IoT Research Competition Server

JR. PENTESTING COMPETITION

Submitted from

[redacted]

Team_015:

[redacted]

Introduction/ Executive Summary

Classification Definitions

Risk Classifications

Level	Score	Description
Critical	**10**	The vulnerability poses an immediate threat to the organization. Successful exploitation may permanently affect it, so remediation should be performed immediately.
High	**7-9**	The vulnerability poses an urgent threat to the organization, and remediation should be prioritized.
Medium	**4-6**	Successful exploitation is possible and may result in notable disruption of business functionality. This vulnerability should be remediated when feasible.
Low	**1-3**	The vulnerability poses a negligible threat to the organization. Its presence should be noted and remediated if possible.
Informational	**0**	These findings do not clearly threaten the organization, but they may cause business processes to function differently than desired or reveal sensitive information about the company.

Exploitation Likelihood Classifications

Likelihood	Description
Likely	Exploitation methods are well-known and can be performed using publicly available tools. Low-skilled attackers and automated tools could successfully exploit the vulnerability with minimal difficulty.
Possible	Exploitation methods are well-known. They may be performed using public tools, but require configuration. Understanding the underlying system is required for successful exploitation.

Unlikely	Exploitation requires a deep understanding of the underlying systems or advanced technical skills. Precise conditions may be required for successful exploitation.

Business Impact Classifications

Impact	Description
Major	Successful exploitation may result in large disruptions of critical business functions across the organization and significant financial damage.
Moderate	Successful exploitation may cause significant disruptions to non-critical business functions.
Minor	Successful exploitation may affect a few users without causing much disruption to routine business functions.

Challenges

HIGH RISK (8/10)

Exploitation Likelihood	**Possible**
Business Impact	**Severe**
Remediation Difficulty	**Easy**

Windows

1. Bonus: SMB enumeration

For this flag, I used recon to figure out which server had SNB ports open. We can achieve this by using the command.
nmap -sC -p –sV 192.168.105.5

After we confirmed that the server is running netbios-ssn and microsoft-ds, we can then run the command.
netexec smb 192.168.105.5 to see if the version is vulnerable. Then, we run the command.
smbclient -L 192.168.105.5 -m NT1 –N, which displays the list of shares for that server.

Figure 1: SMB Results

2. Bonus: Interns will pull an Intern

Following that, we can use the same server IP to connect to a share that we used to display the list of shares.

Using this command,
`smbclient //192.168.105.5/Sharedfolder -m NT1 -N`
Displaying

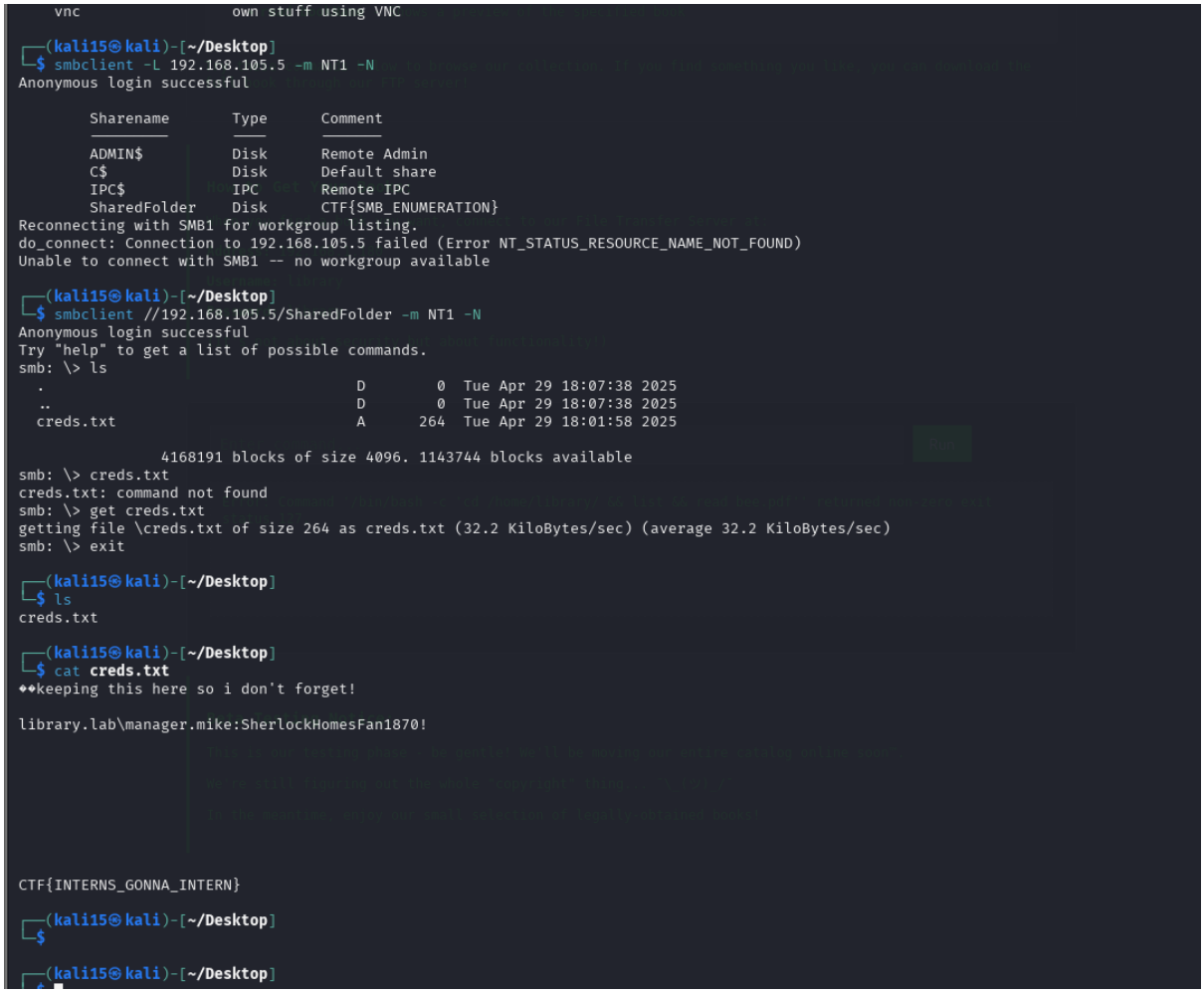
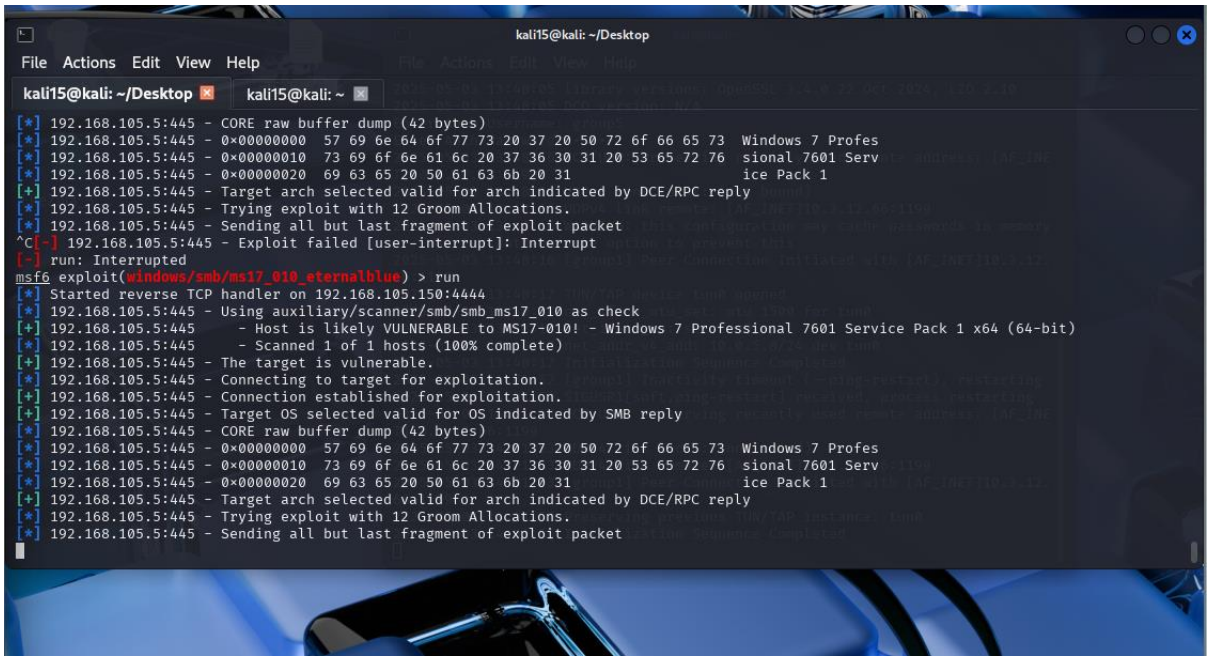


Figure 2: Interns' Results

Then, to get the file, we used the command get creds.txt, which allowed us to download the file onto our machine.
This can be a problem, as the Disk did not require a password to access the file. Making it easy for a malicious actor to download company-sensitive data onto their machines.

3. Intern Workstation

Like the other challenges, we first did rec-con on the server that we be attacking which was the Intern-server. We used Nmap to see what ports were open, we noticed that Windows 7 was running. So, we used this information to our advantage by running Metasploit to see if we can exploit using a known vulnerability called “Eternalblue”. This can allow the malicious actor to again access to the vulnerable machine.

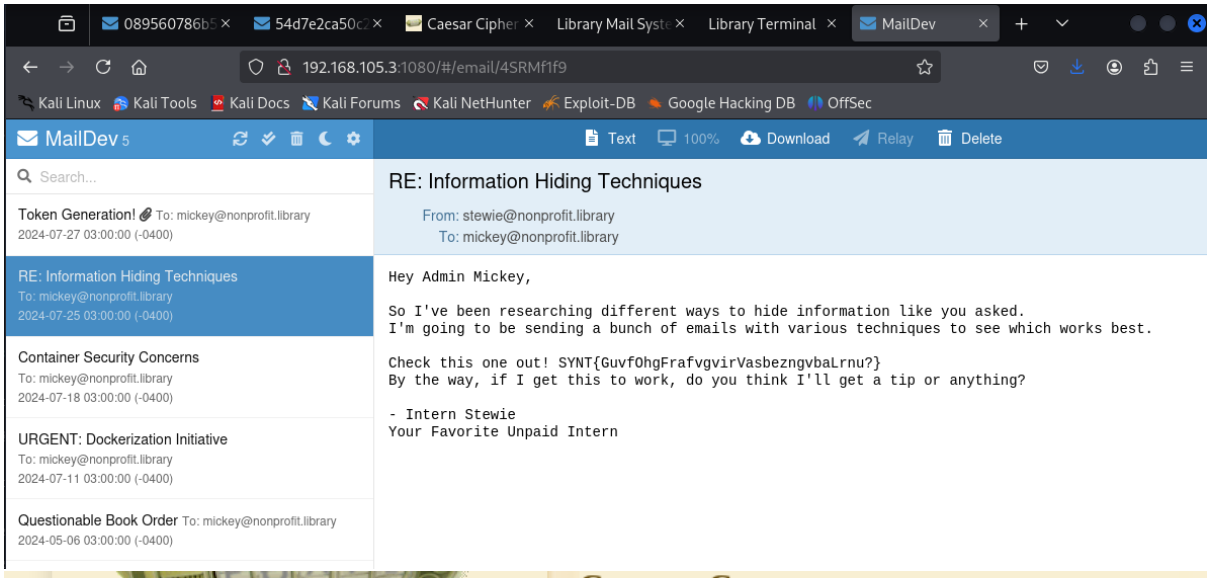


]

Linux

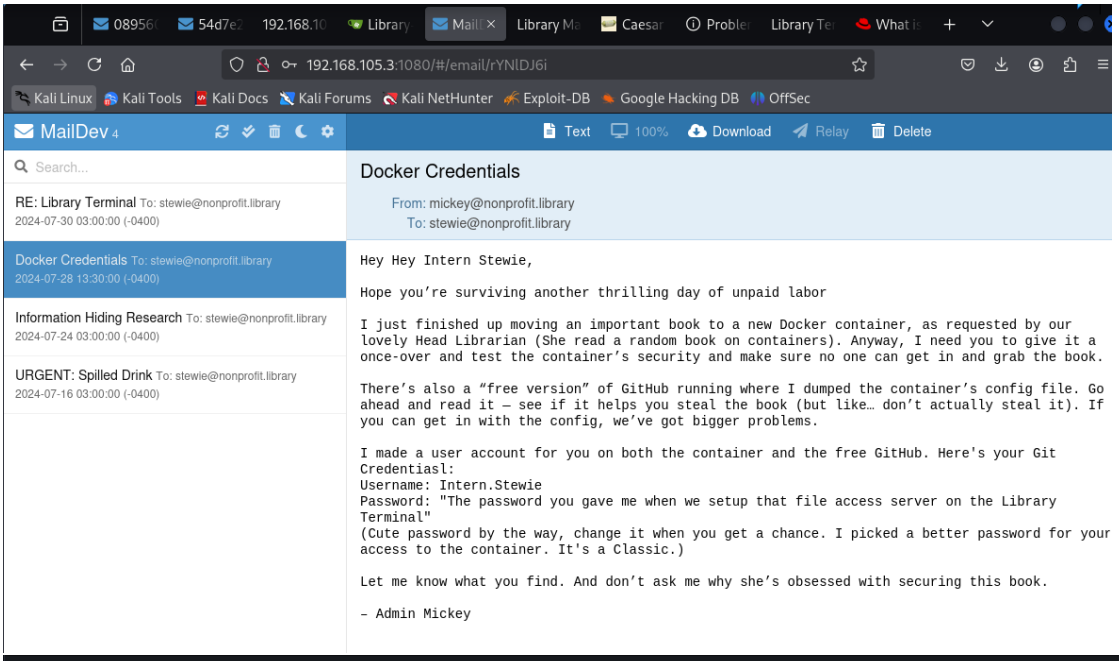
1. Bonus: Sensitive

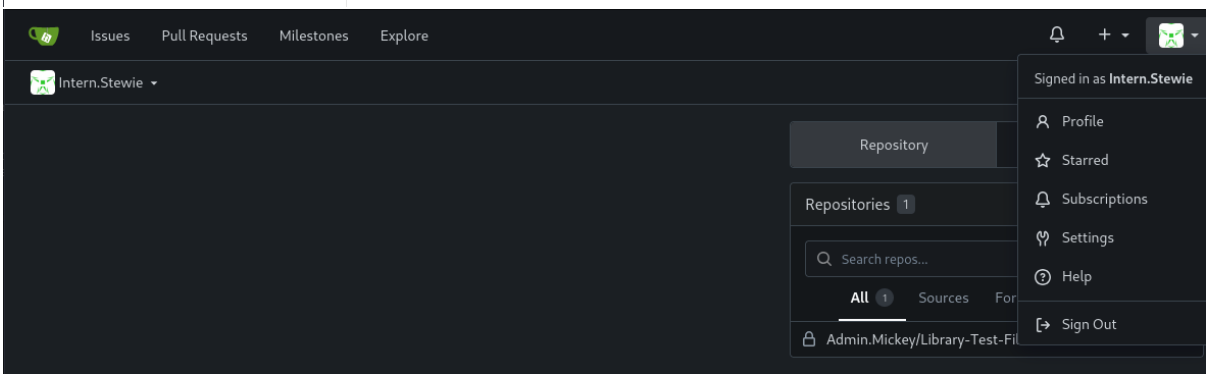
Using SQL injection technique, we could access the mail server and Stewie's account using their username. Anything after ' or 1=1 limit 1 -- will deem any input necessary to login invalid and not needed to access the service, where 1=1 means everything is true and the LIMIT clause will essentially limit how much output will be returned by the query and prevent further activity that would hinder performance. Looking through emails, we found a test flag under a reply email that used the Caesar Cipher technique! The technique practically converts each letter to a different fixed position in the alphabet and depends on how far the shift the user sets it to. For example, if the user sets the distance 4 letters away for each fixed position then 'A' would be changed to 'D'. Using a program that brute forces several instances of a cipher, one that uses 13 shifts, we were able to find the flag that had a human readable phrase.



2. Bonus: It's a Secure!
   For this bonus flag, we used the previous information gained from using John the Ripper and the library etc/shadow file to gain Intern Stewie's password that would be later used to get into the GitHub-esque (GITEA) program. As well as accessing Stewie's emails using the same SQL injection technique, we found Docker credentials with the username of "Intern.Stewie" and information hinting that the password is the one set

up during the creation of the FTP server with the Library Terminal. Using those credentials, we were able to access a file that contained the Docker creation code including a flag in the bottom as a comment.





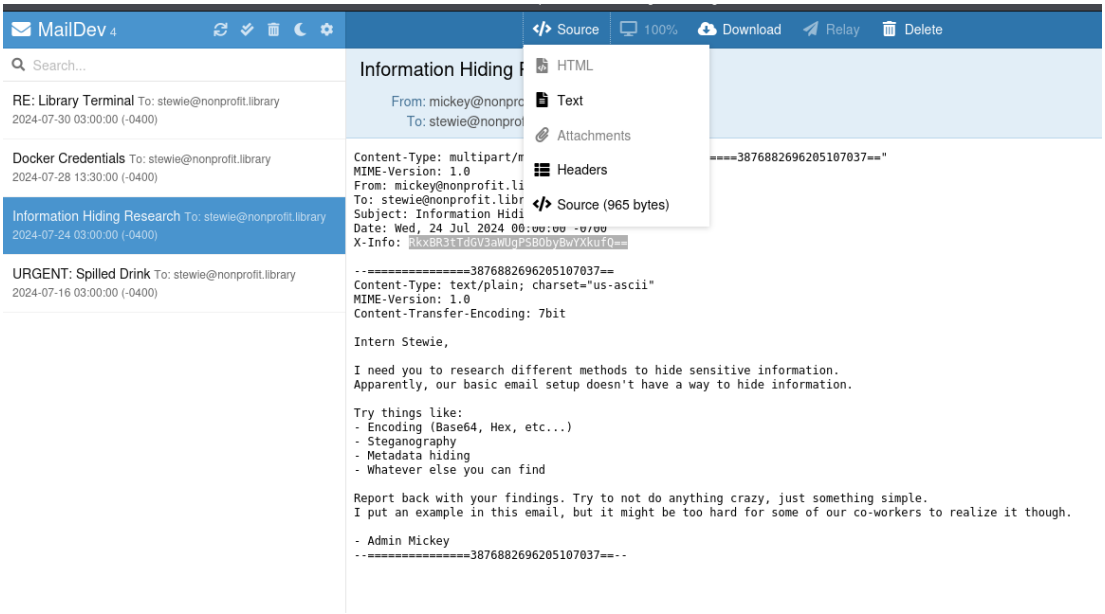
main / Library-Test-Files / Dockerfile

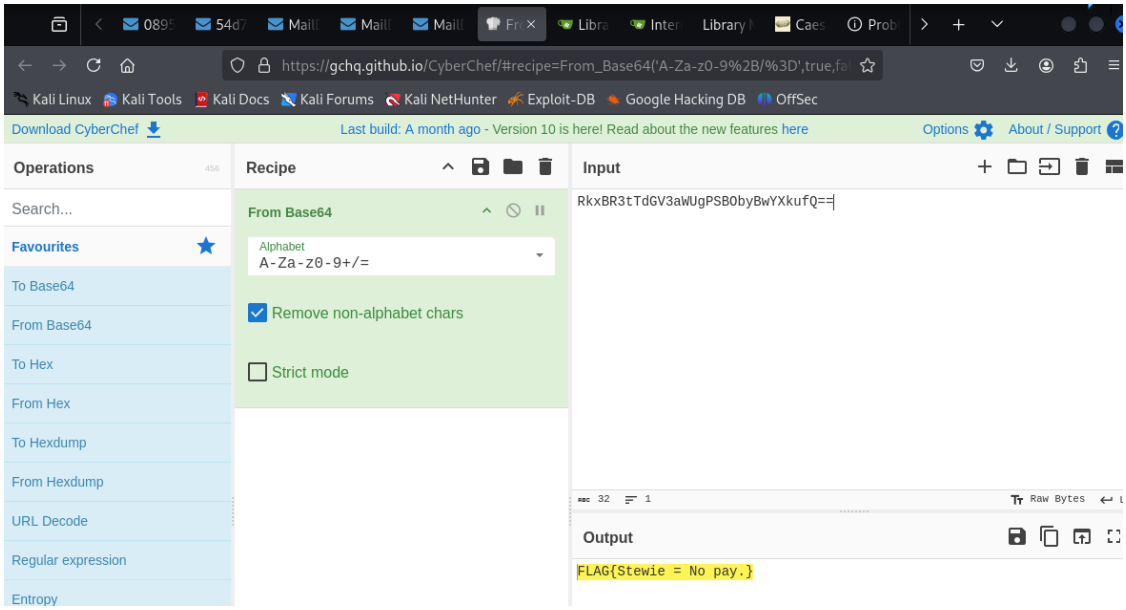
Admin.Mickey fc312748ef Update Dockerfile 4 days ago

27 lines | 712 B | Docker — Raw | Permalink | Blame | History

```
#Secure Vault Docker Container. Access with SSH :)
FROM debian:12

# Install base packages
RUN apt-get update && apt-get install -y \
    openssh-server \
    docker.io \
    && rm -rf /var/lib/apt/lists/*

# Enable SSH
RUN mkdir /var/run/sshd

# Create Stewie's Account
RUN useradd -m -s /bin/bash Intern.Stewie \
    && echo "Intern.Stewie:p@ssw0rd" | chpasswd \
    && usermod -aG docker Intern.Stewie
RUN echo 'root:ADDASECUREROOTPASSWORDHERE' | chpasswd

# Configure SSH
RUN echo "PermitRootLogin no" >> /etc/ssh/sshd_config \
    && echo "PasswordAuthentication yes" >> /etc/ssh/sshd_config

#Run SSH
CMD ["/usr/sbin/sshd","-D"]

# Note: Add this message to the container afterwards CTF{Docker_Is_Secure}
```

3. Bonus: No pay
   Continuing to snoop around Stewie's email, we discovered an email discussing Information Hiding Techniques including encoding and metadata hiding. These hints caused us to look at the source code of the email since that was an option. One line of source code contained content for X-Info that was a possible encoding for Base64. To test that theory, we used a Base64 encoder/decoder to reveal any sensitive information and found the flag.





4. Library Terminal

HIGH RISK (8/10)	
Exploitation Likelihood	**Possible**
Business Impact	**Severe**
Remediation Difficulty	**Easy**

Technical Description: The FTP server is vulnerable to exploitation through a SQL-like injection on the public facing Library Terminal allowing access to password hashes.

Business Impact: As the FTP server hosts the digital assets for the library, exploitation could lead to data loss or manipulation. This would affect customers who use this service and reduce trust in the library's services. Additionally, poor password management could affect other library services if administrator passwords are reused.

Affected Systems:

- FTP Server
- Library Terminal

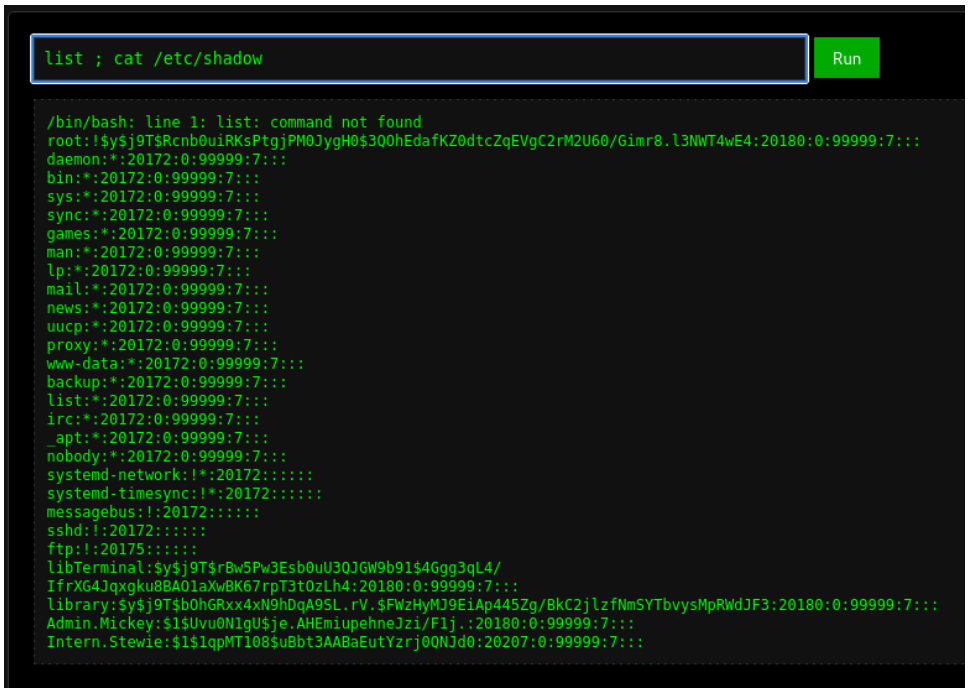
Remediation Steps:

- Secure Library Terminal interface to prevent SQL injections
- Secure password hash files and storing hashes in plaintext
- Implement a standard for stronger passwords to prevent hash hacking

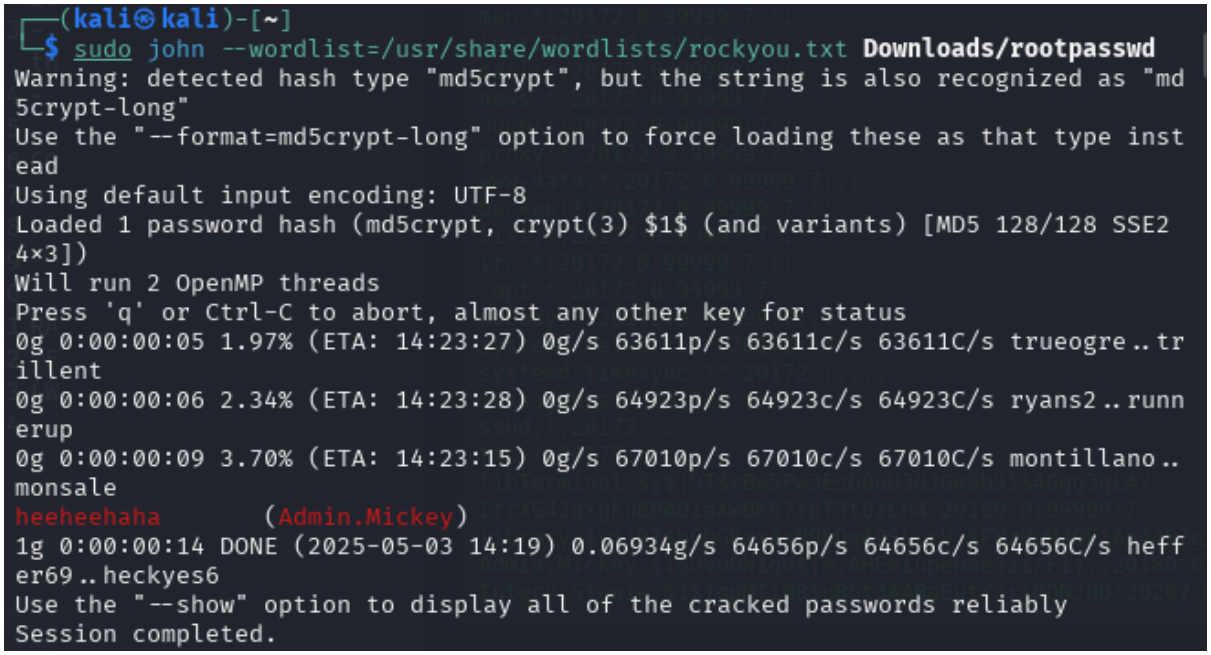
Steps for Reproduction:

1. Access the Library Terminal Interface at 192.168.105.2:5000

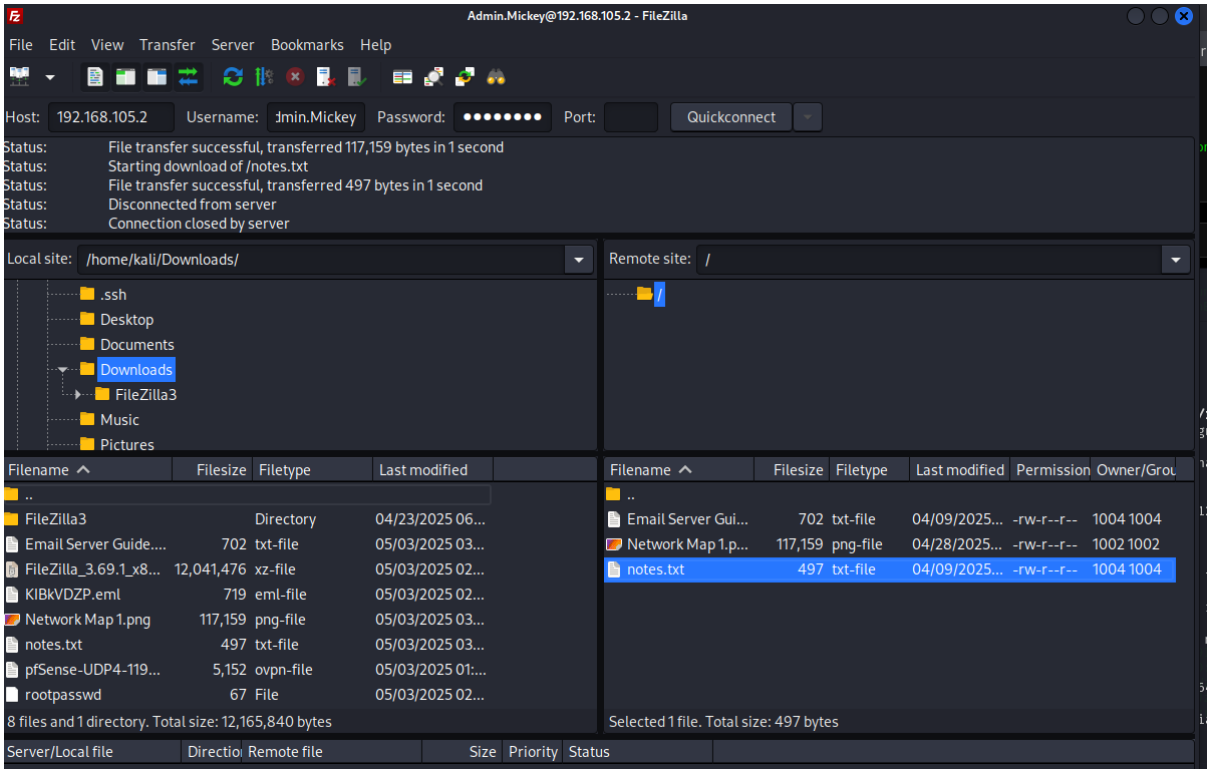
2. In the terminal, type `list ; cat /etc/shadow`



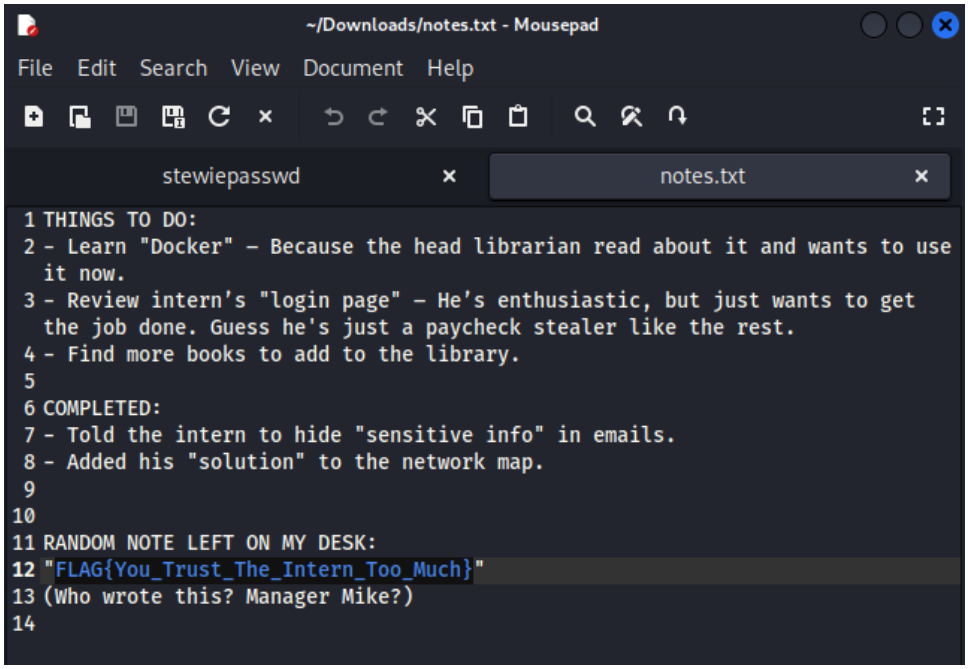
3. Copy admin hash to a text file.
4. Run John the Ripper script to find weak password hashes.



5. Access FTP server using admin credentials. Filezilla was used to access the FTP server.



6. File located after browsing through FTP server. Target file was notes.txt.



Conclusion

The IoT Research Competition Server serves as a fundamental element of the Junior PenTesting Competition, offering a dynamic and secure platform for participants to translate theoretical knowledge into practical, real-world applications. By emulating diverse IoT environments, the server not only enhances technical competencies but also fosters ethical hacking practices, critical thinking, and problem-solving skills that are indispensable for emerging cybersecurity professionals.

As IoT technologies continue to proliferate across various industries, competitions of this nature play a pivotal role in equipping the next generation of defenders with the necessary tools to safeguard interconnected systems. The server is meticulously designed to ensure that each participant acquires valuable, hands-on experience in identifying and mitigating risks, thereby underscoring the significance of proactive cybersecurity in an increasingly interconnected global landscape.

References/Appendix/Tools Used

- Reference Guides from Notion of Competition

Table of Figures