Version 9.6.3 - Queries Release Notes

New Queries:

Language	Group	Name	CWE
Rust	Rust_Critical	Command_Injection	77
Rust	Rust_Critical	Second_Order_SQL_Injection	89
Rust	Rust_Critical	SQL_Injection	89
Rust	Rust_Critical	Stored_Command_Injection	77
Rust	Rust_Critical	Stored_XSS	79
Rust	Rust_High_Risk	Connection_String_Injection	99
Rust	Rust_High_Risk	Interactive_Absolute_Path_Traversal	36
Rust	Rust_High_Risk	Interactive_Relative_Path_Traversal	23
Rust	Rust_High_Risk	JWT_No_Signature_Verification	287
Rust	Rust_High_Risk	Reflected_XSS	79
Rust	Rust_Low_Visibility	JWT_Excessive_Expiration_Time	613
Rust	Rust_Low_Visibility	JWT_Lack_of_Expiration_Time	613
Rust	Rust_Low_Visibility	JWT_No_Expiration_Time_Validation	613
Rust	Rust_Low_Visibility	JWT_No_NotBefore_Validation	304
Rust	Rust_Medium_Threat	Broken_or_Risky_Hashing_Function	327
Rust	Rust_Medium_Threat	Command_Argument_Injection	78
Rust	Rust_Medium_Threat	DoS_by_Sleep	834
Rust	Rust_Medium_Threat	Encoding_Used_Instead_of_Encryption	311
Rust	Rust_Medium_Threat	Environment_Variable_Injection	454
Rust	Rust_Medium_Threat	Hardcoded_Cryptographic_Key	321
Rust	Rust_Medium_Threat	Hardcoded_Salt	760
Rust	Rust_Medium_Threat	JWT_Sensitive_Information_Exposure	201
Rust	Rust_Medium_Threat	JWT_Use_Of_Hardcoded_Secret	798
Rust	Rust_Medium_Threat	Privacy_Violation	359
Rust	Rust_Medium_Threat	Stored_Absolute_Path_Traversal	22
Rust	Rust_Medium_Threat	Stored_Command_Argument_Injection	78
Rust	Rust_Medium_Threat	Stored_Environment_Variable_Injection	454
Rust	Rust_Medium_Threat	Stored_Relative_Path_Traversal	23
Rust	Rust_Medium_Threat	Unchecked_Input_for_Loop_Condition	606

Changed Queries:

Language	Group	Name	CWE	Changed Fields
Apex	Apex_ISV_Quality_Rules	Old_API_Version	0	**Source** has changed
CPP	CPP_Buffer_Overflow	Improper_Null_Termination	170	**Source** has changed
CPP	CPP_Low_Visibility	NULL_Pointer_Dereference	476	**Source** has changed
CPP	CPP_Medium_Threat	Memory_Leak	401	**Source** has changed
CPP	CPP_MISRA_C_2012	R20_01_Include_Directive_Precedence	0	**Source** has changed
CSharp	CSharp_High_Risk	Reflected_XSS_All_Clients	79	**Source** has changed
CSharp	CSharp_High_Risk	Stored_XSS	79	**Source** has changed
CSharp	CSharp_Low_Visibility	Improper_Resource_Shutdown_or_Release	404	**Source** has changed
CSharp	CSharp_Low_Visibility	Information_Exposure_via_Headers	200	**Source** has changed
CSharp	CSharp_Low_Visibility	Log_Forging	117	**Source** has changed
CSharp	CSharp_Low_Visibility	Trust_Boundary_Violation_in_Session_Variables	501	**Source** has changed

Language	Group	Name	CWE	Changed Fields
CSharp	CSharp_Low_Visibility	Unencrypted_Web_Config_File	312	**Source** has changed
CSharp	CSharp_Medium_Threat	Buffer_Overflow	120	DescriptionId changed from 120 to **1999**
CSharp	CSharp_Medium_Threat	Improper_Restriction_of_XXE_Ref	611	**Source** has changed
CSharp	CSharp_Medium_Threat	Parameter_Tampering	472	**Source** has changed
CSharp	CSharp_Medium_Threat	SSRF	74	**Source** has changed
Go	Go_High_Risk	Second_Order_SQL_Injection	89	**Source** has changed
Go	Go_High_Risk	SQL_Injection	89	**Source** has changed
Java	Java_Android	Allowed_Backup	530	**Source** has changed
Java	Java_Android	Client_Side_Injection	89	**Source** has changed
Java	Java_Android	Debuggable_App	668	**Source** has changed
Java	Java_Android	Exported_Content_Provider_Without_Protective_Permissions	668	**Source** has changed
Java	Java_Android	Exported_Service_Without_Protective_Permissions	668	**Source** has changed
Java	Java_Android	Failure_To_Implement_Least_Privilege	250	**Source** has changed
Java	Java_Android	Insecure_Android_SDK_Version	477	**Source** has changed
Java	Java_Android	No_Installer_Verification_Implemented	829	**Source** has changed
Java	Java_Android	Use_of_WebView_AddJavascriptInterface	749	**Source** has changed
Java	Java_Android	Weak_Encryption	326	**Source** has changed
Java	Java_Best_Coding_Practice	Hardcoded_Absolute_Path	426	**Source** has changed
Java	Java_High_Risk	Reflected_XSS_All_Clients	79	**Source** has changed
Java	Java_High_Risk	Stored_XSS	79	**Source** has changed
Java	Java_Low_Visibility	Improper_Resource_Access_Authorization	285	**Source** has changed
Java	Java_Low_Visibility	Log_Forging	117	**Source** has changed
Java	Java_Low_Visibility	Open_Redirect	601	**Source** has changed
Java	Java_Low_Visibility	Sensitive_Cookie_in_HTTPS_Session_Without_Secure_Attribute	614	**Source** has changed
Java	Java_Low_Visibility	Trust_Boundary_Violation_in_Session_Variables	501	**Source** has changed
Java	Java_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Java	Java_Low_Visibility	Use_Of_Hardcoded_Password_In_Config	260	**Source** has changed
Java	Java_Medium_Threat	Parameter_Tampering	472	**Source** has changed
Java	Java_Medium_Threat	Plaintext_Storage_of_a_Password	256	**Source** has changed
Java	Java_Medium_Threat	Privacy_Violation	359	**Source** has changed
Java	Java_Medium_Threat	Use_of_a_One_Way_Hash_without_a_Salt	759	**Source** has changed
Java	Java_Struts	Struts_Unvalidated_Action_Form	108	**Source** has changed
JavaScript	JavaScript_Low_Visibility	Client_JQuery_Deprecated_Symbols	477	**Source** has changed
JavaScript	JavaScript_Medium_Threat	Frameable_Login_Page	829	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	SSL_Verification_Bypass	599	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Use_of_Broken_or_Risky_Cryptographic_Algorithm	327	**Source** has changed
JavaScript	JavaScript_Vue	Declaration_of_Multiple_Vue_Components_per_File	710	**Source** has changed
JavaScript	JavaScript_Vue	Inconsistent_Component_Top_Level_Elements_Ordering	710	**Source** has changed
JavaScript	JavaScript_XS	XS_CSRF	352	**Source** has changed
Python	Python_High_Risk	Code_Injection	94	**Source** has changed
Python	Python_High_Risk	Command_Injection	77	**Source** has changed
Python	Python_High_Risk	Connection_String_Injection	99	**Source** has changed
Python	Python_High_Risk	LDAP_Injection	90	**Source** has changed
Python	Python_High_Risk	Local_File_Inclusion	829	**Source** has changed
Python	Python_High_Risk	OS_Access_Violation	77	**Source** has changed
Python	Python_High_Risk	Reflected_XSS_All_Clients	79	**Source** has changed
Python	Python_High_Risk	Resource_Injection	99	**Source** has changed

Language	Group	Name	CWE	Changed Fields
Python	Python_High_Risk	SQL_Injection	89	**Source** has changed
Python	Python_High_Risk	Unsafe_Deserialization	502	**Source** has changed
Python	Python_High_Risk	XPath_Injection	643	**Source** has changed
Python	Python_Low_Visibility	Command_Argument_Injection	88	**Source** has changed
Python	Python_Low_Visibility	Log_Forging	117	**Source** has changed
Python	Python_Low_Visibility	Marshmallow_Dumping_Without_Validation	1173	**Source** has changed
Python	Python_Low_Visibility	Use_Of_Hardcoded_Password	259	**Source** has changed
Python	Python_Medium_Threat	CSRF	352	**Source** has changed
Python	Python_Medium_Threat	Header_Injection	113	**Source** has changed
Python	Python_Medium_Threat	Missing_HSTS_Header	346	**Source** has changed
Python	Python_Medium_Threat	Object_Access_Violation	610	**Source** has changed
Python	Python_Medium_Threat	Open_Redirect	601	**Source** has changed
Python	Python_Medium_Threat	Path_Traversal	22	**Source** has changed
Python	Python_Medium_Threat	Privacy_Violation	359	**Source** has changed
Python	Python_Medium_Threat	ReDoS_Injection	400	**Source** has changed
Python	Python_Medium_Threat	SSRF	918	**Source** has changed
Python	Python_Medium_Threat	Uncontrolled_Format_String	134	**Source** has changed

Changed Source:

Apex / Apex_ISV_Quality_Rules / Old_API_Version

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 //Give warning if api version is 2 releases old.
 //NOTE: The current version and amount of old versions are hardcoded in the query.
-//current version is: 56.0
-double currentVersion = 56.0;
+//current version is: 59.0
+double currentVersion = 59.0;
 //the amount of versions back to warn is: 2
 double n = 2;
 double oldVersion = (currentVersion - n);
```

CPP / CPP_Buffer_Overflow / Improper_Null_Termination

Code changes

```
---
+++
@@ -112,6 +112,50 @@
     GetAncOfType<MethodInvokeExpr>();
 sanitizers.Add(allRefsOfRelevantParams.GetParameters(sanitizedStrncpy, 0));

+// in strncpy, when the size_t is greater than the length of the source in the strncpy function, the result must be removed.
+CxList strncpyCalls = methods.FindByShortName("strncpy");
+CxList rankSpecifiers = Find_RankSpecifier();
+
```

```
+foreach (CxList call in strncpyCalls) {
+   CxList src = parameters.GetParameters(call, 1).CxSelectDomProperty<Param>(x => x.Value);
+   int srcLength = -1;
+
+   CxList srcString = src.FindByAbstractValue(x => x is StringAbstractValue);
+   if (srcString.Count != 0) srcLength = srcString.GetName().Length;
+
+   CxList srcDefinition = declarators.FindDefinition(src);
+
+   if (src.FindByType<UnknownReference>().Count != 0) {
+       CxList rankSpecifier = rankSpecifiers.GetByAncs(srcDefinition.GetAncOfType<VariableDeclStmt>());
+       List<long> sizesOfArrayPar = new List<long> {};
+       sizesOfArrayPar.AddRange(rankSpecifier.CxSelectElementValue<RankSpecifier, long>(x => x.Dimensions));
+
+       if (sizesOfArrayPar.Count > 0) {
+           if (sizesOfArrayPar[0] > 1) srcLength = (int) sizesOfArrayPar[0];
+       }
+   }
+   else if (srcDefinition.GetAssigner().FindByType<ArrayCreateExpr>().Count != 0) {
+       CxList val = srcDefinition.GetAssigner().FindByType<ArrayCreateExpr>();
+       try{
+           CxList sizes = val.CxSelectDomProperty<ArrayCreateExpr>(x => x.Sizes);
+           CxList num = Find_Integer_Literals();
+           srcLength = int.Parse(num.FindById(sizes[0].NodeId).GetName());
+       } catch(Exception){};
+   }
+
+   CxList sizeT = parameters.GetParameters(call, 2).CxSelectDomProperty<Param>(x => x.Value);
+   long? sizeT_Value = -1;
+   sizeT.FindByAbstractValue(x => {
+       if (x is IntegerIntervalAbstractValue) {
+           sizeT_Value = (x as IntegerIntervalAbstractValue).UpperIntervalBound;
+           return true;}
+       else
+           return false;});
+
+   if (sizeT_Value != -1 & srcLength != -1 & sizeT_Value > srcLength)
+       sanitizers.Add(parameters.CxSelectDomProperty<Param>(x => x.Value).GetParameters(call, 0));
+}
+

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

CPP / CPP_Low_Visibility / NULL_Pointer_Dereference

Code changes

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

```
@@ -10,13 +10,10 @@
 pointerTotarget -= pointerTotarget.GetByAncs(ifs);

 // Remove pointers used as indexerRefs
-CxList indexerRefs = Find_IndexerRefs();
-pointerTotarget -= pointerTotarget.GetByAncs(indexerRefs);
+pointerTotarget -= pointerTotarget.GetByAncs(Find_IndexerRefs());

 ////////////////////// Null Values ////////////////////////
-CxList nullValues = All.NewCxList();
-nullValues.Add(Find_NullLiteral());
-nullValues.Add(Find_CharLiteral().FindByName("\0"));
+CxList nullValues = All.NewCxList(Find_NullLiteral(), Find_CharLiteral().FindByName("\0"));
 CxList zero = Find_IntegerLiterals().FindByShortName("0");

 //Remove 0 that are in IterationStmt
@@ -57,11 +54,10 @@
 CxList influencing = nullValues - removeZeroInReturn;

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

 // Remove the 0 that are in conditions (if, while, for, etc...)
@@ -113,7 +109,7 @@
 sanitizers.Add((secondAssignments.GetFathers() * assignees).FindByShortName("Pointer"));
 //Same as above but for a transitive scenario such as "int x = 0; int * y = &x; *y = 9;"
 CxList nullValueAssigners = assignees.GetAssigner() * nullValues;
-CxList nullValuesInfluencingAssignees = (Find_Unarys() * assignees).FindByShortName("Pointer")
+CxList nullValuesInfluencingAssignees = (unary * assignees).FindByShortName("Pointer")
     .DataInfluencedBy(nullValueAssigners);
 sanitizers.Add(nullValuesInfluencingAssignees.GetLastNodesInPath());

@@ -192,4 +188,6 @@
         }
     }
 }
-result.Add(pointerToInfluencing);
+
+result.Add(Find_Null_Array_Access(), // Include access like buff[x]=x after buff has been set to null
+           pointerToInfluencing);
```

CPP / CPP_Medium_Threat / Memory_Leak

Code changes

```
---
+++
@@ -107,7 +107,8 @@


 // 8. Disregard allocation of auto_ptr variables (they perform automatic cleanup
 // when the object is no longer needed)
-allocatedReferences -= allocatedReferences.FindByType("auto_ptr");
+CxList autoPtrDecls = Find_Smart_Pointer_Declarators(new[]{"auto_ptr"});
+allocatedReferences -= allocatedReferences.FindAllReferences(autoPtrDecls);


 // Remove variables that have destructor assigned to array creation expressions to be deleted
 delDeallocation -= varToDelete;
```

CPP / CPP_MISRA_C_2012 / R20_01_Include_Directive_Precedence

Code changes

```
---
+++
@@ -42,7 +42,7 @@
    {
        int fileId = include
            .CxSelectElementValue<CSharpGraph, LinePragma>(p => p.LinePragma)
-           .Select(s => s.GetFileId())
+           .Select(s => s.FileId)
            .FirstOrDefault();
        if (dicIncludes.ContainsKey(fileId))
        {
@@ -60,7 +60,7 @@
    {
        int fileId = targetElement
            .CxSelectElementValue<CSharpGraph, LinePragma>(p => p.LinePragma)
-           .Select(s => s.GetFileId())
+           .Select(s => s.FileId)
            .FirstOrDefault();
        if (dicTargetElements.ContainsKey(fileId))
        {
```

CSharp / CSharp_High_Risk / Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -28,4 +28,7 @@
    sanitized.Add(Find_ASP_MVC_Minimal_Filter_Sanitizers(sanitized));


    result = inputs.InfluencingOnAndNotSanitized(outputs, sanitized).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+   // When the flow goes through an "Remote input" (a request to a remote server) it is not vulnerable
```

```
+   result -= result.IntersectWithNodes(Find_Remote_Requests());
 }
```

CSharp / CSharp_High_Risk / Stored_XSS

Code changes

```
---
+++
@@ -38,4 +38,7 @@
        inputs.InfluencingOnAndNotSanitized(outputs, sanitize));


    result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
+
+   // When the flow goes through an "Remote input" (a request to a remote server) it is not vulnerable
+   result -= result.IntersectWithNodes(Find_Remote_Requests());
 }
```

CSharp / CSharp_Low_Visibility / Improper_Resource_Shutdown_or_Release

Code changes

```
---
+++
@@ -60,15 +60,15 @@


 typeOfObjectQuery.Add(TypeRef.FindAllReferences(ClassInheritsFrom).GetFathers());
 typeOfObjectQuery -= MethodDecl;
-CxList variableDecl = typeOfObjectQuery.FindByType(typeof(VariableDeclStmt));
+CxList variableDecl = typeOfObjectQuery.FindByType<VariableDeclStmt>();
 typeOfObjectQuery.Add(Find_Declarators().GetByAncs(variableDecl));
 CxList RefTypeOfObjectQuery = All.FindAllReferences(typeOfObjectQuery);


 CxList ResourceAllocationInstances = CreateExpr.FindByShortNames(ResourceTypeNames);
 ResourceAllocationInstances.Add(CreateExpr * typeOfObjectQuery);
-ResourceAllocationInstances.Add(RefTypeOfObjectQuery.FindByType(typeof(MethodInvokeExpr)));
-CxList wrappingObjects = All.FindAllReferences(All.GetClass(ResourceAllocationInstances.GetAncOfType(typeof(ConstructorDecl))))
-    .GetFathers().FindByType(typeof(ObjectCreateExpr));
+ResourceAllocationInstances.Add(RefTypeOfObjectQuery.FindByType<MethodInvokeExpr>());
+CxList wrappingObjects = All.FindAllReferences(All.GetClass(ResourceAllocationInstances.GetAncOfType<ConstructorDecl>()))
+    .GetFathers().FindByType<ObjectCreateExpr>();



 /*
@@ -130,8 +130,7 @@
 ResourceAllocationInstances = ResourceAllocationInstances.DataInfluencingOn(ObjAssigneeReferences);


 /* Collect the Try statements Block */
-CxList TryEnds = ResourceAllocationInstances.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-
+CxList TryEnds = ResourceAllocationInstances.GetLastNodesInPath();
```

```
 CxList TryBlocks = All.NewCxList();
 foreach(CxList tryCatch in Trys){
@@ -151,19 +150,18 @@




-CxList TryBlock = (ResourceAllocationInstances.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly) + wrappingObjects).GetByAncs(TryBlocks);
-
+CxList TryBlock = All.NewCxList(ResourceAllocationInstances.GetFirstNodesInPath(), wrappingObjects).GetByAncs(TryBlocks);


 /* Collect Methods that close resources passed through parameter */
 CxList allCloseMethod = MethodInvoke.FindByShortNames(ResourceCloseMethods);
-CxList RefMethodClosesInvoke = All.FindAllReferences(MethodDecl.GetMethod(allCloseMethod)).FindByType(typeof(MethodInvokeExpr));
+CxList RefMethodClosesInvoke = All.FindAllReferences(MethodDecl.GetMethod(allCloseMethod)).FindByType<MethodInvokeExpr>();
 CxList closes = TryEnds * allCloseMethod;

 /*join the close methods with return on try*/
 CxList returnStmt = base.Find_ReturnStmt();
-CxList trysFilter = returnStmt.GetAncOfType(typeof(TryCatchFinallyStmt));
+CxList trysFilter = returnStmt.GetAncOfType<TryCatchFinallyStmt>();
 foreach(CxList myTry in trysFilter){
-   CxList curFinally = All.GetFinallyClause(myTry.GetAncOfType(typeof(TryCatchFinallyStmt)));
+   CxList curFinally = All.GetFinallyClause(myTry.GetAncOfType<TryCatchFinallyStmt>());
     if(curFinally.TryGetCSharpGraph<StatementCollection>().Count > 0){
         closes.Add(allCloseMethod.GetByAncs(curFinally));
     }
@@ -176,7 +174,7 @@
 CxList closedResources = All.NewCxList();
 foreach(CxList myTry in TryBlock){

-   CxList curFinally = All.GetFinallyClause(myTry.GetAncOfType(typeof(TryCatchFinallyStmt)));
+   CxList curFinally = All.GetFinallyClause(myTry.GetAncOfType<TryCatchFinallyStmt>());
     //we test id we the statement try have a finally
     if(curFinally.TryGetCSharpGraph<StatementCollection>().Count > 0){
         // remove resource that close on the finally block
@@ -184,18 +182,21 @@
         CxList elemtToClose = All.FindAllReferences(closes.GetByAncs(curFinally).GetTargetOfMembers());
         CxList elemtToCloseRight =  elemtToClose.GetAssigner().GetByAncs(myTry);
         CxList elemtToCloseLeft = elemtToClose.GetByAncs(myTry);
-        closedResources.Add(elemtToCloseRight);
-        closedResources.Add(elemtToCloseLeft);
+        closedResources.Add(elemtToCloseRight, elemtToCloseLeft);

         // remove resources that are wrapped and are close along recurring to auxiliar method
-        CxList openedResources = All.FindAllReferences(RefMethodClosesInvoke.GetByAncs(curFinally).GetTargetOfMembers()).GetByAncs(TryBlocks).FindByAssignmentSide(CxList.AssignmentSide.Left);
-        CxList resourcesWrapperClass = All.FindDefinition(TypeRef.FindByFathers(openedResources.GetAssigner().FindByType(typeof(ObjectCreateExpr))));
-        CxList ResourcesAllocatedOnWrapperConstructor = ResourceAllocationInstances.GetByAncs(constructorDecl.GetByAncs(resourcesWrapperClass));
+        CxList openedResources = All.FindAllReferences(RefMethodClosesInvoke.GetByAncs(curFinally).GetTargetOfMembers())
```

```
+           .GetByAncs(TryBlocks).FindByAssignmentSide(CxList.AssignmentSide.Left);
+       CxList resourcesWrapperClass = All.FindDefinition(TypeRef.FindByFathers(openedResources.GetAssigner()
+           .FindByType<ObjectCreateExpr>()));
+       CxList ResourcesAllocatedOnWrapperConstructor = ResourceAllocationInstances.GetByAncs(constructorDecl
+           .GetByAncs(resourcesWrapperClass));
        closedResources.Add(ResourcesAllocatedOnWrapperConstructor);

        // remove resources that are wrapped, opened through a method and are close along recurring to auxiliar method
-       CxList methodsReturningResources = All.FindDefinition(openedResources.GetAssigner().FindByType(typeof(MethodInvokeExpr)));
-       closedResources.Add(ResourceAllocationInstances.GetByAncs(ResourceAllocationInstances.GetByAncs(methodsReturningResources)));
+       CxList methodsReturningResources = All.FindDefinition(openedResources.GetAssigner().FindByType<MethodInvokeExpr>());
+       closedResources.Add(ResourceAllocationInstances.GetByAncs(ResourceAllocationInstances
+           .GetByAncs(methodsReturningResources)));
    }
 }


@@ -207,13 +208,23 @@
 //Remove reults from Test Methods
 tempResult -= tempResult.GetByAncs(Find_Test_Methods());

+
+// Exclude resources properly disposed/closed in finally statements
+CxList finallyCloses = allCloseMethod.GetByAncs(Trys.CxSelectDomProperty<TryCatchFinallyStmt>(_ => _.Finally));
+CxList resourceInstances = ResourceAllocationInstances.GetFirstNodesInPath();
+foreach(CxList res in tempResult.GetFirstNodesInPath())
+{
+   CxList methodDecl = res.GetAncOfType<MethodDecl>();
+   closedResources.Add(res.InfluencingOnAndNotSanitized(finallyCloses.GetByAncs(methodDecl), resourceInstances - res));
+}
+tempResult -= closedResources;
 /*
   Keep only the longest flow starting from the resource
   It is considered the first node of the flow, the place where the resource is allocated
  */
 Dictionary<int,Tuple<int, CxList>> validFlows = new Dictionary<int, Tuple<int, CxList>>();
 foreach (CxList res in tempResult.GetCxListByPath(true)){
-   CSharpGraph init = res.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly).GetFirstGraph();
+   CSharpGraph init = res.GetFirstNodesInPath().GetFirstGraph();
    CxList allNodes = res.GetStartAndEndNodes(CxList.GetStartEndNodesType.AllNodes);
    int numNodes = allNodes.Count;
    if(init != null){
```

CSharp / CSharp_Low_Visibility / Information_Exposure_via_Headers

Code changes

```
---
+++
@@ -33,7 +33,7 @@
    CxList httpProtocolList = (memberList* webConfig).FindByName("system.webServer",false);
```

```
    CxList xPoweredList = (webConfig * stringList).FindByShortName("X-Powered-By");
    CxList customHeadersList = xPoweredList.GetAssignee().GetTargetOfMembers().FindByName("*customHeaders", false);
-   List<int> listFiles = customHeadersList.CxSelectElementValues<CSharpGraph,int>(x => x.LinePragma.GetFileId());
+   List<int> listFiles = customHeadersList.CxSelectElementValues<CSharpGraph,int>(x => x.LinePragma.FileId);
    CxList toRemoveFromHttpProtocol = All.NewCxList();
    foreach(int aux in listFiles){
        toRemoveFromHttpProtocol.Add(httpProtocolList.FindByFileId(aux));
```

CSharp / CSharp_Low_Visibility / Log_Forging

Code changes

```
---
+++
@@ -4,3 +4,6 @@
 CxList sanitize = Find_Log_Sanitizers();

 result = Log.InfluencedByAndNotSanitized(Inputs, sanitize);
+
+// When the flow goes through an "Remote input" (a request to a remote server) it is not vulnerable
+result -= result.IntersectWithNodes(Find_Remote_Requests());
```

CSharp / CSharp_Low_Visibility / Trust_Boundary_Violation_in_Session_Variables

Code changes

```
---
+++
@@ -37,3 +37,6 @@
     .FindByFathers(indexerRefs.FindByAssignmentSide(CxList.AssignmentSide.Left)));

 result = sinks.InfluencedByAndNotSanitized(inputs, sanitizers);
+
+// When the flow goes through an "Remote input" (a request to a remote server) it is not vulnerable
+result -= result.IntersectWithNodes(Find_Remote_Requests());
```

CSharp / CSharp_Low_Visibility / Unencrypted_Web_Config_File

Code changes

```
---
+++
@@ -1,15 +1,22 @@
 /// <summary>
-/// This query returns all the web config files that are unencrypted
+/// This query flag only sections in web config files that are unencrypted
 /// </summary>

 CxList webConfigs = Find_Web_Config();

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

```
+       "keyinfo","keyname","encryptionmethod", "encryptedkey", "encrypteddata","cipherdata","ciphervalue"}, false);
+
+string[] targetConfigSections = new string[]{
+   "connectionStrings", "appSettings", "identity",
+   "machineKey", "sessionState", "smtp"};
+CxList usnafeWebConfigSections = xmlTokens.FindByShortNames(targetConfigSections, false);


 // encrypted keynames
-xmlTokens = xmlTokens.GetMembersOfTarget();
-xmlTokens -= xmlTokens.FindByShortName("configprotectionprovider", false);
+CxList safeSections = Find_MemberAccesses().FindByShortName("configprotectionprovider", false).GetTargetOfMembers();
+CxList vulnerableConfigSections = usnafeWebConfigSections - safeSections;


-result = All.GetClass(xmlTokens);
+// Flag only Section Parent Name
+result =  vulnerableConfigSections.GetAncOfType<IfStmt>().CxSelectDomProperty<IfStmt>(_ => _.Condition);
```

CSharp / CSharp_Medium_Threat / Improper_Restriction_of_XXE_Ref

Code changes

```
---
+++
@@ -6,19 +6,30 @@
 CxList objectCreations = Find_ObjectCreations();
 CxList unknownList = Find_Unknown_References();

-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
+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
+   Find_XXE_XPathDocument());                   //NOT Sanitized by default in versions < 4.5.2

 CxList xxeSanitzer = Find_XXE_Sanitize();
 CxList allRefsElmentSanitized = unknownList.FindAllReferences(xxeSanitzer.GetTargetOfMembers());
 XXE -= allRefsElmentSanitized.GetMembersOfTarget();

-CxList sanitizers = All.NewCxList();
-sanitizers.Add(integers);
-sanitizers.Add(objectCreations.FindByType("XmlNodeReader"));
+CxList sanitizers = All.NewCxList(
+   integers,
+   objectCreations.FindByType("XmlNodeReader"));
```

```
+
+//XmlReader is safe unless the "DtdProcessing" property of
+//the "settings"(XmlReaderSettings) object is set to "Parse"
+CxList xmlReaderSinks = Find_Methods().FindByMemberAccess("XmlReader.Create");
+CxList xmlReaderResults = XXE.IntersectWithNodes(xmlReaderSinks);
+XXE -= xmlReaderResults;
+
+CxList dtdProcessingParse = Find_MemberAccesses().FindByMemberAccess("DtdProcessing.Parse");
+CxList xmlReaderXxeInfluencedByParse = xmlReaderResults.DataInfluencedBy(dtdProcessingParse).GetLastNodesInPath();
+CxList vulnerableXmlReaderXXE = xmlReaderResults.IntersectWithNodes(xmlReaderXxeInfluencedByParse);
+XXE.Add(vulnerableXmlReaderXXE);


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

CSharp / CSharp_Medium_Threat / Parameter_Tampering

Code changes

```
---
+++
@@ -14,3 +14,6 @@


 result = db.InfluencedByAndNotSanitized(input, sanitize);
 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+// When the flow goes through an "Remote input" (a request to a remote server) it is not vulnerable
+result -= result.IntersectWithNodes(Find_Remote_Requests());
```

CSharp / CSharp_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -206,3 +206,6 @@


 //Add to results
 result.Add(socketPaths);
+
+// When the flow goes through an "Remote input" (a request to a remote server) it is not vulnerable
+result -= result.IntersectWithNodes(Find_Remote_Requests());
```

Go / Go_High_Risk / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,14 +1,12 @@
 // Sources
 //  - 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_DB_Out_GORM());


 // Sanitizers
-CxList sanitizers = Find_DB_Sanitize();
+CxList sanitizers = All.NewCxList(Find_DB_Sanitize(), Find_DB_Sanitize_GORM());


 // Sinks for Second Order SQL Injection are SQL statements
 CxList outputs = Find_DB_In();


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

Go / Go_High_Risk / SQL_Injection

Code changes

```
---
+++
@@ -1,9 +1,7 @@
 CxList inputPaths = Find_Inputs();


-CxList dbPaths = Find_DB_Out();
-dbPaths.Add(Find_DB_In());
+CxList dbPaths = All.NewCxList(Find_DB_Out(), Find_DB_In(), Find_DB_Out_GORM());


-CxList sanitizers = Find_DB_Sanitize();
+CxList sanitizers = All.NewCxList(Find_DB_Sanitize(), Find_DB_Sanitize_GORM());


-result = inputPaths.InfluencingOnAndNotSanitized(dbPaths, sanitizers);
-result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+result = inputPaths.InfluencingOnAndNotSanitized(dbPaths, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Java / Java_Android / Allowed_Backup

Code changes

```
---
+++
@@ -1,4 +1,6 @@
+// This query looks for backable apps -> android:allowBackup is not set or set as true
 if (Find_Android_Java().Count > 0)
 {
-   result = Common_Android.Allowed_Backup();
+   result.Add(cxXPath.FindXmlNodesByLocalName("*AndroidManifest.xml", 2, "application"));
+   result -= cxXPath.FindXmlNodesByLocalName("*AndroidManifest.xml", 2, "application", true, "allowBackup", "false");
 }
```

Java / Java_Android / Client_Side_Injection

Code changes

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

```
@@ -42,38 +42,28 @@
    contentProviderFirstParameter,
    All.GetParameters(textChanged, 0));

-CxList strings = Find_Strings();

 //Add possible inputs from other applications that are allowed using intent filters.
 CxList getIntents = methods.FindByShortName("getIntent");
-CxList sendIntentFilters = strings.FindByName(@"""android.intent.action.SEND""");

-//Find the activities containing the filters
-CxList filterFathers = sendIntentFilters.GetAncOfType<IfStmt>();
-filterFathers = filterFathers.GetFathers().GetAncOfType<IfStmt>();
-filterFathers = filterFathers.GetFathers().GetAncOfType<IfStmt>();
+const string namespaceURI = "http://schemas.android.com/apk/res/android";
+IEnumerable<CxXmlDoc> androidManifests = cxXPath.GetXmlFiles("AndroidManifest.xml", true);
+string XPath = "//action[@*[name()='android:name' and .='android.intent.action.SEND']]/ancestor::activity";

-CxList activityNames = Find_Android_Settings().FindByName(@"*ACTIVITY.ANDROID_NAME");
-activityNames = activityNames.GetByAncs(filterFathers);
-activityNames = strings.GetByAncs(activityNames.GetFathers());
-
-foreach (CxList activityFather in filterFathers)
+List<string> classNames = new List<string>();
+foreach (CxXmlDoc androidManifest in androidManifests)
 {
-   //Get the relevant activity name
-   CxList activity = activityNames.GetByAncs(activityFather);
-   CSharpGraph g = activity.TryGetCSharpGraph<CSharpGraph>();
-   if (g != null && g.ShortName != null)
+   XPathNavigator navigator = androidManifest.CreateNavigator();
+   XPathNodeIterator nodeIterator = navigator.Select(XPath);
+   while (nodeIterator.MoveNext())
    {
-       string curName = g.ShortName.Trim('"');
-       string[] curNameSplit = curName.Split('.');
-       curName = curNameSplit[curNameSplit.Length - 1];
-       //Find the class matching the activity name
-       CxList curClass = classes.FindByShortName(curName);
-       //Find the intents in curClass
-       CxList curIntents = getIntents.GetByAncs(curClass);
-       inputs.Add(curIntents);
+       XPathNavigator currentNodeNavigator = nodeIterator.Current;
+       string activityName = currentNodeNavigator.GetAttribute("name", namespaceURI);
+       string[] actNameSplit = activityName.Split('.');
+       classNames.Add(actNameSplit[actNameSplit.Length - 1]);
    }
 }
```

```
+inputs.Add(getIntents.GetByAncs(classes.FindByShortNames(classNames)));


 CxList recieveMethod = methodDeclaration.GetByAncs(Find_Android_Exported_BroadcastReceiver()).FindByShortName("onRecieve");
 inputs.Add(All.GetParameters(recieveMethod, 1));
```

Java / Java_Android / Debuggable_App

Code changes

```
---
+++
@@ -1,4 +1,4 @@
 if(Find_Android_Java().Count > 0)
 {
-   result = Common_Android.Debuggable_App();
+   result = cxXPath.FindXmlAttributesByNameAndValue("*AndroidManifest.xml", 2, "debuggable", "true");
 }
```

Java / Java_Android / Exported_Content_Provider_Without_Protective_Permissions

Code changes

```
---
+++
@@ -1,4 +1,76 @@
-if(Find_Android_Settings().Count > 0 && Find_Android_Java().Count > 0)
+if(cxScan.IsFrameworkActive("Android") && Find_Android_Java().Count > 0)
 {
-   result = Common_Android.Exported_Content_Provider_Without_Protective_Permissions();
+   const string namespaceURI = "http://schemas.android.com/apk/res/android";
+   IEnumerable<CxXmlDoc> androidManifests = cxXPath.GetXmlFiles("AndroidManifest.xml", true);
+
+   // Collect unsafe permissions (no protectionLevel or set to 'normal')
+   string noProtectionPermission = "//permission[not(@*[name()='android:protectionLevel'])]";
+   string normalProtectionPermission = "//permission[@*[name()='android:protectionLevel' and .='normal']]";
+
+   List<string> unsafePermissions = new List <string>();
+   foreach (CxXmlDoc androidManifest in androidManifests)
+   {
+       XPathNavigator navigator = androidManifest.CreateNavigator();
+
+       XPathNodeIterator nodeIterator = navigator.Select(noProtectionPermission);
+       while (nodeIterator.MoveNext())
+       {
+           XPathNavigator currentNodeNavigator = nodeIterator.Current;
+           unsafePermissions.Add(currentNodeNavigator.GetAttribute("name", namespaceURI));
+       }
+
+       nodeIterator = navigator.Select(normalProtectionPermission);
+       while (nodeIterator.MoveNext())
+       {
+           XPathNavigator currentNodeNavigator = nodeIterator.Current;
```

```
+           unsafePermissions.Add(currentNodeNavigator.GetAttribute("name", namespaceURI));
+       }
+   }
+
+   // Provider has 'android:exported' attribute set to true
+   string exportedProvider = "//*[name()='provider'][@*[name()='android:exported' and .='true']]";
+   if (Insecure_Android_SDK_Version().Count > 0)
+   {
+       // If the SDK version is less or equal to 16, providers without 'android:exported' attribute
+       // are exported by default
+       exportedProvider =
+           "//*[name()='provider'][@*[name()='android:exported' and .='true'] or not(@*[name()='android:exported'])]";
+   }
+
+   foreach (CxXmlDoc androidManifest in androidManifests)
+   {
+       XPathNavigator navigator = androidManifest.CreateNavigator();
+       XPathNodeIterator exportedIterator = navigator.Select(exportedProvider);
+       foreach (XPathNavigator exported in exportedIterator)
+       {
+           string permissionAttrValue = exported.GetAttribute("permission", namespaceURI);
+           string readPermissionAttrValue = exported.GetAttribute("readPermission", namespaceURI);
+           string writePermissionAttrValue = exported.GetAttribute("writePermission", namespaceURI);
+
+           // Either 'permission' or both 'readPermission' and 'writePermission' have to be set
+           if (permissionAttrValue == "" && (readPermissionAttrValue == "" || writePermissionAttrValue == ""))
+           {
+               result.Add(cxXPath.CreateXmlNode(exported.Clone(), androidManifest, 2, false));
+               continue;
+           }
+
+           if (permissionAttrValue != "" && unsafePermissions.Contains(permissionAttrValue))
+           {
+               result.Add(cxXPath.CreateXmlNode(exported.Clone(), androidManifest, 2, false));
+               continue;
+           }
+
+           if (readPermissionAttrValue != "" && unsafePermissions.Contains(readPermissionAttrValue))
+           {
+               result.Add(cxXPath.CreateXmlNode(exported.Clone(), androidManifest, 2, false));
+               continue;
+           }
+
+           if (writePermissionAttrValue != "" && unsafePermissions.Contains(writePermissionAttrValue))
+           {
+               result.Add(cxXPath.CreateXmlNode(exported.Clone(), androidManifest, 2, false));
+               continue;
+           }
```

```
+       }
+   }
 }
```

Java / Java_Android / Exported_Service_Without_Protective_Permissions

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-if(Find_Android_Settings().Count > 0 && Find_Android_Java().Count > 0)
+if(cxScan.IsFrameworkActive("Android") && Find_Android_Java().Count > 0)
 {
    result = Common_Android.Exported_Service_Without_Protective_Permissions();
 }
```

Java / Java_Android / Failure_To_Implement_Least_Privilege

Code changes

```
---
+++
@@ -4,10 +4,9 @@
 CxList typeRef = Find_TypeRef();
 CxList strings = Find_Strings();
 CxList methods = Find_Methods();
-CxList settings = Find_Android_Settings();
-CxList androidPermission = settings.FindByShortName("*android.permission.*", false);
+CxList androidPermission = cxXPath.FindXmlAttributesByNameAndValue("*AndroidManifest.xml", 2, "name", "android\\.permission.*", true);
 // Application Requiered Network access but not uses it
-CxList permissionInternet = androidPermission.FindByShortName("*android.permission.INTERNET*", false);
+CxList permissionInternet = androidPermission.FindByShortName("INTERNET*", false);
 CxList usingNetwork = All.NewCxList(
    typeRef.FindByTypes(new string[]{ "HttpClient", "OkHttpClient" }),
    methods.FindByMemberAccess("URL.openConnection", false),
@@ -21,8 +20,8 @@

 // Application Requiered SMS access but not uses it
 CxList permissionSMS = androidPermission.FindByShortNames(new string[]{
-       "*android.permission.SEND_SMS*",
-       "*android.permission.RECEIVE_SMS*" }, false);
+       "SEND_SMS*",
+       "RECEIVE_SMS*" }, false);

 CxList usingSMS = All.FindByShortName("*SmsManager*");

@@ -31,7 +30,7 @@
 }

 // Read SMS content
-CxList permissionSmsContect = androidPermission.FindByShortName("*android.permission.READ_SMS*");
```

```
+CxList permissionSmsContect = androidPermission.FindByShortName("READ_SMS*");


 // Find SMS content string
 if(permissionSmsContect.Count > 1)
@@ -53,10 +52,10 @@
 CxList intentToCall = actionCall.GetParameters(createIntent);


 CxList permissionPhone = androidPermission.FindByShortNames(new string[] {
-       "*android.permission.READ_PHONE_STATE*",
-       "*android.permission.MODIFY_PHONE_STATE*",
-       "*android.permission.PROCESS_OUTGOING_CALLS*",
-       "*android.permission.CALL_PHONE*"}, false);
+       "READ_PHONE_STATE*",
+       "MODIFY_PHONE_STATE*",
+       "PROCESS_OUTGOING_CALLS*",
+       "CALL_PHONE*"}, false);


 CxList usingPhone = All.FindByShortName("*TelephonyManager*");
 usingPhone.Add(intentToCall);
@@ -67,8 +66,8 @@


 // Application Requiered GPS access but does not use it
 CxList permissionGPS = androidPermission.FindByShortNames(new string[] {
-       "*android.permission.ACCESS_FINE_LOCATION*",
-       "*android.permission.ACCESS_COARSE_LOCATION*"}, false);
+       "ACCESS_FINE_LOCATION*",
+       "ACCESS_COARSE_LOCATION*"}, false);


 CxList usingGPS = All.NewCxList(
    All.FindByShortNames(new string[]{ "*LocationManager*", "*LocationListener*" }, false),
@@ -81,8 +80,8 @@
 // Application Requiered Contacts access but not uses it


 CxList permissionContacts = All.NewCxList(androidPermission.FindByShortNames(new string[]{
-   "*android.permission.READ_CONTACTS*",
-   "*android.permission.WRITE_CONTACTS*" }, false));
+   "READ_CONTACTS*",
+   "WRITE_CONTACTS*" }, false));


 CxList usingContacts = All.NewCxList(All.FindByShortName("*ContactsContract*"),
    All.FindByNames(new string [] {"*android.provider.CallLog*","*Contacts.People*",
@@ -94,12 +93,12 @@
 }


 // Application Required to be able to disable the device (very dangerous!).
-CxList permissionBrick = androidPermission.FindByShortName("*android.permission.BRICK*", false);
+CxList permissionBrick = androidPermission.FindByShortName("BRICK*", false);
 result.Add(permissionBrick);
```

```
 // Application Requiered access to write on external storage but not uses it
 CxList permissionExternalStorage =
-    androidPermission.FindByShortName("*android.permission.WRITE_EXTERNAL_STORAGE*", false);
+    androidPermission.FindByShortName("WRITE_EXTERNAL_STORAGE*", false);

 CxList usingExternalStorage = All.FindByShortName(@"*/sdcard/*");
 usingExternalStorage.Add(methods.FindByMemberAccess("Environment.getExternalStorageDirectory"));
@@ -108,7 +107,7 @@
     result.Add(permissionExternalStorage);
 }
 // Application Requiered access to use camera but not uses it
-CxList permissionCamera = androidPermission.FindByShortName("*android.permission.CAMERA*", false);
+CxList permissionCamera = androidPermission.FindByShortName("CAMERA*", false);
 CxList usingCamera = methods.FindByMemberAccess("*Camera.open*");
 if ((permissionCamera.Count > 0) && (usingCamera.Count == 0))
 {
@@ -116,7 +115,7 @@
 }

 // Application Requiered access to Record Audio
-CxList permissionRecordAudio = androidPermission.FindByShortName("*android.permission.RECORD_AUDIO*", false);
+CxList permissionRecordAudio = androidPermission.FindByShortName("RECORD_AUDIO*", false);
 CxList usingMicrophone = All.GetParameters(All.FindByName("*MediaRecorder.AudioSource.MIC*"), 0);
 if ((permissionRecordAudio.Count > 0) && (usingMicrophone.Count == 0)){
@@ -124,7 +123,7 @@
 }

 // Application Requiered ACCESS_NETWORK_STATE access but not uses it
-CxList networkState = androidPermission.FindByShortName("*android.permission.ACCESS_NETWORK_STATE*", false);
+CxList networkState = androidPermission.FindByShortName("ACCESS_NETWORK_STATE*", false);
 CxList connectivityManager = typeRef.FindByType("ConnectivityManager");

 if ((networkState.Count > 0) && (connectivityManager.Count == 0)){
@@ -132,7 +131,7 @@
 }

 // NFC
-CxList permissionNfc = androidPermission.FindByName("\"android.permission.NFC\"");
+CxList permissionNfc = androidPermission.FindByShortName("NFC");

 CxList ndefMsgCallback = All.FindByTypes(new string[] { "CreateNdefMessageCallback", "NfcAdapter" });

@@ -140,7 +139,7 @@
     result.Add(permissionNfc);

 // Manage accounts (add / remove / change credentials)
```

```
-CxList permissionManageAccounts = androidPermission.FindByName("\"android.permission.MANAGE_ACCOUNTS\"");
+CxList permissionManageAccounts = androidPermission.FindByShortName("MANAGE_ACCOUNTS");

 CxList manageAccounts = All.NewCxList(
    methods.FindByMemberAccesses("AccountManager", new string[]{
@@ -156,7 +155,7 @@
    result.Add(permissionManageAccounts);

 // Change configuration
-CxList permissionChangeConfig = androidPermission.FindByName("\"android.permission.CHANGE_CONFIGURATION\"");
+CxList permissionChangeConfig = androidPermission.FindByName("CHANGE_CONFIGURATION");
 CxList changesConfig = methods.FindByMemberAccess("Configuration.set*");
 if(permissionChangeConfig.Count > 0 && changesConfig.Count == 0)
 {
```

Java / Java_Android / Insecure_Android_SDK_Version

Code changes

```
---
+++
@@ -1 +1,5 @@
-result = Common_Android.Insecure_Android_SDK_Version();
+// Pattern to search from 0 to 16 which are considered not safe version
+string unsafeVersionPattern = "\\b(?:[0-9]|1[0-6])\\b";
+
+result = cxXPath.FindXmlAttributesByNameAndValue("*AndroidManifest.xml", 2,
+   "minSdkVersion", unsafeVersionPattern, true);
```

Java / Java_Android / No_Installer_Verification_Implemented

Code changes

```
---
+++
@@ -8,7 +8,15 @@
 CxList ifStatement = installerPackageName.GetAncOfType<IfStmt>();
 ifStatement.Add(installerPackageName.GetAncOfType<TernaryExpr>());

+string query = "//activity/@*[namespace-uri()='http://schemas.android.com/apk/res/android' and local-name()='name']";
 if(ifStatement.Count == 0)
-{
-   result = Find_Android_Settings().FindByMemberAccess("ACTIVITY.ANDROID_NAME");
+{
+   foreach (CxXmlDoc doc in cxXPath.GetXmlFiles("*AndroidManifest.xml"))
+   {
+       XPathNodeIterator nodeIterator = doc.CreateNavigator().Select(query);
+       while (nodeIterator.MoveNext())
+       {
+           result.Add(cxXPath.CreateXmlNode(nodeIterator.Current.Clone(), doc, 2, false));
+       }
+   }
```

```
 }
```

Java / Java_Android / Use_of_WebView_AddJavascriptInterface

Code changes

```
---
+++
@@ -18,10 +18,11 @@
    }
 }
 else{
-   // Find sdk version in Manifest.xml files
-   CxList sdkVersionVar = Find_Android_Settings().GetByAncs(All.FindByName("MANIFEST.USES_SDK.ANDROID_MINSDKVERSION"));
-   CxList SdkVersionVal = Find_Strings().GetByAncs(sdkVersionVar.GetAncOfType<AssignExpr>());
-   isInt = int.TryParse(SdkVersionVal.GetName(), out sdkVersion);
+   // Find sdk version in Manifest.xml files
+   CxList unsafeSdkVersion = Insecure_Android_SDK_Version();
+   if(unsafeSdkVersion.Count > 0){
+       isInt = true;
+   }
 }

 if(!isInt || sdkVersion < 17) // addJavascriptInterface vulnerability is fixed in API 17 and above
```

Java / Java_Android / Weak_Encryption

Code changes

```
---
+++
@@ -4,18 +4,12 @@
 // provider encryption default for AES
 // The query looks for use of DES or AES with ECB block encryption
 ////////////////////////////////////////////////////////////////
-CxList unknowRef = Find_UnknownReference();
+CxList strings = Find_Strings();

-CxList parameters = All.NewCxList(Find_Strings(), unknowRef);
+CxList parameters = All.NewCxList(strings, Find_MemberAccesses(), Find_UnknownReference());
 CxList cipherGetInstance = Find_Methods().FindByMemberAccess("Cipher.getInstance");
 CxList encryptionAlgorithm = parameters.GetParameters(cipherGetInstance, 0);
+CxList encryptionStrings = strings.FindByShortNames(new string[]{ "AES", "AES/ECB*", "DES*" });

-CxList assigner = All.NewCxList(Find_Declarators(), unknowRef);
-encryptionAlgorithm.Add(assigner.FindAllReferences(encryptionAlgorithm).GetAssigner());
-CxList encryptionStrings = encryptionAlgorithm.FindByType<StringLiteral>();
-
-result = encryptionStrings.FindByShortNames(new string[]{
-   "AES",
-   "AES/ECB*",
-   "DES*"
```

```
-});
+result = encryptionAlgorithm * encryptionStrings;
+result.Add(encryptionAlgorithm.DataInfluencedBy(encryptionStrings));
```

Java / Java_Best_Coding_Practice / Hardcoded_Absolute_Path

Code changes

```
---
+++
@@ -1 +1,7 @@
 result = Common_Best_Coding_Practice.Hardcoded_Absolute_Path();
+string[] xmlFiles = new []{"*dwr.xml","*pom.xml","*AndroidManifest.xml",
+       "*build.xml","*web.xml","*weblogic.xml","*weblogic.xml"};
+
+foreach(string file in xmlFiles){
+   result.Add(cxXPath.FindXmlAttributesByValue(file, 2, "(c|C|d|D):\\.*", true));
+}
```

Java / Java_High_Risk / Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -2,7 +2,7 @@
 CxList methods = Find_Methods();
 CxList methodDecls = Find_MethodDecls();
-CxList inputs = Find_Interactive_Inputs_NoRemote();
+CxList inputs = Find_Interactive_Inputs();
 inputs -= Find_Properties_Input();

 CxList findAttrMembers = methods.FindByMemberAccess("pagecontext.findAttribute").GetMembersOfTarget();
@@ -52,3 +52,7 @@
 CxList getRequestSessionMethods = Find_GET_Request_Session_Methods();
 //Removing results that come from stored sessions
 result -= result.IntersectWithNodes(getRequestSessionMethods);
+
+// When the flow goes through a "Remote input" (a request to a remote server) it is not vulnerable
+CxList remoteInputs = All.NewCxList(Find_Remote_ReadMethods(), Find_Remote_Connections());
+result -= result.IntersectWithNodes(remoteInputs);
```

Java / Java_High_Risk / Stored_XSS

Code changes

```
---
+++
@@ -26,3 +26,7 @@

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

```
+// When the flow goes through a "Remote input" (a request to a remote server) it is not vulnerable
+CxList remoteInputs = All.NewCxList(Find_Remote_ReadMethods(), Find_Remote_Connections());
+result -= result.IntersectWithNodes(remoteInputs);
```

Java / Java_Low_Visibility / Improper_Resource_Access_Authorization

Code changes

```
---
+++
@@ -113,7 +113,7 @@
    foreach (CxList tmp in small)
    {
        CSharpGraph sg = tmp.GetFirstGraph();
-       int fileId = sg.LinePragma.GetFileId();
+       int fileId = sg.LinePragma.FileId;
        CxList cxList;
        if (!smallDic.TryGetValue(fileId, out cxList))
        {
@@ -130,7 +130,7 @@
    foreach (CxList tmp1 in big)
    {
        CSharpGraph sg1 = tmp1.GetFirstGraph();
-       int fileId1 = sg1.LinePragma.GetFileId();
+       int fileId1 = sg1.LinePragma.FileId;
        CxList cxList1;
        if (!bigDic.TryGetValue(fileId1, out cxList1))
        {
```

Java / Java_Low_Visibility / Log_Forging

Code changes

```
---
+++
@@ -22,3 +22,7 @@
 result = log.InfluencedByAndNotSanitized(inputs, sanitize);
 // only get results that intersect with the log output parameters
 result = result.IntersectWithNodes(All.GetByAncs(logParams));
+
+// When the flow goes through a "Remote input" (a request to a remote server) it is not vulnerable
+CxList remoteInputs = All.NewCxList(Find_Remote_ReadMethods(), Find_Remote_Connections());
+result -= result.IntersectWithNodes(remoteInputs);
```

Java / Java_Low_Visibility / Open_Redirect

Code changes

```
---
+++
@@ -5,3 +5,7 @@
 sanitize.Add(Find_Read_NonDB(), Find_ObjectCreations().FindByShortName("File*"));


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

```
+
+// When the flow goes through a "Remote input" (a request to a remote server) it is not vulnerable
+CxList remoteInputs = All.NewCxList(Find_Remote_ReadMethods(), Find_Remote_Connections());
+result -= result.IntersectWithNodes(remoteInputs);
```

Java / Java_Low_Visibility / Sensitive_Cookie_in_HTTPS_Session_Without_Secure_Attribute

Code changes

```
---
+++
@@ -5,11 +5,12 @@
 CxList trues = All.FindByShortName("true");
 CxList secured = trues.GetParameters(setSecure);

-CxList webFiles = Find_Web_Files();
 // Verify if in the webFiles (*.xml) the fields in the session-config more precisely
 //the http-only and the secure fields are setted true
-if ( webFiles.FindByName("WEB_APP.SESSION_CONFIG.COOKIE_CONFIG.HTTP_ONLY.TEXT").GetAssigner().FindByShortName("true").Count == 0 ||
-   webFiles.FindByName("WEB_APP.SESSION_CONFIG.COOKIE_CONFIG.SECURE.TEXT").GetAssigner().FindByShortName("true").Count == 0)
+CxList secureFields = cxXPath.FindXmlNodesByLocalNameAndValue("*web.xml", 2, "http-only", "true");
+secureFields.Add(cxXPath.FindXmlNodesByLocalNameAndValue("*web.xml", 2, "secure", "true"));
+
+if ( secureFields.Count == 0)
 {
    // Find the added cookies
    CxList addCookie = methods.FindByMemberAccess("response.addCookie");
```

Java / Java_Low_Visibility / Trust_Boundary_Violation_in_Session_Variables

Code changes

```
---
+++
@@ -1,6 +1,16 @@
 CxList input = Find_Interactive_Inputs();

-CxList setAttr = Find_SetAttribute_Implicit_Objects();
+CxList setAttr = Find_Implicit_Object_Members().FindByMemberAccesses(new string[]{"Session", "HttpSession"}, new string[]{"setAttribute", "putValue"});
+
+CxList invocations = Find_Methods();
+string[] relevantNames = new string[] {
+   "*session.setAttribute",
+   "*session.putValue"
+   };
+
+setAttr.Add(invocations.FindByMemberAccesses(new string[] { "session.putValue" }),
+   invocations.FindByNames(relevantNames, false));
+
 CxList setAttrParams = All.GetParameters(setAttr);

 CxList sanitizers = Find_General_Sanitize();
```

```
@@ -8,3 +18,7 @@


 result = setAttrParams.InfluencedByAndNotSanitized(input, sanitizers);
 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
+
+// When the flow goes through a "Remote input" (a request to a remote server) it is not vulnerable
+CxList remoteInputs = All.NewCxList(Find_Remote_ReadMethods(), Find_Remote_Connections());
+result -= result.IntersectWithNodes(remoteInputs);
```

Java / Java_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -49,6 +49,11 @@
 CxList strEQ = strLiterals.GetMembersOfTarget().FindByShortName("equals");
 strEQ = psw.GetByAncs(strEQ);
 equalsPassword.Add(strEQ);
+
+//Find passwords in maps
+CxList mapPut = methods.FindByMemberAccess("Map.put");
+CxList mapParam2 = stringLiterals.GetParameters(mapPut, 1);
+CxList mapPassword = stringLiterals.GetParameters(mapParam2.GetAncOfType<MethodInvokeExpr>(), 0) * passwordString;


 // Find password in assignments
 CxList assignPassword = pswInLSide.GetAncOfType<AssignExpr>();
@@ -137,11 +142,9 @@
 CxList setPropWithPass = passAndStrings.GetByAncs(setProp);
 result.Add(stringLiterals.GetParameters(setPropWithPass.GetAncOfType<MethodInvokeExpr>(), 1));


-// ANT build file
-result.Add(passwordString.FindByAssignmentSide(CxList.AssignmentSide.Right).FindByFileName("*build.xml"));
-
 // All
 result.Add(equalsPassword,
+           mapPassword,
            assignPassword,
            paramsAffectedByString,
            hardcodedPasswordInMethod,
```

Java / Java_Low_Visibility / Use_Of_Hardcoded_Password_In_Config

Code changes

```
---
+++
@@ -7,6 +7,24 @@
 // Sensitive information inside these kinds of files should be caught.
 CxList androidPasswordsInXML = Find_Android_Hardcoded_Password_In_Xml();


+List<string> pswdIncludeList = new List<string> {
```

```
+       "*password*", "*psw", "psw*", "pwd*", "*pwd", "*authKey*", "pass*", "cipher*", "*cipher", "*pass", "adgangskode",
+       "benutzerkennwort", "chiffre", "clave", "codewort", "contrasena", "contrasenya", "geheimcode", "geslo", "heslo",
+       "jelszo", "kennwort", "losenord", "losung", "losungswort", "lozinka", "modpas", "motdepasse", "parol", "parola",
+       "parole", "pasahitza", "pasfhocal", "passe", "passord", "passwort", "pasvorto", "paswoord", "salasana",
+       "schluessel", "schluesselwort", "senha", "chave", "cifra", "sifre", "wachtwoord", "wagwoord", "watchword",
+       "zugangswort", "parolachiave", "parola chiave", "parolechiavi", "parole chiavi", "paroladordine",
+       "verschluesselt", "sisma"};
+
+List<string> pswdExcludeList = new List<string>{
+       "*passable*", "*passage*", "*passenger*", "*passer*", "*passing*", "*passion*", "*passive*",
+       "*passover*", "*passport*", "*passed*", "*compass*", "*bypass*", "pass-through", "passthru", "passthrough",
+       "passbytes", "passcount", "passratio", "err_pass*"};
+
+// ANT build file
+CxList builds = cxXPath.FindXmlAttributesByValue("*build.xml", 2, "^(?!\\$\\{).*", true);
+CxList passInBuild = builds.FindByShortNames(pswdIncludeList);
+passInBuild -= passInBuild.FindByShortNames(pswdExcludeList);


 // Remove upper case string
 // Example : PASSWORD
@@ -33,3 +51,4 @@


 result = (smallPassword * strLiterals).GetAssignee();
 result.Add(androidPasswordsInXML);
+result.Add(passInBuild);
```

Java / Java_Medium_Threat / Parameter_Tampering

Code changes

```
---
+++
@@ -14,3 +14,7 @@


 result = db.InfluencedByAndNotSanitized(input, sanitize);
 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+// When the flow goes through a "Remote input" (a request to a remote server) it is not vulnerable
+CxList remoteInputs = All.NewCxList(Find_Remote_ReadMethods(), Find_Remote_Connections());
+result -= result.IntersectWithNodes(remoteInputs);
```

Java / Java_Medium_Threat / Plaintext_Storage_of_a_Password

Code changes

```
---
+++
@@ -13,7 +13,9 @@
 CxList proploadfromfile = propload.FindByParameters(fileinputstr);
 //Included gettargetofmembers because there is no flow from load to its target
 CxList inputs = Find_FileStreams();
-inputs.Add(proploadfromfile, proploadfromfile.GetTargetOfMembers());
```

```
+inputs.Add(proploadfromfile,
+   proploadfromfile.GetTargetOfMembers(),
+   Find_Cloud_Outputs());

 CxList sanitize = Find_General_Sanitize();
```

Java / Java_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -87,7 +87,8 @@
 CxList outputs = All.NewCxList(
    Find_Outputs(),
    exceptions,
-   exceptionsCtorsWithSuper
+   exceptionsCtorsWithSuper,
+   Find_Cloud_Outputs()
 );

 // Define sanitize
```

Java / Java_Medium_Threat / Use_of_a_One_Way_Hash_without_a_Salt

Code changes

```
---
+++
@@ -4,6 +4,13 @@

 CxList sinks = Find_Digest_Commands();

+// Remove sinks influenced by a SecureRandom
+CxList methods = Find_Methods();
+CxList safeRandom = methods.FindByMemberAccesses(new string[] {"SecureRandom.next*"});
+CxList safeAsRandom = Find_ObjectCreations().FindByType("SecureRandom").GetAssignee().FindByType("Random");
+safeRandom.Add(unkRefs.FindAllReferences(safeAsRandom));
+sinks -= sinks.DataInfluencedBy(Find_Param().CxSelectDomProperty<Param>(x => x.Value).GetParameters(safeRandom));
+
 CxList possibleSalts = unkRefs.FindByShortNames(new string[]{"*salt*", "*nonce*"});
 CxList sanitizers = sinks.DataInfluencedBy(possibleSalts).GetLastNodesInPath();
 sanitizers.Add(Find_Password_Hash_Sanitize());
```

Java / Java_Struts / Struts_Unvalidated_Action_Form

Code changes

```
---
+++
@@ -1,40 +1,34 @@
 // Test struts version
 if(Find_Struts1_Presence().Count > 0)
```

```
-{
-
-   CxList strings = Find_Strings();
-   CxList formBeans = All.NewCxList();
-   CxList validationForms = All.NewCxList();
+{
+   result = All.NewCxList();

-   Func<string, string, CxList, CxList> getXmlAttributes = (fileName, xpath, strings) => {
-
-       CxList xmlAttributes = All.NewCxList();
-
-       foreach (CxXmlDoc doc in cxXPath.GetXmlFiles(fileName)){
-           XPathNodeIterator nodeIterator = doc.CreateNavigator().Select(xpath);
-
-           while (nodeIterator.MoveNext()){
-               string field = nodeIterator.Current.ToString();
-               if(!string.IsNullOrEmpty(field)){
-                   int lineNumber = int.Parse(((IXmlLineInfo) nodeIterator.Current).LineNumber.ToString());
-                   xmlAttributes.Add(strings.FindByShortNames(field).FindByPosition(lineNumber));
-               }
-           }
-       }
-       return xmlAttributes;
-   };
-
-   formBeans = getXmlAttributes("*struts-config.xml", "//struts-config/form-beans/form-bean/@name", strings);
-   validationForms = getXmlAttributes("*validation.xml", "//form-validation/formset/form/@name", strings);
-
-   foreach (CxList formBean in formBeans)
-   {
-       StringLiteral str = formBean.TryGetCSharpGraph<StringLiteral>();
-       string strName = str.ShortName.Trim(new char[] {'"'});
-
-       if (validationForms.FindByShortName('"' + strName + '"').Count == 0)
-       {
-           result.Add(str.NodeId, str);
-       }
+   List<string> validFormNames = new List<string>();
+   foreach (CxXmlDoc doc in cxXPath.GetXmlFiles("*validation.xml")){
+       XPathNodeIterator nodeIterator = doc.CreateNavigator()
+           .Select("//form-validation/formset/form");
+
+       while (nodeIterator.MoveNext()){
+           string formName = nodeIterator.Current.GetAttribute("name", "");
+           if(!string.IsNullOrEmpty(formName)){
+               validFormNames.Add(formName);
+           }
```

```
+       }
+   }
+
+   foreach (CxXmlDoc doc in cxXPath.GetXmlFiles("*struts-config.xml")){
+       XPathNodeIterator nodeIterator = doc.CreateNavigator()
+           .Select("//struts-config/form-beans/form-bean/@name");
+
+       while (nodeIterator.MoveNext()){
+           XPathNavigator bean = nodeIterator.Current.Clone();
+           string beanName = bean.ToString();
+           if(!string.IsNullOrEmpty(beanName) && !validFormNames.Contains(beanName)){
+               int lineNumber = int.Parse(((IXmlLineInfo) nodeIterator.Current).LineNumber.ToString());
+               result.Add(cxXPath.FindXmlAttributesByNameAndValue("*struts-config.xml", 2, "name", beanName)
+                   .FindByPosition(lineNumber)
+                   .FindByFileName("*WEB-INF*"));
+           }
+       }
    }
 }
```

JavaScript / JavaScript_Low_Visibility / Client_JQuery_Deprecated_Symbols

Code changes

```
---
+++
@@ -1,12 +1,12 @@
 // Find all instances of JQuery objects use.
-CxList JQueryObj = All.FindByShortNames(new List<string>{"jquery","$"}, false);
+CxList JQueryObj = All.FindByShortNames(new string[]{"jquery","$"}, false);
 CxList jqMethods = Find_JQuery_Methods();
 CxList Methods = Find_All_JQuery_Methods_Including_Aliases(jqMethods, jqMethods, 0);
 CxList selectors = All.GetParameters(JQueryObj, 0);

 // List of deprecated methods
 // https://api.jquery.com/category/deprecated/
-List<string> deprecatedMethodNames = new List<string> {
+string[] deprecatedMethodNames = new string[] {
     // Deprecated in 1.3
     "boxModel",
     "browser",
@@ -49,24 +49,8 @@

 CxList deprecatedMethods = Methods.FindByShortNames(deprecatedMethodNames);

-// List of deprecated selectors
-// https://bugs.jquery.com/ticket/9400
-List<string> deprecatedSelectorNames = new List<string> {
-   ":button",
-   ":checkbox",
```

```
-   ":file",
-   ":image",
-   ":input",
-   ":password",
-   ":radio",
-   ":submit",
-   ":text",
-   ":reset"
-};
-CxList deprecatedSelectors = selectors.FindByShortNames(deprecatedSelectorNames);
-
 // Selectors Deprecated 3.4
-List<string> selectorsDeprecated = new List<string> {
+string[] selectorsDeprecated = new string[] {
        "*:eq*",
        "*:even",
        "*:first",
@@ -102,6 +86,5 @@
 deprecatedMethods -= nonDeprecatedLoadMethod;

 result.Add(deprecatedMethods,
-       deprecatedSelectors,
        selecDeprecated,
        toggleDeprecatedUsage);
```

JavaScript / JavaScript_Medium_Threat / Frameable_Login_Page

Code changes

```
---
+++
@@ -15,12 +15,14 @@
 CxList parameters = Find_Param();
 CxList fieldDecls = Find_FieldDecls();

-CxList loginRequestPath = strings.FindByShortNames(new List<string>{"*login*", "*auth*", "*signin*"});
+CxList loginRequestPath = strings.FindByShortNames(new string[]{"*login*", "*auth*", "*signin*"});
 CxList methodsWithAuthRoute = methods.FindByParameters(loginRequestPath);
+methodsWithAuthRoute -= methodsWithAuthRoute.FindByShortName("post", false); //POSTs are not valid inputs
 CxList exportsMethods = All.FindByShortName("cxExports*").GetMembersOfTarget().GetAssigner();

 // sanitizers
-CxList safeXFrameValue = strings.FindByShortNames(new List<string>{"*deny*", "*sameorigin*", "*allow-from*", "*allow\\-from*"}, false);
+CxList safeXFrameValue = strings.FindByShortNames(
+   new string[]{"*deny*", "*sameorigin*", "*allow-from*", "*allow\\-from*"}, false);
 CxList xFrameOptions = strings.FindByShortName("*x-frame-options*", false);
 CxList sanitizers = All.NewCxList();

@@ -40,27 +42,28 @@
 responseParameter.Add(responseHapi);
```

```
 CxList funParamsInMethWithAuthRoute = methodDecls.FindByFathers(parameters.GetParameters(methodsWithAuthRoute));
-CxList funParamsExportMethods = All.NewCxList();
-funParamsExportMethods.Add(funParamsInMethWithAuthRoute, exportsMethods);
+CxList funParamsExportMethods = All.NewCxList(funParamsInMethWithAuthRoute, exportsMethods);
 CxList riskyResponseParams = responseParameter.GetParameters(funParamsExportMethods);

 CxList methodsWithResponse = methodDecls * responseParameter.GetFathers().GetFathers();
 CxList methodsWithRequestAndResponse = methodsWithResponse * requestParameter.GetFathers().GetFathers();
-CxList targetsOfResponses = unknownRefs.FindAllReferences(responseParameter.GetParameters(methodsWithRequestAndResponse)).GetMembersOfTarget();
+CxList targetsOfResponses = unknownRefs.FindAllReferences(
+   responseParameter.GetParameters(methodsWithRequestAndResponse)).GetMembersOfTarget();

 // sink nodes are the nodes that send the response
 // catch the most possible
-CxList sendResponses = targetsOfResponses.FindByShortNames(new List<string>{"send","end"});
-CxList riskySendResponses = All.NewCxList();
-riskySendResponses.Add(sendResponses);
+CxList sendResponses = targetsOfResponses.FindByShortNames(new string[]{"send","end"});
+CxList riskySendResponses = All.NewCxList(sendResponses);

-CxList riskyLoginHapiResponses = paramDecls.GetParameters(loginHapiHandlers - sendResponses.GetByAncs(loginHapiHandlers), 1);
+CxList riskyLoginHapiResponses = paramDecls.GetParameters(
+   loginHapiHandlers - sendResponses.GetByAncs(loginHapiHandlers), 1);
 // Both response.setHeader(headername, headervalue),
 // response.append(headername, headervalue) and set(headername, headervalue),
 // allow setting a single header value.
-CxList appendingHeaderMethods = targetsOfResponses.FindByShortNames(new List<string>{"setHeader", "append", "set"});
-CxList safeResponseHeader = appendingHeaderMethods.FindByParameters(xFrameOptions).FindByParameters(safeXFrameValue).GetTargetOfMembers();
+CxList appendingHeaderMethods = targetsOfResponses.FindByShortNames(new string[]{"setHeader", "append", "set"});
+CxList safeResponseHeader = appendingHeaderMethods.FindByParameters(xFrameOptions)
+   .FindByParameters(safeXFrameValue).GetTargetOfMembers();
 sanitizers.Add(safeResponseHeader);
 riskyResponseParams -= riskyResponseParams.FindAllReferences(safeResponseHeader);
 riskySendResponses -= riskySendResponses.FindAllReferences(safeResponseHeader);
@@ -94,15 +97,16 @@
 CxList actionFields = fieldDecls.FindByShortName("action");
 CxList safeActionFields = safeXFrameValue.GetByAncs(actionFields).GetAncOfType<FieldDecl>();
 CxList unsafeActionFields = actionFields - safeActionFields;
-CxList helmetFrameguardMethods = All.NewCxList();
-helmetFrameguardMethods.Add(helmetMethods, frameguardMethods);
-CxList helmetOrFrameguardMiddleware = helmetFrameguardMethods.GetByAncs(expressMembers).GetFathers().GetAncOfType<MethodInvokeExpr>();
-CxList unsafeMiddleware = unsafeActionFields.GetByAncs(helmetOrFrameguardMiddleware).GetAncOfType<MethodInvokeExpr>().GetFathers().GetAncOfType<MethodInvokeExpr>();
-CxList safeRouters = expressRefs.FindAllReferences((helmetOrFrameguardMiddleware - unsafeMiddleware).GetTargetOfMembers()).GetMembersOfTarget();
+CxList helmetFrameguardMethods = All.NewCxList(helmetMethods, frameguardMethods);
+CxList helmetOrFrameguardMiddleware = helmetFrameguardMethods.GetByAncs(expressMembers)
+   .GetFathers().GetAncOfType<MethodInvokeExpr>();
```

```
+CxList unsafeMiddleware = unsafeActionFields.GetByAncs(helmetOrFrameguardMiddleware)
+   .GetAncOfType<MethodInvokeExpr>().GetFathers().GetAncOfType<MethodInvokeExpr>();
+CxList safeRouters = expressRefs.FindAllReferences(
+   (helmetOrFrameguardMiddleware - unsafeMiddleware).GetTargetOfMembers()).GetMembersOfTarget();
 riskyResponseParams -= riskyResponseParams.GetByAncs(safeRouters);
 riskySendResponses -= riskySendResponses.GetByAncs(safeRouters);
 riskyLoginHapiResponses -= riskyLoginHapiResponses.GetByAncs(safeRouters);


-
 result = riskySendResponses.InfluencedByAndNotSanitized(riskyResponseParams, sanitizers);
 result.Add(riskyLoginHapiResponses);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / SSL_Verification_Bypass

Code changes

```
---
+++
@@ -1,11 +1,8 @@
-/*This query looks for "rejectUnauthorized" property set to false for the "https" and "tls" modules,
-the "requestCert" property set to true and the "rejectUnauthorized" property set to false
-for "express" and "hapi" servers and
-the "insecure" property set to true for the "request-promise" middleware*/
+/*This query looks for "rejectUnauthorized" property set to false for the "https" modules
+and the "insecure" property set to true for the "request-promise" middleware*/

 CxList imports = Find_Import() - XS_Find_All();
-CxList hapiAndExpress = All.NewCxList();
-CxList httpsAndtls = All.NewCxList();
+CxList https = All.NewCxList();
 CxList requestPromise = All.NewCxList();
 CxList httpVariants = All.NewCxList();


@@ -18,19 +15,15 @@


 foreach(CxList imp in imports){
     Import import = imp.TryGetCSharpGraph<Import>();
-    if(import.ImportedFilename.Contains("hapi") || import.ImportedFilename.Contains("express"))
-    {
-        hapiAndExpress.Add(imp);
-    }
-    if(import.ImportedFilename.Contains("https") || import.ImportedFilename.Contains("tls"))
+    if(import.ImportedFilename.Contains("https"))
     {
-        httpsAndtls.Add(imp);
+        https.Add(imp);
     }
     if(import.ImportedFilename.Contains("request-promise"))
     {
         requestPromise.Add(imp);
```

```
    }
-   if(import.ImportedFilename.Contains("https") || import.ImportedFilename.Contains("http")){
+   if(import.ImportedFilename.Contains("https")){
        httpVariants.Add(imp);
    }
 }
@@ -40,22 +33,12 @@
 CxList trueBoolean = booleanLiterals.FindByShortName("true");

 CxList rejectUnauthorizedProperty = fieldDelcs.FindByShortName("rejectUnauthorized");
-CxList requestCertProperty = fieldDelcs.FindByShortName("requestCert");
 CxList insecureProperty = fieldDelcs.FindByShortName("insecure");
 CxList insecureHttpParser = fieldDelcs.FindByShortName("insecureHTTPParser");

-CxList trueRequestCert = requestCertProperty.GetAssigner().FindByShortName("true");
 CxList falseReject = rejectUnauthorizedProperty.GetAssigner().FindByShortName("false");
+CxList httpsFiles = https.FindByFiles(falseReject);

-CxList trueReqCertFiles = hapiAndExpress.FindByFiles(trueRequestCert);
-CxList falseRejectFiles = hapiAndExpress.FindByFiles(falseReject);
-CxList hapiAndExpressFiles = trueReqCertFiles * falseRejectFiles;
-
-CxList httpsAndTlsFiles = httpsAndtls.FindByFiles(falseReject);
-CxList toExcludeRequestCertFiles = httpsAndTlsFiles.FindByFiles(requestCertProperty);
-httpsAndTlsFiles -= toExcludeRequestCertFiles;
-
-result.Add(rejectUnauthorizedProperty.FindByFiles(hapiAndExpressFiles).DataInfluencedBy(falseBoolean),
-   rejectUnauthorizedProperty.FindByFiles(httpsAndTlsFiles).DataInfluencedBy(falseBoolean),
+result.Add(rejectUnauthorizedProperty.FindByFiles(httpsFiles).DataInfluencedBy(falseBoolean),
     insecureProperty.FindByFiles(requestPromise).DataInfluencedBy(trueBoolean),
     insecureHttpParser.FindByFiles(httpVariants).DataInfluencedBy(trueBoolean));
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Use_of_Broken_or_Risky_Cryptographic_Algorithm

Code changes

```
---
+++
@@ -5,12 +5,6 @@
 CxList allInfluByRequireCrypto = methodInvoke.DataInfluencedBy(cryptoRequires);
 //All methodInvoke influenced by required library
 CxList methInvFromRequire = allInfluByRequireCrypto * methodInvoke;
-
-//PBKDF2 applies pseudorandom function HMAC-SHA1 to derive a key of given length from the given password, salt and iterations
-CxList pbkdf2Meth = methInvFromRequire.FindByMemberAccesses(new string[]{
-   "*.pbkdf2",
-   "*.pbkdf2Sync"
-});

 //support MD5 MD2 MD4 SHA1
```

```
 CxList hashStrings = strings.FindByShortNames(
@@ -38,7 +32,6 @@
 CxList cryptoMethods = methInvFromRequire.FindByMemberAccesses(cryptoMethodsList);


 result.Add(
-   pbkdf2Meth,
-   Find_Encrypt(),
+   Find_Unsafe_Encrypt(),
    cryptoMethods.DataInfluencedBy(hashStrings),
    createCipherMembers.DataInfluencedBy(badCiphers));
```

JavaScript / JavaScript_Vue / Declaration_of_Multiple_Vue_Components_per_File

Code changes

```
---
+++
@@ -17,12 +17,12 @@
        try
        {
            CSharpGraph comp = component.GetFirstGraph();
-           if(fileIds.Contains(comp.LinePragma.GetFileId())) {
+           if(fileIds.Contains(comp.LinePragma.FileId)) {
                multiComponentFiles.Add(component);
            }
            else
            {
-               fileIds.Add(comp.LinePragma.GetFileId());
+               fileIds.Add(comp.LinePragma.FileId);
            }
        }
        catch
```

JavaScript / JavaScript_Vue / Inconsistent_Component_Top_Level_Elements_Ordering

Code changes

```
---
+++
@@ -22,7 +22,7 @@
            try
            {
                CSharpGraph comp = component.GetFirstGraph();
-               int fileId = comp.LinePragma.GetFileId();
+               int fileId = comp.LinePragma.FileId;

                if(!fileToNodes.ContainsKey(fileId))
                {
```

JavaScript / JavaScript_XS / XS_CSRF

Code changes

```
---
+++
@@ -78,7 +78,7 @@
                        string accessFolderName = accessFileName.Remove(accessFileName.LastIndexOf(cxEnv.Path.DirectorySeparatorChar));
                        if(originalFileFolder.Contains(accessFolderName))
                        {
-                           xsaccessMappingDictionary[name].Add(g.LinePragma.GetFileId());
+                           xsaccessMappingDictionary[name].Add(g.LinePragma.FileId);
                        }
                    }
                    lastIndexOfBackSlash = father.LastIndexOf(cxEnv.Path.DirectorySeparatorChar);
```

Python / Python_High_Risk / Code_Injection

Code changes

```
---
+++
@@ -3,7 +3,9 @@
 CxList dynamicMethodInvoke = Find_By_Short_Names_With_Refs(dynamicMethods);
 dynamicMethodInvoke.Add(methods.FindByMemberAccess("os.popen"));

-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Interactive_Inputs());
+
 CxList systemJs = All.FindByShortName("system_js");
 CxList systemJsMembers = systemJs.GetMembersOfTarget();
 CxList systemJsMemberAccess = systemJsMembers.FindByType(typeof(MemberAccess)).InfluencedBy(inputs);
```

Python / Python_High_Risk / Command_Injection

Code changes

```
---
+++
@@ -1,19 +1,34 @@
 // sources
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
-
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Interactive_Inputs());
+CxList unkRef = Find_UnknownReference();
 CxList imports = Find_Imports();
 CxList inputsInLeftOfAssign = inputs.FindByAssignmentSide(CxList.AssignmentSide.Left);
 inputsInLeftOfAssign.Add(Find_MemberAccesses().GetByAncs(inputsInLeftOfAssign));
 inputs -= inputsInLeftOfAssign;
-
+
+// When the user input is not in the leftmost position at an array it is not vulnerable
+// to Command Injection ex. command = ["ls", "-l", user_input].
+CxList arrayInit = Find_ArrayInitializer();
```

```
+
+CxList firstValuesArray = arrayInit.CxSelectElements<ArrayInitializer>
+   (_ => _.InitialValues, 0).FindByType<UnknownReference>();
+CxList allValuesArray = arrayInit.CxSelectElements<ArrayInitializer>
+   (_ => _.InitialValues, -1);
+
+CxList invalidNodes = unkRef.GetByAncs(allValuesArray);
+CxList validNodes = unkRef.GetByAncs(firstValuesArray);
+
 // sanitizers
 CxList sanitize = Find_Command_Injection_Sanitize();


 // sinks
 CxList commands = Find_Command_Execution();
-
 String[] osMethods = new string[] {"input"};
 CxList osMethodsList = Find_Methods_By_Import("os", osMethods, imports);
 result = commands.InfluencedByAndNotSanitized(inputs, sanitize);


-result.Add(osMethodsList * inputs);
+CxList initialValuesValidInputs = result.IntersectWithNodes(validNodes);
+result -= result.IntersectWithNodes(invalidNodes);
+
+result.Add(osMethodsList * inputs, initialValuesValidInputs);
```

Python / Python_High_Risk / Connection_String_Injection

Code changes

```
---
+++
@@ -1,5 +1,6 @@
 CxList con = Find_DB_Connections();
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs()
+   , Find_Cloud_Interactive_Inputs());


 CxList sanitize = Find_Sanitize();
 sanitize.Add(Find_Integers());
```

Python / Python_High_Risk / LDAP_Injection

Code changes

```
---
+++
@@ -1,4 +1,5 @@
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Interactive_Inputs());
 CxList outputs = Find_LDAP_Inputs();
 CxList sanitize = Find_LDAP_Sanitize();
```

Python / Python_High_Risk / Local_File_Inclusion

Code changes

```
---
+++
@@ -1,5 +1,6 @@
 // user input -> unvalidated/unsanitized -> dynamic module import
-CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());
+CxList inputs = All.NewCxList(Find_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Inputs());
 CxList methods = Find_Methods();
 CxList customAtts = Find_CustomAttribute();
 CxList unkRefs = Find_UnknownReference();
```

Python / Python_High_Risk / OS_Access_Violation

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Inputs();
+CxList inputs = Find_Inputs() - Find_Invalid_Django_Injection_Inputs();
 CxList methods = Find_Methods();
 CxList unkrefs = Find_UnknownReference();
 List<string> osMethods = new List<string>{
```

Python / Python_High_Risk / Reflected_XSS_All_Clients

Code changes

```
---
+++
@@ -1,4 +1,5 @@
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Interactive_Inputs());
 CxList outputs = Find_XSS_Outputs();


 CxList sanitized = Find_XSS_Sanitize();
```

Python / Python_High_Risk / Resource_Injection

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList methods = Find_Methods();
 CxList socket = Find_Members("socket.bind", methods);
-CxList inputs = Find_Inputs();
+CxList inputs = Find_Inputs() - Find_Invalid_Django_Injection_Inputs();
 CxList paths = inputs.DataInfluencingOn(socket);
```

```
 result = paths.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Python / Python_High_Risk / SQL_Injection

Code changes

```
---
+++
@@ -1,5 +1,6 @@
 CxList db = Find_SQL_DB_In();
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Interactive_Inputs());
 CxList sanitized = Find_SQL_Sanitize();

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

Python / Python_High_Risk / Unsafe_Deserialization

Code changes

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

-CxList insecureMethods = Find_Methods_By_Import("pandas",new string[]{"read_pickle"});
+CxList insecureMethods = Find_Methods_By_Import("pandas", new string[]{"read_pickle"});

 List<string> pickleMethodNames = new List<string>{"load", "loads", "noload", "Unpickler"};
 CxList pickleMemberAccess = methods.FindByMemberAccess("pickle.*");
@@ -16,7 +16,8 @@
 CxList safeConditions = conditions.DataInfluencedBy(hash).GetLastNodesInPath();
 insecureMethods -= insecureMethods.GetByAncs(safeConditions.GetAncOfType<IfStmt>());

-CxList inputs = All.NewCxList(Find_Inputs(), Find_Read(), Find_Cloud_Inputs());
+CxList inputs = All.NewCxList(Find_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Read(), Find_Cloud_Inputs());
 inputs -= insecureMethods;

 CxList sanitizers = Find_Sanitize();
```

Python / Python_High_Risk / XPath_Injection

Code changes

```
---
+++
@@ -28,7 +28,8 @@

 CxList xPathParams = All.GetParameters(xPath);

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

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


 CxList sanitized = All.NewCxList(
    Find_Sanitize(),
```

Python / Python_Low_Visibility / Command_Argument_Injection

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 // sources
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs();
 inputs -= inputs.FindByAssignmentSide(CxList.AssignmentSide.Left);


 result = Find_Command_Argument_Injection(inputs);
```

Python / Python_Low_Visibility / Log_Forging

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs();
 CxList logs = Find_Log_Outputs();


 CxList sanitize = Find_Integers();
```

Python / Python_Low_Visibility / Marshmallow_Dumping_Without_Validation

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs();
 CxList methods = Find_Methods();


 CxList schemaClass = Find_Methods_By_Import("marshmallow", new string[]{"Schema"}).GetAncOfType<ClassDecl>();
```

Python / Python_Low_Visibility / Use_Of_Hardcoded_Password

Code changes

```
---
+++
@@ -10,6 +10,9 @@
 CxList strings = Find_Strings();
 CxList strLiterals = strings - emptyStringNull;
```

```
+//when the hardcoded string includes a space or dot we believe it is not a password string
+strLiterals -= strLiterals.FindByNames("* *", "*.*");
+
 // Find password in an initialization operation (declaration or assignment)
 CxList initializedPassword = psw.GetAssigner() * strLiterals;


@@ -21,7 +24,7 @@
 CxList equalsPassword = psw.GetFathers() * eq;
 equalsPassword = strLiterals.FindByFathers(equalsPassword);

-//Passwords as method parameter
+//Passwords in connection method parameter
 CxList methods = Find_Methods();
 CxList connection = methods.FindByShortName("*connect*", false);
 connection.Add(Find_DB_Conn_Strings());
@@ -45,20 +48,22 @@
 sanitize.Add(undefinedMethods);

 // Add the parameter itself, or whatever is influencing it
-CxList paramsAffectedByString = (connetionParam2 * strLiterals);
-paramsAffectedByString.Add(connetionParam2.InfluencedByAndNotSanitized(strLiterals, sanitize));
-paramsAffectedByString.Add((connetionParam1 * strLiterals));
-paramsAffectedByString.Add(connetionParam1.InfluencedByAndNotSanitized(strLiterals, sanitize));
-
-paramsAffectedByString *= psw;
+CxList paramsAffectedByString = All.NewCxList((connetionParam2 * strLiterals),
+   connetionParam2.InfluencedByAndNotSanitized(strLiterals, sanitize),
+   connetionParam1 * strLiterals,
+   connetionParam1.InfluencedByAndNotSanitized(strLiterals, sanitize)) * psw;


 CxList setPasswordMethod = methods.FindByShortName("setPassword", false);
 CxList passwordParams = strings.GetParameters(setPasswordMethod);
 CxList hardcodedPasswordInMethod = setPasswordMethod.DataInfluencedBy(passwordParams);


+CxList passwordAsMethodParameter = (Find_Param().GetParameters(methods) * psw).CxSelectDomProperty<Param>(_ => _.Value);
+CxList stringPasswordAsMethodParam = All.NewCxList((passwordAsMethodParameter * strLiterals),
+   passwordAsMethodParameter.InfluencedByAndNotSanitized(strLiterals, sanitize).GetLastNodesInPath());
+
 // pwds assigned to indexer refs
 CxList indRefPwds = psw.GetAncOfType<IndexerRef>().GetAssigner() * strLiterals;


 // All
-result.Add(initializedPassword, equalsPassword, paramsAffectedByString,
+result.Add(initializedPassword, equalsPassword, paramsAffectedByString, stringPasswordAsMethodParam,
     hardcodedPasswordInMethod, pwdInConnectioParam, indRefPwds);
```

Python / Python_Medium_Threat / CSRF

Code changes

```
---
+++
@@ -8,7 +8,7 @@

 CxList protectedContent = Find_Django_CSRF_Sanitize();

-CxList requests = Find_Interactive_Inputs();
+CxList requests = Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs();

 List<string> updatesMethodsList = new List<string>{"*update*", "*delete*","*insert*","*save*"};
```

Python / Python_Medium_Threat / Header_Injection

Code changes

```
---
+++
@@ -20,6 +20,6 @@
 requests.Add(requestsGet);
 requests.Add(Find_Header_Outputs());

-CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());
+CxList inputs = All.NewCxList(Find_Inputs() - Find_Invalid_Django_Injection_Inputs(), Find_Cloud_Inputs());

 result = requests.DataInfluencedBy(inputs);
```

Python / Python_Medium_Threat / Missing_HSTS_Header

Code changes

```
---
+++
@@ -1,3 +1,3 @@
-result = Common_Medium_Threat.Missing_HSTS_Header();
-result.Add(Find_Flask_Talisman_Missing_HSTS_Header());
-result.Add(Find_Django_Missing_HSTS_Header());
+result.Add(Common_Medium_Threat.Missing_HSTS_Header(),
+   Find_Flask_Talisman_Missing_HSTS_Header(),
+   Find_Django_Missing_HSTS_Header());
```

Python / Python_Medium_Threat / Object_Access_Violation

Code changes

```
---
+++
@@ -16,8 +16,10 @@
 CxList attrMethods = methods.FindByShortNames(attrMethodNames);

 //If the second argument of the attr methods is a string literal, is not vulnerable
-CxList attrMethodsSndParam = stringLiterals.GetParameters(attrMethods, 1);
-CxList sanitizedMethods = attrMethods.FindByParameters(attrMethodsSndParam);
+CxList attrMethodsSndParam = Find_Expressions().GetParameters(attrMethods, 1);
```

```
+CxList sanitizingParams = All.NewCxList(attrMethodsSndParam * stringLiterals,
+   stringLiterals.InfluencingOn(attrMethodsSndParam).GetLastNodesInPath().NotInfluencedBy(inputs));
+CxList sanitizedMethods = attrMethods.FindByParameters(sanitizingParams);
 attrMethods -= sanitizedMethods;


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

Python / Python_Medium_Threat / Open_Redirect

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());
+CxList inputs = All.NewCxList(Find_Inputs() - Find_Invalid_Django_Injection_Inputs(), Find_Cloud_Inputs());
 CxList redirects = Find_Redirects();


 CxList sanitizers = Find_Open_Redirect_Sanitizers(redirects);
```

Python / Python_Medium_Threat / Path_Traversal

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 CxList unknRefs = Find_UnknownReference();
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Bas_Server_Inputs(), Find_Cloud_Interactive_Inputs());
-CxList pandasSink = Find_Read_Pandas();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Bas_Server_Inputs(), Find_Cloud_Interactive_Inputs());CxList pandasSink = Find_Read_Pandas();
 CxList membersOfOsPath = unknRefs.FindByShortName("os")
     .GetMembersOfTarget().FindByShortName("path").GetMembersOfTarget();
 CxList osPathSinks = membersOfOsPath.FindByShortNames(new string[]{"exists", "isfile", "isdir"});
```

Python / Python_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -75,4 +75,3 @@


 // find all Personal Info that are influencing an output
 result = outputs.InfluencedByAndNotSanitized(personal_info, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
-result.Add(outputs * personal_info);
```

Python / Python_Medium_Threat / ReDoS_Injection

Code changes

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

```
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs();
 CxList sanitizers = Find_Sanitize();
 sanitizers.Add(Find_Base64_Encode());
 CxList regexPaternParams = Find_Regex_Pattern_Params();
```

Python / Python_Medium_Threat / SSRF

Code changes

```
---
+++
@@ -1,4 +1,5 @@
-CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList inputs = All.NewCxList(Find_Interactive_Inputs() - Find_Invalid_Django_Injection_Inputs(),
+   Find_Cloud_Interactive_Inputs());
 CxList requests = Find_Remote_Requests();

 // Removing XXE results as they are duplicates of Command_Injection query
```

Python / Python_Medium_Threat / Uncontrolled_Format_String

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());
+CxList inputs = All.NewCxList(Find_Inputs() - Find_Invalid_Django_Injection_Inputs(), Find_Cloud_Inputs());
 CxList sanitizers = Find_Integers();
 CxList prints = Find_Methods().FindByShortName("print");
```