BLOG (HTTPS://WWW.INCAPSULA.COM/BLOG/CATEGORY/BLOG), BOTS & DDOS (HTTPS://WWW.INCAPSULA.COM/BLOG/CATEGORY/BOTS-DDOS), SECURITY (HTTPS://WWW.INCAPSULA.COM/BLOG/CATEGORY/SECURITY) · OCTOBER 26, 2016

Breaking Down Mirai: An IoT DDoS Botnet Analysis

Ben Herzberg (https://www.incapsula.com/blog/author/ben-herzberg)

Dima Bekerman (https://www.incapsula.com/blog/author/dimab)

Igal Zeifman (https://www.incapsula.com/blog/author/igal)

By now many of you have heard that on September 20, 2016, the website of renowned security journalist Brian Krebs was hit with one of the largest distributed denial of service (https://www.incapsula.com/ddos/denial-of-service.html) attacks (DDoS) to date.

The magnitude of that attack, the star status of its target within the InfoSec community and the heaps of drama that followed (http://fortune.com/2016/09/27/google-krebs-project-shield-hack/) made this one of the most high-profile DDoS stories of the year.

On September 30, the story saw another development when a HackForum user by the name of 'Anna-senpai' leaked the source code for Mirai—the botnet (https://www.incapsula.com/ddos/botnet-ddos.html) malware behind the attacks. It was speculated that in doing so the perpetrator was trying to hide his tracks, rightfully concerned about the repercussions of taking a swing at Brian.

Since the source code was published, the Imperva Incapsula security team has been digging deep to see what surprises Mirai may hold. In this post we'll share:

- Our own encounters with Mirai botnets
- The results of our investigation of Mirai's source code

Update:

New Mirai scanner released: We developed a scanner that can check whether one or more devices on your network is infected by or vulnerable to Mirai. You can find the beta of the Mirai Scanner here (https://www.incapsula.com/mirai-scanner/).

If you missed out "Deep Dive into the Mirai Botnet" hosted by Ben Herzberg check out our video recording (http://embed.vidyard.com/share/F8ERekYaN8bor6ZQZKk9f3) of the event.

Close Encounters of the Third Kind

A thorough review of Mirai's source code allowed us to create a strong signature with which we could identify Mirai's activity on our network. We then turned to our logs and examined recent assaults to see if any of them carried Mirai's fingerprints.

Sure enough, we found the Mirai botnet was responsible for a slew of GRE floods that were mitigated by our service on August 17. Using a hit-and-run tactic, the attack peaked at 280 Gbps and 130 Mpps, both indicating a very powerful botnet.

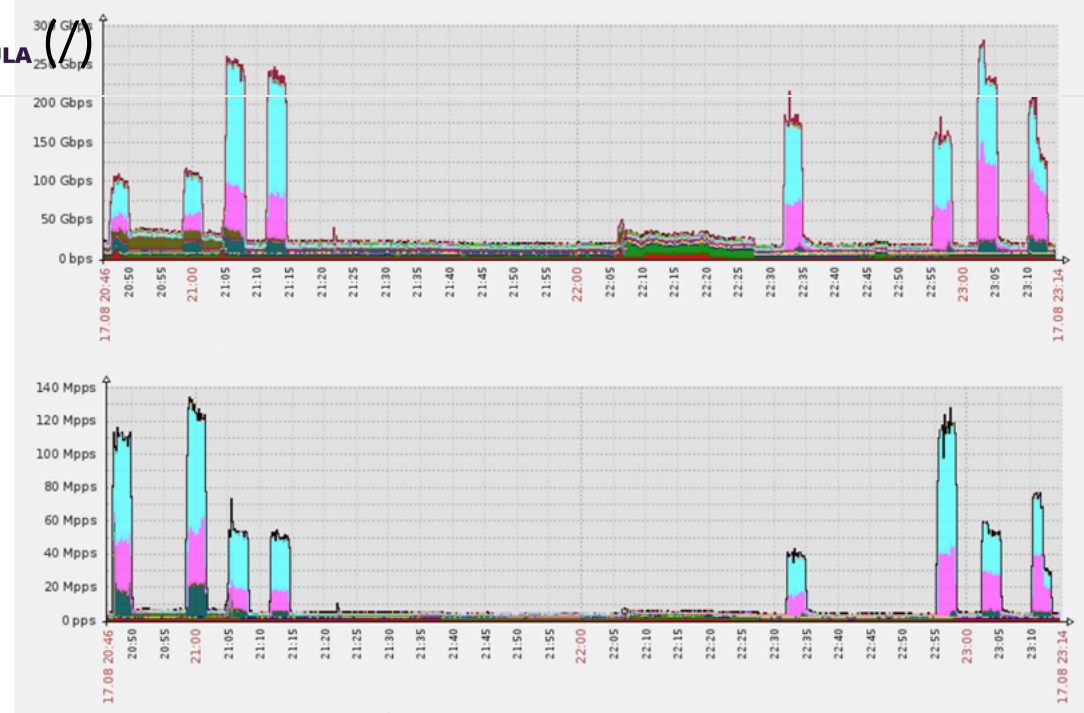

Figure 1: Mitigating a slew of Mirai-powered GRE floods, peaking at 280 Gbps/130 Mpps

Investigation of the attack uncovered 49,657 unique IPs which hosted Mirai-infected devices. As previously reported, these were mostly CCTV cameras—a popular choice (https://www.incapsula.com/blog/cctv-ddos-botnet-back-yard.html) of DDoS botnet herders. Other victimized devices included DVRs and routers.

Overall, IP addresses of Mirai-infected devices were spotted in 164 countries. As evidenced by the map below, the botnet IPs are highly dispersed, appearing even in such remote locations as Montenegro, Tajikistan and Somalia.

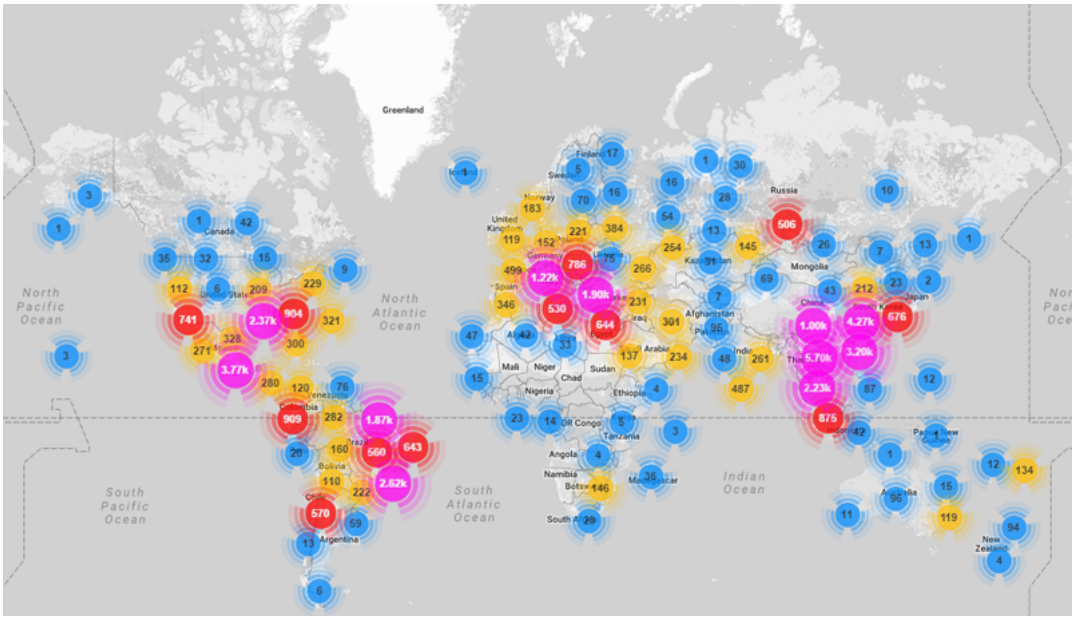

Figure 2: Geo-locations of all Mirai-infected devices uncovered so far

COUNTRY	% OF MIRAI BOTNET IPS
Vietnam	12.8%
Brazil	11.8%
United States	10.9%
China	8.8%
Mexico	8.4%
South Korea	6.2%
Taiwan	4.9%
Russia	4.0%
Romania	2.3%
Colombia	1.5%

Figure 3: Top countries of origin of Mirai DDoS attacks

Interestingly, since the source code was made public, we've also seen a few new Mirai-powered assaults. This time they took the form of low-volume application layer HTTP floods, one of which was even directed against our domain (www.incapsula.com).

Characterized by relative low requests per second (RPS) counts and small numbers of source IPs, these looked like the experimental first steps of new Mirai users who were testing the water after the malware became widely available. Likely, these are signs of things to come and we expect to deal with Mirai-powered attacks in the near future.

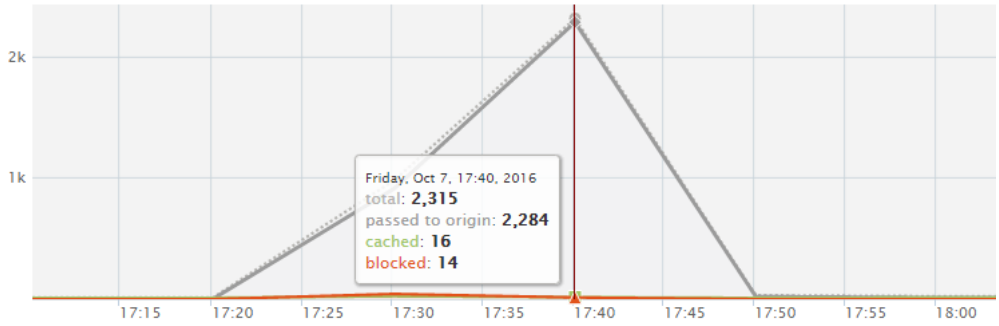

Figure 4: Mirai botnet launching a short-lived HTTP flood against incapsula.com

Source Code Analysis

Mirai (https://st.drweb.com/static/new-www/news/2016/september/Investigation_of_Linux.Mirai_Trojan_family_en.pdf)is a piece of malware that infects IoT devices and is used as a launch platform for DDoS attacks. Mirai's C&C (command and control) code is coded in Go, while its bots are coded in C.

Like most malware in this category, Mirai is built for two core purposes:

1. Locate and compromise IoT devices to further grow the botnet.
2. Launch DDoS attacks based on instructions received from a remote C&C.

To fulfill its recruitment function, Mirai performs wide-ranging scans of IP addresses. The purpose of these scans is to locate under-secured IoT devices that could be remotely accessed via easily guessable login credentials—usually factory default usernames and passwords (e.g., admin/admin).

Mirai uses a brute force technique for guessing passwords a.k.a. dictionary attacks (https://en.wikipedia.org/wiki/Dictionary_attack) based on the following list:

```
root     xc3511
root     vizxv
root     admin
admin    admin
root     888888
root     xmhdipc
root     default
root     juantech
root     123456
root     54321
support  support
```

Mirai's attack function enables it to launch HTTP floods and various network (OSI layer 3-4) DDoS attacks. When attacking HTTP floods, Mirai bots hide (https://www.incapsula.com/blog/was-that-really-a-google-bot-crawling-my-site.html) behind the following default user-agents:

```
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko) Version/9.1.2 Safari/601.
```

For network layer assaults, Mirai is capable of launching GRE IP and GRE ETH floods, as well as SYN and ACK floods, STOMP (Simple Text Oriented Message Protocol) floods, DNS floods and UDP flood attacks.

Mira also seems to possess some bypass capabilities, which allow it to circumvent security solutions:

```
#define TABLE_ATK_DOSARREST                    45  // "server: dosarrest"
#define TABLE_ATK_CLOUDFLARE_NGINX      46  // "server: cloudflare-nginx"

if (util_stristr(generic_memes, ret, table_retrieve_val(TABLE_ATK_CLOUDFLARE_NGINX, NULL)) != -1)
                        conn->protection_type = HTTP_PROT_CLOUDFLARE;

if (util_stristr(generic_memes, ret, table_retrieve_val(TABLE_ATK_DOSARREST, NULL)) != -1)
                        conn->protection_type = HTTP_PROT_DOSARREST;
```

While this may seem like a standard source code, Mirai also has a few quirks that we found especially intriguing…

Mirai's "Don't Mess With" List

One of the most interesting things revealed by the code was a hardcoded list of IPs Mirai bots are programmed to avoid when performing their IP scans.

This list, which you can find below, includes the US Postal Service, the Department of Defense, the Internet Assigned Numbers Authority (IANA) and IP ranges belonging to Hewlett-Packard and General Electric.

```
127.0.0.0/8               - Loopback
0.0.0.0/8                 - Invalid address space
3.0.0.0/8                 - General Electric (GE)
15.0.0.0/7                - Hewlett-Packard (HP)
56.0.0.0/8                - US Postal Service
10.0.0.0/8                - Internal network
192.168.0.0/16            - Internal network
172.16.0.0/14             - Internal network
100.64.0.0/10             - IANA NAT reserved
169.254.0.0/16            - IANA NAT reserved
198.18.0.0/15             - IANA Special use
224.*.*.*+                - Multicast
6.0.0.0/7                 - Department of Defense
11.0.0.0/8                - Department of Defense
21.0.0.0/8                - Department of Defense
22.0.0.0/8                - Department of Defense
26.0.0.0/8                - Department of Defense
28.0.0.0/7                - Department of Defense
30.0.0.0/8                - Department of Defense
33.0.0.0/8                - Department of Defense
55.0.0.0/8                - Department of Defense
214.0.0.0/7               - Department of Defense
```

This list is interesting, as it offers a glimpse into the psyche of the code's authors. On the one hand, it exposes concerns of drawing attention to their activities. A concern we find ironic, considering that this malware was eventually used in one of the most high-profile attacks to date.

On the other hand, the content list is fairly naïve—the sort of thing you would expect from someone who learned about cyber security from the popular media (or maybe from this Wiki page (https://en.wikipedia.org/wiki/List_of_assigned_/8_IPv4_address_blocks)), not a professional cyber criminal.

Together these paint a picture of a skilled, yet not particularly experienced, coder who might be a bit over his head. That is unless some IP ranges were cleared off the code before it was released.

A Territorial Predator

Another interesting thing about Mirai is its "territorial" nature. The malware holds several killer scripts meant to eradicate other worms and Trojans, as well as prohibiting remote connection attempts of the hijacked device.

For example, the following scripts close all processes that use SSH, Telnet and HTTP ports:


```
killer_kill_by_port(htons(23))  // Kill telnet service
killer_kill_by_port(htons(22))  // Kill SSH service
killer_kill_by_port(htons(80))  // Kill HTTP service
```

These locate/eradicate other botnet processes from memory, a technique known as memory scraping:

```
#DEFINE TABLE_MEM_QBOT            // REPORT %S:%S
#DEFINE TABLE_MEM_QBOT2           // HTTPFLOOD
#DEFINE TABLE_MEM_QBOT3           // LOLNOGTFO
#DEFINE TABLE_MEM_UPX             // \X58\X4D\X4E\X4E\X43\X50\X46\X22
#DEFINE TABLE_MEM_ZOLLARD         // ZOLLARD
```

And this function searches and destroys the Anime (https://evosec.eu/new-iot-malware/) malware—a "competing" piece of software, which is also used to compromise IoT devices:

```
searching for .anime process
       table_unlock_val(TABLE_KILLER_ANIME);
                // If path contains ".anime" kill.
                if (util_stristr(realpath, rp_len - 1, table_retrieve_val(TABLE_KILLER_ANIME, NULL)) != -1)
                {
                    unlink(realpath);
                    kill(pid, 9);
                }
 table_lock_val(TABLE_KILLER_ANIME);
```

The purpose of this aggressive behavior is to:

1. Help Mirai maximize the attack potential of the botnet devices.
2. Prevent similar removal attempts from other malware.

These offensive and defensive measures shine a light on the turf wars being waged by botnet herders—a step away from the multi-tenant botnets (https://www.incapsula.com/blog/botnet-landscape-social-graph-analysis.html) we previously encountered in our research. So much for honor among thieves.

From Russia with Love?

Lastly, it's worth noting that Mirai code holds traces of Russian-language strings despite its English C&C interface. Here, for instance, Russian is used to describe the "username" and "password" login fields:

```
// Get username
this.conn.SetDeadline(time.Now().Add(60 * time.Second))
this.conn.Write([]byte("\033[34;1mпользователь\033[33;3m: \033[0m"))
// Get password
this.conn.SetDeadline(time.Now().Add(60 * time.Second))
this.conn.Write([]byte("\033[34;1mпароль\033[33;3m: \033[0m"))
```

This opens the door for speculation about the code's origin, serving as a clue that Mirai was developed by Russian hackers or—at least—a group of hackers, some of whom were of Russian origin.

Other bits of code, which contain Rick Rolls' jokes next to Russian strings saying "я люблю куриные наггетсы" which translates to "I love chicken nuggets" provide yet more evidence of the Russian heritage of the code authors, as well as their age demographic.

What Can You Do to Prevent IoT Botnet from Spreading

While DDoS attacks from Mirai botnets can be mitigated, there's no way to avoid being targeted. However, as a device owner, there are things you can do to make the digital space safer for your fellow Internet citizens:

1. Stop using default/generic passwords.
2. Disable all remote (WAN) access to your devices. To verify that your device is not open to remote access, you can use this tool (http://www.yougetsignal.com/tools/open-ports/) to scan the following ports: SSH (22), Telnet (23) and HTTP/HTTPS (80/443).