acunetix

Acunetix Web Vulnerability Scanner Vulnerability Scripting SDK

Quick Introduction

All the web vulnerability checks in Acunetix Web Vulnerability Scanner are scripts. Thanks to the introduction of scripting, the user is capable to define vulnerability tests that are more flexible and less prone to reporting false positives. Such change also allows us to make more comprehensive and complete web security checks.

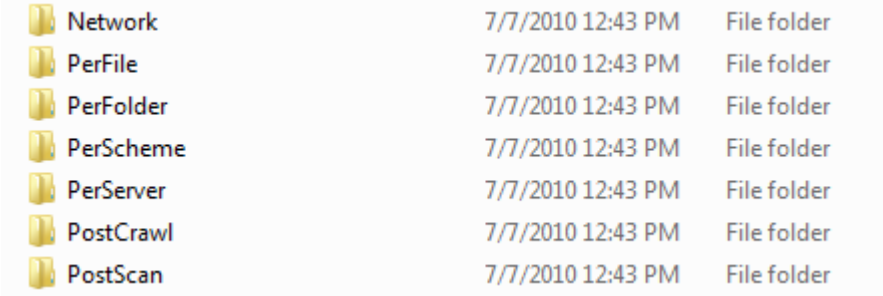
The scope of this document is to explain how these scripts (web vulnerability checks) are implemented in Acunetix Web Vulnerability Scanner and to help you write your own web security checks.

1. Script types

All the scripts are stored in the Data\Scripts sub directory in your Acunetix WVS Program Data installation directory C:\ProgramData\Acunetix WVS X\Data\Scripts.

Note: X in the directory path is the version number.

Acunetix WVS has 7 types of scripts. Each script type has its own sub directory, as displayed in the screen shot below.



Below is a description of the purpose of every sub directory. If you script your own vulnerability checks, it is important to store the new script under the appropriate directory.

1. Network directory - Network scripts
    - All the scripts placed in this directory are executed after the port scanning module is completed. As the name implies, these scripts are the only security checks related to network security. These scripts can verify if a certain TCP port is open and launch various security checks against the discovered network service. E.g. one Network script checks if the port 21 (FTP) is open and if anonymous FTP access is permitted.
2. PerFile directory Per File scripts
    - Scripts placed in this directory will be executed for each file discovered by the crawler. E.g. here you can place scripts that are checking for backup files or perform various text searches on file contents.
3. PerFolder directory - Per Folder scripts
    - These scripts are executed for each directory discovered by the crawler. E.g. in this folder you can place scripts that are checking if directory listing is permitted in a particular directory.
4. PerScheme directory - Per Scheme scripts
    - These scripts are executed on each input scheme. Input scheme are an abstraction of application inputs. E.g. a combination of GET and POST parameters is an input scheme. Acunetix WVS defines input schemes for each application input that needs to be tested (for HTTP Headers, for Cookies, for GET/POST parameters, for File Uploads (multipart/form-data)). In this folder you can find security checks for XSS, SQL injection and the other application input tests.
5. PerServer directory - Per Server scripts
    - The scripts in this directory are executed only once in the beginning of the scan. In this directory you can place scripts that are performing some kind of intelligence gathering checks, e.g. detecting the web server type.
6. PostCrawl directory - Post Crawl scripts
    - Scripts placed in this directory are executed after the crawling is completed. They could be executed multiple times if there are several scan instances. In this folder you can implement tests that are analyzing the crawl results looking for various patterns/problems. E.g. in this directory there are scripts that are trying to discover more files by using the Apache Content Negotiation technique, such as the one revealed by Stefano Di Paola.
7. PostScan directory - Post Scan scripts
    - In this directory scripts that are executed once after the scan is completed are stored. Such scripts are only executed once, regardless of how many scan instances there are. E.g. in this directory there are scripts that are testing for various Stored vulnerabilities (like Stored XSS, Stored SQL injection, Stored File Inclusion, Stored Directory Traversal, Stored Code Execution, Stored File Tampering, Stored PHP Code Execution).

2. Sample scripts

In this section, you can find a number of sample scripts to help you getting started. Acunetix WVS test scripts are written using the JavaScript language.

2.1 PerFile sample script

This script will be executed on each file discovered by the crawler and will replace the file extension with the '.bak'. Then it will make an HTTP request to verify if the backup file exists.

```
// Acunetix WVS SDK
// sample script to test for backup files (very basic)
// demonstration for 'per file' scripts

// scanURL is a TURL object containing the scan URL
// scanURL.url returns the url as string
var targetUrl = new TURL(scanURL.url);

// get the file that is currently tested
var file = getCurrentFile();

// extract filename (without extension)
// add the .bak extension to the filename
var fileName = getFileName(file.name) + ".bak";
// construct URI
var uri = file.path + plain2url(fileName);

// create HTTP Job
var http = new THTTPJob();
http.url = targetUrl;
http.verb = 'GET';

// set uri
http.URI = uri;

// execute request
http.execute();

// check for errors and if the files exists or not
if (!http.wasError && !http.notFound)
{
    // log the vulnerability
    logInfo('[TEST] Found backup file "' + uri  + '"');
    // show response body
```

```
    trace(http.response.body);
}
```

2.2 PerFolder sample script

The following script will be executed on each folder. It will search for a fixed directory listing pattern in the response body and display a log message if successful.

```
// Acunetix WVS SDK
// sample script to look for directory listings (very basic)
// basically this script will search in the response body for a fixed listing pattern
// demonstration for 'per folder' scripts

// get the directory that is currently tested
var dir = getCurrentDirectory();

// extract the response body
var dirBody = dir.response.body;

// search for directory listing pattern
if (dirBody.indexOf('<title>Index of /') != -1)
{
    // log the vulnerability
    logInfo('[TEST] Found directory listing on "' + dir.fullPath  + '"');
}
```

2.3 PerScheme sample script

This script will be executed on each input scheme and will check if the input scheme is vulnerable to XSS (cross-site scripting). This script has been overly simplified in order to make it easier to understand. Therefore it will detect only very simple XSS vulnerabilities and is prone to false positives. As a comparison, this script has only 50 lines of code from our actual XSS script, which has 440 lines of code :)

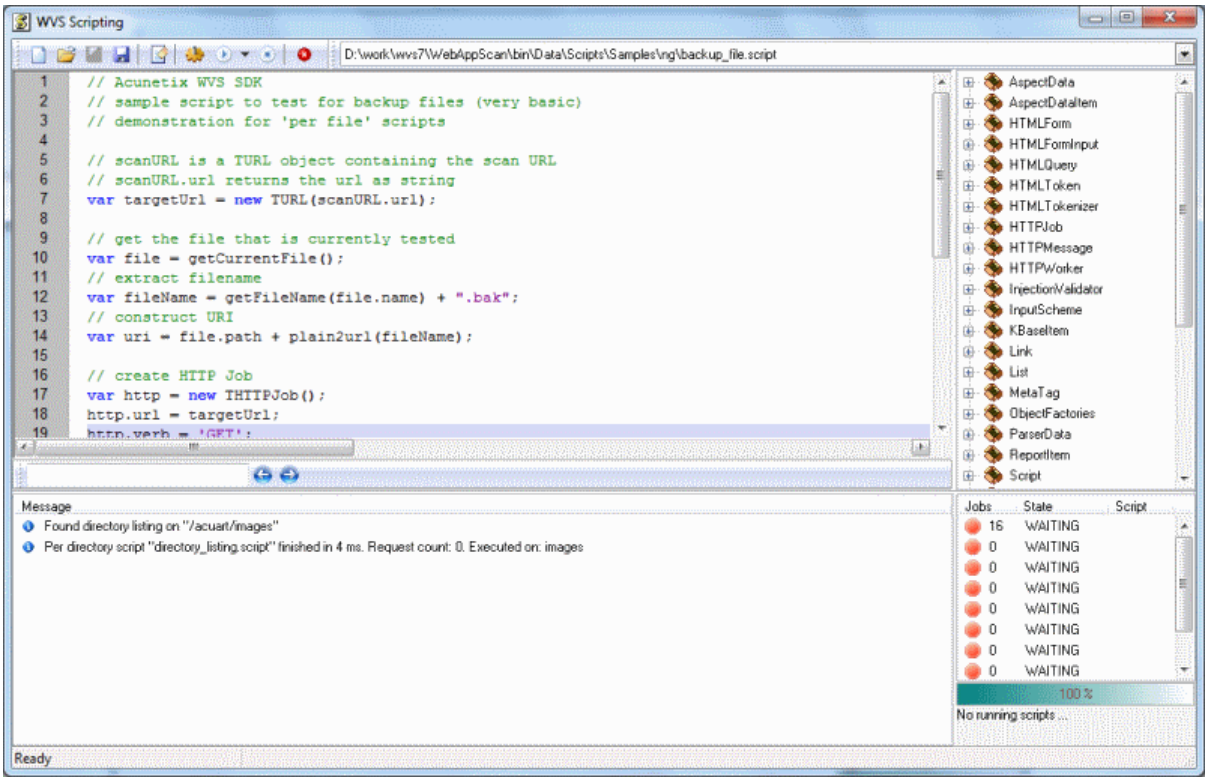
```
// Acunetix WVS SDK
// very basic XSS test to detect obvious XSS vulnerabilities (prone to false positives & false negatives)
// demonstration for 'per scheme' scripts

// scanURL is a TURL object containing the scan URL
// scanURL.url returns the url as string
var targetUrl = new TURL(scanURL.url);
```

```
// get current scheme
var scheme = getCurrentScheme();

// a scheme can have multiple inputs
for (var i=0;i<scheme.inputCount; i++)
{
    // each input can have multiple variations
    var variations = scheme.selectVariationsForInput(i);
    for (var j=0; j < variations.count; j++)
    {
                        // load variation
                        scheme.loadVariation(variations.item(j));
                        // set input value to our payload <XSS>
                        scheme.setInputValue(i, '<XSS>');

                        // create a HTTP Job (request)
                        var job = new THTTPJob();

                        // set the job URL to targetUrl
                        job.url = targetUrl;

                        // populate scheme in the newly created job
                        scheme.populateRequest(job);

                        // execute HTTP job
                        job.execute();

                        // check for error
                        if(!job.wasError)
                        {
                                // get response body
                                var responseBody = job.response.body;

                                // look for our payload
                                if(responseBody.indexOf("<XSS>")!=-1)
                                {
                                        // log vulnerability
                                        logInfo('[TEST]
Found XSS on input scheme "' + scheme.getInputName(j) + '" on url "' + scheme.path + '"');
                                }
                        }
```

```
    }
}
```

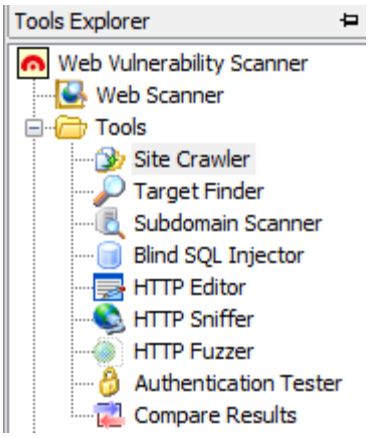
3. Testing scripts

Acunetix Web Vulnerability Scanner SDK includes a tool named "WVS Scripting". Such tool is very useful for writing and testing your own scripts. To use this tool, open the SDK directory and copy the file named WVSS.EXE in the Acunetix WVS installation directory. Then double click this file to start the application.

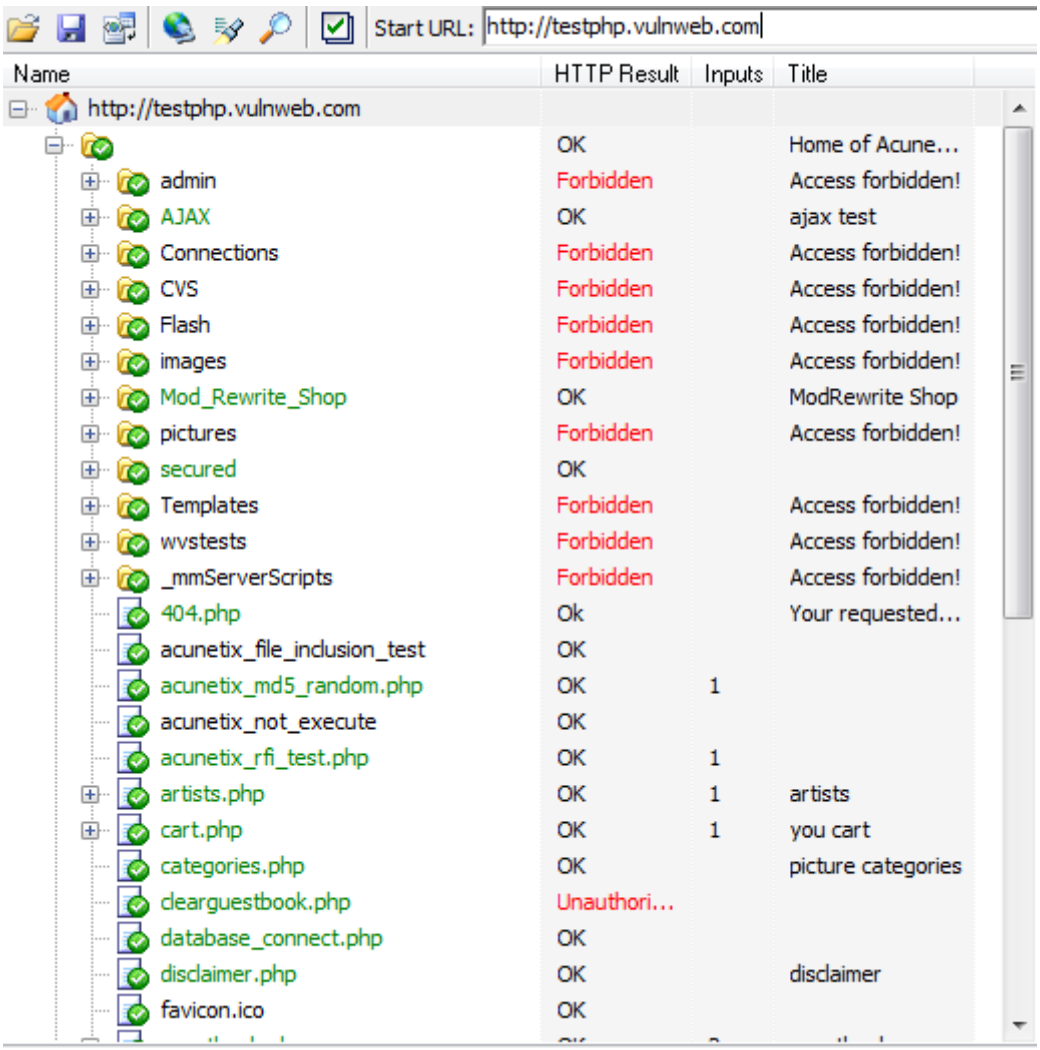


As an example, we will test the first script (the one that checks for backup files). Click the Open button in the WVS Scripting tool and select the *bak_file.script* file to be loaded into the editor. In order to test this script, you need to have some saved crawl results. The script will be executed against those results.

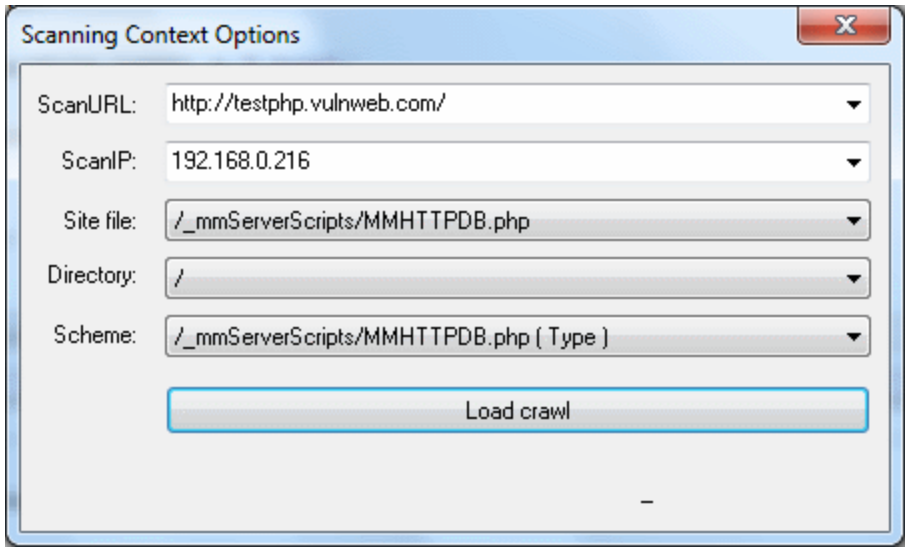
Start Acunetix WVS, choose the crawler tool and crawl our test website http://testphp.vulnweb.com.



After the crawling is completed, you should see the crawling results. It should look like image below.

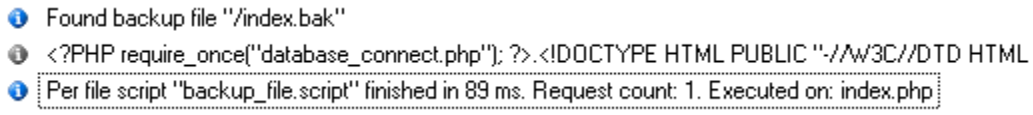


Save the crawl results into a file. This file will be used to test our script. Go back to the WVS Scripting tool and click on the 'Set Target' button. You should see the window bellow:



Click on the 'Load crawl' button to load the previously saved crawl results. Because you want to test a script that will be executed against a file, you need to select that file. Click on the 'Site file:' popup and choose '/index.php'. In this example, we will choose this file because we know that this file has a backup on testphp.vulnweb.com. You can close the 'Scanning Context Options window'.

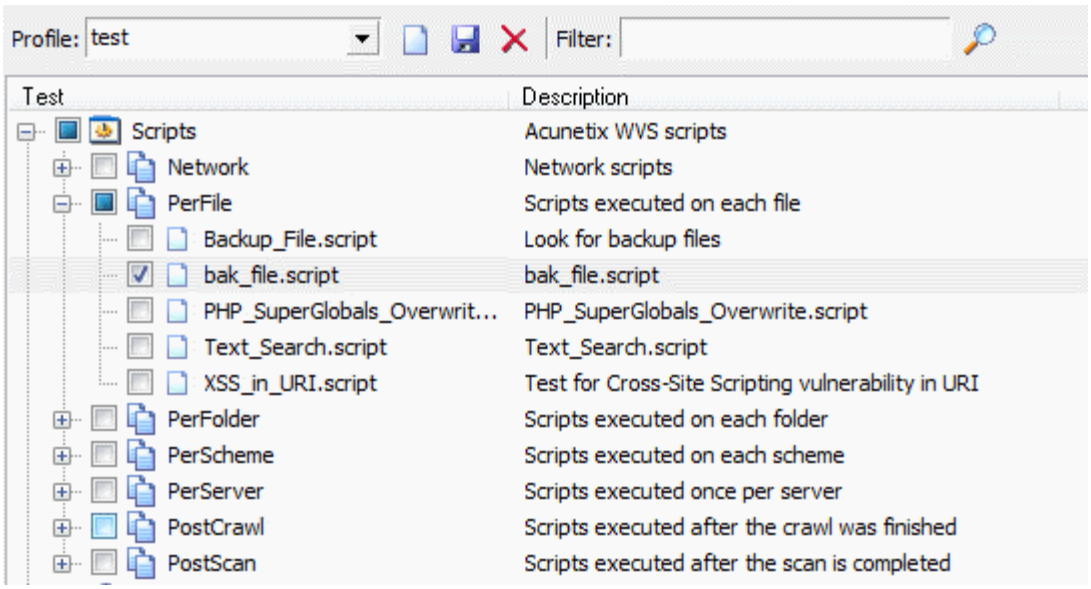
To start the script, click on the 'Run current script' popup button and choose 'Run per file'. After the script is executed you will see the results in the Message window. If the script was successful, you should see something like the below:



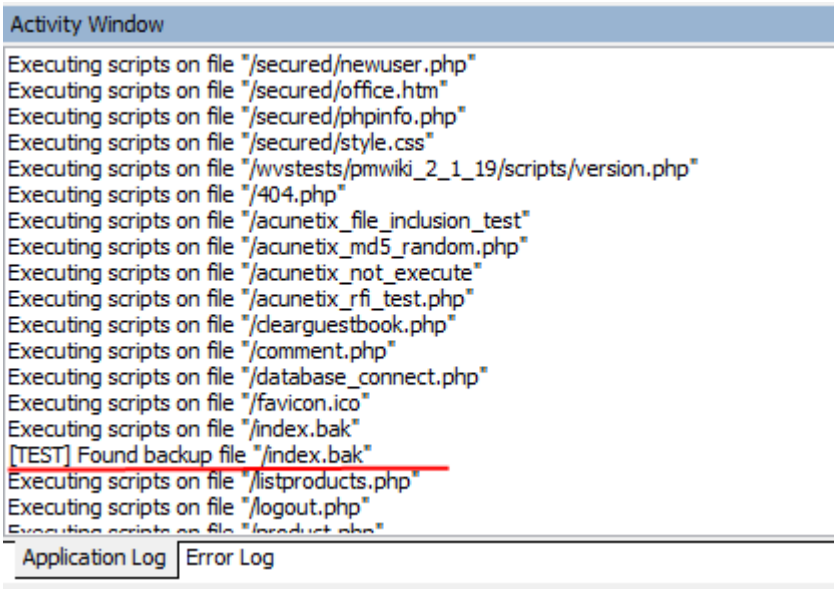
4. Using scripts with Acunetix WVS

After the scripts were tested and are working properly, you need to copy the scripts into the scripts directory in the Acunetix WVS installation directory. In the below example, we will continue to use the test script '*bak_file.script',* which should be copied to the PerFile directory. To be sure that the new script works properly, create a custom scanning profile that will include only the new test script.

Open Acunetix WVS, and navigate to Configuration->Scanning profiles from the Tools Explorer windows pane, and define a new test scanning profile and select the new test script. Once this procedure is ready, it should look something like:



Start a scan and use the newly created scanning profile. If the new script works well, you should see a log message for each backup file discovered by our script, as highlighted in the below screen shot.



NOTE: For the script to be executed in all the future scans, this script must be added to the scanning profile you will be using for future scans.

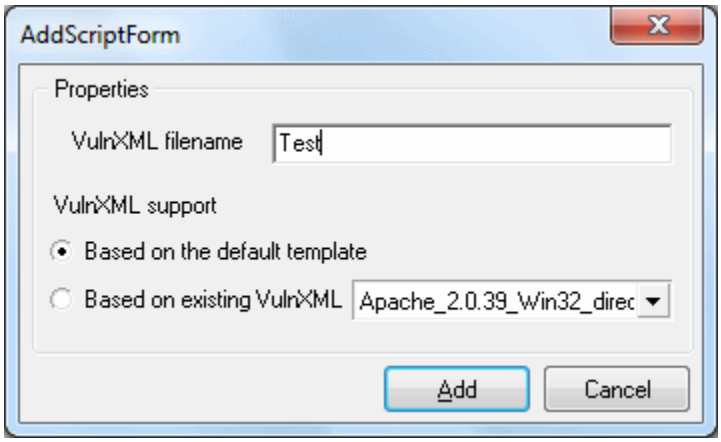
5. Generating a security alert in Web Alerts node

In the previous example the function logInfo was used to write information in the Activity Window, as can be seen in the below example;

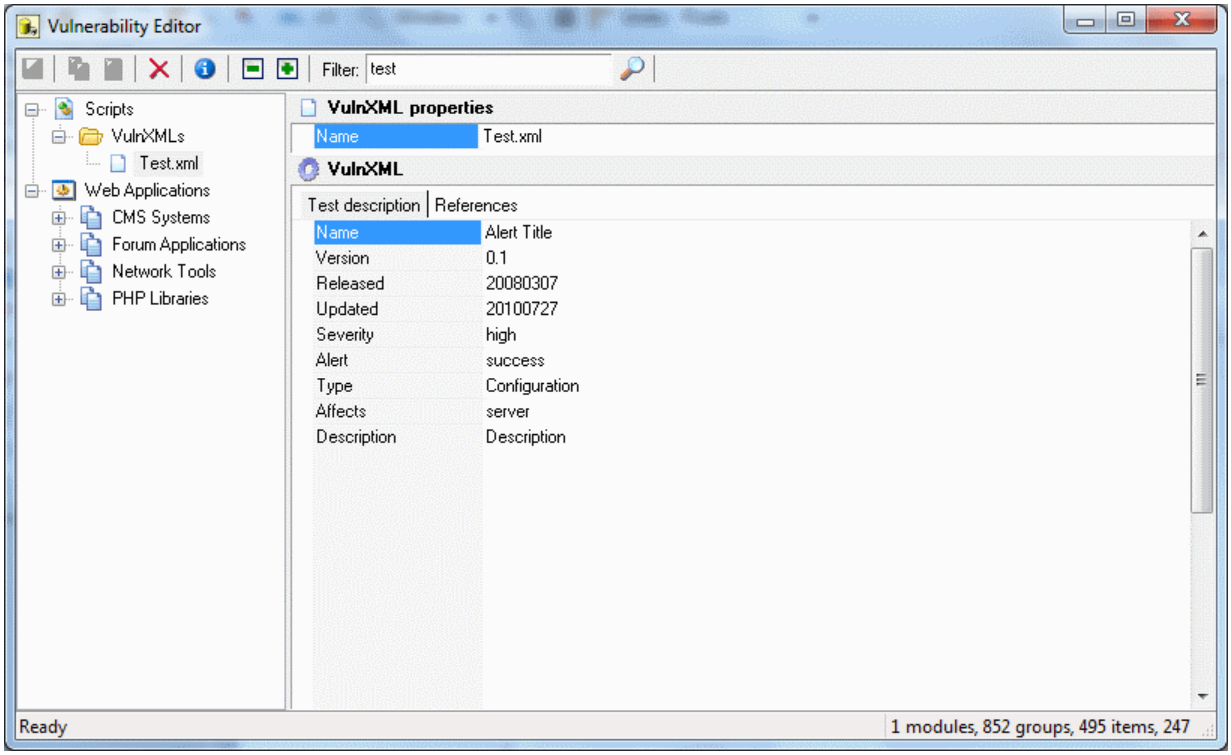
```
// log vulnerability
logInfo('[TEST]
Found XSS on input scheme '" + scheme.getInputName(j) + "' on url '" + scheme.path + "'");
```

At this stage you would probably want to generate a security alert that will appear in the "Web Alerts" node instead of a simple log entry in the activity window. To generate security alerts, first you need to create a VulnXML file and fill all the required information about the alert. Follow the below procedure to create a VulnXML file;

1. Start the Vulnerability Editor from the Acunetix WVS program group.
2. Right click the Scripts > VulnXMLs node and select 'Add Vulnerability'.



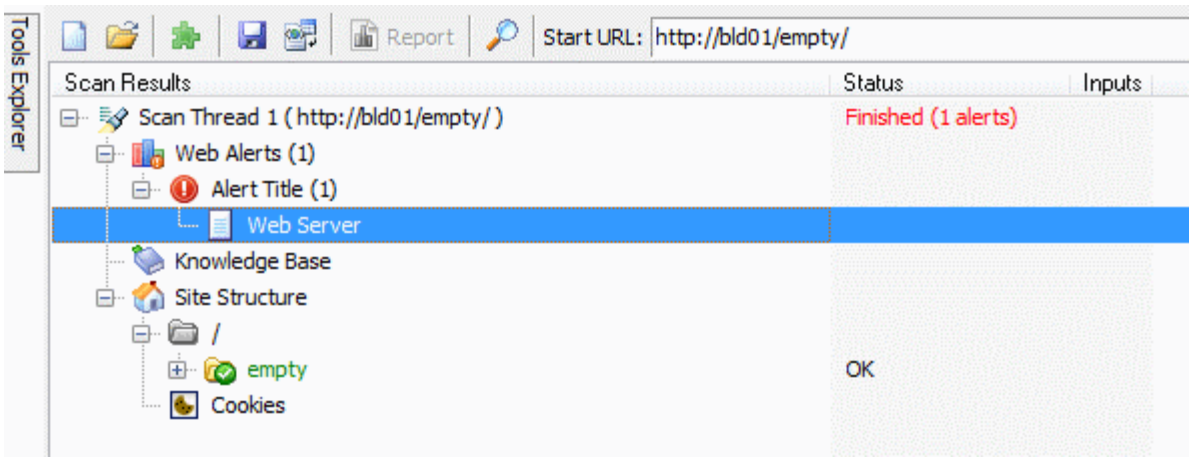
3. Enter the VulnXML filename and select if you should use a default empty template or an existing VulnXML as base file.
4. Once the VulnXML file is generated, enter the Vulnerability details, such as name, Severity, Description etc, as seen in the below screen shot.



5. Once the VulnXMLfile is created, you need to configure the script to issue an alert once conditions are met. Below is an example of such Syntax;

```
// how issue an alert
// create a report item object
var ri = new TReportItem();

// load our previously created VulnXML
ri.LoadFromFile('Test.xml');

// adjust details if necessary, like
ri.affects = "Web Server";
ri.request = lastJob.request.toString();
ri.response = lastJob.response.headersString;
ri.fullResponse = lastJob.response.body;

// add the alert
AddReportItem(ri);
```

6. If both Syntax and VulnXML file are correct, if all script conditions during a scan are met, you should be able to see an alert in the web alerts node, as can be seen in the below screen shot.


Tools Explorer
Report
Start URL: http://bld01/empty/
Scan Results
Status
Inputs
Scan Thread 1 (http://bld01/empty/)
Finished (1 alerts)
Web Alerts (1)
Alert Title (1)
Web Server
Knowledge Base
Site Structure
/
empty
OK
Cookies