Version 9.6.2 - Queries Release Notes

New Queries:

Language	Group	Name	CWE
Python	Python_AWS_Lambda	Unrestricted_Delete_S3	639
Python	Python_Best_Coding_Practice	Use_of_Unknown_Fields	0
Python	Python_Low_Visibility	Marshmallow_Dumping_Without_Validation	1173

Changed Queries:

Language	Group	Name	CWE	Changed Fields
Apex	Apex_Force_com_Code_Quality	Test_Assert_Without_Message	0	**Source** has changed
Apex	Apex_Force_com_Code_Quality	Test_Methods_With_No_Assert	0	**Source** has changed
CPP	CPP_Buffer_Overflow	Buffer_Overflow_Wrong_Buffer_Size	131	**Source** has changed
Go	Go_AWS_Lambda	Permission_Manipulation_In_S3	285	**Source** has changed
Go	Go_AWS_Lambda	Unrestricted_Read_S3	639	**Source** has changed
Go	Go_AWS_Lambda	Unrestricted_Write_S3	639	**Source** has changed
Go	Go_High_Risk	Stored_Command_Injection	77	**Source** has changed
Go	Go_High_Risk	Stored_XSS_All_Clients	79	**Source** has changed
Go	Go_Low_Visibility	Stored_Command_Argument_Injection	88	**Source** has changed
Go	Go_Medium_Threat	Stored_Absolute_Path_Traversal	36	**Source** has changed
Go	Go_Medium_Threat	Stored_Relative_Path_Traversal	23	**Source** has changed
Groovy	Groovy_Best_Coding_Practice	Exposure_of_Resource_to_Wrong_Sphere	493	**Source** has changed
Groovy	Groovy_Low_Visibility	Object_Hijack	491	**Source** has changed
Groovy	Groovy_Low_Visibility	Unsynchronized_Access_To_Shared_Data	567	**Source** has changed
Java	Java_AWS_Lambda	AWS_Credentials_Leak	200	**Source** has changed
Java	Java_AWS_Lambda	DynamoDB_NoSQL_Injection	74	**Source** has changed
Java	Java_AWS_Lambda	Permission_Manipulation_in_S3	285	**Source** has changed
Java	Java_AWS_Lambda	Unrestricted_Write_S3	639	**Source** has changed
Java	Java_Best_Coding_Practice	Exposure_of_Resource_to_Wrong_Sphere	493	**Source** has changed
Java	Java_Best_Coding_Practice	Potentially_Serializable_Class_With_Sensitive_Data	499	**Source** has changed
Java	Java_High_Risk	Unsafe_Reflection	470	**Source** has changed
Java	Java_Low_Visibility	Exposure_of_System_Data	497	**Source** has changed
Java	Java_Low_Visibility	Object_Hijack	491	**Source** has changed
Java	Java_Low_Visibility	Plaintext_Storage_in_a_Cookie	315	**Source** has changed
Java	Java_Low_Visibility	Unsynchronized_Access_To_Shared_Data	567	**Source** has changed
Java	Java_Medium_Threat	DoS_by_Sleep	834	**Source** has changed
Java	Java_Medium_Threat	Privacy_Violation	359	**Source** has changed
Java	Java_Spring	Spring_defaultHtmlEscape_Not_True	10711	**Source** has changed
Java	Java_Spring	Spring_Missing_HSTS_Header	346	**Source** has changed
JavaScript	JavaScript_High_Risk	Client_DOM_Stored_XSS	79	**Source** has changed
JavaScript	JavaScript_High_Risk	Client_DOM_XSS	79	**Source** has changed
JavaScript	Javascript_Kony	Kony_Reflected_XSS	79	**Source** has changed
JavaScript	Javascript_Kony	Kony_Stored_XSS	79	**Source** has changed
JavaScript	JavaScript_Medium_Threat	Unchecked_Input_For_Loop_Condition	606	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Cleartext_Storage_Of_Sensitive_Information	312	**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

Language	Group	Name	CWE	Changed Fields
JavaScript	JavaScript_Server_Side_Vulnerabilities	Password_Weak_Encryption	261	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Poor_Database_Access_Control	285	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Second_Order_SQL_Injection	89	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	Sensitive_Information_Over_HTTP	319	**Source** has changed
JavaScript	JavaScript_Server_Side_Vulnerabilities	SQL_Injection	89	**Source** has changed
JavaScript	JavaScript_AWS_Lambda	Race_Condition_Global_Scope	366	Name changed from Race_Condition_Concurrent_Instances to **Race_Condition_Global_Scope**
Kotlin	Kotlin_Medium_Threat	DoS_by_Sleep	834	**Source** has changed
PHP	PHP_High_Risk	Command_Injection	77	**Source** has changed
PHP	PHP_Low_Visibility	Error_Messages_Misconfiguration	209	**Source** has changed
PHP	PHP_Low_Visibility	Unsafe_Use_Of_Target_Blank	1022	**Source** has changed
PHP	PHP_Medium_Threat	Missing_Encryption_of_Sensitive_Data	311	**Source** has changed
Python	Python_AWS_Lambda	Unrestricted_Read_S3	639	**Source** has changed
Python	Python_AWS_Lambda	Unrestricted_Write_S3	639	**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	Reflected_XSS_All_Clients	79	**Source** has changed
Python	Python_High_Risk	Second_Order_SQL_Injection	89	**Source** has changed
Python	Python_High_Risk	SQL_Injection	89	**Source** has changed
Python	Python_High_Risk	Stored_XSS	79	**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_Medium_Threat	Communication_Over_HTTP	319	**Source** has changed
Python	Python_Medium_Threat	Cookie_Poisoning	472	**Source** has changed
Python	Python_Medium_Threat	Django_Missing_Object_Level_Authorization	862	**Source** has changed
Python	Python_Medium_Threat	Hardcoded_Password_in_Connection_String	547	**Source** has changed
Python	Python_Medium_Threat	Header_Injection	113	**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	Parameter_Tampering	472	**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_In_Replace	400	**Source** has changed
Python	Python_Medium_Threat	SSRF	918	**Source** has changed
Python	Python_Medium_Threat	Stored_Command_Injection	77	**Source** has changed
Python	Python_Medium_Threat	Stored_LDAP_Injection	90	**Source** has changed
Python	Python_Medium_Threat	Uncontrolled_Format_String	134	**Source** has changed
Python	Python_AWS_Lambda	Race_Condition_Global_Scope	366	Name changed from Race_Condition_Concurrent_Instances to **Race_Condition_Global_Scope**

Deprecated Queries:

Language	Group	Name	CWE
JavaScript	Javascript_Kony	Kony_Reflected_XSS	79
JavaScript	Javascript_Kony	Kony_Stored_XSS	79

Changed Source:

Apex / Apex_Force_com_Code_Quality / Test_Assert_Without_Message

Code changes

```
---
+++
@@ -1,12 +1,18 @@
 CxList isTest = All.FindByCustomAttribute("istest").GetFathers();
-CxList testClasses = isTest.FindByType(typeof(ClassDecl));
-testClasses.Add(isTest.FindByType(typeof(MethodDecl)));
+CxList testClasses = isTest.FindByTypes(typeof(ClassDecl), typeof(MethodDecl));

 CxList members = Find_UnknownReference()
-    .FindByShortName("system").GetMembersOfTarget();
+    .FindByShortNames("system", "assert").GetMembersOfTarget();

-CxList assertEquals = members.FindByShortNames(new List<string>{"assertNotEquals","assertEquals"}, false);
+// Apex Assert Class (58.0v)
+CxList assertEquals = members.FindByShortNames(new List<string>{"assertNotEquals", "assertEquals",
+       "areEqual", "areNotEqual", "isInstanceOfType", "isNotInstanceOfType"}, false);
+CxList failAssertMethod = members.FindByShortName("fail", false);
+CxList isAssertMethods = members.FindByShortNames(new List<string>{"isFalse", "isTrue", "isNotNull", "isNull"}, false);
+
 CxList validResult = assertEquals.FilterByDomProperty<MethodInvokeExpr>(x => x.Parameters.Count < 3);
+validResult.Add(isAssertMethods.FilterByDomProperty<MethodInvokeExpr>(x => x.Parameters.Count < 2),
+    failAssertMethod.FilterByDomProperty<MethodInvokeExpr>(x => x.Parameters.Count < 1));

 assertEquals = members.FindByShortName("assert", false);
```

Apex / Apex_Force_com_Code_Quality / Test_Methods_With_No_Assert

Code changes

```
---
+++
@@ -1,11 +1,11 @@
 CxList testCode = Find_Test_Code();
-CxList classDecl = testCode.FindByType(typeof(ClassDecl)); // all class decls in test code
-CxList testClasses = testCode.FindByCustomAttribute("istest").GetAncOfType(typeof(ClassDecl));
+CxList classDecl = testCode.FindByType<ClassDecl>();   // all class decls in test code
+CxList testClasses = testCode.FindByCustomAttribute("istest").GetAncOfType<ClassDecl>();
                                                        // all @isTest classes (in test code)
 CxList nonTestClasses = classDecl - testClasses;       // classes under test code that are not @isTest
 testCode -= testCode.GetByAncs(nonTestClasses);

-CxList testMethods = testCode.FindByType(typeof(MethodDecl));
+CxList testMethods = testCode.FindByType<MethodDecl>();

 // Remove the test utility methods (just static methods in test classes that don't have @isTest or testmethod modifier).
 CxList isTest = All.FindByCustomAttribute("istest").GetFathers();
```

```
@@ -14,14 +14,19 @@
 CxList methodInvoke = Find_Methods();

 CxList assertMethods = methodInvoke.FindByMemberAccesses(new string[] {"System.assert",
-   "System.assertEquals","System.assertNotEquals"}, false);
+   "System.assertEquals", "System.assertNotEquals"}, false);
+
+// Apex Assert Class (58.0v)
+assertMethods.Add(methodInvoke.FindByMemberAccesses("Assert", new string[]{
+       "areEqual", "areNotEqual", "fail", "isFalse", "isInstanceOfType",
+       "isNotInstanceOfType", "isNotNull", "isNull", "isTrue"}, false));

 // Check also calls to assert methods
 int numAssert = 0;
 for(int i = 0; i < 10 && assertMethods.Count > numAssert; i++)
 {
    numAssert = assertMethods.Count;
-   assertMethods.Add(All.FindAllReferences(assertMethods.GetAncOfType(typeof(MethodDecl))));
+   assertMethods.Add(All.FindAllReferences(assertMethods.GetAncOfType<MethodDecl>()));
 }

 CxList noAssert = testMethods - assertMethods;
@@ -49,6 +54,7 @@
 CxList noAssertTestReferences = testCode.FindAllReferences(noAssert) - noAssert;
 CxList noAssertNoTestReferences = (All - testCode).FindAllReferences(noAssert) - noAssert;
-CxList calledOnlyFromTest = testCode.FindDefinition(noAssertTestReferences) - testCode.FindDefinition(noAssertNoTestReferences);
+CxList calledOnlyFromTest = testCode.FindDefinition(noAssertTestReferences) -
+   testCode.FindDefinition(noAssertNoTestReferences);

 result = noAssert - calledOnlyFromTest;
```

CPP / CPP_Buffer_Overflow / Buffer_Overflow_Wrong_Buffer_Size

Code changes

```
---
+++
@@ -1,15 +1,30 @@
+//Types size in bytes
+Dictionary<string, int> typeSizeInBytes = new Dictionary<string, int>();
+typeSizeInBytes.Add("bool", 1);
+typeSizeInBytes.Add("char", 1);
+typeSizeInBytes.Add("__int8", 1);
+typeSizeInBytes.Add("__int16", 2);
+typeSizeInBytes.Add("short int", 2);
+typeSizeInBytes.Add("wchar_t", 2);
+typeSizeInBytes.Add("__wchar_t", 2);
+typeSizeInBytes.Add("float", 4);
+typeSizeInBytes.Add("__int32", 4);
```

```
+typeSizeInBytes.Add("int", 4);
+typeSizeInBytes.Add("double", 8);
+typeSizeInBytes.Add("__int64", 8);
+typeSizeInBytes.Add("long int", 8);
+typeSizeInBytes.Add("long double", 8);
+typeSizeInBytes.Add("longlong int", 8);
+
+CxList arrays = Find_ArrayCreateExpr();
+CxList typeRefs = Find_TypeRef();
 CxList methodInvokes = Find_Methods();
 CxList parameters = Find_Parameters();
 CxList unknownRefs = Find_Unknown_References();
 CxList integers = Find_Integers();
 integers.Add(Find_Integer_Literals());

-CxList customAttribute = Find_CustomAttribute().FindByShortName("CxNotLiteralBufferSize").GetFathers();
-customAttribute.Add(customAttribute.FindByType<CustomAttributeCollection>().GetFathers());
-CxList arrays = Find_ArrayCreateExpr().GetByAncs(customAttribute);
-
 CxList nodesWithAbsVal = All.NewCxList(unknownRefs, Find_Strings(), Find_CharLiteral(), integers);
-
 // Helper delegate to compare two parameters as for their AbsValue
 Func <CxList, string, CxList, bool> CompareExprsByAbsVal = delegate(CxList fstParam, string compareOp, CxList sndParam){

@@ -20,7 +35,24 @@
             IAbstractValue absValue = paramExpr.AbsValue;
             if (absValue is ObjectAbstractValue)
             {
+
                 paramAbsValue = (absValue as ObjectAbstractValue).AllocatedSize;
+
+                CxList arrayCreate = fstParam.InfluencedBy(arrays);
+                if(arrayCreate.Count > 0 && paramAbsValue != null){
+                    arrayCreate = arrayCreate.GetFirstNodesInPath()
+                                    .GetAncOfType(typeof(VariableDeclStmt), typeof(FieldDecl), typeof(ConstantDeclStmt));
+                    TypeRef typeRef = typeRefs.GetByAncs(arrayCreate).TryGetCSharpGraph<TypeRef>();
+
+                    if(typeRef != null){
+                        string typeName = (((typeRef.TypeSize != TypeSizeModifiers.Default) ? typeRef.TypeSize + " " : "") +
+                            typeRef.ResolvedTypeName).ToLower();
+                        int sizeInBytes;
+                        if(typeSizeInBytes.TryGetValue(typeName, out sizeInBytes)){
+                            paramAbsValue = paramAbsValue.Multiply(new IntegerIntervalAbstractValue(sizeInBytes)) as IntegerIntervalAbstractValue;
+                        }
+                    }
+                }
             }
             else if (absValue is IntegerIntervalAbstractValue)
```

```
            {
@@ -33,14 +65,14 @@
            }
        }
        return paramAbsValue;
-   };
+       };

    Expression firstParamExpr = fstParam.TryGetCSharpGraph<Expression>();
    Expression secondParamExpr = sndParam.TryGetCSharpGraph<Expression>();

    IntegerIntervalAbstractValue firstParamAbsValue = GetExprAbsValue(firstParamExpr);
    IntegerIntervalAbstractValue secondParamAbsValue = GetExprAbsValue(secondParamExpr);
-
+
    if (firstParamAbsValue != null && secondParamAbsValue != null)
    {
        IAbstractValue absValueResult = FalseAbstractValue.Default;
@@ -60,7 +92,7 @@
    }

    return false;
-};
+   };
```

```
 // Inputs
 CxList inputs = All.NewCxList(Find_Unbounded_Inputs(), Find_Read(), Find_DB());
@@ -137,13 +169,10 @@
            paramsToRemove.Add(parameters.GetParameters(method, 2));
            source -= paramsToRemove;
        }
-
+
        // Keep useful nodes
        CxList sizeWithSize = sizeWithValue.GetByAncs(size);
        CxList destinationWithSize = nodesWithAbsVal.GetByAncs(destination);
-
-       if(arrays.InfluencingOn(destinationWithSize).Count > 0)
-           continue;

        // Check if the size parameter contains size/strlen methods, which act as sanitizers
        CxList sizeInfluencedBySanitizer = sizeWithSize.InfluencedByAndNotSanitized(sizeParameters, Find_ParamDecl());
@@ -172,7 +201,7 @@

        bool smaller = CompareExprsByAbsVal(destinationWithSize, "<", sizeWithSize);
        bool greater = CompareExprsByAbsVal(destinationWithSize, ">", sizeWithSize);
-
+
```

```
        if (!smaller && greater)
        {
            continue;
```

Go / Go_AWS_Lambda / Permission_Manipulation_In_S3

Code changes

```
---
+++
@@ -1,15 +1,5 @@
+CxList inputs = Find_Interactive_Inputs();
 // S3 Put, Upload, etc...
 CxList outputs = S3_Find_DB_In_BucketConf();
-CxList s3types = S3_Find_DB_In_Types();

-// S3 Object creations
-CxList s3creates = s3types.GetAncOfType<ObjectCreateExpr>();
-
-// Identifying created object keys
-CxList objinputkeys = Find_Param().GetByAncs(s3creates).FilterByDomProperty<Param>(x => x.Name == "Key");
-CxList objinputkeysValue = Find_UnknownReference().GetByAncs(objinputkeys);
-
-// Inputs (definitions)
-CxList inputs = All.FindDefinition(objinputkeysValue);
-
-result = inputs.InfluencingOn(outputs);
+result = inputs.InfluencingOn(outputs).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Go / Go_AWS_Lambda / Unrestricted_Read_S3

Code changes

```
---
+++
@@ -1,21 +1,41 @@
 //FIND S3
-CxList s3API = S3_Find_DB_Out();
+CxList outputs = S3_Find_DB_Out();
+CxList inputs = Find_Interactive_Inputs();

-//FIND VALUE PARAMS KEY
-CxList objInputKeys = Find_Param().FilterByDomProperty<Param>(x => x.Name == "Key");
-CxList objInputKeysValue = Find_UnknownReference().GetByAncs(objInputKeys).InfluencingOn(s3API);
+//Sanitization
+CxList ifStmt = Find_Ifs();
+CxList metaDataChecks = Find_IndexerRefs().FindByShortName("Metadata").GetByAncs(ifStmt);
+CxList metadataIndices = Find_String_Literal().GetByAncs(metaDataChecks).FindByShortName("user_id").GetFathers()
+   .CxSelectDomProperty<IndexerRef>(_ => _.Target);

-//FIND IFSTMT TO SANITIZE
-CxList ifStmt = Find_Ifs();
```

```
+CxList s3Commands = S3_Find_Commands();
+CxList defer = Find_CustomAttribute().FindByShortName("defer").GetFathers();
+defer.Add(defer.FindByType<CustomAttributeCollection>().GetFathers());
+CxList deferOutputs = outputs.GetByAncs(defer);

-//FILTER BY METADATA CHECK
-CxList metaDataChecks = Find_IndexerRefs().FindByShortName("Metadata").GetByAncs(ifStmt);
-CxList strMetadataIndices = Find_String_Literal().GetByAncs(metaDataChecks).FindByShortName("user_id");
+CxList sanitizers = All.NewCxList();
+foreach(CxList check in metadataIndices.GetTargetOfMembers()){
+   //If Statement checking the user
+   CxList commandInfCheck = s3Commands.InfluencingOn(check).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+   CxList commandInfOutputs = commandInfCheck.GetFirstNodesInPath().InfluencingOn(outputs).GetLastNodesInPath();
+   int checkLine = check.GetFirstGraph().LinePragma.Line;
+   sanitizers.Add(commandInfOutputs.Filter(_ => _.Line > checkLine));
+
+   //defer
+   CxList methodDecl = check.GetAncOfType<MethodDecl>();
+   sanitizers.Add(deferOutputs.GetByAncs(methodDecl));
+}

-//GET SANITIZERS
-CxList sanitizers = objInputKeysValue.InfluencingOn(strMetadataIndices.GetFathers());
+CxList paths = outputs.InfluencedByAndNotSanitized(inputs, sanitizers).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+CxList pathsSinks = paths.GetLastNodesInPath();
+foreach(CxList sink in pathsSinks){
+
+   CxList duplicates = sink.GetTargetOfMembers().InfluencingOn(pathsSinks);
+
+   string firstMember = sink.CxSelectElementValue<CSharpGraph, string>(_ => _.ShortName).FirstOrDefault();
+   CxList dups = duplicates.GetLastNodesInPath().Filter(_ => _.ShortName == firstMember);
+
+   if(dups.Count > 0) sanitizers.Add(sink);
+}

-//Find relevant inputs
-CxList inputs = All.FindDefinition(objInputKeysValue);
-
-result = s3API.InfluencedByAndNotSanitized(inputs, sanitizers);
+result = paths.SanitizeCxList(sanitizers);
```

Go / Go_AWS_Lambda / Unrestricted_Write_S3

Code changes

```
---
+++
@@ -1,26 +1,20 @@
-CxList allparams = Find_Param();
-CxList unknownRefs = Find_UnknownReference();
```

```
+CxList inputs = Find_Interactive_Inputs();


 // S3 Put, Upload, etc...
 CxList outputs = S3_Find_DB_In_Methods();
-CxList s3types = S3_Find_DB_In_Types();


-// S3 Object creations
-CxList s3creates = s3types.GetAncOfType<ObjectCreateExpr>();
+//Sanitizers
+CxList ifStmt = Find_Ifs();
+CxList metaDataChecks = Find_IndexerRefs().FindByShortName("Metadata").GetByAncs(ifStmt);
+CxList metadataIndices = Find_String_Literal().GetByAncs(metaDataChecks).FindByShortName("user_id").GetFathers();


-// Identifying created object keys
-CxList objinputkeys = allparams.GetByAncs(s3creates).FilterByDomProperty<Param>(x => x.Name == "Key");
-CxList objinputkeysvalue = unknownRefs.GetByAncs(objinputkeys);
+CxList s3Types = S3_Find_DB_In_Types();
+CxList headObjects = s3Types.FindByShortName("HeadObjectInput").InfluencingOn(metadataIndices)
+   .GetFirstNodesInPath();
+CxList headObjectParams = Find_Param().GetByAncs(headObjects).FilterByDomProperty<Param>(_ => _.Name == "Key")
+   .CxSelectDomProperty<Param>(_ => _.Value);


-// Inputs (definitions)
-CxList inputs = All.FindDefinition(objinputkeysvalue);
+outputs -= outputs.GetByAncs(s3Types.InfluencedBy(headObjectParams).GetLastNodesInPath());


-// S3 headcommands that verify if the object already exists
-CxList headCommands = outputs.FindByShortName("HeadObject", false);
-headCommands.Add(All.FindAllReferences(headCommands.GetAssignee(0)));
-CxList headCommandKeys = objinputkeysvalue.InfluencingOn(headCommands);
-
-// Sanitizers are the head command's keys' declarators
-CxList sanitizers = All.FindDefinition(headCommandKeys);
-
-result = inputs.InfluencingOnAndNotSanitized(outputs, sanitizers);
+result = inputs.InfluencingOn(outputs);
+result.Add(inputs * outputs);
```

Go / Go_High_Risk / Stored_Command_Injection

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 /*** Finding inputs and sanitizers ***/
 CxList inputs = Find_Read();
-inputs.Add(Find_DB_Out());
+inputs.Add(Find_DB_Out(), S3_Find_DB_Out(), DynamoDB_Find_Inputs());
 CxList sanitizers = Find_Command_Injection_Sanitize();
```

```
 sanitizers.Add(Find_Test_Code());
```

Go / Go_High_Risk / Stored_XSS_All_Clients

Code changes

```
---
+++
@@ -9,7 +9,7 @@
 CxList outputs = All.NewCxList();
 CxList sanitize = All.NewCxList();

-fromDB.Add(Find_DB());
+fromDB.Add(Find_DB(), S3_Find_DB_Out(), DynamoDB_Find_Inputs());

 fromFiles = Find_Read();

@@ -36,7 +36,7 @@
 CxList missedResults = All.NewCxList();
 foreach ( CxList cx in firstPath.GetCxListByPath() )
 {
-   CxList arg = arguments.GetParameters(cx.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly), 0);
+   CxList arg = arguments.GetParameters(cx.GetLastNodesInPath(), 0);

    // only make the concatenation if the second path is not sanitized
    CxList secondPath = arg.InfluencingOnAndNotSanitized(outputs, sanitize);
```

Go / Go_Low_Visibility / Stored_Command_Argument_Injection

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 /*** Finding inputs and sanitizers ***/
-CxList inputs = Find_Read();
-inputs.Add(Find_DB_Out());
+CxList inputs = All.NewCxList(Find_Read(), Find_DB_Out(), S3_Find_DB_Out(), DynamoDB_Find_Inputs());
+
 CxList sanitizers = Find_Command_Injection_Sanitize();
 sanitizers.Add(Find_Test_Code());
```

Go / Go_Medium_Threat / Stored_Absolute_Path_Traversal

Code changes

```
---
+++
@@ -1,7 +1,9 @@
 CxList inputs = All.NewCxList();
 inputs.Add(
    Find_Read(),
```

```
-   Find_DB_Out());
+   Find_DB_Out(),
+   S3_Find_DB_Out()
+   , DynamoDB_Find_Inputs());

 CxList outputs = Find_Absolute_Path_Sinks();
```

Go / Go_Medium_Threat / Stored_Relative_Path_Traversal

Code changes

```
---
+++
@@ -1,7 +1,9 @@
 CxList inputs = All.NewCxList();
 inputs.Add(
    Find_Read(),
-   Find_DB_Out());
+   Find_DB_Out(),
+   S3_Find_DB_Out(),
+   DynamoDB_Find_Inputs());

 CxList outputs = Find_Relative_Path_Sinks();
```

Groovy / Groovy_Best_Coding_Practice / Exposure_of_Resource_to_Wrong_Sphere

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 CxList allFields = Find_Field_Decl();
 CxList allPublicFields = allFields.FindByFieldAttributes(Modifiers.Public);
 //CxList allProtectedFields = allFields.FindByFieldAttributes(Modifiers.Protected);
-CxList allConstFields = allFields.FindByFieldAttributes(Modifiers.Sealed);
+CxList allConstFields = allFields.FindByFieldAttributes(Modifiers.Final | Modifiers.Sealed);

 result = //allProtectedFields +
    allPublicFields - allConstFields;
```

Groovy / Groovy_Low_Visibility / Object_Hijack

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 CxList cloneable = All.InheritsFrom("Cloneable");
-CxList methodDecl = All.FindByType(typeof(MethodDecl));
+CxList methodDecl = Find_MethodDecls();
 CxList clone = methodDecl.GetByAncs(cloneable).FindByShortName("clone");
```

```
-result = clone - clone.FindByFieldAttributes(Modifiers.Sealed);
+result = clone - clone.FindByFieldAttributes(Modifiers.Sealed | Modifiers.Final);
```

Groovy / Groovy_Low_Visibility / Unsynchronized_Access_To_Shared_Data

Code changes

```
---
+++
@@ -1,19 +1,21 @@
 CxList logs = All.FindByTypes(new string [] {"Log", "Logger"});

-// Remove false sinks such asTypeRef and GenericTypeRef
-CxList no_logs = All - logs;
-no_logs -= no_logs.FindByType(typeof(ThisRef));
+// Remove false sinks such as TypeRef and GenericTypeRef
+CxList noLogs = All - logs;
+noLogs -= noLogs.FindByType(typeof(ThisRef));

-CxList statics =
-   no_logs.FindAllReferences(no_logs.FindByFieldAttributes(Modifiers.Static) -
-                             no_logs.FindByFieldAttributes(Modifiers.Sealed) -
-                             no_logs.FindByFieldAttributes(Modifiers.Readonly));
+CxList staticNoLogs = noLogs.FindByFieldAttributes(Modifiers.Static) -
+   All.NewCxList(noLogs.FindByFieldAttributes(Modifiers.Sealed),
+   noLogs.FindByFieldAttributes(Modifiers.Final),
+   noLogs.FindByFieldAttributes(Modifiers.Readonly));
+
+CxList statics = noLogs.FindAllReferences(staticNoLogs);
 statics -= Find_Methods();

 CxList staticDecl = statics.FindByType(typeof(Declarator));
 statics -= staticDecl;
-
-CxList inputs = Find_Interactive_Inputs() + All.FindByMemberAccess("ServerRequest.getAttribute");
+
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), All.FindByMemberAccess("ServerRequest.getAttribute"));
 CxList threadSafetyIssue = inputs.InfluencingOnAndNotSanitized(statics, staticDecl);

 CxList inputInitializers = inputs.GetByAncs(staticDecl);
@@ -35,16 +37,16 @@
 CxList initMethods = Find_MethodDecls().FindByShortName("init").GetByAncs(ServletClasses);
 StaticFields -= StaticFields.GetByAncs(initMethods);

-CxList Locks = All.FindByType(typeof(LockStmt));
+CxList Locks = Find_LockStmt();
 StaticFields = StaticFields - StaticFields.GetByAncs(Locks);

 CxList nullLiteral = Find_NullLiteral();
 nullLiteral -= nullLiteral.FindByRegex("null");
```

```
-CxList nonNull = All - nullLiteral - StaticFields - StaticFields.GetMembersOfTarget();
-StaticFields = All * StaticFields.DataInfluencingOn(nonNull).DataInfluencedBy(nonNull)
-   + StaticFields.GetByAncs(All.FindByType(typeof(PostfixExpr)).GetFathers());
+CxList nonNull = All - All.NewCxList(nullLiteral, StaticFields, StaticFields.GetMembersOfTarget());
+StaticFields = All.NewCxList(All * StaticFields.DataInfluencingOn(nonNull).DataInfluencedBy(nonNull),
+   StaticFields.GetByAncs(Find_PostfixExpr()).GetFathers());

 CxList StaticFieldsDef = All.FindDefinition(StaticFields);
 StaticFields -= StaticFieldsDef;

-result = threadSafetyIssue + StaticFields;
+result = All.NewCxList(threadSafetyIssue, StaticFields);
```

Java / Java_AWS_Lambda / AWS_Credentials_Leak

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 CxList unknownRefs = Find_UnknownReference();

 //All Outpus
-CxList outputs = S3_Find_DB_Out();
+CxList outputs = S3_Find_DB_In();
 outputs.Add(Find_Outputs());
 //Aws credentials
```

Java / Java_AWS_Lambda / DynamoDB_NoSQL_Injection

Code changes

```
---
+++
@@ -2,6 +2,6 @@

 CxList sanitizers = DynamoDB_Find_NoSQL_Injection_Sanitize();

-CxList outputs = DynamoDB_Find_DB_Out();
+CxList outputs = DynamoDB_Find_DB_In();

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

Java / Java_AWS_Lambda / Permission_Manipulation_in_S3

Code changes

```
---
+++
@@ -8,7 +8,7 @@
 CxList inputs = Find_AWSLambda_Inputs();

 //Find Param s3API Out
```

```
-CxList s3APIOut = S3_Find_DB_Out();
+CxList s3APIOut = S3_Find_DB_In();
 CxList paramOut = Find_Params().GetByAncs(s3APIOut);
 CxList unknownRefsParamsOut = Find_UnknownReference().GetByAncs(paramOut);
```

Java / Java_AWS_Lambda / Unrestricted_Write_S3

Code changes

```
---
+++
@@ -3,7 +3,7 @@
 CxList unknownReferences = Find_UnknownReference();

 // Sinks
-CxList s3Outs = S3_Find_DB_Out();
+CxList s3Outs = S3_Find_DB_In();

 //Find conditions influenced by s3API and their fathers
 CxList conditions = Find_Conditions().InfluencedBy(s3APISanitizers);
```

Java / Java_Best_Coding_Practice / Exposure_of_Resource_to_Wrong_Sphere

Code changes

```
---
+++
@@ -2,7 +2,7 @@
 //Public fields
 CxList allPublicFields = allFields.FindByFieldAttributes(Modifiers.Public);
 //Constant fields
-CxList allConstFields = allPublicFields.FindByFieldAttributes(Modifiers.Sealed);
+CxList allConstFields = allPublicFields.FindByFieldAttributes(Modifiers.Final);
 //Static fields
 CxList allStaticFields = allPublicFields.FindByFieldAttributes(Modifiers.Static);
```

Java / Java_Best_Coding_Practice / Potentially_Serializable_Class_With_Sensitive_Data

Code changes

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

 CxList methodDecl = Find_MethodDeclaration();
 // Find "writeObject" that is final and has exactly one parameter
-CxList writeObject = methodDecl.FindByShortName("writeObject").FindByFieldAttributes(Modifiers.Sealed);
+CxList writeObject = methodDecl.FindByShortName("writeObject").FindByFieldAttributes(Modifiers.Final);
 CxList writeObjectParams0 = All.GetParameters(writeObject, 0);
 CxList writeObjectParams1 = All.GetParameters(writeObject, 1);
 writeObject =
```

Java / Java_High_Risk / Unsafe_Reflection

Code changes

```
---
+++
@@ -21,4 +21,8 @@
 sanitizeMethods.Add(methods.FindByMemberAccess("SecurityManager.check*"));
 sanitize.Add(inputs.InfluencingOn(sanitizeMethods));

+//Concatenation with hardcoded strings
+CxList binaryExprs = Find_Strings().GetFathers().FindByType<BinaryExpr>();
+sanitize.Add(binaryExprs);
+
 result = inputs.InfluencingOnAndNotSanitized(outputs, sanitize);
```

Java / Java_Low_Visibility / Exposure_of_System_Data

Code changes

```
---
+++
@@ -1,6 +1,5 @@
-CxList deadCode = Find_Dead_Code_Contents();
-
-CxList getFromSystem = Find_Methods().FindByMemberAccess("System.getenv");
+CxList methods = Find_Methods();
+CxList getFromSystem = methods.FindByMemberAccess("System.getenv");

 CxList inputs = All.NewCxList(getFromSystem);

@@ -8,7 +7,10 @@

 CxList outputs = interactiveOutputs.FindByShortNames(new List<string> {"print*", "write*"});

-CxList sanitize = Find_Integers();
-sanitize.Add(deadCode);
+CxList sanitize = All.NewCxList(Find_Integers(), Find_Dead_Code_Contents());
+CxList javaSqlMethods = methods.FindByMemberAccess("DriverManager.getConnection");
+CxList connections = Find_UnknownReference().FindAllReferences(javaSqlMethods.GetAssignee());
+javaSqlMethods.Add(connections.GetMembersOfTarget().FindByShortName("createStatement"));
+sanitize.Add(javaSqlMethods);

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

Java / Java_Low_Visibility / Object_Hijack

Code changes

```
---
+++
@@ -1,4 +1,4 @@
 CxList cloneable = All.InheritsFrom("Cloneable");
 CxList clone = Find_MethodDecls().GetByAncs(cloneable).FindByShortName("clone");
```

```
-result = clone - clone.FindByFieldAttributes(Modifiers.Sealed);
+result = clone - clone.FindByFieldAttributes(Modifiers.Final);
```

Java / Java_Low_Visibility / Plaintext_Storage_in_a_Cookie

Code changes

```
---
+++
@@ -10,10 +10,22 @@
 CxList strings = Find_Strings();

 CxList toRemove = Find_Empty_Strings();
-toRemove.Add(strings.FindByShortNames(new string [] {"\"-\"", "\";\"", "\":\"", "\"_\"", "\"/\"", "\"\\\""})); //dividers
+toRemove.Add(strings.FindByShortNames(new string [] {"\"-\"", "\";\"", "';", "\":\"", "\"_\"", "\"/\"", "\"\\\""})); //dividers

 CxList plainText = strings - toRemove;

+CxList binaryExprs = plainText.GetAncOfType<BinaryExpr>()
+   .FilterByDomProperty<BinaryExpr>(_ => _.Operator == BinaryOperator.Add);
+CxList descendantBinary = All.NewCxList();
+foreach(CxList binaryExpr in binaryExprs){
+   descendantBinary.Add(binaryExprs.GetByAncs(binaryExpr) - binaryExpr);
+}
+plainText -= plainText.GetByAncs(descendantBinary);
+
 CxList sanitize = Find_General_Sanitize();
+CxList javaSqlMethods = methods.FindByMemberAccess("DriverManager.getConnection");
+CxList connections = Find_UnknownReference().FindAllReferences(javaSqlMethods.GetAssignee());
+javaSqlMethods.Add(connections.GetMembersOfTarget().FindByShortName("createStatement"));
+sanitize.Add(javaSqlMethods);

-result = plainText.InfluencingOnAndNotSanitized(cookieParam, sanitize);
+result = plainText.InfluencingOnAndNotSanitized(cookieParam, sanitize).ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Java / Java_Low_Visibility / Unsynchronized_Access_To_Shared_Data

Code changes

```
---
+++
@@ -13,7 +13,7 @@
 noLogs -= noLogs.FindByType<ThisRef>();

 CxList statics = noLogs.FindByFieldAttributes(Modifiers.Static);
-CxList unwanted = noLogs.FindByFieldAttributes(Modifiers.Sealed | Modifiers.Readonly);
+CxList unwanted = noLogs.FindByFieldAttributes(Modifiers.Sealed | Modifiers.Final | Modifiers.Readonly);
 unwanted.Add(locations);
 statics -= unwanted;
```

Java / Java_Medium_Threat / DoS_by_Sleep

Code changes

```
---
+++
@@ -49,6 +49,6 @@
    sanitizedSchedules,
    sanitizedTimeUnitSleep,
    Find_BinaryExpr().FindByName("=="),
-   All.FindAllReferences(Find_FieldDecls().FindByFieldAttributes(Modifiers.Sealed)));
+   All.FindAllReferences(Find_FieldDecls().FindByFieldAttributes(Modifiers.Final)));


 result = vulnerableSleepInvokes.InfluencedByAndNotSanitized(Inputs, sanitizers);
```

Java / Java_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -5,6 +5,9 @@
 CxList literals = All.NewCxList(strings, integerLiteral);
 CxList nullLiteral = Find_NullLiteral();
 CxList typeRefs = Find_TypeRef();
+CxList unknRefs = Find_UnknownReference();
+CxList methods = Find_Methods();
+CxList constructors = Find_ConstructorDecl();

 // Find names that are suspected to be personal info, e.g. String PASSWORD, Integer SSN
 // Remove string literals, such as x = "password"
@@ -63,7 +66,7 @@
 //Get unknown references with sensitive type
 CxList sensitiveTypeVars = typeRefs.FindByType(sensitiveClass).GetAncOfType<VariableDeclStmt>();
 CxList sensitiveTypeDecls = Find_Declarators().GetByAncs(sensitiveTypeVars);
-CxList sensitiveTypeUsage = Find_UnknownReference().FindAllReferences(sensitiveTypeDecls);
+CxList sensitiveTypeUsage = unknRefs.FindAllReferences(sensitiveTypeDecls);
 CxList sensitiveResponse = sensitiveTypeUsage * Find_HTTP_Responses();

 CxList private_info = All.NewCxList(
@@ -75,7 +78,7 @@

 // 3) Add exceptions (that could be thrown) to outputs.
 CxList exceptions = Find_ObjectCreations().FindByName("*Exception");
-CxList exceptionsCtors = Find_ConstructorDecl().FindByName("*Exception");
+CxList exceptionsCtors = constructors.FindByName("*Exception");

 // Handle the case where the super (base) constructor of the exception is used to create a new throwable exception
 CxList exceptionsCtorsWithSuper = exceptionsCtors.FilterByDomProperty<ConstructorDecl>(c => c.BaseParameters.Count > 0);
@@ -93,7 +96,12 @@
 // Add additional "integer" sanitizers
 sanitize.Add(All.FindByShortNames(new string[] {"size", "length", "Index*", "indexOf"}, false),
```

```
    All.FindByName("*boolean.class.cast", StringComparison.OrdinalIgnoreCase),
-   Find_Methods().FindByMemberAccess("Boolean.parse*"));
+   methods.FindByMemberAccess("Boolean.parse*"));
+
+CxList javaSqlMethods = methods.FindByMemberAccess("DriverManager.getConnection");
+CxList connections = unknRefs.FindAllReferences(javaSqlMethods.GetAssignee());
+javaSqlMethods.Add(connections.GetMembersOfTarget().FindByShortName("createStatement"));
+sanitize.Add(javaSqlMethods);

 // Split personal_info into variables and constants
 CxList variableRef = personal_info - allConstRef;
@@ -107,6 +115,10 @@
 // remove the declaration from the references of the variables and constants
 variableRef -= declarator;
 allConstRef -= declarator;
+
+// remove duplicate results from object creations
+CxList variableRefsConstructs = variableRef.GetByAncs(constructors);
+variableRef -= variableRefsConstructs.InfluencedBy(variableRefsConstructs.FindByType<ParamDecl>());

 // Find all constants that are assigned from an input (directly or indirectly) and are influencing an output
 CxList constInfluencedByInputAux = inputs.InfluencingOnAndNotSanitized(outputs, sanitize);
```

Java / Java_Spring / Spring_defaultHtmlEscape_Not_True

Code changes

```
---
+++
@@ -12,14 +12,20 @@
        XPathNodeIterator springNodeIterator = doc.CreateNavigator().Select("//*[contains(name(),'spring')]|//*[contains(text(),'spring')]");
        if(!springNodeIterator.MoveNext()) break;

-       String xpath = "//context-param/param-name[text()='defaultHtmlEscape']/following-sibling::param-value[text()!='true']";
-       XPathNodeIterator nodeIterator = doc.CreateNavigator().Select(xpath);
-       if(nodeIterator.MoveNext()){
-           result = cxXPath.CreateXmlNode(nodeIterator.Current, doc, 2, false);
+       String xpathHtmlEscapeTrue = @"//*[local-name() = 'context-param']/descendant::*[local-name() = 'param-name'
+           and text()='defaultHtmlEscape']/following-sibling::*[local-name() = 'param-value' and text() = 'true']";
+       XPathNodeIterator htmlEscapeTrue =  doc.CreateNavigator().Select(xpathHtmlEscapeTrue);
+       if(htmlEscapeTrue.MoveNext()) break;
+
+       String xpathHtmlEscapeFalse = @"//*[local-name() = 'context-param']/descendant::*[local-name() = 'param-name'
+           and text()='defaultHtmlEscape']/following-sibling::*[local-name() = 'param-value' and not(text() = 'true')]";
+       XPathNodeIterator htmlEscapeFalse = doc.CreateNavigator().Select(xpathHtmlEscapeFalse);
+       if(htmlEscapeFalse.MoveNext()){
+           result = cxXPath.CreateXmlNode(htmlEscapeFalse.Current, doc, 2, false);
        }
        else{
            XPathNodeIterator rootNodeIterator = doc.CreateNavigator().Select("/*");
```

```
-           nodeIterator.MoveNext();
+           htmlEscapeFalse.MoveNext();
            result = cxXPath.CreateXmlNode(rootNodeIterator.Current, doc, 2, false);
        }
    }
```

Java / Java_Spring / Spring_Missing_HSTS_Header

Code changes

```
---
+++
@@ -17,6 +17,32 @@
 CxList setHeader = methods.FindByMemberAccess("HttpServletResponse.setHeader");
 CxList setHeaderHsts = strings.GetParameters(setHeader).FindByShortName("Strict-Transport-Security");
 CxList hstsHeaders = setHeader.FindByParameters(setHeaderHsts);
+
+//Remove hsts headers with valid max age value
+CxList headerValues = strings.GetParameters(hstsHeaders, 1);
+CxList validHeaderValues = headerValues.FindByRegex(@"(?<=max-age\s*=\s*)(.*[1-9]\d{8,}|[3-9][1-9][5-9][3-9][6-9][0-9][0-9][0-9])");
+hstsHeaders -= hstsHeaders.FindByParameters(validHeaderValues);
+
+//HttpSecurity object
+CxList httpSecurity = Find_UnknownReference().FindByType("HttpSecurity").GetByAncs(methods.FindByShortName("headers"));
+
+CxList hstsHttpSecurity = methods.FindByShortName("httpStrictTransportSecurity");
+CxList toRemove = All.NewCxList();
+foreach(CxList httpHsts in hstsHttpSecurity){
+   if((httpHsts.GetLeftmostTarget() * httpSecurity).Count < 1)
+       toRemove.Add(httpHsts);
+}
+hstsHttpSecurity -= toRemove;
+
+CxList httpSecurityMaxAge = methods.FindByShortName("maxAgeInSeconds").FindByParameters(
+           Find_IntegerLiterals().FilterByDomProperty<IntegerLiteral>(_ => _.Value >= 31536000));
+CxList invalidHstsHttpSecurity = hstsHttpSecurity - hstsHttpSecurity.GetByAncs(httpSecurityMaxAge);
+hstsHeaders.Add(invalidHstsHttpSecurity);
+
+//HttpSecurity disabled
+CxList httpSecurityDisabled = hstsHttpSecurity.GetByAncs(methods.FindByShortName("disable"));
+hstsHeaders.Add(httpSecurityDisabled);
+
 CxList setHeaderClasses = hstsHeaders.GetAncOfType<ClassDecl>();
 CxList sanitizedHeaders = setHeaderClasses * sanitizedWebFilterClasses;
 CxList vulnerableHeaders = setHeaderClasses - sanitizedHeaders;
```

JavaScript / JavaScript_High_Risk / Client_DOM_Stored_XSS

Code changes

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

```
@@ -1,19 +1,37 @@
 //web storage stored xss
 CxList outputs = Find_Outputs_XSS();

-CxList sanitize = basic_Sanitize();
-sanitize.Add(Find_XSS_Sanitize());
+CxList sanitize = All.NewCxList(
+   basic_Sanitize(),
+   Find_XSS_Sanitize());

-CxList sources = Find_Storage_Outputs();
-sources.Add(Find_Cookie(), Find_DB_Out(), Find_XHR_Response());
+CxList sources = All.NewCxList(
+   Find_Storage_Outputs(),
+   Find_Cookie(),
+   Find_DB_Out(),
+   Find_XHR_Response());

 //All sources and elements influenced by them, that have a sanitizer in their ancestors, are also sanitizers
-CxList sourcesAndInfluencedBy = All.NewCxList();
-sourcesAndInfluencedBy.Add(sources,
-                       All.InfluencedBy(sources, CxList.InfluenceAlgorithmCalculation.NewAlgorithm));
+CxList sourcesAndInfluencedBy = All.NewCxList(
+   sources,
+   All.InfluencedBy(sources, CxList.InfluenceAlgorithmCalculation.NewAlgorithm));

 sanitize.Add(sourcesAndInfluencedBy.GetByAncs(sanitize),
            Find_SAPUI_XSS_Sanitized_Outputs());
+
+//Bad sinks
+CxList jQuerySink = Find_JQuery_Methods().FindByShortName("$");
+CxList innerHTMLs = Find_Members("innerHTML");
+CxList innerHTMLVars = innerHTMLs.GetAssignee();
+CxList badInnerhtmlSinks = innerHTMLVars * innerHTMLs;
+badInnerhtmlSinks.Add(
+   innerHTMLs.FindByAssignmentSide(CxList.AssignmentSide.Right),
+   innerHTMLs.DataInfluencedBy(innerHTMLVars).GetLastNodesInPath());
+CxList toRemove = All.NewCxList(
+   jQuerySink.GetByAncs(innerHTMLs),
+   jQuerySink.FindByAssignmentSide(CxList.AssignmentSide.Right),
+   badInnerhtmlSinks);
+outputs -= toRemove;

 result = outputs.InfluencedByAndNotSanitized(sources, sanitize, CxList.InfluenceAlgorithmCalculation.NewAlgorithm);
```

JavaScript / JavaScript_High_Risk / Client_DOM_XSS

Code changes

```
---
+++
@@ -1,5 +1,7 @@
 CxList constructors = Find_ConstructorDecl();
 CxList thisRef = Find_ThisRef();
+CxList binExprs = Find_BinaryExpr();
+CxList fieldDecls = Find_FieldDecls();

 CxList rClass = Find_MemberAccesses().GetByAncs(Find_ClassDecl().InheritsFrom("React.Component"));
 CxList propsOutputs = rClass.FindByShortName("props").GetMembersOfTarget();
@@ -9,13 +11,33 @@
 // Remove location outputs where a hardcoded prefix is used
 CxList urlOutputs = Find_Outputs_Redirection().FindByShortNames(new List<string>{ "location", "href" });
 CxList location = urlOutputs.FindByShortName("location");
-CxList locationPrefix =  location.GetAssigner().CxSelectDomProperty<BinaryExpr>(x => x.Left).FindByType<StringLiteral>();
+CxList locationPrefix = location.GetAssigner().CxSelectDomProperty<BinaryExpr>(x => x.Left).FindByType<StringLiteral>();
 urlOutputs -= locationPrefix.GetAncOfType<BinaryExpr>().GetAssignee();
 urlOutputs -= React_Find_PropertyKeys();
 outputs.Add(urlOutputs);

 CxList inputs = Find_Inputs_NoWindowLocation();
 inputs.Add(propsOutputs);
+
+// Exclude inputs that are suffixes in a string only the leftmost input in a string is considered valid to control the scheme.
+CxList validLeftMostPrefix = inputs.GetFathers().CxSelectDomProperty<BinaryExpr>(x => x.Left).FindByType<MemberAccess>();
+CxList invalidInputSufix = inputs.FindDescendantsOfType<MemberAccess>(binExprs) - validLeftMostPrefix;
+
+// An exception when there is an empty string preceding the vulnerable input, still making it valid to control the scheme.
+CxList emptyStringBinExpr = binExprs.InfluencedBy(Find_Empty_Strings());
+CxList validInputAfterEmptyString = emptyStringBinExpr.CxSelectDomProperty<BinaryExpr>(x => x.Right).FindByType<MemberAccess>();
+validInputAfterEmptyString = validInputAfterEmptyString * inputs;
+
+// Another exception, when there is a html heading like '<h1>' + input + '</h1>', it is still vulnerable
+CxList inputPrefixHtmlTag = inputs.GetFathers().CxSelectDomProperty<BinaryExpr>(x => x.Left).FindByType<StringLiteral>();
+CxList binaryExprBeforeInput = (inputPrefixHtmlTag.GetByAncs(fieldDecls)).GetFathers();
+
+CxList rightInputBetweenHtmlTag = binaryExprBeforeInput.CxSelectDomProperty<BinaryExpr>(x => x.Right)
+   .FindByType<MemberAccess>();
+rightInputBetweenHtmlTag = rightInputBetweenHtmlTag * inputs;
+
+inputs -= invalidInputSufix;
+inputs.Add(validInputAfterEmptyString, rightInputBetweenHtmlTag);

 CxList sanitize = basic_Sanitize();
 sanitize.Add(Find_XSS_Sanitize(),
```

JavaScript / Javascript_Kony / Kony_Reflected_XSS

Code changes

```
---
```

```
+++
```

```
@@ -1 +1 @@
```

```
-// Deprecated
```

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

JavaScript / Javascript_Kony / Kony_Stored_XSS

Code changes

```
---
```

```
+++
```

```
@@ -1 +1 @@
```

```
-// Deprecated
```

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

JavaScript / JavaScript_Medium_Threat / Unchecked_Input_For_Loop_Condition

Code changes

```
---
```

```
+++
```

```
@@ -60,16 +60,55 @@
```

```
 // sanitizers
```

```
 CxList sanitizers = Find_DoS_Sanitizers();
```

```


-//add to sanitizers when input is checked in an if stmt against an AnyAbstractValue
-//since we can't determine the value we assume it is sanitizing
-CxList loopConditionRefs = unkRefs.FindAllReferences(loopConditions);
-CxList relevantConditions = unkRefs.FindByFathers(conditions).Intersect(loopConditionRefs).GetFathers();
-//we ignore the abstract value of the reference to the loop condition itself, we want to check
-//what is it beeing compared to
-CxList targetRef = unkRefs.FindByFathers(relevantConditions) - loopConditionRefs;
-//if it is compared to something that we don't know the value of, it is sanitized
-CxList comparisson = targetRef.FindByAbstractValue(_=>_ is AnyAbstractValue).GetFathers();
-CxList ifAssigns = loopConditionRefs.GetByAncs(comparisson.GetFathers());
-sanitizers.Add(ifAssigns.InfluencingOn(unboundLoopConditions).GetLastNodesInPath());
+//add to sanitizers when input is checked in an if stmt
+CxList ifStmtConditions = Find_Ifs().CxSelectDomProperty<IfStmt>(_ => _.Condition).FindByType<BinaryExpr>()
+    .FindByShortNames("<", "<=", "==", ">", ">=");
+CxList equalConditions = ifStmtConditions.FindByShortName("==");
+CxList conditionsValidValues = All.NewCxList(
+    ifStmtConditions.FindByShortNames("<", "<=").CxSelectDomProperty<BinaryExpr>(_ => _.Left),
+    ifStmtConditions.FindByShortNames(">", ">=").CxSelectDomProperty<BinaryExpr>(_ => _.Right),
+    equalConditions.CxSelectDomProperty<BinaryExpr>(_ => _.Left),
+    equalConditions.CxSelectDomProperty<BinaryExpr>(_ => _.Right));


+CxList exprs = Find_Expressions();
+CxList assignExprs = Find_AssignExpr();
+foreach(CxList cond in ifStmtConditions){
+
+    /*Direct If Statement
```

```
+       if(totalService < MAX_ITERATIONS){
+           for (let i = 0; i < totalService; i++)
+   */
+   CxList conditionValidVals = conditionsValidValues.GetByAncs(cond);
+   conditionValidVals = exprs.FindAllReferences(conditionValidVals).GetByAncs(cond.GetAncOfType<IfStmt>());
+   sanitizers.Add(conditionValidVals * unboundLoopConditions);
+
+   /*Previously Sanitized If Statement
+       if (totalService > MAX_LOOPS) {
+           totalService = MAX_LOOPS;
+       }
+       for (let i = 0; i < totalService; i++)
+   */
+   BinaryExpr binary = cond.TryGetCSharpGraph<BinaryExpr>();
+   BinaryOperator binaryOperator = binary.Operator;
+   CxList assigns = assignExprs.GetByAncs(cond.GetAncOfType<IfStmt>()).FilterByDomProperty<AssignExpr>(
+       _ => _.Operator == AssignOperator.Assign).CxSelectDomProperty<AssignExpr>(_ => _.Left);
+   CxList binaryLeft = exprs.FindAllReferences(cond.CxSelectDomProperty<BinaryExpr>(_ => _.Left))
+       .GetByAncs(cond.GetAncOfType<IfStmt>());
+   CxList binaryRight = exprs.FindAllReferences(cond.CxSelectDomProperty<BinaryExpr>(_ => _.Right))
+       .GetByAncs(cond.GetAncOfType<IfStmt>());
+
+   if(binaryOperator == BinaryOperator.GreaterThan || binaryOperator == BinaryOperator.GreaterThanOrEqual){
+       assigns = (assigns * binaryLeft).GetAncOfType<AssignExpr>()
+           .CxSelectDomProperty<AssignExpr>(_ => _.Right) * binaryRight;
+   }
+   else if(binaryOperator == BinaryOperator.LessThan || binaryOperator == BinaryOperator.LessThanOrEqual){
+       assigns = (assigns * binaryRight).GetAncOfType<AssignExpr>()
+           .CxSelectDomProperty<AssignExpr>(_ => _.Right) * binaryLeft;
+   }
+   else{
+       continue;
+   }
+   assigns = assigns.GetAncOfType<AssignExpr>();
+   sanitizers.Add(assigns.InfluencingOnAndNotSanitized(unboundLoopConditions, assignExprs - assigns).GetLastNodesInPath());
+}
 result = inputs.InfluencingOnAndNotSanitized(unboundLoopConditions, sanitizers);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Cleartext_Storage_Of_Sensitive_Information

Code changes

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

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

JavaScript / JavaScript_Server_Side_Vulnerabilities / Insecure_Storage_of_Sensitive_Data

Code changes

```
---
+++
@@ -18,10 +18,15 @@
 // Support for electron library
 CxList electron = Find_Require("electron");
 CxList electronSession = Find_MemberAccesses().FindByShortNames("session", "defaultSession");
-session.Add( electron,
+session.Add(electron,
     electronSession.GetMembersWithTargets(electron));

-CxList encrypt = Find_Encrypt();
+
+CxList assignToBool = Find_AssignExpr().CxSelectDomProperty<AssignExpr>(_ => _.Right) * Find_BooleanLiteral();
+assignToBool = assignToBool.GetAncOfType<AssignExpr>().CxSelectDomProperty<AssignExpr>(_ => _.Left);
+
+CxList sanitizers = All.NewCxList(Find_Encrypt(), Find_Integers(),
+   Find_Expressions().GetByAncs(assignToBool));

 CxList SourcesList = All.NewCxList();
 CxList assignRight = All.FindByAssignmentSide(CxList.AssignmentSide.Right);
@@ -45,12 +50,12 @@
     }                               // {parameter: secret} , {parameter: secret + token} ...
 }

-// Remove assigning of empty strings from secrets
-CxList emptyString = Find_String_Literal().FindByShortName("");
-SourcesList -= emptyString;
+// Remove assigning of strings from secrets
+CxList strings = Find_String_Literal();
+SourcesList -= strings;
 SourcesList -= Find_NullLiteral();

-result = SourcesList.InfluencingOnAndNotSanitized(session, encrypt);
+result = SourcesList.InfluencingOnAndNotSanitized(session, sanitizers);

 // Flows into DB In that go through the parameters (we don't want flow through the target)
 List<string> nonStorageMethods = new List<string> {
@@ -69,11 +74,25 @@

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

```
+
+//When dealing with Sequelize.query methods,
+//only vulnerable if flow passes trough 1st parameter and query options (2nd param) are set to either INSERT or UPDATE
+CxList sequelizeMethods = dbIn.FindByMemberAccess("Sequelize.query");
+CxList vulnSeqQueryMethods = Find_SensitiveData_Vuln_Sequelize_Methods();
+CxList parameters = Find_Parameters().CxSelectDomProperty<Param>(_ => _.Value);
+CxList seqFirstParam = parameters.GetParameters(vulnSeqQueryMethods, 0);
+
+result.Add(SourcesList.InfluencingOnAndNotSanitized(vulnSeqQueryMethods, sanitizers).IntersectWithNodes(seqFirstParam));
+
+CxList toRemove = All.NewCxList(
+   sequelizeMethods,
+   All.GetByAncs(sequelizeMethods));
+dbIn -= toRemove;


 CxList outputs = All.NewCxList(Find_Cloud_Storage_Outputs(), dbIn);


-result.Add(SourcesList.InfluencingOnAndNotSanitized(outputs, encrypt));
+result.Add(SourcesList.InfluencingOnAndNotSanitized(outputs, sanitizers));


 // 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

```
---
+++
@@ -1,4 +1,4 @@
-CxList personalInfo = Find_Personal_Info()-Find_String_Literal();
+CxList personalInfo = Find_Personal_Info() - Find_String_Literal();
 CxList sanitizers = NodeJS_Find_Encrypt();
 sanitizers.Add(All.GetParameters(Find_ObjectCreations().FindByShortName("Sequelize")),
             Find_Integers());
@@ -11,11 +11,26 @@
              All_Passwords());
 personalInfo -= toRemove;


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


-result = personalInfo.InfluencingOnAndNotSanitized(storage, sanitizers)
+//When dealing with Sequelize.query methods,
+//only vulnerable if flow passes trough 1st parameter and query options (2nd param) are set to either INSERT or UPDATE
+CxList sequelizeMethods = nodeJsDbIn.FindByMemberAccess("Sequelize.query");
```

```
+CxList vulnSeqQueryMethods = Find_SensitiveData_Vuln_Sequelize_Methods();
+CxList parameters = Find_Parameters().CxSelectDomProperty<Param>(_ => _.Value);
+CxList seqFirstParam = parameters.GetParameters(vulnSeqQueryMethods, 0);
+
+result = personalInfo.InfluencingOnAndNotSanitized(vulnSeqQueryMethods, sanitizers).IntersectWithNodes(seqFirstParam);
+
+CxList toRemoveSequelize = All.NewCxList(
+   sequelizeMethods,
+   All.GetByAncs(sequelizeMethods));
+storage -= toRemoveSequelize;
+
+result.Add(personalInfo.InfluencingOnAndNotSanitized(storage, sanitizers)
    .ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow)
-   .ReduceFlowByPragma();
+   .ReduceFlowByPragma());
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Password_Weak_Encryption

Code changes

```
---
+++
@@ -16,6 +16,13 @@
 CxList db = NodeJS_Find_DB_Base();

 CxList candidates = resource.DataInfluencingOn(weakCrypt).DataInfluencingOn(pass);
-candidates.Add(unknownRef.DataInfluencingOn(pass));
+candidates.Add(unknownRef.DataInfluencingOn(weakCrypt).DataInfluencingOn(pass));

-result = candidates.DataInfluencingOn(db, CxList.InfluenceAlgorithmCalculation.NewAlgorithm);
+CxList sqlQuery = db.FindByShortName("query", false);
+db -= sqlQuery;
+CxList sqlQueryParams = All.GetParameters(sqlQuery);
+CxList validFlowResults = candidates.DataInfluencingOn(sqlQuery, CxList.InfluenceAlgorithmCalculation.NewAlgorithm)
+   .IntersectWithNodes(sqlQueryParams);
+
+result.Add(candidates.DataInfluencingOn(db, CxList.InfluenceAlgorithmCalculation.NewAlgorithm),
+   validFlowResults);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Poor_Database_Access_Control

Code changes

```
---
+++
@@ -8,21 +8,33 @@
    online catalog whose products are displayed
    according to an ID coming from the end-user.
 */
-
+CxList variables = Find_UnknownReference();
+CxList memberAccesses = Find_MemberAccesses();
```

```
 CxList query = NodeJS_Find_DB_Base();
-CxList queryParam = All.GetParameters(query);
-CxList variables = Find_UnknownReference();
-CxList memberAccesses = Find_MemberAccesses();
-CxList input = NodeJS_Find_Interactive_Inputs() + NodeJS_Find_Read();
+
+//For Sequelize.query methods, only if the flow passes through the 1st parameter is it vulnerable
+CxList sequelizeQueryMethods = query.FindByMemberAccess("Sequelize.query");
+CxList seqQueryFirstParam = All.GetParameters(sequelizeQueryMethods, 0);
+query -= sequelizeQueryMethods;
+
+CxList queryParam = All.NewCxList(
+   All.GetParameters(query),
+   seqQueryFirstParam,
+   NodeJS_MongoDB_Input_Methods());
+
+CxList input = All.NewCxList(
+   NodeJS_Find_Interactive_Inputs(),
+   NodeJS_Find_Read());

 List <string> relevantNames = new List<string> {
-   "_id", "id_", "id",
-   "account", "accountid", "account_id", "_account",
-   "product", "productid", "product_id",
-   "user", "user_id", "user_account"
-   };
-CxList varsAndMemberAccesses = All.NewCxList();
-varsAndMemberAccesses.Add(variables, memberAccesses);
+       "_id", "id_", "id",
+       "account", "accountid", "account_id", "_account",
+       "product", "productid", "product_id",
+       "user", "user_id", "user_account"
+       };
+
+CxList varsAndMemberAccesses = All.NewCxList(variables, memberAccesses);
+
 CxList externalSources = varsAndMemberAccesses.FindByShortNames(relevantNames, false).InfluencedBy(input);
-result = queryParam.InfluencedBy(externalSources).InfluencedBy(input);
+result = queryParam.InfluencedBy(externalSources).InfluencedBy(input).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

JavaScript / JavaScript_Server_Side_Vulnerabilities / Second_Order_SQL_Injection

Code changes

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

```
+CxList outputs = NodeJS_Find_SQL_DB_IN().GetLastNodesInPath();
+CxList inputs = All.NewCxList(NodeJS_Find_DB_Out(), Find_Cloud_Storage_Inputs(), NodeJS_Find_Read());
 CxList sanitize = NodeJS_Find_Sanitize();

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

JavaScript / JavaScript_Server_Side_Vulnerabilities / Sensitive_Information_Over_HTTP

Code changes

```
---
+++
@@ -7,7 +7,7 @@
 CxList vulnerableRequests = httpRequests.FindByParameters(httpsString);

 CxList sensitiveInfo = Find_Personal_Info();
-CxList sanitizers = NodeJS_Find_Encrypt();
+CxList sanitizers = Find_Encrypt();

 result = sensitiveInfo.InfluencingOnAndNotSanitized(vulnerableRequests, sanitizers)
     .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 = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs(), NodeJS_Find_Read());
+CxList inputs = All.NewCxList(NodeJS_Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList sanitize = NodeJS_Find_Sanitize();

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

Kotlin / Kotlin_Medium_Threat / DoS_by_Sleep

Code changes

```
---
+++
@@ -67,6 +67,6 @@
     sanitizedSchedules,
     sanitizedTimeUnitSleep,
     Find_BinaryExpr().FindByShortName("=="),
-    All.FindAllReferences(Find_FieldDecls().FindByFieldAttributes(Modifiers.Sealed)));
+    All.FindAllReferences(Find_FieldDecls().FindByFieldAttributes(Modifiers.Final)));

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

PHP / PHP_High_Risk / Command_Injection

Code changes

```
---
+++
@@ -2,5 +2,4 @@
 CxList dynamic_shell = Find_Command_Execution();
 CxList sanitize = Find_Command_Injection_Sanitize();
 CxList inputs = Find_Interactive_Inputs();
-
 result = inputs.InfluencingOnAndNotSanitized(dynamic_shell, sanitize);
```

PHP / PHP_Low_Visibility / Error_Messages_Misconfiguration

Code changes

```
---
+++
@@ -18,7 +18,7 @@
 CxList errorConfig = All.NewCxList(
     iniSet.FindByParameters(configParam),
     errorReporting
-);
+   );
 errorConfig -= errorConfig.FindByParameters(falsyParams);
 CxList iniSetMisConfig = errorConfig.FindByShortName("ini_set", false);
 errorConfig -= iniSetMisConfig;
@@ -34,7 +34,7 @@
     errorConfig,
     phpIniMisconfig,
     parameters.GetParameters(iniSetMisConfig, 0)
-);
+   );

 // php.ini Good config
 regex = @"(?<=^[^;]*)(display_(startup_)?errors)(?=\s*=\s*([oO][fF]{2}|0)?\r?$)"
@@ -43,15 +43,16 @@

 // Missing config results
 options = RegexOptions.ExplicitCapture;
+
+List<string> extensions = new List<string> {"*.php", "*.php3", "*.php4", "*.php5", "*.inc", "*.phtml", "*.phtm", "*.twig"};
 foreach(string config in configNames)
 {
     if(goodConfig.FindByShortName(config, false).Count > 0 || misConfigs.FindByShortName(config, false).Count > 0)
         continue;
     regex = @"^(?<missing_config_" + config + ">)";
-    CxList placeHolder = All.FindByRegexExt(regex, "*.*", CxList.CxRegexOptions.ForceContentWithNamedGrouping, options)
-        .FilterPlugins().FirstOrDefault();
+    CxList placeHolder = All.FindByRegexExt(regex, extensions, true, CxList.CxRegexOptions.ForceContentWithNamedGrouping, options).FilterPlugins().FirstOrDefault();
     if (placeHolder is not null)
-        misConfigs.Add(placeHolder);
+    misConfigs.Add(placeHolder);
```

```
 }


 result = misConfigs;
```

PHP / PHP_Low_Visibility / Unsafe_Use_Of_Target_Blank

Code changes

```
---
+++
@@ -12,9 +12,10 @@
 string relPattern = @"rel\s*=\s*('|""|\\"")?(noopener\s?|noreferrer\s?)+('|""|\\"")?";


 // 1.1 - target=_blank on HTML
+List<string> extensions = new List<string> {"*.php", "*.php3", "*.php4", "*.php5", "*.inc", "*.phtml", "*.phtm", "*.twig"};


 // Find for all HTML comments from href pattern
-CxList hrefMatches = All.FindByRegexExt(anchorHREFPattern, "*.*", true, RegexOptions.IgnoreCase);
+CxList hrefMatches = All.FindByRegexExt(anchorHREFPattern, extensions, true, CxList.CxRegexOptions.None, RegexOptions.IgnoreCase);


 // Target and Rel Regex to be used on foreach
 Regex regexTarget = new Regex(targetPattern, RegexOptions.Compiled);
```

PHP / PHP_Medium_Threat / Missing_Encryption_of_Sensitive_Data

Code changes

```
---
+++
@@ -1,14 +1,22 @@
 CxList outputs = All.NewCxList(
-   Find_DB_In(),
+   Find_DB_In() - Find_Raw_SQL_Selects(),
    Find_File_Write_Outputs());


 CxList sanitizers = All.NewCxList(
-   Find_Hashing_Sanitize(),
+   Find_Hashing_Functions(),
    Find_Encryption_Sanitize());


-//Add PASSWORD_BCRYPT because its not a general hashing sanitizer
-CxList parameters = Find_Params().CxSelectDomProperty<Param>(x => x.Value).FindByShortName("PASSWORD_BCRYPT", false);
-sanitizers.Add(Find_Hashing_Functions().FindByParameters(parameters));
+string[] heuristics = new []{
+       "*Account*","*credentials*","*Credit*","*secret*",
+       "*SocialSecurity*","*SSN","SSN*","dob", "username",
+       "auth*", "*address*", "*mobile*","*telephone*", "pwd", "*password", "user_login"};
+
+CxList possiblePii = All.NewCxList(
+   // indexer refs whose key matches he heuristic
+   Find_Strings().FindByShortNames(heuristics, false).FindByFathers(Find_IndexerRefs()).GetFathers(),
+   // any unk ref that matches the heuristics
```

```
+   Find_UnknownReference().FindByShortNames(heuristics, false));


 result = Find_Interactive_Inputs().InfluencingOnAndNotSanitized(outputs, sanitizers)
+   .IntersectWithNodes(possiblePii)
    .ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);
```

Python / Python_AWS_Lambda / Unrestricted_Read_S3

Code changes

```
---
+++
@@ -16,6 +16,7 @@


    // sanitizers
    CxList sanitizers = Find_WhiteListSanitizers();
+   sanitizers.Add(Find_AWSLambda_S3_Objects_Sanitizers());
    CxList ifStmt = Find_Ifs();
    CxList ifStmtCond = Find_Expressions().FindByFathers(ifStmt);
```

Python / Python_AWS_Lambda / Unrestricted_Write_S3

Code changes

```
---
+++
@@ -22,10 +22,11 @@
    CxList tryStmts = relevantCatchStmts.GetFathers().FindByType<TryCatchFinallyStmt>();


-   CxList s3RefsInTryBlock = s3Refs.GetByAncs(tryStmts);
+   CxList sanitizers = s3Refs.GetByAncs(tryStmts);
+   sanitizers.Add(Find_AWSLambda_S3_Objects_Sanitizers().GetLeftmostTarget());


    // Remove sanitized s3DbInMethods
-   s3DbInMethods -= s3DbInMethods.InfluencedBy(s3RefsInTryBlock).GetLastNodesInPath();
+   s3DbInMethods -= s3DbInMethods.InfluencedBy(sanitizers).GetLastNodesInPath();


    // Key parameter used to identify objects from s3
    CxList paramsOfRelevantMethods = parameters.GetParameters(s3DbInMethods);
```

Python / Python_High_Risk / Code_Injection

Code changes

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


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

```
 CxList systemJs = All.FindByShortName("system_js");
 CxList systemJsMembers = systemJs.GetMembersOfTarget();
 CxList systemJsMemberAccess = systemJsMembers.FindByType(typeof(MemberAccess)).InfluencedBy(inputs);
@@ -17,3 +17,4 @@
 }


 result.Add(inputs.DataInfluencingOn(dynamicMethodInvoke));
+result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Python / Python_High_Risk / Command_Injection

Code changes

```
---
+++
@@ -1,5 +1,5 @@
 // sources
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());


 CxList imports = Find_Imports();
 CxList inputsInLeftOfAssign = inputs.FindByAssignmentSide(CxList.AssignmentSide.Left);
```

Python / Python_High_Risk / Connection_String_Injection

Code changes

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


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

Python / Python_High_Risk / LDAP_Injection

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Interactive_Inputs();
+CxList inputs = All.NewCxList(Find_Interactive_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 +1,5 @@
 // user input -> unvalidated/unsanitized -> dynamic module import
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());
 CxList methods = Find_Methods();
 CxList customAtts = Find_CustomAttribute();
 CxList unkRefs = Find_UnknownReference();
```

Python / Python_High_Risk / Reflected_XSS_All_Clients

Code changes

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

 CxList sanitized = Find_XSS_Sanitize();
```

Python / Python_High_Risk / Second_Order_SQL_Injection

Code changes

```
---
+++
@@ -1,9 +1,8 @@
-CxList dbOutRead = All.NewCxList();
-dbOutRead.Add(Find_DB_Out(), Find_Read());
+CxList dbOutRead = All.NewCxList(Find_DB_Out(), Find_Read(), Find_Cloud_Storage_Inputs());

 CxList dbIn = Find_SQL_DB_In();

-CxList sanitize = Find_Sanitize();
+CxList sanitize = Find_SQL_Sanitize();

 CxList methods = Find_Methods();

@@ -14,8 +13,12 @@
 CxList pyodbcRefs = refs.InfluencedBy(pyodbcMethods).GetLastNodesInPath();
 pyodbcRefs -=  dbIn.FindByParameters(Find_BinaryExpr()).GetLastNodesInPath();

+CxList execute = Find_DB_In_MySQL().FindByShortName("execute")
+   .FilterByDomProperty<MethodInvokeExpr>(_ => _.Parameters.Count > 1);
+
 sanitize.Add(Find_Sanitize_Django_ORM(),
    Find_Sanitize_Flask_SQL_Alchemy(),
-   pyodbcRefs);
+   pyodbcRefs,
+   execute);
```

```
 result = dbOutRead.InfluencingOnAndNotSanitized(dbIn, sanitize);
```

Python / Python_High_Risk / SQL_Injection

Code changes

```
---
+++
@@ -1,2 +1,5 @@
-CxList sqlInjection = Common_High_Risk.SQL_Injection();
-result = sqlInjection.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+CxList db = Find_SQL_DB_In();
+CxList inputs = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
+CxList sanitized = Find_SQL_Sanitize();
+
+result = inputs.InfluencingOnAndNotSanitized(db, sanitized).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Python / Python_High_Risk / Stored_XSS

Code changes

```
---
+++
@@ -1,14 +1,7 @@
-CxList db = Find_DB_Out();
-CxList read = Find_Read();
-CxList remoteRequests = Find_Remote_Requests();
-
-CxList inputs = All.NewCxList();
-inputs.Add(db, read, remoteRequests);
+CxList inputs = All.NewCxList(Find_DB_Out(), Find_Read(), Find_Remote_Requests(), Find_Cloud_Storage_Inputs());

 CxList outputs = Find_XSS_Outputs();
-CxList sanitize = Find_XSS_Sanitize();
-sanitize.Add(Find_Remote_Requests_Sanitize());
-sanitize.Add(Find_Base64_Encode());
+CxList sanitize = All.NewCxList(Find_XSS_Sanitize(), Find_Remote_Requests_Sanitize(), Find_Base64_Encode());

 result = inputs.InfluencingOnAndNotSanitized(outputs, sanitize, CxList.InfluenceAlgorithmCalculation.NewAlgorithm);
 result = result.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Python / Python_High_Risk / Unsafe_Deserialization

Code changes

```
---
+++
@@ -2,22 +2,21 @@

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

-var pickleMethodNames = new List<string>{"load", "loads", "noload", "Unpickler"};
+List<string> pickleMethodNames = new List<string>{"load", "loads", "noload", "Unpickler"};
 CxList pickleMemberAccess = methods.FindByMemberAccess("pickle.*");
```

```
 insecureMethods.Add(pickleMemberAccess.FindByShortNames(pickleMethodNames));
 insecureMethods.Add(methods.FindByMemberAccess("shelve.open"));

-var hashMethodNames = new List<string>{"md5", "sha256"};
+List<string> hashMethodNames = new List<string>{"md5", "sha256"};
 CxList hashMemberAccess = methods.FindByMemberAccess("hashlib.*");
 CxList hash = hashMemberAccess.FindByShortNames(hashMethodNames);

-CxList IfsWithInsecureMethods = insecureMethods.GetAncOfType(typeof(IfStmt));
-CxList conditions = All.FindByFathers(IfsWithInsecureMethods).FindByType(typeof(Expression));
-CxList safeConditions = conditions.DataInfluencedBy(hash).GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
-insecureMethods -= insecureMethods.GetByAncs(safeConditions.GetAncOfType(typeof(IfStmt)));
+CxList IfsWithInsecureMethods = insecureMethods.GetAncOfType<IfStmt>();
+CxList conditions = All.FindByFathers(IfsWithInsecureMethods).FindByType<Expression>();
+CxList safeConditions = conditions.DataInfluencedBy(hash).GetLastNodesInPath();
+insecureMethods -= insecureMethods.GetByAncs(safeConditions.GetAncOfType<IfStmt>());

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

 CxList sanitizers = Find_Sanitize();
```

Python / Python_High_Risk / XPath_Injection

Code changes

```
---
+++
@@ -9,7 +9,7 @@
 CxList lxml = Find_Methods_By_Import("lxml*", lxmlMethods, imports);

 //ElementTree
-String[] elemTree = new string[]{"*find","*findall","*findtext", "*iterfind"};
+String[] elemTree = new string[]{"*iterfind"};
 CxList elementTree = Find_Methods_By_Import("xml.etree.ElementTree", elemTree, imports);

 //py-dom-xpath
@@ -19,18 +19,21 @@
 String[] pyxml = new string[]{"*Evaluate"};
 CxList pyXml = Find_Methods_By_Import("xml", pyxml, imports);

+CxList xPath = All.NewCxList(
+   libXml2,
+   lxml,
+   pyDomXpath,
+   pyXml,
+   elementTree);
```

```
-CxList xPath = All.NewCxList();
-xPath.Add(libXml2);
-xPath.Add(lxml);
-xPath.Add(pyDomXpath);
-xPath.Add(pyXml);
-xPath.Add(elementTree);
+CxList xPathParams = All.GetParameters(xPath);

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

-CxList sanitized = Find_Sanitize();
-sanitized.Add(Find_PY_DOM_XPath_Sanitizers(pyDomXpath));
-sanitized.Add(Find_Base64_Encode());
+CxList sanitized = All.NewCxList(
+   Find_Sanitize(),
+   Find_PY_DOM_XPath_Sanitizers(pyDomXpath),
+   Find_Base64_Encode());

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

Python / Python_Medium_Threat / Communication_Over_HTTP

Code changes

```
---
+++
@@ -11,7 +11,7 @@
 CxList httpStrings = Find_Strings().FilterByDomProperty<StringLiteral>(s =>
     s.Value.StartsWith("http") && !s.Value.StartsWith("https"));

-inputs.Add(httpStrings);
+inputs.Add(httpStrings, Find_Cloud_Interactive_Inputs());

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

Python / Python_Medium_Threat / Cookie_Poisoning

Code changes

```
---
+++
@@ -2,7 +2,7 @@
 CxList unkRefs = Find_UnknownReference();
 CxList cookies = Find_Cookies();
 //Find_Inputs() returns Find_Cookies()
-CxList inputs = Find_Inputs() - cookies;
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs()) - cookies;
 //Find all the cookies references.
```

```
 CxList allRefsCookies = All.FindAllReferences(cookies);
 CxList indexerRefs = allRefsCookies.FindByType<IndexerRef>();
```

Python / Python_Medium_Threat / Django_Missing_Object_Level_Authorization

Code changes

```
---
+++
@@ -1,4 +1,4 @@
-CxList inputs = Find_Django_Inputs().GetAncOfType<MethodDecl>();
+CxList inputs = All.NewCxList(Find_Django_Inputs().GetAncOfType<MethodDecl>(), Find_Cloud_Interactive_Inputs());
 CxList sinks = Find_DB_In_Django();
 sinks.Add(Find_Django_QuerySet_Methods());
```

Python / Python_Medium_Threat / Hardcoded_Password_in_Connection_String

Code changes

```
---
+++
@@ -1,10 +1,10 @@
 // Find the string literals containing "password"
 CxList psw = Find_Password_Strings(100);
+
 CxList emptyString = Find_Empty_Strings();
 CxList nullLiteral = All.FindByShortName("none", false);
-CxList emptyStringNull = All.NewCxList();
-emptyStringNull.Add(emptyString, nullLiteral);
+CxList emptyStringNull = All.NewCxList(emptyString, nullLiteral);

 CxList strLiterals = Find_Strings() - emptyStringNull;

@@ -13,19 +13,18 @@
 CxList connection = methods.FindByShortName("*connect*", false);
 connection.Add(
     Find_DB_Conn_Strings(),
-    Find_Methods_By_Import("requests", new string [] {"post", "put"}));
+    Find_Methods_By_Import("requests", new [] {"post", "put"}));

 result = connection.DataInfluencedBy(psw).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
+
+CxList ancsPsw = All.GetByAncs(All.GetParameters(connection)
+    .FindByType<ArrayInitializer>()).FindByType<UnknownReference>() * psw;

 // Find connection strings that contain a password in their initialization
 CxList connetionParam2 = All.GetParameters(connection, 2);
 CxList connetionParam1 = All.GetParameters(connection, 1);

-CxList ancsPsw = All.GetByAncs(All.GetParameters(connection)
```

```
-   .FindByType<ArrayInitializer>()).FindByType<UnknownReference>() * psw;
-
-CxList connetionParams = All.NewCxList();
-connetionParams.Add(connetionParam1, connetionParam2);
+CxList connetionParams = All.NewCxList(connetionParam1, connetionParam2);


 CxList pwdInConnectioParam = strLiterals.GetByAncs(connetionParams * psw);
 pwdInConnectioParam.Add(ancsPsw);
@@ -51,6 +50,29 @@
 string urlRegexStr2 = @"(mongodb|amqp)://[^\s]+:[^\s]+@";
 System.Text.RegularExpressions.Regex urlRegexCompiled2 = new System.Text.RegularExpressions.Regex(urlRegexStr2);


+
+// Multiline strings are converted into concatenations, so they need to be taken into consideration as well
+CxList possibleFormatTargets = All.NewCxList(Find_BinaryExpr().GetByBinaryOperator(BinaryOperator.Add), strLiterals);
+
+CxList extraCases = All.NewCxList(
+   // Used directly in a function as parameter
+   Find_Param().CxSelectDomProperty<Param>(x=>x.Value),
+   // Used in format construct  ( % operator )
+   possibleFormatTargets.FilterByDomProperty<CSharpGraph>(el => el._father is MemberAccess));
+
+// Remove sub-concatenations to remove amount of results
+extraCases -= extraCases * result.GetStartAndEndNodes(CxList.GetStartEndNodesType.AllNodes);
+// Remove single UnkRef nodes
+extraCases -= extraCases.FindByType<UnknownReference>();
+
+// Grab the elements using absint
+result.Add(extraCases.FindByAbstractValue(x =>
+   x is StringAbstractValue abVal && abVal.Content.Length < 1000
+       && (urlRegexCompiled.Match(abVal.Content).Success || urlRegexCompiled2.Match(abVal.Content).Success)
+));
+
+// Now look to other string lionterals
+strLiterals -= strLiterals.GetByAncs(extraCases);
 foreach (CxList suspectedString in strLiterals)
 {
    CSharpGraph gr = suspectedString.TryGetCSharpGraph<CSharpGraph>();
@@ -58,11 +80,9 @@
    {
        continue;
    }
-   if (gr.ShortName.Length < 1000)
+   if (gr.ShortName.Length < 1000 &&
+       (urlRegexCompiled.Match(gr.ShortName).Success || urlRegexCompiled2.Match(gr.ShortName).Success))
    {
-       if (urlRegexCompiled.Match(gr.ShortName).Success || urlRegexCompiled2.Match(gr.ShortName).Success)
-       {
```

```
-           result.Add(suspectedString);
-       }
+               result.Add(suspectedString);
    }
 }
```

Python / Python_Medium_Threat / Header_Injection

Code changes

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

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

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

Python / Python_Medium_Threat / Object_Access_Violation

Code changes

```
---
+++
@@ -1,7 +1,7 @@
 //Finds unsanatized user inputs to attribute functions
 CxList methods = Find_Methods();
 CxList stringLiterals = Find_String_Literal();
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());
 CxList sanitize = Find_Sanitize();
 sanitize.Add(Find_Base64_Encode());
```

Python / Python_Medium_Threat / Open_Redirect

Code changes

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

 CxList sanitizers = Find_Open_Redirect_Sanitizers(redirects);
```

Python / Python_Medium_Threat / Parameter_Tampering

Code changes

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

```
@@ -1,11 +1,11 @@
-CxList input = Find_Interactive_Inputs();
+CxList input = All.NewCxList(Find_Interactive_Inputs(), Find_Cloud_Interactive_Inputs());
 CxList db = Find_DB_In();
 CxList strings = Find_Strings();



 List<string> tablesNames = new List<string>{
        "*orders*","*credit*","*invoice*","*booking*","*bill*",
-       "*payment*","*account*", "*cash*","*customer*" };
+       "*payment*","*account*", "*cash*","*customer*","*client*"};

 CxList tables = All.FindByShortNames(tablesNames, false);

@@ -28,9 +28,11 @@
 CxList db3 = db.DataInfluencedBy(Select).DataInfluencedBy(Where);
 db3 -= db3.DataInfluencedBy(And);
 CxList sanitize = Find_Parameter_Tampering_Sanitize();
+CxList dbAll = All.NewCxList(db2, db3);

-CxList dbAll = All.NewCxList();
-dbAll.Add(db2);
-dbAll.Add(db3);
+result.Add(dbAll.InfluencedByAndNotSanitized(input, sanitize));
-result = dbAll.InfluencedByAndNotSanitized(input, sanitize);
+if(Find_Methods_By_Import("marshmallow", new string[]{"Schema"}).Count > 0){
+   CxList cleanInputs = sanitize.InfluencedBy(input).GetFirstNodesInPath();
+   result = result - result.InfluencedBy(cleanInputs);
+}
```

Python / Python_Medium_Threat / Path_Traversal

Code changes

```
---
+++
@@ -1,7 +1,8 @@
-CxList inputs = Find_Interactive_Inputs();
-inputs.Add(Find_Bas_Server_Inputs());
+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 membersOfOsPath = Find_UnknownReference().FindByShortName("os")
+CxList membersOfOsPath = unknRefs.FindByShortName("os")
     .GetMembersOfTarget().FindByShortName("path").GetMembersOfTarget();
 CxList osPathSinks = membersOfOsPath.FindByShortNames(new string[]{"exists", "isfile", "isdir"});


@@ -12,35 +13,31 @@
```

```
 obj.Add(methods);


 //File access
-CxList files = obj.FindByName("open");
-files.Add(obj.FindByName("file"));
-files.Add(Find_Methods_By_Import("io", new string[]{"FileIO", "open", "BufferedReader", "BufferedRandom",
-   "BufferedRWPair", "TextIOBase", "TextIOWrapper"}));
-files.Add(Find_Methods_By_Import("codecs", new string[]{"StreamReader", "StreamReaderWriter", "StreamRecoder"}));
-files.Add(Find_Methods_By_Import("fileinput", new string[]{"input", "FileInput"}));
-files.Add(Find_Methods_By_Import("linecache", new string[]{"getline"}));
-
-//Directory access
-files.Add(Find_Methods_By_Import("os.path", new string[]{"join", "walk"}));
-files.Add(Find_Methods_By_Import("macpath", new string[]{"walks"}));
-files.Add(Find_Methods_By_Import("dircache", new string[]{"listdir", "opendir"}));
-files.Add(Find_Methods_By_Import("glob", new string[]{"glob", "iglob"}));
-files.Add(Find_Methods_By_Import("fnmatch", new string[]{"filter"}));
-
-//pathlib
-files.Add(Find_Methods_By_Import("pathlib", new string[]{
-   "Path", "PurePath", "PosixPath", "PurePosixPath", "WindowsPath", "PureWindowsPath"}));
+CxList files = All.NewCxList(
+   obj.FindByNames("open", "file"),
+   Find_Methods_By_Import("io", new string[]{"FileIO", "open", "BufferedReader", "BufferedRandom",
+   "BufferedRWPair", "TextIOBase", "TextIOWrapper"}),
+   Find_Methods_By_Import("codecs", new string[]{"StreamReader", "StreamReaderWriter", "StreamRecoder"}),
+   Find_Methods_By_Import("fileinput", new string[]{"input", "FileInput"}),
+   Find_Methods_By_Import("linecache", new string[]{"getline"}),
+   //Directory access
+   Find_Methods_By_Import("os.path", new string[]{"join", "walk"}),
+   Find_Methods_By_Import("macpath", new string[]{"walks"}),
+   Find_Methods_By_Import("dircache", new string[]{"listdir", "opendir"}),
+   Find_Methods_By_Import("glob", new string[]{"glob", "iglob"}),
+   Find_Methods_By_Import("fnmatch", new string[]{"filter"}),
+   //pathlib
+   Find_Methods_By_Import("pathlib", new string[]{
+   "Path", "PurePath", "PosixPath", "PurePosixPath", "WindowsPath", "PureWindowsPath"})
+   );


 CxList filesMethodsAll = files.GetMembersOfTarget();


-CxList filesMethods = filesMethodsAll.FindByShortName("Close");
-filesMethods.Add(filesMethodsAll.FindByShortName("Dispose"));
-
+CxList filesMethods = filesMethodsAll.FindByShortNames("Close", "Dispose");
 files -= filesMethods.GetTargetOfMembers();


 //Add Pandas Support Any file input/output that contains unsanitized user input should be considered as a flow sink:
```

```
-files.Add(pandasSink);
-files.Add(osPathSinks);
+files.Add(pandasSink, osPathSinks);

 CxList sanitized = Find_Integers();
 sanitized.Add(Find_Methods_By_Import("os.path", new string[]{"abspath", "basename", "commonprefix",
@@ -48,8 +45,8 @@

 //Formatting strings with % and format() should be considered sanitizers
 CxList format = Find_BinaryExpr().FindByRegex(@"%");
-sanitized.Add(All.GetByAncs(format).FindByType(typeof(UnknownReference)));
-sanitized.Add(methods.FindByShortName("format"));
-sanitized.Add(Find_Sanitize_Pandas());
+sanitized.Add(unknRefs.FindDescendantsOfType<UnknownReference>(format),
+   methods.FindByShortName("format"),
+   Find_Sanitize_Pandas());

 result = files.InfluencedByAndNotSanitized(inputs, sanitized).ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
```

Python / Python_Medium_Threat / Privacy_Violation

Code changes

```
---
+++
@@ -4,19 +4,13 @@
 inputs.Add(Find_DB_Out());

 CxList strings = Find_Strings();
-CxList integerLiteral = Find_IntegerLiterals();
-CxList nullLiteral = Find_NullLiteral();
-CxList literals = All.NewCxList();
-literals.Add(strings);
-literals.Add(integerLiteral);
-literals.Add(nullLiteral); // nullLiterals are treated separately

-// Find names that are suspected to be personal info, eg. String PASSWORD, Integer SSN, and remove string literals, such as x="password"
+// 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;

 //limit to personal info influenced by input
-personal_info = personal_info.DataInfluencedBy(inputs).GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly) +
-   personal_info * inputs;
+personal_info = personal_info.DataInfluencedBy(inputs).GetLastNodesInPath() + personal_info * inputs;
 // 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)
@@ -53,28 +47,32 @@
```

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

 // Define sanitize
 CxList sanitize = Find_DB(); // in some languages is called Find_DB, Find_DB_In, Find_DB_Input
 sanitize.Add(Find_DB_Conn_Strings());
+
 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());
 encrypt -= Find_Decrypt();
+
 CxList encoded = All.FindByShortName("*Encode*", false); // all variables labelled encode(ed) are considered safe
-CxList encRemove = encoded.FindByShortName("*UnEncode*", false);;
-encRemove.Add(encoded.FindByShortName("*Decode*", false));
-encRemove.Add(encoded.FindByShortName("*URLEncode*", false)); // URLEncode method is a part of .NET and is not a sanitizer
-encRemove.Add(encoded.FindByShortName("*HTMLEncode*", false)); // HTMLEncode method is a part of .NET and is not a sanitizer
-encRemove.Add(encoded.FindByShortName("*EncodeHTML*", false));
+CxList encRemove = encoded.FindByShortNames(
+   new 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);
 // base64.encondestring is not a sanitizer
 encRemove.Add(Find_Methods_By_Import("base64", new string[] {"encodestring"}));

 encoded -= encRemove;
-sanitize.Add(encrypt);
-sanitize.Add(encoded);
+sanitize.Add(encrypt, encoded);

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

Python / Python_Medium_Threat / ReDoS_In_Replace

Code changes

```
---
+++
@@ -1,6 +1,6 @@
 CxList replace = Find_Replace();
```

```
 CxList evilStrings = Find_Evil_Strings();
-CxList inputs = Find_Inputs();
+CxList inputs = All.NewCxList(Find_Inputs(), Find_Cloud_Inputs());

 CxList inputs_vars = inputs.GetAncOfType(typeof(AssignExpr)); // Assign or declare
 inputs_vars.Add(inputs.GetAncOfType(typeof(Declarator)));
```

Python / Python_Medium_Threat / SSRF

Code changes

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

Code changes

```
---
+++
@@ -1,12 +1,10 @@
 // sources
-CxList inputs = Find_DB_Out();
-inputs.Add(Find_Read());
-inputs.Add(Find_Remote_Requests());
+CxList inputs = All.NewCxList(Find_DB_Out(), Find_Read(), Find_Remote_Requests(), Find_Cloud_Storage_Inputs());

 // sanitizers
-CxList sanitize = Find_Command_Injection_Sanitize();
-sanitize.Add(Find_Remote_Requests_Sanitize());
-sanitize.Add(Find_Base64_Encode());
+CxList sanitize = All.NewCxList(Find_Command_Injection_Sanitize(),
+   Find_Remote_Requests_Sanitize(),
+   Find_Base64_Encode());

 // sinks
 CxList commands = Find_Command_Execution();
```

Python / Python_Medium_Threat / Stored_LDAP_Injection

Code changes

```
---
+++
@@ -1,11 +1,8 @@
-CxList inputs = Find_Read();
-inputs.Add(Find_DB_Out());
-inputs.Add(Find_Remote_Requests());
```

```
+CxList inputs = All.NewCxList(Find_Read(), Find_DB_Out(), Find_Remote_Requests(), Find_Cloud_Storage_Inputs());


-
-CxList sanitize = Find_LDAP_Sanitize();
-sanitize.Add(Find_Remote_Requests_Sanitize());
-sanitize.Add(Find_Base64_Encode());
+CxList sanitize = All.NewCxList(Find_LDAP_Sanitize(),
+   Find_Remote_Requests_Sanitize(),
+   Find_Base64_Encode());
 CxList outputs = Find_LDAP_Inputs();


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

Python / Python_Medium_Threat / Uncontrolled_Format_String

Code changes

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