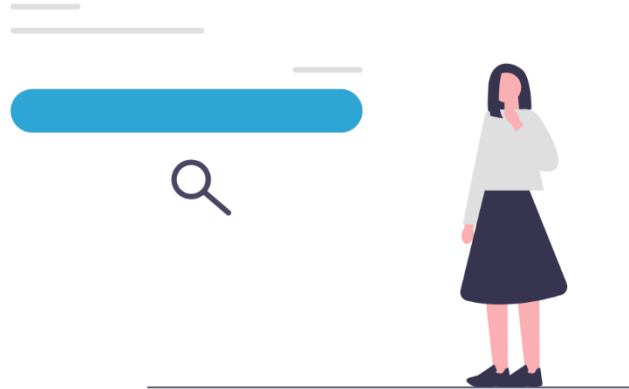


Enumerating a Web Application

Created by Colin McLean

Adapted by Jamie O'Hare

August 2023



Aim: This exercise aims to explore the fundamental principles of web application enumeration. Throughout the exercise you will be exposed to manual methods as well as sophisticated tooling to accomplish comprehensive enumeration of the target web application.

Note: The information contained in this document is for educational purposes only.

If any link does not work, please attempt to use the Wayback Machine (linked below) to view the content prior to its inaccessibility.



The Wayback Machine

<https://archive.org/web/>



Caution!

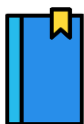
Use of these tools and techniques found within may breach the Computer Misuse Act, therefore, ensure you have permission to interact with the target machines.

Contents

1	Fantastic Tools and Where to Find Them	4
1.1	Lab Environment	4
1.1.1	Vulnerable Docker Containers	4
1.2	Windows Tools	5
1.3	Kali Tools	6
1.3.1	What if the tools are not there?	6
2	What is Enumeration?.....	7
2.1	Introduction To Enumeration	7
2.1.1	An Example of an Enumeration Methodology.....	7
2.1.2	Enumeration in a Penetration Test	8
3	Identifying Underlying Web Technologies.....	9
3.1	Identifying Technologies Manually	9
3.1.1	Passive Methods	9
3.1.2	Active Methods	18
3.2	Identifying Technologies Using Tooling	23
3.2.1	Passive Methods	23
3.2.2	Using WhatCMS.....	24
3.2.3	Using Wappalyzer.....	25
3.2.4	Active Methods	26
4	Navigating the Web like a Spider	30
4.1	Manual Spidering.	30
4.1.1	Using a Proxy	30
4.2	Automated Spidering.....	31
4.2.1	Using OWASP ZAP	31
4.2.2	Using Burpsuite	33
4.2.3	Using GoSpider	34
4.2.4	Using Other Tools.....	37
5	Headaches with HTTP Headers	38

5.1	Identifying HTTP Headers	39
5.1.1	Passive Methods	39
5.1.2	Active Methods	40
6	Identifying Endpoints.....	41
6.1	Identifying JavaScript Files and Links	41
6.2	Identifying HTTP Commands	42
7	Discovering Hidden Elements	44
7.1	Brute-forcing hidden folders and files	44
7.1.1	Using dirb	44
7.1.2	Using dirbuster	45
7.1.3	Using gobuster	46
7.1.4	Using dirsearch.....	47
8	Writing Vulnerability Reports	48
8.1	Scenario 1 - An Exposed Admin Panel	48
8.2	Scenario 2 - A Subdomain Takeover	49
8.3	Scenario 3 - Poor HTTP Header Implementation	49
9	Appendix	51
9.1	Appendix A - NMAP Scripts	51
9.2	Appendix B - Nikto Scanning.....	51

1 FANTASTIC TOOLS AND WHERE TO FIND THEM

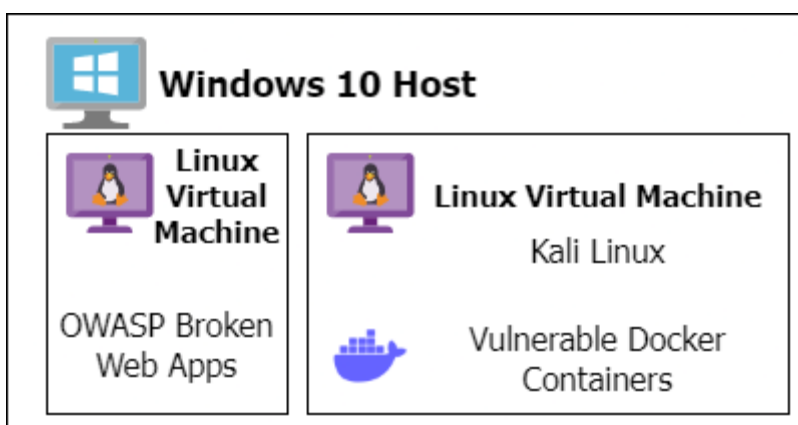


Please Note

This section is for reference.

1.1 LAB ENVIRONMENT

For this module's lab exercises, the network topology is as follows: A Windows 10 Host (192.168.1.254) operating 2 Linux Virtual Machines. One VM is running an altered version of the OWASP Broken Web Apps bundle (192.168.1.100), and another is a boutique installation of Kali Linux with several tools preinstalled, and various vulnerable docker containers (192.168.1.253). The figure below contains a visual representation of this topology. The following table presents the credentials of each system.



System	Username	Password
Windows 10 Host	admin	netlab
OWASP Broken Web Apps	root	owaspbwa
Kali Linux	kali	kali

1.1.1 Vulnerable Docker Containers

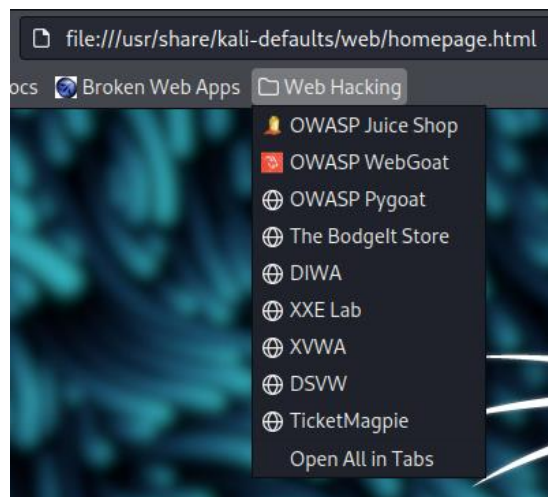
While some vulnerable applications exist on an isolated Virtual Machine, some are much closer to home! Through the power of containerisation (thanks to Docker!), there are multiple vulnerable web applications available on the Kali Linux Virtual Machine. You can find the corresponding docker-compose files in the `docker_targets` directory found on the Desktop, as shown in the figure below.

```
(kali㉿kali)-[~/Desktop]
$ tree
.
├── docker_targets
│   ├── ag_targets
│   │   └── docker-compose.yml
│   ├── api_hacking
│   │   └── docker-compose.yml
│   └── web_hacking
│       └── docker-compose.yml
```

To get these targets to run, use the command `docker-compose up`, when in the same directory as the `.yml` file, as seen below.

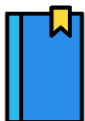
```
(kali㉿kali)-[~/Desktop/docker_targets/web_hacking]
$ sudo docker-compose up
[sudo] password for kali:
```

Afterwards, you can find links to these targets on Firefox's bookmarks bar.



1.2 WINDOWS TOOLS

While there are not many tools which we will run on the Windows 10 host, there may be the odd time we do. You can find these in the “tools” folder.

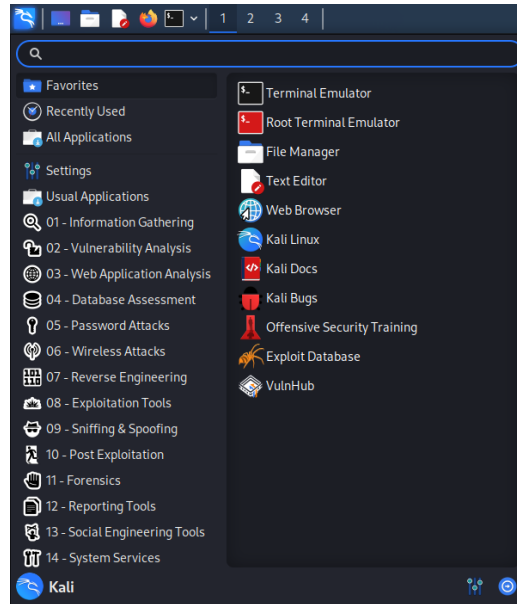


Take note!

You may come across references to a “tools” folder, this is in the University’s lab environment and accessible there via the URI `\\hannah\tools\pentest`

1.3 KALI TOOLS

In contrary to Windows, there will many tools found installed on Kali which we will make ample use of. One way to find these tools is through Kali's search bar found by clicking the top left Kali logo icon. Alternatively, tools can be found through the command line interface.



1.3.1 What if the tools are not there?

If you are looking for a tool and it's not there, just install it! These machines are sandboxes for you do as you wish. This may require you to get familiar with some of the different installer packages such as apt-get install or go get. However, do not threat as there is plenty of tutorials out there to learn from.

2 WHAT IS ENUMERATION?

2.1 INTRODUCTION TO ENUMERATION

After footprinting, if any, the next phase of the penetration testing process is the enumeration phase. Intense interrogation of the target system takes place in this phase, intending to explore and map the entire attack surface area comprehensively. Results of a thorough investigation of the target inform subsequent efforts can be best spent where there is a greater chance of return.

As you will see, there is a diverse range of enumeration methods, with many tools and techniques accomplishing similar goals. Yet, commonplace among most is the sheer volume of traffic these tools can create. In a typically 'active' phase, enumeration tools can sometimes encounter throttling or blocking from target systems in mature systems. In less mature systems, accidental denial of service can be common. Therefore, it may be beneficial to limit the speed at which some tools operate.

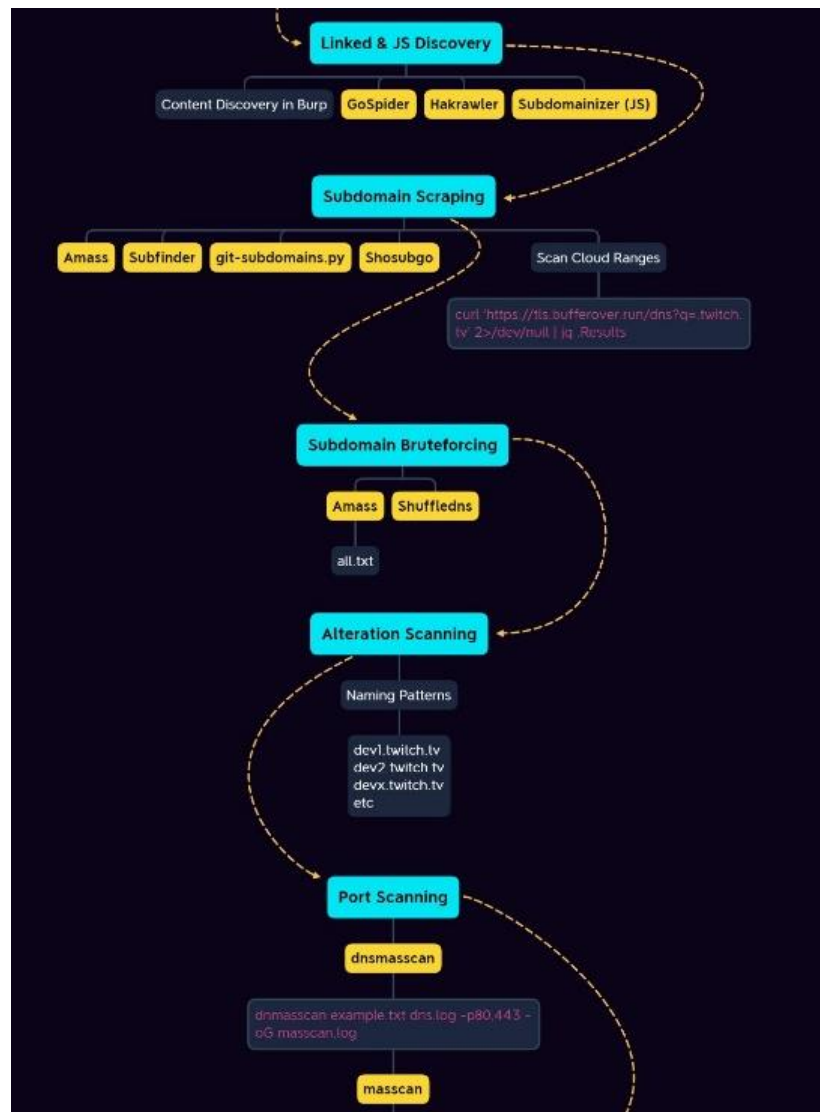
The results coming from a successful enumeration phase may include:

- A comprehensive site map.
- A list of all technologies used.
- Notes regarding any security measures.

2.1.1 An Example of an Enumeration Methodology

Jason Haddix's (@jhaddix) Bug Hunter Methodology is a popular procedure aimed at finding vulnerabilities for bug bounty hunters. Yet, despite this focus on bug bounties, the techniques and flow outline are applicable to other types of security assessment.

In a tweet by Stefan Rows (@ceos3c), part of the Bug Hunter Methodology was visualised. The figure below displays an excerpt of this methodology. While the Figure provides the focus, and the tools used in each step, it doesn't explain each step's goals.



2.1.2 Enumeration in a Penetration Test

"You can only test what you see."

This adage applies to enumeration in a web application security assessment. Failure to do a thorough job of the enumeration will likely lead to many undiscovered vulnerabilities. Having many possible undiscovered vulnerabilities is particularly problematic, especially when the hacker only needs to find and exploit one. As such, enumeration is often of paramount importance.

3 IDENTIFYING UNDERLYING WEB TECHNOLOGIES

When analysing a website, there are various technologies and frameworks at play that contribute to its functionality, design, and interactivity. By identifying these underlying web technologies, you can gain valuable insights into the website's structure, performance, and potential vulnerabilities. This process involves examining the website's component elements and using specialized tools to uncover details such as frameworks, content management systems (CMS), server configurations.

On a web app penetration test, it is highly likely you will encounter a target that makes use of specific well-known applications or frameworks (e.g., WordPress, phpBB, Mediawiki, etc). Knowing this underlying technology will aid the testing process significantly.

This information can be derived in many ways. Sometimes as simple as just reading the loaded webpage, while in more complex situations by sending the web server specific commands and analysing the output, as versions may respond differently.

3.1 IDENTIFYING TECHNOLOGIES MANUALLY

Performing identification manually allows for the tester to tailor their examination towards aspects or technologies of interest. However, it is important to note that while manual identification has its advantages, it can also be time-consuming, especially for large and complex targets.

3.1.1 Passive Methods

While passive methods typically involve gathering information without directly interacting with the target, it can also be construed as interacting with the target in a conventional manner. In web application penetration testing, this means manually navigating the website like a normal user.

3.1.1.1 Manual Browsing

On some webpages, you can just simply find the underlying technology used. Remarkably, this was a common practice for a long time. In fact, it is still common practice within some circles such as forums. With the rise of services offering easy website building and hosting, this information disclosure practice is having a renaissance. The following links showcase some examples of this practice.



A Blog “Powered by” a Common CMS.

blogs.glowscotland.org.uk/glowblogs/STEMcentralinmotion/



A Squarespace Designer’s website.

<https://www.kirstym.com/>



Jamie’s Personal Site.

<https://oha.re>

Some applications even have pages detailing the website’s technologies. A good example of this is linked below.



Dundee Science Centre’s About This Site page.

<https://www.dundeesciencecentre.org.uk/about-this-site>

If looking horizontally, you can Google dork using the tagline. The below link showcases that for the XenForo forum software.



Google Dork for XenForo Sites

["Community platform by XenForo" -site:xenforo.com - Google Search](https://www.google.com/search?q=Community+platform+by+XenForo+-site:xenforo.com)

Although potentially longwinded, many sites often credit who designed the website. One can follow this breadcrumb trail back to the website developer’s careers page to identify the technology expertise they are looking for.



The DCA’s Website disclosing the Web Design company commissioned

<https://www.dca.org.uk/>

It is important to remember that this type of information disclosure is not a necessary a finding to include in the penetration test report. It will depend on the client's appetite. That said, if a web site advertised that it was running out of date software, then there will be much bigger issues to report.

3.1.1.2 Examining Banners

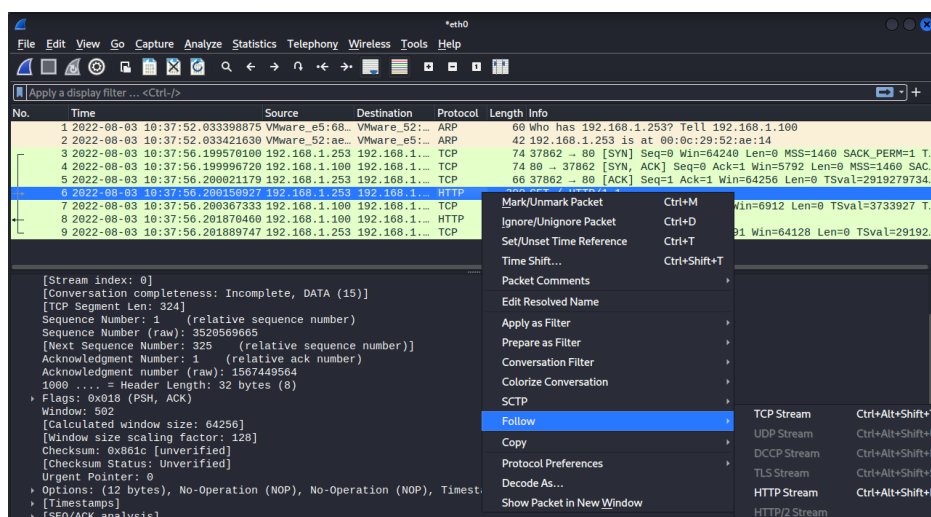
By examining the traffic generated through normal browsing, it is possible to determine technologies. This identification can be accomplished by investigating the HTTP header provided in the responses.



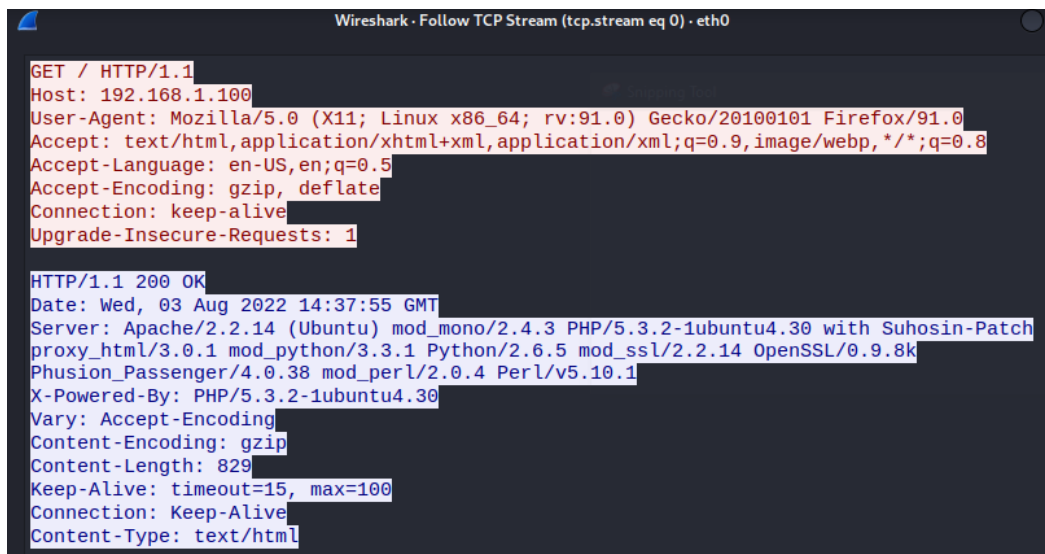
Caution!

Ensure you have powered on the Kali Linux and Web App Virtual Machines.

- Start by running Wireshark from Kali, then when prompted start a traffic capture on the eth1.
- Once Wireshark is capturing traffic, navigate to 192.168.1.100 via a web browser, and select the bWAPP application.
- Return to Wireshark, it should resemble something like the Figure below.



- Identify a HTTP packet with the destination of 192.168.1.100, on this packet right click, follow, then select TCP Steam. This will create a new window which should look like the Figure below.

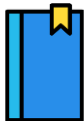


The image shows a Wireshark packet capture window titled "Wireshark - Follow TCP Stream (tcp.stream eq 0) - eth0". It displays an HTTP GET request in red text and the corresponding 200 OK response in blue text. The request includes headers for Host, User-Agent, Accept, Accept-Language, Accept-Encoding, Connection, and Upgrade-Insecure-Requests. The response includes headers for Date, Server, X-Powered-By, Vary, Content-Encoding, Content-Length, Keep-Alive, Connection, and Content-Type.

```
GET / HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Wed, 03 Aug 2022 14:37:55 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch
proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k
Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 829
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

The above figure shows the GET request (in red) generated by going to the <http://192.168.1.100> page. When the server serves the page requested back to the client, it sends the response (in blue). This response shows various HTTP headers including Server and X-Powered-By. These two headers are of the most interest to an Ethical Hacker. However, not all servers are configured to provide such verbose responses.



Take note!

Ensure you can access the Dockerized Vulnerable Applications by browsing to their address first. If accessible, then proceed with the tasks. If not, spend time debugging it.

- Restart capturing from Wireshark on Loopback: lo.
- Once Wireshark is capturing traffic, navigate to <http://127.0.0.1:3000>
- Return to Wireshark, follow the appropriate TCP stream, and it should resemble something like the Figure below.


```
Wireshark - Follow TCP Stream (tcp.stream eq 3) - Loopback: lo

GET / HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Wed, 03 Aug 2022 14:15:29 GMT
ETag: W/"7d2-182640fbd59"
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Wed, 03 Aug 2022 14:40:21 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
```

In comparison to the previous server response, there is no Server or X-Powered-By headers. This server has been configured to not publish this information; perhaps you'll have to use other methods to find that out!

Moving on, these headers can leak the technologies used, and the specific versions currently operating. One can use this information to identify known vulnerabilities by crosschecking the data with vulnerabilities databases.



The National Vulnerability Database.

<https://nvd.nist.gov/>

- On the NVD website, navigate to the CPE search, then when prompted enter Apache 2.2.14, as identified in the HTTP header sever response.

Search Results (Refine Search)

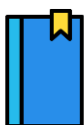
Search Parameters: Then

- Keyword: Apache 2.2.14
- CPE Status: FINAL
- CPE Naming Format: 2.3

Vendor	Product
cpe:2.3:a:apache:cassandra:2.2.14:*:*:*:*:* View CVEs	cassandra
cpe:2.3:a:apache:cloudstack:2.2.14:*:*:*:*:* View CVEs	cloudstack
cpe:2.3:a:apache:http_server:2.2.14:*:*:*:*:* View CVEs	http_server
cpe:2.3:a:apache:jspwiki:2.2.14:beta:*:*:*:* View CVEs	jspwiki

- In the search results, there will be 4 different results. The one we are looking for is the one containing http_server. Click the “View CVEs” link next to this result.
- The linked page contains all known vulnerabilities for that specific version of Apache, as seen in the Figure below.

Vuln ID	Summary	CVSS Severity
CVE-2022-31813	Apache HTTP Server 2.4.53 and earlier may not send the X-Forwarded-* headers to the origin server based on client side Connection header hop-by-hop mechanism. This may be used to bypass IP based authentication on the origin server/application. Published: June 09, 2022; 1:15:09 PM -0400	V3.1: 9.8 CRITICAL V2.0: 7.5 HIGH
CVE-2022-30556	Apache HTTP Server 2.4.53 and earlier may return lengths to applications calling r:wsread() that point past the end of the storage allocated for the buffer. Published: June 09, 2022; 1:15:09 PM -0400	V3.1: 7.5 HIGH V2.0: 5.0 MEDIUM



Take note!

The webserver will not be vulnerable to all these vulnerabilities, some require, some require misconfigurations or are present in optional features.

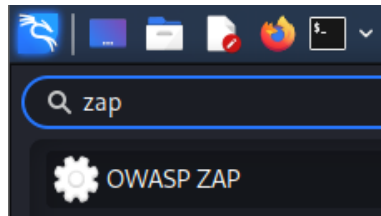
Using this information, one can enumerate through the vulnerabilities with proof of concepts or written exploits and try them against the target. However, that is outwith the scope of this phase of the penetration testing process.

3.1.1.3 Using a Proxy

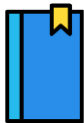
3.1.1.3.1 Using OWASP ZAP

Through usage of OWASP ZAP as a proxy, one can collect the response banner much easier than using Wireshark.

- In Kali Linux, run the OWASP ZAP proxy, this can be accomplished by searching for it in the menus, as seen in the Figure below.

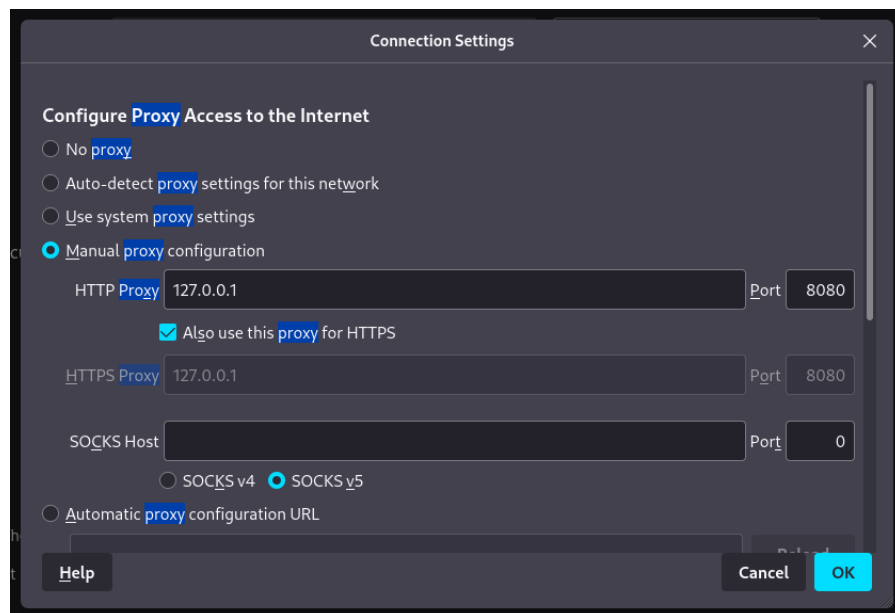


- Then configure the Firefox web browser to use the ZAP proxy, as seen in the Figure below.

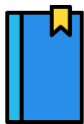
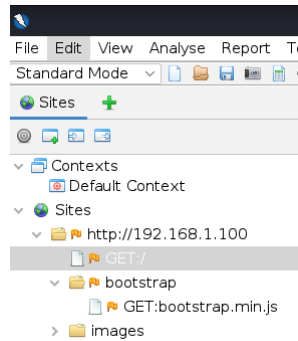


Take note!

Ensure the “Also use this proxy for HTTPS” is enabled!



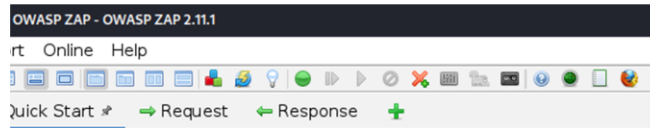
- From Firefox, browse to any vulnerable application. Once you have, OWASP ZAP should start to populate the Sites tab, like Figure below.



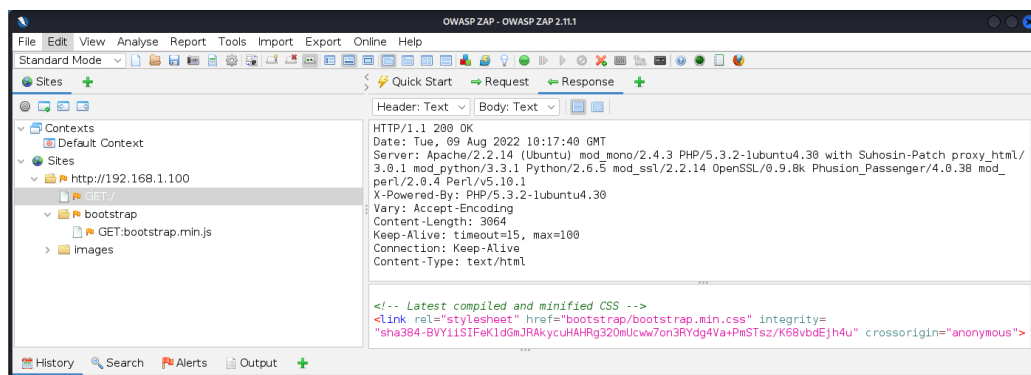
Take note!

If the population does not take place, it is likely due to your proxy configuration, give it another look to check over. If you are sure the configuration is correct, then it might be the pesky “ZAP HUD” feature.

Ensure the green icon, 3rd from the right on the taskbar at the top of the window, is inactive. It should look something like the picture below.



- Select a request found within the Sites tab, then open the Response tab, and examine the HTTP response for that associated request.



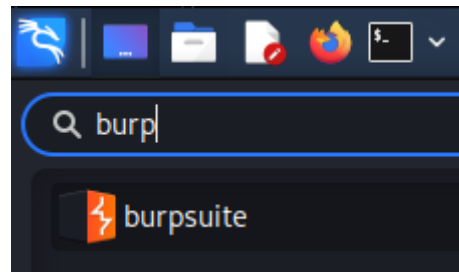
Caution!

Make sure to disable the proxy settings, or else you may complicate further exercises.

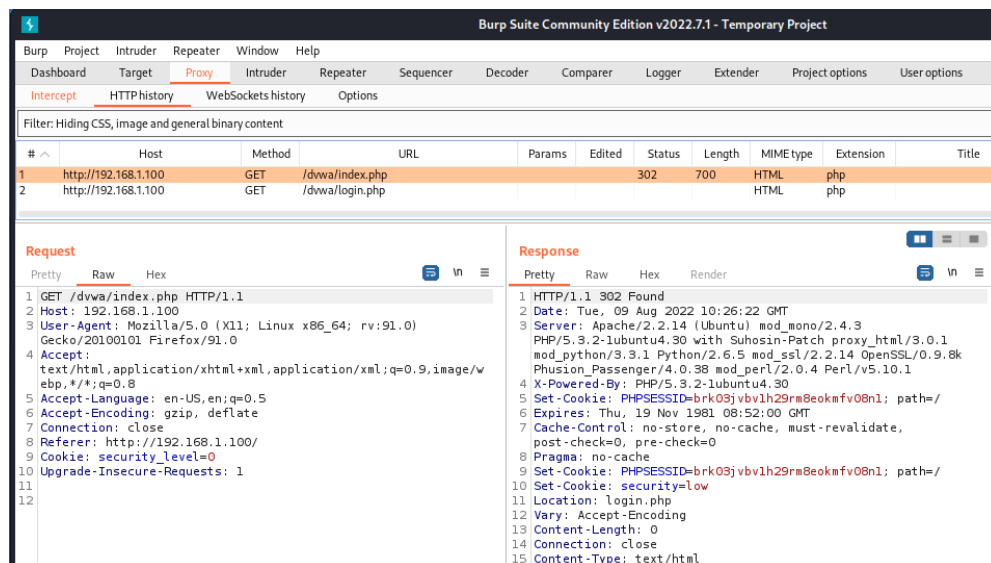
3.1.1.3.2 Using Burpsuite

Like OWASP ZAP, through the usage of Burpsuite as a proxy, one can easily gather response banners.

- In Kali Linux, run Burpsuite, this can be accomplished by either searching for it in the menus as seen in Figure below, or by entering the command Burpsuite.



- Next step is to configure the Firefox web browser to enable use of the proxy (as seen previously in the OWASP ZAP exercise above).
- Then from Firefox, browse to any vulnerable application. Once you have, navigate to the Proxy tab, then the associated HTTP history sub tab. From this view, select a HTTP request in the top pane, and find the response in the lower right pane. As seen in the Figure below.
- Examine this HTTP response and identify the headers which leak the underlying technologies.



Caution!

Make sure to disable the proxy settings, or else you may complicate further exercises.

3.1.2 Active Methods

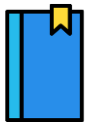
The previously explored passive methods were simply additions to the typical user interaction on the target website. The forthcoming active methods will achieve the same results, however, the methods employed are not typical for a normal user. Additionally, these methods can be implemented into automation scripts.

3.1.2.1 Using Netcat

Netcat allows us to make raw connections and send simple commands. For HTTP, this means we can send commands like GET and obtain responses.

3.1.2.1.1 GET Method

The HTTP GET command is the most used http method, which simply retrieves a web page from a web server.



Take note!

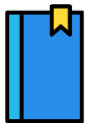
Ensure you can access the Vulnerable Applications by browsing to their address first. If accessible, then proceed with the tasks. If not, spend time debugging it.

- First, browse to 192.168.1.100, and select the bWAPP application, then return to the Kali terminal, and enter the following. You will not get a response until the end. Afterwards, you must press enter twice.

```
nc 192.168.1.100 80
```

```
GET /bWAPP/login.php HTTP/1.1
```

```
host: 192.168.1.100
```



Take note!

You will want to copy and paste the above commands, as the connection created by netcat can be closed before you write them manually.

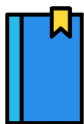
- Scroll up to examine the header, it should resemble something like the Figure below.

```
(kali@kali)-[~/Desktop/github/wappy]
$ nc 192.168.1.100 80
GET /bWAPP/login.php HTTP/1.1
host: 192.168.1.100

HTTP/1.1 200 OK
Date: Wed, 03 Aug 2022 14:44:36 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch pr
oxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passe
nger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Set-Cookie: PHPSESSID=7sbe6napo3o2egbfulh5pi4h67; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4019
Content-Type: text/html

<!DOCTYPE html>
<html>

<head>
```



Take note!

Look at the information provided by the Response header. We have discovered the Web Server version and other information about the server setup.

3.1.2.1.2 HEAD Method

You will notice the last command required you to scroll through the HTML contained in the response. You can avoid this lengthy response by using the HEAD command. This can be useful when we are automating information gathering against multiple targets.

- Enter the following command, and input into a terminal window on Kali.

```
nc 192.168.1.100 80
```

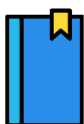
```
HEAD /bWAPP/login.php HTTP/1.1
```

```
host: 192.168.1.100
```

- Then you must press enter twice. The output should look something like the Figure below.

```
(kali@kali)-[~/Desktop/github/wappy]
$ nc 192.168.1.100 80
HEAD /bwAPP/login.php HTTP/1.1
host: 192.168.1.100

HTTP/1.1 200 OK
Date: Wed, 03 Aug 2022 14:46:07 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Set-Cookie: PHPSESSID=hd3ifpi2125cb2oootm3lj21n3; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Type: text/html
```



Take note!

The HEAD command can be disabled by web servers, this may add to inaccurate results.

Other tools exist which can perform the same functionality, some with an even greater feature set.



Socat (Socket CAT)

<https://linux.die.net/man/1/socat>



OpenSSL

<https://www.openssl.org/>

3.1.2.2 Examining Robots.txt

Web spiders (such as Google) retrieve a web page and then recursively traverse hyperlinks to retrieve further web content. Their accepted behaviour is specified by the “Robots Exclusion Protocol” of the robots.txt file in the web root directory. The file can be accessed by simply specifying it in the URL. E.g., www.xxxxxx.com/robots.txt. An example of a robots.txt is below: -

```
User-agent: *
Disallow: /search
Disallow: /groups
Disallow: /images
```

In some circumstances, a particular folder present in the robots.txt and thus the web application, may indicate the technology used. However, more broadly, the robots.txt file can help hackers identify folders of particular interest.


The user-agent entry can either be used to target specific spiders but would normally be * (i.e., all spiders). The disallow directive informs spiders not to examine specific folders. Why would a system administrator use the disallow directive? Probably to avoid Google from indexing it. This peculiar behaviour may lead to directing hackers towards interesting paths and pages.

- Pick any web site that you are familiar with and examine the robots.txt file.



Discord's robots.txt.

- <https://discordapp.com/robots.txt>



Question.

Investigate the robots.txt file found on the VM web apps.

- <http://192.168.1.100/robots.txt>
- <http://192.168.1.100/dvwa/robots.txt>
- <http://192.168.1.100/bWAPP/robots.txt>

3.1.2.3 Using Curl

Curl is a command line tool for doing all sorts of URL manipulations and transfers. It can be very useful when scripting. In this exercise, we will use Curl to retrieve the robots.txt file.

- From a Kali Linux issue the command:

```
curl -O http://192.168.1.100/robots.txt
```

- While printing out on the command line may be good, when the file is small, what if it quite long, and we only want a specific part of it. Well, that's when we can pipe the output to grep.
- Try the following command to see how you could filter the output.

```
curl -s http://192.168.1.100/bWAPP/robots.txt | grep "Disallow"
```

- The output should resemble the Figure below.

```
(kali@kali)-[~]
$ curl -s http://192.168.1.100/bWAPP/robots.txt | grep "Disallow"
Disallow:
Disallow: /
Disallow: /admin/
Disallow: /documents/
Disallow: /images/
Disallow: /passwords/
```

While this is a good method for closely examining one file, what if we wanted to download multiple files for larger analysis? That's where we can use Wget.

3.1.2.4 Using Wget

Wget is a useful command tool for retrieving files from HTTP, HTTPS and FTP. It has a lot of functionality that can be used by web testers. It is available under different OS platforms. To showcase how it works, we will retrieve robots.txt.

- From a Kali Linux issue the command:

```
wget http://192.168.1.100/robots.txt
```

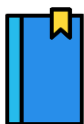
- The output should resemble the Figure below.

```
(kali@kali)-[~]
$ wget http://192.168.1.100/robots.txt
--2022-08-03 10:55:22-- http://192.168.1.100/robots.txt
Connecting to 192.168.1.100:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30 [text/plain]
Saving to: 'robots.txt'

robots.txt          100%[=====>]          30  --.-KB/s   in 0s

2022-08-03 10:55:22 (4.71 MB/s) - 'robots.txt' saved [30/30]
```

- Print the file to the terminal using the cat command, then pipe this through to grep as seen in the previous curl example.



Take note!

Redownloading the robots.txt and comparing it to previous iterations may be a good way to keep on changes in a client's web application.

3.2 IDENTIFYING TECHNOLOGIES USING TOOLING

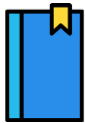
Using specific tooling to perform technology identification automatically improves efficiency - in both speed and scale. However, it is important to note that while using tools has its advantages, it can have limitations.

3.2.1 Passive Methods

Passive methods generally involve the tester not interacting with the target system. However, a grey area begins to form when discussing the use of 3rd party tools which interact with the target on the tester's behalf. Is this still passive? I tend to lean towards yes this is passive, unless the 3rd party tool is more intense than "normal user behaviour".

3.2.1.1 *Using 3rd Party Online Tools*

The next section details the usage of various online tools which can identify the underlying technologies used in web applications. As these tools are free to use, there are limitations. Therefore when completing this exercise it is common for these tools to rate-limit, throttle, or eventually block you from using the tool. As such you may wish to choose just one of the following tools.



Take note!

3rd party tools such as NetCraft, WhatCMS, and Wappalyzer are restricted from what they can access. You couldn't use them on an intranet test for example!

3.2.1.1.1 Using Netcraft

From the netcraft web site, find out the web Server software and version that www.abertay.ac.uk is running.

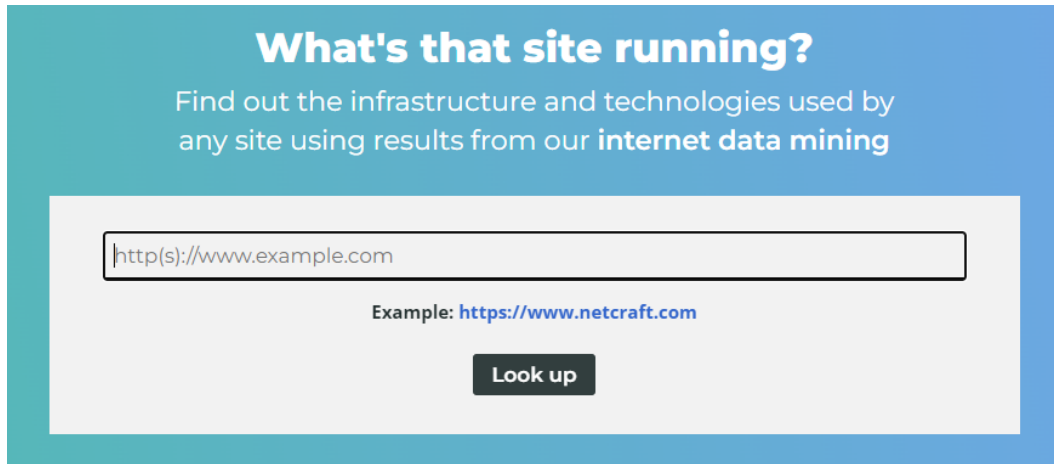


Caution!

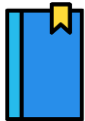
It is likely that Netcraft will rate-limit and/or throttle interactions.

- Browse to <https://sitereport.netcraft.com/>

- Scroll down! – You’ll eventually find the widget as seen in the Figure below.



- Provide a target website, such as Abertay.ac.uk, make sure to include the full url containing http(s) where appropriate.



Take note!

Netcraft provides historical information and other material, which might be useful to an attacker.

3.2.2 Using WhatCMS

Similar to Netcraft, the site WhatCMS scans the target website to identify the Content Management System (CMS) used.



Caution!

It is likely that WhatCMS will rate-limit and/or throttle interactions.



WhatCMS' Website.

<https://whatcms.org/>

- Browse to WhatCMS' website and provide a target, such as Abertay.ac.uk.

- Once complete, the results should appear below the search bar, as seen the Figure below.

Examine these results.

What CMS Is This Site Using?

Currently detecting 1326 website powering technologies

Q Detect CMS

✓ Success

www.abertay.ac.uk uses

Category	Software	Version
Other CMS, CMS	Umbraco	
Web Framework	Microsoft ASP.NET	
CDN	Azure CDN	

3.2.3 Using Wappalyzer

Similar to the Netcraft and WhatCMS, however, perhaps more beefier, Wappalyzer can identify many aspects regarding the target web application.

Wappalyzer's Website.

<https://www.wappalyzer.com/>

Wappalyzer

[Products](#) ▼
[Pricing](#)
[Resources](#) ▼
[Sign in](#)

Sign up free

Identify technologies on websites

Find out the technology stack of any website. Create lists of websites that use certain technologies, with company and contact details. Use our tools for lead generation, market analysis and competitor research.

Q



It is likely that Wappalyzer will rate-limit and/or throttle interactions.

3.2.4 Active Methods

3.2.4.1 Using WhatWeb

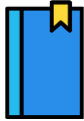


- <https://github.com/urbanadventurer/WhatWeb>

26 | Page

- First, from a terminal, use -h to discover the options for running the whatweb tool.

whatweb -h



Take note!

Ensure you can access the Vulnerable Applications by browsing to their address first. If accessible, then proceed with the tasks. If not, spend time debugging it.

- Next, we will want to test out the tool. To do we will try it out against a target.

whatweb 192.168.1.100

- The output of the command should look like the Figure below.

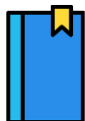
```
(kali@kali)-[~]
$ whatweb 192.168.1.100
http://192.168.1.100 [200 OK] Apache[2.2.14][mod_mono/2.4.3,mod_perl/2.0.4,mod_python/3.3.1,mod_ssl/2.2.14,proxy_html/3.0.1], Bootstrap, Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1], IP[192.168.1.100], OpenSSL[0.9.8k], PHP[5.3.2-1ubuntu4.30][Suhosin-Patch], Passenger[4.0.38], Perl[5.10.1], Python[2.6.5], Script, X-Powered-By[PHP/5.3.2-1ubuntu4.30]
```

- Next, we will want to test out the tool. To do we will try it out against a target.

whatweb 127.0.0.1:3000

- The output of the command should look like the Figure below.

```
(kali@kali)-[~]
$ whatweb 127.0.0.1:3000
http://127.0.0.1:3000 [200 OK] Country[RESERVED][ZZ], HTML5, IP[127.0.0.1], JQuery[2.2.4], Script[module], Title[OWASP Juice Shop], UncommonHeaders[access-control-allow-origin,x-content-type-options,feature-policy,x-recruiting], X-Frame-Options[SAMEORIGIN]
```



Take note!

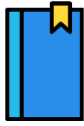
Make sure to read the output in detail and understand what each of the results mean.

3.2.4.2 Using NMAP

While specific fingerprinting tools exist, such as WhatWeb, more generic tools can also accomplish the same results. NMAP is one such tool.

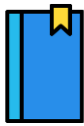
- From a Kali terminal issue, the following command to port scan the web server and hopefully identify associated applications also operating on the target:

```
sudo nmap -p- -sS 192.168.1.100
```



Take note!

The -p- flag is shorthand for scanning all possible 65535 ports (but not port 0).
The -sS flag enables SYN scanning, which is much faster than a full TCP scan.



Take note!

Sudo is required due to the permissions required to create the custom SYN packets used in a -sS scan.



If interested, read more about the SYN scanning process in the NMAP book.

- <https://nmap.org/book/synscan.html>

- The output should resemble the Figure below.

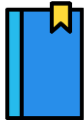
```
(kali@kali)-[~]
└─$ sudo nmap -p- -sS 192.168.1.100
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-03 10:57 EDT
Nmap scan report for 192.168.1.100
Host is up (0.0039s latency).
Not shown: 65526 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
143/tcp   open  imap
443/tcp   open  https
445/tcp   open  microsoft-ds
5001/tcp  open  complex-link
8080/tcp  open  http-proxy
8081/tcp  open  blackice-icecap
MAC Address: 00:0C:29:E5:68:6D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 6.45 seconds
```

- From these results, we can see no service is running on a TCP port higher than 8081, therefore we can tailor our more intensive scan (and thus waste less time).

```
nmap -p 1-10000 -sV 192.168.1.100
```

```
(kali@kali)~$ sudo nmap -p 1-10000 -sV 192.168.1.100
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-03 10:59 EDT
Nmap scan report for 192.168.1.100
Host is up (0.0032s latency).
Not shown: 9991 closed tcp ports (reset)
PORT      STATE SERVICE        VERSION
22/tcp    open  ssh            OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http           Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30)
139/tcp   open  netbios-ssn    Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp   open  imap           Courier Imapd (released 2008)
443/tcp   open  ssl/http       Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30)
445/tcp   open  netbios-ssn    Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
5001/tcp  open  java-object    Java Object Serialization
8080/tcp  open  http           Apache Tomcat/Coyote JSP engine 1.1
8081/tcp  open  http           Jetty 6.1.25
1 service unrecognized despite returning data. If you know the service/version, please submit
SF-Port5001-TCP:V=7.92%I=7%D=8/3%Time=62EA8D74%P=x86_64-pc-linux-gnu%r(NUL
SF:L,4,"\xac\xed\x05");
MAC Address: 00:0C:29:E5:68:6D (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

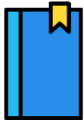


Take note!

The -sV flag attempts to fingerprint the applications located on the target ports, like the previously used WhatWeb, but for all type of applications.

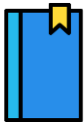
The command output shown above shows there may be a web server running on ports 8080 and 8081.

- Prove this finding by browsing to <http://192.168.1.100:8080> and <http://192.168.1.100:8081>.



Take note!

If you want to be thorough, you would crosscheck all discovered applications with the NVD, and other sources.



Take note!

NMAP has further uses in Web Application Hacking. Appendix A includes an exercise to explore this further functionality.

4 NAVIGATING THE WEB LIKE A SPIDER

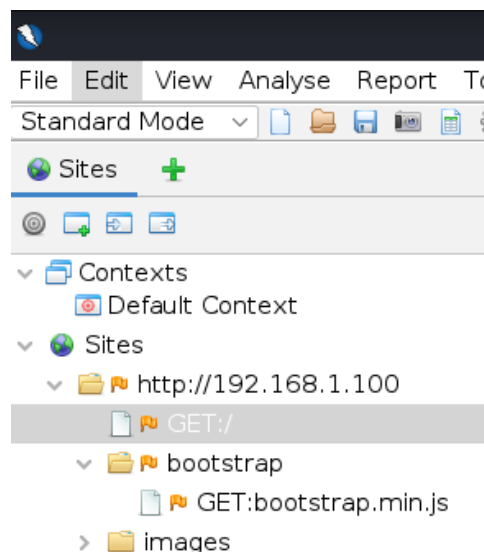
Web application spidering, also known as web crawling, is the name given to the process of navigating through webpages to accumulate a comprehensive map of the website. The term "spidering" is derived from the way a spider navigates through a web by following links from one page to another. While this can be performed manually, it is commonly done through automation in which a script systematically visits web pages, follows hyperlinks, and gathers data from each visited page.

It's important to note that while web application spidering can be a powerful tool for various legitimate purposes, it can also be misused for unethical or illegal activities, such as data scraping or website cloning. As such many websites have measures in place to prevent or limit automated crawling to protect their data and server resources.

4.1 MANUAL SPIDERING.

4.1.1 Using a Proxy

As you navigate a website while using a proxy, it will begin to create a site map as seen in the figure below. As the user is required to navigate to each page, creating a map of the site might take a considerable time. As such, automated spidering is more efficient.

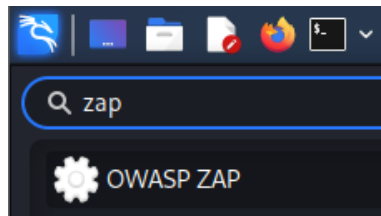


4.2 AUTOMATED SPIDERING.

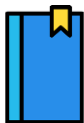
4.2.1 Using OWASP ZAP

We will use OWASP ZAP to spider the WackoPicko website.

- In Kali Linux, run the OWASP ZAP proxy, this can be accomplished by searching for it in the menus, as seen in the Figure below.

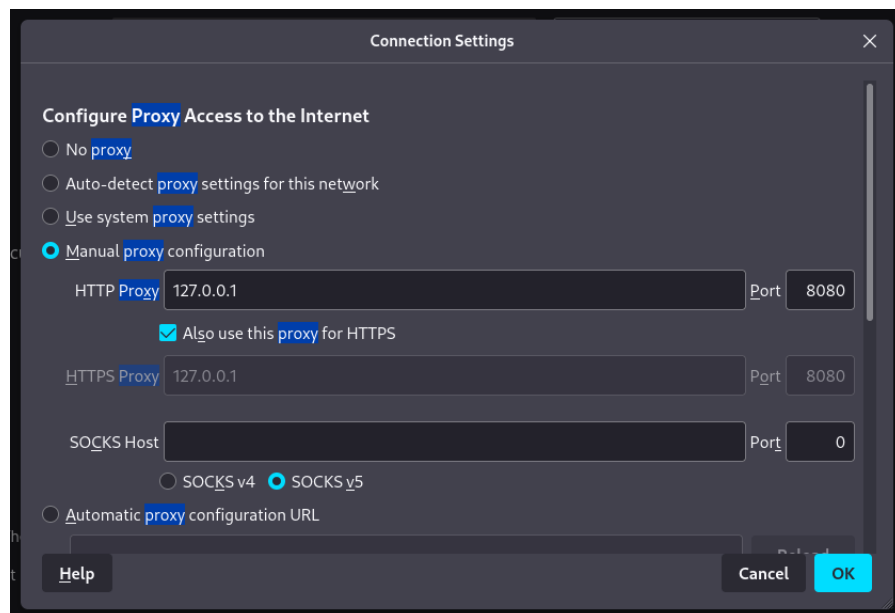


- Then configure the Firefox web browser to use the ZAP proxy, as seen in the Figure below.



Take note!

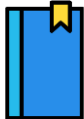
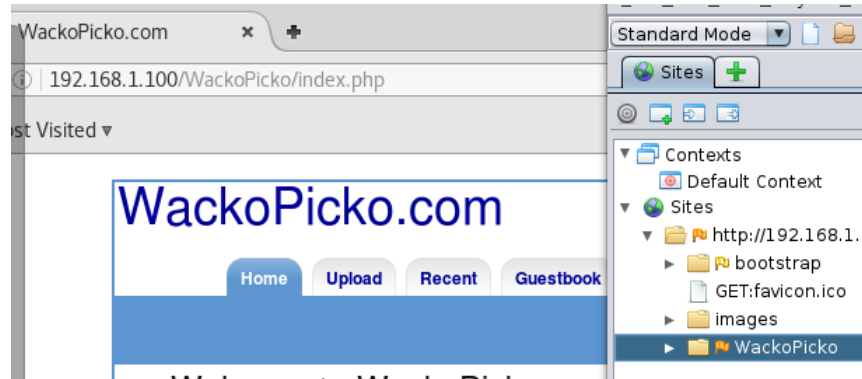
Ensure the “Also use this proxy for HTTPS” is enabled!



- From Firefox, browse to the Wackopicko site located at the url below, and authenticate with the credentials bryce/bryce.

<http://192.168.1.100/WackoPicko/index.php>

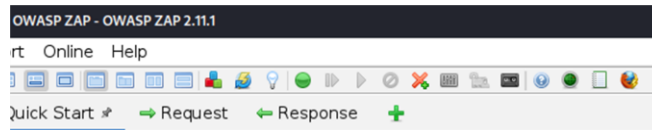
- Once you have successfully navigated to the page above, OWASP ZAP should start to populate a WackoPicko directory in the sites tab - as seen in the screenshot below.



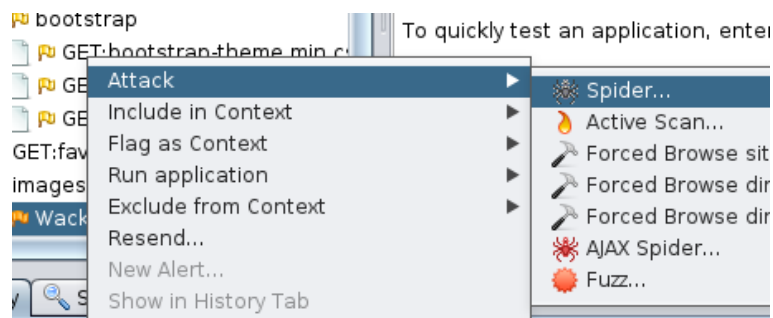
Take note!

If the WackoPicko directory is not in this tab, it is likely due to your proxy configuration, give it another look to check over. If you are sure the configuration is correct, then it might be the pesky “ZAP HUD” feature.

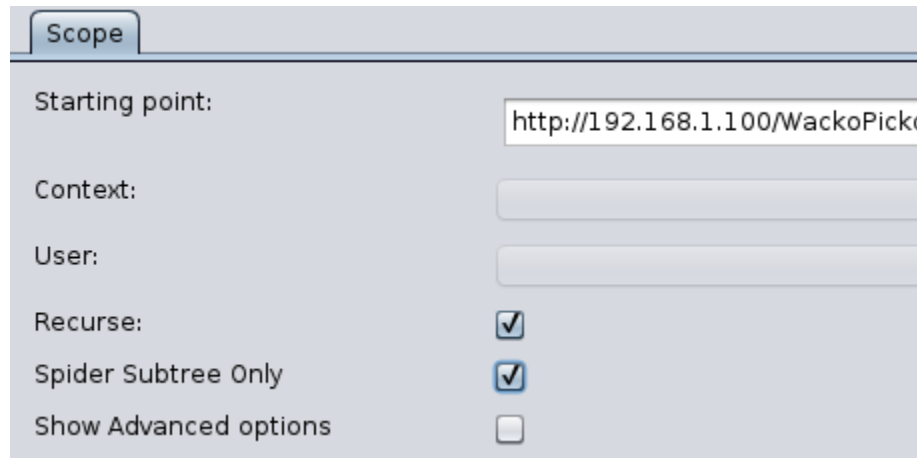
Ensure the green icon, 3rd from the right on the taskbar at the top of the window, is inactive. It should look something like the picture below.



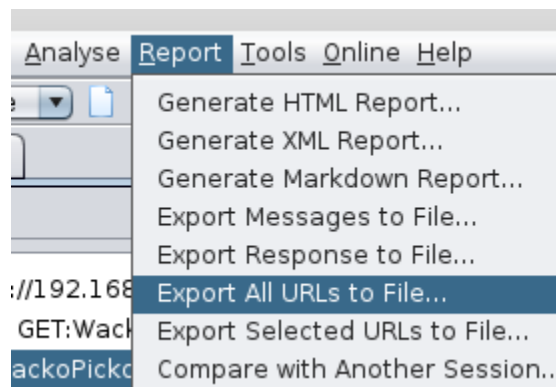
- In OWASP ZAP, right-click on WackoPicko folder, and navigate the menus (as seen in the Figure below) to select Spider option.



- You will be greeted by a prompt, where you can configure the spider's settings. To ensure a targeted approach when spidering, make sure the "Spider Subtree Only" is selected. Then run the spider - it may take a few seconds to collate all the information.



- Once complete, it is possible to export all the URLs discovered to a file for further analysis.



- Once you've explored the results, make sure to turn off the Proxy settings in Firefox!



Caution!

Make sure to disable the proxy settings, or else you may complicate further exercises.

4.2.2 Using Burpsuite

Unfortunately, Burpsuite's spider functionality is not available on the free community edition.

4.2.3 Using GoSpider

GoSpider is an open-source web application spidering tool written in Go, hence the name. Like previous exercises, this Go tool is a possible alternative to OWASP Zap's spidering feature.



GoSpider's Github Repo:

<https://github.com/jaeles-project/gospider>

- First, run GoSpider with the -h flag to examine all available features.

`gospider -h`

```
(kali@kali)-[~]
$ gospider -h
Fast web spider written in Go - v1.1.6 by @thebl4ckturtle & @j3ssiej3j

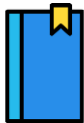
Usage:
  gospider [flags]

Flags:
  -s, --site string           Site to crawl
  -S, --sites string         Site list to crawl
  -p, --proxy string          Proxy (Ex: http://127.0.0.1:8080)
  -o, --output string         Output folder
  -u, --user-agent string     User Agent to use
                              web: random web user-agent
                              mobi: random mobile user-agent
                              or you can set your special user-agent (default "web")
  --cookie string             Cookie to use (testA=a; testB=b)
  -H, --header stringArray    Header to use (Use multiple flag to set multiple header)
  --burp string               Load headers and cookie from burp raw http request
  --blacklist string          Blacklist URL Regex
  --whitelist string          Whitelist URL Regex
  --whitelist-domain string   Whitelist Domain
```

- Like the previous exercise, we will want to run the tool against the WackoPicko site. The output should look like that seen in the figure below.

`gospider -s "http://192.168.1.100/WackoPicko"`

```
(kali㉿kali)-[~]
$ gospider -s "http://192.168.1.100/WackoPicko/"
[url] - [code-200] - http://192.168.1.100/WackoPicko/
[href] - http://192.168.1.100/WackoPicko/css/blueprint/screen.css
[href] - http://192.168.1.100/WackoPicko/css/blueprint/print.css
[href] - http://192.168.1.100/WackoPicko/css/stylings.php
[href] - http://192.168.1.100/WackoPicko/
[href] - http://192.168.1.100/WackoPicko/users/home.php
[href] - http://192.168.1.100/WackoPicko/pictures/upload.php
[href] - http://192.168.1.100/WackoPicko/pictures/recent.php
[href] - http://192.168.1.100/WackoPicko/guestbook.php
[href] - http://192.168.1.100/WackoPicko/users/login.php
[href] - http://192.168.1.100/WackoPicko/users/register.php
[href] - http://192.168.1.100/WackoPicko/users/sample.php?userid=1
[href] - http://192.168.1.100/WackoPicko/calendar.php
[href] - http://192.168.1.100/WackoPicko/admin/index.php?page=login
[href] - mailto:contact@wackopicko.com
[href] - http://192.168.1.100/WackoPicko/tos.php
[form] - http://192.168.1.100/WackoPicko/
```



Take note!

You should notice a considerable difference in execution speed, along with the number of results in comparison to OWASP Zap.

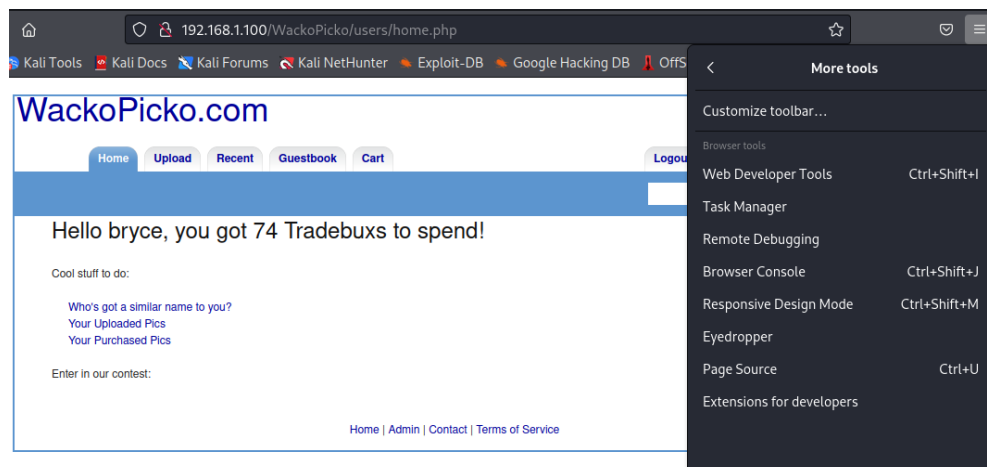
Next, we will supply the tool with the authentication cookie from the WackoPicko, rerunning the tool again to observe any differences.

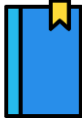


Take note!

It is typical for modern websites to restrict many pages behind login pages. To circumvent this, we can provide authentication cookies to our spiders.

- Start by authenticating on WackoPicko, then using navigating to More Tools under Menu, as seen in the Figure below. Select the Web Developer Tools from this More Tools list.

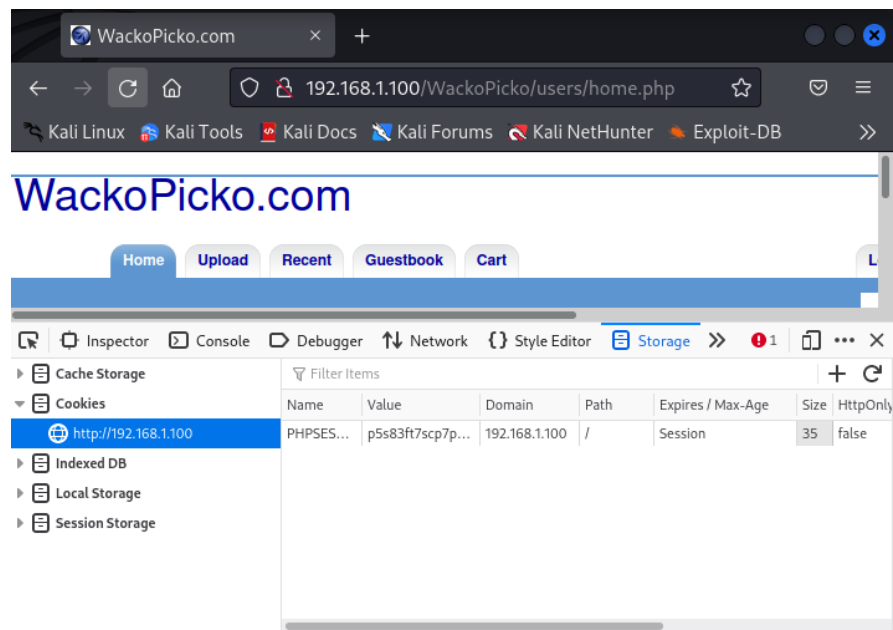




Take note!

Alternatively, use the Ctrl+Shift+I hotkey.

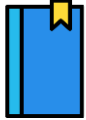
- From the Web Developer Tools, navigate to the Storage tab, and select the Cookies option on the left. This will reveal the PHPSESSION cookie provided by the website when you logged in. This step can be seen in the Figure below.



- Copy the Name and Value into the command line arguments for gospider, as seen in the Figure below.

```
gospider -s "http://192.168.1.100/WackoPICKO" --cookie "PHPSESSION=[COOKIE]"
```

```
(kali@kali)-[~]
$ gospider -s "http://192.168.1.100/WackoPICKO/" --cookie "PHPSESSION=p5s83ft7scp7pob85st1vqdbu0"
[url] - [code-200] - http://192.168.1.100/WackoPICKO/
[href] - http://192.168.1.100/WackoPICKO/css/blueprint/screen.css
[href] - http://192.168.1.100/WackoPICKO/css/blueprint/print.css
[href] - http://192.168.1.100/WackoPICKO/css/stylings.php
[href] - http://192.168.1.100/WackoPICKO/
[href] - http://192.168.1.100/WackoPICKO/users/home.php
[href] - http://192.168.1.100/WackoPICKO/pictures/upload.php
[href] - http://192.168.1.100/WackoPICKO/pictures/recent.php
[href] - http://192.168.1.100/WackoPICKO/guestbook.php
[href] - http://192.168.1.100/WackoPICKO/cart/review.php
[href] - http://192.168.1.100/WackoPICKO/users/logout.php
[href] - http://192.168.1.100/WackoPICKO/users/register.php
[href] - http://192.168.1.100/WackoPICKO/users/sample.php?userid=1
[href] - http://192.168.1.100/WackoPICKO/calendar.php
[href] - http://192.168.1.100/WackoPICKO/
[href] - http://192.168.1.100/WackoPICKO/admin/index.php?page=login
[href] - mailto:contact@wackopicko.com
[href] - http://192.168.1.100/WackoPICKO/tos.php
[form] - http://192.168.1.100/WackoPICKO/
```



Take note!

You should notice a difference in output in comparison to the unauthenticated results.

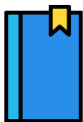
If you are struggling to identify any differences, then a bit of Command Line Kung Fu can help you out.

- Rerun the two commands, however, this time pipe the output to respective files. As seen in the Figure below.

```
(kali@kali)-[~]  
$ gospider -s "http://192.168.1.100/WackoPicko/" > scan1  
  
(kali@kali)-[~]  
$ gospider -s "http://192.168.1.100/WackoPicko/" --cookie "PHPSESSID=p5s83ft7scp7pob85st1vqdbu0" > scan2
```

- Then use the diff command while providing the two different files as arguments. The output of this command will highlight the differences between each file. As seen in the Figure below.

```
(kali@kali)-[~]  
$ diff scan1 scan2  
10c10,11  
< [href] - http://192.168.1.100/WackoPicko/users/login.php  
_____  
> [href] - http://192.168.1.100/WackoPicko/cart/review.php  
> [href] - http://192.168.1.100/WackoPicko/users/logout.php
```



Take note!

There isn't much different in this example, however, in some circumstances there can be significant differences in (un)authenticated spidering.

4.2.4 Using Other Tools

There are plenty of other tools you can use for spidering. The links below highly just a few.



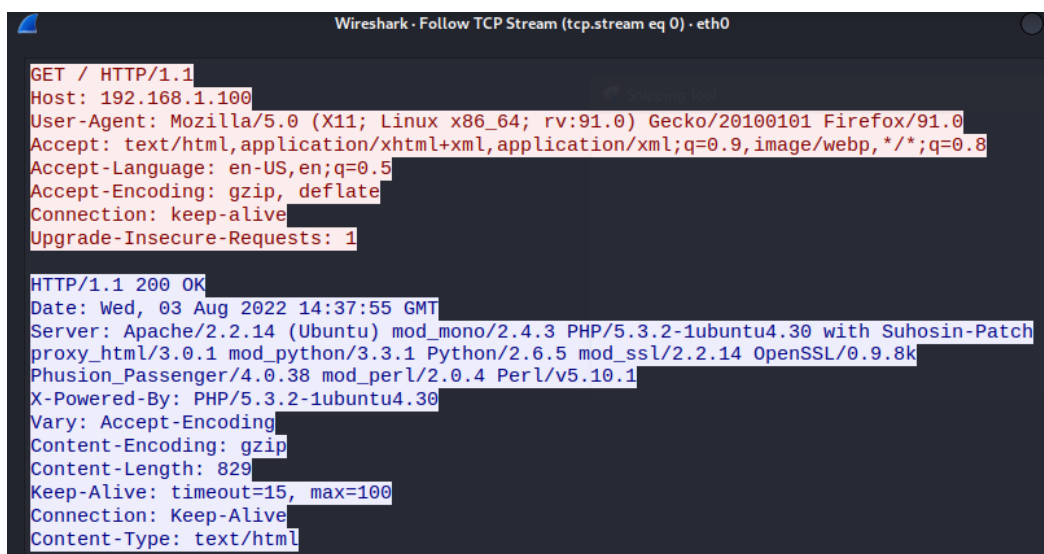
Meg tool

<https://github.com/tomnomnom/meg>

5 HEADACHES WITH HTTP HEADERS

HTTP headers are a fundamental part of the Hypertext Transfer Protocol. HTTP headers are pieces of additional information sent by the server in the response to an HTTP request made by a client. These additional headers provide metadata about the request or the response and can convey various details related to the web application and its server.

We've previously come across HTTP Headers in this exercise when fingerprinting the server. The below figure shows correspondence between server and client including ample use of HTTP Headers.



```
Wireshark · Follow TCP Stream (tcp.stream eq 0) · eth0

GET / HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Wed, 03 Aug 2022 14:37:55 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch
proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k
Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 829
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Some HTTP Headers look to add functionality to an application, others look to improve security. Regardless of their function, they further add to potential attack vectors to an already complex environment. The links below contain some fantastic resources on HTTP headers.



Mozilla Web Docs on HTTP Headers

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>



OWASP Secure Headers Project

<https://owasp.org/www-project-secure-headers/>

5.1 IDENTIFYING HTTP HEADERS

There are many possible headers found in a HTTP response, a subset of these are security related. When enabled, these headers can improve the resilience of a web application against many common attacks, including cross-site scripting (XSS). The following exercises look at identifying these security related HTTP headers.

5.1.1 Passive Methods

5.1.1.1 Manual Browsing and Wireshark

As previously explored looking at the network traffic generated when browsing the web can reveal the various HTTP headers implemented on a given page or site.

5.1.1.2 Using Security Headers

Security Headers, created by security researcher Scott Helme, is an online tool which scans the website provided for the relevant security headers and creates a report card.




Security Headers' website.

<https://securityheaders.com/>



Caution!

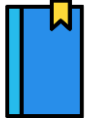
It is likely that Security Headers will rate-limit and/or throttle interactions.

Security Headers
Sponsored by  **Report URI**

Scan your site now

☐ Hide results ☒ Follow redirects

- Scan a website you are familiar with, and explore the associated report card generated.



Take note!

Select 'Hide Results' to avoid the results being shared publicly.



Analysis of the adoption of security headers in HTTP

<https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-ifs.2016.0621>

5.1.2 Active Methods

The functionality of these sites like Security Headers is limited to websites accessible on the Internet. Thus, when testing internal sites cannot be tested in the same manner, as such, you should find alternative tools. Thankfully, some of the features in previously explored tools have this functionality.

5.1.2.1 Using NMAP

The NMAP contains scripts which check for HTTP headers.

- From a terminal in Kali, run the following commands against the target.

```
nmap 192.168.1.100 -p80,443 --script=http-headers
```

```
nmap 192.168.1.100 -p80,443 --script=http-methods
```

5.1.2.2 Using Nikto

Nikto also contains a plugin to check for HTTP headers.

- From a terminal in Kali, run the following commands against the target.

```
nikto -Plugins headers -h http://192.168.1.100
```

```
nikto -Plugins headers -h http://192.168.1.100:8080
```

```
nikto -Plugins headers -h http://192.168.1.100:9080
```


6 IDENTIFYING ENDPOINTS

Endpoints in a web application refer to specific URLs or API routes that serve as entry points for interaction with the application's functionality. These endpoints are critical components to scrutinise, as they often represent potential attack surfaces where vulnerabilities may lurk. Meticulously examining these endpoints can help identify security weaknesses, such as input validation flaws, authentication and authorisation issues, injection vulnerabilities, and more. However, to do so first we must identify where these end points are.

To do so you will have to investigate each page and linked resource. This can be accomplished in spidering. However, when investigating resources linked JavaScript files, this can go beyond the skillset of spidering tools.

Going beyond spidering, a tester will have to investigate each page and linked resource.

Beyond simply spidering, we can investigate the end points of an application to get an understanding how the web application operates.

6.1 IDENTIFYING JAVASCRIPT FILES AND LINKS

For modern web apps, the use of client-side JavaScript for the frontend is becoming more popular. Like the comments and metadata in HTML code, many programmers also hardcode sensitive information in JavaScript variables on the frontend. Sensitive information can include Private API Keys, internal IP addresses, sensitive routes, or even credentials.

While information like this can increase the attack surface area, and some information may even be valid report findings (Sensitive info such as passwords and API keys), these JavaScript files can also be examined for further vulnerabilities. These can include identifying potentially exploitable code or discovering outdated and unsupported frameworks.

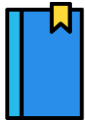
Obviously, this can be done manually by inspecting the files sent across to the client. However, when examining much larger websites, it may be necessary to look at automated methods. One tool you can use is Hakrawler.



Hakrawler Tool

<https://github.com/hakluke/hakrawler>

- Navigate to the Hakrawler GitHub page. Locate the Installation instructions and follow them.



Take note!

Hakrawler has multiple set of installation instructions, feel free to pick which method you prefer.

- Run the tool against a site such as juiceshop located at 127.0.0.1:3000 and examine the output.

```
(kali㉿kali)-[~]  
$ echo http://127.0.0.1:3000/#/ | hakrawler  
http://cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js  
http://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js  
http://127.0.0.1:3000/runtime.js  
http://127.0.0.1:3000/polyfills.js  
http://127.0.0.1:3000/vendor.js  
http://127.0.0.1:3000/main.js
```

- Click on the links to the files identified to examine the discovered JavaScript files.

There are many tools within this area, you should explore them via the links below.



LinkFinder Tool

<https://github.com/GerbenJavado/LinkFinder>



Deobfuscator Tool

<http://deobfuscate.io/>

6.2 IDENTIFYING HTTP COMMANDS

HTTP headers are constantly evolving area of web application security, and if you want to learn more about it, check out the link below for a good resource to start

6.2.1.1 The Options Method

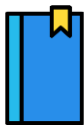
The OPTIONS command in HTTP requests is used to discover what HTTP commands are supported by that web server, and page. This may be disabled by some admins and as such, the output will not display the “Allow:” field in the response.

- First, browse to 192.168.1.100, and select the bWAPP application, then return to the Kali terminal, and enter the following:

```
nc 192.168.1.100 80
```

```
OPTIONS / HTTP/1.1
```

```
host: 192.168.1.100
```

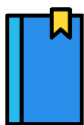
**Take note!**

You will want to copy and paste the above commands, as the connection created by netcat can be closed before you write them manually.

- Examine the response, it should resemble something like the Figure below.

```
HTTP/1.1 200 OK
Date: Mon, 15 Sep 2014 08:09:39 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fas
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 0
Content-Type: text/html
X-Pad: avoid browser bug
```

- Notice that the response we receive does not contain the “Allow” record, meaning that we cannot view which HTTP methods are enabled. From the screenshot, we can see the TRACE command is enabled. Enabling superfluous commands, could be considered dangerous. In particular, the TRACE command may be used in certain Cross-site scripting attacks.

**Take note!**

Various commands can be disabled when configuring the Web Server.

7 DISCOVERING HIDDEN ELEMENTS

Many web applications will have content that is not obtainable through the spidering method. This content is often tied to pages since removed, or accidentally exposed to the Internet. Brute-force applications attempt to find this content by sending requests to the server. However, the effectiveness of the tool is dependent on the wordlist provided.

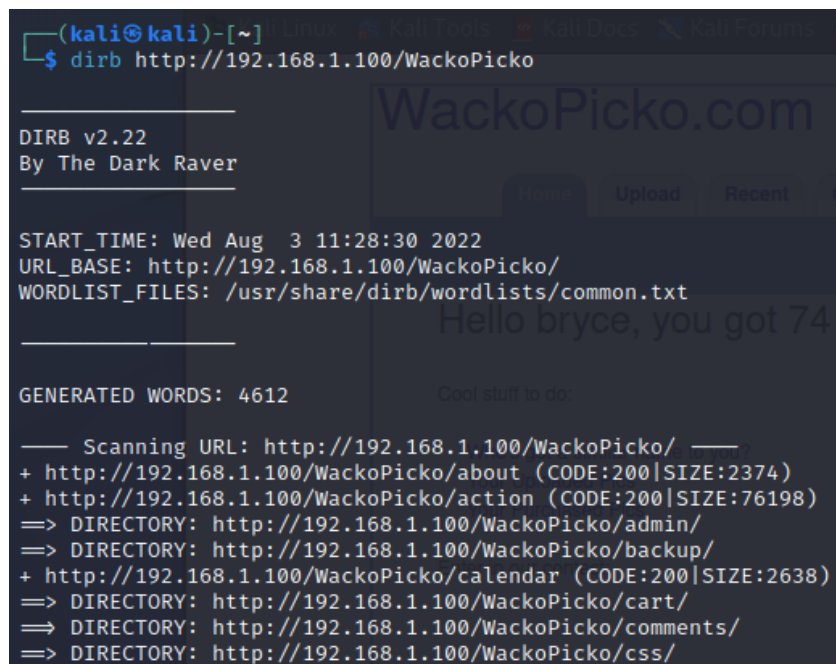
7.1 BRUTE-FORCING HIDDEN FOLDERS AND FILES.

7.1.1 Using dirb

One common bruteforcing tool is dirb, it is preinstalled on Kali. To see the tool in use, follow the instructions below.

- From a Kali terminal, input the following command to bruteforce the WackoPicko application. The output should resemble the Figure below.

dirb http://192.168.1.100/WackoPicko



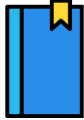
```
(kali㉿kali)-[~]
└─$ dirb http://192.168.1.100/WackoPicko

DIRB v2.22
By The Dark Raver

START_TIME: Wed Aug 3 11:28:30 2022
URL_BASE: http://192.168.1.100/WackoPicko/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

— Scanning URL: http://192.168.1.100/WackoPicko/ —
+ http://192.168.1.100/WackoPicko/about (CODE:200|SIZE:2374)
+ http://192.168.1.100/WackoPicko/action (CODE:200|SIZE:76198)
=> DIRECTORY: http://192.168.1.100/WackoPicko/admin/
=> DIRECTORY: http://192.168.1.100/WackoPicko/backup/
+ http://192.168.1.100/WackoPicko/calendar (CODE:200|SIZE:2638)
=> DIRECTORY: http://192.168.1.100/WackoPicko/cart/
=> DIRECTORY: http://192.168.1.100/WackoPicko/comments/
=> DIRECTORY: http://192.168.1.100/WackoPicko/css/
```



Take note!

Dictionaries are at `/usr/share/dirb/wordlists` & `/usr/share/dirbuster/wordlists`.

- Run the program again, and this time specify the word list you wish to use.

dirb <http://192.168.1.100/WackoPicko> /usr/share/dirb/wordlists/common.txt

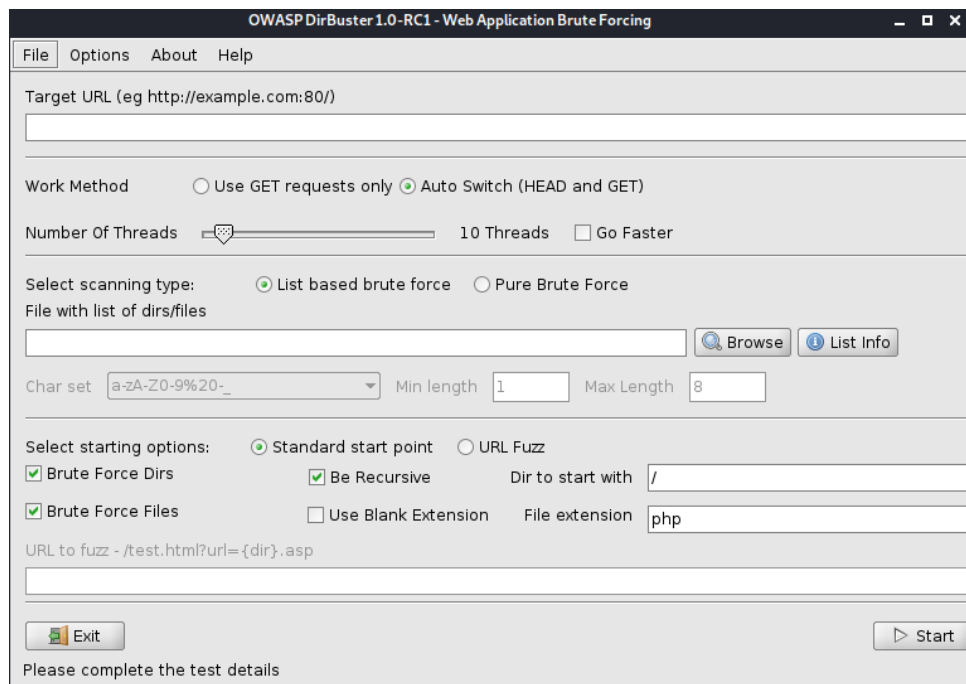
7.1.2 Using dirbuster

Beyond Dirb, there is DirBuster. This tool is a multi-threaded java application (yuck!) designed to brute force directories and files names on web/application servers. This GUI application is easy to operate.

- It can be launched by issuing the following command from the terminal.

```
dirbuster
```

- This will launch the GUI interface, as pictured below, where you can alter the settings to your hearts content.



- Launch a bruteforce attempt at the WackoPicko application, using the same wordlist as previously, and compare the timings.

7.1.3 Using gobuster

Another bruteforcing tool is gobuster. As you probably figured, this tool is a bruteforcing tool written in Go, hence the name. Unlike dirbuster, there isn't a GUI for this tool, it's all command line.



gobuster's GitHub repository.

- <https://github.com/OJ/gobuster>

- Start by checking the options for running the tool by using the -h flag.

Gobuster -h

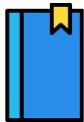
```
(kali@kali)-[~]
$ gobuster -h
Usage of gobuster:
-P string      Password for Basic Auth (dir mode only)
-U string      Username for Basic Auth (dir mode only)
-a string      Set the User-Agent string (dir mode only)
-c string      Cookies to use for the requests (dir mode only)
-cn           Show CNAME records (dns mode only, cannot be used with '-i' option)
-e           Expanded mode, print full URLs
-f           Append a forward-slash to each directory request (dir mode only)
-fw          Force continued operation when wildcard found
-i           Show IP addresses (dns mode only)
-k           Skip SSL certificate verification
-l           Include the length of the body in the output (dir mode only)
-m string      Directory/File mode (dir) or DNS mode (dns) (default "dir")
-n           Don't print status codes
```

- Next, launch the target towards the WackoPicko application with the same wordlist as previously used.

gobuster -u <http://192.168.1.100/WackoPicko> -w /usr/share/dirb/wordlists/common.txt

```
(kali㉿kali)-[~]
$ gobuster -u http://192.168.1.100/WackoPicko -w /usr/share/dirb/wordlists/common.txt

Gobuster v2.0.1           OJ Reeves (@TheColonial)
=====
[+] Mode           : dir
[+] Url/Domain      : http://192.168.1.100/WackoPicko/
[+] Threads        : 10
[+] Wordlist        : /usr/share/dirb/wordlists/common.txt
[+] Status codes    : 200,204,301,302,307,403
[+] Timeout        : 10s
=====
2022/08/03 11:31:45 Starting gobuster
=====
/.hta (Status: 403)
/.htpasswd (Status: 403)
/.htaccess (Status: 403)
/.svn (Status: 403)
/.svn/entries (Status: 403)
/about (Status: 200)
/action (Status: 200)
/admin (Status: 301)
/backup (Status: 301)
/calendar (Status: 200)
/cart (Status: 301)
/comments (Status: 301)
/css (Status: 301)
/error (Status: 200)
/guestbook (Status: 200)
/images (Status: 301)
/include (Status: 403)
/index.php (Status: 200)
/index (Status: 200)
/pictures (Status: 301)
/test (Status: 200)
/tos (Status: 200)
/upload (Status: 301)
/users (Status: 301)
```



Take note!

You should notice a considerable execution speed difference between dirb and dirbuster and gobuster.

7.1.4 Using dirsearch

Another similar tool in which can perform this task is dirsearch. The open-source tool is still receiving daily updates and tweaks. Learn more about the tool, and perhaps try it yourself at the link below.



dirsearch's GitHub repository.

- <https://github.com/maurosoria/dirsearch>

8 WRITING VULNERABILITY REPORTS

While enumeration does not explicitly try to identify vulnerabilities, the process can lead to interesting discoveries. However, finding these curiosities is only half the battle. The penetration tester or bug bounty hunter must then report this to the client. For some, writing the associated report can be as hard as finding the vulnerability. Good written communication skills are essential for effective vulnerability reporting. To help develop these vital soft skills, the following exercises focus on practicing how to write vulnerability reports.



Consider...

Use this exercise to commence with independent personal research. Start by understanding each vulnerability, and then judge how best to present this to the respective client

8.1 SCENARIO 1 - AN EXPOSED ADMIN PANEL

After a critical success, a local independent video game company was bought over by a larger publishing house. With this buy over, a restructure of the infrastructure took place, with old technologies being ported to new hardware. As part of this restructure, the developer's intranet server was relocated. As customary during the acquisition process, a penetration test was undertaken.

One target within scope was the intranet web server and application. On this web server, a new web application on a typical port was identified. Comparing the scan results with previous penetration testing reports reveals this application was not operating during previous engagements. After further investigation, this new application only appears to be serving the one web page. A non-descript login page with the title "Admin Portal". While brute forcing this login page was out of scope, discovering this new service and page is worth noting in the report.



Writing Exercise

Write a report detailing this admin page, and the possible security implications that may arise from its current existence.

**Caution!**

The source of this new application is unknown; however, it could be business critical. Perhaps take a hypothetical approach rather than presuming severity.

8.2 SCENARIO 2 - A SUBDOMAIN TAKEOVER

Over the summer, the National History Museum showcased a new seasonal exhibit detailing the average Scottish summer holidays throughout the years. This exhibit was curated by Edinburgh Napier University's International Tourism Management Programme. As part of the exhibit, a subdomain hosted by in a 3rd party cloud was populated with content. The University managed this subdomain without much oversight from the Museum. Once the exhibit was finished, the web server hosting the subdomain was taken offline, however, the DNS record for the exhibit's web page remained. In the most recent engagement, you have been able to successfully takeover this subdomain.

**Writing Exercise**

Write a vulnerability report for the possible subdomain takeover identified above.

**Question**

What is the best way to provide a Proof-of-Concept in a subdomain for a client, while minimising any potential damage to their brand image?


**Subdomain Takeover: Proof Creation for Bug Bounties**

<https://Oxpatrik.com/takeover-proofs/>

8.3 SCENARIO 3 - POOR HTTP HEADER IMPLEMENTATION

Upon opening the new public fitness centre, the council launched the associated website. Unlike other council-ran fitness facilities, this website is rather intricate with functionality such as the ability to track your gym visits, create a routine, book sessions, hire pitches and vote on events. This website was created by an expensive boutique web development firm. Thankfully, they are quite clued up when it comes to

cybersecurity and launch their products after rigorous internal and external security assessments. However, in the most recent penetration testing engagement of the fitness centre's website you have noticed the HTTP Header implementation is significantly lacking behind modern standards. The results of the associated Security Headers scan can be found in the Figure below.

Security Report Summary	
	Site:
	IP Address:
	Report Time: 10 Aug 2022 09:18:54 UTC
	Headers: ✖ Content-Security-Policy ✖ X-Frame-Options ✖ X-Content-Type-Options ✖ Referrer-Policy ✖ Permissions-Policy
	Warning: Grade capped at A, please see warnings below.



Writing Exercise

Write a vulnerability report for the possible issues created by poor HTTP headers implementation - using the Security Headers scan as a basis for your report.



Caution!

HTTP Headers are not the be-all and end-all, while some headers are symptomatic of larger issues, others can be perceived as perfectionist.

9 APPENDIX

9.1 APPENDIX A - NMAP SCRIPTS

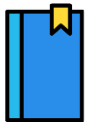
The NMAP Scripting Engine (NSE) is one of NMAP's most powerful and flexible features. It allows users to write (and share) simple scripts to automate a wide variety of networking tasks. There are many that are useful in Web Application testing.



A list of the NMAP scripts is available at the link below.

- <http://nmap.org/nsedoc/index.html>

- From the link above, select Scripts, and scroll down to see a list of the script starting with HTTP.



Take note!

In Kali, these scripts are installed at `/usr/share/nmap/scripts`

- Run the following commands against the target.

```
nmap 192.168.1.100 -p80,443 --script=http-apache-negotiation
```

```
nmap 192.168.1.100 -p80,443 --script=http-comments-displayer
```

```
nmap 192.168.1.100 -p80,443 --script=http-date
```

9.2 APPENDIX B - NIKTO SCANNING

Nikto is a Linux-based open-source web server scanner which can perform comprehensive tests against web servers.



The Nikto Homepage.

- <https://cirt.net/nikto2>

- Run the following command to show the Nikto plugins available. The output should look like the Figure below.

nikto -V

```
(kali㉿kali)-[~]  
$ nikto -V
```

File	Version
Nikto main	2.1.6
LibWhisker	2.5
db_404_strings	2.003
db_content_search	2.000
db_dir_traversal	1.0
db_dir_traversal	2.1.6
db_domino	2.1.6
db_drupal	1.00
db_embedded	2.004
db_favicon	2.010
db_headers	2.008
db_httptoptions	2.002
db_multiple_index	2.005
db_outdated	2.017
db_parked_strings	2.001
db_realms	2.002
db_server_msgs	2.006
db_tests	2.021
db_variables	2.004
nikto_apache_expect_xss.plugin	2.04
nikto_apacheusers.plugin	2.06
nikto_auth.plugin	2.04
nikto_cgi.plugin	2.06
nikto_clientaccesspolicy.plugin	1.00
nikto_content_search.plugin	2.05

- It is possible to run these plugins individually, however, let's just run them all with Nikto together. You can do that by inputting the following command.

nikto -h http://192.168.1.100

```
(kali㉿kali)-[~]  
$ nikto -h http://192.168.1.100  
- Nikto v2.1.6  
  
+ Target IP: 192.168.1.100  
+ Target Hostname: 192.168.1.100  
+ Target Port: 80  
+ Start Time: 2022-08-03 11:08:36 (GMT-4)
```

- The above Figure is cropped, as the output is often quite verbose. Now run the same command against the other web applications on that target system. Perhaps pipe the output to a new file.

nikto -h <http://192.168.1.100:8080> > port8080app

nikto -h <http://192.168.1.100:9080> > port9080app



Question.

Investigate the output created from these scans, what useful information is contained within?