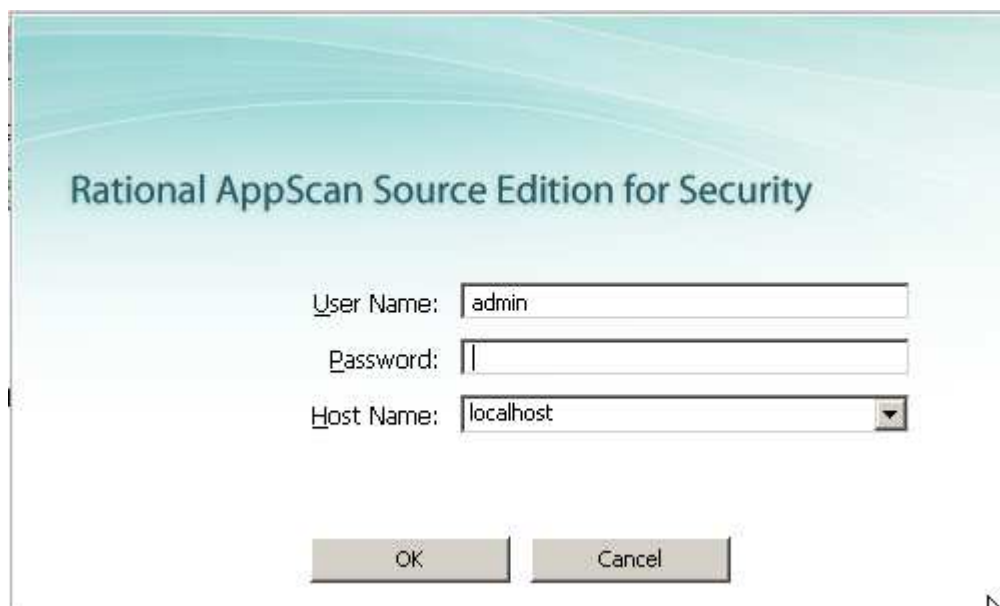


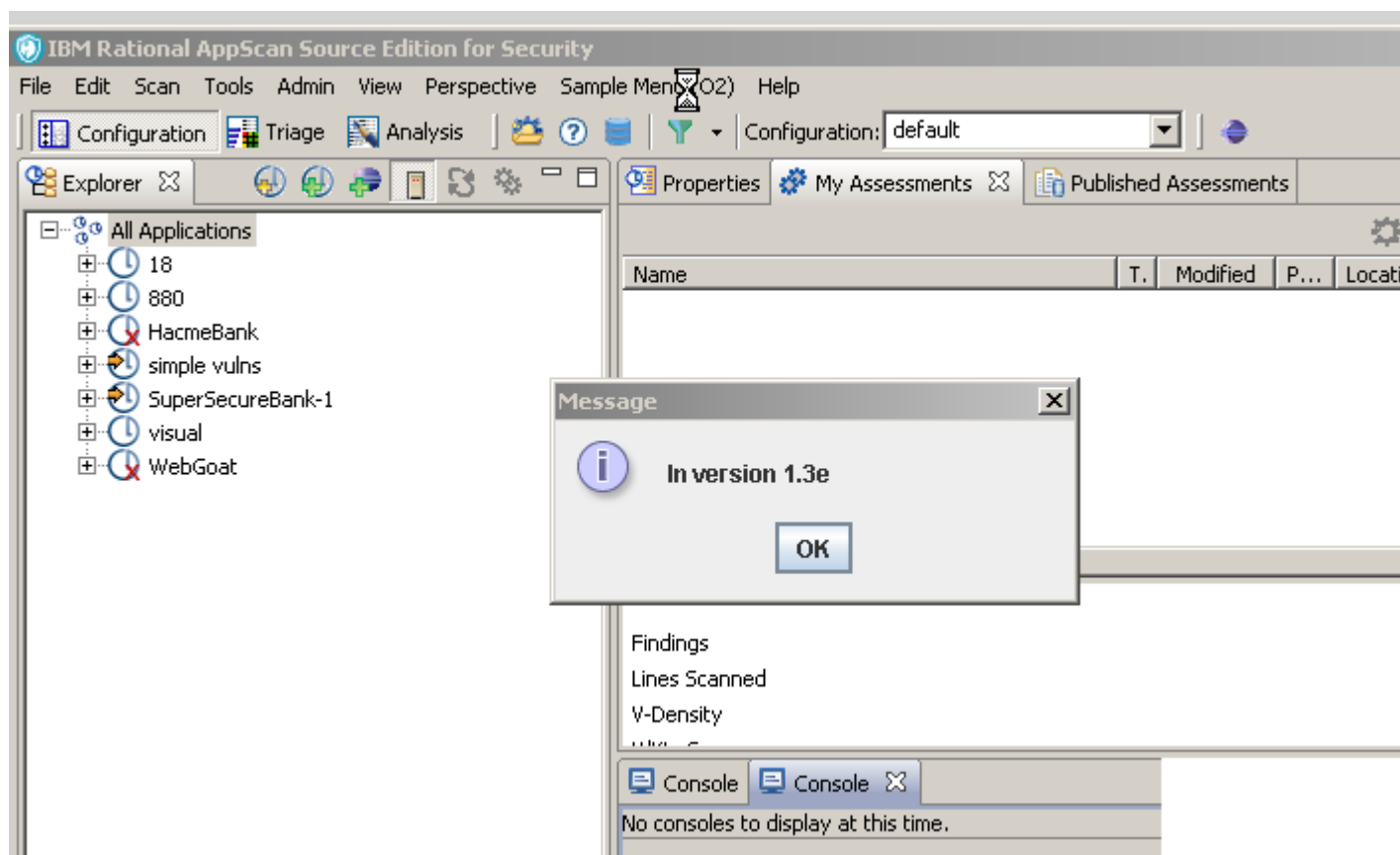
# Integrating AppScan with TeamMentor (PoC 1)

## PoC integrating TeamMentor with AppScan Source (Java App)

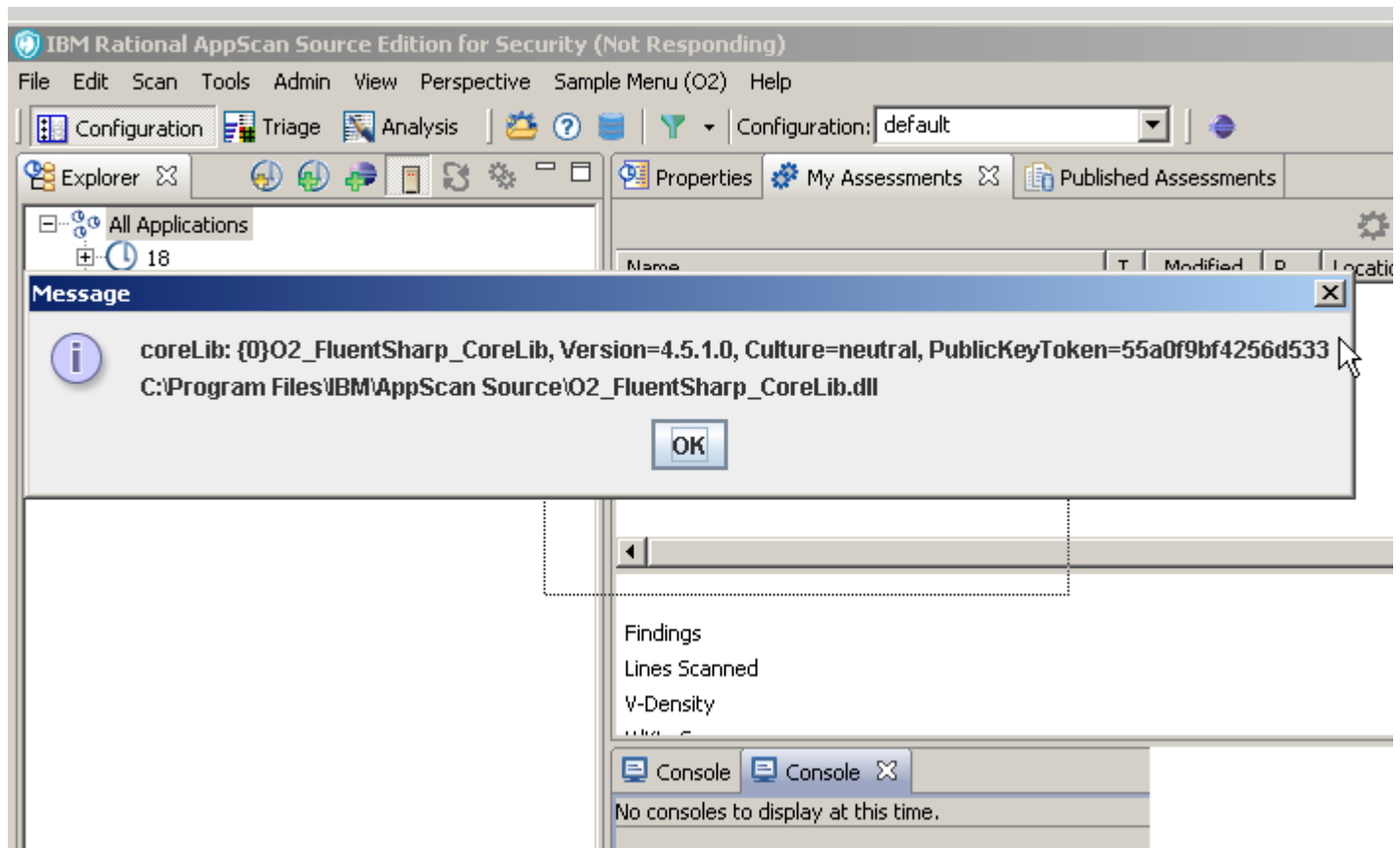
Start AppScan source (for the current version of this Poc , it needs to run under Admin privs since the O2 temp dirs are on the Program Files folder)



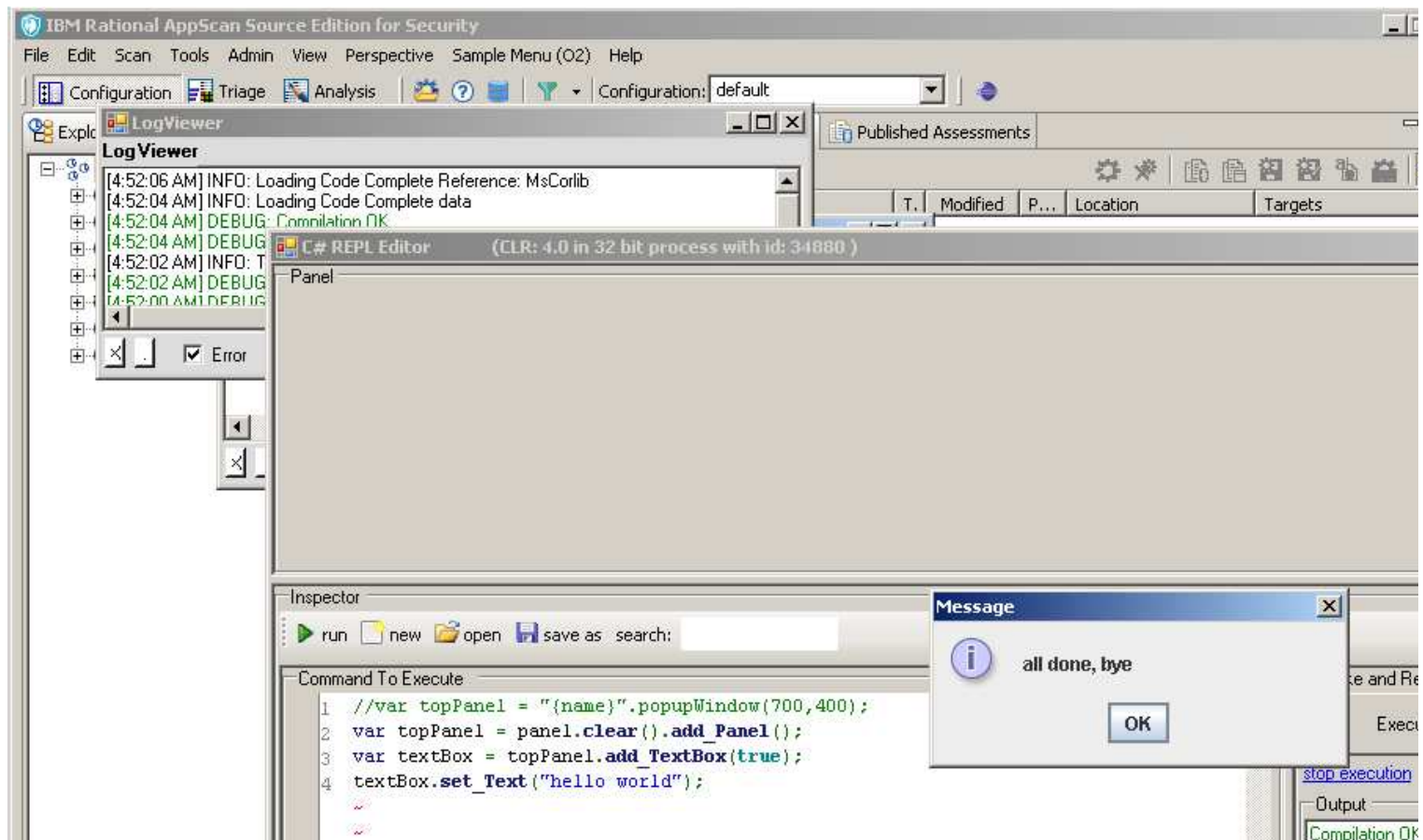
## Open O2 via the Test PlugIn extension



which will load up the O2 dlls



this will open an o2 Logviewer and an C# REPl Environment



In the C# REpl run the script below (which will sync TeamMentor's guidance with the selected vulnerability type (using win32 window's text))

```

//return "done";
var maxloop = 10000;
Dictionary<IntPtr, string> items = null;
var lastMatch = "";
var browser = "TeamMentor Guidance".popupWindow().add_WebBrowser();
"login".popupWindow().add_WebBrowser().open("http://teammmentor32.apphb.com/Login");
var tmServer = "http://teammmentor32.apphb.com/article/";
Action<string> showGuidance =
    (key)=> {
        if (key!=lastMatch)
        {
            "Showing guidance for: {0}".debug(key);
            browser.open(tmServer + key);
            lastMatch = key;
        }
    };
for(int i=0 ; i < maxloop ; i ++)
{
    items = new ExitApp_Test().test();

    Func<string, bool> hasItem =
        (itemToFind)=>{
            var targets = items.Where((item)=>item.Value==itemToFind);
            return targets.size() > 0;
        };

    if (hasItem("Injection.SQL"))
        showGuidance("Sql_Injection");
        //"FOUND SQLi".debug();
    else if (hasItem("Validation.Required"))
        showGuidance("All_Input_Is_Validated");
        //"FOUND validReq".debug();
    500.sleep();
}
return "done";

if(true)
{
    var valueToFind = "TeamMentor Guidance";
    // = "Conf value: "; //
    var valueToReplace = valueToFind + "****";
    var targets = items.Where((item)=>item.Value==valueToFind);
    "There are {0} Targets".error(targets.size());
    foreach(var target in targets)
    {
        NativeUtils.SetWindowText(target.Key, valueToReplace);
        "New value: {0}".info(NativeUtils.GetWindowText(target.Key));
    }
}
//return NativeUtils.GetWindowText(target.Key);

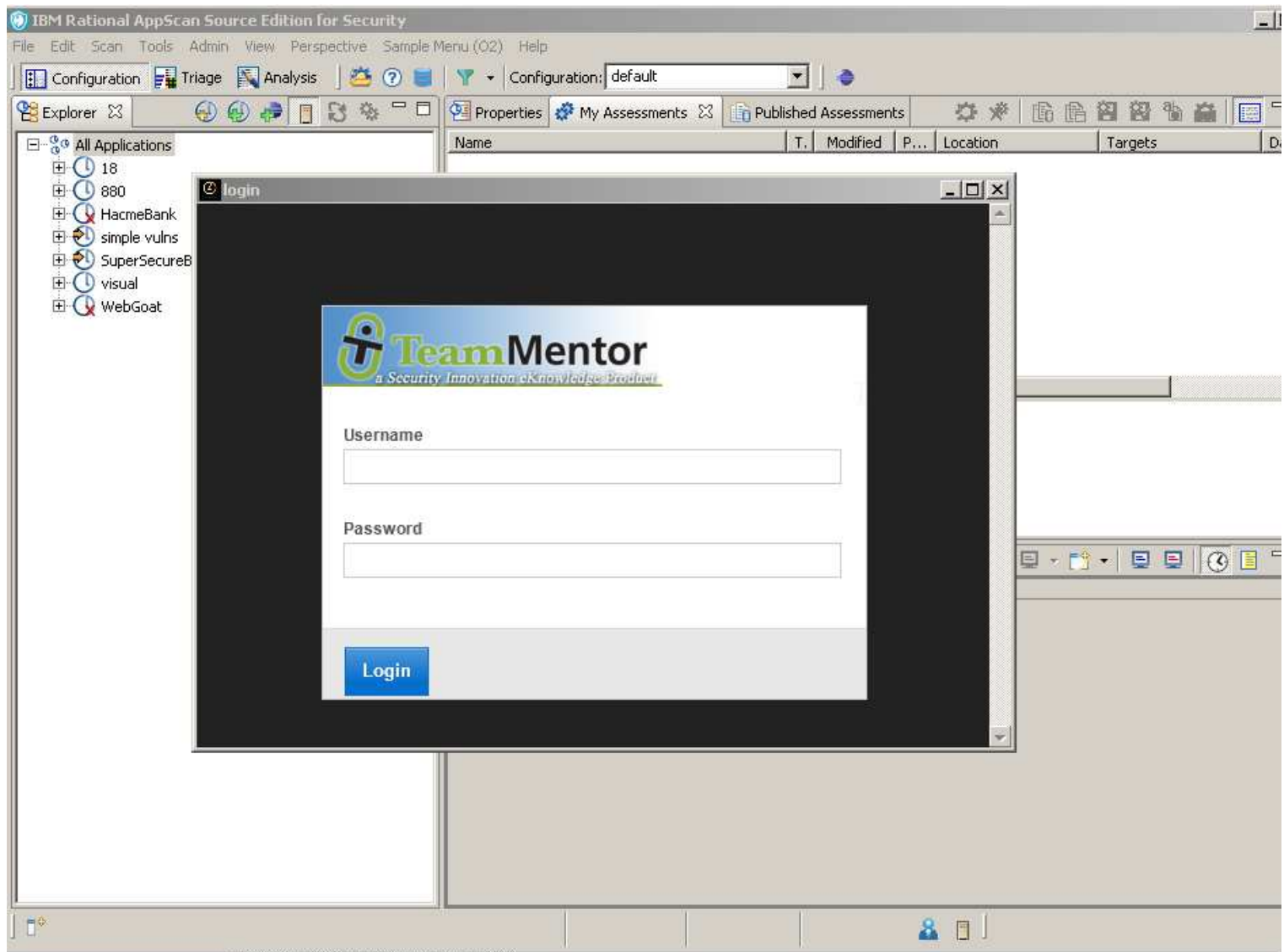
items.Values.showInfo();

return items.Values;

//O2File:C:\_WorkDir\O2\O2 Install\O2.Platform.Scripts\3rdParty\HawkEye\WindowFinder.cs
//O2File:C:\_WorkDir\O2\O2 Install\_TempDir_v4.5.1.0\_XRules_Local\Control_MessageListener.cs
//using System.Collections;

```

when started you will be asked to login



Scan a project or open an assessment:





Select an 'Validation Required' finding

IBM Rational AppScan Source Edition for Security

File Edit Scan Tools Admin View Perspective Sample Menu (02) Help

Configuration Triage Analysis Configuration: default

Findings (267)

AppDOS (1)  
Authentication.Crec  
Authentication.Crec  
Cryptography.Poor  
Info (139)  
Injection.DBConnec  
Injection.SQL (18)  
Validation.Required  
    High (2)  
    Low (8)


Trace	Seve...	Classification	API	Source
	High	Type II	System.Data.Common.DbCommand....	System.Data.Co
	High	Type II	System.Data.Common.DbCommand....	System.Data.Co
	Low	Vulnerability	System.Data.DataRow.set_Item	System.Data.Co
	Low	Vulnerability	System.Data.DataRow.set_Item	System.Data.Co
	Low	Vulnerability	System.Data.DataRow.set_Item	System.Data.Co
	Low	Vulnerability	System.Data.DataRow.set_Item	System.Data.Co
	Low	Type II	System.Data.DataRow.set_Item	System.Data.Co
	Low	Type II	System.Data.DataRow.set_Item	System.Data.Co
	Low	Type II	System.Data.DataRow.set_Item	System.Data.Co

Trace

SSBService.svc.cs

```
196         DataRow dr = dt.NewRow();
197         dr["accountID"] = reader.GetInt64(1);
198         dr["balance"] = string.Format("{0:C}", reader.Ge
199         dr["LevelName"] = reader.GetString(3);
200         dr["LevelDescription"] = reader.GetString(4);
201         dr["TypeName"] = reader.GetString(5);
202         dr["TypeDescription"] = reader.GetString(6);
203         dr["Status"] = reader.GetString(7);
204
205         totalValue += reader.GetInt64(2);
206
207         dt.Rows.Add(dr);
208     }
209
210     DataRow footer = dt.NewRow();
211     footer["accountID"] = "<strong>Total:</strong>";
```

TeamMentor Guidance

  
a Security Innovation eKnowledge Product

All Input Is Validated

Technology	Category	Phase	Type
ASP.NET 4.0	Input and Data Validation	Implementation	Checklist Item

Applies To

- ASP.NET 4.0

What to Check For

Ensure that all input parameters from form fields, query strings, cookies and HTTP headers are validated. Validation routines should check for length, range, format, and type. Validation should check first for known valid and safe data and then for malicious, dangerous data.

Why

An attacker passing malicious input can attempt SQL injection, cross-site scripting, and other injection attacks that aim to exploit your application vulnerabilities.

How To Check

To check for this problem, use the following steps:

- Find all sources of input.** During design time identify all of the potential sources of input parameters to your application. See the source code to discover sources of input that may have been missed in the design. Compile a list that you can use in the following steps. The most common sources are:

- Form Fields
- Query Strings