**Ophiuchi walkthrough**

# **Index**

[Index 1](#_Toc197635781)

[List of pictures 1](#_Toc197635782)

[Disclaimer 2](#_Toc197635783)

[Reconnaissance 2](#_Toc197635784)

[Initial foothold 2](#_Toc197635785)

[User flag 2](#_Toc197635786)

[Privilege escalation 5](#_Toc197635787)

[Personal comments 5](#_Toc197635788)

[References 6](#_Toc197635789)

# **List of pictures**

[Figure 1 - nMap scan results 2](#_Toc197635773)

[Figure 2 - Java exploit program 3](#_Toc197635774)

[Figure 3 - Yaml exploit input 3](#_Toc197635775)

[Figure 4 - First shell 3](#_Toc197635776)

[Figure 5 - Home folder for tomcat user 4](#_Toc197635777)

[Figure 6 - Credentials found 4](#_Toc197635778)

[Figure 7 - SSH connection as admin user 5](#_Toc197635779)

[Figure 8 - Info for privilege escalation 5](#_Toc197635780)

# **Disclaimer**

I do this box to learn things and challenge myself. I’m not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

# **Reconnaissance**

The results of an initial nMap scan are the following:

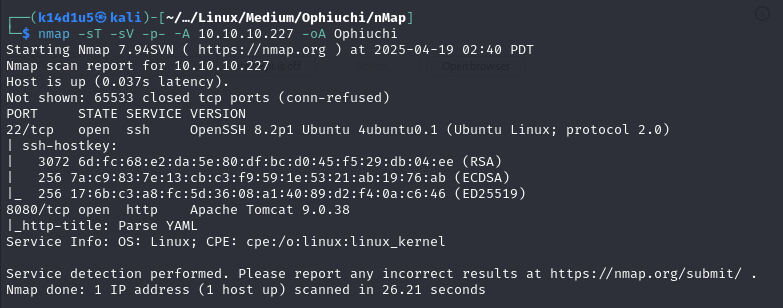


Figure 1 - nMap scan results

Open ports are 22 and 8080. Therefore, I found SSH service enabled and a web application running on port 8080. Also, nMap identified Linux (probably Ubuntu) as OS.

# **Initial foothold**

First of all, I browsed to the web application running on port 8080. There, I found a YAML parser. Therefore, I searched some exploit on the Internet and I luckily found one.

# **User flag**

I tried the exploit I found. Therefore, I modified the Java script so that I let the target to download a shell, give to it the execution permission and execute it to obtain a shell. At this point, I compiled it following the instructions and I inserted the yaml payload in the web application to let the target to download the .jar file I just generated. However, something didn’t work in my try. I spent a lot of time to try to understand why because I was pretty sure it was the correct way to follow to exploit the application. After a lot of time and several searches on the Internet I found out that I just needed to compile using an old version (11) of Java. Finally, at this point, I was able to obtain the first shell on the target. At the end, the payload Java script I used was the following:

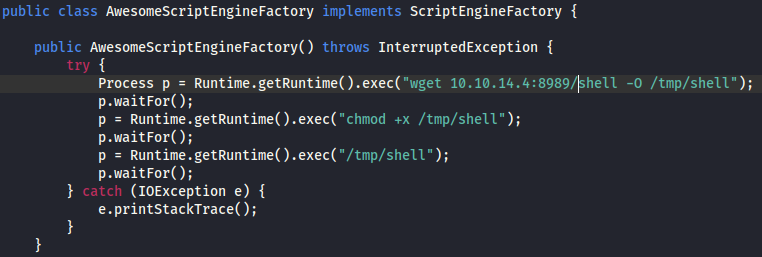


Figure 2 - Java exploit program

Also, the yaml payload I used in the web application was:



Figure 3 - Yaml exploit input

In this way, I obtained the first shell with user:

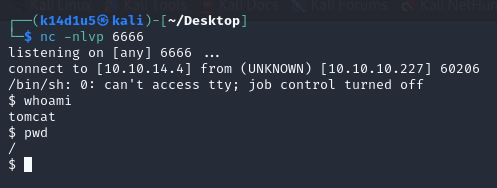


Figure 4 - First shell

However, I was not able to retrieve the user flag yet. Therefore, I needed to perform lateral movement to impersonate the user. To achieve this goal, I looked for some interesting information on the file system. One of the first places I looked for was the server root and or the home folder of the user I impersonate. I found where it was reading the file, as shown in the following figure:

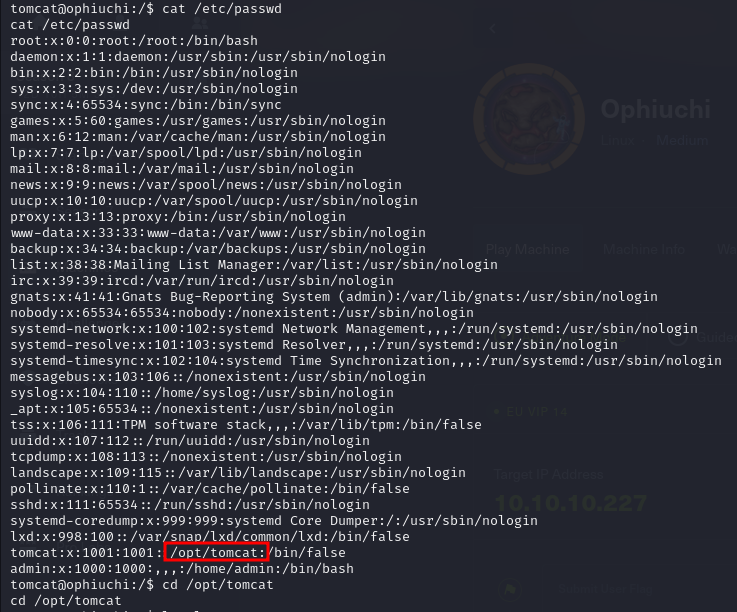


Figure 5 - Home folder for tomcat user

Luckily, in that folder I found the user credentials:

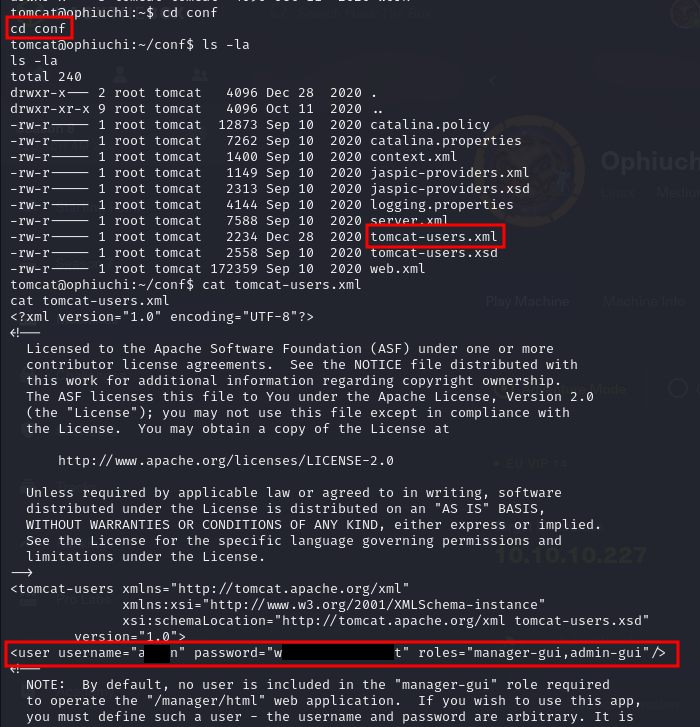


Figure 6 - Credentials found

Using these credentials, I was able to connect to the target via SSH as the user, as shown in the following figure:



Figure 7 - SSH connection as admin user

Even I forgot the screenshot, at this point I was able to retrieve the user flag in user home folder.

# **Privilege escalation**

As usual, one of the first check I did to find out how to escalate my privileges was to check what I can run as sudo. In this case, I was lucky because I can run a go script using sudo and no password:

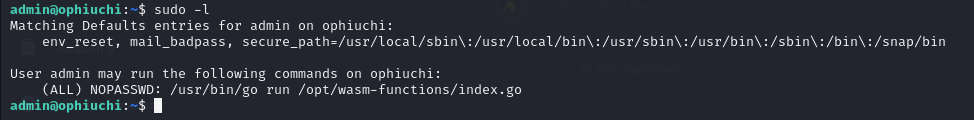


Figure 8 - Info for privilege escalation

I investigate the script and I found out that it run a script named if a specific condition was matched. Also, the script is referred with a relative path to refer to the current directory. To match the condition I named before, I needed to create a specific .wat file which contains an function that return the value . This file I need to convert in .wasm using tool. At this point I created the file which contain a reverse shell and I executed the GO script in the folder where I put the .wasm and deploy.sh files to obtain a root shell and retrieve the root flag. Again, I am sorry I forgot the screenshot to show the root shell and flag.

# **Personal comments**

I had some little issues with this box. The exploit didn’t initially work and I didn’t understand why (it was a different situation than the Java version). After I also find out that I needed an old Java version to compile the program exploit. Another hard point was that I needed to create the wasm file by the wat code. Initially, I though I was able to compile directly in wasm, but results were very different and the exploit didn’t work. In conclusion, I evaluate this box as medium.

# **References**

* Yaml deserialization exploit: <https://swapneildash.medium.com/snakeyaml-deserilization-exploited-b4a2c5ac0858>;
* Exploiting Yaml: <https://github.com/artsploit/yaml-payload>;
* Walkthrough where I find out I needed an old version of Java to compile: <https://medium.com/@aniketdas07770/hackthebox-ophiuchi-writeup-571796fc02df>;
* WAT language: <https://coderundebug.com/learn/wat/introduction/>.