



Scanning HacmeBank Web-Services

F1 ‘How-To Series’

Ounce Labs, Inc.
100 Fifth Avenue
Waltham, MA
(781) 290-5333

OVERVIEW AND OBJECTIVES

INTRODUCTION

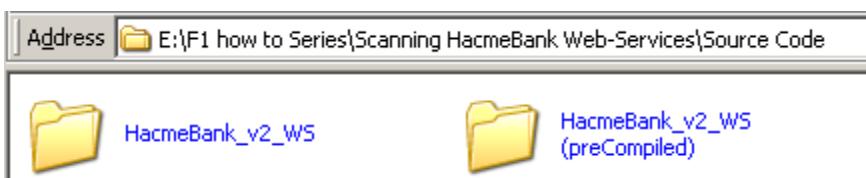
This document contains detailed technical information on how to use Ounce's technology (CORE + F1) to analyse the HacmeBank's Web-services.

SUPPORT MATERIALS

The *Scanning HacmeBank Web-Services.zip* file contains all materials required for this how to

Address E:\F1 how to Series\Scanning HacmeBank Web-Services			
Name	Size	Type	
Custom F1 Scripts		File Folder	
F1-Add-ons required for this How-to		File Folder	
Saved Assessments		File Folder	
Source Code		File Folder	
F1 How To - Scanning HacmeBank WebServices v0.5.doc	3,485 KB	Microsoft Word Doc...	

The Source Code of the HacmeBank WebServices and the respective precompiled directory:



The assessment files that will be created

Address E:\F1 how to Series\Scanning HacmeBank Web-Services\Saved Assessments			
Name	Size	Type	
HacmeBank_Ws - after all Callbacks.xml	181 KB	XML Document	
HacmeBank_Ws - after making ChangeUserPassword a callback.xml	98 KB	XML Document	
HacmeBank_Ws 1stScan.xml	94 KB	XML Document	

The script used to manually create Callbacks



The F1 Add-ons required:

Name	Size	Type	Date Modified
CSharp_Scripts		File Folder	6/27/2008 1:05 PM
dotNet_CallbacksMaker_5_03		File Folder	6/27/2008 1:18 PM
Scan_5_03		File Folder	6/27/2008 1:03 PM
TextSearch		File Folder	6/27/2008 1:13 PM
ViewAssessmentRun		File Folder	6/27/2008 1:06 PM
f1AddOn_CSharp_Scripts.exe	2 KB	Shortcut	6/27/2008 1:05 PM
f1AddOn_dotNet_CallbacksMaker.exe	2 KB	Shortcut	6/27/2008 1:07 PM
f1AddOn_Scan_5_03.exe	2 KB	Shortcut	6/27/2008 1:03 PM
f1AddOn_TextSearch.exe	2 KB	Shortcut	6/27/2008 1:13 PM
f1AddOn_ViewAssessmentRun_5_03.exe	2 KB	Shortcut	6/27/2008 1:06 PM

Each F1 Add-on is packaged as a unit (i.e. with all dependencies included) and is designed to be executed independently:

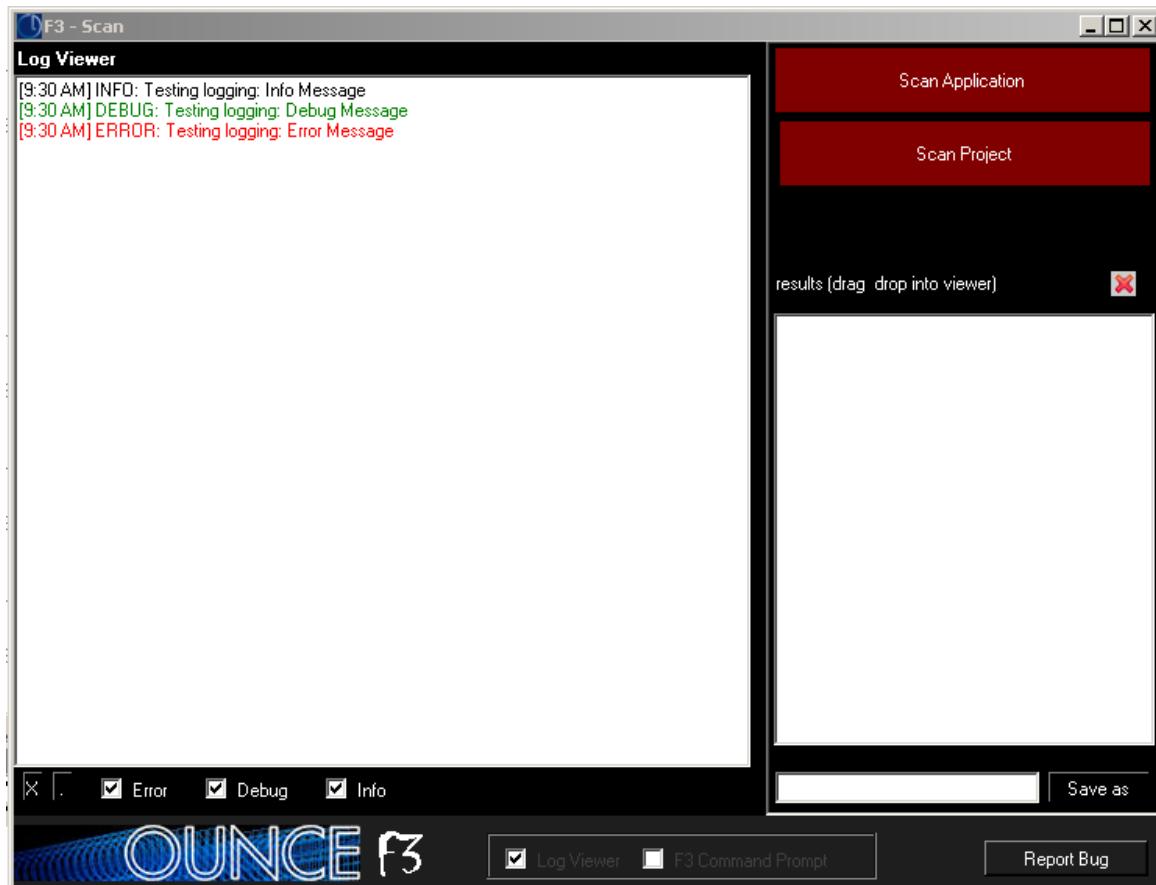
Name	Size	Type	Date Modified
F1AddOn_CustomRules_5_03...	56 KB	Application	6/27/2008 12:29 AM
F1AddOn_DataViewers.exe	84 KB	Application	6/27/2008 12:29 AM
F1AddOn_dotNet_CallbacksM...	36 KB	Application	6/27/2008 12:29 AM
F1AddOn_ounceMySql_5_03....	52 KB	Application	6/27/2008 12:29 AM
F1_Core.Lib.dll	372 KB	Application Extension	6/27/2008 12:29 AM
MySQL.Data.dll	260 KB	Application Extension	6/27/2008 12:29 AM
F1_Core.Lib.pdb	724 KB	Program Debug Dat...	6/27/2008 12:29 AM
F1AddOn_CustomRules_5_03...	76 KB	Program Debug Dat...	6/27/2008 12:29 AM
F1AddOn_DataViewers.pdb	134 KB	Program Debug Dat...	6/27/2008 12:29 AM
F1AddOn_dotNet_CallbacksM...	36 KB	Program Debug Dat...	6/27/2008 12:29 AM
F1AddOn_ounceMySql_5_03....	56 KB	Program Debug Dat...	6/27/2008 12:29 AM

SCANNING WEBSITE

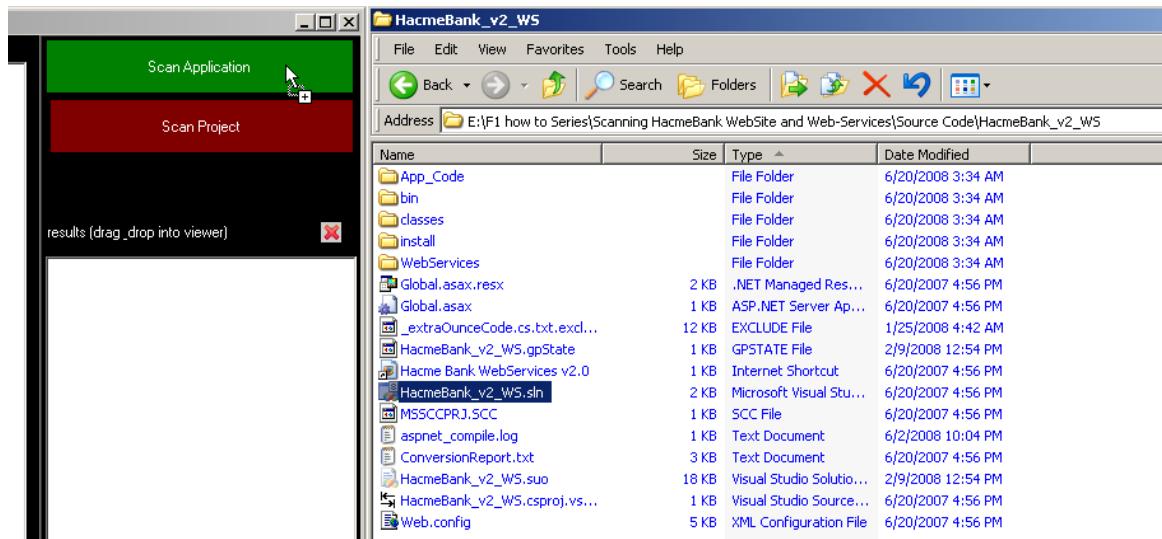
Open f1AddOn_Scan_5.03.exe



And that will open the scan interface in 'F3 Mode'



Drag & Drop the Solution file into the 'Scan Application' Box:



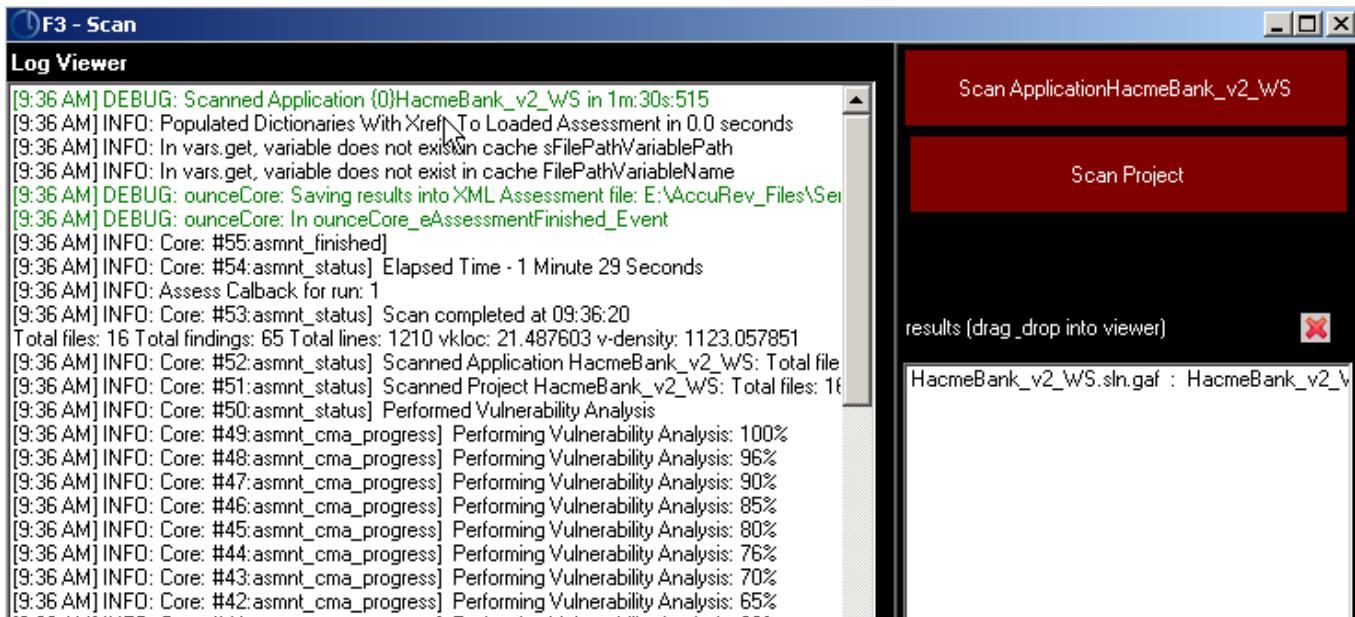
This will trigger the scan

```

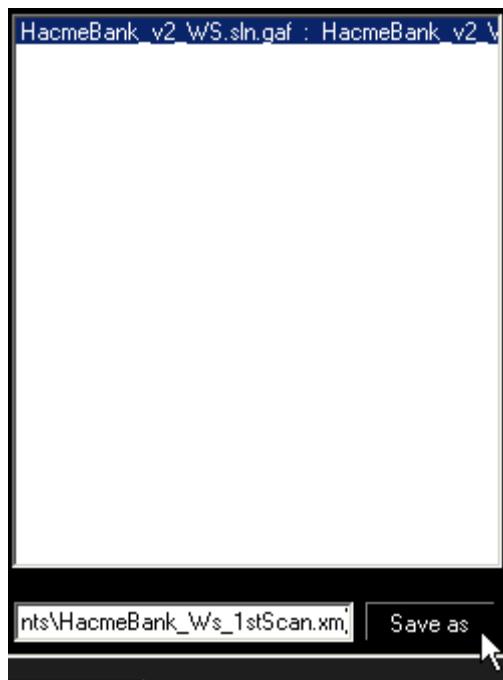
[F3 - Scan]
Log Viewer
[9:34 AM] INFO: Core: #2:asmnt_status] Scanning Project HacmeBank_v2_WS (1 of 1)
[9:34 AM] INFO: Core: #1:asmnt_status] New Scan started at 09:34:51
[9:34 AM] INFO: Assesment Started, ID:0
[9:34 AM] INFO: Scanning Assesment: E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS
[9:34 AM] DEBUG: Adding Callbacks
[9:34 AM] DEBUG: All ok, user logged in
[9:34 AM] DEBUG: Configuring connection to Ounce Core and logging in
[9:34 AM] DEBUG: Service Status: Running
[9:34 AM] DEBUG: Service is running, so trying to start it
[9:34 AM] DEBUG: Service Status: Stopped
[9:34 AM] DEBUG: Service is started, so trying to stop it
[9:34 AM] INFO: Restarting Service: Ounce Core
[9:34 AM] INFO: In vars.get, variable does not exist in cache fFactory
[9:34 AM] INFO: In vars.get, variable does not exist in cache raRunAe
[9:34 AM] DEBUG: Creating message with command Text: DndQueue.ascx_DndQueue.registerDnl
[9:30 AM] INFO: Testing logging: Info Message
[9:30 AM] DEBUG: Testing logging: Debug Message
[9:30 AM] ERROR: Testing logging: Error Message

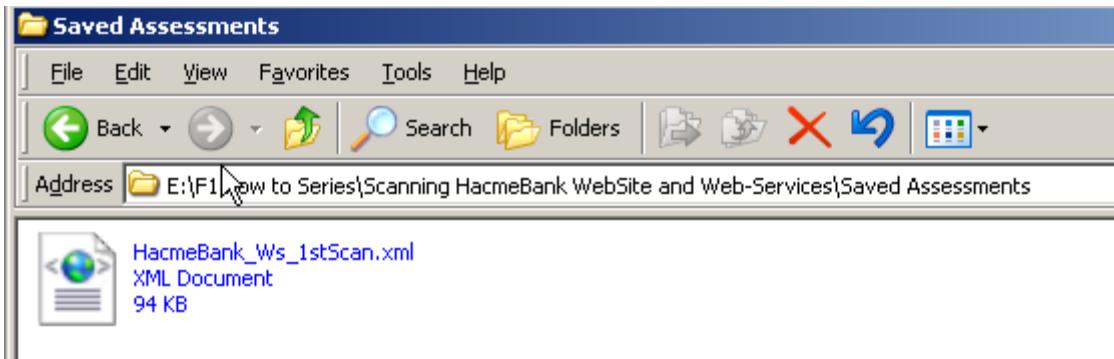
```

When the scan is completed:



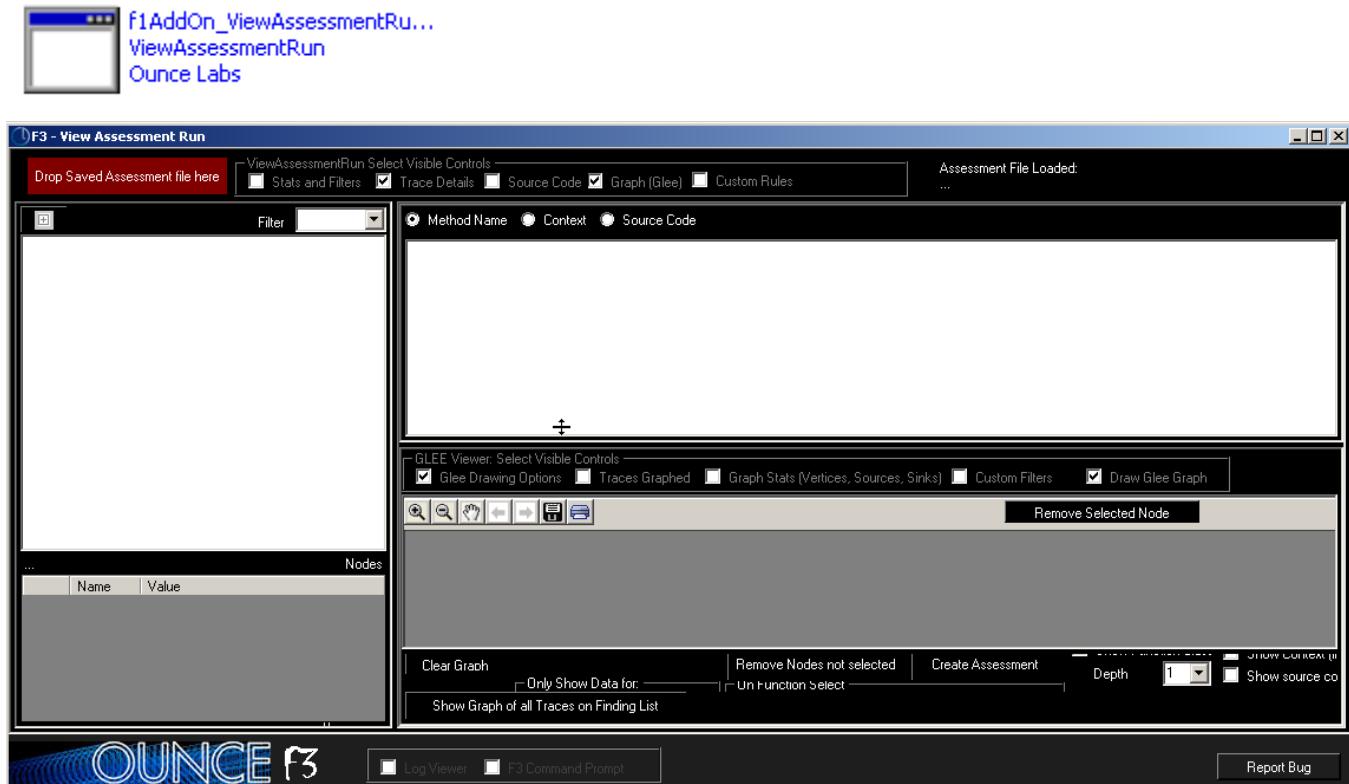
To save the assessment, select the entry in the results list box, enter the path and filename and click on 'Save As'



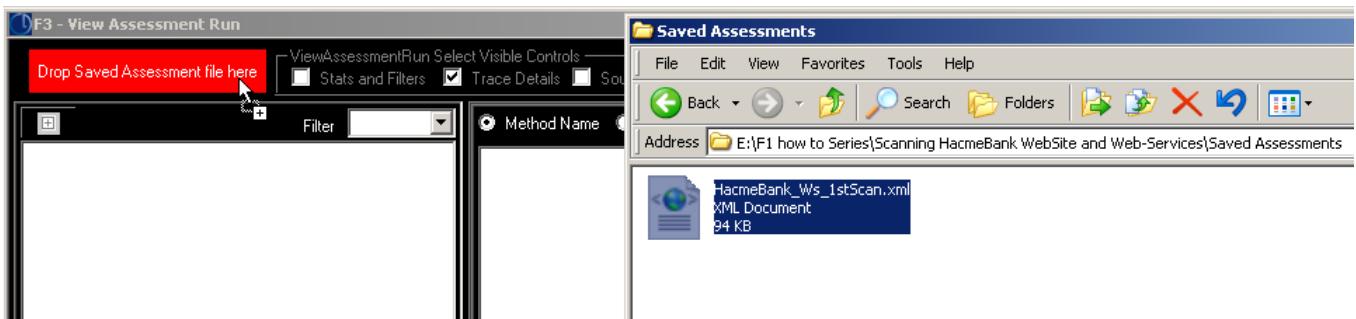


VIEWING SAVED ASSESSMENT:

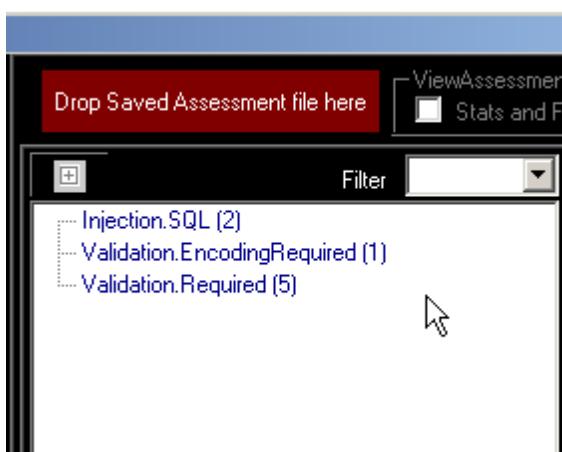
Open f1AddOn_ViewAssessmentRun.exe



Drag and drop the previously created Saved assessment file into the 'Drop Saved Assessment file here' area:

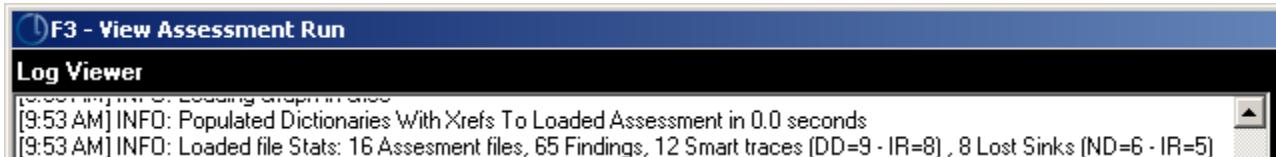


This will load the saved assessment file which in this first pass will have 7 unique traces:

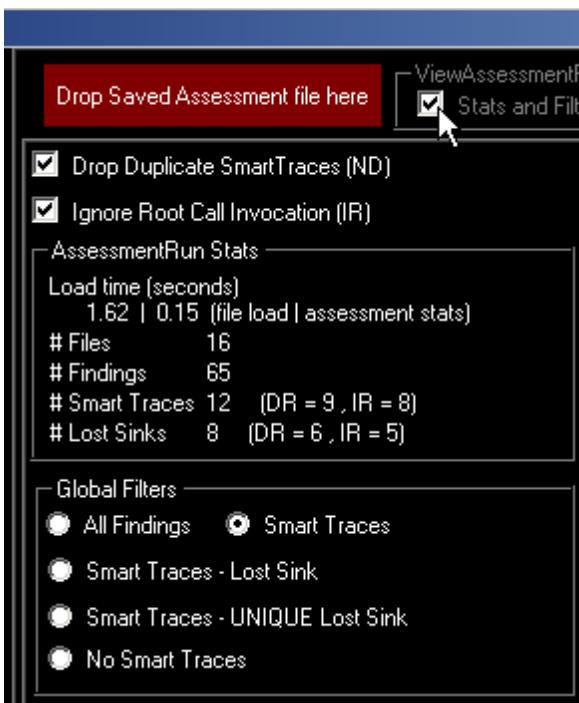
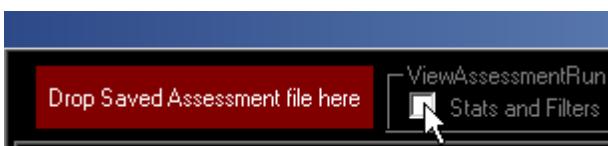


To see the log viewer click on the 'Log Viewer' checkbox at the bottom of the F3 gui:





[9:53 AM] INFO: Populated Dictionaries With Xrefs To Loaded Assessment in 0.0 seconds
[9:53 AM] INFO: Loaded file Stats: 16 Assessment files, 65 Findings, 12 Smart traces (DD=9 - IR=8) , 8 Lost Sinks (ND=6 - IR=5)
(Explain how to read the stats)



Drop Duplicate SmartTraces (ND)

Ignore Root Call Invocation (IR)

AssessmentRun Stats

Load time (seconds)
1.62 | 0.15 (file load | assessment stats)

Files 16
Findings 65
Smart Traces 12 (DR = 9, IR = 8)
Lost Sinks 8 (DR = 6, IR = 5)

Global Filters

- All Findings Smart Traces
- Smart Traces - Lost Sink
- Smart Traces - UNIQUE Lost Sink
- No Smart Traces

AccessControl (1)
Authentication.Credentials.Unprotected (1)
ErrorHandling.RevealDetails.Message (1)
Info (39)
Injection.SQL (6)
Logging.Required (1)
Malicious.Trigger (7)
Validation.EncodingRequired (1)
Validation.Required (8)

Vulnerability Type	Nodes
Name	Value
Validation.Required	5

Drop Saved Assessment file here

View Assessment Run Select Visible Controls

Stats and Filters Trace Details Source

Drop Duplicate SmartTraces (ND)

Ignore Root Call Invocation (IR)

AssessmentRun Stats

Load time (seconds)
1.62 | 0.15 (file load | assessment stats)

Files 16
Findings 65
Smart Traces 12 (DR = 9, IR = 8)
Lost Sinks 8 (DR = 6, IR = 5)

Global Filters

- All Findings Smart Traces
- Smart Traces - Lost Sink
- Smart Traces - UNIQUE Lost Sink
- No Smart Traces

Validation.Required (5)

Vulnerability Type	
Name	Value
Validation.Required	5

Drop Saved Assessment file here

ViewAssessmentRun Select Visible Controls
 Stats and Filters Trace Details Source Code

Drop Duplicate SmartTraces (ND)
 Ignore Root Call Invocation (IR)

AssessmentRun Stats

Load time (seconds)
1.62 | 0.15 (file load | assessment stats)

Files 16
Findings 65
Smart Traces 12 (DR = 9 , IR = 8)
Lost Sinks 8 (DR = 6 , IR = 5)

Global Filters

All Findings Smart Traces
 Smart Traces - Lost Sink
 Smart Traces - UNIQUE Lost Sink
 No Smart Traces

Validation.Required (5)

Vulnerability Type

Name	Value

Validation.Required (5)

- command1 . System.Data.SqlClient.SqlCommand.ExecuteReader ()

Validation.Required (5)

- command1 . System.Data.SqlClient.SqlCommand.ExecuteReader ()

Filter

vuln_type
caller_name
lost_sink
source
known_sink
source_code

Filter ▾

```

- System.Data.Common.DbDataReader.GetName(int):string
- System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.
- System.Decimal.op_Explicit(System.Decimal):int
- System.Decimal.ToString():string
- System.Object.GetType():System.Type

```



```

System.Data.Common.DbDataReader.GetName(int):string
System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.SqlDataReader
System.Decimal.op_Explicit(System.Decimal):int
System.Decimal.ToString():string
System.Object.GetType():System.Type

```

```

- System.Data.SqlClient.SqlCommand.ExecuteReader():System.
- System.Data.Common.DbDataReader.get_Item(int):object
- System.Object.ToString():string
- System.Collections.ArrayList.Add(object):int
- System.Collections.ArrayList.GetEnumerator():System.Collections.
- System.Collections.IEnumerator.get_Current():object
- HacmeBank_v2_WS.DataFactory.recalculateAccountBalance(
- System.String.Concat(string:string:string):string
- HacmeBank_v2_WS.SqlServerEngine.returnArrayListFromSt
- HacmeBank_v2_WS.SqlServerEngine.executeSQLCom
- System.Data.SqlClient.SqlCommand.SqlCommand(st)
- System.Data.SqlClient.SqlCommand.ExecuteReader()

```

GLEE Viewer: Select Visible Controls

Glee Drawing Options Traces Graphed Graph Stats (Vertices, S)

Remove Selected Node

recalculateAllAccountsBalances

ExecuteReader

Vulnerability Type

	Name	Value
▶	vuln Name	206911
	Vuln Type	Vulnerability.Validation.Required
	Context	command1 . System.Data.SqlClient.SqlCommand.ExecuteReader()
	Severity	2
	Confidence	3

Nodes

Clear Graph Remove Nodes not se
Only Show Data for: Un Function Select
Show Graph of all Traces on Finding List

F3 - View Assessment Run

Drop Saved Assessment file here

ViewAssessmentRun Select Visible Controls

Stats and Filters Trace Details Source Code Graph (Glee) Custom Rules

F3 - View Assessment Run

Drop Saved Assessment file here

ViewAssessmentRun Select Visible Controls

Assessment File Loaded: E:\F1 how to Scan\Scanning HacmeBank WebSite and Web-Services\Saved Assessments\HacmeBank_V2_1stScan.xml

Method Name Context Source Code

```
//SqlConnection Global.globalSqlServerConnection = new SqlConnection();
string text1 = sqlQueryToExecute;
SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
Global.globalSqlServerConnection.Open();
SqlDataReader executeReader_Result = command1.ExecuteReader();
return executeReader_Result;
```

public static ArrayList returnArrayListFromSQLQuery_containing_First()
{
 ArrayList QueryResults = new ArrayList();
 SqlDataReader reader1 = executeSQLCommand_returnSqlDataReader(sqlQueryToExecute);
 if (reader1.Read())
 do
 {
 QueryResults.Add(reader1[0].ToString());
 } while (reader1.NextResult());
 reader1.Close();
}

Vulnerability Type Nodes

Name	Value
vuln Name	206911
Vuln Type	Vulnerability.Validation.Required
Context	command1. System.Data.SqlClient.SqlCo...
Severity	2
Confidence	3

GLEE Viewer: Select Visible Controls

recalculateAllAccountsBalances

ExecuteReader

Clear Graph Remove Nodes not selected Create Assessment Depth 1 Show source code (inside node)

Show Graph of all Traces on Finding List

Log Viewer F3 Command Prompt Remove Selected Node Report Bug

Method Name Context Source Code

```
System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.SqlDataReader
-- System.Data.Common.DbDataReader.get_Item(int):object
-- System.Object.ToString():string
-- System.Collections.ArrayList.Add(object):int
System.Collections.ArrayList.GetEnumerator():System.Collections.IEnumerator
System.Collections.IEnumerator.get_Current():object
HacmeBank_v2_WS.DataFactory.calculateAccountBalance(string):string
-- System.String.Concat(string:string:string):string
HacmeBank_v2_WS.SqlServerEngine.returnArrayListFromSQLQuery_containing_FirstRow(string)
-- HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand_returnSqlDataReader(string)
-- System.Data.SqlClient.SqlCommand.SqlCommand(string,System.Data.SqlClient.SqlConnection):void
-- System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.SqlDataReader
```

```
Global.createServerConnection();
//SqlConnection Global.globalSqlServerConnection = new SqlConnection();
string text1 = sqlQueryToExecute;
SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
Global.globalSqlServerConnection.Open();
SqlDataReader executeReader_Result = command1.ExecuteReader();
return executeReader_Result;
```

public static ArrayList returnArrayListFromSQLQuery_containing_First()
{
 ArrayList QueryResults = new ArrayList();
 SqlDataReader reader1 = executeSQLCommand_returnSqlDataReader(sqlQueryToExecute);
 if (reader1.Read())
 do
 {
 QueryResults.Add(reader1[0].ToString());
 } while (reader1.NextResult());
 reader1.Close();
}

Line Numbers Tabs Spaces Invalid Lines EOL Markers HRuler VRuler

Multiple views

Method Name Context Source Code

- Method Name
- Context
- Source Code

HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand_returnSqlDataReader(string):System.Data.SqlClient.SqlDataReader

- > HacmeBank_v2_WS.DataFactory.calculateAllAccountsBalances():string
- > HacmeBank_v2_WS.SqlServerEngine.returnArrayListFromSQLQuery_containing_FirstFieldFromAllRows(string):System.Collections.ArrayList
- > HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand_returnSqlDataReader(string):System.Data.SqlClient.SqlDataReader
 - > System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.SqlDataReader
 - > System.Data.Common.DbDataReader.get_Item(int):object
 - > System.Object.ToString():string
 - > System.Collections.ArrayList.Add(object):int
- > System.Collections.ArrayList.GetEnumerator():System.Collections.IEnumerator
- > System.Collections.IEnumerator.get_Current():object

HacmeBank_v2_WS.DataFactory.calculateAccountBalance(string):string

- > System.String.Concat(string:string:string):string
- > HacmeBank_v2_WS.SqlServerEngine.returnArrayListFromSQLQuery_containing_FirstRow(string):System.Collections.ArrayList
 - > HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand_returnSqlDataReader(string):System.Data.SqlClient.SqlDataReader
 - > System.Data.SqlClient.SqlCommand.SqlCommand(string,System.Data.SqlClient.SqlConnection):void
 - > System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.SqlDataReader

By Context:



The screenshot shows a code editor interface with three filter buttons at the top: 'Method Name', 'Context', and 'Source Code'. The 'Context' button is highlighted with a black border. Below the buttons is a tree view of code. The root node is 'HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand_returnSqlDataReader(string)'. It has two children: 'bankAccounts = HacmeBank_v2_WS.SqlServerEngine.returnArrayListFromSQLQueryContaining_FirstFieldFromAllRows(sqlQuery)' and 'HacmeBank_v2_WS.DataFactory.calculateAccountBalance(account)'. The first child has a nested node 'reader1 = HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand_returnSqlDataReader(sqlQuery)'. This node has several properties and methods listed under it, such as 'executeReader_Result', 'command1', 'System.Data.SqlClient.SqlCommand.ExecuteReader()', etc. The second child also has some nested nodes and properties.

(note how the SQL injection is easier to read)



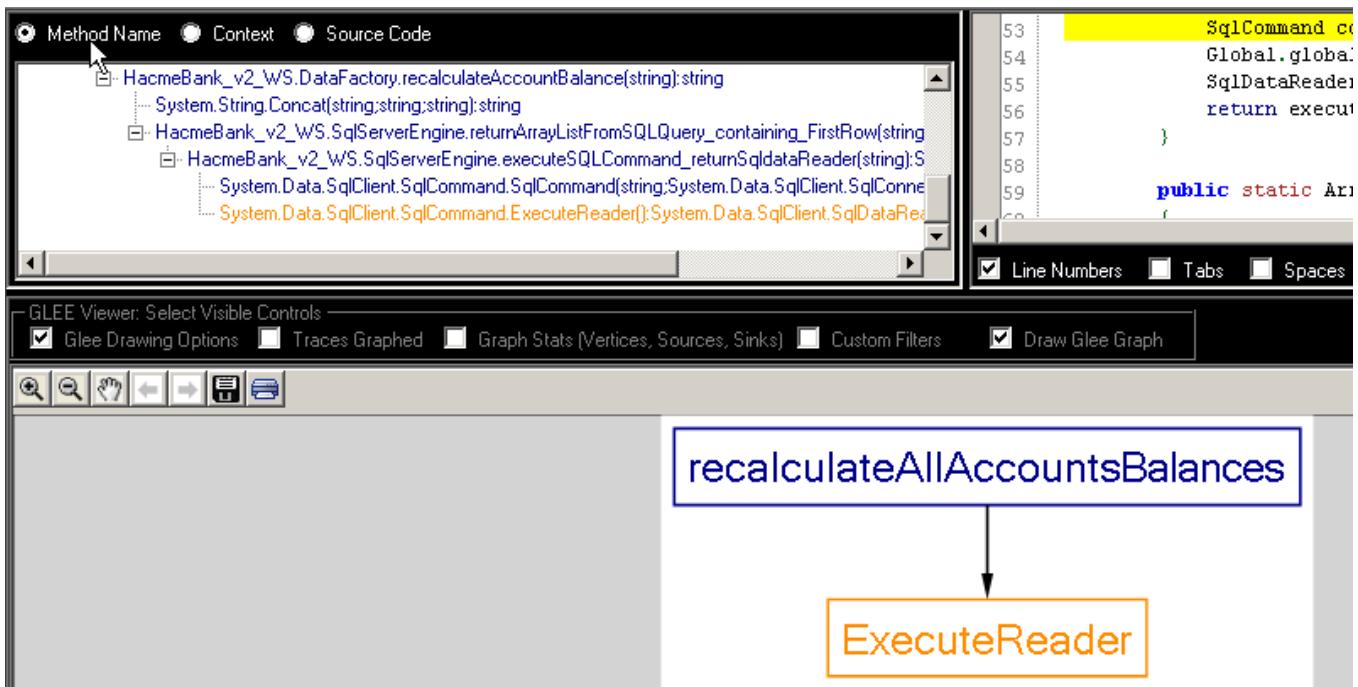
This screenshot is similar to the previous one, but the code is annotated with red boxes and arrows to highlight specific parts. Red boxes are placed around the 'ArrayList bankAccounts' declaration, the 'SqlDataReader reader1' declaration, and the 'SqlDataReader executeReader_Result' assignment. Arrows point from these annotations to the corresponding code lines. The rest of the code structure is identical to the first screenshot.

GLEE GRAPH

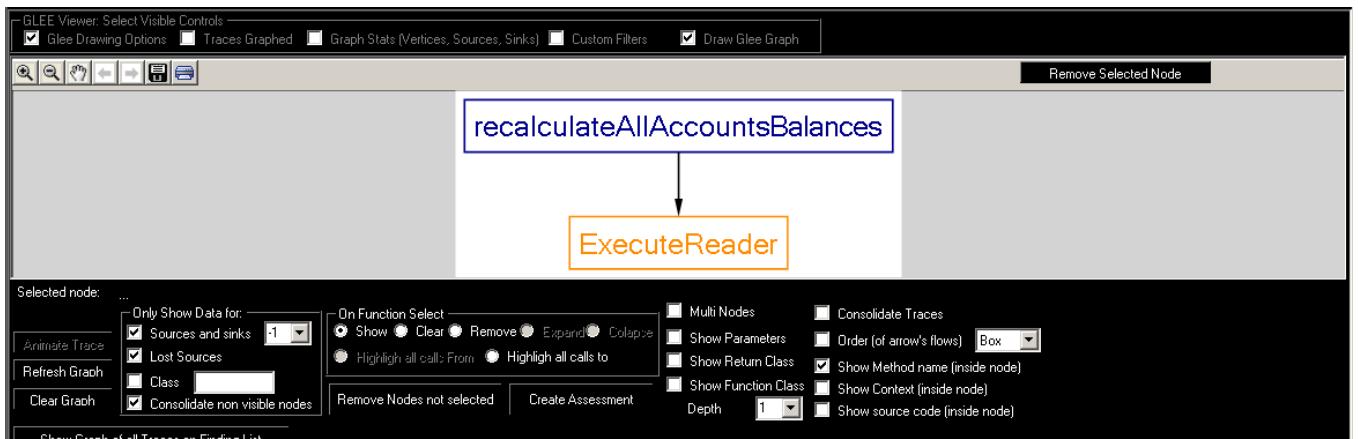
Let's now focus on the GLEE Graph



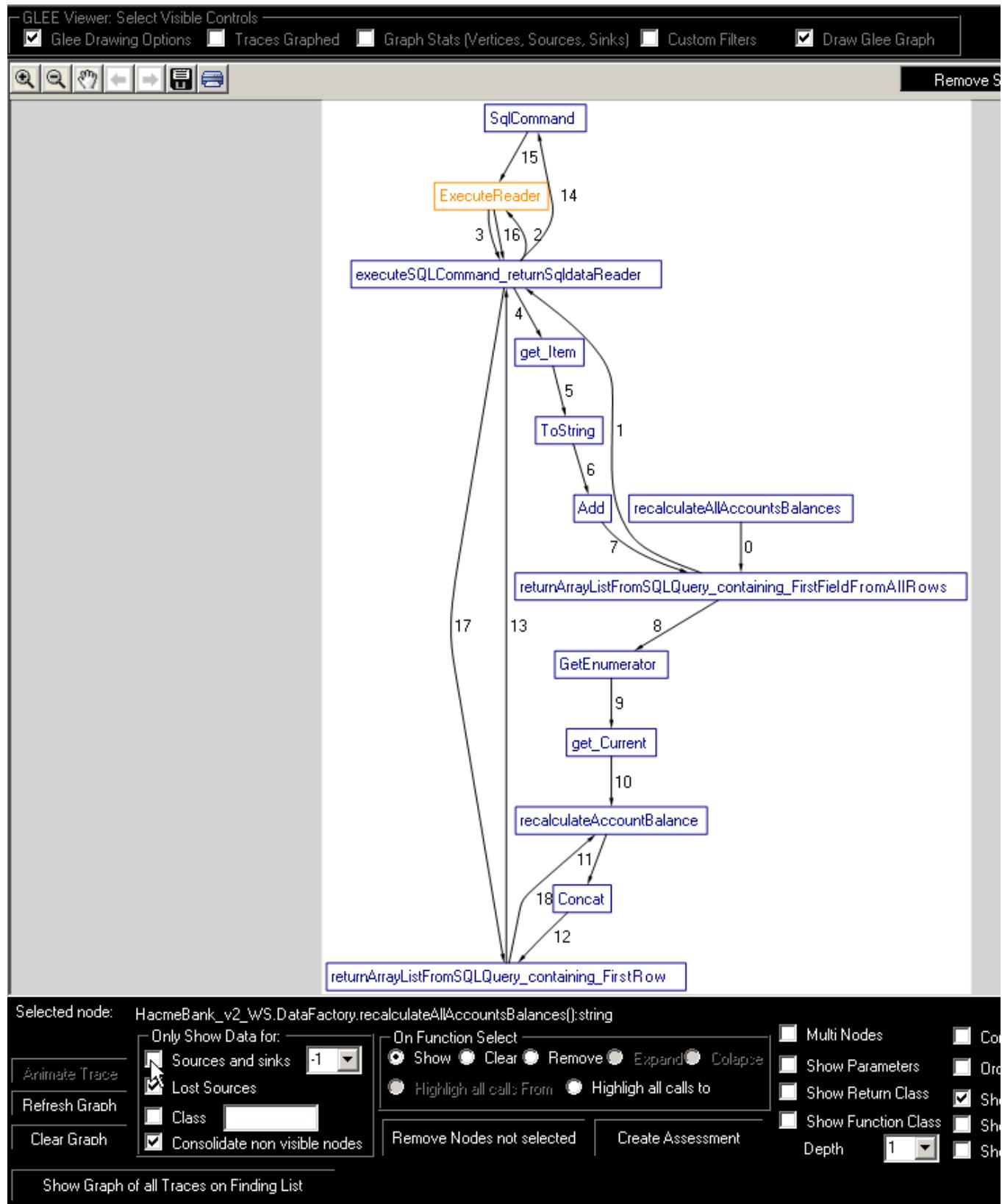
Go back to the 'Method Name' Filter



You might need to resize the options section 'Glee Drawing options'



The numbers between the nodes are the sequence (from 0 till 16)



This is not be best trace to Highlight GLEEs features, lets pick up another one:

The screenshot displays the GLEE application interface across four panels:

- Top Bar:** Shows "ViewAssessmentRun Select Visible Controls" and checkboxes for "Stats and Filters" (checked), "Trace Details" (checked), "Source Code" (unchecked), "Graph (Glee)" (checked), and "Custom Rules" (unchecked).
- Global Filters Panel:** Shows radio buttons for "All Findings" (selected), "Smart Traces" (selected), "Smart Traces - Lost Sink", "Smart Traces - UNIQUE Lost Sink", and "No Smart Traces".
- Tree View Panel:** Shows a tree structure under "[Injection_SQL (2)]":
 - new System.Data.SqlClient.SqlCommand . System
 - new System.Data.SqlClient.SqlCommand . System
 - Validation.EncodingRequired (1)
 - Validation.Required (5)
- Filter Panels:** Three separate windows show dropdown filters:
 - Top Filter:** Set to "vuln_type". Options include vuln_type, caller_name, lost_sink, source, known_sink, and source_code.
 - Middle Filter:** Set to "caller_name". Options include vuln_type, caller_name, lost_sink, source, known_sink, and source_code.
 - Bottom Filter:** Set to "source". Options include vuln_type, caller_name, lost_sink, source, known_sink, and source_code.

```
+-- System.Data.SqlClient.SqlCommand.ExecuteReader():System.Data.SqlClient.SqlDataReader
  |-- System.Web.HttpRequest.get_UserHostAddress():string
```

Filter source

- vuln_type
- caller_name
- lost_sink
- source**
- known_sink
- source_code

```
+-- System.Data.SqlClient.SqlCommand.SqlCommand(string;System.Data.SqlClient.SqlConnection):void
  |-- System.Web.HttpResponse.Write(string):void
```

Filter known_sink

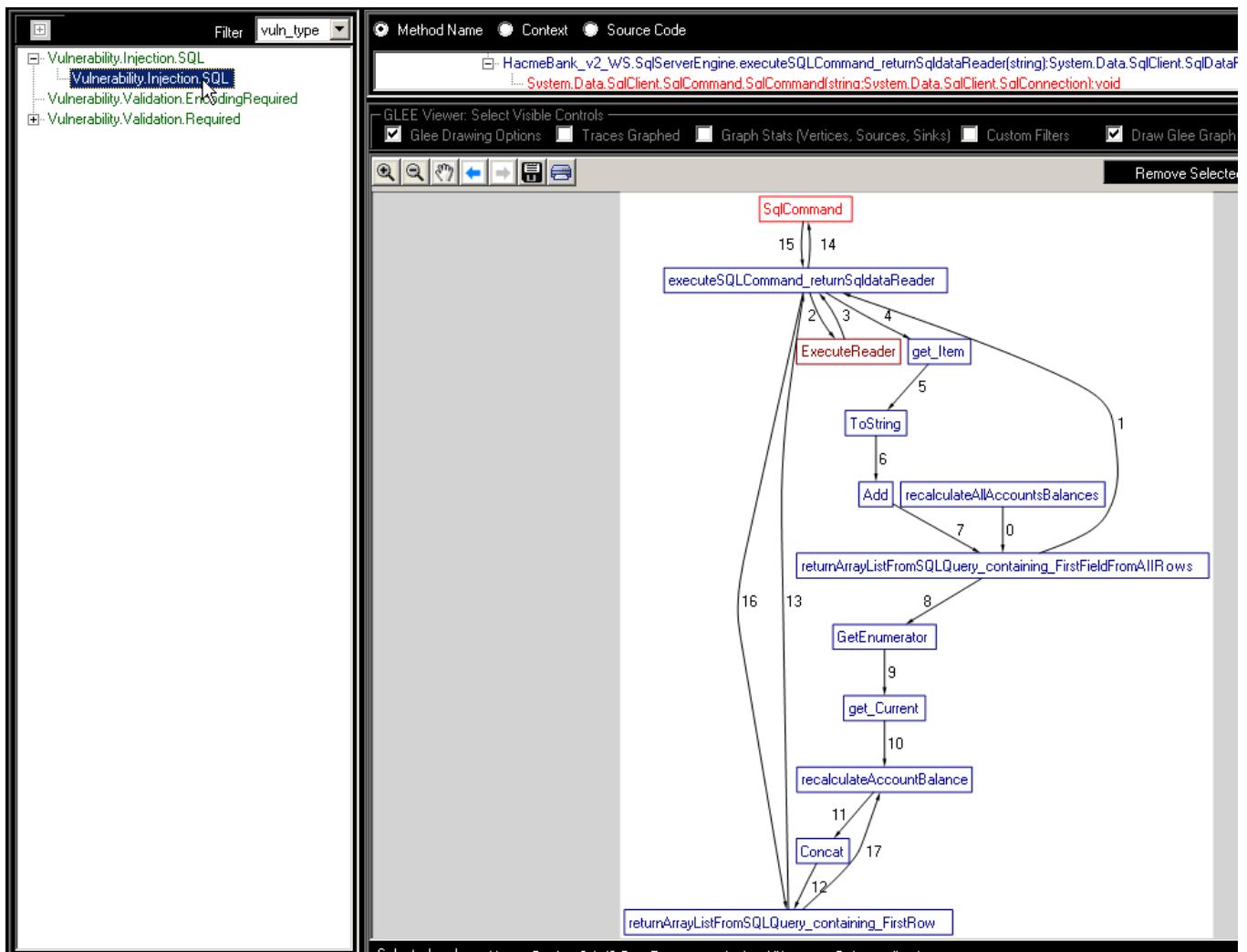
- vuln_type
- caller_name
- lost_sink
- source
- known_sink**
- source_code

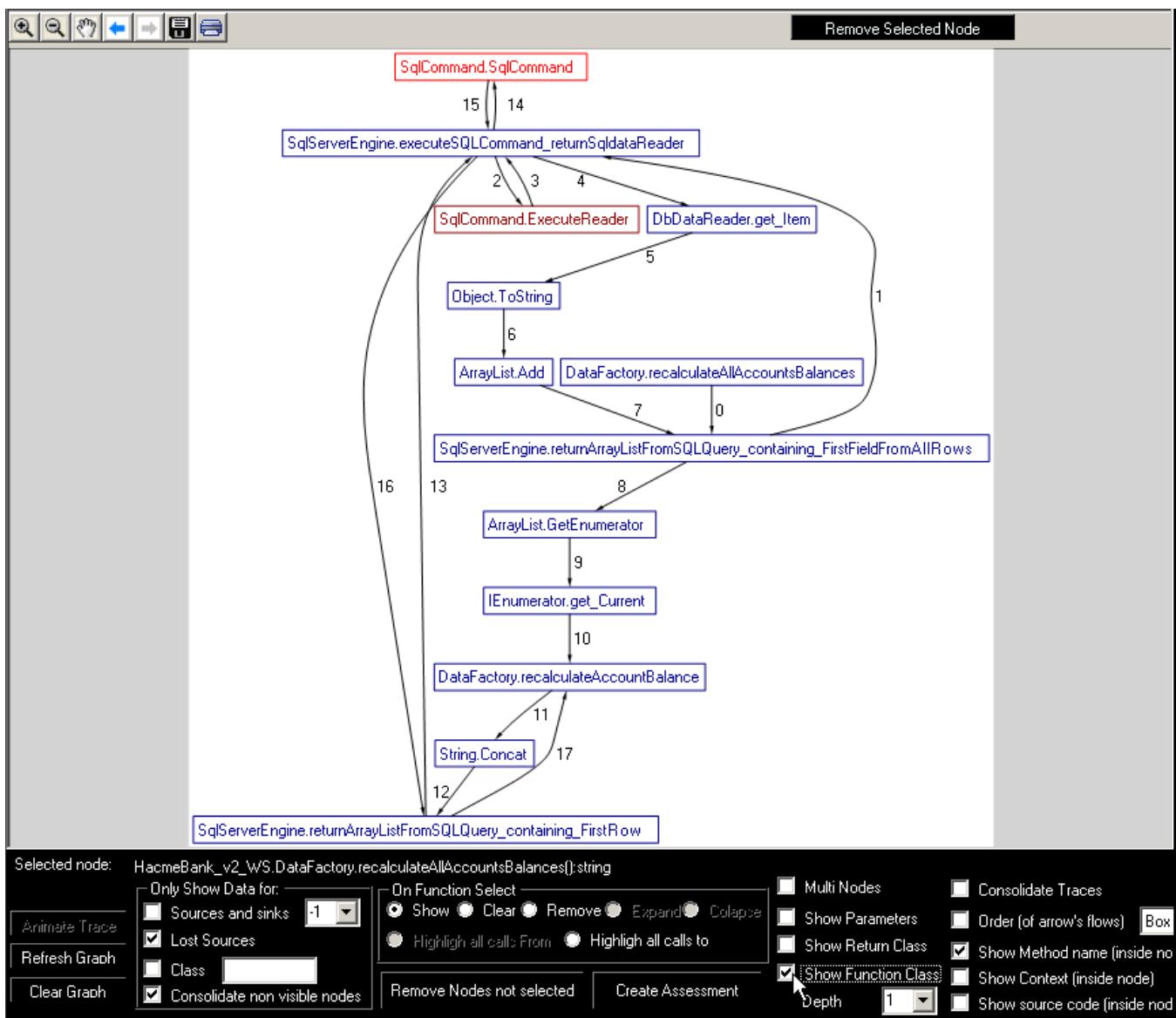
```
--> HttpContext.Current.Response.Write("<h3> Your current IP is: " + HttpContext.Current.Request.U
+-- SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
+-- SqlDataReader executeReader_Result = command1.ExecuteReader();
```

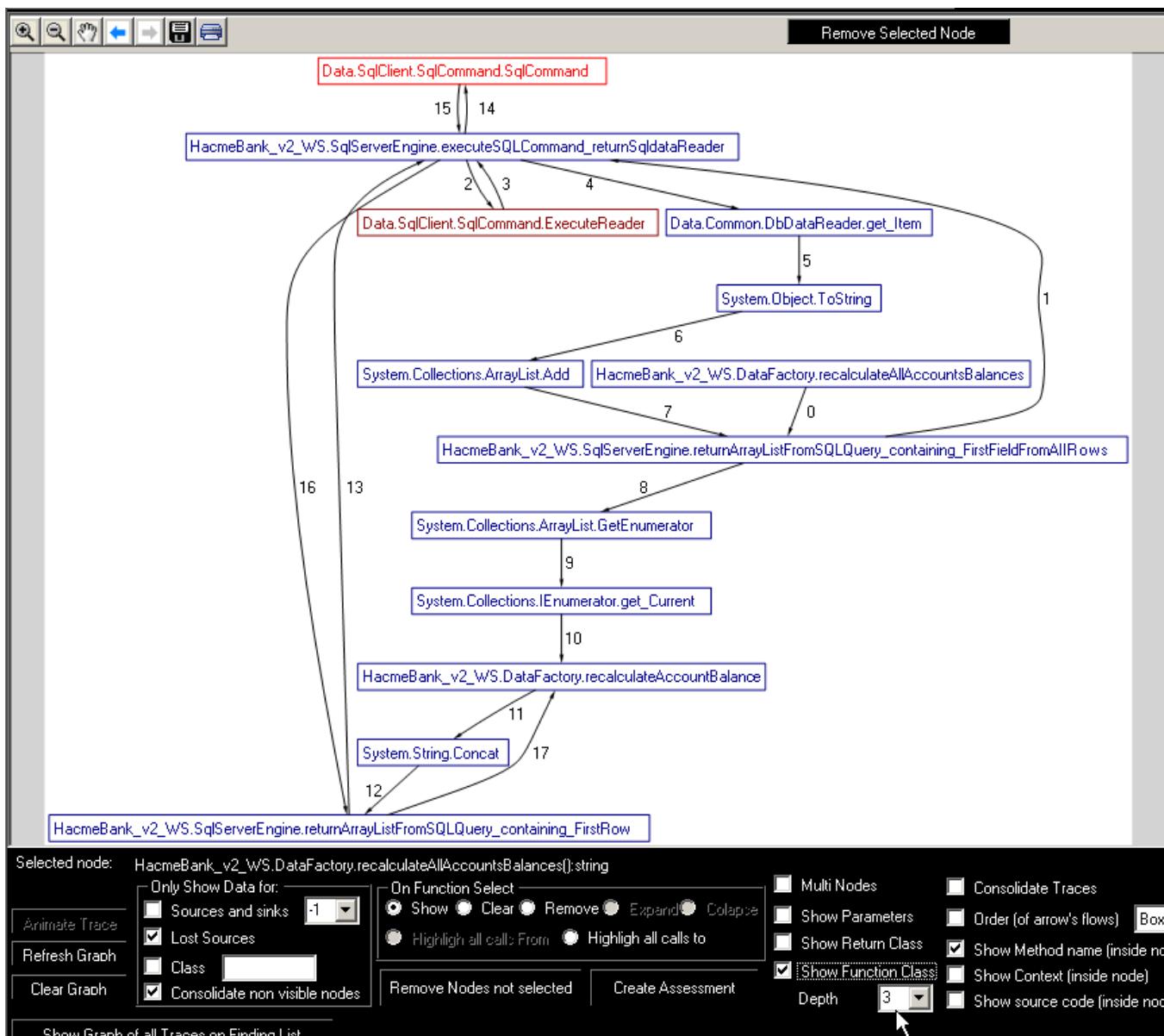
Filter known_sink

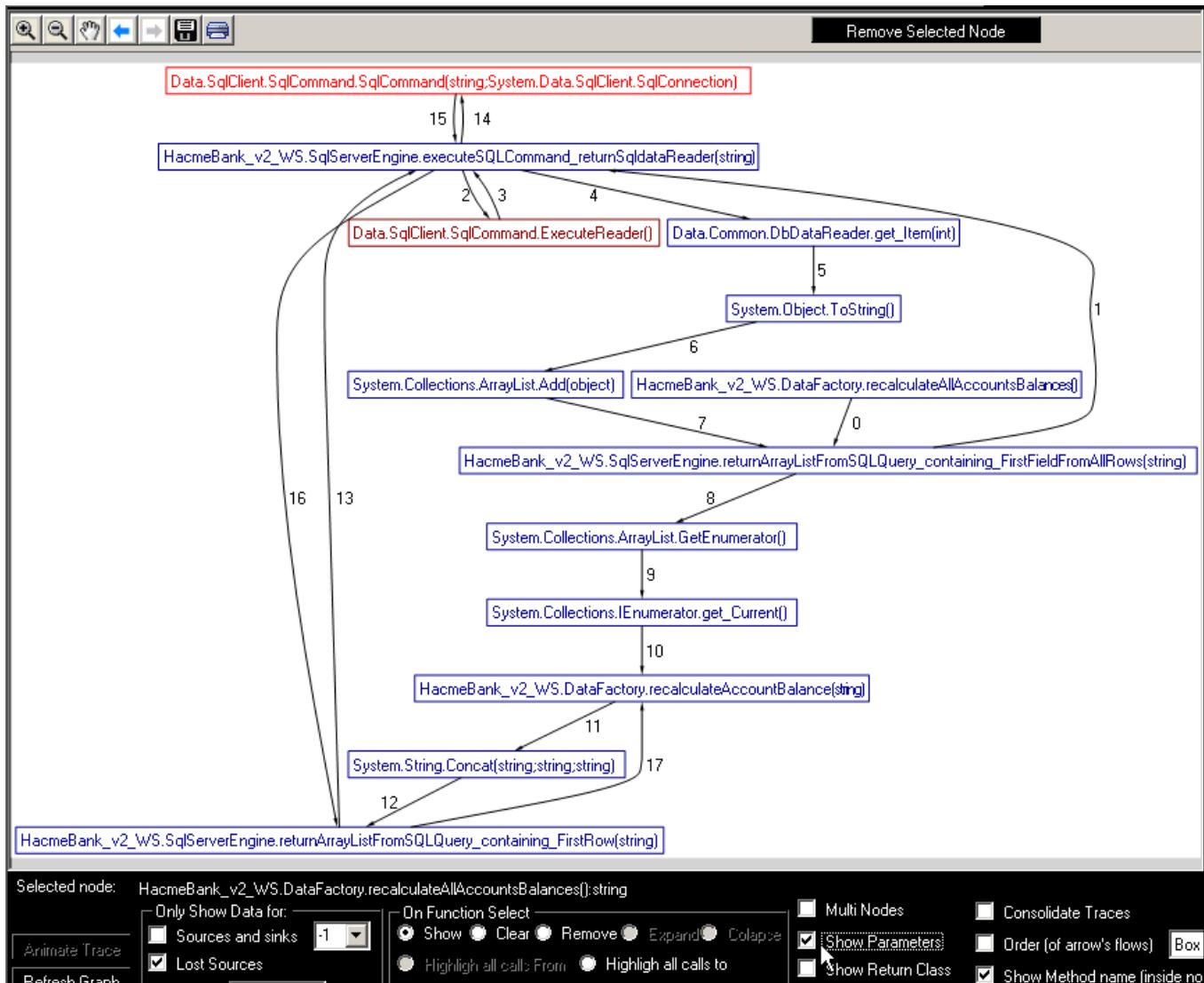
- vuln_type
- caller_name
- lost_sink
- source
- known_sink
- source_code**

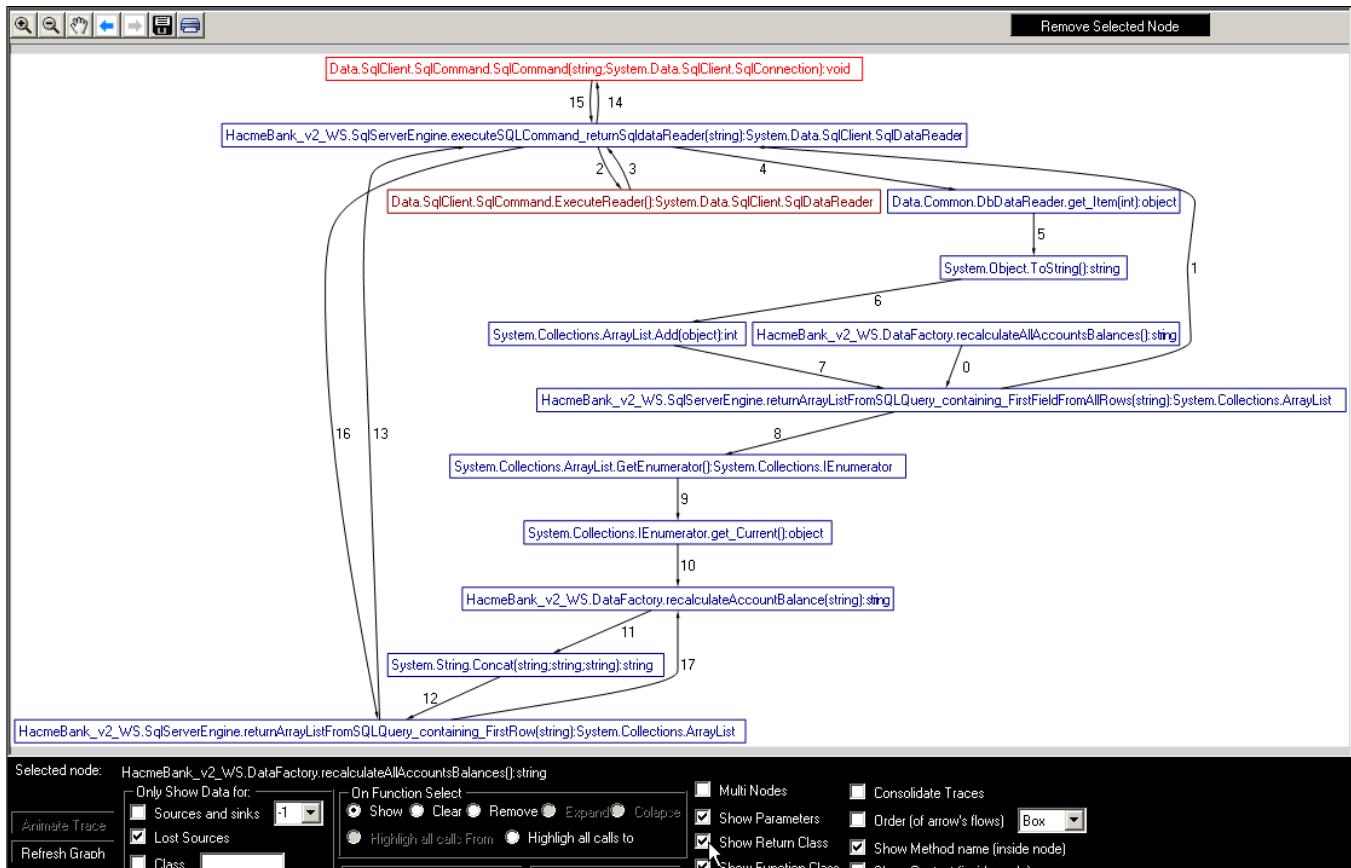
Go back to the vuln_type filter and select 2nd 'Vulnerability.Injectin.SQL'

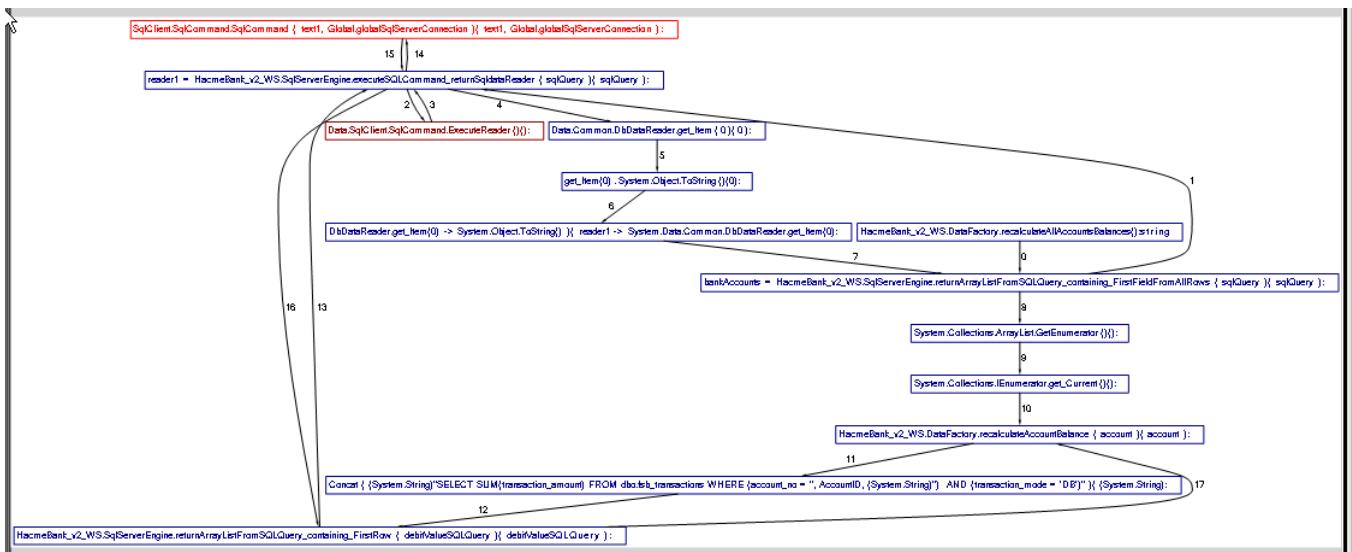








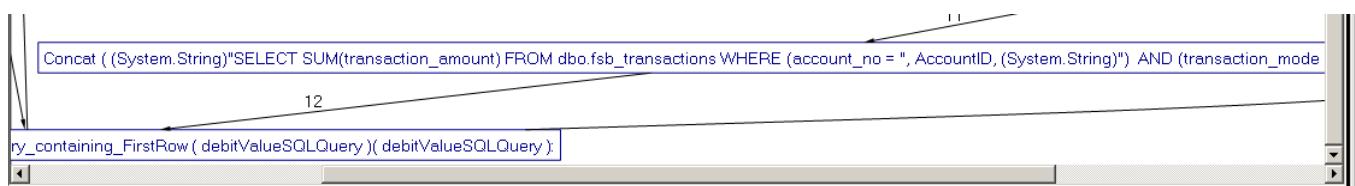




Zoom in

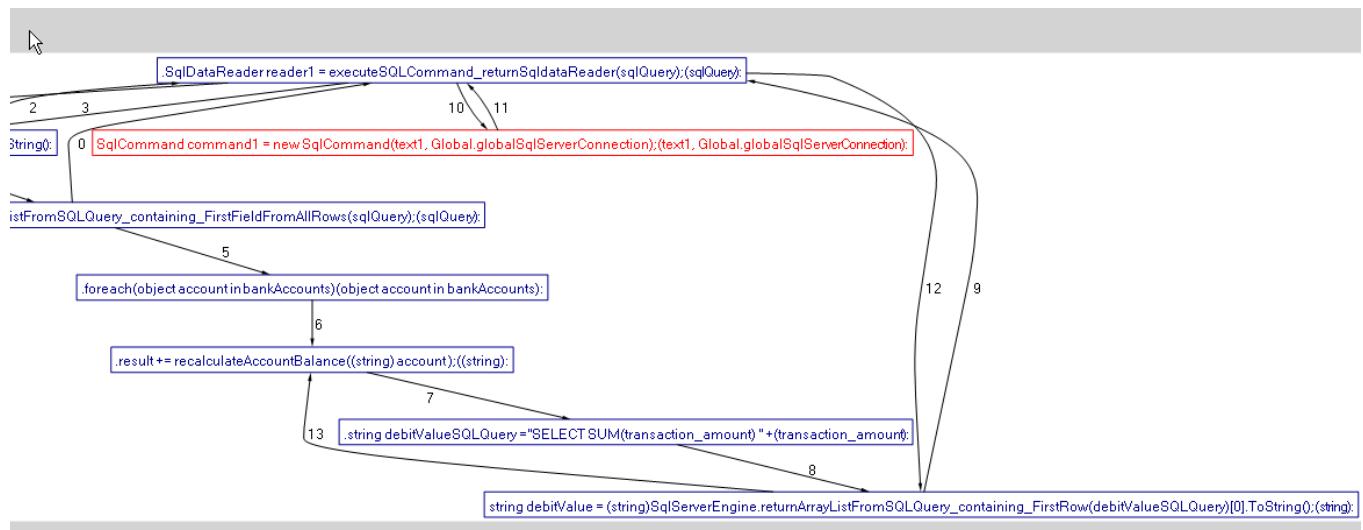


And note how with viewing the Context it becomes easier to read what is happening:

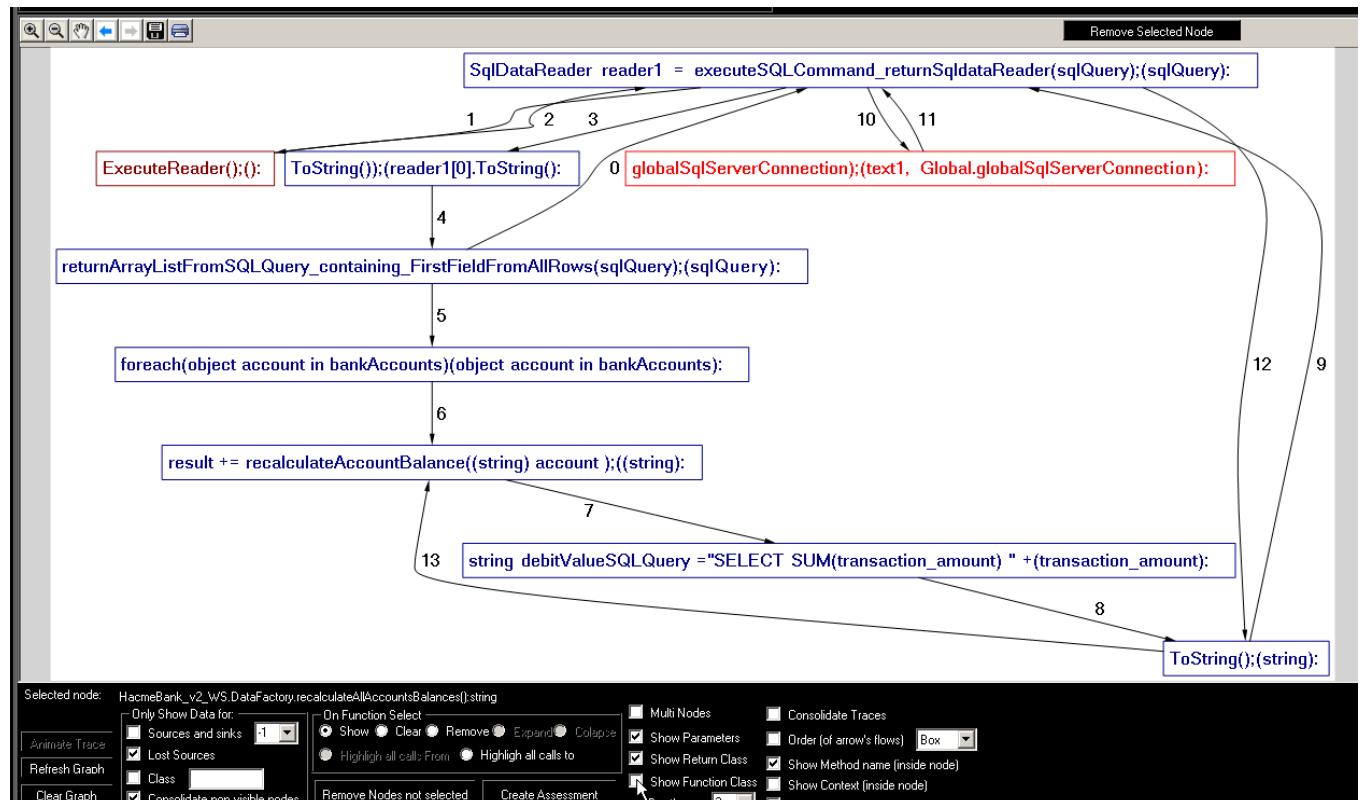


But the best view is the source code one:





(uncheck Show Function Class)

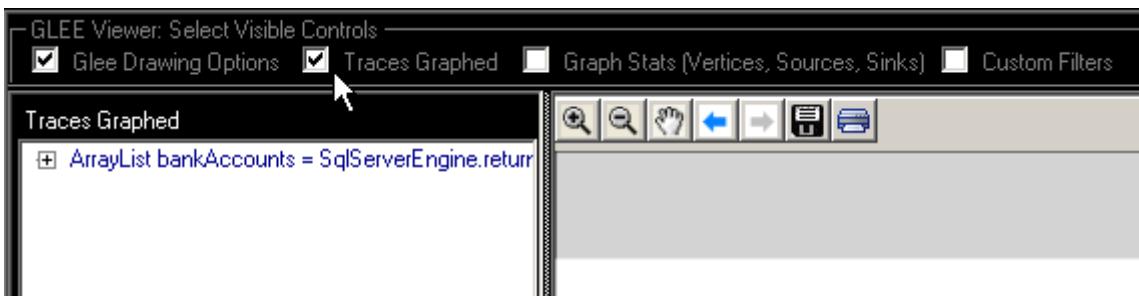


This Smart trace supports viewing multiple traces at the same time.

By default only one is shown. To see which on check the *Traces Graphed* checkbox



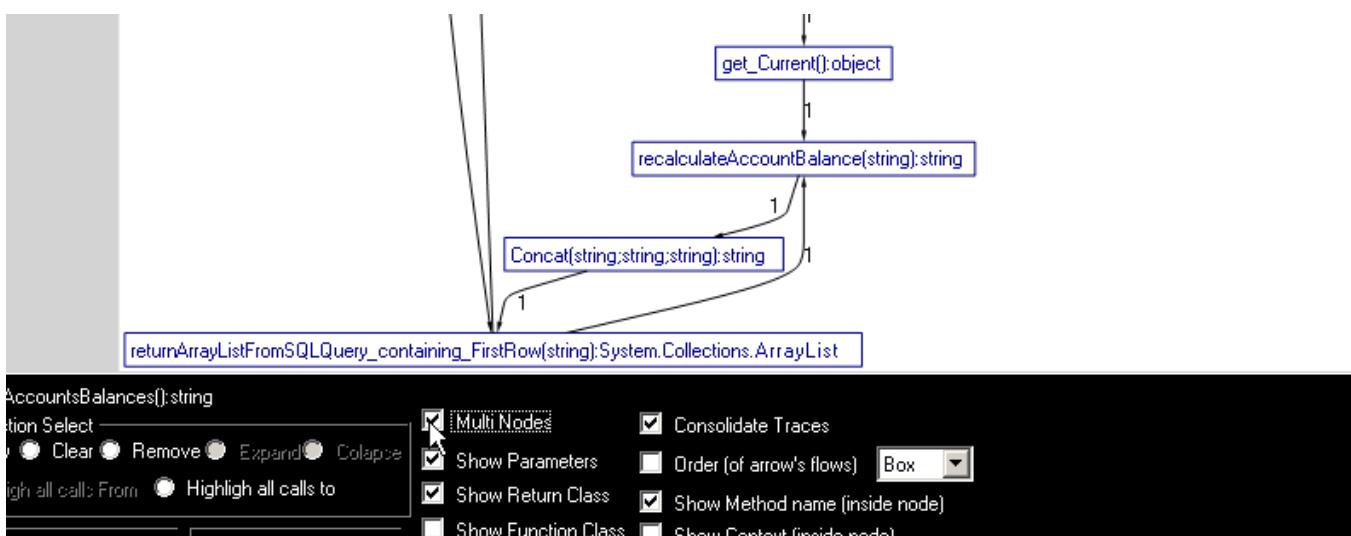
And note how the listbox only has one entry



Before adding more nodes go back to the *Method Name* view mode:



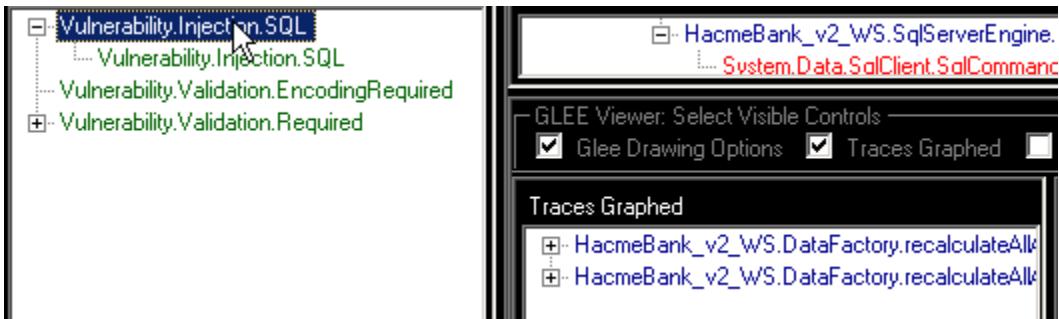
And check the Multi nodes



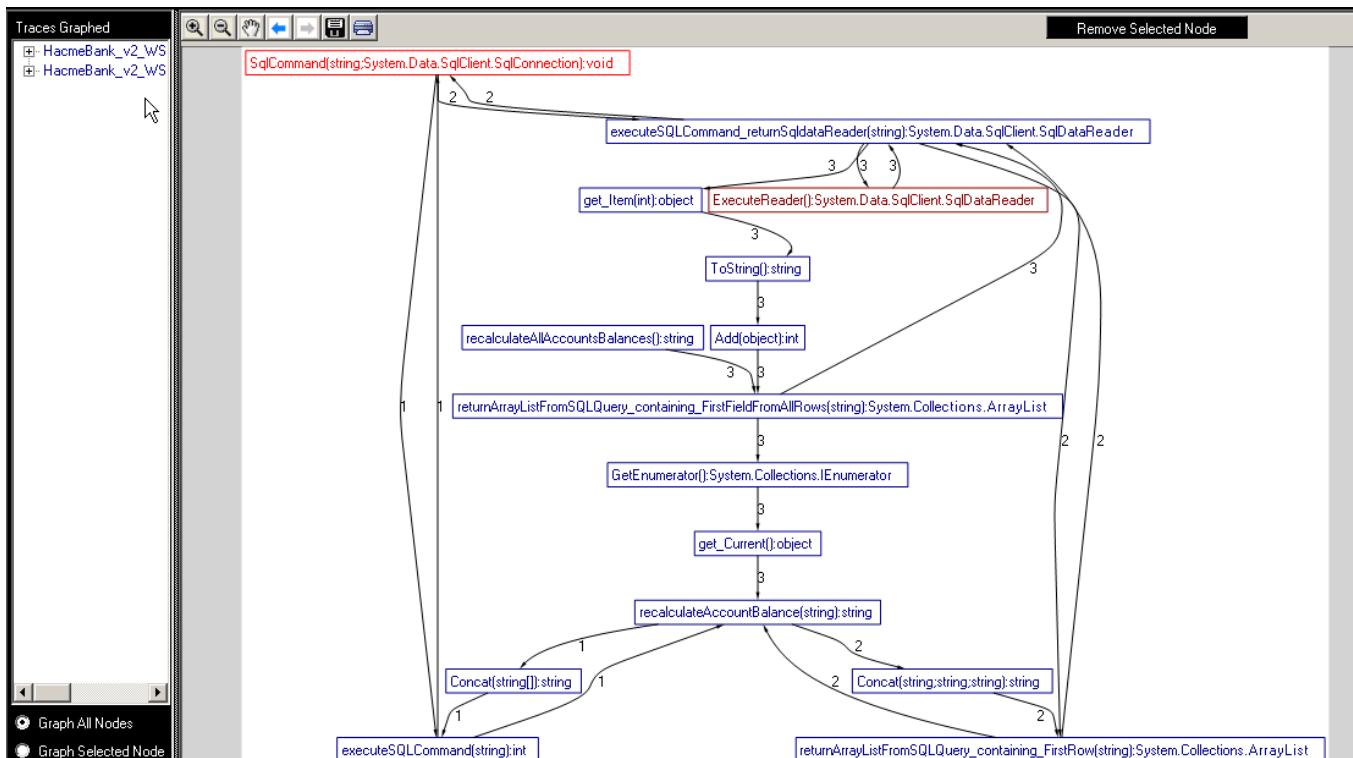
(note how all the numbers changed to 1.

The reason is because the selecting *Multi Nodes* automatically selects the *Consolidate Traces*(more details on why bellow))

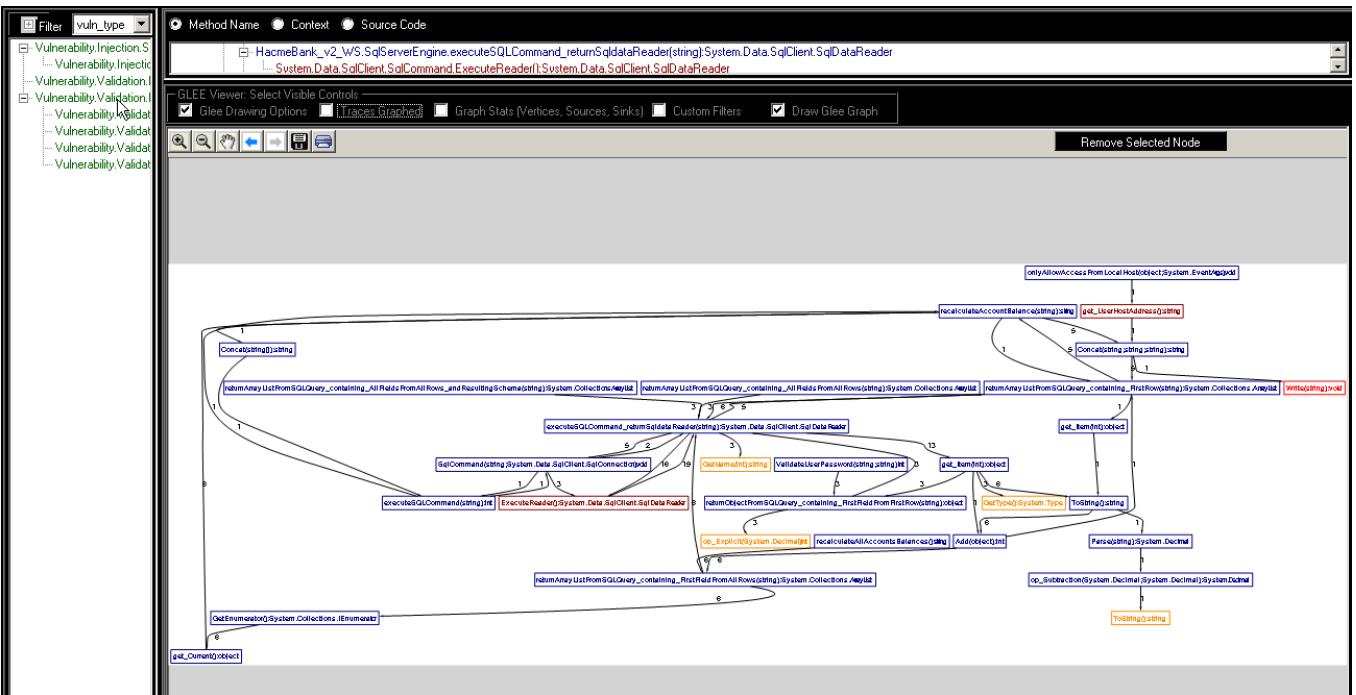
Now go back to the list of findings with traces and select the other SQL Injection trace:



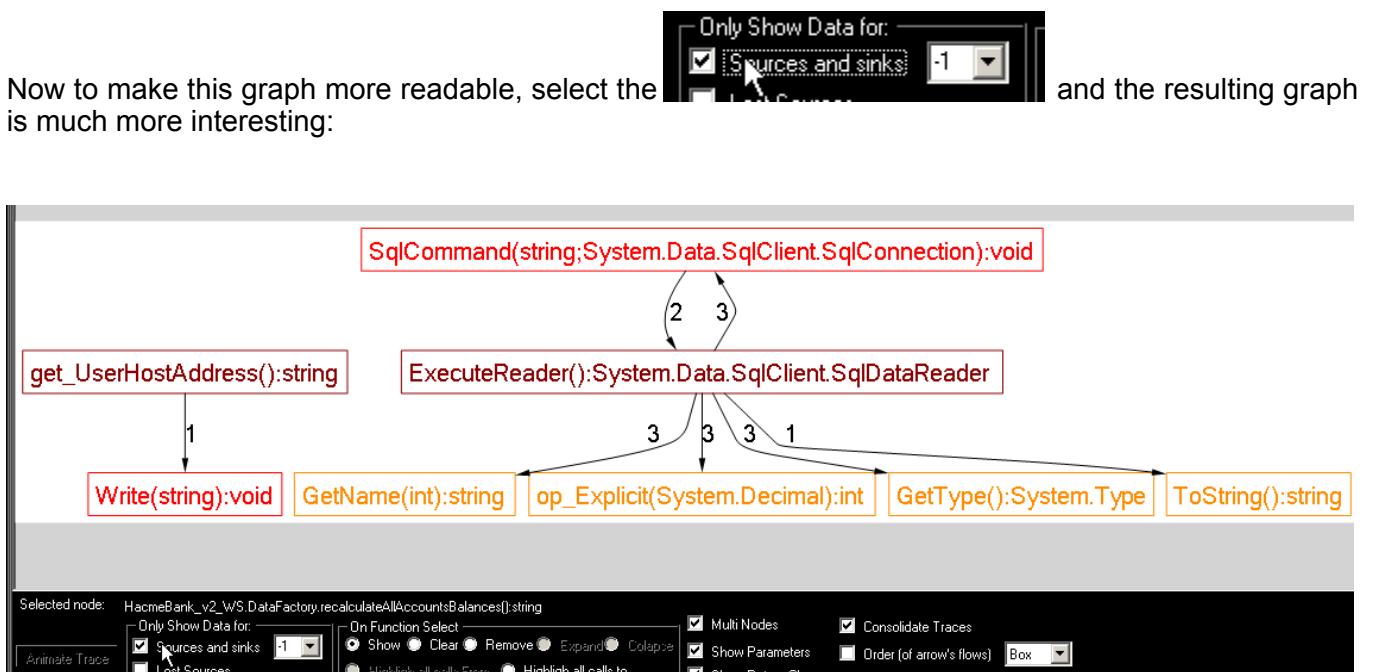
And note how the GLEE viewer is now showing both traces:



Click on the other traces so draw them all (when there are sub notes, all will be added when selecting the root node)

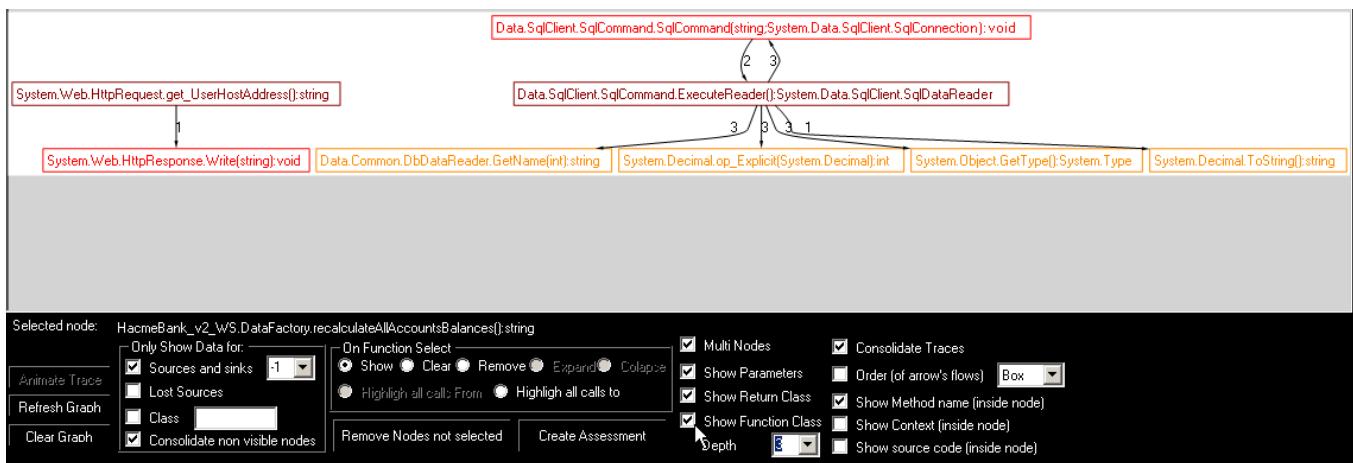


Note that in this graph we now have all sources sinks and lost sinks (the reason why there are only 4 lost sinks is because `ExecuteReader` is both a source and Lost Sink)

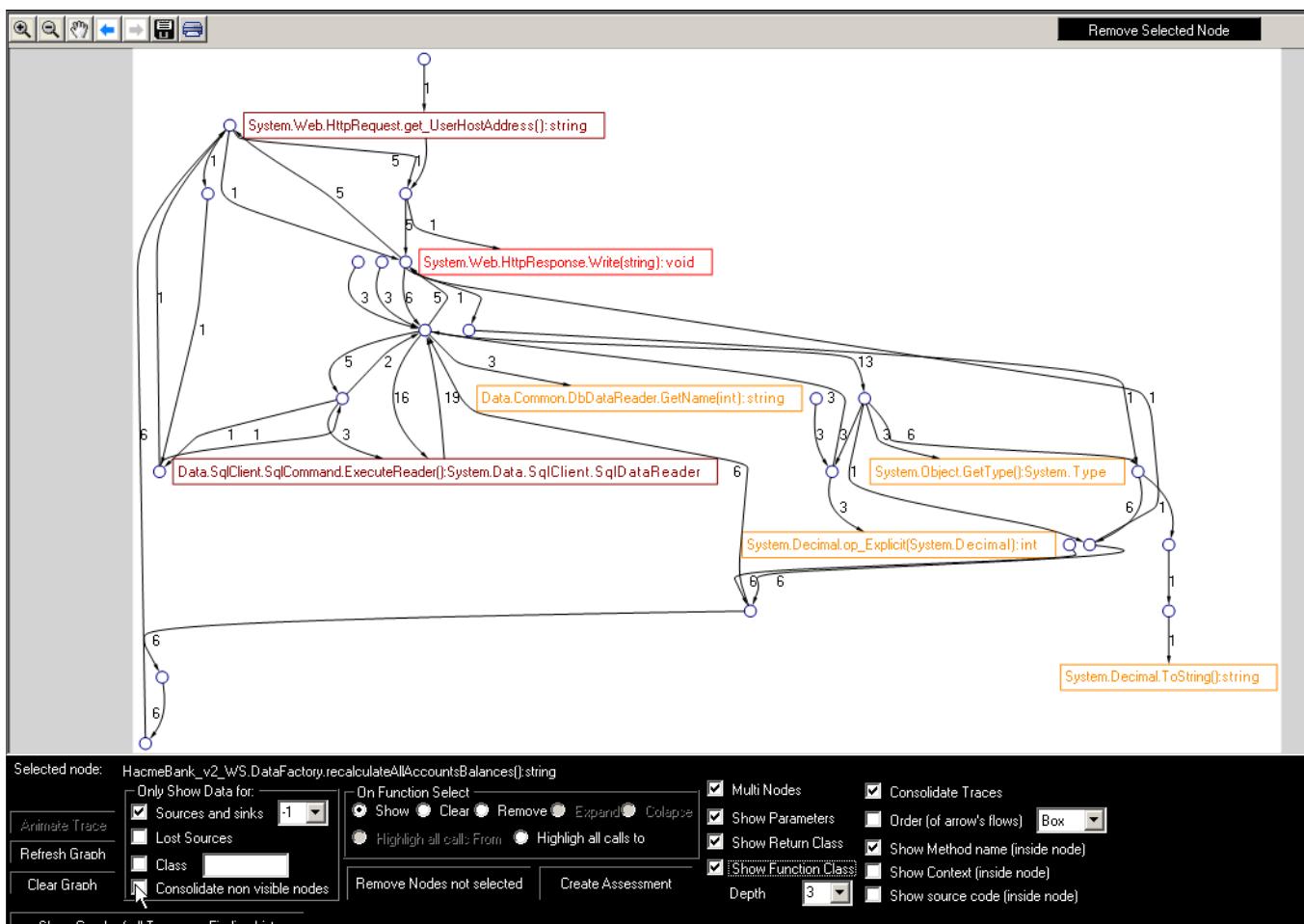


This is a graph that only shows the sources and sinks (the numbers in the lines are the number of cases

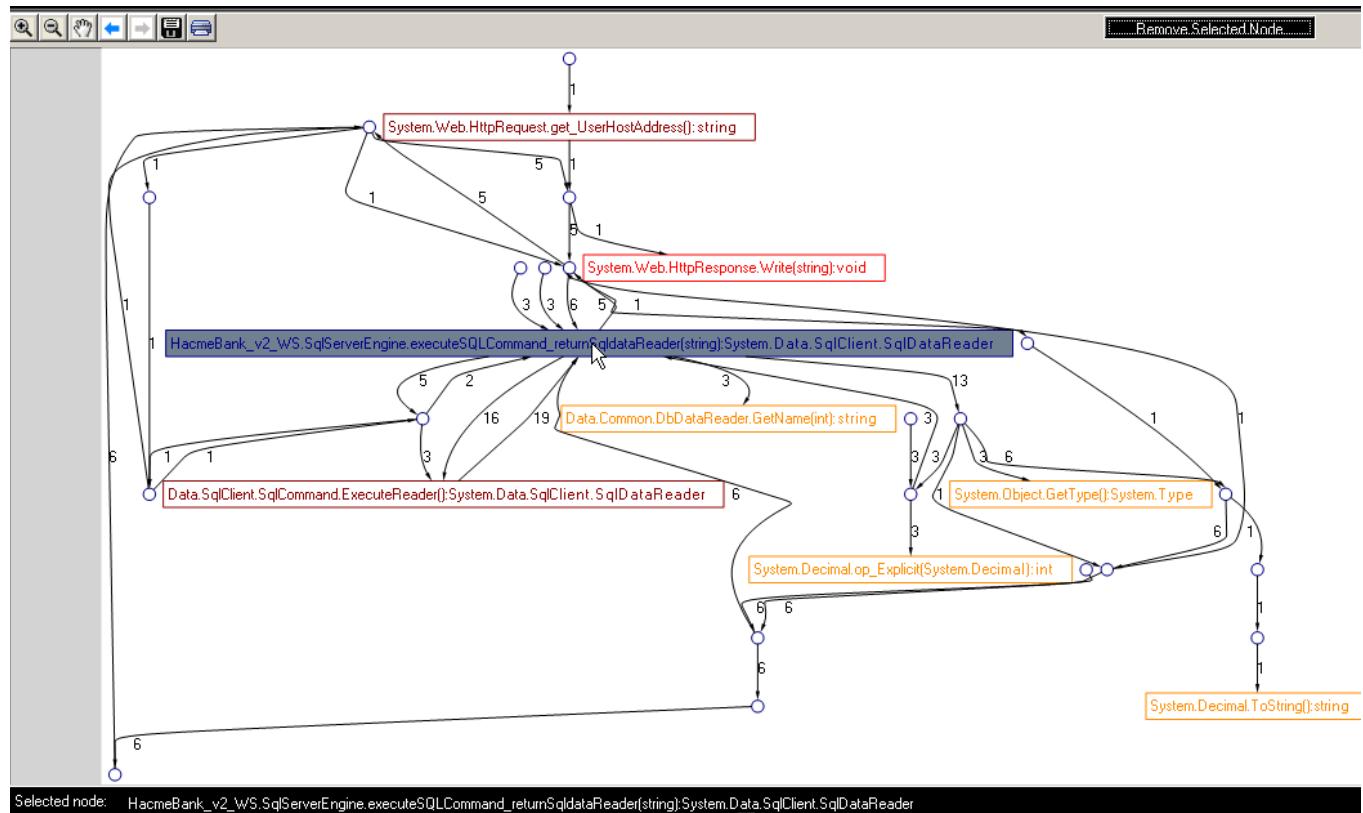
with that particular source->target method)



To get a fell how many methods exist between source and sink uncheck *Consolidate non visible nodes*



To see what a particular node is, just click on the small circle

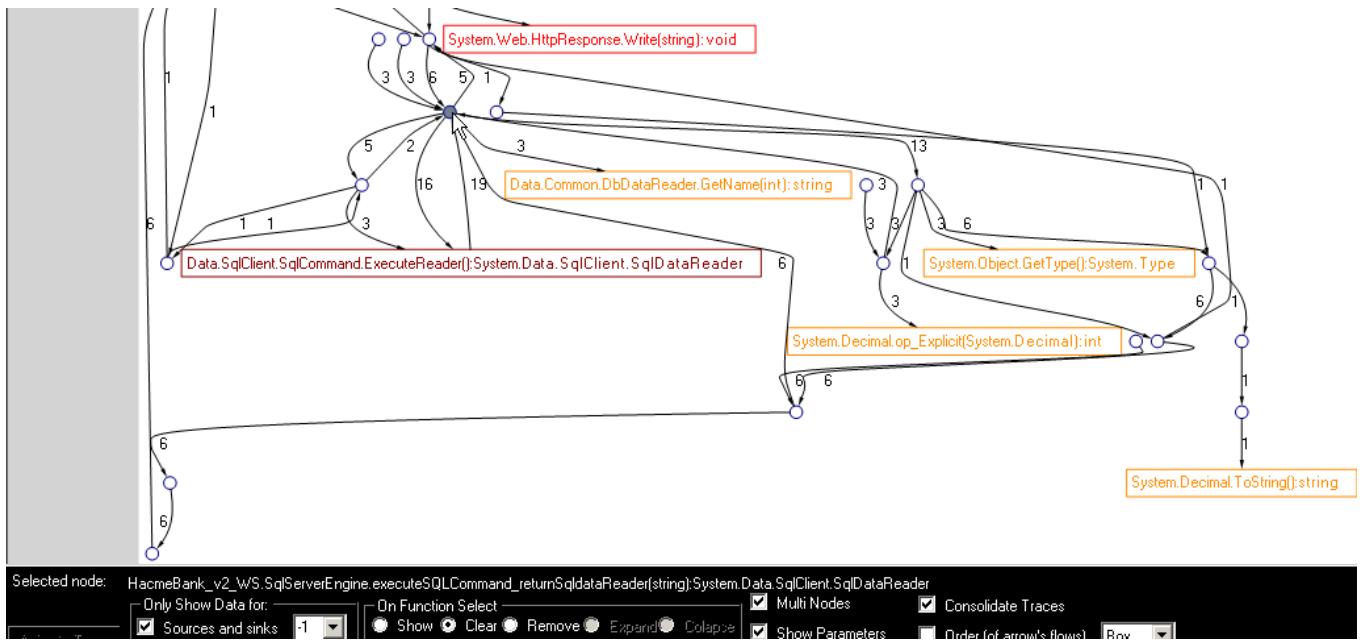


The behaviour of the node-left-click-event is controlled by the following settings:



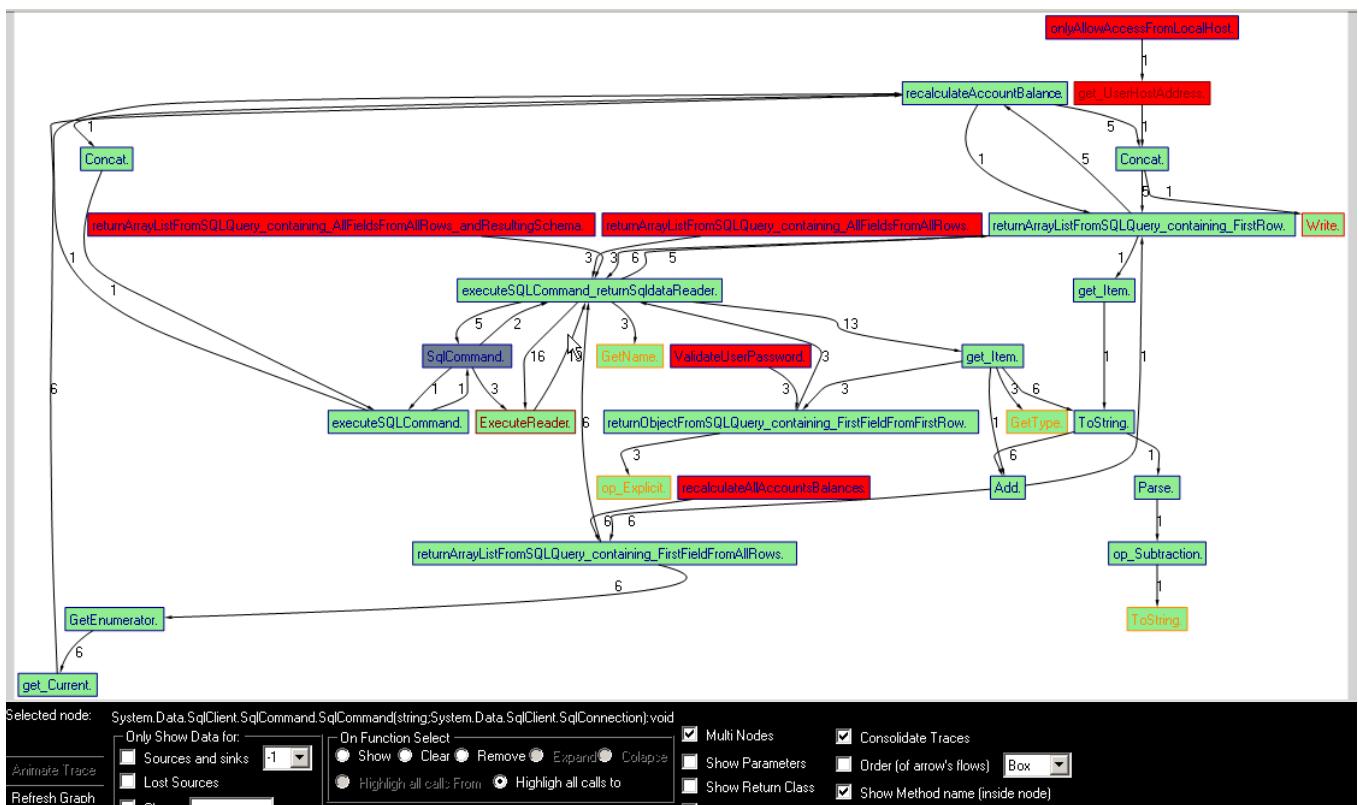
In the case above the default *Show* option was used.

The *Clear* option is set to the Right-click and will collapse the node:



Selecting  allows to remove nodes with left-click

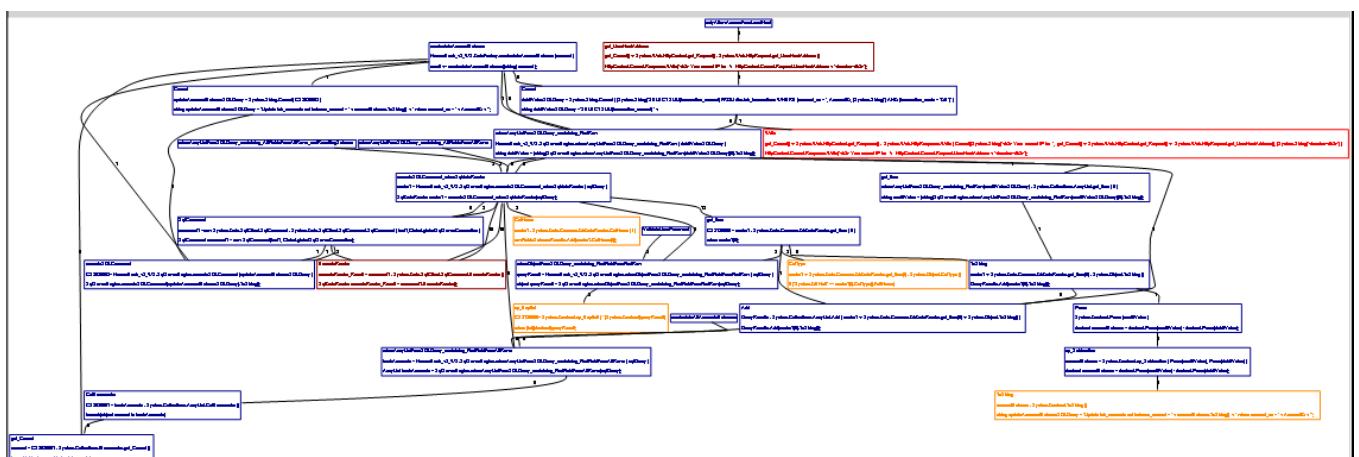
 will colour code calls TO the selected method (in red) and calls FROM the selected method (in green).



(although this example is not the best one (it works best when there are different types of vulnerabilities shown))

It is also possible to view the source code and context inside each box:

- Show Method name (inside node)
- Show Context (inside node)
- Show source code (inside node)



```

Web.HttpResponse.Write
get_Current()-> System.Web.HttpContext.get_Response().System.Web.HttpResponse.Write(Concat((System.String)"<h3> Your current IP is: " ,get_Current()-> System.Web.HttpContext.Current.Response.Write("<h3> Your current IP is: "+ HttpContext.Current.Request.UserHostAddress+"</center></h3>"));

```



To go back to the normal GLEE Viewing mode uncheck

SQL INJECTION CASES

A quick look at the two traces with SQL Injection show that they are only exploitable if the attacker is able to control the account_id value:

```

Injection.SQL (2)
new System.Data.SqlClient.SqlConnection();
new System.Data.SqlClient.SqlCommand();
Validation.EncodingRequired (1)
Validation.Required (5)

HacmeBank_v2_WS.SqlServerEngine.executeSQLCommand(string)int
ArrayList bankAccounts = SqlServerEngine.returnArrayListFromSQLQuery_containing_FirstFieldFromAllRows(sqlQuery);
ArrayList bankAccounts = SqlServerEngine.returnArrayListFromSQLQuery_containing_FirstFieldFromAllRows(sqlQuery);
SqlDataReader reader1 = executeSQLCommand_returnSqlDataReader(sqlQuery);
SqlDataReader executeReader_Result = command1.ExecuteReader();
QueryResults.Add(reader1[0].ToString());
QueryResults.Add(reader1[0].ToString());
QueryResults.Add(reader1[0].ToString());
foreach(object account in bankAccounts)
foreach(object account in bankAccounts)
result += recalculateAccountBalance([string] account );
string updateAcccountBalanceSQLQuery = "Update fsb_accounts set balance_amount = "+ accountBalance.ToString() + " where account_no = "+ AccountID + "
SqlServerEngine.executeSQLCommand(updateAcccountBalanceSQLQuery).ToString();
SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);

.....
result += recalculateAccountBalance
string debitValueSQLQuery = $"
string debitValue = (string)SqlSe
string debitValue = (string)SqlSe
SqlDataReader reader1 = e
SqlCommand command
169
170
171
172
173
174
175
176
177
{
    string debitValueSQLQuery = "SELECT SUM(transaction_amount) " +
        "FROM dbo.fsb_transactions " +
        "WHERE (account_no = " + AccountID + ") " +
        "AND (transaction_mode = 'DB')";
    string creditValueSQLQuery = "SELECT SUM(transaction_amount) " +
        "FROM dbo.fsb_transactions " +
        "WHERE (account_no = " + AccountID + ") " +
        "AND (transaction_mode = 'CR')";
}

```

So unless we find another vulnerability that allows the injection of payloads on the AccountID value this will not be exploitable.

Since we don't have more traces let's look at the findings with no traces.

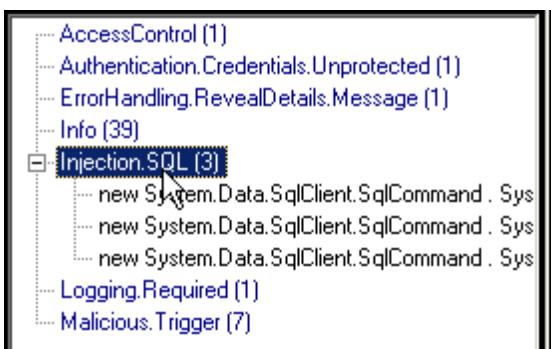
Select again the *Stats and Filters*



and in the left-hand options, select the *No Traces* radio button



The original set of findings had 3 SQL injections with no traces:



Here are the 3 findings:

```

public static int executeSQLCommand_onLocalSqlServer(string sqlQueryToExecute)
{
    // ...
    Global.createSqlServerConnection();
    SqlConnection Global.globalSqlServerConnection = new SqlConnection(ConfigurationSettings.AppSettings["GlobalConnectionString"]);
    string text1 = sqlQueryToExecute;
    SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
    Global.globalSqlServerConnection.Open();
    int executeNonQuery_Result = command1.ExecuteNonQuery();
    Global.globalSqlServerConnection.Close();
    return executeNonQuery_Result ;
}

public static int executeSQLCommand(string sqlQueryToExecute)
{
    Global.createSqlServerConnection();
    SqlConnection Global.globalSqlServerConnection = new SqlConnection(ConfigurationSettings.AppSettings["GlobalConnectionString"]);
    string text1 = sqlQueryToExecute;
    SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
    Global.globalSqlServerConnection.Open();
    int executeNonQuery_Result = command1.ExecuteNonQuery();
    Global.globalSqlServerConnection.Close();
    return executeNonQuery_Result;
}

```

The findings tree view shows the following categories:

- Info (39)
- Injection.SQL (3)** (selected)
- new System.Data.SqlClient.SqlCommand . Sys
- new System.Data.SqlClient.SqlCommand . Sys
- new System.Data.SqlClient.SqlCommand . Sys
- Logging.Required (1)
- Malicious.Trigger (7)

```

    public static SqlDataReader executeSQLCommand_returnSqlDataReader(string sqlQueryToExecute)
    {
        Global.createSqlServerConnection();
        //SqlConnection Global.globalSqlServerConnection = new SqlConnection(ConfigurationSettings.AppSettings["ConnectionString"]);
        string text1 = sqlQueryToExecute;
        SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
        Global.globalSqlServerConnection.Open();
        SqlDataReader executeReader_Result = command1.ExecuteReader();
        return executeReader_Result ;
    }

```

Looking at the above 3 examples one quickly see that if tainted data arrives at the following methods we have an SQL injection:

```

public static int executeSQLCommand_onLocalSqlServer(string sqlQueryToExecute)
public static SqlDataReader executeSQLCommand_returnSqlDataReader(string sqlQueryToExecute)
public static int executeSQLCommand(string sqlQueryToExecute)

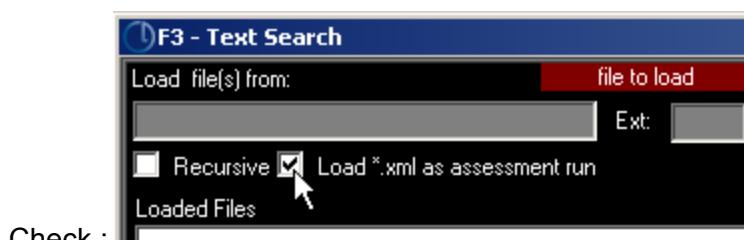
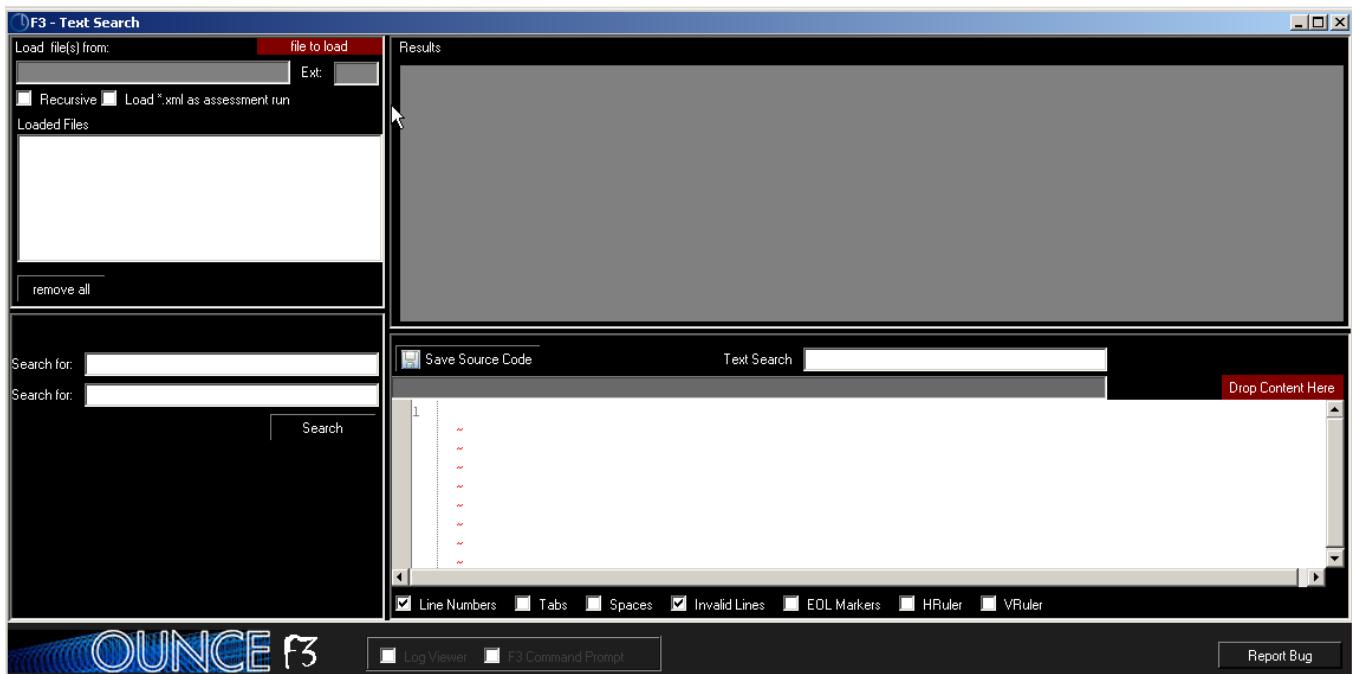
```

TEXT SEARCH

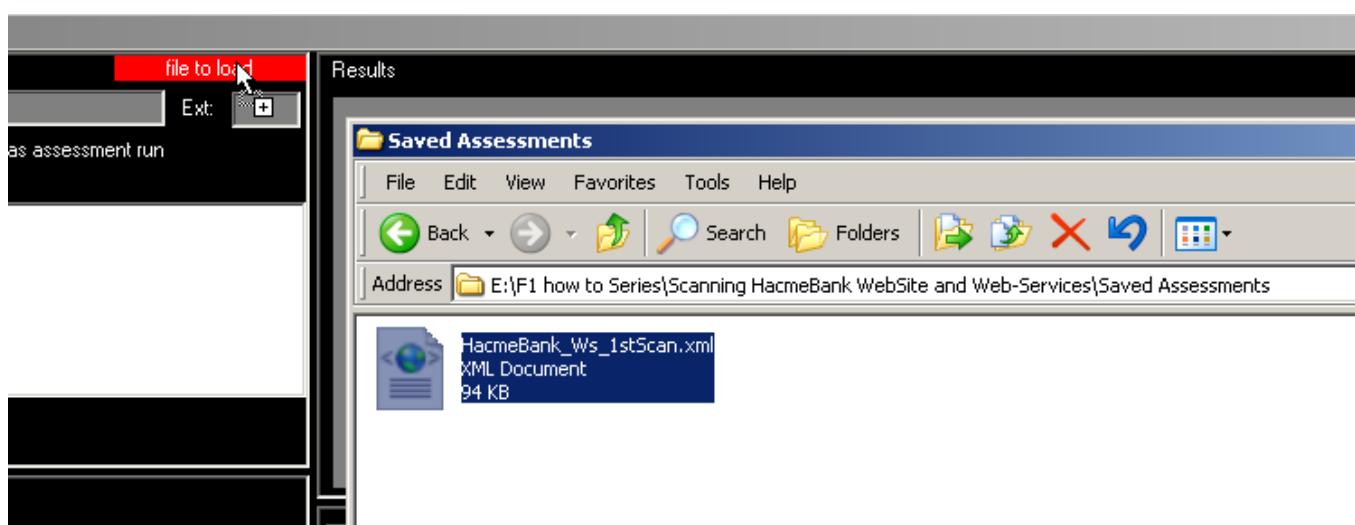
Using RegEx search to find out if there is a path from the exposed webservices functions to these methods

Open f1AddOn_TextSearch.exe





Drag and Drop the previously created Saved Assessment file



And that will load for search all files that had a finding in this project:

F3 - Text Search

Load file(s) from:

Recursive Load *.xml as assessment run

Loaded Files

```
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\DataFactory.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\HttpModule_onlyAllowLocalAccess.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\SqlServerEngine.cs
```

Just to make sure we have all files

F3 - Text Search

Load file(s) from:

Recursive Load *.xml as assessment run

Check: Loaded Files

file to load

Ext: *.cs

Enter *.cs on

Enter the path to the source code files:

F3 - Text Search

Load file(s) from:

E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS

Recursive Load *.xml as assessment run

Press enter to load all *.cs from this directory

F3 - Text Search

Load file(s) from: file to load

E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS Ext: *.cs

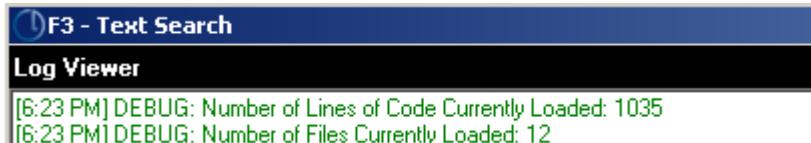
Recursive Load *.xml as assessment run

Loaded Files

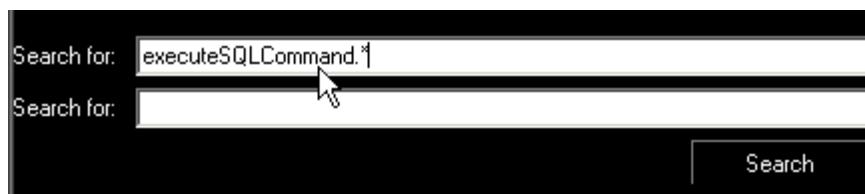
```
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\AssemblyInfo.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\DataFactory.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\FileManagement.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\HttpModule_onlyAllowLocalAccess.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\InstallTools.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\PaymentProviders.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\classes\SqlServerEngine.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\Global.asax.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\install\Install.asmx.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\WebServices\AccountManagement.asmx.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\WebServices\UserManagement.asmx.cs
E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS\App_Code\WebServices\UsersCommunity.asmx.cs
```

remove all

The LogView will provide details of how many files and text lines are loaded



Enter `executeSQLCommand.*` on the Search for textbox and click Search

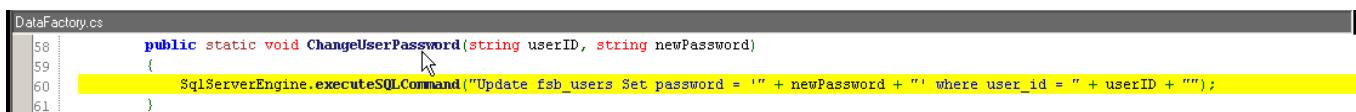


And the search results will show many cases of vulnerable code:

path	file	line #	match text	match line
E...	DataFactory.cs	36	executeSQLCommand("Insert into fsb_users (user_name)...")	SqlServerEngine.executeSQLCommand("Insert into fsb_users (user_name,login_id,password,creation_date)Values (" + userName + "','" + loginID + "','" + userPa...
E...	DataFactory.cs	41	executeSQLCommand("Update fsb_users set user_name...")	SqlServerEngine.executeSQLCommand("Update fsb_users set user_name = '" + userName + "',login_id = '" + loginID + "',password = '" + userPassword + "' w...
▶ E...	DataFactory.cs	59	executeSQLCommand("Update fsb_users Set password...")	SqlServerEngine.executeSQLCommand("Update fsb_users Set password = '" + newPassword + "' where user_id = '" + userID + "'");
E...	DataFactory.cs	64	executeSQLCommand("Delete from fsb_users where use...")	return SqlServerEngine.executeSQLCommand("Delete from fsb_users where user_ID ='" + userID + "'");
E...	DataFactory.cs	112	executeSQLCommand(sourceAccountTransaction))	if (1 == SqlServerEngine.executeSQLCommand(sourceAccountTransaction))
E...	DataFactory.cs	113	executeSQLCommand(destinationAccountTransaction))	if (1 == SqlServerEngine.executeSQLCommand(destinationAccountTransaction))
E...	DataFactory.cs	114	executeSQLCommand(createFundTransferItem)))))	if (1 == SqlServerEngine.executeSQLCommand(createFundTransferItem))
E...	DataFactory.cs	150	executeSQLCommand("Insert into fsb_accounts " +	SqlServerEngine.executeSQLCommand("Insert into fsb_accounts " +

Save Source Code Text Search Drop Content Here

For example the `ChangeUserPassword` function of the `DataFactory.cs` file:



A quick search for `ChangeUserPassword`



Results in three matches:

	path	file	line #	match text	match line
▶	E...	DataFactory.cs	57	ChangeUserPassword	public static void ChangeUserPassword(string userID, string newPassword)
	E...	UserManagement.asmx.cs	108	ChangeUserPassword	public void ChangeUserPassword(string sessionId, string userID, string newPassword)
	E...	UserManagement.asmx.cs	110	ChangeUserPassword	HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID,newPassword);

The first one is the ChangeUserPassword (which we've seen before)

Results

	path	file	line #	match text	match line
▶	E...	DataFactory.cs	57	ChangeUserPassword	public static void ChangeUserPassword(string userID, string newPassword)
	E...	UserManagement.asmx.cs	108	ChangeUserPassword	public void ChangeUserPassword(string sessionId, string userID, string newPassword)
	E...	UserManagement.asmx.cs	110	ChangeUserPassword	HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID,newPassword);

Save Source Code Text Search

```
DataFactory.cs
57
58     public static void ChangeUserPassword(string userID, string newPassword)
59     {
60         SqlServerEngine.executeSQLCommand("Update fsb_users Set password = '" + newPassword + "' where user_id = " + userID + "");
61     }
62
```

The 2nd is a WebMethod call

Results

	path	file	line #	match text	match line
	E...	DataFactory.cs	57	ChangeUserPassword	public static void ChangeUserPassword(string userID, string newPassword)
▶	E...	UserManagement.asmx.cs	108	ChangeUserPassword	public void ChangeUserPassword(string sessionId, string userID, string newPassword)
	E...	UserManagement.asmx.cs	110	ChangeUserPassword	HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID,newPassword);

Save Source Code Text Search

```
UserManagement.asmx.cs
107
108     [WebMethod]
109     public void ChangeUserPassword(string sessionId, string userID, string newPassword)
110     {
111         HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID,newPassword);
112     }
```

Results				
	path	file	line #	match text
	E..	DataFactory.cs	57	ChangeUserPassword
	E..	UserManagement.asmx.cs	108	ChangeUserPassword
▶	E..	UserManagement.asmx.cs	110	ChangeUserPassword

```
107
108     [WebMethod]
109     public void ChangeUserPassword(string sessionID, string userID, string newPassword)
110     {
111         HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID, newPassword);
112     }
113
```

So clearly this is exploitable since there is a direct path from the externally faced WebMethod to the Sql Injection entry point.:

```
UserManagement.asmx.cs
107
108     [WebMethod]
109     public void ChangeUserPassword(string sessionID, string userID, string newPassword)
110     {
111         HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID, newPassword);
112     }
113

DataFactory.cs
57
58     public static void ChangeUserPassword(string userID, string newPassword)
59     {
60         SqlServerEngine.executeSQLCommand("Update fsb_users Set password = '" + newPassword + "' where user_id = " + userID + "'");
61     }
62

36
37     public static int executeSQLCommand(string sqlQueryToExecute)
38     {
39         Global.createSqlServerConnection();
40         //SqlConnection Global.globalSqlServerConnection = new SqlConnection(ConfigurationSettings.AppSettings.
41         string text1 = sqlQueryToExecute;
42         SqlCommand command1 = new SqlCommand(text1, Global.globalSqlServerConnection);
43         Global.globalSqlServerConnection.Open();
44         int executeNonQuery_Result = command1.ExecuteNonQuery();
45         Global.globalSqlServerConnection.Close();
46         return executeNonQuery_Result;
47     }
```

This should be an easy issue for Ounce to detect. The problem is that at the moment the Ounce AE doesn't recognize the WebMethod as a location where Tainted data will originate. In Ounce's nomenclature this is called a **Callback** (a good example of a callback is the *main(String[] args)* where the AE treats all parameters from that methods as tainted).

In 6.0 there is support for callbacks from the GUI, but in 5.0x we need to do it differently.

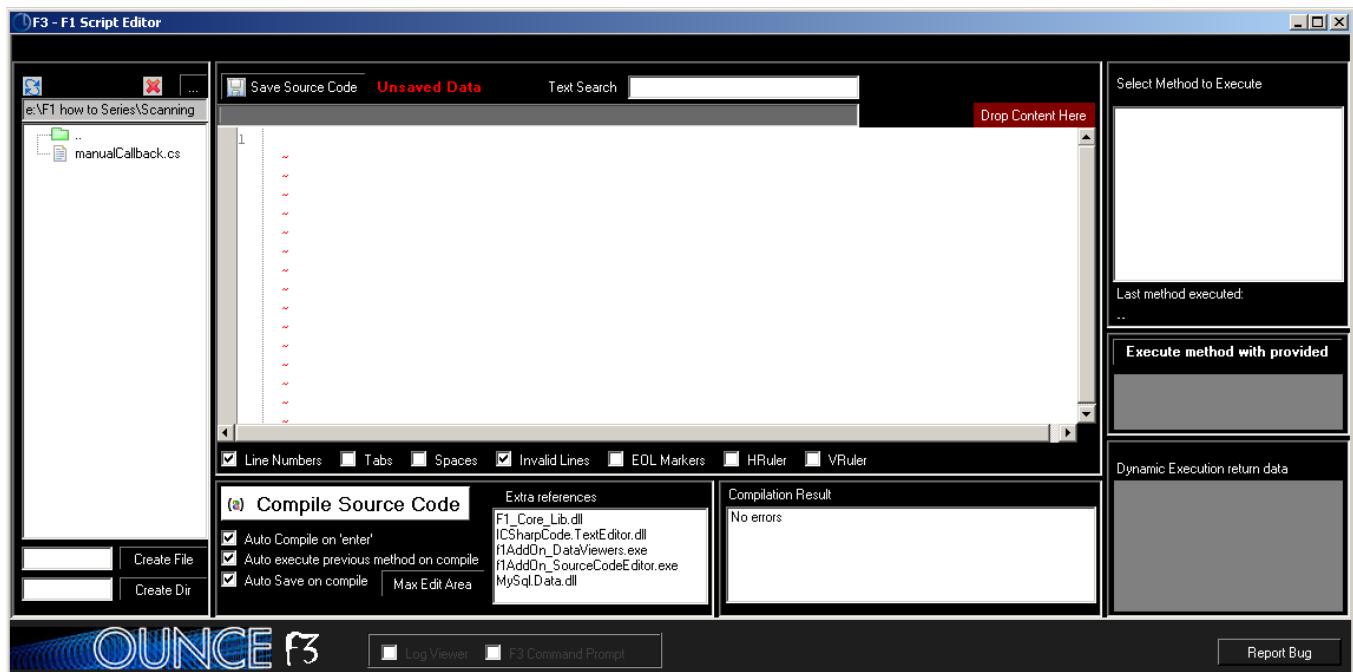
Using F1, there are 4 ways to create a Callback:

- 1) If there is a trace that covers the target method, we can use the *Custom Rules* Add-On (in this case there isn't one, so the *Custom Rules* Add-on will be covered in another *F1 how-to* document)
- 2) Once a consolidated F1CirData (from Cir Dumps) file is created we can use it with the *Custom Rules* Add-On (the problem is that the CIR doesn't contain the attributes data (i.e the [WebMethod]) so in this case we would still need to manually select which methods were WebMethods and should be marked as Callbacks (this will be covered in another *F1 how-to*)
- 3) Call the exposed F1 API that creates the Callbacks (see below)
- 4) Use the F1 Add-On *dotNet_CallbacksMaker_5_03* (see below)

USING F1 API TO CREATE A CALLBACK

F1 exposes most of its internal functions for invocation and scripting inside F1. In this first example we will invoke directly the function that creates the rules callbacks

Start the *f1AddOn_CSharp_Scripts.exe*



Before we get into the script example let's confirm that the desired API function is there.

Open the F3 Command Prompt

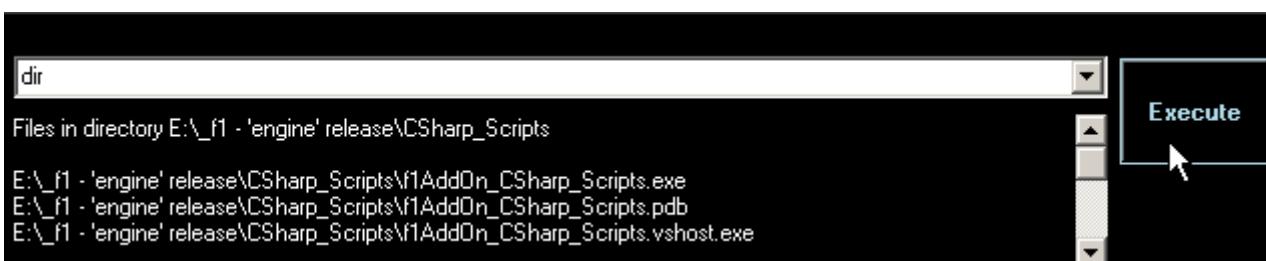


The Command Prompt allows the direct execution of commands, with some support of Intellisense

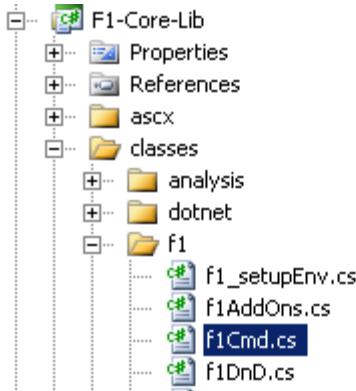
Click on the TextBox and press space to see a list of available commands and classes:



Typing `dir` and clicking on Execute (or pressing on Enter) will show the files in the current directory



`dir` in this case is a public static function located inside the `F1_core_lib.dll` dll and inside the `f1Cmd.cs` class file



```

namespace ounceLabs.F1.core_lib.classes.f1
{
    [core_lib.classes.F1_ExposedAndInvokableByF1]
    public class f1Cmd
    {

.....
        public static String dir()
        {
            List<String> lsFiles = new List<string>();
            files.getListOfAllFilesFromDirectory(lsFiles,sCurrentf1CmdWorkDirectory, false, "**.*",false);
            StringBuilder sbResult = new StringBuilder();
            sbResult.AppendLine("Files in directory " + sCurrentf1CmdWorkDirectory);
            sbResult.AppendLine();
            foreach (string sFile in lsFiles)
                sbResult.AppendLine(sFile);
            return sbResult.ToString();
        }
}

```

EXAMPLE OF USING THE COMMAND PROMPT

A good example of interaction between the *Command Prompt* and F1 is using debugMsg class to send a message to the LogViewer:



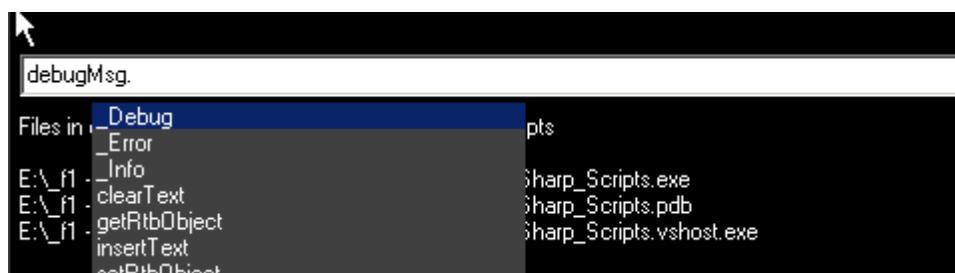
Make the LogViewer Visible:

And type the letters *de* on the *CommandPrompt* TextBox

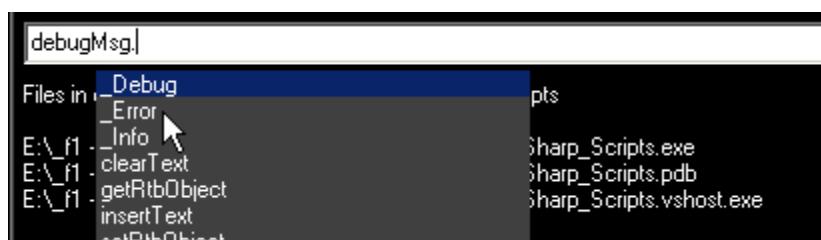


(note how the intelisense changes as the letters are presed)

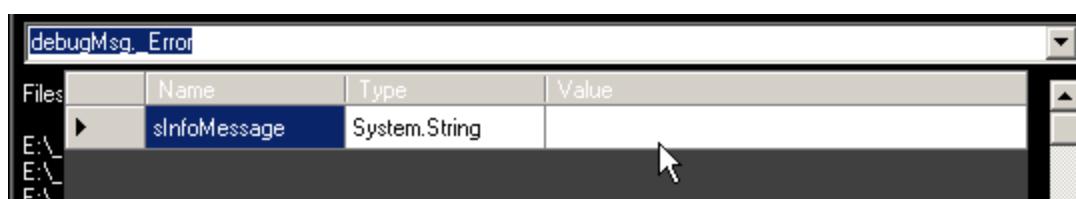
To select the current highlighted choice press on 'Left Arrow'



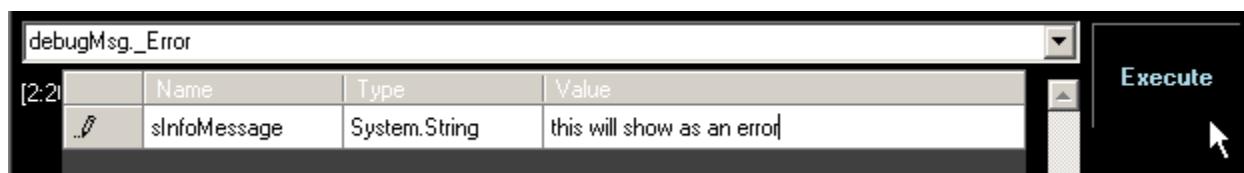
Or use the mouse to manually select from the ListBox the desired method:



Once a function is selected, a DataGridView will be dynamically populated with placeholders for the required parameters:



In this case just click on the *Value* cell and enter the message to send:



Click *execute*:



And note on the LogViewer the new entry:



Note: all past commands are added to a local cache which is visible by opening up the dropdown list:



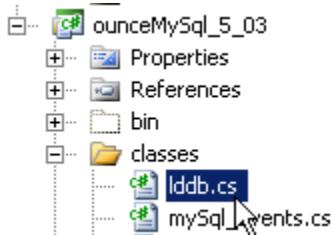
And accessed using the up and down arrows

INVOKING THE CREATE CALLBACK FROM THE *COMMAND PROMPT*

In the *Command Prompt* enter lddb.



The list shown are all exposed methods from the lddb class which is part of the *ounceMySql_5_03* dll



```

public static void action_DeleteAllCallbacks()
{
    String sSql = "DELETE from lddb_rec where lddb_rec.object=oResult = oounceMySql_5_03.executeSqlQuery(s);
}

public static void action_DeleteCallback(UInt32 UDb_Id)
{
    String sSql = String.Format("DELETE from lddb_rec object=oResult = oounceMySql_5_03.executeSqlQuery(s;
}

public static void action_DeleteAllCustomRules(bool bAlsoDeleteEdited)
{
    String sSql;
    if (bAlsoDeleteEdited)
        sSql = "DELETE from lddb_rec where lddb_rec.ad
object=oResult = oounceMySql_5_03.executeSqlQuery(s;
    if (oResult != null)
        debugMsg._Debug("# of rules deleted: {0}", oRe
}

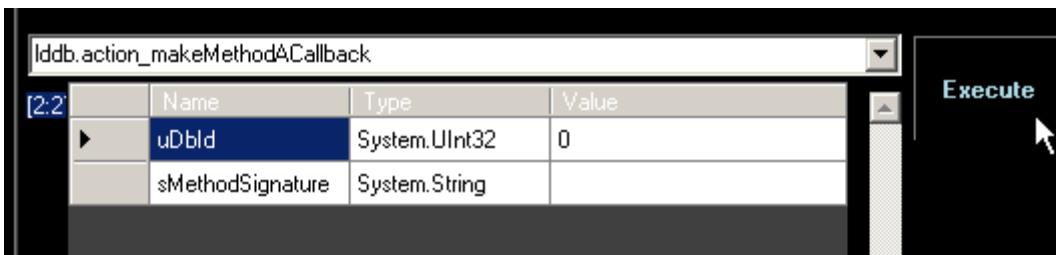
public static void action_DeleteActionId(UInt32 uActionObjectID)
{
    // finally set all vuln_id in lddb_actionobjects t
}

```

In Command Prompt enter call and select `action_makeMethodACallback`



And if the Db_Id and signature are known, enter the values and click execute:

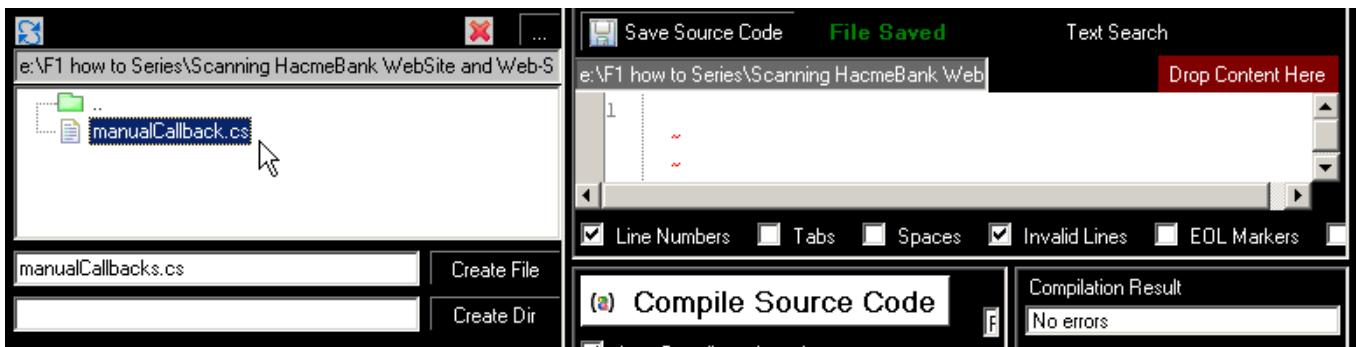


INVOKING THE CREATE CALLBACK FROM A SCRIPT

Going back to the `f1AddOn_CSharp_Scripts` create a new file called `manualCallbacks.cs`



Click on the newly created file:



And paste the following code :

```
using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
using ounceLabs.F1.core_classes;

class manualCallbacks
{
    public static String Hello()
    {
        String sMsg = "World";
        debugMsg._Info(sMsg);
        return sMsg;
    }
}
```

BUG ALERT: Sometimes when opening the CSharp Code editor on new files (i.e files with no content, the Editor Control will stop behaving normally. If that happens, just close the [f1AddOn_CSharp_Scripts](#) Add-On and try again (the 2nd time around always works ok)

Now click on *Compile Source Code*

The screenshot shows the F1 Source Code Editor interface. On the left, a code editor window displays a C# script named 'man' with the following content:

```

1  using System;
2  using System.Drawing;
3  using System.Threading;
4  using System.Windows.Forms;
5  using ounceLabs.F1.core_lib.classes;
6
7  class manualCallbacks
8  {
9      public static String Hello()
10     {
11         String sMsg = "World";
12         debugMsg._Info(sMsg);
13         return sMsg;
14     }
15 }
16
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~

```

The status bar at the bottom indicates 'Line Numbers', 'Tabs', 'Spaces', 'Invalid Lines', 'EOL Markers', 'HRule', and 'VRuler'.

The right side of the interface includes a 'Select Method to Execute' window with a tree view showing 'manualCallbacks' and 'Hello'. Below it are sections for 'Last method executed' (empty), 'Execute method with provided' (disabled), and 'Dynamic Execution return data' (empty).

A 'Compile Source Code' button is visible at the bottom left of the main editor area. A tooltip for this button shows options: 'Auto Compile on 'enter'', 'Auto execute previous method on compile', and 'Auto Save on compile'. It also includes a 'Max Edit Area' button.

The 'Compilation Result' section shows 'No errors'.

There should be no errors and the top right ListBox should now be populated with a dynamic list of the *manualCallbacks* class

Click on **manualCallbacks** and on **Execute method with provided** to execute this method:

```

8   class manualCallbacks
9   {
10     public static String Hello()
11     {
12       String sMsg = "World";
13       debugMsg._Info(sMsg);
14       return sMsg;
15     }
16   }
17

```

Last method executed:
Hello

Dynamic Execution return data
World

F3 - F1 Script Editor

Log Viewer

```

[3:47 PM] INFO: World
[3:43 PM] INFO: Added type manualCallbacks as var
[3:43 PM] DEBUG: Dynamically compiling Source code...
[3:14 PM] DEBUG: Dynamically compiling Source code...
[3:14 PM] DEBUG: Source code file loaded: e:\F1 how to Series\Scanning Hacm
[3:14 PM] DEBUG: Loading File: e:\F1 how to Series\Scanning HacmeBank Web
[3:12 PM] INFO: Added type manualCallbacks as var
[3:12 PM] DEBUG: Dynamically compiling Source code...
[3:12 PM] DEBUG: Source code file loaded: e:\F1 how to Series\Scanning Hacm
[3:12 PM] DEBUG: Loading File: e:\F1 how to Series\Scanning HacmeBank Web
[3:07 PM] DEBUG: in resolveDataGridValuesBasedOnCurrentCmdParameters, no
[2:27 PM] ERROR:
[2:27 PM] DEBUG: executing Method: Void _Error(System.String) with #1 params
[2:26 PM] ERROR: this will show as an error
[2:26 PM] DEBUG: executing Method: Void _Error(System.String) with #1 params
[2:15 PM] DEBUG: executing Method: System.String dir() with #0 params
[2:15 PM] DEBUG: executing Method: System.String dir() with #0 params
[2:10 PM] DEBUG: Dynamically compiling Source code...
[2:10 PM] DEBUG: Source code file loaded: e:\F1 how to Series\Scanning Hacm
[2:10 PM] DEBUG: Loading File: e:\F1 how to Series\Scanning HacmeBank Web
[2:10 PM] DEBUG: Source code file loaded: e:\F1 how to Series\Scanning Hacm
[2:10 PM] DEBUG: Loading File: e:\F1 how to Series\Scanning HacmeBank Web
[2:10 PM] INFO: Added type SimpleTests as var
[2:10 PM] DEBUG: Dynamically compiling Source code...

```

Save Source Code File Saved Text Search

e:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Co

```

1  using System;
2  using System.Drawing;
3  using System.Threading;
4  using System.Windows.Forms;
5  using ounceLabs.F1.core_lib.classes;
6
7
8  class manualCallbacks
9  {
10    public static String Hello()
11    {
12      String sMsg = "World";
13      debugMsg._Info(sMsg);
14      return sMsg;
15    }
16

```

Just to show a couple more features of the Script Editor:

Enter the following code below the *Hello* function

```

public static String DynamicString(String sDynamicString)
{
  String sMsg = "This is a: " + sDynamicString;
  debugMsg._Info(sMsg);
  return sMsg;
}

```

And click on *Compile Source Code*

```
8 class manualCallbacks
9 {
10     public static String Hello()
11     {
12         String sMsg = "World";
13         debugMsg._Info(sMsg);
14         return sMsg;
15     }
16
17     public static String DynamicString(String sDynamicString)
18     {
19         String sMsg = "This is a: " + sDynamicString;
20         debugMsg._Info(sMsg);
21         return sMsg;
22     }
23 }
24
```

Line Numbers Tabs Spaces Invalid Lines EOL Markers HRuler VRuler

Compile Source Code references
 Auto Compile on 'enter'
 Auto execute previous method on compile
 Auto Save on compile Max Edit Area

Compilation Result
No errors

The compilation result should show no errors and the dynamic list of methods should now have the *DynamicString* function

Click on this *DynamicString* node and note how the DataGridView below now has a row with the required parameters (in this case the String *sDynamicString*)

Select Method to Execute

- ☐ manualCallbacks
 - └= Hello
 - └= DynamicString

Last method executed:
DynamicString

Execute method with provided Parameters

	Name	Type	Value
▶	sDynamicString	System.String	

Enter a value in the editable cell and click on 'Execute method with provided Parameters'

```
8     class manualCallbacks
9     {
10         public static String Hello()
11         {
12             String sMsg = "World";
13             debugMsg._Info(sMsg);
14             return sMsg;
15         }
16
17         public static String DynamicString(String sDynamicString)
18         {
19             String sMsg = "This is a: " + sDynamicString;
20             debugMsg._Info(sMsg);
21             return sMsg;
22         }
23     }
```

Execute method with provided Parameters

Name	Type	Value
sDynamicString	System.String	Dynamic test

Dynamic Execution return data
This is a: Dynamic test

ERROR HANDLING

When errors occur in the C# code detailed error messages are shown in the Compilation Result TextBox

```
14     return sMsg;
15 }
16 aaaa
17 publc static String DynamicString(String sDynamicString)
18 {
19     String sMsg = "This is a: " + sDynamicString;
20     debugMsg._Info(sMsg);
21     return sMsg;
22 }
23 }
```

Line Numbers Tabs Spaces Invalid Lines EOL Markers HRuler VRuler

(i) **Compile Source Code** References Compilation Result

Auto Compile on 'enter'
 Auto execute previous method on compile
 Auto Save on compile

F1_Core.Lib.dll
ICSharpCode.TextEdit
or.dll
f1AddOn_DataViewers
.exe

17:2:CS1585:Member modifier 'public' must precede the member type and name

Click on the error and its location will be highlighted

The screenshot shows a script editor interface with the following details:

- Code Area:**

```

15 }
16     aaaa
17     public static String DynamicString(String sDynamicString)
18     {
19         String sMsg = "This is a: " + sDynamicString;
20         debugMsg._Info(sMsg);
21         return sMsg;
22     }
23 }
24 
```
- Toolbars:** Line Numbers, Tabs, Spaces, Invalid Lines, EDL Markers, HRuler, VRuler.
- Compile Source Code Panel:** Shows the file `F1_Core.Lib.dll` and the tab `ICSharpCode.TextEdit`.
- Compilation Result Panel:** Displays the error message: `17:2:CS1585:Member modifier 'public' must precede the member type and name`.

TIPs: When the cursor is inside the Source Code Edit area, if the **Auto Compile on 'enter'** is set automatic compilation can be triggered by the following sequence: {Select an text area (Shift+right-arrow)} + Enter. Then if **Auto execute previous method on compile** is also set, the last executed method will be automatically re-executed (this simple feature saves considerable about of time since it automates a process that would take at least 3 clicks and at least 20 secs).

The **Auto Save on compile** auto saves on compile and will collapse the left and right panels (only leaving the script editor and preferences panels)

CREATING CALLBACK

On the script editor, replace the existing code with

```

using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
using ounceLabs.F1.core_lib.classes;

class manualCallbacks
{
    public static String Hello()
    {
        String sMsg = "World";
        debugMsg._Info(sMsg);
        return sMsg;
    }

    public static void CreateCallback(UInt32 uDatatabaseId, String sNewCallbackName)
    {
        // code
    }
}

```

The code should compile with no errors:

The screenshot shows a custom F1 script editor interface. The main window displays the following C# code:

```

1  using System;
2  using System.Drawing;
3  using System.Threading;
4  using System.Windows.Forms;
5  using ounceLabs.F1.core_lib.classes;
6
7
8  class manualCallbacks
9  {
10     public static String Hello()
11     {
12         String sMsg = "World";
13         debugMsg._Info(sMsg);
14         return sMsg;
15     }
16
17     public static void CreateCallback(UInt32 uDatatabaseId, String sNewCallbackName)
18     {
19         // code
20     }
21 }
22

```

The sidebar on the right contains the following sections:

- Select Method to Execute:** Shows a tree view with a single node: `manualCallbacks` > `Hello`.
- Last method executed:** `CreateCallback`
- Execute method with:** A table showing parameters:

Name	Type	Value
uDa...	Sys...	0
sNe...	Sys...	
- Dynamic Execution return data:** Info: Method CreateCallback e>

At the bottom of the editor, there are checkboxes for Line Numbers, Tabs, Spaces, Invalid Lines, EOL Markers, HRuler, and VRuler. The status bar shows '(a) Compile Source Code' and 'F1 Core Lib.dll'.

Add the following using reference:

```
using ounceLabs.F1.f1AddOn.ounceMySql.classes;
```

and line inside the CreateCallback function:

```
lddb.action_makeMethodACallback(uDatatabaseId, sNewCallbackName);
```

and that's it:

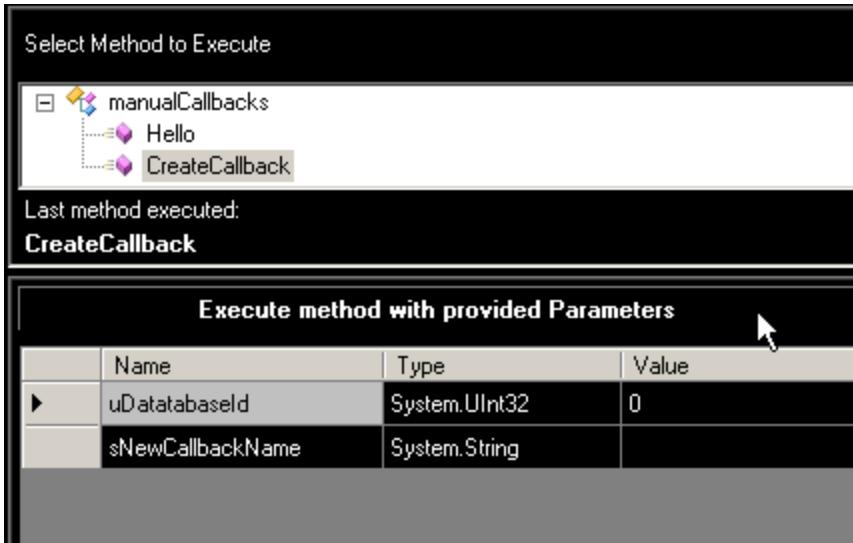
The screenshot shows the F1 script editor with the following code:

```

1  using System;
2  using System.Drawing;
3  using System.Threading;
4  using System.Windows.Forms;
5  using ounceLabs.F1.core_lib.classes;
6  using ounceLabs.F1.f1AddOn.ounceMySql.classes;
7
8
9  class manualCallbacks
10 {
11     public static String Hello()
12     {
13         String sMsg = "World";
14         debugMsg._Info(sMsg);
15         return sMsg;
16     }
17
18     public static void CreateCallback(UInt32 uDatatabaseId, String sNewCallbackName)
19     {
20         lddb.action_makeMethodACallback(uDatatabaseId, sNewCallbackName);
21     }
22 }

```

Compile it and execute it using the right-hand side dynamic execution environment:



EXECUTING SCRIPT CREATED USING COMMAND PROMPT

Another way to execute this method is to use the *CommandPrompt*.

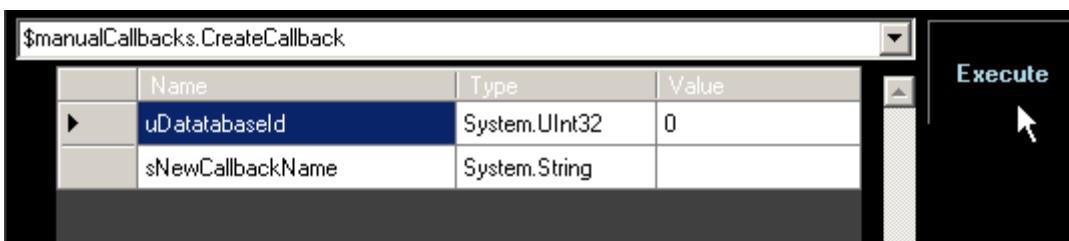
On successful compilation the newly created class is placed on a global (to F1) dynamic variable which can be invoked from the *Command Prompt*.



Select the *manualCallbacks* and press . (dot) to show the functions available)



Select *CreateCallback* enter the desired values and click execute



ADDING MORE USEFUL FUNCTIONS TO SCRIPT

Replace the previous script with this code:

```
using System;
using System.Data;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
using ounceLabs.F1.core_lib.classes;
using ounceLabs.F1.f1AddOn.ounceMySql.classes;

class manualCallbacks
{
    public static String Hello()
    {
        String sMsg = "World";
        debugMsg._Info(sMsg);
        return sMsg;
    }
    public static void testCase()
    {
        //deleteAllCallbacks(); // to delete all callbacks in database
        deleteCallback(2,"callbackSignature_1(string;string):void"); // to delete the
one that will be created next
        createCallback(2,"callbackSignature_1(string;string):void");
        createCallback(2,"callbackSignature_2(string;string):void"); // this will throw
an error the 2nd time it is executed (duplicate callback entries are not allowed)
        listAllCallbacksInDatabase();
    }

    public static void createCallback(UInt32 uDatatabaseId, String sNewCallbackName)
    {
        lddb.action_makeMethodACallback(uDatatabaseId, sNewCallbackName);
    }

    public static void listAllCallbacksInDatabase()
    {
        DataTable dtDataTable =
ounceMySql_5_03.getDataTableWith_Callbacks();
        debugMsg._Debug("Listing all Callbacks in lddb database ({0}
items)",dtDataTable.Rows.Count);
        foreach(DataRow dgvRow in dtDataTable.Rows)
            debugMsg._Info("\t{0}",
System.Text.Encoding.ASCII.GetString((byte[])dgvRow["signature"]));
    }

    public static void deleteAllCallbacks()
    {
        lddb.action_DeleteAllCallbacks();
    }

    public static void deleteCallback(UInt32 uDatatabaseId, String sCallbackToDelete)
    {
        lddb.action_DeleteCallback(uDatatabaseId,sCallbackToDelete);
    }
}
```

The script should compile with no errors.

The screenshot shows the F1 Script Editor interface. On the left is a code editor with the following C# code:

```

18 public static void testCase()
19 {
20     //deleteAllCallbacks();           // to delete all callbacks in database
21     deleteCallback(2,"callbackSignature_1(string:string):void"); // to delete the one that will be
22     createCallback(2,"callbackSignature_1(string:string):void");
23     createCallback(2,"callbackSignature_2(string:string):void"); // this will thrown an error the
24     listAllCallbacksInDatabase();
25 }
26
27 public static void createCallback(UInt32 uDatatabaseId, String sNewCallbackName)
28 {
29     lddb.action_makeMethodACallback(uDatatabaseId, sNewCallbackName);
30 }
31
32 public static void listAllCallbacksInDatabase()
33 {
34     DataTable dtDataTable = ounceMySql_5_03.getDataTableWith_Callbacks();
35     debugMsg._Debug("Listing all Callbacks in lddb database ((0) items)",dtDataTable.Rows.Count);
36     foreach(DataRow dgvRow in dtDataTable.Rows)
37         debugMsg._Info("\t{0}", System.Text.Encoding.ASCII.GetString((byte[])dgvRow["signature"]));
38 }
39
40 public static void deleteAllCallbacks()
41 {
42     lddb.action_DeleteAllCallbacks();
43 }
44
45 public static void deleteCallback(UInt32 uDatatabaseId, String sCallbackToDelete)
46 {
47     lddb.action_DeleteCallback(uDatatabaseId,sCallbackToDelete);
48 }
49 }
50

```

At the bottom of the code editor are several checkboxes: Line Numbers, Tabs, Spaces, Invalid Lines, EOL Markers, HRuler, VRuler. Below the code editor are two tabs: "Compile Source Code" (selected) and "F1_Core.Lib.dll". The status bar shows "No errors".

To the right of the code editor is a LogViewer window titled "manualCallbacks". It shows the following log entries:

- Last method executed: testCase
- Execute method with provided
- Dynamic Execution return data
- Info: Method testCase executed sucessf

Execute `testCase` and the LogViewer should look like this the first time it is executed:

The Log Viewer window shows the following log entries:

```

[12:36 PM] INFO: callbackSignature_2(string:string):void
[12:36 PM] INFO: callbackSignature_1(string:string):void
[12:36 PM] DEBUG: Listing all Callbacks in lddb database (2 items)
[12:36 PM] DEBUG: Making method [2] callbackSignature_2(string:string):void a Callback
[12:36 PM] DEBUG: Making method [2] callbackSignature_1(string:string):void a Callback
[12:36 PM] INFO: Added type manualCallbacks as var
[12:36 PM] DEBUG: Dynamically compiling Source code...

```

And like this the 2nd time:

The Log Viewer window shows the following log entries:

```

[12:36 PM] INFO: callbackSignature_1(string:string):void
[12:36 PM] INFO: callbackSignature_2(string:string):void
[12:36 PM] DEBUG: Listing all Callbacks in lddb database (2 items)
[12:36 PM] ERROR: in action_makeMethodACallback: There was already an entry in the db for this rule, so can't add a callback: callbackSignature_2(string:string):void
[12:36 PM] DEBUG: Making method [2] callbackSignature_2(string:string):void a Callback
[12:36 PM] DEBUG: Making method [2] callbackSignature_1(string:string):void a Callback
[12:36 PM] INFO: Added type manualCallbacks as var
[12:36 PM] DEBUG: Dynamically compiling Source code...

```

USING DOTNET_CALLBACKSMAKER_5_03 ADD-ON TO AUTOMATICALLY FIND WEB SERVICES FUNCTIONS AND MAKE THEM ALL CALLBACKS

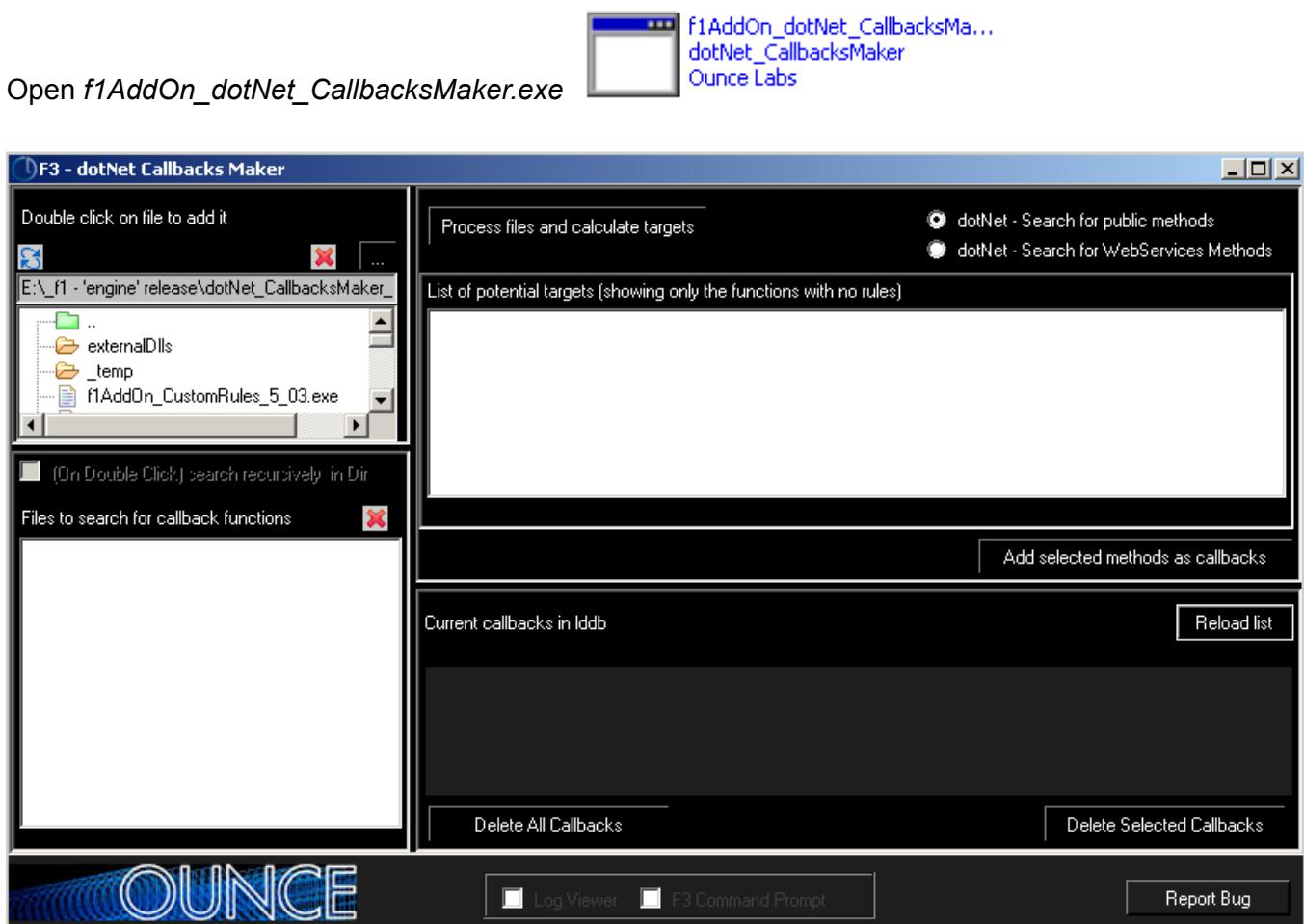
There were three main problems with the previous examples:

- 1) the need to know the correct function signature in a format that will be recognized by CORE's AE
- 2) the detection process of the methods marked with the [WebMethod] attribute
- 3) the manual (one function at the time) process

Ideally what we need is to have an automated process to analyze a .Net *.dll file and automatically create the Callbacks rules.

And this is exactly what the *dotNet_CallbacksMaker_5_03* Add-on does

FINDING WEBSERVICES FUNCTIONS AND ADDING 1 CALLBACK



TIPs: For ASP.NET websites the dlls are not automatically created on build, so to access one can:

- 1) use the temp files created by CORE before a scan:

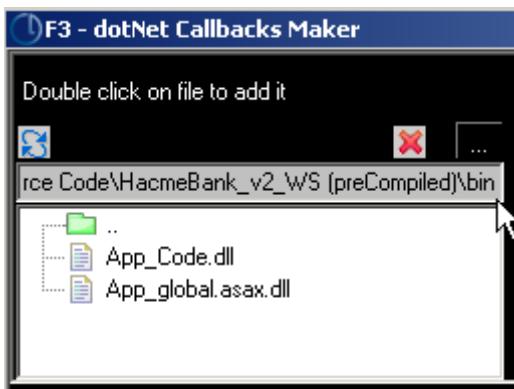
Address C:\WINDOWS\Temp\2			
Name	Size	Type	Date Modified
bin		File Folder	6/23/2008 9:35 AM
install		File Folder	6/23/2008 9:35 AM
WebServices		File Folder	6/23/2008 9:35 AM

- 2) run the aspnet_compiler.exe tool

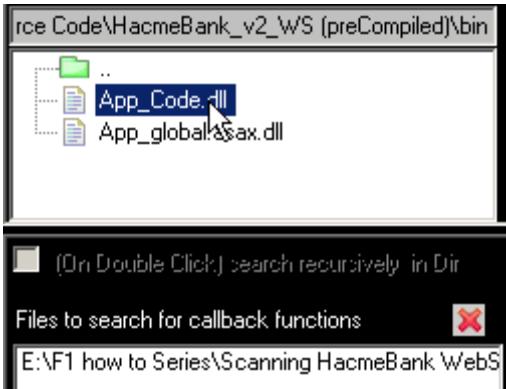
```
aspnet_compiler -p HacmeBank_v2_WS -v /HacmeBank_v2_WS "HacmeBank_v2_WS (preCompiled)"
```

Address E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS (preCompiled)\bin			
Name	Size	Type	Date Modified
accountmanagement.asmx.f0e9dea3.compiled	1 KB	COMPILED File	6/26/2008 2:19 PM
App_Code.compiled	1 KB	COMPILED File	6/26/2008 2:19 PM
App_Code.dll	28 KB	Application Extension	6/26/2008 2:19 PM
App_global.asax.compiled	1 KB	COMPILED File	6/26/2008 2:19 PM
App_global.asax.dll	4 KB	Application Extension	6/26/2008 2:19 PM
install.asmx.c29b009d.compiled	1 KB	COMPILED File	6/26/2008 2:19 PM
usermanagement.asmx.f0e9dea3.compiled	1 KB	COMPILED File	6/26/2008 2:19 PM
userscommunity.asmx.f0e9dea3.compiled	1 KB	COMPILED File	6/26/2008 2:19 PM

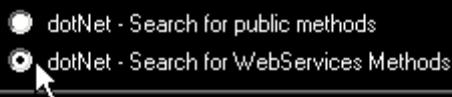
Using the top-left directory control, navigate to the directory with the precompiled .Net dll (enter the address and press enter: *E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS (preCompiled)\bin*)



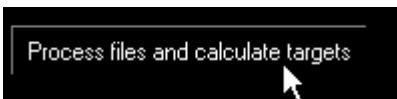
Double click on App_Code.dll to add it to the ListBox with *Files to search for callbacks functions*



Select the *dotnet – Search for WebServices Methods* radio box so that only the Web Methods are shown



Click on *Process files and calculate targets*



And all methods with the [WebMethod] attribute are listed:



Following the example shown before in this document, we will use the ChangeUserPassword function which was vulnerable to SQL Injection

:

UserManagement.asmx.cs

```
107      [WebMethod]
108      public void ChangeUserPassword(string sessionID, string userID, string newPassword)
109      {
110          HacmeBank_v2_WS.DataFactory.ChangeUserPassword(userID,newPassword);
111      }
112  }
```

To *ChangeUserPassword* a callback, select it and press on *Add selected methods as callbacks*

List of potential targets (showing only the functions with no rules)

```
HacmeBank_v2_WS.WS_AccountManagement.CreateAccount(string;string;string;string;string;string;string;string):void
HacmeBank_v2_WS.WS_AccountManagement.ExecuteSqlQuery(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_AccountManagement.GetAccountDetails_using_AccountID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_AccountManagement.GetAccountTransactionDetails_using_TransactionID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_AccountManagement.GetAccountTransactions_using_AccountID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_AccountManagement.GetLoanRates(string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_AccountManagement.GetUserAccounts_using_UserID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_AccountManagement.MakePayment_Using_CreditCard(string;string;string;string;int;string):int
HacmeBank_v2_WS.WS_AccountManagement.RequestALoan(string;string;int;int;System.Decimal;string):int
HacmeBank_v2_WS.WS_AccountManagement.TransferFunds(string;string;string;double;string):int
HacmeBank_v2_WS.WS_UserManagement.ChangeUserPassword(string;string;string):void
HacmeBank_v2_WS.WS_UserManagement.CreateUser(string;string;string;string):void
HacmeBank_v2_WS.WS_UserManagement.DeleteUser(string;string):int
HacmeBank_v2_WS.WS_UserManagement.GetUserDetail_using_loginID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_UserManagement.GetUserDetail_using(userID:string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_UserManagement.GetUserDetail_using_userName(string;string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_UserManagement.ListCurrentUsers(string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_UserManagement.Login(string;string):string
HacmeBank_v2_WS.WS_UserManagement.UpdateUserDetails(string;string;string;string;string):void
HacmeBank_v2_WS.WS_UsersCommunity.DeleteMessage(string;string):int
HacmeBank_v2_WS.WS_UsersCommunity.GetPostedMessages(string):System.Collections.ArrayList
HacmeBank_v2_WS.WS_UsersCommunity.PostMessage(string;string;string;string):void
```

Add selected methods as callbacks

Click on *Reload List* and confirm that the rule has been added

Current callbacks in lddb

Reload list

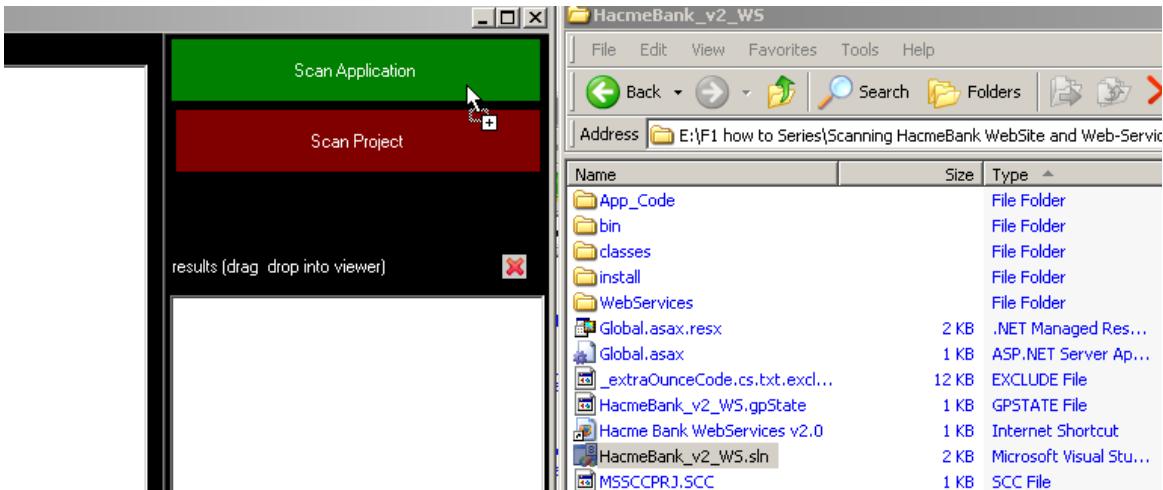
db_id	vuln_id	vuln_name	signature	added	modified	packag	class	version	callback
3	3011134	HacmeBank_v2_WS.WS_UserManagement.ChangeUserPassword	X	1	0			0	1

RE-SCANNING APPLICATION AND ANALYSING NEW TRACE

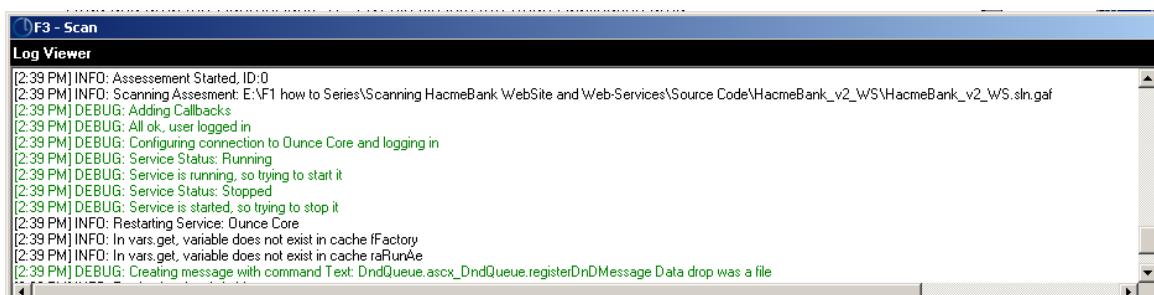
With the new Callback in place we are need to rescan the application:



Drag and drop the HacmeBank_v2_WS.sln file into the *Scan Application* area

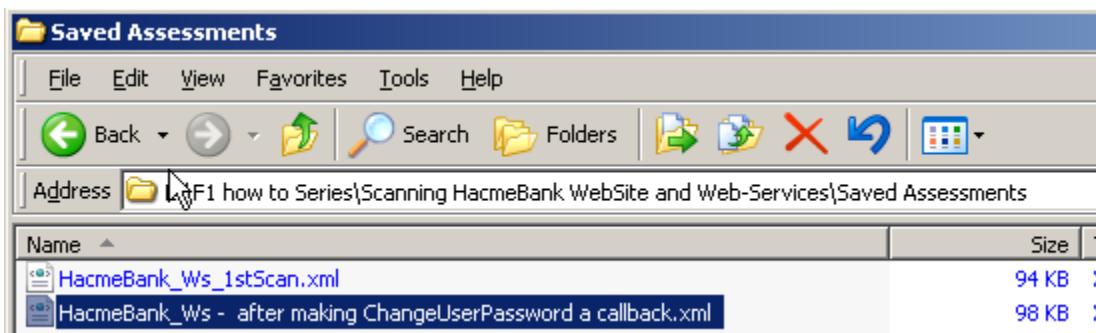
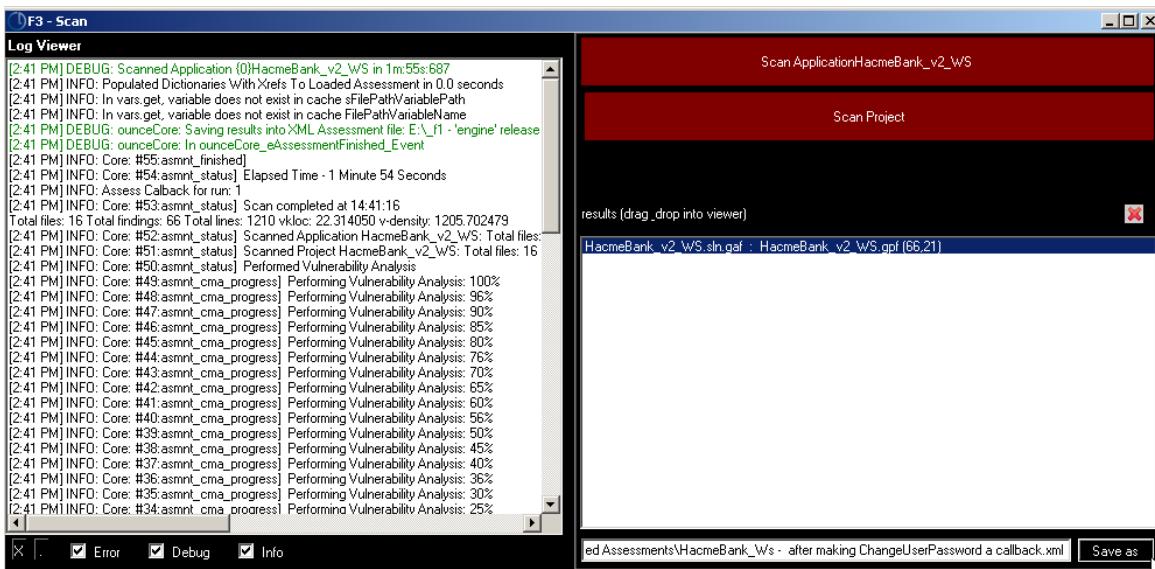


If this is the first time a scan is executed from the current open instance of f1AddOn_Scan_5_03.exe the Ounce Core Service will be restarted and the connection re-established:



IMPORTANT NOTE: After making any Custom Rules changes on the CORE database, the Ounce CORE services must be restarted! F1 makes direct changes to the CORE's database and at the moment those changes cannot be dynamically reloaded

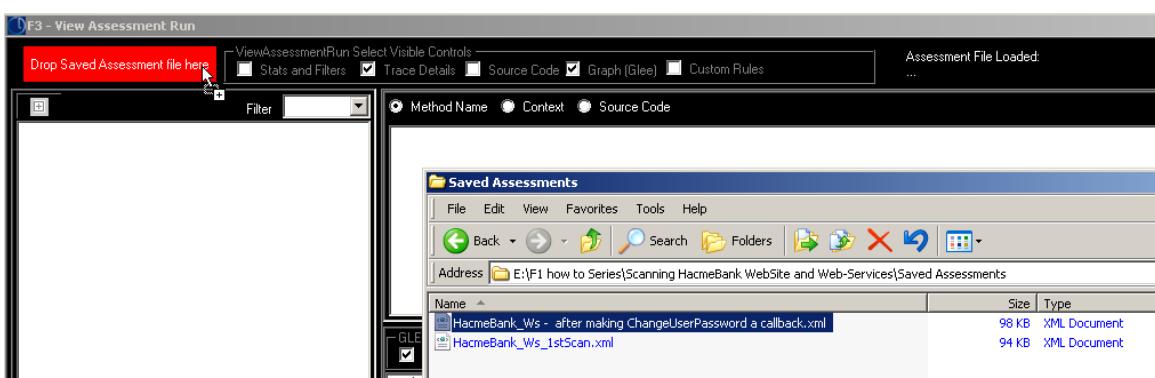
Once scan is completed select the new line in the *results* ListBox to save the assessment to a local folder:



Open to [f1AddOn_ViewAssessmentRun_5_03.exe](#)

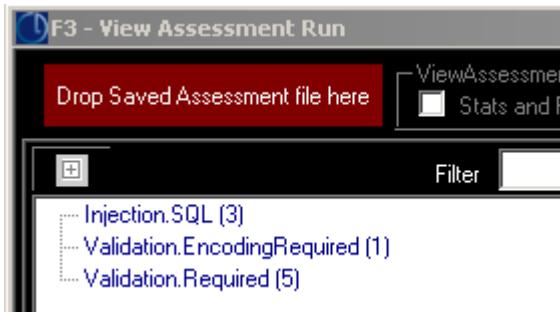


Drag and drop the Saved Assessment file into the *Drop Saved Assessment file here* control area

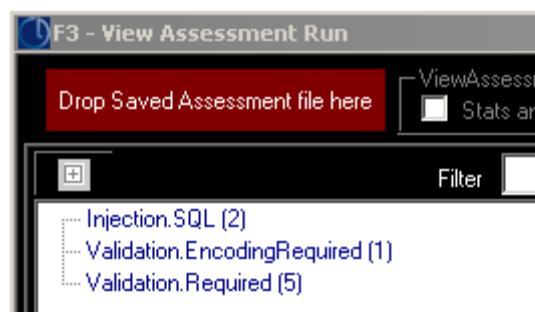


And the scan will be loaded (below are the number of traces in the current scan (with the **ChangeUserPassword** callback) and in the previous one (with no callbacks))

Current Scan



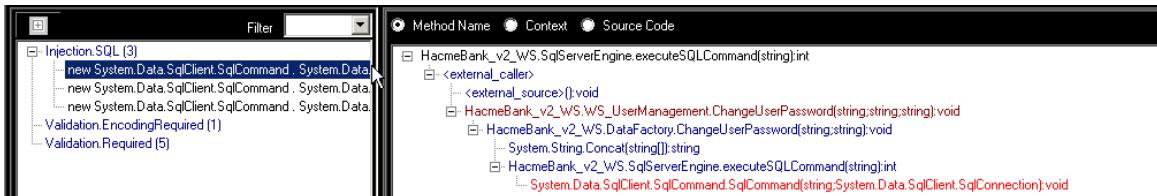
Previous scan



Click on *Injection.SQL (3)* to expand the node¹

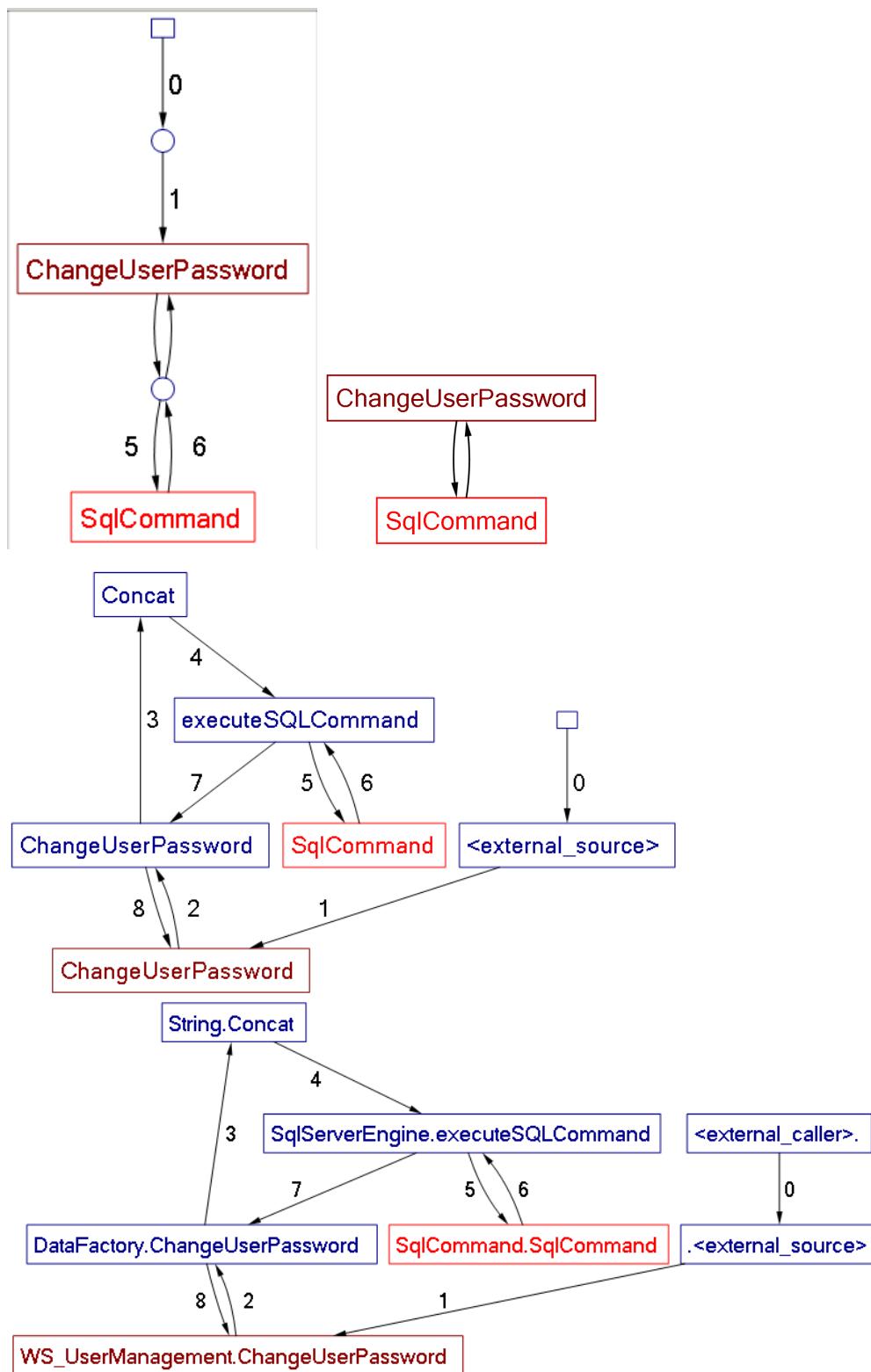


Click on the first child node and see that now we have a trace from the ChangeUserPassword source to the SqlCommand Sink



The GLEE graph can also be used to analyze this trace:

¹ This TreeView delay loads the findings (i.e. only when selected) which allows faster rendering of this control for assessment files that have large number of findings





ADDING ALL CALLBACKS, RESCANNING AND VIEWING

Go back to the *f1AddOn_CustomRules_5_03.exe*, select all functions and add them:

List of potential targets (showing only the functions with no rules)

```

HacmeBank_v2_WS_Ws_AccountManagement.CreateAccount(string;string;string;string;string;string;string):void
HacmeBank_v2_WS_Ws_AccountManagement.ExecuteSqlQuery(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_AccountManagement.GetAccountDetails_using_AccountID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_AccountManagement.GetAccountTransactionDetails_using_TransactionID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_AccountManagement.GetAccountTransactions_using_AccountID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_AccountManagement.GetLoanRates(string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_AccountManagement.GetUserAccounts_using_UserID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_AccountManagement.MakePayment_Using_CreditCard(string;string;string;string;int;string):int
HacmeBank_v2_WS_Ws_AccountManagement.RequestALoan(string;string;int;int;System.Decimal;string):int
HacmeBank_v2_WS_Ws_AccountManagement.TransferFunds(string;string;string;double;string):int
HacmeBank_v2_WS_Ws_UserManagement.ChangeUserPassword(string;string;string):void
HacmeBank_v2_WS_Ws_UserManagement.CreateUser(string;string;string;string):void
HacmeBank_v2_WS_Ws_UserManagement.DeleteUser(string;string):int
HacmeBank_v2_WS_Ws_UserManagement.GetUserDetail_using_loginID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_UserManagement.GetUserDetail_using_userID(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_UserManagement.GetUserDetail_using_userName(string;string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_UserManagement.ListCurrentUsers(string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_UserManagement.Login(string;string):string
HacmeBank_v2_WS_Ws_UserManagement.UpdateUserDetails(string;string;string;string;string):void
HacmeBank_v2_WS_Ws_UsersCommunity.DeleteMessage(string;string):int
HacmeBank_v2_WS_Ws_UsersCommunity.GetPostedMessages(string):System.Collections.ArrayList
HacmeBank_v2_WS_Ws_UsersCommunity.PostMessage(string;string;string;string):void

```

Add selected methods as callbacks

The LogViewer should look like this (the error is caused by the fact that we previously added the *ChangeUserPassword* function as a callback):

F3 - dotNet Callbacks Maker

Log Viewer

```
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UsersCommunity.PostMessage(string:string:string:string):void a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UsersCommunity.GetPostedMessages(string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UsersCommunity.DeleteMessage(string:string):int a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.UpdateUserDetails(string:string:string:string):void a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.Login(string:string):string a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.ListCurrentUsers(string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.GetUserDetail_using.userName(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.GetUserDetail_using.userID(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.GetUserDetail_using.loginID(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.DeleteUser(string:string):int a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.CreateUser(string:string:string):void a Callback
[3:13 PM] ERROR: in action_makeMethodACallback: There was already an entry in the db for this rule, so can't add a callback: HacmeBank_v2_WS.WS_UserManagement.ChangeUserPassword(string:string:string):void
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_UserManagement.ChangeUserPassword(string:string:string):void a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.TransferFunds(string:string:string:double:string):int a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.RequestALoan(string:string:int:System.Decimal:string):int a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.MakePayment_Using_CreditCard(string:string:string:int:string):int a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.GetUserAccounts_using.UserID(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.GetLoanRates(string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.GetAccountTransactions_using_AccountID(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.GetAccountTransactionDetails_using_TransactionID(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.GetAccountDetails_using_AccountID(string:string) System.Collections.ArrayList a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.ExecuteSqlQuery(string:string):void a Callback
[3:13 PM] DEBUG: Making method [3] HacmeBank_v2_WS.WS_AccountManagement.CreateAccount(string:string:string:string:string):void a Callback
```

Click on *Reload List* will also confirm that the callbacks are in place:

Current callbacks in lddb

db_id	vuln_id	vuln_name	sign	addr	modi	pack	class	versi	callb
3	3011134	HacmeBank_v2_WS.WS_UserManagement.ChangeUserPassword	x	1	0			0	1
3	3011135	HacmeBank_v2_WS.WS_AccountManagement.CreateAccount	x	1	0			0	1
3	3011136	HacmeBank_v2_WS.WS_AccountManagement.ExecuteSqlQuery	x	1	0			0	1
3	3011137	HacmeBank_v2_WS.WS_AccountManagement.GetAccountDetails_...	x	1	0			0	1
3	3011138	HacmeBank_v2_WS.WS_AccountManagement.GetAccountTransact...	x	1	0			0	1
3	3011139	HacmeBank_v2_WS.WS_AccountManagement.GetAccountTransact...	x	1	0			0	1
3	3011140	HacmeBank_v2_WS.WS_AccountManagement.GetLoanRates	x	1	0			0	1
3	3011141	HacmeBank_v2_WS.WS_AccountManagement.GetUserAccounts_u...	x	1	0			0	1
3	3011142	HacmeBank_v2_WS.WS_AccountManagement.MakePayment_Usin...	x	1	0			0	1
3	3011143	HacmeBank_v2_WS.WS_AccountManagement.RequestALoan	x	1	0			0	1
3	3011144	HacmeBank_v2_WS.WS_AccountManagement.TransferFunds	x	1	0			0	1
3	3011145	HacmeBank_v2_WS.WS_UserManagement.CreateUser	x	1	0			0	1
3	3011146	HacmeBank_v2_WS.WS_UserManagement.DeleteUser	x	1	n			0	1

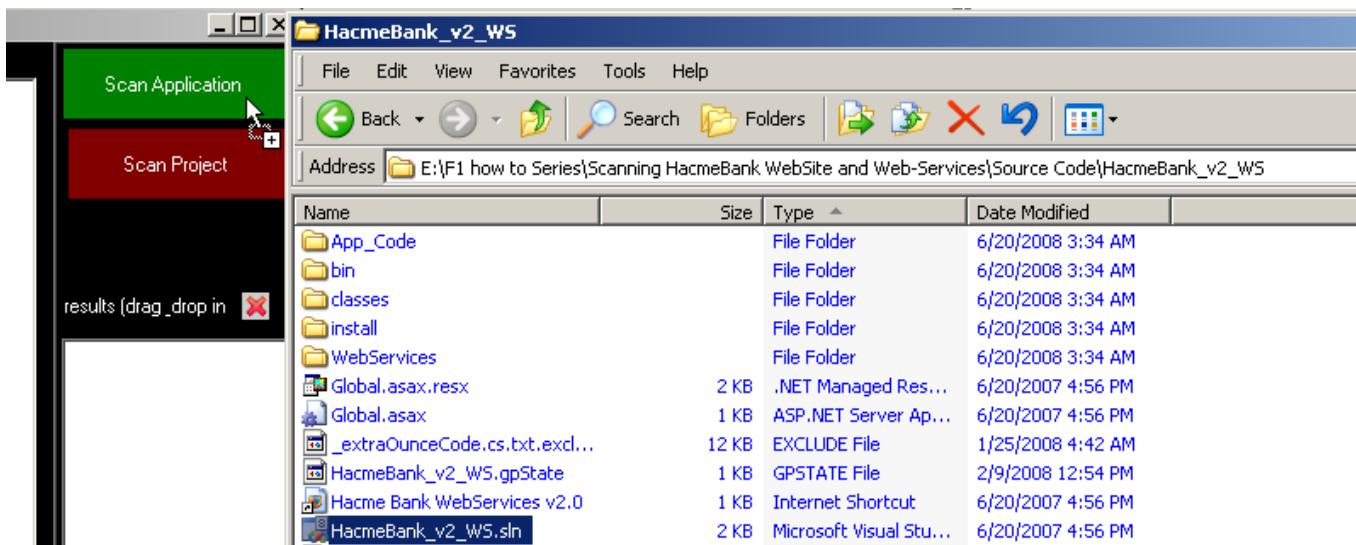
If the Scan Add-On is still open, close it



And reopen it:



Drag and drop the solution to scan it:



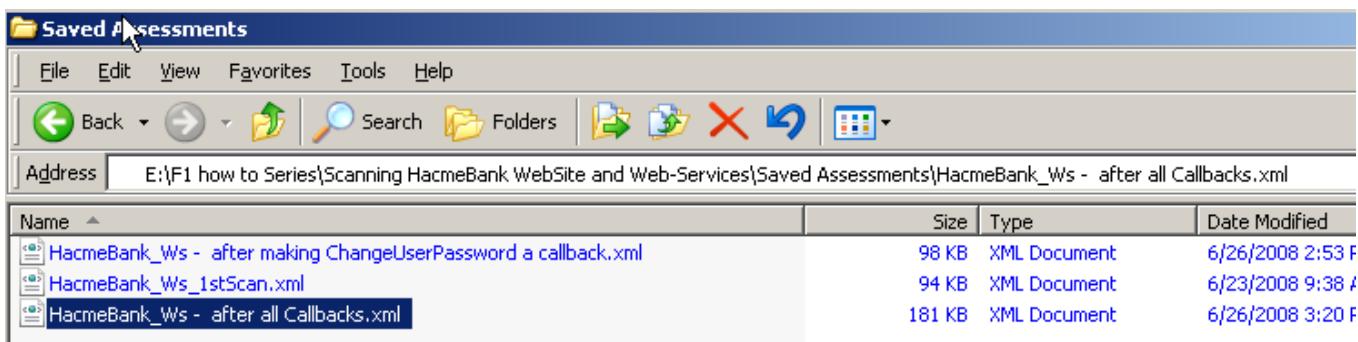
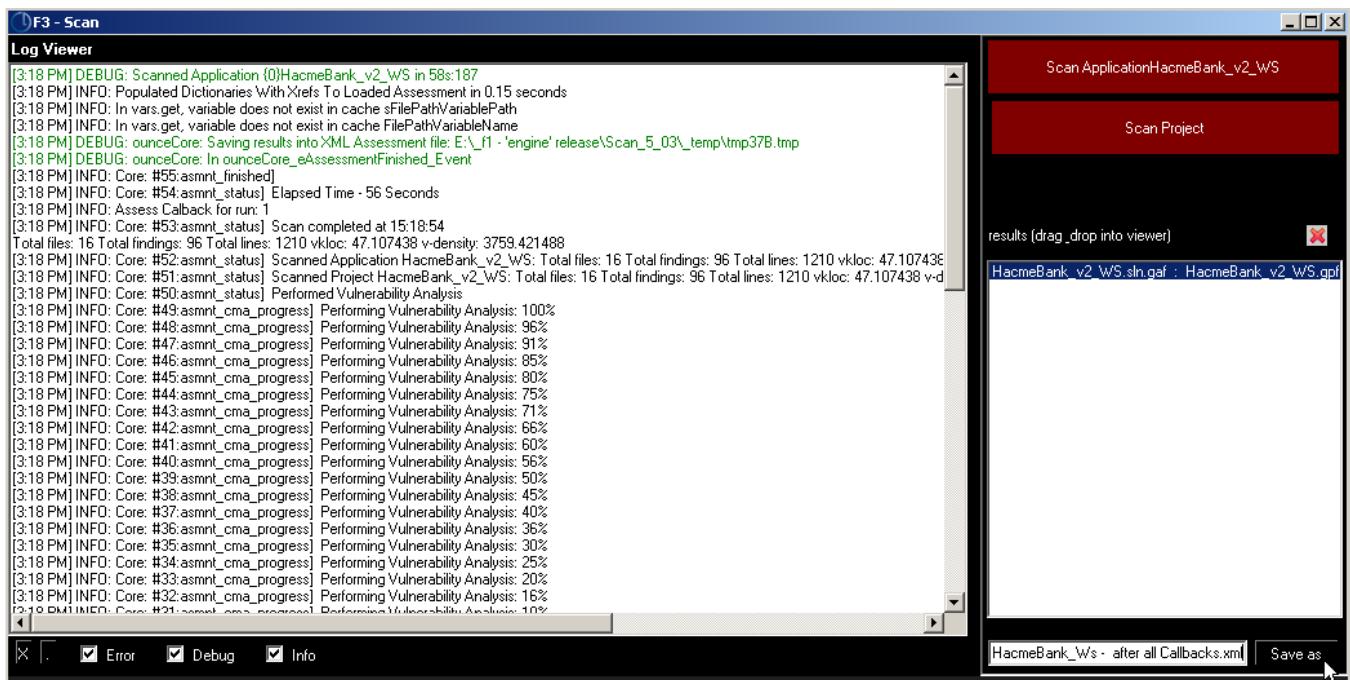
The Ounce Core will be restarted and a new scan will be triggered:

```

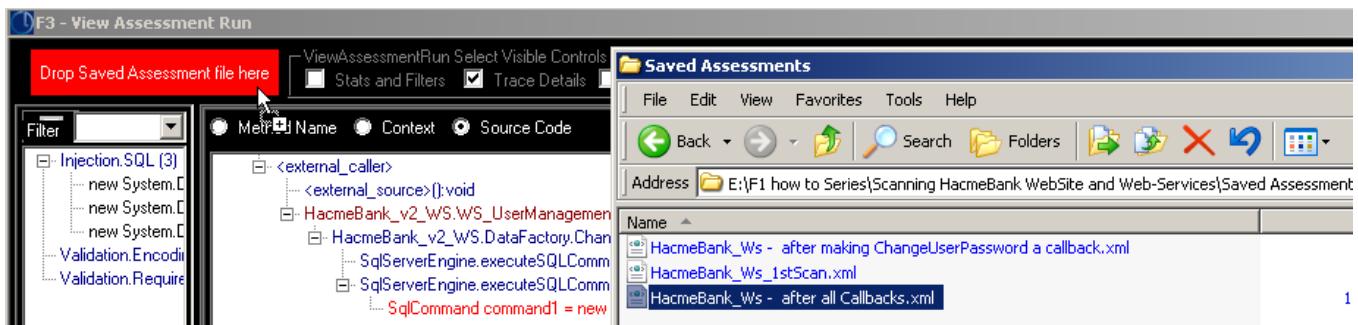
[3:18 PM] INFO: Core: #4:asmnt_status] Scanning C:\WINDOWS\TEMP
[3:18 PM] INFO: Core: #3:asmnt_status] Preparing project for scan...
[3:17 PM] INFO: Core: #2:asmnt_status] Scanning Project HacmeBank_v2_WS
[3:17 PM] INFO: Core: #1:asmnt_status] New Scan started at 15:17
[3:17 PM] INFO: Assesment Started, ID:0
[3:17 PM] INFO: Scanning Assessment: E:\F1 how to Series\Scanning HacmeBank WebSite and Web-Services\Source Code\HacmeBank_v2_WS
[3:17 PM] DEBUG: Adding Callbacks
[3:17 PM] DEBUG: All ok, user logged in
[3:17 PM] DEBUG: Configuring connection to Ounce Core and logger
[3:17 PM] DEBUG: Service Status: Running
[3:17 PM] DEBUG: Service is running, so trying to start it
[3:17 PM] DEBUG: Service Status: Stopped
[3:17 PM] DEBUG: Service is started, so trying to stop it
[3:17 PM] INFO: Restarting Service: Ounce Core
[3:17 PM] INFO: In vars.get, variable does not exist in cache fFactor
[3:17 PM] INFO: In vars.get, variable does not exist in cache raRun
[3:17 PM] DEBUG: Creating message with command Text: DndQueue
[3:17 PM] INFO: Testing logging: Info Message
[3:17 PM] DEBUG: Testing logging: Debug Message
[3:17 PM] ERROR: Testing logging: Error Message

```

Once the scan is completed, select the new item in the *results* list and save it:



Drag and drop the assessment in to the *View Assessment Run* Add-On:



And note that there are 24 SQL Injection traces:

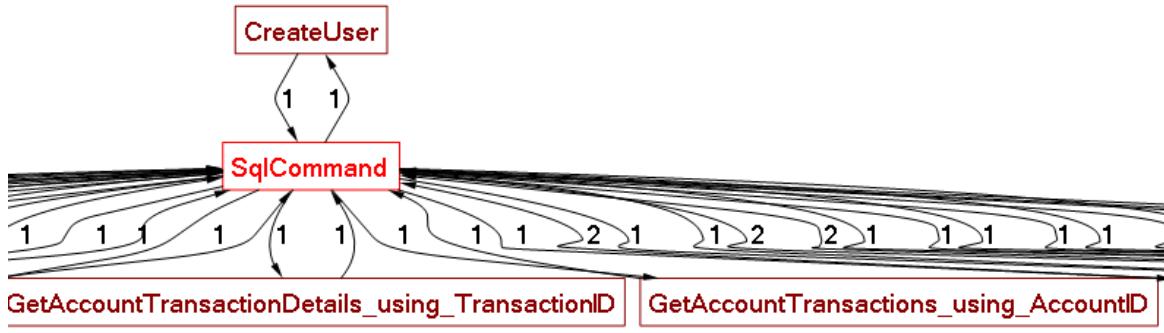
A screenshot of a software window titled "F3 - View Assessment Run". The main area contains a red button labeled "Drop Saved Assessment file here". Below it is a tree view of assessment results. The root node is "Injection.SQL (24)", which has two children: "Validation.EncodingRequired (1)" and "Validation.Required (5)". A toolbar at the top includes a plus sign icon and a "ViewAssessmentRun" button.

VIEWING ‘INSECURE PATTERN’ FOR SQL INJECTION

Here is an interesting view created using GLEE analysis.

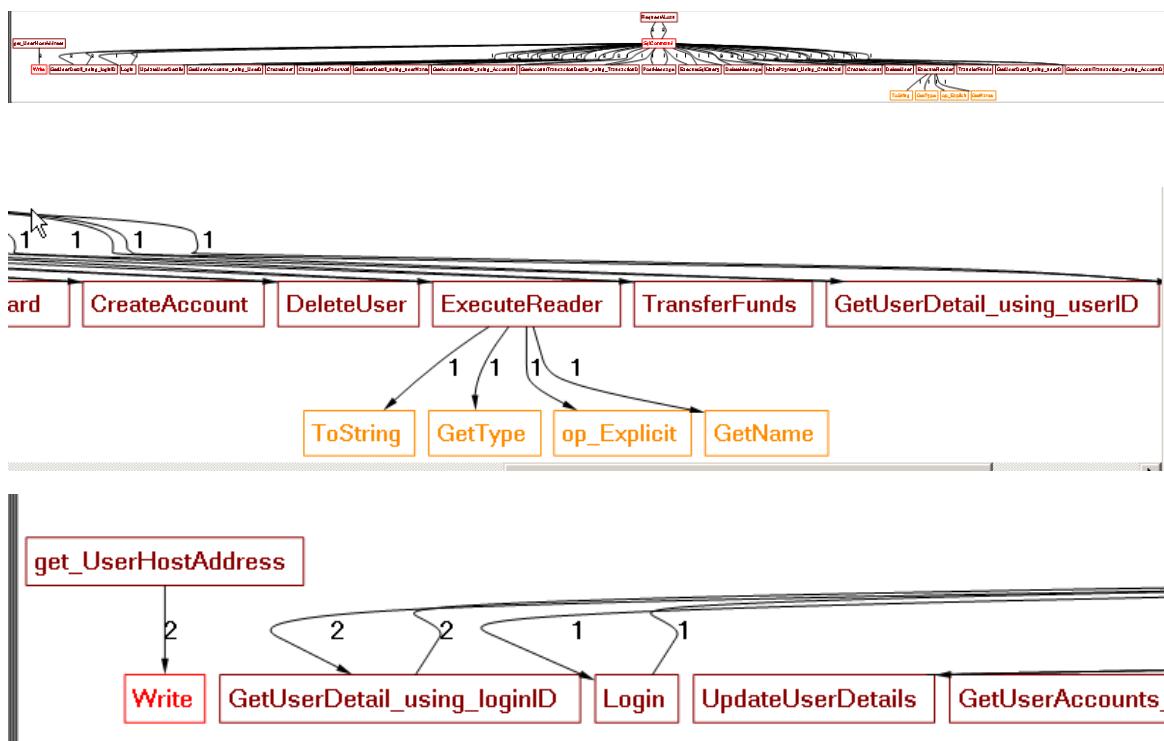


This shows that all SQL injection vulnerabilities are created by one sink



VIEWING ONLY SOURCE AND SINKS FOR ALL TRACES

This is a view with all traces (the SQL Injection and the Validation ones)



(NOT COMPLETED)