
+//Filter any types
 CxList sparams = sizeofParams;
 foreach (CxList sop in sparams)
 {
    CSharpGraph g = sop.GetFirstGraph();
-   CxList types = All.FindByType(g.FullName) +
-       All.FindByType(g.ShortName) -
-       sizeofParams -
-       All.FindByShortName(g.FullName) -
+   CxList types = All.FindByType(g.FullName) +
+       All.FindByType(g.ShortName) -
+       sizeofParams -
+       All.FindByShortName(g.FullName) -
        All.FindByShortName(g.ShortName);
-   if (types.Count > 0)
+   if (types.Count > 0)
    {
        sizeofParams -= sop;
    }
 }
-
-// Find the relevant sizeof's methods
+
+//Find the relevant sizeofs methods
 result = sizeofParams.GetAncOfType(typeof(MethodInvokeExpr));
```

CSharp / CSharp_Best_Coding_Practice / Aptca_Methods_Call_Non_Aptca_Methods

Code changes

CSharp / CSharp_Best_Coding_Practice / Dynamic_SQL_Queries

Code changes

```
---
+++
@@ -1,45 +1,48 @@
-CxList db = Find_DB_Base();
-CxList dbMethods = All.GetMethod(db); // All methods that contain a DB (others are usually irrelevant)
+CxList sqlDbIn = Find_SQL_DB_In();
+sqlDbIn.Add(sqlDbIn.GetMembersOfTarget());
+CxList db = sqlDbIn.FindByType<MethodInvokeExpr>();
+CxList methodDecls = Find_MethodDecls();
+CxList dbMethods = methodDecls.GetMethod(db); // All methods that contain a DB (others are usually irrelevant)
 CxList csDB = All.GetByAncs(dbMethods);
 CxList methods = Find_Methods();
+CxList unkRefs = Find_Unknown_References();
+CxList declsAndMethods = All.NewCxList(methods, methodDecls);
```

```
 // Add all methods called by the DB methods - up to 3 levels of calls
 for (int i = 0; i < 3; i++)
 {
-   CxList dbDelta = All.FindAllReferences(methods * csDB);
+   CxList dbDelta = declsAndMethods.FindAllReferences(methods * csDB);
     dbMethods.Add(dbDelta);
     csDB.Add(All.GetByAncs(dbDelta));
 }

 //Add binaries influenced by dynamic values
-CxList relevantDynamicValues = All.NewCxList();
-relevantDynamicValues.Add(Find_Unknown_References(), Find_MemberAccesses(), methods);
+CxList relevantDynamicValues = All.NewCxList(unkRefs, Find_MemberAccesses(), methods);
 CxList dynamicBinary = relevantDynamicValues.FindByFathers(Find_BinaryExpr()).GetFathers().FindByType<BinaryExpr>();
 CxList binary = csDB.FindByFathers(dynamicBinary).GetFathers().FindByType<BinaryExpr>();

-CxList stringMethods = (csDB * methods).FindAllReferences(All.FindByReturnType("String"));
+CxList stringMethods = (csDB * methods).FindAllReferences(methodDecls.FindByReturnType("String"));
 CxList append = csDB.FindByShortName("append", false);

 CxList formats = methods.FindByMemberAccess("String.Format", false);
 CxList formatParameters = formats.FindByParameters(All.GetParameters(formats, 1));

-CxList dbVariables = csDB.FindByType(typeof(UnknownReference));
-dbVariables.Add(csDB.FindByType(typeof(Declarator)));
+CxList dbVariables = csDB.FindByTypes(typeof(UnknownReference), typeof(Declarator));
 CxList str = dbVariables.FindByTypes(new String[] {"String", "System.String"});
 str.Add(stringMethods);

 CxList integers = Find_Integers() - db;

 CxList concat = binary.InfluencingOnAndNotSanitized(str, integers);
-concat.Add(str.GetByAncs(binary.GetByAncs(db)));
-concat.Add(append);
-concat.Add(formatParameters);
+concat.Add(
+   str.GetByAncs(binary.GetByAncs(db)),
+   append,
+   formatParameters);

 CxList sanitize = Find_Parameters();
 CxList substring = methods.FindByMemberAccess("String.substring", false);
-sanitize.Add(All.GetByAncs(All.GetParameters(substring)));
-sanitize.Add(integers);
+sanitize.Add(
+   All.GetByAncs(unkRefs.GetParameters(substring)),
+   integers);
```

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

CSharp / CSharp_Best_Coding_Practice / Exposure_of_Resource_to_Wrong_Sphere

Code changes

CSharp / CSharp_Best_Coding_Practice / Insufficient_Logging_of_Sensitive_Operations

Code changes

CSharp / CSharp_Best_Coding_Practice / Leftover_Debug_Code

Code changes

CSharp / CSharp_Best_Coding_Practice / Missing_XML_Validation

Code changes

CSharp / CSharp_Best_Coding_Practice / Pages_Without_Global_Error_Handler

Code changes

CSharp / CSharp_Best_Coding_Practice / Routed_Deprecated_Code

Code changes

CSharp / CSharp_Best_Coding_Practice / Unchecked_Error_Condition

Code changes

CSharp / CSharp_Best_Coding_Practice / Undocumented_API

Code changes

CSharp / CSharp_Best_Coding_Practice / Unvalidated_Arguments_Of_Public_Methods

Code changes

CSharp / CSharp_Best_Coding_Practice / Visible_Pointers

Code changes

CSharp / CSharp_Heuristic / Heuristic_2nd_Order_SQL_Injection

Code changes

CSharp / CSharp_Heuristic / Heuristic_CSRF

Code changes

CSharp / CSharp_Heuristic / Heuristic_Parameter_Tampering

Code changes

CSharp / CSharp_Heuristic / Heuristic_SQL_Injection

Code changes

CSharp / CSharp_High_Risk / Connection_String_Injection

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList con = Find_Connection_String();
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList sanitize = Find_Connection_String_Sanitize();

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

CSharp / CSharp_High_Risk / Deserialization_of_Untrusted_Data_MSMQ

Code changes

CSharp / CSharp_High_Risk / JWT_No_Signature_Verification

Code changes

CSharp / CSharp_High_Risk / Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -3,7 +3,8 @@
     CxList methods = Find_Methods();
     CxList parameters = Find_Parameters();

-    CxList inputs = Find_Interactive_Inputs();
+    CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+    inputs -= Find_ListBox_Safe_Inputs();

     CxList outputs = Find_XSS_Outputs() - Find_Console_Outputs();

@@ -14,7 +15,9 @@
```

```
    CxList lambdaExpr = Find_LambdaExpr();
    CxList lambdaMethods = methods.GetByAncs(lambdaExpr);
    CxList isHtmlEncode = lambdaMethods.FindByMemberAccess("HttpUtility.HtmlEncode");
-   CxList linqSanitizers = linqWhereAndSelect.FindByParameters(isHtmlEncode.GetByAncs(parameters.GetParameters(linqWhereAndSelect)).GetAncOfType(typeof(LambdaExpr)));
+   CxList linqSanitizers = linqWhereAndSelect.FindByParameters(
+       isHtmlEncode.GetByAncs(parameters.GetParameters(linqWhereAndSelect)).GetAncOfType<LambdaExpr>()
+   );

    // FileInfo can be a sanitizer:
    CxList fileInfos = All.FindByType("FileInfo").GetMembersOfTarget();
```

CSharp / CSharp_High_Risk / Resource_Injection

Code changes

CSharp / CSharp_High_Risk / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -3,13 +3,14 @@
 // Inputs for Second Order SQL Injection can be
 //   - data retrieval from DB
 //   - reading from a file
-CxList inputs = Find_DB_Out();
-inputs.Add(Find_Read());
+CxList inputs = All.NewCxList(Find_DB_Out(), Find_Read(), Find_Cloud_Storage_Inputs());

 // DB operations that return a Boolean or Integer are not a threat
 inputs -= methods.FindByShortName("*ExecuteNonQuery");

-CxList sanitize = Find_SQL_Sanitize();
+// Second order results should not be returned when they contain a valid SQLi sub-flow
+// e.g. when data read from a DB is written to something that can be an interactive input like a text field
+CxList sanitize = All.NewCxList(Find_SQL_Sanitize(), Find_SQL_Injection().GetFirstNodesInPath());

 // Sinks for Second Order SQL Injection are SQL statements
 CxList sink = Find_DB_In();
@@ -18,4 +19,6 @@

 //Present only results where there's string manipulation, flows where data is executed
 //directly as read are to be considered FPs
-result = allResults - allResults.SanitizeCxList(Find_Stored_String_Manipulation()).GetFirstNodesInPath();
+result = allResults.IntersectWithNodes(Find_Stored_String_Manipulation());
+
+result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

CSharp / CSharp_High_Risk / Stored_XSS

Code changes

```
---
+++
@@ -4,9 +4,6 @@
    CxList outputs = Find_XSS_Outputs();
    CxList sanitize = Find_XSS_Sanitize();

-   result = inputs.InfluencingOnAndNotSanitized(outputs, sanitize);
-   result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
-
    //Creating flow for FileStream.Read(buffer)
    CxList objectCreate = Find_ObjectCreations();

@@ -17,14 +14,28 @@

    if(fileStream.Count > 0){
        CxList methods = Find_Methods();
-       CxList unknownRefs = Find_UnknownReference();

        CxList fileStreamMethods = methods.GetMembersWithTargets(fileStream);
        fileStreamMethods = fileStreamMethods * inputs;
-       CxList firstParam = unknownRefs.GetParameters(fileStreamMethods, 0).FindByType(typeof(UnknownReference));
+       CxList firstParam = Find_UnknownReference().GetParameters(fileStreamMethods, 0).FindByType<UnknownReference>();

        CxList fileStreamFlow = outputs.InfluencedByAndNotSanitized(firstParam, sanitize);
-       fileStreamFlow = fileStreamMethods.ConcatenatePath(fileStreamFlow);
-       result.Add(fileStreamFlow);
+       if(fileStreamFlow.Count > 0){
+           fileStreamFlow = fileStreamMethods.ConcatenatePath(fileStreamFlow);
+           result.Add(fileStreamFlow);
+       }
    }
+
+   //Ensuring that results follow the flow of the parameter in Write method outputs
+   CxList writeOutputs = outputs.FindByType<MethodInvokeExpr>().FindByShortName("Write", false).FindByFileName("*.cs");
+   outputs -= writeOutputs;
+   CxList writeOutputsParams = All.GetParameters(writeOutputs);
+   writeOutputsParams.Add(All.GetByAncs(writeOutputsParams));
+   CxList writeResults = inputs.InfluencingOnAndNotSanitized(writeOutputs, sanitize);
+
+   result.Add(
+       writeResults.IntersectWithNodes(writeOutputsParams),
+       inputs.InfluencingOnAndNotSanitized(outputs, sanitize));
+
+   result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
 }
```

CSharp / CSharp_High_Risk / Unsafe_Reflection

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList comCreations = Find_ObjectCreations().FindByTypes(new String[]{
    "CodeSnippetExpression",
    "CodeEntryPointMethod",
```

CSharp / CSharp_Low_Visibility / Cross_Site_History_Manipulation

Code changes

CSharp / CSharp_Low_Visibility / Heap_Inspection

Code changes

CSharp / CSharp_Low_Visibility / Impersonation_Issue

Code changes

CSharp / CSharp_Low_Visibility / Improper_Resource_Shutdown_or_Release

Code changes

CSharp / CSharp_Low_Visibility / Information_Exposure_Through_an_Error_Message

Code changes

CSharp / CSharp_Low_Visibility / Information_Leak_Through_Persistent_Cookies

Code changes

CSharp / CSharp_Low_Visibility / Insufficiently_Protected_Credentials

Code changes

CSharp / CSharp_Low_Visibility / JavaScript_Hijacking

Code changes

CSharp / CSharp_Low_Visibility / JWT_Excessive_Expiration_Time

Code changes

CSharp / CSharp_Low_Visibility / JWT_Use_Of_Hardcoded_Secret

Code changes

CSharp / CSharp_Low_Visibility / Missing_Function_Level_Authorization

Code changes

CSharp / CSharp_Low_Visibility / Open_Redirect

Code changes

CSharp / CSharp_Low_Visibility / Overly_Permissive_Cross_Origin_Resource_Sharing_Policy

Code changes

CSharp / CSharp_Low_Visibility / Password_In_Comment

Code changes

CSharp / CSharp_Low_Visibility / Potential_ReDoS_In_Static_Field

Code changes

CSharp / CSharp_Low_Visibility / Reliance_on_DNS_Lookups_in_a_Decision

Code changes

CSharp / CSharp_Low_Visibility / Stored_Command_Argument_Injection

Code changes

CSharp / CSharp_Low_Visibility / Trust_Boundary_Violation_in_Session_Variables

Code changes

CSharp / CSharp_Low_Visibility / Unencrypted_Web_Config_File

Code changes

CSharp / CSharp_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

CSharp / CSharp_Low_Visibility / Use_of_Insufficiently_Random_Values

Code changes

CSharp / CSharp_Medium_Threat / Buffer_Overflow

Code changes

CSharp / CSharp_Medium_Threat / CGI_XSS

Code changes

CSharp / CSharp_Medium_Threat / Cookie_Injection

Code changes

```
---
+++
@@ -5,7 +5,7 @@
 //    and written to response (Cookies.Add) are reported!
 ////////////////////////////////////////////////////////////////////////////


-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList sanitize = All.FindByMemberAccess("Convert.ToBase64String");
 sanitize.Add(
    Find_Encrypt(),
```

CSharp / CSharp_Medium_Threat / Data_Filter_Injection

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 // Data Filter Manipulation (Information Leak)

 CxList db = Find_Disconnected_DB_Access();
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList sanitized = Find_SQL_Sanitize();


 result = inputs.InfluencingOnAndNotSanitized(db, sanitized);
```

CSharp / CSharp_Medium_Threat / DoS_by_Sleep

Code changes

CSharp / CSharp_Medium_Threat / Excessive_Data_Exposure

Code changes

```
---
+++
@@ -127,4 +127,5 @@
 CxList sanitizers = All.FindAllReferences(sensitiveSanitizers);
 sanitizers.Add(Find_Boolean_Operators());


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

CSharp / CSharp_Medium_Threat / HttpOnlyCookies

Code changes

CSharp / CSharp_Medium_Threat / HTTP_Response_Splitting

Code changes

CSharp / CSharp_Medium_Threat / Improper_Locking

Code changes

CSharp / CSharp_Medium_Threat / Improper_Restriction_of_XXE_Ref

Code changes

```
---
+++
@@ -1,15 +1,15 @@
 //Find XXE (XML External Entity vulnerability) in C# using:
 //XmlReader, XmltextReader, XmlDocument, XDocument

-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList integers = Find_Integers();
 CxList objectCreations = Find_ObjectCreations();
 CxList unknownList = Find_Unknown_References();

-CxList XXE = Find_XXE_XmlReader();     //Sanitized by default
-XXE.Add(Find_XXE_XmlTextReader());     //NOT Sanitized by default in versions < 4.5.2
-XXE.Add(Find_XXE_XDocument());     //Sanitized by default
-XXE.Add(Find_XXE_XmlDocument());       //NOT Sanitized by default in versions < 4.5.2
+CxList XXE = All.NewCxList(Find_XXE_XmlReader(), //Sanitized by default
+   Find_XXE_XmlTextReader(),                    //NOT Sanitized by default in versions < 4.5.2
+   Find_XXE_XDocument(),                        //Sanitized by default
+   Find_XXE_XmlDocument());                     //NOT Sanitized by default in versions < 4.5.2

 XXE.Add(Find_XXE_XPathDocument());  //NOT Sanitized by default in versions < 4.5.2
```

CSharp / CSharp_Medium_Threat / Insecure_Cookie

Code changes

CSharp / CSharp_Medium_Threat / Insufficient_Connection_String_Encryption

Code changes

CSharp / CSharp_Medium_Threat / JWT_Sensitive_Information_Exposure

Code changes

CSharp / CSharp_Medium_Threat / Missing_Object_Level_Authorization

Code changes

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


 CxList methodInvk = Find_Methods();
 CxList classes = Find_Classes();
@@ -16,16 +16,16 @@
 CxList methodsWithIntArgs = methodDecls.FindByParameters(intParameters);


 CxList authorize = customAttributes.FindByCustomAttribute("Authorize");
-CxList safeAuthorize = (authorize.FindCustomAttributeParameters("Authorize")).GetAncOfType(typeof(CustomAttribute));
+CxList safeAuthorize = (authorize.FindCustomAttributeParameters("Authorize")).GetAncOfType<CustomAttribute>();
 CxList vulnerableAuthorize = authorize - safeAuthorize;
-CxList safeMethods = safeAuthorize.GetAncOfType(typeof(MethodDecl)) * methodsWithIntArgs;
+CxList safeMethods = safeAuthorize.GetAncOfType<MethodDecl>() * methodsWithIntArgs;
 CxList vulnerableMethods = vulnerableAuthorize.GetAncOfType<MethodDecl>() * methodsWithIntArgs;
 CxList vulnerableClasses = vulnerableAuthorize.GetAncOfType<ClassDecl>() * classes;

 // This has the "AllowAnonymous" annotation, explicitly saying dev intended this function to be consumed by anyone.
 // While in reality this is a massive issue, it is an intentional design, so it should't be a result
 CxList allowAnonymous = customAttributes.FindByCustomAttribute("AllowAnonymous");
-CxList irrelevantMethods = allowAnonymous.GetAncOfType(typeof(MethodDecl)) * methodsWithIntArgs;
+CxList irrelevantMethods = allowAnonymous.GetAncOfType<MethodDecl>() * methodsWithIntArgs;
 irrelevantMethods.Add(safeMethods);


 CxList methodsToGetClass = methodsWithIntArgs - vulnerableMethods;
@@ -34,11 +34,11 @@
 vulnerableMethods.Add(relevantMethods);


 CxList isInRole = methodInvk.FindByMemberAccess("*.IsInRole");
-CxList isInRoleMethod = isInRole.GetAncOfType(typeof(MethodDecl));
+CxList isInRoleMethod = isInRole.GetAncOfType<MethodDecl>();
 vulnerableMethods = vulnerableMethods - isInRoleMethod;


 CxList isInRoleMethodRefs = methodInvk.FindAllReferences(isInRoleMethod);
-CxList irrelevantRoleMethod = isInRoleMethodRefs.GetAncOfType(typeof(MethodDecl)) * vulnerableMethods;
+CxList irrelevantRoleMethod = isInRoleMethodRefs.GetAncOfType<MethodDecl>() * vulnerableMethods;
 vulnerableMethods = vulnerableMethods - irrelevantRoleMethod;


 // Finding Minimal API associated endpoints, that don't make use of the
@@ -50,7 +50,7 @@
     vulnerableMethods.Add(unsafeMinimalMethods);
```

```
    CxList lambdaControllers = Find_LambdaExpr().GetParameters(unsafeMinimalMethods);
-   inputs.Add(Find_ParamDecl().GetParameters(lambdaControllers));
+   inputs.Add(parameters.GetParameters(lambdaControllers));
 }

 CxList binarysInIfs = binaryExpr.GetByAncs(ifs);
```

CSharp / CSharp_Medium_Threat / MVC_View_Injection

Code changes

CSharp / CSharp_Medium_Threat / No_Request_Validation

Code changes

CSharp / CSharp_Medium_Threat / Parameter_Tampering

Code changes

```
---
+++
@@ -8,7 +8,7 @@
 // 3.We use some get method of data structure.
 ////////////////////////////////////////////////////////////////////////
-CxList input = Find_Interactive_Inputs();
+CxList input = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList db = Find_DB_For_Parameter_Tampering();
 CxList sanitize = Find_Parameter_Tampering_Sanitize();
```

CSharp / CSharp_Medium_Threat / Path_Traversal

Code changes

```
---
+++
@@ -18,5 +18,5 @@
    result from Find_Interactive_Inputs.
 */

-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 result = Find_Path_Traversal(inputs);
```

CSharp / CSharp_Medium_Threat / Persistent_Connection_String

Code changes

CSharp / CSharp_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -66,11 +66,12 @@
 CxList exceptionsCtorsWithSuper = exceptionsCtors.FilterByDomProperty<ConstructorDecl>(c => c.BaseParameters.Count > 0);

 // Define outputs
-CxList outputs = Find_Outputs();
-outputs.Add(
+CxList outputs = All.NewCxList(
+   Find_Outputs(),
    exceptions,
    exceptionsCtorsWithSuper,
-   Find_Log_Outputs());
+   Find_Log_Outputs(),
+   Find_Cloud_Interactive_Outputs());

 // Define sanitize
 CxList sanitize = Find_DB_In(); // In some languages is called Find_DB, Find_DB_In, Find_DB_Input
```

CSharp / CSharp_Medium_Threat / Race_Condition_within_a_Thread

Code changes

CSharp / CSharp_Medium_Threat / ReDoS_By_Regex_Injection

Code changes

```
---
+++
@@ -1 +1,2 @@
-result = Find_ReDoS(Find_Inputs(), false);
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Interactive_Inputs());
+result = Find_ReDoS(inputs, false);
```

CSharp / CSharp_Medium_Threat / Session_Fixation

Code changes

CSharp / CSharp_Medium_Threat / SSL_Verification_Bypass

Code changes

CSharp / CSharp_Medium_Threat / SSRF

Code changes

CSharp / CSharp_Medium_Threat / Stored_Command_Injection

Code changes

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

```
@@ -1,5 +1,8 @@
-CxList inputs = Find_Read()+ Find_DB_Out()  +
-   All.FindByMemberAccess("Process.GetProcesses");
+CxList inputs = All.NewCxList(
+   Find_Read(),
+   Find_DB_Out(),
+   All.FindByMemberAccess("Process.GetProcesses"),
+   Find_Cloud_Storage_Inputs());


 CxList exec = Find_Command_Execution();
 CxList sanitize = Find_Sanitize();
```

CSharp / CSharp_Medium_Threat / Stored_LDAP_Injection

Code changes

```
---
+++
@@ -28,13 +28,13 @@
            "Directory",
            "File"
            };
-   CxList objects = Find_Unknown_References() + All.FindByType(typeof(Declarator));
+   CxList objects = All.NewCxList(Find_Unknown_References(), Find_Declarators());
    fileIO.Add(objects.FindByTypes(relevantTypes));
    fileIO.Add(All.FindByShortNames(relevantClasses));
    fileIO = fileIO.GetMembersOfTarget(); // We want the methods, not the objects
    fileIO.Add(Find_ASP_MVC_Controller_File_Result());
 }

-CxList inputs =  dbOut + fileIO;
+CxList inputs =  All.NewCxList(dbOut, fileIO, Find_Cloud_Storage_Inputs());
 result = ldap.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
 result.Add(DNParams.InfluencedByAndNotSanitized(inputs, sanitizersDN).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow));
```

CSharp / CSharp_Medium_Threat / Stored_Path_Traversal

Code changes

```
---
+++
@@ -18,7 +18,7 @@
    result from Find_Read() + Find_DB_Out();.
 */
 CxList inputs = All.NewCxList();
-inputs.Add(Find_Read(), Find_DB_Out());
+inputs.Add(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

```
---
+++
@@ -1,8 +1,8 @@
 CxList XPath = Find_XPath_Output();

-CxList inputs = Find_Read();
+CxList inputs = All.NewCxList(Find_Read(), Find_Cloud_Storage_Inputs());
 inputs.Add(Find_DB_Out());

 CxList sanitized = Find_XPath_Injection_Sanitizers();

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

CSharp / CSharp_Medium_Threat / Unsafe_Object_Binding

Code changes

CSharp / CSharp_Medium_Threat / Use_of_Cryptographically_Weak_PRNG

Code changes

CSharp / CSharp_Medium_Threat / Use_of_Hard_coded_Cryptographic_Key

Code changes

```
---
+++
@@ -64,10 +64,10 @@
     sinks.Add(All.GetParameters(inv, 0));
 }

-
+
 //SymetricAlgoritms that can influence CreateEncryptor by flow
 CxList createEncryptorMethod = methods.FindByShortName("CreateEncryptor");
-CxList symetricAlgoritmsDecls = Find_Declarators().GetAssigner().FindByShortNames(new List<string>{"AesCryptoServiceProvider","DESCryptoServiceProvider","RC2CryptoServiceProvider","RijndaelCryptoServiceProvider","TripleDESCryptoServiceProvider"}, false);
+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);

@@ -82,7 +82,7 @@
     "AesCryptoServiceProvider.CreateEncryptor",
     "DESCryptoServiceProvider.CreateEncryptor",
     "RC2CryptoServiceProvider.CreateEncryptor",
-    "RijndaelCryptoServiceProvider.CreateEncryptor",
+    "RijndaelManaged.CreateEncryptor",
     "TripleDESCryptoServiceProvider.CreateEncryptor"
```

```
    });
```

```
@@ -97,7 +97,6 @@
```

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

```
-
```

```
 //netcore
 CxList objAesGcmCcm = Find_ObjectCreations().FindByTypes(new String[]{"AesGcm","AesCcm"});
 CxList paramObjKey = All.GetParameters(objAesGcmCcm);
```

CSharp / CSharp_WebConfig / CookieLess_Authentication

Code changes

CSharp / CSharp_WebConfig / DebugEnabled

Code changes

CSharp / CSharp_WebConfig / Elmah_Enabled

Code changes

CSharp / CSharp_WebConfig / HardcodedCredentials

Code changes

CSharp / CSharp_Windows_Phone / Failure_to_Implement_Least_Privilege

Code changes

CSharp / CSharp_Windows_Phone / Hard_Coded_Cryptography_Key

Code changes

CSharp / CSharp_Windows_Phone / Insufficient_Application_Layer_Protect

Code changes

Dart / Dart_Mobile_Low_Visibility / Use_of_Non_Cryptographic_Random

Code changes

```
---
```

```
+++
```

```
@@ -1 +1,3 @@
```

```
-result = Find_Use_of_Non_Cryptographic_Random();
```

```
+// To avoid duplicate results don't flag them here if they exist in Use_of_Cryptographically_Weak_PRNG
```

```
+CxList possibleDuplicates = Find_Use_of_Cryptographically_Weak_PRNG().GetFirstNodesInPath();
```

```
+result = Find_Use_of_Non_Cryptographic_Random() - possibleDuplicates;
```

Dart / Dart_Mobile_Medium_Threat / Use_of_Cryptographically_Weak_PRNG

Code changes

```
---

+++

@@ -1,6 +1 @@

-CxList weakRandoms = Find_Use_of_Non_Cryptographic_Random();

-

-CxList outputs = Find_Hashing();

-outputs.Add(Find_Encryption());

-

-result = outputs.InfluencedBy(weakRandoms);

+result = Find_Use_of_Cryptographically_Weak_PRNG();
```

Go / Go_Medium_Threat / Missing_HttpOnly_Cookie

Code changes

```
---

+++

@@ -1,11 +1,2 @@

-CxList falseAbstractValues = Find_False_Abstract_Value();

-

-// Gin Web Framework SetCookie Method

-CxList ginSetCookie = Find_Gin_Gonic_Types(new List<string>() {"SetCookie"});

-

 // 7th SetCookie method param is the httpOnly

-CxList sanitizedParams = All.GetParameters(ginSetCookie) - All.GetParameters(ginSetCookie, 6);

-

-result = Find_Unsafe_Cookie_Flags("HttpOnly");

-result.Add(falseAbstractValues.InfluencingOnAndNotSanitized(ginSetCookie, sanitizedParams));

-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);

+result = Find_Unsafe_Cookie_Flags("HttpOnly", 7);
```

Go / Go_Medium_Threat / Missing_Secure_Cookie

Code changes

```
---

+++

@@ -1,11 +1,2 @@

-CxList falseAbstractValues = Find_False_Abstract_Value();

-

-// Gin Web Framework SetCookie Method

-CxList ginSetCookie = Find_Gin_Gonic_Types(new List<string>() {"SetCookie"});

-

-// 6th SetCookie method param is the secure

-CxList sanitizedParams = All.GetParameters(ginSetCookie) - All.GetParameters(ginSetCookie, 5);

-

-result = Find_Unsafe_Cookie_Flags("Secure");

-result.Add(falseAbstractValues.InfluencingOnAndNotSanitized(ginSetCookie, sanitizedParams));

-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);

+// 6th SetCookie method param is the secure
```

```
+result = Find_Unsafe_Cookie_Flags("Secure", 6);
```

Java / Java_High_Risk / Code_Injection

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Queue_Inputs());
 CxList code = Find_Code_Injection_Outputs();
 CxList sanitize = Find_General_Sanitize();
 sanitize.Add(Find_GetInitParameter_Method());
```

Java / Java_High_Risk / Connection_String_Injection

Code changes

```
---
+++
@@ -4,7 +4,7 @@
 con.Add(Find_Object_Create().FindInitialization(Find_Declarators().FindByType("DirContext")));
 con.Add(unks.GetParameters(unks.FindByType("DirContext").GetMembersOfTarget().FindByShortName("addToEnvironment")));

-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(),Find_Queue_Inputs());
 CxList sanitize = Find_CollectionAccesses();
 sanitize.Add(Find_Integers(),
            methods.FindByMemberAccesses(new string[]{"Properties.getProperty", "Configuration.setProperty"}),
```

Java / Java_High_Risk / JSF_Local_File_Inclusion

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Queue_Inputs());
 CxList allTypes = All.NewCxList();
 allTypes.Add(Find_TypeRef(), Find_Declarators(), Find_ObjectCreations(), Find_UnknownReference());
```

Java / Java_High_Risk / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,11 +1,12 @@
 CxList dbOut = Find_DB_Out();

-CxList read = All.NewCxList();
-read.Add(Find_Read_NonDB(),
+CxList read = All.NewCxList(
```

```
+   Find_Read_NonDB(),
    Find_FileSystem_Read(),
    Find_Files_Open(),
    Find_Vulnerable_Nio_Files_Methods(),
-   Find_Vulnerable_Io_File_Methods());
+   Find_Vulnerable_Io_File_Methods(),
+   Find_Cloud_Storage_In());

 CxList dbIn = Find_SQL_DB_In();
```

Java / Java_High_Risk / Stored_XSS

Code changes

```
---
+++
@@ -17,13 +17,13 @@
    read,
    Find_Vulnerable_Nio_Files_Methods(),
    Find_Vulnerable_Io_File_Methods(),
-   getRequestSessionMethods);
+   getRequestSessionMethods,
+   Find_Cloud_Storage_In());

 CxList outputs = Find_XSS_Outputs();

 // Sanitizations
-CxList sanitized = All.NewCxList();
-sanitized.Add(Find_XSS_Sanitize(), Find_XSS_Outputs_Sanitize(outputs));
+CxList sanitized = All.NewCxList(Find_XSS_Sanitize(), Find_XSS_Outputs_Sanitize(outputs));

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

Java / Java_High_Risk / Unsafe_Reflection

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 CxList methods = Find_Methods();
 CxList ifs = Find_Ifs();

-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Queue_Inputs());

 CxList outputs = methods.FindByNames("System.loadLibrary", "System.load");
```

Java / Java_Medium_Threat / CGI_Stored_XSS

Code changes

```
---
+++
@@ -2,11 +2,12 @@
    CxList db = Find_DB_Out();
    CxList methods = Find_Methods();

-   CxList read = Find_Read_NonDB();
-   read.Add(
+   CxList read = All.NewCxList(
+       Find_Read_NonDB(),
        Find_FileSystem_Read(),
        Find_Vulnerable_Io_File_Methods(),
-       Find_Vulnerable_Nio_Files_Methods());
+       Find_Vulnerable_Nio_Files_Methods(),
+       Find_Cloud_Storage_In());

    CxList outputs = All.GetParameters(Find_Console_Outputs());
```

Java / Java_Medium_Threat / Dangerous_File_Inclusion

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Queue_Inputs());
 CxList include = Find_File_Inclusion();
 CxList sanitize = Find_General_Sanitize();
```

Java / Java_Medium_Threat / Improper_Restriction_of_XXE_Ref

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 // Find XXE (XML External Entity vulnerability) in Java
 CxList methods = Find_Methods();
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Storage_In());

 CxList xxe = Find_XXE_Requests();
 xxe.Add(Find_XXE_Spring());
```

Java / Java_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -1,13 +1,11 @@
 // This query searches for variables and constants that could contain personal sensitive data,
 // which is streamed to an output.
-
-CxList methods = Find_Methods();
-
 CxList strings = Find_Strings();
 CxList integerLiteral = Find_IntegerLiterals();
 CxList literals = All.NewCxList();
 literals.Add(strings, integerLiteral);
 CxList nullLiteral = Find_NullLiteral();
+CxList typeRefs = Find_TypeRef();

 // Find names that are suspected to be personal info, e.g. String PASSWORD, Integer SSN
 // Remove string literals, such as x = "password"
@@ -66,9 +64,9 @@
 // We must find responses with sensitive types
 //Get classes with sensitive fields
 CxList sensitiveClass = personal_info.GetAncOfType<ClassDecl>();
-sensitiveClass.Add(Find_TypeRef().FindByType(sensitiveClass).GetAncOfType<ClassDecl>());
+sensitiveClass.Add(typeRefs.FindByType(sensitiveClass).GetAncOfType<ClassDecl>());
 //Get unknown references with sensitive type
-CxList sensitiveTypeVars = Find_TypeRef().FindByType(sensitiveClass).GetAncOfType<VariableDeclStmt>();
+CxList sensitiveTypeVars = typeRefs.FindByType(sensitiveClass).GetAncOfType<VariableDeclStmt>();
 CxList sensitiveTypeDecls = Find_Declarators().GetByAncs(sensitiveTypeVars);
 CxList sensitiveTypeUsage = Find_UnknownReference().FindAllReferences(sensitiveTypeDecls);
 CxList sensitiveResponse = sensitiveTypeUsage * Find_HTTP_Responses();
@@ -80,7 +78,7 @@
     sensitiveResponse
     );

-personal_info = private_info;
+personal_info = All.NewCxList(private_info);

 // 3) Add exceptions (that could be thrown) to outputs.
 CxList exceptions = Find_ObjectCreations().FindByName("*Exception");
@@ -90,8 +88,12 @@
 CxList exceptionsCtorsWithSuper = exceptionsCtors.FilterByDomProperty<ConstructorDecl>(c => c.BaseParameters.Count > 0);

 // Define outputs
-CxList outputs = Find_Outputs();
-outputs.Add(exceptions, exceptionsCtorsWithSuper);
+CxList outputs = All.NewCxList(
+   Find_Outputs(),
+   exceptions,
+   exceptionsCtorsWithSuper
```

```
+);
+
 // Define sanitize
 CxList sanitize = Find_DB();
 sanitize.Add(Find_Encrypt(), Find_UnitTest_Code(), Find_HashSanitize());
```

Java / Java_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Queue_Inputs());

 // Remove argc/argv from inputs
 CxList mainDeclarations = All.FindByShortName("*main*").FindByType<MethodDecl>();
```

Java / Java_Medium_Threat / Stored_Command_Injection

Code changes

```
---
+++
@@ -1,8 +1,7 @@
 // Query Stored Command Injection
 // ==============================
-CxList inputs = Find_Read_NonDB();
-inputs.Add(Find_DB_Out());
+CxList inputs = All.NewCxList(Find_Read_NonDB(),Find_DB_Out(), Find_Cloud_Storage_In());

 CxList exec = Find_Command_Injection_Outputs();
 CxList sanitize = Find_Command_Injection_Sanitize();
```

Java / Java_Medium_Threat / Stored_LDAP_Injection

Code changes

```
---
+++
@@ -1,5 +1,4 @@
-CxList inputs = Find_Read_NonDB();
-inputs.Add(Find_DB_Out());
+CxList inputs = All.NewCxList(Find_Read_NonDB(), Find_DB_Out(), Find_Cloud_Storage_In());

 CxList sanitize = Find_LDAP_Sanitize();
 sanitize.Add(Find_GetInitParameter_Method());
```

Java / Java_Medium_Threat / XQuery_Injection

Code changes

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

```
@@ -1,6 +1,6 @@
 // Finds the XQuery Injections that can be done on database
 // This query may share some results with SQL_Injection
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(),Find_Queue_Inputs());
 CxList dbIns = Find_DB_In();

 dbIns.Add(Find_XQuery_Injection_Additional_DBIn());
```

JavaScript / JavaScript_High_Risk / Client_DOM_Code_Injection

Code changes

```
---
+++
@@ -1,14 +1,2 @@
 CxList inputs = Find_Inputs_NoWindowLocation();
-CxList Eval = Find_Outputs_CodeInjection();
-CxList sanitize = Code_Injection_Sanitize();
-
-result.Add(Eval.FindByType<Param>() * inputs - sanitize);
-
-// Remove inputs which are implicitely passed as parameters to anonymous functions.
-CxList imAParam = inputs.GetParameters(Find_ObjectCreations());
-CxList lambda = Find_LambdaExpr();
-imAParam.Add(inputs.GetParameters(lambda));
-inputs -= imAParam;
-
-result.Add((Eval).InfluencedByAndNotSanitized(inputs, sanitize, CxList.InfluenceAlgorithmCalculation.NewAlgorithm),
-          Find_Source_Equals_Sink(inputs, Eval));
+result = Find_Client_DOM_Code_Injection(inputs);
```

JavaScript / JavaScript_High_Risk / Client_DOM_Stored_Code_Injection

Code changes

```
---
+++
@@ -1,8 +1,7 @@
-CxList storedOut = Find_Cookie();
-storedOut.Add(Find_Storage_Outputs(), Find_DB_Out(), Find_XHR_Response());
-CxList Eval = Find_Outputs_CodeInjection();
-CxList sanitize = Code_Injection_Sanitize();
+CxList storedInputs = All.NewCxList(
+   Find_DB_Out(),
+   Find_Cookie(),
+   Find_Storage_Outputs(),
+   Find_XHR_Response());

-result.Add(Eval.FindByType<Param>() * storedOut - sanitize,
-       Eval.InfluencedByAndNotSanitized(storedOut, sanitize, CxList.InfluenceAlgorithmCalculation.NewAlgorithm),
-       Find_Source_Equals_Sink(storedOut, Eval));
```

```
+result = Find_Client_DOM_Code_Injection(storedInputs);
```

JavaScript / JavaScript_High_Risk / Prototype_Pollution

Code changes

```
---
+++
@@ -8,7 +8,7 @@
 CxList methods = Find_Methods();
 CxList strings = Find_Strings();
 // inputs
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Interactive_Inputs());

 // sanitizers
 CxList sanitizers = All.NewCxList();
```

JavaScript / JavaScript_Low_Visibility / Unsafe_Use_Of_Target_blank

Code changes

```
---
+++
@@ -1,22 +1,36 @@
-//1st part - Typescript
+// Generals
 CxList methods = Find_Methods();
 CxList unknwnRefs = Find_UnknownReference();
-//Find variables assigned by window.open();
-//e.g var window = window.open();
+CxList stringLiterals = Find_String_Literal();
+CxList inputs = Find_Inputs();
+CxList conditions = Find_Conditions();
+CxList nullLiteral = Find_NullLiteral();
+
+// Create list of parameters and literals to be used in query
+CxList parameters = Find_Param();
+parameters.Add(stringLiterals, unknwnRefs);
+
+// 1st part - Typescript
+
+// Find variables assigned by window.open();
+// e.g var window = window.open();
 CxList windowOpen = methods.FindByMemberAccess("window.open");

-CxList stringLiterals = Find_String_Literal();
-//The only vulnerable cases are: if the second parameter of window.open is an empty string or the string "_blank"
+// The only vulnerable cases are: if the second parameter of window.open is an empty string or the string "_blank"
 CxList targetBlankArguments = stringLiterals.FindByShortName("_blank");
 targetBlankArguments.Add(stringLiterals.FindByShortName(""));
```

```
-//When there is no second parameter at all, the default is blank.
-CxList windowOpenSecondParameters = All.GetParameters(windowOpen, 1);
+// When there is no second parameter at all, the default is blank.
+CxList windowOpenThirdParameters = parameters.GetParameters(windowOpen, 2);
+
+// When there is no second parameter at all, the default is blank.
+CxList windowOpenSecondParameters = parameters.GetParameters(windowOpen, 1);
 CxList windowOpenWithoutParameters = windowOpen - windowOpen.FindByParameters(windowOpenSecondParameters);
-//We are left with all the window.open methods with at least 2 parameters.
+
+// We are left with all the window.open methods with at least 2 parameters.
 windowOpen -= windowOpenWithoutParameters;

-//The list of window.open parameters that are either "_blank" or "".
+// The list of window.open parameters that are either "_blank" or "".
 CxList unsafeDirectArguments = windowOpenSecondParameters * targetBlankArguments;

 /* In case the 2nd parameter isn't a StringLiteral, check if "_blank"
@@ -26,34 +40,39 @@
 CxList targetUnsafeArg = windowOpenSecondParameters.DataInfluencedBy(targetBlankArguments);
 targetUnsafeArg.Add(unsafeDirectArguments);

+// Add those without second parameter to the unsafe list.
 windowOpen = windowOpen.FindByParameters(targetUnsafeArg);
-//Add those without second parameter to the unsafe list.
 windowOpen.Add(windowOpenWithoutParameters);
+
+// Get third parameters sanitizers from window.open
+// e.g window.open('http://www.example.com?ReportID=1', '_blank', 'noopener');
+CxList sanitizersFromWindowOpen = windowOpenThirdParameters.FindByShortNames(new [] {"*noopener*", "*noreferrer*"});
+windowOpen -= sanitizersFromWindowOpen;

 CxList leftSide = All.FindByAssignmentSide(CxList.AssignmentSide.Left);
 CxList varOfWindowOpen = windowOpen.GetAssignee(leftSide);

-CxList inputs = Find_Inputs();
-CxList conditions = Find_Conditions();
 CxList interestingConditions = conditions.DataInfluencedBy(inputs).GetLastNodesInPath();
 CxList interestingStatements = interestingConditions.GetAncOfType<IfStmt>();

-//sanitization 1st part
-CxList nullLiteral = Find_NullLiteral();
+// Sanitization 1st part
 CxList windowOpener = All.FindByMemberAccess("*.opener");
 CxList windowOpenerNull = nullLiteral.GetAssignee(windowOpener);
-//remove window.opener = null if it is influenced by the user input
+
+// Remove window.opener = null if it is influenced by the user input
```

```
 windowOpenerNull -= windowOpenerNull.GetByAncs(windowOpenerNull.GetAncOfType<IfStmt>() * interestingStatements);


 CxList referencesWindowOpenerNull = All.FindAllReferences(windowOpenerNull.GetTargetOfMembers());
-//Remove variables that has the opener = null
-//e.g window.opener = null;
+
+// Remove variables that has the opener = null
+// e.g window.opener = null;
 varOfWindowOpen -= referencesWindowOpenerNull;
 result.Add(varOfWindowOpen);


-//2nd part - HTML
+// 2nd part - HTML
+
 string pattern = @"<a\s+((?<anch><)|(?<-anch>>)|[^<>])+>(?(anch)(?!))";
-List < string > extensions = new List<string>{"*.inc", "*.asax", "*.ascx", "*.aspx", "*.master"
+List<string> extensions = new List<string> {"*.inc", "*.asax", "*.ascx", "*.aspx", "*.master"
         , "*.cshtml", "*.jsf", "*.jsp", "*.jspf", "*.xhtml", "*.vm"
         , "*.handlebars", "*.hbs", "*.htm", "*.html", "*.jade", "*.pug"
         , "*.apexp", "*.page", "*.component", "*.php", "*.php3", "*.php4"
@@ -63,20 +82,19 @@
 CxList matches = All.FindByRegexExt(pattern, extensions, false, CxList.CxRegexOptions.None, RegexOptions.None);
 CxList htmlResults = All.NewCxList();

-
-//Find this pattern in anchors:
-//e.g <a href="http://www.w3schools.com" target="_blank">
+// Find this pattern in anchors:
+// e.g <a href="http://www.w3schools.com" target="_blank">
 string targetPattern = @"target\s*=\s*('|""|\\"")_?blank('|""|\\"")";
 Regex regexTarget = new Regex(targetPattern);


-//Do not add anchors if they have rel = "noopener" or rel = "noopener noreferrer"
-//e.g  <a href="http://www.w3schools.com" target="_blank" rel = "noopener noreferrer">
-//e.g  <a href="http://www.w3schools.com" target="_blank" rel = "noopener">
+// Do not add anchors if they have rel = "noopener" or rel = "noopener noreferrer"
+// e.g  <a href="http://www.w3schools.com" target="_blank" rel = "noopener noreferrer">
+// e.g  <a href="http://www.w3schools.com" target="_blank" rel = "noopener">
 string relPattern = @"rel\s*=\s*('|""|\\"")?(noopener\s?|noreferrer\s?)+('|""|\\"")?";
 Regex regexRel = new Regex(relPattern);


-//Do not add anchors if they have href="mailto:*"
-//e.g  <a href="mailto:support.dummy@mail.com" target="blank">
+// Do not add anchors if they have href="mailto:*"
+// e.g  <a href="mailto:support.dummy@mail.com" target="blank">
 string mailtoPattern = @"href\s*=\s*('|""|\\"")mailto:";
 Regex regexMailto = new Regex(mailtoPattern);
```

```
@@ -95,40 +113,50 @@
    }
 }


-//find setAttribute methods (it can sanitize the html)
+// Find setAttribute methods (it can sanitize the html)
 CxList setAttribute = methods.FindByShortName("setAttribute");
-List <string> relAttributes = new List<string>{"\"noopener\"", "\"noreferrer\"", "\"noopener noreferrer\"", "\"noreferrer noopener\""};
-//find first parameter of setAttribute
+string[] relAttributes = new[] {"\"noopener\"", "\"noreferrer\"", "\"noopener noreferrer\"", "\"noreferrer noopener\""};
+
+// Find first parameter of setAttribute
 CxList setAttributeFirstParam = stringLiterals.GetParameters(setAttribute, 0).FindByShortName("\"rel\"");
-//setAttribute will sanitize with a combination of "rel" with "noopener" or "noopener noreferrer"
+
+// SetAttribute will sanitize with a combination of "rel" with "noopener" or "noopener noreferrer"
 CxList sanitizers = setAttribute.FindByParameters(setAttributeFirstParam);
 sanitizers = setAttribute.FindByParameters(stringLiterals.GetParameters(sanitizers, 1).FindByShortNames(relAttributes));
-//Unsafe_Use_of_Target_Blank
-//Individual sanitizer
-//Find methods that can use anchor tag
+
+// Find methods that can use anchor tag (Individual sanitizer)
 CxList elementsByTagName = sanitizers.GetTargetOfMembers().GetTargetOfMembers().FindByShortName("getElementsByTagName");
 CxList interstingParam = stringLiterals.FindByShortName("a");
-//Find methods that use anchor tag
+
+// Find methods that use anchor tag
 CxList elementsByTagNameMethod = elementsByTagName.FindByParameters(interstingParam);
-//Find only methods that are arrays
+
+// Find only methods that are arrays
 CxList individualSanitizers = elementsByTagNameMethod.GetMembersOfTarget() - sanitizers;


 CxList toRemove = All.NewCxList();


-foreach(CxList individualSanitizer in individualSanitizers){
+foreach(CxList individualSanitizer in individualSanitizers)
+{
     CSharpGraph gIndex = individualSanitizer.TryGetCSharpGraph<CSharpGraph>();
+
+   // Try to get the integer that access to the array position
     int index = 0;
-   //Try to get the integer that access to the array position
     bool parsed = int.TryParse(gIndex.ShortName, out index);
-   if(parsed){
-       //Get file name of node
+
```

```
+   if(parsed)
+   {
+       // Get file name of node
        string fileName = gIndex.LinePragma.FileName;
-       //Find results that are from the sanitization file
+
+       // Find results that are from the sanitization file
        CxList resultsOfFile = htmlResults.FindByFileName(fileName);
+
        int counter = 0;
-       foreach(CxList res in resultsOfFile){
-           //if current result is the same that the index, then the result is sanitized
-           //this will assume that the resultsOfFile came by the order of line and column due to regex behavior
+       foreach(CxList res in resultsOfFile)
+       {
+           // If current result is the same that the index, then the result is sanitized
+           // this will assume that the resultsOfFile came by the order of line and column due to regex behavior
            if(counter == index){
                toRemove.Add(res);
                break;
@@ -138,18 +166,21 @@
    }
 }

-//This will sanitize all anchors in file
+// This will sanitize all anchors in file
 CxList fileSanitizers = sanitizers - individualSanitizers.GetMembersOfTarget();

-List<string> sanitizedFiles = new List<string>();
-//Get files that use setAttribute sanitized
-foreach(CxList sanitizer in fileSanitizers){
+List <string> sanitizedFiles = new List<string>();
+
+// Get files that use setAttribute sanitized
+foreach(CxList sanitizer in fileSanitizers)
+{
    CSharpGraph fName = sanitizer.TryGetCSharpGraph<CSharpGraph>();
    sanitizedFiles.Add(fName.LinePragma.FileName);
 }

-if(sanitizedFiles.Count > 0){
-   //If the the result is in santized file then it should be removed
+if(sanitizedFiles.Count > 0)
+{
+   // If the the result is in santized file then it should be removed
    foreach(CxList htmlResult in htmlResults){
        CSharpGraph fName = htmlResult.TryGetCSharpGraph<CSharpGraph>();
        if(sanitizedFiles.Contains(fName.LinePragma.FileName)){
```

```
@@ -160,7 +191,7 @@

 htmlResults -= toRemove;

-//Get the React query results and remove the repeated results found in this query
+// Get the React query results and remove the repeated results found in this query
 CxList reactResults = React_Find_TargetBlank();
 foreach (CxList reactResult in reactResults)
 {
@@ -176,11 +207,11 @@
 result.Add(htmlResults);



-/*Add link elements creations and clicks. Ex:
-* const a = document.createElement("a")
-* a.href = untrustedURL
-* a.target = "_blank"
-* a.click()
+/* Add link elements creations and clicks. Ex:
+*  const a = document.createElement("a")
+*  a.href = untrustedURL
+*  a.target = "_blank"
+*  a.click()
 */
 CxList targetBlanks = stringLiterals.FindByShortName("_blank");
 CxList documentCreateElementA = methods.FindByMemberAccess("document.createElement");
@@ -188,18 +219,18 @@
 CxList documentCreateMembers = unknwnRefs.FindAllReferences(documentCreateElementA.GetAssignee()).GetMembersOfTarget();
 CxList clickMembers = documentCreateMembers.FindByShortName("click");

-//remove clicks that has sanitized rel value
-List<string> relValues = new List<string>{"noopener", "noreferrer", "noopener noreferrer", "noreferrer noopener"};
+// Remove clicks that has sanitized rel value
+string[] relValues = new []{"noopener", "noreferrer", "noopener noreferrer", "noreferrer noopener"};
 CxList relStrings = stringLiterals.FindByShortNames(relValues);
 clickMembers -= clickMembers.DataInfluencedBy(relStrings).GetLastNodesInPath();
 documentCreateMembers = unknwnRefs.FindAllReferences(clickMembers.GetLeftmostTarget()).GetMembersOfTarget();

-//filter clicks that have a href assignment
+// Filter clicks that have a href assignment
 CxList hrefValue = documentCreateMembers.FindByShortName("href").GetAssigner();
 clickMembers = clickMembers.DataInfluencedBy(hrefValue).GetLastNodesInPath();
 documentCreateMembers = unknwnRefs.FindAllReferences(clickMembers.GetLeftmostTarget()).GetMembersOfTarget();

-//filter clicks that have target "_blank"
+// Filter clicks that have target "_blank"
 CxList targetBlank = documentCreateMembers.FindByShortName("target").GetAssigner() * targetBlanks;
 clickMembers = clickMembers.DataInfluencedBy(targetBlank).GetLastNodesInPath();
```

JavaScript / JavaScript_Medium_Threat / CSV_Injection

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 // Find CSV Injection
 // Finds tainted data written to csv files
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(),Find_Cloud_Interactive_Inputs());
 if(cxScan.IsFrameworkActive("SAPUI"))
 {
    inputs.Add(Find_SAPUI_OData_Reads());
```

JavaScript / JavaScript_Medium_Threat / XML_External_Entities_XXE

Code changes

```
---
+++
@@ -1,4 +1,7 @@
-CxList inputs = Find_Inputs_NoWindowLocation();
+CxList inputs = All.NewCxList(
+   Find_Inputs_NoWindowLocation(),
+   Find_Cloud_Interactive_Inputs());
+
 inputs.Add(All.FindAllReferences(inputs.GetAssignee()));


 CxList fields = Find_Declarators();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Absolute_Path_Traversal

Code changes

JavaScript / JavaScript_Server_Side_Vulnerabilities / Cleartext_Storage_Of_Sensitive_Information

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList personalInfo = Find_Personal_Info();
-CxList cookieSet = Hapi_Find_Cookie_Set();
+CxList outputs = All.NewCxList(Hapi_Find_Cookie_Set());
 CxList sanitizers = NodeJS_Find_Encrypt();


-result = personalInfo.InfluencingOnAndNotSanitized(cookieSet, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
-result.Add(personalInfo * cookieSet - sanitizers);
+result = personalInfo.InfluencingOnAndNotSanitized(outputs, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result.Add(personalInfo * outputs - sanitizers);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Code_Injection

Code changes

```
---

+++

@@ -1,4 +1,4 @@

-CxList inputs = NodeJS_Find_Interactive_Inputs();

+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());

 CxList output = NodeJS_Find_Outputs_CodeInjection();

 CxList sanitize = NodeJS_Find_General_Sanitize();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Command_Injection

Code changes

```
---

+++

@@ -1,4 +1,4 @@

-CxList inputs = NodeJS_Find_Interactive_Inputs();

+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());

 CxList output = NodeJS_Find_Outputs_CommandInjection();

 CxList sanitize = NodeJS_Find_General_Sanitize();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Cookie_Poisoning

Code changes

```
---

+++

@@ -1,4 +1,4 @@

-CxList inputs = NodeJS_Find_Interactive_Inputs();

+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());

 CxList cookiesSet = Hapi_Find_Cookie_Set();

 CxList sanitize = NodeJS_Find_Encrypt();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Excessive_Data_Exposure

Code changes

```
---

+++

@@ -1,7 +1,7 @@

 CxList dbOut = NodeJS_Find_DB_Out();

 CxList personalInfo = Find_Personal_Info();

-CxList outputs = NodeJS_Find_Interactive_Outputs();

 CxList methods = Find_Methods();

+CxList outputs = All.NewCxList(NodeJS_Find_Interactive_Outputs());


 CxList sanitizers = methods.FindByName("*._.omit"); // lodash omit function

 sanitizers.Add(Find_Encrypt(), NodeJS_Find_General_Sanitize());
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Insecure_Storage_of_Sensitive_Data

Code changes

```
---
+++
@@ -36,7 +36,7 @@
    else if (secret.TryGetCSharpGraph<FieldDecl>() is FieldDecl field &&
            field.Declarators.Count > 0)
    {
-       var expr = field.Declarators[0].InitExpression;
+       Expression expr = field.Declarators[0].InitExpression;
        SourcesList.Add(expr.NodeId, expr);
    }
    else
@@ -69,9 +69,11 @@

 // remove results of methods that doesn't perform any change on database
 dbIn -= dbIn.GetByAncs(dbIn.GetAncOfType<MethodInvokeExpr>().FindByShortNames(nonStorageMethods));
+encrypt.Add(dbIn.GetTargetOfMembers());

-encrypt.Add(dbIn.GetTargetOfMembers());
-result.Add(SourcesList.InfluencingOnAndNotSanitized(dbIn, encrypt));
+CxList outputs = All.NewCxList(Find_Cloud_Storage_Outputs(), dbIn);
+
+result.Add(SourcesList.InfluencingOnAndNotSanitized(outputs, encrypt));

 // remove duplicate results where 'secret' and 'secret + token' are in sources
 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -11,8 +11,10 @@
             All_Passwords());
 personalInfo -= toRemove;

-CxList storage = NodeJS_Find_DB_IN();
-storage.Add(NodeJS_Find_Write());
+CxList storage = All.NewCxList(
+   NodeJS_Find_DB_IN(),
+   NodeJS_Find_Write(),
+   Find_Cloud_Outputs());

 result = personalInfo.InfluencingOnAndNotSanitized(storage, sanitizers)
    .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / MongoDB_NoSQL_Injection

Code changes

```
---
+++
@@ -17,7 +17,7 @@
 CxList methodsToIgnore = marsDbIn.FindByShortNames(ignoreMethods);
 marsDbIn = marsDbIn - methodsToIgnore;

-CxList inputs = NodeJS_Find_Interactive_Inputs() + NodeJS_Find_Read();
+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Inputs(), NodeJS_Find_Read());

 CxList sanitize = NodeJS_Find_Sanitize();
 // Consider only flows that contain the first parameter
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Parameter_Tampering

Code changes

```
---
+++
@@ -5,7 +5,7 @@
 CxList urMA = All.NewCxList();
 urMA.Add(uRef, mA);

-CxList input = NodeJS_Find_Interactive_Inputs();
+CxList input = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList db = NodeJS_Find_DB_IN();
 CxList strings = NodeJS_Find_Strings();
 //Support for select query as a string
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Plaintext_Storage_of_a_Password

Code changes

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

 CxList allNewtypes = All.NewCxList();
 allNewtypes.Add(decls, Find_FieldDecls(), Find_ObjectCreations(), Find_TypeRef(), unkRefs);
-CxList dbWrite = All.NewCxList();
+CxList dbWrite = Find_Cloud_Storage_Outputs();

 CxList excludeElements = methods - Find_Param();
 CxList nodeJsDbIn = All.GetParameters(NodeJS_Find_DB_IN());
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Privacy_Violation

Code changes

```
---
+++
@@ -10,12 +10,17 @@
 pi = pi.DataInfluencedBy(inputs).GetLastNodesInPath() +
     pi * inputs;
```

```
-CxList outputs = NodeJS_Find_Interactive_Outputs();
-outputs.Add(methods.FindByMemberAccesses(new String[]{"console.log"}, false));
+CxList outputs = All.NewCxList(
+   NodeJS_Find_Interactive_Outputs(),
+   methods.FindByMemberAccesses(new String[]{"console.log"}, false),
+   Find_Cloud_Interactive_Outputs());
+
 CxList loggers = methods.FindByName("*.winston.createLogger*").GetAssignee();
 outputs.Add( Find_UnknownReference().FindAllReferences(loggers).GetMembersOfTarget());
 CxList sanitizers = Find_Integers();
 sanitizers.Add(Find_Encrypt());
-result = outputs.InfluencedByAndNotSanitized(pi, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
-CxList infoOut = outputs * pi;
-result.Add(infoOut);
+
+CxList infoOut = outputs * pi;
+result.Add(
+   outputs.InfluencedByAndNotSanitized(pi, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow),
+   infoOut);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Reflected_XSS

Code changes

```
---
+++
@@ -9,7 +9,6 @@
 CxList react = Find_Require("React").GetMembersOfTarget();
 CxList createElement = react.FindByShortName("createElement");

-List <string> reactXssOutputs = new List<string> {"dangerouslySetInnerHTML", "href", "src"};
 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
@@ -19,8 +18,7 @@
 CxList renders = res.GetMembersOfTarget().FindByShortName("render");
 CxList rendersSanitizedParams = parameters.GetParameters(renders, 0);

-CxList inputs = NodeJS_Find_Interactive_Inputs();
-inputs.Add(AngularJS_Find_Inputs());
+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), AngularJS_Find_Inputs(), Find_Cloud_Interactive_Inputs());

 CxList sanitize = NodeJS_Find_XSS_Sanitize();
 sanitize.Add(Find_XSS_Sanitize());
@@ -67,7 +65,7 @@
    outputs -= NodeJS_Find_Swig_Interactive_Outputs();
 }

-CxList decls = Find_Declarators();
+CxList decls = All.NewCxList(declarators);
```

```
 decls.Add(paramDecls);


 CxList catchs = Find_Catch();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList outputs = NodeJS_Find_SQL_DB_IN();
-CxList inputs = NodeJS_Find_DB_Out();
+CxList inputs = All.NewCxList(NodeJS_Find_DB_Out(), Find_Cloud_Storage_Inputs());
 CxList sanitize = NodeJS_Find_Sanitize();


 result = outputs.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / SQL_Injection

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList outputs = NodeJS_Find_SQL_DB_IN().GetLastNodesInPath();
-CxList inputs = NodeJS_Find_Interactive_Inputs() + NodeJS_Find_Read();
+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs(), NodeJS_Find_Read());
 CxList sanitize = NodeJS_Find_Sanitize();


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

JavaScript / JavaScript_Server_Side_Vulnerabilities / Stored_Code_Injection

Code changes

```
---
+++
@@ -1,5 +1,4 @@
-CxList inputs = NodeJS_Find_Read();
-inputs.Add(NodeJS_Find_DB_Out());
+CxList inputs = All.NewCxList(NodeJS_Find_Read(), NodeJS_Find_DB_Out(), Find_Cloud_Storage_Inputs());


 CxList output = NodeJS_Find_Outputs_CodeInjection();
 CxList sanitize = NodeJS_Find_General_Sanitize();
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Stored_Path_Traversal

Code changes

```
---
+++
@@ -1,6 +1,5 @@
 CxList nodeDbOut = NodeJS_Find_DB_Out();
-CxList Inputs = All.NewCxList();
-Inputs.Add(nodeDbOut, NodeJS_Find_Read());
```

```
+CxList Inputs = All.NewCxList(nodeDbOut, NodeJS_Find_Read(),Find_Cloud_Storage_Inputs());

 ////////////////////////////////////////////////////////////////////////////////////////////////

 CxList methodInvoke = Find_Methods();

 CxList onlyParams = Find_Parameters();

@@ -37,6 +36,7 @@


 //in case path parameter of type ==>  aaa + bbb

 CxList partsOfPath = urMA.GetByAncs(pathOut);

+

 outputs.Add(pathOut, partsOfPath);

 if(Hapi_Find_Server_Instance().Count > 0)

 {
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Stored_XSS

Code changes

```
---

+++

@@ -1,11 +1,13 @@

 CxList methods = Find_Methods();


 //INPUTS (outputs from DB)

-CxList outputDB = NodeJS_DB_Output_Methods();

-outputDB.Add(NodeJS_Find_Mongoose_DB_Out(),

+CxList outputDB = All.NewCxList(

+            NodeJS_DB_Output_Methods(),

+            NodeJS_Find_Mongoose_DB_Out(),

             NodeJS_Find_Sequelize_DB_Out(),

             NodeJS_Find_Sqlite_DB_Out(),

-            NodeJS_Find_Read());

+            NodeJS_Find_Read(),

+            Find_Cloud_Storage_Inputs());


 CxList stats = All.FindByShortName("stat").FindByType<MethodInvokeExpr>();

 outputDB.Add(All.GetParameters(All.GetParameters(stats, 2), 1));
```

Kotlin / Kotlin_High_Risk / Deserialization_of_Untrusted_Data

Code changes

```
---

+++

@@ -1,20 +1,19 @@

-result = Common_High_Risk.Deserialization_of_Untrusted_Data();

 CxList objCreates = Find_ObjectCreations();

 CxList methods = Find_Methods();

 CxList inputs = Find_Inputs();


 CxList sanitizers = Find_Deserialization_Sanitizers();

-sanitizers.Add(objCreates.FindByMemberAccess("HessianFactory"));
```

```
-CxList outputs = All.NewCxList();
-outputs.Add(objCreates.FindByShortName("XStream"));
+CxList outputs = Find_Unsafe_Deserializers();
+


 CxList objRead = methods.FindByMemberAccesses(
    new string[]{
    "ObjectInputStream.readObject",
    "HessianInput.readObject",
    "Hessian2Input.readObject",
-   "Hessian2Input.readStreamingObject"
+   "Hessian2Input.readStreamingObject",
+   "XStream.fromXML"
    });


 outputs.Add(objRead);
@@ -27,8 +26,9 @@
        "ObjectOutputStream.writeObject",
        "HessianOutput.writeObject",
        "Hessian2Output.writeObject"
-   });
-CxList arraysFeedingReads = objCreates.FindByType("ByteArrayOutputStream").InfluencingOn(objRead).GetFirstNodesInPath();
+   }).GetTargetOfMembers();
+CxList arraysFeedingReads = objCreates.GetFathers().FindByType("ByteArrayOutputStream").InfluencingOn(objRead).GetFirstNodesInPath();
 outputs.Add(objWrites.InfluencedBy(arraysFeedingReads).GetLastNodesInPath());
+outputs -= outputs.InfluencedBy(methods.FindByMemberAccess("HessianFactory.setWhitelist"));


 result.Add(outputs.InfluencedByAndNotSanitized(inputs, sanitizers));
```

Kotlin / Kotlin_Medium_Threat / Use_of_Hardcoded_Cryptographic_Key

Code changes

```
---
+++
@@ -1,9 +1,13 @@
+CxList methods = Find_Methods();
+
 CxList secretKey = Find_ObjectCreations().FindByShortName("SecretKeySpec");
 CxList possibleParam = All.GetParameters(secretKey, 0);


-CxList sanitizers = All.GetParameters(Find_Methods().FindByMemberAccess("Cipher.getInstance"));
+CxList sanitizers = All.NewCxList();
+sanitizers.Add(
+   All.GetParameters(methods.FindByMemberAccess("Cipher.getInstance")),
+   methods.FindByMemberAccess("*.generateKey"));


 result = possibleParam.InfluencedByAndNotSanitized(Find_String_Literal(), sanitizers);


-CxList keys = Common_Medium_Threat.Use_of_Hard_coded_Cryptographic_Key();
```

```
-result.Add(keys);
+result.Add(Common_Medium_Threat.Use_of_Hard_coded_Cryptographic_Key());
```

PHP / PHP_Best_Coding_Practice / Exposure_of_Resource_to_Wrong_Sphere

Code changes

```
---
+++
@@ -1,18 +1,9 @@
-CxList allFields = All.FindByType(typeof(FieldDecl));
-CxList allPublicFields = allFields.FindByFieldAttributes(Modifiers.Public);
-CxList allConstFields = allFields.FindByFieldAttributes(Modifiers.Readonly);
+CxList classDecls = Find_ClassDecl();
+CxList allFields = Find_FieldDecls().GetByAncs(classDecls);
+CxList publicFields = allFields.FindByFieldAttributes(Modifiers.Public);
+publicFields -= allFields.FindByFieldAttributes(Modifiers.Readonly);

-CxList suspiciousResults = allPublicFields - allConstFields;
-CxList classDecl = All.FindByType(typeof(ClassDecl));
-//Remove results that are on the same line of class declaretion
-//in order to remove implicit class variable declaration
-List<KeyValuePair<string, int>> classList = new List<KeyValuePair<string, int>>();
-foreach (CxList decl in classDecl){
-   LinePragma lp = decl.GetFirstGraph().LinePragma;
-   classList.Add(new KeyValuePair<string, int>(lp.FileName,lp.Line));
-}
-foreach (CxList res in suspiciousResults){
-   LinePragma lp = res.GetFirstGraph().LinePragma;
-   if (!classList.Contains(new KeyValuePair<string, int>(lp.FileName,lp.Line)))
-       result.Add(res);
-}
+// remove results that are on the same line of class declaration
+// in order to remove implicit class variable declaration
+publicFields -= classDecls.FindByPositions(publicFields, CxList.CxPositionProximity.FindInLine, false);
+result = publicFields;
```

PHP / PHP_Best_Coding_Practice / Hardcoded_Absolute_Path

Code changes

```
---
+++
@@ -1,3 +1,4 @@
-result = All.FindByRegex(@"(?:\""|\')"
-   + @"(?:[a-z]|[A-Z])"
-   + @"\:(?:\\|\/)");
+result = Find_Strings().FindByRegex(@"(?:\""|\')"
+   + @"(?:[a-zA-Z])"
+   + @"\:(?:\\|\/)"
+   + @"[^""']*(?:""|')");
```

PHP / PHP_Best_Coding_Practice / Possible_Global_Variable_Overwrite

Code changes

```
---
+++
@@ -1,29 +1,25 @@
-//Get all method calls (extract, parse_str, mb_parse_str) that can overwrite global variables
-CxList possible_overwrite = Find_Methods().FindByShortName("extract");
-CxList uref = All.FindByType(typeof(UnknownReference));
+// Get all method calls (extract, parse_str, mb_parse_str) that can overwrite global variables.
+CxList methods = Find_Methods();
+CxList possibleOverwrite = methods.FindByShortName("extract", false);

-//vulnerability exists in PHP extract() methods with flags EXTR_OVERWRITE (default flag) and EXTR_IF_EXISTS.
-CxList non_overwrite_flags = uref.FindByShortNames(new List<String>()
-   { "EXTR_SKIP", "EXTR_PREFIX_SAME", "EXTR_PREFIX_ALL", "EXTR_PREFIX_INVALID", "EXTR_PREFIX_IF_EXISTS", "EXTR_REFS" });
-CxList toRemove = non_overwrite_flags.GetAncOfType(typeof(MethodInvokeExpr)).FindByShortName("extract");
+// Vulnerability exists in PHP extract() methods with flags EXTR_OVERWRITE (default flag) and EXTR_IF_EXISTS.
+CxList nonOverwriteFlags = Find_UnknownReference().FindByShortNames(
+   "EXTR_SKIP", "EXTR_PREFIX_SAME", "EXTR_PREFIX_ALL",
+   "EXTR_PREFIX_INVALID", "EXTR_PREFIX_IF_EXISTS", "EXTR_REFS");
+nonOverwriteFlags = nonOverwriteFlags.GetParameters(possibleOverwrite, 1);
+possibleOverwrite -= nonOverwriteFlags.GetAncOfType<MethodInvokeExpr>();

-//vulnerability exists in methods parse_str and mb_parse_str when called without result parameter.
-//When there is no result parameter, values are set on global variables.
-CxList allParams = Find_Params();
-CxList parseMethod = Find_Methods().FindByShortNames(new List<string>{"parse_str","mb_parse_str"});
-foreach (CxList method in parseMethod) {
-   if (allParams.GetParameters(method).Count < 2) {
-       possible_overwrite.Add(method);
-   }
-}
+// Vulnerability exists in methods parse_str and mb_parse_str when called without result parameter.
+// When there is no result parameter, values are set on global variables.
+CxList parseMethod = methods.FindByShortNames(new[]{ "parse_str", "mb_parse_str" }, false);
+possibleOverwrite.Add(parseMethod.FindByNumberOfParameters(1));
+possibleOverwrite -= All.NewCxList(
+   possibleOverwrite.GetMembersWithTargets(),
+   possibleOverwrite.GetTargetsWithMembers());

-toRemove.Add(possible_overwrite.GetTargetOfMembers().GetMembersOfTarget());
-toRemove.Add(possible_overwrite.GetMembersOfTarget().GetTargetOfMembers());
-possible_overwrite -= toRemove;
-
-// Get all methods calls that have an input in the parameter and are already in query Improper_Control_of_Dynamically_Identified_Variables
-CxList inputs = Find_Interactive_Inputs();
-CxList sanitize = Find_General_Sanitize();
-CxList possible_overwrite_inputs = possible_overwrite.InfluencedByAndNotSanitized(inputs, sanitize);
```

```
-
-result = possible_overwrite - possible_overwrite_inputs;
+// remove gloval variable override that are influenced by input
+possibleOverwrite -= possibleOverwrite
+   .InfluencedByAndNotSanitized(Find_Interactive_Inputs(), Find_General_Sanitize())
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
+   .GetLastNodesInPath();
+result = possibleOverwrite;
```

PHP / PHP_Best_Coding_Practice / Unchecked_Error_Condition

Code changes

```
---
+++
@@ -1,8 +1,8 @@
-CxList AllExc = All.FindByType(typeof(Catch));
+CxList catchClauses = Find_Catch();
+result = catchClauses.GetAncOfType<TryCatchFinallyStmt>();

-CxList notGenExc = All.FindAllReferences(AllExc) -
-                   All.FindByName("Exception", false).GetAncOfType(typeof(Catch));
-
-CxList genExc = AllExc - notGenExc;
-
-result = AllExc.GetFathers()  - (AllExc.GetFathers() * genExc.GetFathers());
+// remove general exception type error (Exception)
+CxList catchException = catchClauses
+   .CxSelectDomProperty<Catch>(c => c.CatchExceptionType)
+   .FindByShortName("Exception", false);
+result -= catchException.GetAncOfType<TryCatchFinallyStmt>();
```

PHP / PHP_Best_Coding_Practice / Unclosed_Objects

Code changes

```
---
+++
@@ -1,29 +1,24 @@
-CxList close = Find_Methods().FindByName("*.close", false);
-CxList AllTrys = All.GetAncOfType(typeof(TryCatchFinallyStmt));
-CxList fin = All.NewCxList();
-
-foreach(CxList oneTry in AllTrys)
+// get all close objects in try catch finally statements
+CxList closeOjects = Find_Methods().FindByMemberAccess("*.close", false);
+CxList tryCatchFinallyStmts = closeOjects.GetAncOfType<TryCatchFinallyStmt>();
+foreach(CxList tryCatchFinallyStmt in tryCatchFinallyStmts)
 {
-   TryCatchFinallyStmt t = oneTry.TryGetCSharpGraph<TryCatchFinallyStmt>();
-   fin.Add(All.FindById(t.Finally.NodeId));
-}
```

```
-fin = All.GetByAncs(fin);
-
-CxList Try = close.GetAncOfType(typeof(TryCatchFinallyStmt));
-foreach(CxList oneTry in Try)
-{
-   TryCatchFinallyStmt TryGraph = oneTry.TryGetCSharpGraph<TryCatchFinallyStmt>();
-   CxList curTry = All.FindById(TryGraph.Try.NodeId);
-   CxList TryClose = close.GetByAncs(curTry);
-   CxList AllClose = close.GetByAncs(oneTry);
+   // search for close objects in the current try statements
+   CxList tryStmt = tryCatchFinallyStmt.CxSelectDomProperty<TryCatchFinallyStmt>(t => t.Try);
+   CxList closesInTry = closeOjects.GetByAncs(tryStmt);


-   if( (AllClose - TryClose).Count == 0)
+   // search for close objects in try catch finally statements
+   CxList allCloses = closeOjects.GetByAncs(tryCatchFinallyStmt);
+
+   // if the list have the same elements
+   if(allCloses.Count == closesInTry.Count)
    {
-       if (TryClose.GetAncOfType(typeof(UsingStmt)).Count == 0)
-       {
-           result.Add(TryClose);
-       }
+       // add to result because the close objects
+       // are not closed in a finally statement
+       result.Add(closesInTry);
    }
 }
-
-result -= result.FindByFathers(fin);
+
+// remove all the close objects in finally statements
+CxList finallyStmt = tryCatchFinallyStmts.CxSelectDomProperty<TryCatchFinallyStmt>(t => t.Finally);
+result -= result.FindByFathers(finallyStmt);
```

PHP / PHP_Best_Coding_Practice / Use_Of_Namespace

Code changes

```
---
+++
@@ -1,3 +1,3 @@
-CxList namespaces = Find_NamespaceDecl();
-CxList globalNamespaces = namespaces.FindByShortName("$NS_*");
-result = namespaces - globalNamespaces;
+// get use of namespaces (ignore generated global namespaces, e.g. $NS_*)
+result = Find_NamespaceDecl();
+result -= result.FindByShortName("$NS_*");
```

PHP / PHP_Best_Coding_Practice / Use_Of_Private_Static_Variable

Code changes

```
---
+++
@@ -1,55 +1,15 @@
-//this query looks for explicit private static declaration of variables
-//static variable declaration inside a private method is considered static private
+// This query looks for explicit private static declaration of variables
+// Static variable declaration inside a private method is considered static private

-//find private static declaration of class members (FieldDecl)
-CxList variableDeclStatic = All.FindByType(typeof(FieldDecl))
-   .FindByFieldAttributes(Checkmarx.Dom.Modifiers.Static)
-   .FindByFieldAttributes(Checkmarx.Dom.Modifiers.Private);
+// Find private static declaration of class members (FieldDecl)
+CxList variableDeclStatic = Find_FieldDecls()
+   .FindByFieldAttributes(Modifiers.Static)
+   .FindByFieldAttributes(Modifiers.Private);

-CxList methodDecls = All.FindByType(typeof(MethodDecl));
-CxList unknownRefs = All.FindByType(typeof(UnknownReference));
-CxList declarators = All.FindByType(typeof(Declarator));
+// Find static variables inside private methods
+CxList privateMethods = Find_MethodDecls().FindByFieldAttributes(Modifiers.Private);
+CxList staticVariables  = Find_VariableDeclStmt().FindByFieldAttributes(Modifiers.Static);
+CxList staticVariablesInPrivateMethods = staticVariables.GetByAncs(privateMethods);
+CxList staticDeclaratorsInPrivateMethods = Find_Declarators().GetByAncs(staticVariablesInPrivateMethods);

-CxList privateMethodDecl = methodDecls.FindByFieldAttributes(Checkmarx.Dom.Modifiers.Private);
-
-//find static variables inside private methods
-CxList staticDeclaratorsInPrivateMethod = declarators.GetByAncs(privateMethodDecl).FindByRegex("static");
-
-//Build dicrionary of methodDecl line pragma by fileID
-Dictionary<int, HashSet<int>> methodLinePragmaByFiles = new Dictionary<int, HashSet<int>>();
-foreach(CxList methodDecl in privateMethodDecl)
-{
-   CSharpGraph methodDeclCSG = methodDecl.GetFirstGraph();
-
-   if(methodDeclCSG != null && methodDeclCSG.LinePragma != null && methodDeclCSG.LinePragma.Line != 0)
-   {
-       int fileID = methodDeclCSG.LinePragma.GetFileId();
-       int methodDeclLine = methodDeclCSG.LinePragma.Line;
-
-       if (!methodLinePragmaByFiles.ContainsKey(fileID))
-       {
-           methodLinePragmaByFiles.Add(fileID, new HashSet<int>());
-       }
```

```
-       methodLinePragmaByFiles[fileID].Add(methodDeclLine);
-   }
-}
-
-//remove declarators from line of MethodDecl
-foreach(CxList declarator in declarators)
-{
-   CSharpGraph declaratorCSG = declarator.GetFirstGraph();
-
-   if(declaratorCSG != null && declaratorCSG.LinePragma != null)
-   {
-       int fileID = declaratorCSG.LinePragma.GetFileId();
-       int declaratorLine = declaratorCSG.LinePragma.Line;
-
-       if(methodLinePragmaByFiles.ContainsKey(fileID) && methodLinePragmaByFiles[fileID].Contains(declaratorLine))
-       {
-           staticDeclaratorsInPrivateMethod -= declarator;
-       }
-   }
-}
-
-result.Add(variableDeclStatic);
-result.Add(staticDeclaratorsInPrivateMethod);
+result.Add(variableDeclStatic, staticDeclaratorsInPrivateMethods);
```

PHP / PHP_Best_Coding_Practice / Use_Of_Super_GLOBALS

Code changes

```
---
+++
@@ -1 +1 @@
-result  = All.FindByType(typeof(UnknownReference)).FindByRegex(@"\$GLOBALS");
+result = Find_Super_GLOBALS();
```

PHP / PHP_High_Risk / Code_Injection

Code changes

```
---
+++
@@ -1,27 +1,2 @@
-CxList methods = Find_Methods();
-CxList dynamic_method_invoke = methods.FindByShortNames(new List<string>()
-   { "$_Function", "call_user_func*", "forward_static_call*", "register_shutdown_function", "register_tick_function" });
-
-CxList dynamic_method_creation = methods.FindByShortNames(new List<string>() { "create_function", "eval", "xpath_eval" });
-CxList dynamic_variable_access = methods.FindByShortName("$_Variable");
-
-CxList influencedBydbRefFunc = All.InfluencedBy(All.FindByShortName("*Reflection*"));
-CxList reflectionInvoke = All.FindByShortName("invoke*").GetTargetOfMembers() * influencedBydbRefFunc;
-dynamic_method_invoke.Add(reflectionInvoke);
```

```
-
-CxList firstParam = dynamic_method_invoke;
-firstParam = All.GetParameters(firstParam, 0);
-
 CxList inputs = Find_Interactive_Inputs();
-CxList sanitize = Find_Code_Injection_Sanitize();
-
-result = inputs.InfluencingOnAndNotSanitized(dynamic_method_creation + dynamic_variable_access, sanitize);
-
-CxList arrays = methods.FindByShortName("array");
-
-CxList arraysSecondParam = All.GetParameters(arrays, 1);
-CxList arraysThirdParam = All.GetParameters(arrays, 2);
-CxList relevantArrays = arraysSecondParam.GetAncOfType(typeof(MethodInvokeExpr)) - arraysThirdParam.GetAncOfType(typeof(MethodInvokeExpr));
-CxList arraysFirstParam = All.GetParameters(relevantArrays, 0);
-
-result.Add(inputs.InfluencingOnAndNotSanitized(firstParam, sanitize + arraysFirstParam));
+result = Code_Injection_Sinks(inputs);
```

PHP / PHP_High_Risk / LDAP_Injection

Code changes

```
---
+++
@@ -1,13 +1,16 @@
-CxList methods = Find_Methods();
+// Generals
+CxList inputs = Find_Interactive_Inputs();
+CxList sanitized = Find_General_Sanitize();
+sanitized.Add(Find_LDAP_Replace());
+CxList ldapMethodsParams = Find_LDAP_Methods();

-CxList inputs = Find_Interactive_Inputs();
+// Filter LDAP methods that have parameters influenced by sanitizers
+CxList ldapEscapedParams = Find_UnknownReferences().FindAllReferences(sanitized.GetAssignee());
+CxList ldapEscapedMethodsFromParams = ldapEscapedParams.GetAncOfType<MethodInvokeExpr>();

-CxList sanitized = Find_General_Sanitize() + Find_LDAP_Replace();
+// Remove unwanted parameters from ldap methods parameters
+ldapMethodsParams -= ldapMethodsParams.GetByAncs(ldapEscapedMethodsFromParams);

-
-CxList ldap_find_methods = methods.FindByShortNames(new List<string>(){ "ldap_list", "ldap_read", "ldap_search"});
-
-CxList filter_params = All.GetParameters(ldap_find_methods, 2);
-
-result = filter_params.InfluencedByAndNotSanitized(inputs, sanitized);
-result.Add(filter_params * inputs);
+// Filter parameters from LDAP methods influenced by inputs/sanitizers and add to result
```

```
+result = ldapMethodsParams.InfluencedByAndNotSanitized(inputs, sanitized);
+result.Add(ldapMethodsParams * inputs);
```

PHP / PHP_High_Risk / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,5 +1,9 @@
-CxList dbOut = Find_DB_Out() + Find_Read() + Find_Session_Output();
-CxList dbIn = Find_DB_In();
+// Stored Inputs (database, files, streams, ...)
+CxList dbOut = All.NewCxList(Find_DB_Out(), Find_Read(), Find_Remote_Inputs());
+
+// Raw SQL queries
+CxList rawSql = Find_Raw_SQL_Query();
 CxList sanitize = Find_SQL_Injection_Sanitize();

-result = dbOut.InfluencingOnAndNotSanitized(dbIn, sanitize);
+result = dbOut.InfluencingOnAndNotSanitized(rawSql, sanitize)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

PHP / PHP_High_Risk / SQL_Injection

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList dbIn = Find_DB_In();
+CxList dbIn = Find_Raw_SQL_Query();
 CxList inputs = Find_Interactive_Inputs();
 CxList sanitized = Find_SQL_Injection_Sanitize();
```

PHP / PHP_High_Risk / Stored_XSS

Code changes

```
---
+++
@@ -1,11 +1,15 @@
-CxList db = Find_DB_Out() + Find_Read();
-CxList resFetch = Find_Methods().FindByShortName("*fetch*", false);
-resFetch -= resFetch.FindByShortNames(new List<String>(){ "*fetch_feed*", "*fetchmode*" }, false);
-db.Add(resFetch);
-db -= db.FindByShortName("*rss*", false);
+CxList outputs = Find_Interactive_Outputs();
+CxList sanitizers = Find_XSS_Sanitize();
+CxList stored = All.NewCxList();
+stored.Add(
+   Find_DB_Out(),
+   Find_Stored_Inputs()
```

```
+);


-CxList outputs = Find_Interactive_Outputs();
-CxList sanitize = Find_XSS_Sanitize();
-sanitize.Add(outputs * Find_ContentType_XSS_Sanitizers());
-
-result = db.InfluencingOnAndNotSanitized(outputs, sanitize) + Find_cURL_XSS();
+result.Add(
+   // Duplicated nodes
+   (stored * outputs) - sanitizers,
+   // Flow
+   stored.InfluencingOnAndNotSanitized(outputs, sanitizers)
+       .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
+);
```

PHP / PHP_High_Risk / XPath_Injection

Code changes

```
---
+++
@@ -1,35 +1,2 @@
-CxList methods = Find_Methods();
-
 CxList inputs = Find_Interactive_Inputs();
-CxList sanitized = Find_XPath_Sanitize();
-// Methods which get a filter (possibly tainted) in their first patameter.
-CxList XPath_parm1 =
-   methods.FindByMemberAccess("DOMXPath.evaluate") +
-   methods.FindByMemberAccess("DOMXPath.query") +
-   methods.FindByMemberAccess("XPathContext.xpath_eval_expression") +
-   methods.FindByMemberAccess("XPathContext.xpath_eval") +
-   methods.FindByMemberAccess("XPathContext.xptr_eval") +
-   methods.FindByMemberAccess("SimpleXMLElement.xpath");
-
-CxList memberMethods =
-   /*methods.FindByShortName("evaluate") +*/
-   /*methods.FindByShortName("query") +*/
-   methods.FindByShortNames(new List<String>{ "xpath_eval_expression", "xpath_eval", "xptr_eval", "xpath" });
-CxList unknownReferanceObj = All.FindByType(typeof(UnknownReference));
-XPath_parm1.Add(unknownReferanceObj.GetMembersOfTarget() * memberMethods);
-
-CxList tainted_parm1 = All.GetParameters(XPath_parm1, 0);
-CxList other_params1 = All.GetParameters(XPath_parm1) - tainted_parm1;
-other_params1.Add(XPath_parm1.GetTargetOfMembers());
-result = XPath_parm1.InfluencedByAndNotSanitized(inputs, sanitized + other_params1);
-
-// Methods which get a filter (possible tainted) in their second parameter.
-// As some are stand-alone functions with same name as class methods of the previous group,
-// remove the instances of the first group members from the second group.
```

```
-CxList XPath_parm2 = memberMethods + methods.FindByMemberAccess("SDO_DAS_Relational.executeQuery")
-   - XPath_parm1;
-
-CxList tainted_parm2 = All.GetParameters(XPath_parm2, 1);
-CxList other_params2 = All.GetParameters(XPath_parm2) - tainted_parm2;
-other_params2.Add(XPath_parm2.GetTargetOfMembers());
-result.Add(XPath_parm2.InfluencedByAndNotSanitized(inputs, sanitized + other_params2));
+result = Find_XPath_Injection(inputs);
```

PHP / PHP_Low_Visibility / Deprecated_Functions

Code changes

```
---
+++
@@ -1,6 +1,63 @@
 CxList methods = Find_Methods();
+CxList unk_refs = Find_UnknownReference();

-List<string> names = new List<string> {"mcrypt_ecb", "PDF_add_annotation",
+// 1. Deal with variables and constants
+string[] var_and_consts = new [] {
+       // PHP 8.2
+       "MYSQLI_IS_MARIADB",
+       // PHP 8.1
+       "FILTER_SANITIZE_STRING", "FILTER_SANITIZE_STRIPPED", "FILE_BINARY", "FILE_TEXT",
+       // PHP 8.0
+       "FILTER_SANITIZE_MAGIC_QUOTES", "AI_IDN_ALLOW_UNASSIGNED",
+       // PHP 7.3
+       "AI_IDN_USE_STD3_ASCII_RULES", "FILTER_FLAG_SCHEME_REQUIRED", "FILTER_FLAG_HOST_REQUIRED",
+       // PHP 7.2
+       "INTL_IDNA_VARIANT_2003",
+       // PHP 5.6
+       "HTTP_RAW_POST_DATA"
+       };
+
+CxList var_consts_usages = unk_refs.FindByShortNames(var_and_consts, false);
+// remove locally declared variables, as user can reuse deprecated variable names.
+var_consts_usages -= var_consts_usages.FindAllReferences(var_consts_usages.FindByAssignmentSide(CxList.AssignmentSide.Left));
+// add a deprecated member access as well
+result.Add(var_consts_usages,
+   Find_MemberAccesses().FindByMemberAccess("pdo.fetch_serialize", false));
+
+
+// 2. Direct methods
+string[] names = new [] {
+       // PHP 8.2
+       "utf8_encode", "utf8_decode",
+       // PHP 8.1
+       "date_sunrise", "date_sunset", "strptime", "strftime", "gmstrftime", "mhash",
```

```
+       "mhash_keygen_s2k", "mhash_count", "mhash_get_block_size", "mhash_get_hash_name",
+       "odbc_result_all",
+       // PHP 8.0
+       "enchant_broker_set_dict_path", "enchant_broker_get_dict_path",
+       "enchant_dict_add_to_personal", "enchant_dict_is_in_session",
+       "enchant_broker_free", "enchant_broker_free_dict", "get_magic_quotes_gpc",
+       "get_magic_quotes_runtime", "hebrevc", "convert_cyr_string", "money_format",
+       "ezmlm_hash", "restore_include_path", "openssl_pkey_free", "openssl_free_key",
+       "openssl_x509_free", "shmop_close", "zip_read", "zip_open", "zip_close",
+       "zip_entry_read", "zip_entry_close", "zip_entry_open", "zip_entry_name",
+       "zip_entry_filesize", "zip_entry_compressionmethod", "zip_entry_compressedsize",
+       // PHP 7.4
+       "ldap_control_paged_result", "ldap_control_paged_result_response",
+       // PHP 7.3
+       "fgetss", "gzgetss", "mbereg", "mberegi", "mbereg_replace", "mberegi_replace",
+       "mbsplit", "mbereg_match", "mbereg_search", "mbereg_search_pos",
+       "mbereg_search_regs", "mbereg_search_init", "mbereg_search_getregs",
+       "mbereg_search_getpos", "mbereg_search_setpos", "image2wbmp",
+       // PHP 7.2
+       "png2wbmp", "jpeg2wbmp", "create_function", "gmp_random", "each", "read_exif_data",
+       // PHP 7.1
+       "mcrypt_*", "mdecrypt_generic",
+       // PHP 7.0
+       "ldap_sort",
+
+       // those were in the old version of the query, and were kept
+       "PDF_add_annotation",
        "PDF_add_bookmark", "PDF_add_launchlink", "PDF_add_locallink", "PDF_add_note", "PDF_add_outline",
        "PDF_add_pdflink", "PDF_add_weblink", "PDF_attach_file", "PDF_begin_page", "PDF_begin_template",
        "PDF_close", "PDF_close_pdi", "PDF_findfont", "PDF_get_font", "PDF_get_fontname", "PDF_get_fontsize",
@@ -14,19 +71,116 @@
        "PDF_set_info_subject", "PDF_set_info_title", "PDF_set_leading", "PDF_set_text_matrix",
        "PDF_set_text_rendering", "PDF_set_text_rise", "PDF_set_word_spacing", "PDF_show_boxed",
        "px_set_tablename", "px_set_targetencoding" };
+
+CxList methodCalls = methods.FindByShortNames(names, false);
+methodCalls -= methodCalls.FindByMemberAccess(".*");
+result.Add(methodCalls);
+
+
+// 3. Deal with member method calls
+string[] memberNames = new [] {
+       // PHP 8.2
+       "SplFileInfo._bad_state_ex",
+       // PHP 8.0
+       "ReflectionZendExtension.export", "ReflectionExtension.export",
+       // PHP 7.4
+       "Reflector.export",
```

```
+       // PHP 7.3
+       "SplFileObject.fgetss",

-List<string> memberNames = new List<string> {"GearmanClient.data", "GearmanClient.do", "GearmanClient.echo",
+       // ImageMagick
+       "Imagick.getImageIndex", "Imagick.filter", "Imagick.orderedPosterizeImage",
+       "Imagick.flattenImages", "Imagick.getImageMatteColor", "Imagick.setImageIndex",
+       "Imagick.setImageBiasQuantum", "Imagick.clone", "Imagick.setImageClipMask",
+       "Imagick.getImageExtrema", "Imagick.paintFloodfillImage", "Imagick.getImageChannelExtrema",
+       "Imagick.radialBlurImage", "Imagick.getImageAttribute", "Imagick.transformImage",
+       "Imagick.setImageAttribute", "Imagick.setImageOpacity", "Imagick.roundCorners",
+       "Imagick.averageImages", "Imagick.getImageClipMask", "Imagick.setImageBias", "Imagick.recolorImage",
+       "Imagick.setImageMatteColor", "Imagick.colorFloodfillImage", "Imagick.mapImage",
+       "Imagick.paintOpaqueImage", "Imagick.mosaicImages",
+
+       // Old entries in the original query
+       "GearmanClient.data", "GearmanClient.do", "GearmanClient.echo",
        "GearmanClient.setClientCallback", "GearmanClient.setData", "GearmanJob.complete", "GearmanJob.data",
        "GearmanJob.exception", "GearmanJob.fail", "GearmanJob.status", "GearmanJob.warning", "GearmanTask.create",
        "GearmanTask.function", "GearmanTask.sendData", "GearmanTask.uuid", "MongoClient.dropDB",
        "MongoCursor.slaveOkay", "MongoDB.dropCollection", "MongoDB.execute", "SoapClient.__call" };

-foreach (var name in names)
-{
-   result.Add(methods.FindByName(name));
-}
+result.Add(methods.FindByMemberAccesses(memberNames, false));

-foreach (var memberName in memberNames)
-{
-   result.Add(methods.FindByMemberAccess(memberName));
-}
+
+// 2.1 Specific calls not possible to detect by find by member access
+//    "ReflectionParameter.isCallable", "ReflectionParameter.getClass",
+CxList getReflectionParameters = unk_refs.FindByType("Reflection*", false).GetAncOfType<MethodInvokeExpr>().FindByMemberAccess(".getParameters", false);
+result.Add(methods.FindByMemberAccesses(new[]{".isCallable",".getClass"}).InfluencedBy(getReflectionParameters).GetLastNodesInPath());
+
+
+// 3. Deal with calls whose deprecation is based on parameter count or parameter type
+// auxiliary lists and methods
+CxList parameters = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
+CxList str_parameters = parameters.FindByAbstractValue(av => av is StringAbstractValue);
+Func < IAbstractValue, string[], bool> StrAbsVal = (av, values) => {
+   StringAbstractValue sav = av as StringAbstractValue;
+   return Array.FindIndex(values, str => str.Equals(sav.Content, StringComparison.OrdinalIgnoreCase)) >= 0;
+   };
+
```

```
+
+// 3.1. Methods with deprecated parameter count
+CxList mysqli_get_client_info = All.NewCxList();
+mysqli_get_client_info.Add(
+   methods.FindByMemberAccess("mysqli.get_client_info", false),
+   methods.FindByShortName("mysqli_get_client_info", false));
+mysqli_get_client_info -= mysqli_get_client_info.FindByNumberOfParameters(0);
+
+result.Add(
+   // can't be used with 0 parameters
+   methods.FindByShortName("mb_check_encoding", false).FindByNumberOfParameters(0),
+   // can't be used with 1 parameter
+   methods.FindByShortName("parse_str", false).FindByNumberOfParameters(1),
+   // these can't be used with parameters (just empty list of parameters)
+   mysqli_get_client_info
+   );
+
+
+// 3.2. Password_hash options dictionary can't contain the keyword 'salt'
+CxList password_hash = methods.FindByShortName("password_hash", false);
+CxList password_hash_parameters = parameters.GetParameters(password_hash, 2);
+password_hash_parameters.Add(
+   unk_refs.FindAllReferences(password_hash_parameters)
+   .FindByAssignmentSide(CxList.AssignmentSide.Left)
+   .GetAssigner());
+CxList dicts = password_hash_parameters.FindByType<AssociativeArrayExpr>();
+CxList declarators = Find_Declarators().FindByShortName("salt");
+result.Add(declarators.GetByAncs(dicts));
+
+
+string[] needle_methods = new[]{"strpos","stripos","stristr","strstr","strrchr","strripos","strrpos"};
+result.Add(
+   // 3.3. String search methods can't be used with an integer search value (integer needle)
+   parameters.FindByAbstractValue(x => x is IntegerIntervalAbstractValue).GetParameters(
+   methods.FindByShortNames(needle_methods, false), 1).GetAncOfType<MethodInvokeExpr>(),
+   // 3.4. Arrays/Objects can't be passed as implode first parameter
+   parameters.FindByAbstractValue(x => x is ObjectAbstractValue).GetParameters(
+   methods.FindByShortName("implode", false), 0).GetAncOfType<MethodInvokeExpr>(),
+   // 3.5. False can't be used as first parameter of get_defined_functions
+   parameters.FindByAbstractValue(x => x is FalseAbstractValue).GetParameters(
+   methods.FindByShortName("get_defined_functions", false), 0).GetAncOfType<MethodInvokeExpr>());
+
+
+// 3.6. Methods mb_convert_encoding and stream_filter_append can't be called with specific strings
+CxList mb_convert_encoding_dep_params = str_parameters.FindByAbstractValue(av =>
+   StrAbsVal(av, new[]{"QPrint", "Base64", "Uuencode", "HTML-ENTITIES"}));
+CxList mb_convert_encoding = methods.FindByShortName("mb_convert_encoding", false);
+
```

```
+CxList string_strip_tags_dep_params = str_parameters.FindByAbstractValue(av =>
+   StrAbsVal(av, new[]{"string.strip_tags"}));
+CxList string_strip_tags = methods.FindByShortName("stream_filter_append", false);
+
+result.Add(
+   string_strip_tags_dep_params.GetParameters(string_strip_tags, 1).GetAncOfType<MethodInvokeExpr>(),
+   mb_convert_encoding_dep_params.GetParameters(mb_convert_encoding, 1).GetAncOfType<MethodInvokeExpr>(),
+   mb_convert_encoding_dep_params.GetParameters(mb_convert_encoding, 2).GetAncOfType<MethodInvokeExpr>());
+
+
+// 3.7. Deprecated the definition of __autoload method
+result.Add(Find_MethodDecls().FindByShortName("__autoload", false));
```

PHP / PHP_Low_Visibility / Improper_Exception_Handling

Code changes

```
---
+++
@@ -1,15 +1,7 @@
-CxList allDbInvocations = Find_DB_Methods().FindByType(typeof(MethodInvokeExpr));
-
-CxList ChildrenOfTry = allDbInvocations.GetByAncs(Find_TryCatchFinallyStmt());
-
-// locate error related methods to filter out
-CxList ignoreMethods = allDbInvocations.FindByShortNames(new List<String>(){ "*error*", "*errno*", "sqlstate" });
-
-// Identify suppresses warnings and errors and to ignore list
-CxList suppresseMethods = Find_CustomAttribute().FindByShortName("@").GetAncOfType<MethodInvokeExpr>();
-ignoreMethods.Add(suppresseMethods);
-
-// list of DB functions that throw exceptions
-CxList dbMethods = allDbInvocations - ignoreMethods;
-result = dbMethods;
-result -= ChildrenOfTry;
+result.Add(
+   Improper_Method_Exception_Handling(),
+   Improper_Operator_Exception_Handling(),
+   Improper_Class_Exception_Handling(),
+   Improper_UserFunc_Exception_Handling(),
+   Improper_Match_Exception_Handling()
+);
```

PHP / PHP_Low_Visibility / Improper_Transaction_Handling

Code changes

```
---
+++
@@ -1,26 +1,22 @@
-CxList Commit = All.FindByName("*commit*", false);
-CxList Rollback = All.FindByName("*rollback*", false);
```

```
+// get PDO operations
+CxList pdoTypes = Find_UnknownReference().FindByType("PDO", false);
+CxList pdoMethods = pdoTypes.GetMembersOfTarget().FindByType<MethodInvokeExpr>();

-CxList TryBlock = Commit.GetAncOfType(typeof(TryCatchFinallyStmt));
-foreach(CxList cml in TryBlock)
-{
-   TryCatchFinallyStmt TryGraph = cml.TryGetCSharpGraph<TryCatchFinallyStmt>();
+// get 'commit' operations (from 'msysqli' and 'PDO')
+CxList methods = Find_Methods();
+CxList commit = All.NewCxList(
+   pdoMethods.FindByShortName("commit", false),
+   methods.FindByShortName("mysqli_commit", false));

-   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);
+// get 'rollback' operations (from 'msysqli' and 'PDO')
+CxList rollback = All.NewCxList(
+   pdoMethods.FindByShortName("rollback", false),
+   methods.FindByShortName("mysqli_rollback", false));

+// get catch clauses surrounding 'commit' operation
+CxList tryCatchs = commit.GetAncOfType<TryCatchFinallyStmt>();
+CxList catchClauses = tryCatchs.CxSelectElements<TryCatchFinallyStmt>(t => t.CatchClauses);

-   if( (RollbackInCatch.GetAncOfType(typeof(TryCatchFinallyStmt)) *
-       CommitInTry.GetAncOfType(typeof(TryCatchFinallyStmt))).Count == 0)
-   {
-       result.Add(cml);
-   }
-}
+// add to result try catchs without 'rollback' operation
+catchClauses -= rollback.GetByAncs(catchClauses).GetAncOfType<Catch>();
+result = catchClauses.GetAncOfType<TryCatchFinallyStmt>();
```

PHP / PHP_Low_Visibility / Information_Exposure_Through_an_Error_Message

Code changes

```
---
+++
@@ -1,23 +1,34 @@
```

```
-CxList ctch = All.FindByType(typeof(Catch));
-CxList outputs = Find_Interactive_Outputs();
-
-CxList exc = All.FindAllReferences(ctch) - ctch;
-
+CxList inputs = All.NewCxList();
 CxList methods = Find_Methods();


-CxList errOuts = methods.FindByShortNames(new List<String>(){"error_reporting", "phpinfo"});
-//exclude those that are error_reporting(0);
-CxList ints = All.FindByType(typeof(IntegerLiteral));
-CxList zeroSource = ints.FindByShortName("0");
-CxList zeroSink = errOuts.FindByParameters(zeroSource);
-errOuts -= zeroSink;
-// Find cases such as ini_set('error_reporting', [PARAM]);
-CxList setters = methods.FindByShortName("ini_set");
-CxList source = Find_Strings().FindByShortName("error_reporting");
-CxList errSetters = methods.FindByParameters(source.GetParameters(setters, 0));
-//exclude those that are init_set('error_reporting, 0);
-CxList excludeMethods = methods.FindByParameters(zeroSource.GetParameters(setters, 1));
-errSetters -= excludeMethods;
+// 1. Exceptions
+inputs.Add(Find_Declarators().GetByAncs(Find_VariableDeclStmt().FindByFathers(Find_Catch())));
+// 2. Error Messages
+string[] error_methods = new[]{
+       "error_get_last",
+       "libxml_get_errors","libxml_get_last_error",
+       "xml_get_error_code","xml_error_string",
+       "openssl_error_string",
+       "preg_last_error", "preg_last_error_msg",
+       "json_last_error", "json_last_error_msg",
+       "curl_errno", "curl_strerror", "curl_error"};
+CxList methodSinks = methods.FindByShortNames(error_methods, false);
+methodSinks -= methodSinks.FindByMemberAccess(".*");


-errOuts += errSetters;
-result = outputs.DataInfluencedBy(exc) + errOuts;
+string[] error_accessors = new []{
+   "DateTime.getLastErrors", "mysqli.error"
+};
+
+inputs.Add(
+   methodSinks,
+   methods.FindByMemberAccesses(error_accessors, false),
+   Find_UnknownReference().FindByShortName("php_errormsg")
+);
+
+// 3. Sinks
```

```
+CxList sinks = Find_Interactive_Outputs();
+sinks.Add(sinks.FindByType<MethodRef>().GetFathers());
+
+result.Add(
+   sinks.InfluencedBy(inputs).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow),
+   sinks * inputs);
```

PHP / PHP_Low_Visibility / Information_Leak_Through_Persistent_Cookies

Code changes

```
---
+++
@@ -1,10 +1,9 @@
-CxList psw = Find_Passwords();
-psw.Add(Find_Password_Strings());
-CxList methods = Find_Methods();
-CxList cookie = methods.FindByName("*setcookie*", false);
-cookie.Add(methods.FindByName("*setrawcookie*", false));
+CxList personalInfo = Find_Personal_Info();
+personalInfo.Add(Find_Passwords(), Find_Password_Strings());

-List<String> cookie_arrays = new List<String>(){ "*_COOKIE*", "*HTTP_COOKIE_VARS" };
+CxList sanitizers = Find_Encryption_Functions();
+sanitizers.Add(Find_Hashing_Functions());

-result = cookie.FindByParameters(psw);
-result.Add(psw.GetAncOfType(typeof(IndexerRef)).FindByShortNames(cookie_arrays, false));
+result = personalInfo
+   .InfluencingOnAndNotSanitized(Find_Set_Cookie(), sanitizers)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

PHP / PHP_Low_Visibility / Log_Forging

Code changes

```
---
+++
@@ -1,5 +1,42 @@
-CxList Inputs = Find_Interactive_Inputs();
-CxList Log = Find_Log_Outputs();
+CxList inputs = Find_Interactive_Inputs();
+CxList log = Find_Log_Outputs();
 CxList sanitize = Find_General_Sanitize();
+
+
+// Add further sanitizers that replace new lines
+CxList methods = Find_Methods();
+CxList parameters = Find_Param().CxSelectDomProperty<Param>(x => x.Value);
+CxList str_literals = Find_String_Literal();
+string[] nls = new[]{"\\n", "\\r\\n"};
+CxList new_lines = str_literals.FindByShortNames(nls);
```

```
+new_lines.Add(Find_UnknownReference().FindByShortName("PHP_EOL"));
+
+// str_replace
+CxList str_replaces = methods.FindByShortName("str_replace", false);
+CxList str_replace_params = parameters.GetParameters(str_replaces, 0);
+CxList new_line_params = str_replace_params.InfluencedBy(new_lines).GetLastNodesInPath();
+new_line_params.Add(str_replace_params * new_lines);
+// preg_replace
+CxList preg_replaces = methods.FindByShortName("preg_replace", false);
+CxList preg_replaces_params = str_literals.GetParameters(preg_replaces, 0);
+preg_replaces = preg_replaces_params.Filter(x => x.ShortName.Contains(@"\n")).GetAncOfType<MethodInvokeExpr>();
+
+// strtr, 2 or 3 parameters
+CxList strtr3 = methods.FindByShortName("strtr", false);
+CxList strtr2 = strtr3.FindByNumberOfParameters(2);
+strtr3 = strtr3.FindByNumberOfParameters(3);
+/// 3 parameters
+strtr3 = str_literals.GetParameters(strtr3, 1).Filter(x => x.ShortName.Contains(@"\n")).GetAncOfType<MethodInvokeExpr>();
+/// 2 parameters
+CxList declarators = Find_Declarators().FindByShortNames(nls);
+CxList strtr2Params = parameters.GetParameters(strtr2, 1).InfluencedBy(declarators).GetLastNodesInPath();
+strtr2 = strtr2.InfluencedBy(strtr2Params).GetLastNodesInPath();
+
+
+sanitize.Add(
+   strtr2, strtr3, preg_replaces, str_replaces.InfluencedBy(new_line_params).GetLastNodesInPath()
+   );


-result = Log.InfluencedByAndNotSanitized(Inputs, sanitize);
+// file_put_contents and other act as sanitizers, but are not a sanitizer when they are the sink
+sanitize -= log;
+result = log.InfluencedByAndNotSanitized(inputs, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

PHP / PHP_Low_Visibility / Reliance_on_DNS_Lookups_in_a_Decision

Code changes

```
---
+++
@@ -1,7 +1,8 @@
-CxList cond = All.GetByAncs(Find_Conditions());
-CxList methods = Find_Methods();
-CxList ip = methods.FindByShortNames(new List<String>()
-   { "checkdnsrr", "dns_check_record", "dns_get_mx", "dns_get_record", "gethostbyaddr", "getmxrr" });
+// methods capable of performing a reverse DNS lookup
+CxList reverseDnsLookups = Find_Methods()
+   .FindByShortNames(new[]{ "gethostbyaddr", "dns_get_record" }, false);


-result = ip.DataInfluencingOn(cond) + cond * ip;
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

```
+CxList cond = Find_Conditions();
+result = cond * reverseDnsLookups;
+result.Add(reverseDnsLookups.DataInfluencingOn(cond)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow));
```

PHP / PHP_Low_Visibility / Trust_Boundary_Violation_in_Session_Variables

Code changes

```
---
+++
@@ -1,9 +1,20 @@
-CxList session = All.FindByShortName(@"_SESSION_*");
-CxList leftSideSession = session.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList inputs = Find_Inputs() - leftSideSession;
-CxList sanitize = Find_General_Sanitize();
+CxList inputs = Find_Inputs();
+inputs.Add(Find_Stored_Inputs());

-CxList methods = Find_Methods();
-CxList memcache = methods.FindByShortName("memcache_set") + All.FindByMemberAccess("Memcache.set");
+CxList session = Find_IndexerRefs().FindByShortName("_SESSION");
+session = session.FindByAssignmentSide(CxList.AssignmentSide.Left);
+inputs -= session;

-result = (leftSideSession + memcache).InfluencedByAndNotSanitized(inputs, sanitize);
+// general sanitizers
+CxList sanitizers = Find_General_Sanitize();
+
+// add 'preg_match' method in condition as sanitizers
+CxList pregMatch = Find_Methods().FindByShortName("preg_match", false);
+sanitizers.Add(Find_UnknownReference().GetSanitizerByMethodInCondition(pregMatch));
+
+// add inputs in conditions as sanitizers
+CxList sessionInIfs = session.GetAncOfType<IfStmt>();
+CxList sessionConditions = sessionInIfs.CxSelectDomProperty<IfStmt>(i => i.Condition);
+sanitizers.Add(inputs.InfluencingOn(sessionConditions).GetFirstNodesInPath());
+
+result = session.InfluencedByAndNotSanitized(inputs, sanitizers);
```

PHP / PHP_Low_Visibility / Unsafe_Use_Of_Target_Blank

Code changes

```
---
+++
@@ -1,114 +1,76 @@
-CxList methods = Find_Methods();
+// General
+CxList inputs = Find_Interactive_Inputs();
 CxList strings = Find_Strings();
-CxList outputs = Find_Outputs();
```

```
-//1st part - Output of html anchor strings with the target="_blank" vulnerability
+// Regex Patterns

-//<a...
-string anchorP = @"<a\s+[^>]+";
-//target="_blank"
-string targetP = @"target\s*=\s*('|""|\\"")_?blank('|""|\\"")";
+// HTML anchor with href attribute
+string anchorHREFPattern = @"<a\s+href((?<anch><)|(?<-anch>>)|[^<>])+>(?(anch)(?!))";
+// target="_blank"
+string targetPattern = @"target\s*=\s*('|""|\\"")_?blank('|""|\\"")";
 //rel="noopener" or rel="noopener noreferrer" (sanitizer)
-string relP = @"rel\s*=\s*('|""|\\"")noopener|noopener\s+noreferrer('|""|\\"")";
+string relPattern = @"rel\s*=\s*('|""|\\"")?(noopener\s?|noreferrer\s?)+('|""|\\"")?";

-CxList targetMatches = strings.FindByRegex(targetP);
-//Remove FP (may occur with php embed in html)
-foreach (CxList tM in targetMatches)
+// 1.1 - target=_blank on HTML
+
+// Find for all HTML comments from href pattern
+CxList hrefMatches = All.FindByRegexExt(anchorHREFPattern, "*.*", true, RegexOptions.IgnoreCase);
+
+// Target and Rel Regex to be used on foreach
+Regex regexTarget = new Regex(targetPattern, RegexOptions.Compiled);
+Regex regexRel = new Regex(relPattern, RegexOptions.Compiled);
+
+// Check if the href match has target=_blank and sanitizers
+foreach (CxList href in hrefMatches)
 {
-   CSharpGraph csg = tM.TryGetCSharpGraph<CSharpGraph>();
-   if(!csg.FullName.Contains("blank"))
+   Comment match = href.TryGetCSharpGraph<Comment>();
+   string matchText = match.FullName;
+
+   Match matchTarget = regexTarget.Match(matchText);
+
+   if(matchTarget.Success)
    {
-       targetMatches -= tM;
+       Match matchRel = regexRel.Match(matchText);
+
+       if(!matchRel.Success)
+           result.Add(href);
    }
 }
```

```
-//Find MethodInvokes and AssignExprs which are Ancs of target="_blank" strings
-CxList targetMatchesAncs = targetMatches.GetAncOfType(typeof(MethodInvokeExpr));
-targetMatchesAncs.Add(targetMatches.GetAncOfType(typeof(AssignExpr)));
+// 1.2 - HREF attributes influenced by target=_blank

-foreach (CxList anc in targetMatchesAncs)
+// Create a list of references, strings and expressions
+CxList refsStrsExprs = All.NewCxList(Find_UnknownReference(), strings, Find_ExprStmt());
+
+// Find for href references
+CxList unsafeAttrs = refsStrsExprs.FindByRegex(anchorHREFPattern);
+
+// Get target references from targetPattern
+CxList targetRefs = refsStrsExprs.FindByRegex(targetPattern);
+
+// Remove unwanted target references and add to inputs list
+targetRefs -= targetRefs.FilterByDomProperty<CSharpGraph>(
+   str => !str.FullName.Contains("target"));;
+inputs.Add(targetRefs);
+
+// Get all strings that do not contains noopener and noreferrer
+CxList strSanitizers = strings - strings.FindByShortNames(new [] {"*noopener*", "*noreferrer*" }, false);
+
+// Check the flow between href attribute and inputs (target=_blank)
+foreach (CxList un in unsafeAttrs)
 {
-   //Get all string children, in case of String concat
-   CxList strs = All.GetByAncs(anc).FindByType(typeof(StringLiteral));
-
-   //MethodInvokes and AssignExprs which are Ancs of <a... strings
-   CxList anchorMatches = strs.FindByRegex(anchorP);
-   CxList anchorMatchesAncs = anchorMatches.GetAncOfType(typeof(MethodInvokeExpr));
-   anchorMatchesAncs.Add(anchorMatches.GetAncOfType(typeof(AssignExpr)));
-
-   CxList possibleVuln = anchorMatchesAncs * targetMatchesAncs;
-   //Are there anchors and target="_blank" which share the same father?
-   if (possibleVuln.Count > 0)
+   // Get all references, strings and expressions from href attribute
+   CxList hrefAttributes = refsStrsExprs.GetByAncs(un.GetAncOfType<MethodInvokeExpr>());
+
+   // Get all target=_blank reference on href attribute
+   CxList targetRefOnHREF = hrefAttributes.FindAllReferences(targetRefs);
+
+   // Get all sanitizers on href attribute
+   CxList sanitizersFromHREF = hrefAttributes * strSanitizers;
+
+   if(targetRefOnHREF.Count == 1 && sanitizersFromHREF.Count > 0)
    {
```

```
-       //If so find Ancs of sanitizers (rel)
-       CxList relMatches = strs.FindByRegex(relP);
-       CxList relMatchesAncs = relMatches.GetAncOfType(typeof(MethodInvokeExpr));
-       relMatchesAncs.Add(relMatches.GetAncOfType(typeof(AssignExpr)));
-
-       CxList vuln = possibleVuln * relMatchesAncs;
-       //Do (anchors and target="_blank") share father with sanitizer?
-       if (vuln.Count == 0)
-       {
-           //If no, it's a vulnerability
-           CxList v = All.GetByAncs(anc).FindByType(typeof(StringLiteral)).FindByRegex(targetP);
-           result.Add(outputs.InfluencedBy(v));
-       }
+       // Filter for target=_blank reference influenced by inputs and add to result
+       result.Add(targetRefOnHREF.InfluencedBy(inputs));
    }
 }
-
-//2nd part - Creation of DOMDocuments with the target="_blank" vulnerability
-
-//DOMDocument methods
-CxList createElem = methods.FindByShortNames(new List<string>{ "createElement", "DOMElement" });
-CxList createAttr = methods.FindByShortName("createAttribute");
-CxList appendChd = methods.FindByShortName("appendChild");
-
-//Anchor creation
-CxList anchors = All.GetParameters(createElem).FindByShortName("a");
-//With href attribute
-CxList hrefs = All.GetParameters(createAttr).FindByShortName("href");
-//with target attribute
-CxList targets = All.GetParameters(createAttr).FindByShortName("target");
-//with rel atrtibute
-CxList rels = All.GetParameters(createAttr).FindByShortName("rel");
-
-//'_blank' value of target attribute
-CxList _blank = strings.FindByShortName("_blank");
-//'noopener' value of rel attribute
-CxList noopener = strings.FindByShortName("noopener");
-//'noopener noreferrer' value of rel attribute
-CxList noopener_noreferrer = strings.FindByShortName("noopener noreferrer");
-
-//Get assignee of the anchor elements
-CxList aVars = All.GetByAncs(All.FindByParameters(anchors).GetTargetOfMembers().GetAncOfType(typeof(AssignExpr)))
-   .FindByAssignmentSide(CxList.AssignmentSide.Left);
-
-//for each anchor element
-foreach (CxList a in aVars)
-{
```

```
-   CxList aVarRefs = All.FindAllReferences(a);
-   //Check if it has href attribute
-   CxList hasHref = aVarRefs.InfluencedBy(hrefs);
-   if (hasHref.Count > 0)
-   {
-       //Check if it has target attribute
-       CxList hasTarget = aVarRefs.InfluencedBy(targets);
-       if (hasTarget.Count > 0)
-       {
-           //Check if it has '_blank' value
-           CxList isBlank = aVarRefs.InfluencedBy(_blank);
-           if (isBlank.Count > 0)
-           {
-               //Check if it's missing rel attribute with 'noopener' or 'noopener noreferrer' value
-               CxList hasRel = aVarRefs.InfluencedBy(rels);
-               CxList isNoOpener = aVarRefs.InfluencedBy(noopener);
-               CxList isNoOpenerNoReferrer = aVarRefs.InfluencedBy(noopener_noreferrer);
-               if (hasRel.Count == 0 || (isNoOpener.Count == 0 && isNoOpenerNoReferrer.Count == 0))
-               {
-                   //Possible vulnerability
-                   result.Add(isBlank);
-               }
-           }
-       }
-   }
-}
```

PHP / PHP_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -1,8 +1,10 @@
+CxList binary_expr = Find_BinaryExpr();
 CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null");
+CxList NULL = Find_NullLiteral();
 CxList psw = Find_Passwords();
 CxList strings = Find_Strings();
 CxList methods = Find_Methods();
+CxList parameters = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

 CxList strLiterals = All.NewCxList();
 strLiterals.Add(strings);
@@ -16,7 +18,7 @@
 allPsw.Add(psw, Find_Password_Strings());

 //     (when the hardcoded string includes a space or dot we believe it is not a password string)
-strLiterals -= strLiterals.FindByNames("* *", "*.*", "*/*", "*\\*");
```

```
+strLiterals -= strLiterals.FindByNames("*\t*", "* *", "*.*", "*/*", "*\\*");
 CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);

 // PSW in left side of assignment
@@ -47,7 +49,7 @@
 assignPassword -= notHdPass;

 // Find password in an initialization operation
-CxList eq = Find_BinaryExpr().FindByShortNames(new List<string> { "==","===","!=","!==","|","&","^", "??" });
+CxList eq = binary_expr.FindByShortNames("==","===","??");
 CxList equalsPassword = psw.GetFathers() * eq;
 equalsPassword = strLiterals.FindByFathers(equalsPassword);

@@ -57,39 +59,55 @@
 CxList definePasswords = Find_Declarators().GetByAncs(constantPwds).GetAssigner();
 definePasswords = definePasswords * strLiterals;
 // Add the 'define' first parameters
-CxList defineCmds = methods.FindByShortName("define");
+CxList defineCmds = methods.FindByShortName("define", false);
 CxList firstParams = All.GetParameters(defineCmds, 0);
 CxList secondParams = All.GetParameters(defineCmds, 1);
 CxList hPassInstrDef = All.FindByParameters(firstParams * allPsw).FindByParameters(secondParams * strLiterals);
 definePasswords.Add(strLiterals.GetParameters(hPassInstrDef, 1));

-//Find password in StrComp method  ==> strcmp("hello",pwd,0)
-CxList strcmp = methods.FindByShortName("strcmp");
-CxList strcmpParam1 = All.GetParameters(strcmp, 0);
-CxList strcmpParam2 = All.GetParameters(strcmp, 1);
-CxList hPassInstrcmp = All.FindByParameters(strcmpParam1 * psw).FindByParameters(strcmpParam2 * strLiterals);
-hPassInstrcmp.Add(All.FindByParameters(strcmpParam2 * psw).FindByParameters(strcmpParam1 * strLiterals));
-hPassInstrcmp = psw.GetParameters(hPassInstrcmp);
-hPassInstrcmp = hPassInstrcmp.FindByType<UnknownReference>();
+//Find password in StrComp method  ==> strcmp("hello",pwd) == 0
+CxList strcmp = methods.FindByShortName("strcmp", false);
+CxList strcmpParam1 = parameters.GetParameters(strcmp, 0);
+CxList strcmpParam2 = parameters.GetParameters(strcmp, 1);
+CxList hPassInstrcmp = strcmp.FindByParameters(strcmpParam1 * psw).FindByParameters(strcmpParam2 * strLiterals);
+hPassInstrcmp.Add(strcmp.FindByParameters(strcmpParam2 * psw).FindByParameters(strcmpParam1 * strLiterals));
+// They need to be compared with 0
+CxList strcmp_father = hPassInstrcmp.GetFathers().FindByType<BinaryExpr>();
+CxList zeros = Find_IntegerLiterals().FindByShortName("0");
+// equality with zero
+hPassInstrcmp = strLiterals.GetByAncs(strcmp_father.FindByShortNames("==","===") * zeros.GetFathers());
+
+// Similar to strcmp, but for <=>
+CxList spaceship = binary_expr.FindByShortName("<=>");
+CxList spaceshipPassword = psw.GetFathers() * spaceship;
+
```

```
+CxList spaceshipFathers = spaceshipPassword.GetFathers();
+spaceshipFathers.Add(spaceshipFathers.FindByType<SubExpr>().GetFathers()); // two levels given subexpr element
+spaceshipPassword = strLiterals.GetByAncs(
+   (zeros.GetFathers() * spaceshipFathers).FindByType<BinaryExpr>().FindByShortName("=="));
+
+// hash_equals
+CxList hash_equals = methods.FindByShortName("hash_equals", false);
+CxList hash_equalsParam1 = parameters.GetParameters(hash_equals, 0);
+CxList hash_equalsParam2 = parameters.GetParameters(hash_equals, 1);
+CxList hash_equals_pwd = hash_equals
+   .FindByParameters(hash_equalsParam1 * psw).FindByParameters(hash_equalsParam2 * strLiterals);
+hash_equals_pwd.Add(hash_equals.FindByParameters(hash_equalsParam2 * psw).FindByParameters(hash_equalsParam1 * strLiterals));
+hash_equals_pwd = strLiterals.GetByAncs(hash_equals_pwd);

 // Find hardcoded password as the first parameter of the encodePassword method (Symfony)
 CxList encodePasswordMemberAccess = All.FindByMemberAccess("*.encodePassword", false);
 CxList encodePasswordLiteral = strings.GetParameters(encodePasswordMemberAccess, 0);
 encodePasswordLiteral -= emptyString;

-result.Add(equalsPassword, assignPassword, hPassInstrcmp, definePasswords, encodePasswordLiteral);
+//addPHPArrays
+CxList arr_psw_declarators = psw.GetAncOfType<Declarator>().GetByAncs(Find_AssociativeArrayExpr());
+arr_psw_declarators = arr_psw_declarators * strLiterals.GetFathers();
+// remove if key equals to value (e.g. password => 'password')
+arr_psw_declarators -= arr_psw_declarators.FilterByDomProperty<Declarator>(decl=>
+   decl.ShortName.Equals(decl.InitExpression.ShortName.Substring(1, decl.InitExpression.ShortName.Length - 2)));
+arr_psw_declarators = arr_psw_declarators.CxSelectDomProperty<Declarator>(
+   decl => decl.InitExpression);

-//addPHPArrays
-CxList arr = All.FindByShortName("array") * methods;
-
-CxList inArray = arr.FindByParameters(allPsw);
-CxList paramOfPass = allPsw.GetFathers().FindByType<Param>();
-foreach(CxList pop in paramOfPass)
-{
-   int index = pop.GetIndexOfParameter();
-   if(index % 2 == 0)
-   {
-       CxList array = inArray.FindByParameters(pop);
-       result.Add(strLiterals.GetParameters(array, ++index));
-   }
-}
+result.Add(spaceshipPassword, hash_equals_pwd, equalsPassword, arr_psw_declarators,
+          assignPassword, hPassInstrcmp, definePasswords, encodePasswordLiteral);
```

PHP / PHP_Medium_Threat / CSRF

Code changes

```
---
+++
@@ -1,29 +1,31 @@
-CxList requests = Find_Interactive_Inputs();//Get Requests
+CxList requests = Find_Interactive_Inputs();
+CxList fileWrites = Find_File_Write_Outputs();
+CxList sinks = All.NewCxList(
+   Find_CSRF_DB_In(),
+   fileWrites - fileWrites.GetByAncs(Find_Log_Outputs())
+);

-CxList php_csrf_sanitize = Find_CSRF_Sanitize();//Get XSRF sanitizers
-requests -= requests.GetByAncs(php_csrf_sanitize);//Remove request that are ancestors of xsrf sanitizers
+CxList rawSql = Find_Raw_SQL_Selects();
+CxList dbSelects = Find_DB_Out()
+   .InfluencedBy(rawSql)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
+   .GetLastNodesInPath();
+CxList sanitizers = All.NewCxList();
+sanitizers.Add(
+   // Reads are not CSRF
+   dbSelects,
+   rawSql,
+   // Validation through methods
+   Find_CSRF_Safe_Methods(),
+   // Find validation of Synchronization Token
+   Find_CSRF_Sync_Token()
+);

-CxList strings = Find_Strings();//Get Strings
-CxList binary = Find_BinaryExpr();//Get Binary expressions
-
-
-CxList selectCommand = strings.FindByName("select *", StringComparison.OrdinalIgnoreCase);//Find select strings
-selectCommand.Add(Find_Methods().FindByName("select *", StringComparison.OrdinalIgnoreCase));//Find select methods
-
-CxList sanitizers = selectCommand.Clone();
-
-CxList double_qs = All.FindByShortName("$_DoubleQuotedString", false);//Find All DoubleQuotedStrings
-//Find All DoubleQuotedStrings
-double_qs = double_qs.FindByParameters((All.GetParameters(double_qs).FindByType<StringLiteral>() * selectCommand));
- //Get all select commands that are parameters
-CxList select1 = All.FindByFathers(selectCommand.GetAncOfType<Param>());
-//Get all select commands that are assigment expressions
-CxList select2 = All.FindByFathers((selectCommand - select1).GetAncOfType<AssignExpr>());
-//Add binary that contain select commands
-binary = selectCommand.GetAncOfType<BinaryExpr>() * binary;
-
```

```
-//Add Sanitizers
-sanitizers.Add(binary, select1, select2, double_qs);
-
-CxList db = Find_CSRF_DB_In();
-result = db.InfluencedByAndNotSanitized(requests, sanitizers);
+result.Add(
+   // Find nodes that exist on both lists (but are not sanitizers)
+   (sinks * requests) - sanitizers,
+   // Calculate Flow
+   sinks.InfluencedByAndNotSanitized(requests, sanitizers)
+       .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
+       .ReduceFlowByPragma()
+);
```

PHP / PHP_Medium_Threat / DoS_by_Sleep

Code changes

```
---
+++
@@ -1,14 +1,21 @@
-CxList Inputs = Find_Interactive_Inputs();
-CxList methods = Find_Methods();
-CxList sleepMethods = methods.FindByShortName("sleep", false) +
-   methods.FindByShortName("time_sleep_until", false) +
-   methods.FindByShortName("usleep", false) +
-   methods.FindByShortName("time_nanosleep", false);
+// The largest integer supported in PHP:
+// - int(2147483647) in 32 bit systems
+// - int(9223372036854775807) in 64 bit systems
+long PHP_INT_MAX = 9223372036854775807;


-CxList sleepParameters = All.GetParameters(sleepMethods) - All.FindByType(typeof(Param));
-CxList integersAbstractValues = sleepParameters.FindByAbstractValue(abstractValue => abstractValue is IntegerIntervalAbstractValue);
-IAbstractValue intervalOneToMil = new IntegerIntervalAbstractValue(1, 1000000);
-CxList validIntervals = integersAbstractValues.FindByAbstractValue(abstractValue => abstractValue.IncludedIn(intervalOneToMil));
-CxList vulnerableSleepInvokes = sleepMethods.FindByParameters(sleepParameters - validIntervals);
+// get sleep methods
+string[] sleepMethodsStrg = new[]{ "sleep", "usleep", "time_nanosleep", "time_sleep_until" };
+CxList sleepMethods = Find_Methods().FindByShortNames(sleepMethodsStrg, false);


-result = vulnerableSleepInvokes.DataInfluencedBy(Inputs);
+// get sleep parameters
+CxList parameters = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
+CxList sleepParams = parameters.GetParameters(sleepMethods);
+
+// remove valid sleep methods => [-infinity, PHP_INT_MAX]
+CxList sleepAbsVals = sleepParams.FindByAbstractValue(absValue => absValue is IntegerIntervalAbstractValue);
+IAbstractValue validIntegerAbsVals = new IntegerIntervalAbstractValue(null, PHP_INT_MAX);
+sleepParams -= sleepAbsVals.FindByAbstractValue(absValue => absValue.IncludedIn(validIntegerAbsVals));
```

```
+
+// vulnerable sleep methods
+CxList vulnerableSleepMethods = sleepMethods.FindByParameters(sleepParams);
+result = vulnerableSleepMethods.InfluencedByAndNotSanitized(Find_Interactive_Inputs(), Find_Conditions());
```

PHP / PHP_Medium_Threat / Header_Injection

Code changes

```
---
+++
@@ -1,38 +1,37 @@
+// Generals
 CxList methods = Find_Methods();
 CxList inputs = Find_Interactive_Inputs();
+CxList sanitize = Find_Header_Injection_Sanitize();
+sanitize.Add(Find_General_Sanitize());
+CxList unkRefs = Find_UnknownReferences();

-CxList firstParamMethods = methods.FindByShortNames(new List<String>()
-   { "fsockopen", "get_headers", "imap_open", "ldap_connect", "msession_connect",
-       "pfsockopen", "stream_socket_client", "stream_socket_server" });
-CxList firstParam = All.GetParameters(firstParamMethods, 0);
+// Create lists to be used in query
+CxList taintedParamRefs = All.NewCxList();

-CxList secondAndThirdParamMethods =
-   methods.FindByShortNames(new List<String>() { "curl_setopt", "ftp_chmod", "ftp_put", "ftp_nb_get", "ftp_get" });
+// Second Parameter
+string[] secondParam = new [] { "curl_setopt", "curl_setopt_array", "cyrus_query" };

-CxList secondParamMethods =
-   methods.FindByShortNames(new List<String>()
-   { "curl_setopt_array", "cyrus_query", "socket_bind", "socket_connect", "socket_send", "socket_write",
-       "ftp_exec", "ftp_delete", "ftp_nlist", "ftp_nb_put"});
-secondParamMethods.Add(secondAndThirdParamMethods);
-
-CxList secondParam = All.GetParameters(secondParamMethods, 1);
+// Third Parameter
+string[] thirdParam = new [] { "curl_setopt" };

-CxList thirdParamMethods =
-   methods.FindByShortNames(new List<String>(){ "error_log", "ftp_fget", "ftp_nb_fget" });
-thirdParamMethods.Add(secondAndThirdParamMethods);
-
-CxList thirdParam = All.GetParameters(thirdParamMethods, 2);
+// Fourth Parameter
+string[] fourthParam = new [] { "mail", "mb_send_mail" };

-CxList fourthParamMethods = methods.FindByShortNames(new List<String>(){ "mail", "mb_send_mail" });
```

```
-CxList fourthParam = All.GetParameters(fourthParamMethods, 3);
+// Function to filter for method parameters
+Func < string [], int, CxList > filterParameters = null;

-CxList allParamMethods = methods.FindByShortName("session_register");
-CxList allParam = All.GetParameters(allParamMethods);
+filterParameters = (methodNames, paramIndex) =>
+{
+   CxList headerMethods = methods.FindByShortNames(methodNames, false);
+   return unkRefs.GetParameters(headerMethods, paramIndex);
+};

-CxList tainted_params =  All.NewCxList();
-tainted_params.Add(firstParam, secondParam, thirdParam, fourthParam, allParam);
+taintedParamRefs.Add(
+   // Get second parameters from methods and add to tainted list
+   filterParameters(secondParam, 1),
+   // Get third parameters from methods and add to tainted list
+   filterParameters(thirdParam, 2),
+   // Get fourth parameters from methods and add to tainted list
+   filterParameters(fourthParam, 3));

-CxList sanitize = Find_Header_Injection_Sanitize();
-
-result = tainted_params.InfluencedByAndNotSanitized(inputs, sanitize);
-result.Add(tainted_params * inputs);
+result.Add(taintedParamRefs.InfluencedByAndNotSanitized(inputs, sanitize));
```

PHP / PHP_Medium_Threat / Improper_Restriction_of_Stored_XXE_Ref

Code changes

```
---
+++
@@ -1,26 +1,8 @@
-/// <summary>
-/// This query searches for XML files created from a DB read
-/// Sources    -> DB reads and DB outputs
-/// Sincs  -> XML files
-/// </summary>
+CxList inputs = All.NewCxList(Find_Read(), Find_DB_Out());
+CxList sinks = Improper_Restriction_of_XXE();

-CxList inputs = Find_Read();
-inputs.Add(Find_DB_Out());
-
-CxList xxe_results = Find_XXE_SimpleXml();
-xxe_results.Add(Find_XXE_XmlDOM());
-xxe_results.Add(Find_XXE_XMLReader());
-
```

```
-CxList xxe_results_vulnerable = xxe_results - Find_XXE_Sanitizers();
-
-CxList tempResult = xxe_results_vulnerable.DataInfluencedBy(inputs);
-
-//Extreme case where the source is also an XML file
-foreach(CxList res in tempResult.GetCxListByPath()){
-   CxList start = res.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly);
-   CxList end = res.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-
-   if(start.GetParameters(end).Count == 0) {
-       result.Add(res);
-   }
-}
+result.Add(
+   sinks * inputs,
+   sinks.InfluencedBy(inputs),
+   Find_XXE_XMLReader_Objects(inputs)
+);
```

PHP / PHP_Medium_Threat / Improper_Restriction_of_XXE_Ref

Code changes

```
---
+++
@@ -1,5 +1,8 @@
 CxList inputs = Find_Interactive_Inputs();
-CxList xxe_results = Find_XXE_SimpleXml() + Find_XXE_XmlDOM() + Find_XXE_XMLReader();
-CxList xxe_results_vulnerable = xxe_results - Find_XXE_Sanitizers();
+CxList sinks = Improper_Restriction_of_XXE();

-result = xxe_results_vulnerable.DataInfluencedBy(inputs);
+result.Add(
+   sinks * inputs,
+   sinks.InfluencedBy(inputs),
+   Find_XXE_XMLReader_Objects(inputs)
+);
```

PHP / PHP_Medium_Threat / Missing_HSTS_Header

Code changes

```
---
+++
@@ -1,26 +1,41 @@
 bool show_only_one_HSTS_result = true;
-CxList stringLiterals = Find_Strings();
+
 CxList globalConfig = Find_HSTS_Configuration_In_Code();
+CxList strLiterals = Find_Strings();
+
 if(globalConfig.Count > 0)
```

```
 {
    result = Check_HSTS_Configuration();
 }
 else
 {
-   CxList headerChanges = Find_Methods().FindByShortNames(new List<string>(){"header","setHeader"});
-   CxList paramsOfHeader = stringLiterals.GetParameters(headerChanges);
+   CxList headerChanges = Find_Methods().FindByShortNames(new[]{"header","setHeader","addHeader","addHeaderLine"}, false);
+   CxList paramsOfHeader = strLiterals.GetParameters(headerChanges);
    CxList hstsHeaders = paramsOfHeader.FindByShortName("*Strict-Transport-Security*", false);
-   if(hstsHeaders.Count == 0)
+
+   // check if second parameter is relevant
+   CxList hsts_keywords = hstsHeaders.FindByShortName("Strict-Transport-Security", false);
+   hstsHeaders.Add(
+       strLiterals.GetParameters(hsts_keywords.GetAncOfType<MethodInvokeExpr>(), 1)
+   );
+   hstsHeaders -= hsts_keywords;
+
+   if (hstsHeaders.Count == 0)
    {
        // If a header is explicitly set, we return the first node related to the header.
        // If not, we return the first node in the project
        // This information is more valuable to the user when analyzing the result
-       CSharpGraph res = headerChanges.GetFirstGraph();
-       if(res == null)
+       if (headerChanges.Count > 0)
        {
-           res = All.GetFirstGraph();
+           result.Add(headerChanges.First());
        }
-       result.Add(res.NodeId, res);
+       else
+       {
+           CxList statements = Find_Statements().FilterPlugins() - Find_Import();
+           // If there is no dom, no result can be returned
+           if (statements.Count > 0)
+               result.Add(statements.First());
+       }
    }
    else
    {
@@ -28,7 +43,7 @@
        {
            if(Validate_HSTS_Header(header).Count > 0)
            {
-               result.Add(headerChanges.InfluencedBy(header));
+               result.Add(header);
```

```
                if(show_only_one_HSTS_result)
                {
                    break;
```

PHP / PHP_Medium_Threat / Open_Redirect

Code changes

```
---
+++
@@ -1,9 +1,45 @@
-CxList redirectFunctions = Find_Methods().FindByShortName("header", false);
-CxList redirectLocationStrings = Find_Strings().FindByShortName("*location*", false);
-
-redirectFunctions = redirectFunctions.InfluencedBy(redirectLocationStrings);
-CxList inputs = Find_Interactive_Inputs();
-
+// Generals
+CxList methods = Find_Methods();
+CxList strings = Find_Strings();
 CxList sanitize = Find_General_Sanitize();


-result = redirectFunctions.InfluencedByAndNotSanitized(inputs, sanitize);
+if(All.useConsoleInputs)
+{
+   // The following keys are safe for this query: REMOTE_ADDR, REMOTE_HOST
+   CxList safeStrings = strings.FindByShortNames(new[] {"REMOTE_ADDR", "REMOTE_HOST"});
+   sanitize.Add(safeStrings.GetAncOfType<IndexerRef>().FindByShortName("_SERVER"));
+}
+
+// Find for all header method call
+CxList redirectFunctions = methods.FindByShortName("header", false);
+
+// Find for redirect location strings
+CxList redirectLocationStrings = strings.FindByAbstractValue(absValue
+   => absValue is StringAbstractValue strAbsVal
+   && strAbsVal.Content.ToLower().Trim().Contains("location:"));
+
+//Get only methods from location strings
+redirectFunctions = redirectFunctions.FindByParameters(redirectLocationStrings);
+
+// Find for all allowed hosts
+CxList allowedHosts = strings.FindByAbstractValue(absValue
+   => {
+       if (absValue is StringAbstractValue strAbsVal)
+       {
+           string str = strAbsVal.Content.ToLower().Trim();
+           return str.Contains("https://") || str.Contains("http://");
+       }
+       else
```

```
+          return false;
+   });
+
+// Add allowed hosts and header influenced by str_starts_with to the sanitize list
+sanitize.Add(
+   allowedHosts.GetByAncs(redirectFunctions).GetAncOfType<MethodInvokeExpr>(),
+   methods.FindByShortName("str_starts_with", false));
+
+// Add allowed hosts and header influenced by str_starts_with/in_array to the sanitize list
+sanitize.Add(Find_UnknownReference().GetSanitizerByMethodInCondition(sanitize).GetAncOfType<MethodInvokeExpr>());
+
+// Get all redirect functions influenced by inputs/sanitizers  and add to result
+result = redirectFunctions.InfluencedByAndNotSanitized(Find_Interactive_Inputs(), sanitize);
```

PHP / PHP_Medium_Threat / Parameter_Tampering

Code changes

```
---
+++
@@ -1,15 +1,26 @@
-CxList input = Find_Interactive_Inputs();
-CxList db = Find_DB_In();
-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);
-
-db  = db.DataInfluencedBy(Select).DataInfluencedBy(Where);
-db -= db.DataInfluencedBy(And);
-
+////////////////////////////////////////////////////////////////
+//                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", "Update" and "Delete"
+//     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 inputs = Find_Interactive_Inputs();
+CxList db = Find_DB_For_Parameter_Tampering();
 CxList sanitize = Find_Parameter_Tampering_Sanitize();


-result = db.InfluencedByAndNotSanitized(input, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+// The use of conditional branching through if conditions
+// can be considered a sanitizer if the user input
```

```
+// is being used in an authentication check.
+CxList idsInBinary = sanitize.FindByShortNames(new[]{ "_SESSION", "*id*"}, false);
+idsInBinary = idsInBinary.GetAncOfType<BinaryExpr>();
+CxList binayExprs = idsInBinary.InfluencedBy(inputs).GetLastNodesInPath();
+CxList conditions = binayExprs.GetAncOfType<IfStmt>();
+conditions.Add(binayExprs.GetAncOfType<IterationStmt>());
+sanitize.Add(db.GetByAncs(conditions));
+
+result = db.InfluencedByAndNotSanitized(inputs, sanitize)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

PHP / PHP_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -6,31 +6,18 @@
 CxList find_db_out = Find_DB_Out();
 CxList find_inputs = Find_Inputs();

-CxList literals = All.NewCxList();
-literals.Add(strings, Find_IntegerLiterals(), Find_RealLiterals(), Find_NullLiteral());
+CxList literals = All.NewCxList(strings, Find_IntegerLiterals(), Find_RealLiterals(), Find_NullLiteral());

-// Find names that are suspected to be personal info, eg. String PASSWORD, Integer SSN, and remove string literals, such as x="password"
-CxList personal_info = Find_Personal_Info() - strings;
-
-// 1)exclude variables that are all uppercase - usually describes the pattern of the data, such as PASSWORDPATTERN, PASSORDTYPE...
-CxList upperCase = All.NewCxList();
-foreach (CxList res in personal_info)
-{
-   string name = res.GetName();
-   if (name.ToUpper().Equals(name))
-   {
-       upperCase.Add(res);
-   }
-}
-personal_info -= upperCase;
+// Find names that are suspected to be personal info and remove string literals, such as x="password"
+CxList personal_info = Find_Personal_Info();
+personal_info.Add(Find_Passwords());
+personal_info -= strings;

 // 2) exclude constants that are assigned a literal
-// * Remark: in all VB languages (including ASP), Ruby, PL/SQL, JavaScript and Perl, constants MUST be assigned in their declaration line
 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 null, string or integer literals
 CxList ConstAssignLiteral = literals.FindByFathers(allConstRef.FindByType<Declarator>());

@@ -46,8 +33,7 @@
 // remove from personal_info all references that were removed above
 personal_info -= (allConstRefOrigin - allConstRef);

-CxList inputs = All.NewCxList();
-inputs.Add(find_inputs, find_db_out);
+CxList inputs = All.NewCxList(find_inputs, find_db_out);

 personal_info = personal_info.DataInfluencedBy(inputs).GetLastNodesInPath();
 personal_info.Add(personal_info * inputs);
@@ -75,30 +61,22 @@
 // Define outputs
 CxList outputs = Find_Outputs();
 outputs.Add(exceptions, exceptionsCtorsWithSuper);
-outputs -= All.GetParameters(methods.FindByShortName("fwrite"), 0);
+outputs -= outputs.GetParameters(methods.FindByShortName("fwrite"), 0);

 // Define sanitize
 CxList sanitize = Find_DB_In(); // in some languages is called Find_DB, Find_DB_In, Find_DB_Input
-CxList encrypt = All.FindByShortName("*crypt*", false); // crypt is a PHP function used to encrypt strings, and all variables labelled crypt(ed) are considered safe, as well as DBMS_CRYPTO as output
-encrypt.Add(Find_Encrypt());
+
+//get_debug_type and get_class are sanitized methods in PHP that retrive irrelevant information for privacy violation
+sanitize.Add(Find_Hashing_Sanitize(), Find_Encryption_Sanitize(), Find_Kohana_Encrypt(),
+   methods.FindByShortNames(new []{"get_debug_type", "get_class"}, false));

-encrypt -= encrypt.FindByShortNames(new List<String>{ "*decrypt*", "*unencrypt*" });
-encrypt.Add(All.FindByShortName("*CipherOutputStream*", false));   // CipherOutputStream is a Java class used to encrypt output streams
+sanitize -= sanitize.FindByShortNames(new []{ "*decrypt*", "*unencrypt*" }, false);

-CxList encoded = All.FindByShortName("*Encode*", false); // all variables labelled encode(ed) are considered safe
-encoded.Add(Find_Encode());
-encoded -= encoded.FindByShortNames(new List<String>{ "*UnEncode*", "*Decode*",
-       "*URLEncode*", // URLEncode method is a part of .NET and is not a sanitizer
-       "*HTMLEncode*", // HTMLEncode method is a part of .NET and is not a sanitizer
-       "*EncodeHTML*" }, false);
-
-//get_debug_type and get_class are sanitized methods in PHP that retrive irrelevant information for privacy violation
-sanitize.Add(encrypt, encoded, methods.FindByShortNames("get_debug_type", "get_class"));

 // split personal_info into variables and constants
 CxList variableRef = personal_info - allConstRef;
```

```
-// find declarators of constants and variables so they can be removed - declarators are not a part of the flow from input to output
+// find declarators of constants and variables so they can be removed
 // 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 = personal_info.FindByType<Declarator>();
@@ -108,7 +86,8 @@
 allConstRef -= declarator;

 // find all constants that are assigned from an input (directly or indirectly) and are influencing an output
-CxList ConstInfuelncedByInput = outputs.InfluencedByAndNotSanitized(allConstRef, sanitize).InfluencedByAndNotSanitized(find_inputs, sanitize);
+CxList ConstInfuelncedByInput = outputs.InfluencedByAndNotSanitized(allConstRef, sanitize)
+   .InfluencedByAndNotSanitized(find_inputs, sanitize);

 //remove variables that were influenced by sanitize
 variableRef -= variableRef.InfluencedByAndNotSanitized(sanitize, find_db_out).GetLastNodesInPath();
```

PHP / PHP_Medium_Threat / Session_Fixation

Code changes

```
---
+++
@@ -1,5 +1,30 @@
-CxList sessionCreation = Find_Session_Create();
-if(sessionCreation.Count > 0 && Find_Session_Fixation_Sanitize().Count == 0)
+CxList sessionCreation = Find_Session_Create().GetAncOfType<ExprStmt>();
+
+//Check if session start exists
+if(sessionCreation.Count > 0)
 {
-   result = sessionCreation;
+   CxList sanitizers = Find_Session_Fixation_Sanitize();
+
+   //Get all possible unsecure methods
+   CxList sinkMethods = Find_MethodDecls().FindByShortNames(new []{ "*action*", "*login*", "*auth*",
+       "*validate*", "*handle*", "*password*"}, false);
+
+   //Remove sanitized methods and discard show functions
+   sinkMethods -= sinkMethods.FindByShortName("*show*", false);
+   sinkMethods -= sanitizers.GetAncOfType<MethodDecl>();
+
+   //Get all invokes from possible unsecure methods
+   CxList sinkInvokes = Find_Methods().FindAllReferences(sinkMethods).GetAncOfType<ExprStmt>();
+
+   //Remove sanitized invokes
+   sinkInvokes -= sinkInvokes.FindInScope(sessionCreation, sanitizers);
+
+   //Concatenate session start with unsanitized invokes
+   foreach(CxList session in sessionCreation)
+   {
```

```
+       CxList sessionStmts = session.GetFathers();
+       CxList invokeScope = sinkInvokes.GetByAncs(sessionStmts);
+
+       result.Add(session.ConcatenateAllPaths(invokeScope));
+   }
 }
```

PHP / PHP_Medium_Threat / SSL_Verification_Bypass

Code changes

```
---
+++
@@ -1,27 +1,30 @@
-/*checks if verify_peer of ssl is set to false*/
-CxList falseIndicator = All.FindByShortName("verify_peer").FindByType(typeof(StringLiteral));
-//CxList methods = Find_Methods();
-CxList arrayMethod = All.FindByParameters(falseIndicator).FindByShortName("array");
-CxList boolean = All.FindByShortName("false").FindByType(typeof(BooleanLiteral));
-CxList unknownRef = All.FindByType(typeof(UnknownReference));
+//General
+CxList unkRefs = Find_UnknownReferences();

-CxList potentialBool = All.NewCxList();
-CxList indicatorCounter = All.NewCxList();
-CxList unkAndBool = unknownRef + boolean;
+// Find for all false boolean literal
+CxList falseBooleans = Find_BooleanLiteral().FindByShortName("false", false);
+falseBooleans.Add(unkRefs.FindByAbstractValue(absValue => absValue is FalseAbstractValue));

-foreach(CxList array in arrayMethod)
-{
-   int i;
-   for(i = 0; i < 30; i++)
-   {
-       indicatorCounter = falseIndicator.GetParameters(array, i);
-       if(indicatorCounter.Count > 0)
-       {
-           break;
-       }
-   }
+// Checks if verify_peer, verify and verify_peer_name of ssl is set to false and add to result
+CxList falseIndicator = Find_FieldDecls().FindByShortNames(new [] {"verify", "verify_peer", "verify_peer_name"}, false);
+falseIndicator = falseBooleans.GetByAncs(falseIndicator).GetAncOfType<FieldDecl>();
+result.Add(falseIndicator);

-   potentialBool.Add(unkAndBool.GetParameters(array, i + 1));
-}
-result.Add(potentialBool * boolean);
-result.Add(potentialBool.DataInfluencedBy(boolean));
```

```
+// Find for all false indicators used in the code and add to result
+result.Add(Find_Strings().FindByShortNames(falseIndicator.CxSelectElementValues<FieldDecl, string>(x => x.ShortName), false));
+
+// Find for cURL specific method call
+CxList cURLMethods = Find_Methods().FindByShortNames(new [] { "curl_setopt", "curl_setopt_array" }, false);
+
+// Find for all SSL references and get the Param type
+CxList parameterValues = Find_Params().CxSelectDomProperty<Param>(p => p.Value);
+CxList cURLParams = parameterValues.FindByShortNames(new [] {
+       "CURLOPT_SSL_ENABLE_ALPN", "CURLOPT_SSL_ENABLE_NPN", "CURLOPT_SSL_VERIFYPEER",
+       "CURLOPT_SSL_VERIFYSTATUS", "CURLOPT_PROXY_SSL_VERIFYPEER" }, false);
+
+// Find for all cURL methods that have false as a third parameter
+CxList falseParamcURLMethods = falseBooleans.GetParameters(cURLMethods, 2);
+falseParamcURLMethods = falseParamcURLMethods.GetAncOfType<MethodInvokeExpr>();
+
+// Get second parameter from cURL method that have false on third parameter and add to result
+result.Add(cURLParams.GetParameters(falseParamcURLMethods, 1));
```

PHP / PHP_Medium_Threat / Stored_Code_Injection

Code changes

```
---
+++
@@ -1,28 +1,4 @@
-CxList methods = Find_Methods();
-CxList dynamic_method_invoke = methods.FindByShortNames(new List<String>()
-   { "$_Function", "call_user_func*", "forward_static_call*", "create_function",
-       "register_shutdown_function", "register_tick_function", "eval" });
+CxList inputs = Find_DB_Out();
+inputs.Add(Find_Read());


-CxList dynamic_variable_access = methods.FindByShortName("$_Variable");
-
-CxList reflectionInvoke = All.FindByShortName("invoke*").GetTargetOfMembers().DataInfluencedBy(All.FindByShortName("*Reflection*"));
-
-dynamic_method_invoke.Add(reflectionInvoke);
-
-CxList db = Find_DB_Out() + Find_Read();
-
-CxList sanitize = Find_Code_Injection_Sanitize();
-result = db.InfluencingOnAndNotSanitized(dynamic_method_invoke + dynamic_variable_access, sanitize);
-
-CxList firstParam = dynamic_method_invoke;
-firstParam = All.GetParameters(firstParam, 0);
-
-CxList arrays = methods.FindByShortName("array");
-
-CxList arraysSecondParam = All.GetParameters(arrays, 1);
```

```
-CxList arraysThirdParam = All.GetParameters(arrays, 2);
-CxList relevantArrays = arraysSecondParam.GetAncOfType(typeof(MethodInvokeExpr)) - arraysThirdParam.GetAncOfType(typeof(MethodInvokeExpr));
-CxList arraysFirstParam = All.GetParameters(relevantArrays, 0);
-
-result.Add(db.InfluencingOnAndNotSanitized(firstParam, sanitize + arraysFirstParam));
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result = Code_Injection_Sinks(inputs);
```

PHP / PHP_Medium_Threat / Stored_LDAP_Injection

Code changes

```
---
+++
@@ -1,10 +1,9 @@
-CxList methods = Find_Methods();
-CxList db = Find_DB_Out() + Find_Read();
+// Generals
+CxList db = Find_DB_Out();
+db.Add(Find_Read());
+CxList sanitized = Find_General_Sanitize();
+sanitized.Add(Find_LDAP_Replace());

-CxList sanitized = Find_General_Sanitize() + Find_LDAP_Replace();
-
-CxList ldap_find_methods = methods.FindByShortNames(new List<string>{ "ldap_list", "ldap_read", "ldap_search" });
-CxList filter_params = All.FindByFathers(All.GetParameters(ldap_find_methods, 2));
-
-result = filter_params.InfluencedByAndNotSanitized(db, sanitized);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+// Filter parameters from LDAP methods influenced by db inputs/sanitizers and add to result
+CxList filteredParams = Find_LDAP_Methods().InfluencedByAndNotSanitized(db, sanitized);
+result = filteredParams.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

PHP / PHP_Medium_Threat / Use_of_Hard_coded_Cryptographic_Key

Code changes

```
---
+++
@@ -1,46 +1,19 @@
-CxList emptyString = Find_Empty_Strings();
-CxList NULL = All.FindByName("null");
-CxList keys = All.FindByShortName("*ENC*", false).FindByShortName("*KEY*", false);
+// hardcoded strings
+CxList strings = Find_Strings();

-CxList key_in_lSide = keys.FindByAssignmentSide(CxList.AssignmentSide.Left);
-CxList strLiterals = All.FindByType(typeof(PrimitiveExpr)) - emptyString - NULL;
-CxList lit_in_rSide = strLiterals.FindByAssignmentSide(CxList.AssignmentSide.Right);
+CxList methods = Find_Methods();
+CxList parameters = Find_Param().CxSelectDomProperty<Param>(p => p.Value);
```

```
+string[] thirdParamMethodStrgs = new[]{ "openssl_encrypt", "openssl_decrypt", "openssl_seal", "openssl_open" };
+CxList thirdParamMethod = methods.FindByShortNames(thirdParamMethodStrgs,  false);
+string[] secondParaMethodStrgs = new[]{ "mcrypt_encrypt", "mcrypt_decrypt" };
+CxList secondParamMethod = methods.FindByShortNames(secondParaMethodStrgs, false);

-result = key_in_lSide.FindByFathers(key_in_lSide.GetFathers() * lit_in_rSide.GetFathers());
+CxList keyParam = All.NewCxList(
+   parameters.GetParameters(thirdParamMethod, 2),
+   parameters.GetParameters(secondParamMethod, 1));

-CxList key_in_Field = All.GetByAncs(All.FindByType(typeof(FieldDecl)) + All.FindByType(typeof(ConstantDecl))) * keys;
-
-CxList sanitizers = base.Find_Same_Value_Sanitizers_Exclude_Sinks_and_Sources(key_in_Field + strLiterals);
-// Remove from the sanitizers all string methods that alter the string in a predictable way.
-string[] stringMethods = new string[]
-{
-   "quotemeta",
-   "rtrim",
-   "str_ireplace",
-   "str_pad",
-   "str_repeat",
-   "str_replace",
-   "str_rot13",
-   "str_split",
-   "strchr",
-   "strip_tags",
-   "stripcslashes",
-   "stripslashes",
-   "trim",
-   "stristr",
-   "strpbrk",
-   "strrchr",
-   "strrev",
-   "strstr",
-   "strtok",
-   "strtolower",
-   "strtoupper",
-   "strtr",
-   "substr",
-   "substr_replace",
-   "ucfirst",
-   "ucwords"
-};
-sanitizers -= Find_Methods().FindByShortNames(stringMethods);
-
-result.Add(key_in_Field.InfluencedByAndNotSanitized(strLiterals, sanitizers));
+CxList sanitizers = All.NewCxList(Find_Hashing_Functions(), Find_Encrypt());
+CxList sinks = All.NewCxList(secondParamMethod, thirdParamMethod);
```

```
+result = sinks.InfluencedByAndNotSanitized(strings, sanitizers)
+   .IntersectWithNodes(keyParam)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Ruby / Ruby_Best_Coding_Practice / Dynamic_Render_Path

Code changes

```
---
+++
@@ -1,14 +1,13 @@
-CxList render = All.FindByShortName("render");
+CxList render = All.FindByShortNames("render");
 render -= render.FindAllReferences(All.FindDefinition(render));
-render -= render.GetTargetOfMembers().GetMembersOfTarget();
+render -= render.GetMembersWithTargets();

-CxList dyna = Find_Methods();
-dyna.Add(Find_UnknownReference());
+CxList dyna = All.NewCxList(Find_Methods(), Find_UnknownReference());

 dyna -= Find_Inputs();
 dyna = dyna.GetByAncs(render);

-CxList sanitize = All.GetByAncs(render).FindByType(typeof(AssignExpr));
+CxList sanitize = All.FindDescendantsOfType<AssignExpr>(render);
 sanitize = All.GetByAncs(sanitize);

 result = dyna.InfluencingOnAndNotSanitized(render, sanitize);
```

Scala / Scala_Medium_Threat / Stored_External_XML_Entities_XXE

Code changes

```
---
+++
@@ -1,20 +1,17 @@
-//Inputs
-CxList inputs = Find_Read_NonDB();
+// Sinks
+CxList xxe = Find_XML_Entities_XXE_Outputs();

-//load* methods are sinks, not inputs.
-CxList excludeInputs = inputs.FindByShortName("load*");
-inputs -= excludeInputs;
-CxList parameterInInputs = All.GetParameters(excludeInputs, 0).FindByType<String>();
-inputs.Add(parameterInInputs);
+// Inputs
+CxList storedInputs = Find_Read_NonDB();

+// load* methods are sinks, not inputs.
+storedInputs -= xxe;
```

```
-//Sanitizers
+// Sanitizers
 CxList sanitizers = Find_XML_Entities_XXE_Sanitizers();

-//Sinks
-CxList xxe = Find_XML_Entities_XXE_Outputs();
 // remove sinks influenced by sanitizers
 xxe -= xxe.GetTargetOfMembers().InfluencedBy(sanitizers).GetMembersOfTarget();
 xxe -= xxe.InfluencedBy(sanitizers);

-result = xxe.InfluencedByAndNotSanitized(inputs, sanitizers);
+result = xxe.InfluencedByAndNotSanitized(storedInputs, sanitizers);
```

Swift / Swift_Low_Visibility / Null_Password

Code changes

```
---
+++
@@ -27,7 +27,6 @@
 result.Add(PasswordInDecl, assignPassword);

 // Handle swift subscripts, e.g. dict[password] = nil
-
 CxList subscriptMethods = Find_Methods().FindByShortName("subscript");
 CxList subscriptMethodsWithNullPasswords = All.NewCxList();
@@ -36,3 +35,8 @@
         subscriptMethodsWithNullPasswords.Add(m);

 result.Add(nullLiterals.GetParameters(subscriptMethodsWithNullPasswords, 1));
+
+// Add to result the indexerRefs (e.g. "postDict[password]") influenced by null
+CxList indexerRefs = Find_IndexerRefs();
+CxList passwordIndRef = indexerRefs.CxSelectElements<IndexerRef>(_ => _.Indices, 0).FindByShortName(psw).GetFathers();
+result.Add(passwordIndRef.InfluencedBy(nullLiterals));
```

Swift / Swift_Medium_Threat / Pasteboard_Leakage

Code changes

```
---
+++
@@ -25,7 +25,9 @@

 // Remove sanitized personal informations
 CxList personalInfo = Remove_Sanitized_Personal_Info(UIfields, unsecureCustomPasteboards);
-
+// Add IndexerRefs with personal info in Indice (to be able to find flow)
+CxList personalInfoIndices = Find_IndexerRefs().CxSelectElements<IndexerRef>(_ => _.Indices, 0) * personalInfo;
+personalInfo.Add(personalInfoIndices.GetFathers());
```

```
 if (personalInfo.Count > 0) // if the application is not entirely secured
 {
    CxList output = Find_Interactive_Outputs_User();
@@ -39,7 +41,7 @@
        ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
    // Add unsecure pasteboards
    result.Add(unsecurePasteboards);
-
+
    // Find UIKit Widgets leakage
    string[] uiKitWidgetsTypes = new string[]{ "UITextField", "UITextView" };
    CxList uiKitWidgtes = personalInfo.FindByTypes(uiKitWidgetsTypes);
```

Swift / Swift_Medium_Threat / ReDoS

Code changes

```
---
+++
@@ -1,6 +1,5 @@
-CxList evilStringInputs = Find_Evil_Strings();
+CxList evilStrings = Find_Evil_Strings();
 CxList inputs = Find_Inputs();
-evilStringInputs.Add(inputs);

 CxList match = Find_Match();
 CxList objectCreations = Find_ObjectCreations();
@@ -30,8 +29,8 @@
    regexPatterns.Add(All.GetByAncs(All.GetParameters(regexIsMatch, 1)));

    // Find regex commands that are influenced by evil strings
-   CxList activeEvilRegexes = (evilStringInputs * regexPatterns);
-   activeEvilRegexes.Add(evilStringInputs.InfluencingOnAndNotSanitized(regexPatterns, sanitize));
+   CxList activeEvilRegexes = (evilStrings * regexPatterns);
+   activeEvilRegexes.Add(evilStrings.InfluencingOnAndNotSanitized(regexPatterns, sanitize));

    CxList activeEvilRegexesLastNode = activeEvilRegexes.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartAndEndNodes);
    CxList activeEvilRegexesMethodInvokes = activeEvilRegexesLastNode.GetAncOfType<MethodInvokeExpr>();
@@ -40,8 +39,17 @@

    // Find relevant matches
    result = match.InfluencedByAndNotSanitized(activeEvilRegexes, sanitize);
-   result.Add(match.InfluencedByAndNotSanitized(All.FindAllReferences(activeEvilRegexes2), sanitize));
-   result.Add(evilStringInputs.InfluencingOnAndNotSanitized(methods.FindByShortName("range"), sanitize));
+   result.Add(match.InfluencedByAndNotSanitized(All.FindAllReferences(activeEvilRegexes2), sanitize));
+   // range method
+   CxList rangeMethods = methods.FindByShortName("range");
+   result.Add(evilStrings.InfluencingOnAndNotSanitized(rangeMethods, sanitize));
+   CxList relevantInputs = inputs.NotInfluencingOn(rangeMethods.GetTargetOfMembers());
+   result.Add(relevantInputs.InfluencingOnAndNotSanitized(rangeMethods, sanitize));
```

```
+   // Regex from user input
+   CxList regexesFromInputs = inputs.InfluencingOn(regex);
+   result.Add(regexesFromInputs.InfluencingOnAndNotSanitized(match, sanitize));
+
+
    // Add predicates which execute a match and are compromised:
    CxList nsPredicateList = objectCreations.FindByShortName("NSPredicate");


@@ -55,7 +63,7 @@
    // Add IOS predicates with compromised patterns
    string[] initMethodNames = {@"regularExpressionWithPattern", @"initWithPattern"};
    CxList initMethods = methods.FindByShortNames(new List<string>(initMethodNames));
-   CxList compromisedInitMethods = initMethods.InfluencedByAndNotSanitized(evilStringInputs, sanitize);
+   CxList compromisedInitMethods = initMethods.InfluencedByAndNotSanitized(evilStrings, sanitize);
    CxList compromisedRegex = regex.InfluencedBy(compromisedInitMethods);
    result.Add(All.FindAllReferences(compromisedRegex).GetMembersOfTarget() * match);
 }
```

PHP / PHP_High_Risk / Deserialization_of_Untrusted_Data

Code changes

```
---
+++
@@ -1 +1,19 @@
-result = Common_High_Risk.Deserialization_of_Untrusted_Data();
+CxList inputs = Find_Deserialization_Inputs();
+CxList deserializers = Find_Unsafe_Deserializers();
+CxList sanitizers = Find_Integers();
+
+//remove all the deserializers in coditioned blocks where the condition is influenced by hash
+CxList allCond = Find_Conditions();
+
+CxList comparison = Find_BinaryExpr().GetByAncs(allCond).FindByShortNames(new []{"==", "!="});
+
+CxList relevantComparison = comparison.InfluencedBy(sanitizers);
+relevantComparison = relevantComparison.GetLastNodesInPath();
+
+CxList relevantIfs = relevantComparison.GetAncOfType<IfStmt>();
+relevantIfs.Add(relevantComparison.GetAncOfType<IterationStmt>());
+
+deserializers -= deserializers.GetByAncs(relevantIfs);
+
+result.Add(inputs.InfluencingOnAndNotSanitized(deserializers, sanitizers)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow));
```

PHP / PHP_High_Risk / Reflected_XSS

Code changes

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

```
@@ -1,10 +1,29 @@
 CxList inputs = Find_Interactive_Inputs();
 CxList outputs = Find_Interactive_Outputs();

-// Delete "header" from the outputs because is not a case of Reflected XSS
-CxList methods = Find_Methods();
-outputs -= methods.FindByShortName("header");
+// `readfile` automatically outputs the contents: it's not a sanitizer
+CxList readfile = Find_Methods().FindByShortName("readfile", false);
+readfile.Add(
+   Find_Params().CxSelectDomProperty<Param>(p => p.Value).GetParameters(readfile, 0)
+   );

-CxList sanitized = Find_XSS_Sanitize();
+CxList sanitized = All.NewCxList(
+   Find_XSS_Sanitize(),
+   Find_File_Access() - readfile
+);

-result = inputs.InfluencingOnAndNotSanitized(outputs, sanitized).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+if(All.useConsoleInputs)
+{
+   sanitized.Add(
+       inputs.FindByShortName("argc"),
+       Find_Strings().FindByShortName("argc").GetAncOfType<IndexerRef>().FindByShortName("_SERVER")
+   );
+}
+
+result.Add(
+   // Duplicated nodes
+   (inputs * outputs) - sanitized,
+   // Flow
+   inputs.InfluencingOnAndNotSanitized(outputs, sanitized)
+       .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
+);
```

PHP / PHP_High_Risk / Stored_XPath_Injection

Code changes

```
---
+++
@@ -1,38 +1,13 @@
-CxList methods = Find_Methods();
-CxList db = Find_DB_Out() + Find_Read();
-CxList sanitized = Find_XPath_Sanitize();
+CxList inputs = All.NewCxList();
+inputs.Add(Find_DB_Out(), Find_Read());

-CxList xpath = methods.FindByShortName("xpath_*",false);
```

```
-CxList xpathEval=xpath.FindByShortName("xpath_eval*",false);
-CxList exptr = methods.FindByShortName("xptr_*", false);
+// Find_DB_Out and Find_Read might return unknown references on binds
+// For those, there is no lazy flow, so we need to perform heuristics:
+// 1. Add all references
+inputs.Add(Find_UnknownReference().FindAllReferences(inputs));
+// 2. Remove any that is sanitized
+CxList sanitizers = Find_XPath_Sanitize();
+CxList sanitized = inputs.InfluencedBy(sanitizers);
+inputs -= inputs.FindAllReferences(sanitized);

-// Methods which get a filter (possibly tainted) in their first patameter.
-CxList XPath_parm1 = methods.FindByMemberAccess("DOMXPath.evaluate");
-XPath_parm1.Add(methods.FindByMemberAccess("DOMXPath.query"));
-XPath_parm1.Add(xpathEval.FindByMemberAccess("XPathContext.xpath_eval_expression"));
-XPath_parm1.Add(xpathEval.FindByMemberAccess("XPathContext.xpath_eval"));
-XPath_parm1.Add(exptr.FindByMemberAccess("XPathContext.xptr_eval"));
-XPath_parm1.Add(xpath.FindByMemberAccess("SimpleXMLElement.xpath"));
-
-CxList memberMethods = xpathEval.FindByShortNames(new List<string>{ "xpath_eval_expression", "xpath_eval" });
-memberMethods.Add(exptr.FindByShortName("xptr_eval"));
-memberMethods.Add(xpath.FindByShortName("xpath"));
-
-XPath_parm1.Add(memberMethods.GetMembersOfTarget());
-
-CxList tainted_parm1 = All.GetParameters(XPath_parm1, 0);
-CxList other_params1 = All.GetParameters(XPath_parm1) - tainted_parm1;
-other_params1.Add(XPath_parm1.GetTargetOfMembers());
-result = XPath_parm1.InfluencedByAndNotSanitized(db, sanitized + other_params1);
-
-// Methods which get a filter (possible tainted) in their second parameter.
-// As some are stand-alone functions with same name as class methods of the previous group,
-// remove the instances of the first group members from the second group.
-CxList XPath_parm2 =  memberMethods+
-   methods.FindByMemberAccess("SDO_DAS_Relational.executeQuery") -
-   XPath_parm1;
-
-CxList tainted_parm2 = All.GetParameters(XPath_parm2, 1);
-CxList other_params2 = All.GetParameters(XPath_parm2) - tainted_parm2;
-other_params2.Add(XPath_parm2.GetTargetOfMembers());
-result.Add(XPath_parm2.InfluencedByAndNotSanitized(db, sanitized + other_params2));
+result = Find_XPath_Injection(inputs);
```

PHP / PHP_High_Risk / Unsafe_Reflection

Code changes

```
---
+++
@@ -1,28 +1 @@
```

```
-CxList methods = Find_Methods();
-CxList interactiveInput = Find_Interactive_Inputs();
-
-CxList ReflectionObjs = All.FindByType("Reflection*");
-
-CxList InvokeTargets = methods.FindByShortName("invoke*").GetTargetOfMembers();
-
-// Tainted Class names
-CxList reflectionClsObj = ReflectionObjs.FindByType("ReflectionClass");
-CxList getNewInstanceTargets = methods.FindByShortName("newInstance*").GetTargetOfMembers();
-CxList taintedreflectionClsObj = (reflectionClsObj * getNewInstanceTargets).DataInfluencedBy(interactiveInput);
-result.Add(taintedreflectionClsObj);
-
-// Tainted Function names
-CxList reflectionFuncObj = ReflectionObjs.FindByType("ReflectionFunction");
-CxList tainedReflectionFuncObj = (reflectionFuncObj * InvokeTargets).DataInfluencedBy(interactiveInput);
-result.Add(tainedReflectionFuncObj);
-
-// Tainted Method names
-CxList reflectionMethodObj = ReflectionObjs.FindByType("ReflectionMethod");
-reflectionMethodObj.Add(reflectionClsObj);
-reflectionMethodObj.Add(All.FindByType("CxObject"));
-
-CxList getMethodTargets = methods.FindByShortName("getMethod*").GetTargetOfMembers() * reflectionMethodObj;
-
-CxList affectedInvoke = methods.FindByShortName("invoke*").InfluencedBy(getMethodTargets);
-CxList taintedInvoke = affectedInvoke.DataInfluencedBy(interactiveInput);
-result.Add(taintedInvoke);
+result = Find_Unsafe_Reflection(Find_Interactive_Inputs());
```

PHP / PHP_Low_Visibility / Insufficient_Sanitization_for_XSS

Code changes

```
---
+++
@@ -1,22 +1,19 @@
-CxList inputs = Find_Inputs() + Find_DB();
-CxList outputs = Find_Interactive_Outputs();
 CxList methods = Find_Methods();
+CxList paramValue = Find_Param().CxSelectDomProperty<Param>(p => p.Value);

-// Find all inappropriate sanitizers
-CxList inapprSanitizers = methods.FindByShortName("*htmlentities*", false) +
-   methods.FindByShortName("*htmlspecialchars*", false);
+// find XSS methods
+string[] xssNames = new[]{ "htmlentities", "htmlspecialchars", "filter_var" };
+CxList strReplace = methods.FindByShortName("str_replace", false);
+CxList xssMethods = methods.FindByShortNames(xssNames, false);
+CxList xssParams = All.NewCxList(
```

```
+   paramValue.GetParameters(xssMethods, 0),
+   paramValue.GetParameters(strReplace, 2)
+);


-CxList quotesString = All.FindByType(typeof(UnknownReference)).FindByShortName(@"ENT_QUOTES");
-
-inapprSanitizers -= quotesString.GetAncOfType(typeof(MethodInvokeExpr));
-
-// Look at inapprSanitizers that is affected by inputs, and affecting outputs (potential XSS),
-// but does not pass through a sanitizer
-
-CxList sanitize = Find_XSS_Sanitize() - inapprSanitizers; // XSS_Sanitize may contain some inappropriates
-
-CxList relevantSanitizers = inapprSanitizers
-   .InfluencedByAndNotSanitized(inputs, sanitize)
-   .InfluencingOnAndNotSanitized(outputs, sanitize);
-
-result = relevantSanitizers;
+// look at inappropriate XSS methods that are affected by inputs,
+// and affecting outputs(potential XSS)
+CxList inputs = Find_DB();
+inputs.Add(Find_Inputs());
+result = Find_Interactive_Outputs()
+   .InfluencedByAndNotSanitized(inputs, Find_XSS_Sanitize())
+   .IntersectWithNodes(xssParams);
```

PHP / PHP_Low_Visibility / Use_of_Non_Cryptographic_Random

Code changes

```
---
+++
@@ -1,3 +1 @@
-CxList methods = Find_Methods();
-List < String > bad_random_methods = new List<String>(){"rand", "mt_rand", "srand", "mt_srand", "str_shuffle", "shuffle", "array_rand"};
-result = methods.FindByShortNames(bad_random_methods);
+result = Common_Low_Visibility.Use_of_Non_Cryptographic_Random(false);
```

PHP / PHP_Medium_Threat / Stored_Unsafe_Reflection

Code changes

```
---
+++
@@ -1,29 +1,3 @@
-CxList methods = Find_Methods();
-CxList db = Find_DB_Out() + Find_Read();
-
-CxList ReflectionObjs = All.FindByType("Reflection*");
-CxList invokers = methods.FindByShortName("invoke*");
-CxList InvokeTargets = invokers.GetTargetOfMembers();
-
```

```
-// Tainted Class names
-CxList reflectionClsObj = ReflectionObjs.FindByType("ReflectionClass");
-CxList getNewInstanceTargets = methods.FindByShortName("newInstance*").GetTargetOfMembers();
-CxList taintedreflectionClsObj = (reflectionClsObj * getNewInstanceTargets).DataInfluencedBy(db);
-result.Add(taintedreflectionClsObj);
-
-// Tainted Function names
-CxList reflectionFuncObj = ReflectionObjs.FindByType("ReflectionFunction");
-CxList tainedReflectionFuncObj = (reflectionFuncObj * InvokeTargets).DataInfluencedBy(db);
-result.Add(tainedReflectionFuncObj);
-
-// Tainted Method names
-CxList reflectionMethodObj = ReflectionObjs.FindByType("ReflectionMethod");
-reflectionMethodObj.Add(reflectionClsObj);
-reflectionMethodObj.Add(All.FindByType("CxObject"));
-
-CxList getMethodTargets = methods.FindByShortName("getMethod*").GetTargetOfMembers() * reflectionMethodObj;
-
-CxList affectedInvoke = invokers.InfluencedBy(getMethodTargets);
-CxList taintedInvoke = affectedInvoke.DataInfluencedBy(db);
-result.Add(taintedInvoke);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+CxList sinks = Find_DB_Out();
+sinks.Add(Find_Read());
+result = Find_Unsafe_Reflection(sinks);
```

PHP / PHP_Medium_Threat / User_Controlled_Dynamic_Variable

Code changes

```
---
+++
@@ -1,16 +1,15 @@
 CxList inputs = Find_Interactive_Inputs();
-CxList extract = Find_Methods().FindByShortName("extract");
-CxList uref = All.FindByType(typeof(UnknownReference));

-CxList non_overwrite_flags = uref.FindByShortNames(new List<String>()
-   { "EXTR_SKIP", "EXTR_PREFIX_SAME", "EXTR_PREFIX_ALL", "EXTR_PREFIX_INVALID", "EXTR_PREFIX_IF_EXISTS", "EXTR_REFS" });
+CxList dynamicVars = Find_Dynamic_Variables();

-CxList extractNonOverwriteFlags = non_overwrite_flags.GetAncOfType(typeof(MethodInvokeExpr)).FindByShortName("extract");
-
-//vulnerability exists only in PHP extract() methods with flags EXTR_OVERWRITE (default flag) and EXTR_IF_EXISTS.
-extract -= extractNonOverwriteFlags;
-extract -= extract.GetTargetOfMembers().GetMembersOfTarget();
-extract -= extract.GetMembersOfTarget().GetTargetOfMembers();
+CxList sanitizers = Find_Array_Intersection_Sanitizers();
+sanitizers.Add(Find_General_Sanitize());
```

```
-CxList sanitize = Find_General_Sanitize();
-result = extract.InfluencedByAndNotSanitized(inputs, sanitize);
+result.Add(
+   // Duplicated nodes
+   (dynamicVars * inputs) - sanitizers,
+   // Flow
+   dynamicVars.InfluencedByAndNotSanitized(inputs, sanitizers)
+   .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
+   .ReduceFlowByPragma()
+);
```

Scala / Scala_Medium_Threat / Use_of_Hardcoded_Cryptographic_Key

Code changes

```
---
+++
@@ -1 +1,26 @@
-result = Common_Medium_Threat.Use_of_Hard_coded_Cryptographic_Key();
+CxList methods = Find_Methods();
+CxList strLiteral = Find_String_Literal();
+CxList unknownRefs = Find_UnknownReference();
+
+CxList secretKey = Find_ObjectCreations().FindByShortName("SecretKeySpec");
+CxList possibleParam = All.GetParameters(secretKey, 0);
+
+CxList sanitizers = All.NewCxList();
+sanitizers.Add(
+   All.GetParameters(methods.FindByMemberAccess("Cipher.getInstance")),
+   methods.FindByMemberAccess("*.getKey"));
+
+CxList hardCoded = All.NewCxList();
+hardCoded.Add(strLiteral);
+
+//FieldDecls in class
+CxList fieldDeclsInfluencedStrs = unknownRefs
+               .FindByAssignmentSide(CxList.AssignmentSide.Left)
+               .InfluencedBy(strLiteral)
+               .GetLastNodesInPath();
+
+hardCoded.Add(Find_Declarators().FindAllReferences(fieldDeclsInfluencedStrs));
+
+result = possibleParam.InfluencedByAndNotSanitized(hardCoded, sanitizers);
+
+result.Add(Common_Medium_Threat.Use_of_Hard_coded_Cryptographic_Key());
```