

N

New Variants of Mirai and Analysis



Zhen Ling¹, Yiling Xu¹, Yier Jin², Cliff Zou³, and Xinwen Fu⁴

¹School of Computer Science and Engineering, Southeast University, Nanjing, China

²University of Florida, Gainesville, FL, USA

³University of Central Florida, Orlando, FL, USA

⁴University of Massachusetts Lowell, Lowell, MA, USA

Synonyms

[Botnet](#); [DDoS](#); [IoT](#); [Malware](#)

Definition

Mirai, which means “the future” in Japanese, is a self-propagating malware that infects vulnerable networked IoT devices so as to turn them into part of a botnet that can be leveraged to perform large-scale distributed denial-of-service (DDoS) attacks.

Historical Background

The Mirai botnet was first discovered in August 2016 (Mal 2016). It has been used in massive DDoS attacks, including an attack

on KrebsSecurity in September 2016 which exceeded 600 Gbps (Krebs 2016), an attack on OVH in September 2016 which exceeded 1 Tbps (Klaba 2016), and an attack on Dyn in October 2016 (Hilton 2016) resulting in the cripple of some well-known websites such as GitHub, Twitter, Reddit, Netflix, Airbnb, and many others (Williams 2016). At its peak, Mirai infected 600k vulnerable IoT devices (Antonakakis et al. 2017), whose IP addresses were spotted in 164 countries (Herzberg et al. 2016). Since the source code for Mirai was published on hackforums.net (Annaseni 2016), the techniques can be digged deeply and may be adapted in other malware projects. The original creators of Mirai (i.e., Paras Jha, Josiah White, and Dalton Norman) has pleaded guilty in December 2017 (Department of Justice 2017) and sentenced to probation in September 2018 (Department of Justice 2018).

Foundations

Overview

In this chapter, we first present our analysis of the released source code of the Mirai malware for its architecture, scanning, and propagation strategy (Antonakakis et al. 2017; Kambourakis et al. 2017; Ling et al. 2018). We then discuss why Mirai did not get attention in its propagation phase until it deployed the DDoS attack. Finally, we present a new variant of Mirai that can improve the propagation speed. The goal of the study is to understand the impact of Mirai as both a worm and botnet.

Mirai Botnet

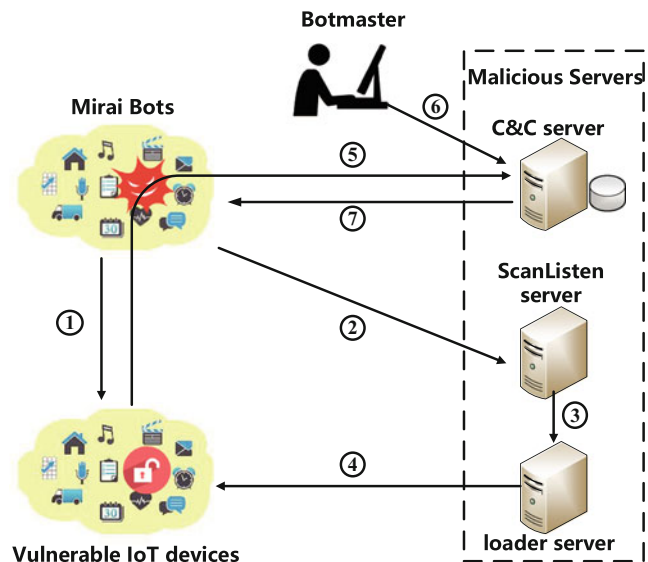
The Mirai botnet comprises four components as shown in Fig. 1: bots, a C&C (command and control) server, a scanListen server, and loader servers. The bots are a group of hijacked IoT devices via the Mirai malware. Starting with a scanning procedure on the port of the telnet service, the bots continue to perform a password dictionary attack using the username/password list already hardcoded in the Mirai malware. In this way, some vulnerable IoT devices are infected. The C&C server monitors the status of the botnet and controls the bots by obeying the commands from a botmaster, e.g., sending attack commands. On receiving the attack commands, the bots deploy various attacks such as distributed deny-of-service (DDoS) attacks. The scanListen server receives information of those newly discovered vulnerable IoT devices and relays it to the loader server. The loader server logs into a vulnerable device and infects it by executing the Mirai malware carried by the loader server. The IoT devices can be classified into three types: (1) IoT devices that close the telnet service port, i.e., 23/2323 port. They are not accessible to the bots since the TCP connections are cut off permanently; (2) IoT devices that open the telnet service port but their username/password is absent in the username/password list of Mirai. The bots

can perform the brute force attack against them but can never successfully compromise them; (3) IoT devices that open the telnet service port and their username/password is available in the list of Mirai. These devices can be compromised. Figure 1 illustrates the propagation procedure of Mirai. The step-by-step process is explained as follows.

- Step 1 – Bots scanning IoT devices:** The propagation starts with a single IP, which is treated as a bot to ensure its consistency. The bot scans and detects vulnerable IoT devices over the Internet. The bot selectively probes the 23/2323 port of public IP addresses through TCP SYN scans. Note that some IP addresses are whitelisted by Mirai, including those of the US Postal Service, the Department of Defense, the Internet Assigned Numbers Authority (IANA), and IP ranges belonging to Hewlett-Packard and General Electric. Once a TCP SYN/ACK packet is acquired from an IoT device, a TCP connection is to be established between the bot and the IP address of the device. In order to log into the telnet server, the bot randomly selects a pair of usernames/password from its dictionary and tries it out to check whether the login is successful. By default,

New Variants of Mirai and Analysis, Fig. 1

Propagation process of Mirai



the dictionary has altogether 62 pairs of usernames/password. However, the password cracking trials are no more than ten times for a target.

- **Step 2 – Bots reporting information to scan-Listen server:** After a bot compromises a vulnerable IoT device, it reports the information of the device to the scanListen server, including the IP address, the port, and the username/password of the target. The domain name and port of scanListen server are also hardcoded into the malware.
- **Step 3 – scanListen server relaying information to loader server:** The scanListen server receives the information of a compromised IoT device from the bot and then forwards it to the loader server.
- **Step 4 – Loader server installing Mirai:** The loader server first builds a TCP connection with the compromised device and then logs onto the device with the cracked username/password. After its login, the loader server starts detecting file transmission tools like *wget* or *tftp* on the target system. If a transmission tool is available, the Mirai malware is directly downloaded from the loader server and executed. Otherwise, the loader server connects to the telnet port of the target, reads a tiny binary file whose function is similar to that of *wget*, and writes *echo* into the target device. The hex characters from the binary file constitute the string for *echo*. Then the output stream of *echo* is redirected into a file and saved at the target device. Therefore, the binary file is transmitted from the loader server to the target. By executing the file, the Mirai malware can be downloaded from the loader server. Once the Mirai is executed, it shuts down Telnet, SSH, and HTTP server on the compromised device to avoid the login of other malwares or administrators. Mirai also binds to port 48101 for the assurance that only one instance of the malwares is running on this device.
- **Step 5 – Bot registering with C&C server:** Once the Mirai malware is installed and executed on a vulnerable device, the device builds a TCP connection, registers with the C&C

server, and gets involved in the Mirai botnet. The domain name and port of the C&C server are hardcoded into the malware. Then the bot performs Step 1.

- **Step 6 – Botmaster commanding bots to conduct attacks:** A botmaster can command the bots to perform attacks by sending instructions via the C&C server. The Mirai malware can be activated to launch various attacks like UDP DoS attack, TCP DoS attack, HTTP attack, and GREP attack.
- **Step 7 – Botnet attacking a target:** Once the C&C server receives the attacking instructions from the botmaster, it conveys the commands to the bots. Finally, the bots perform attacks against a target.

Scanning Strategy

The scanning algorithm of the released source code of the Mirai is worth studying since it is the key to its propagation speed. Therefore, we thoroughly analyze the Mirai scanning algorithm in Algorithm 1. It is discovered that Mirai malware uses a uniform scanning strategy. Specifically, Mirai randomly scans public IP addresses and then randomly selects a pair of usernames/password from a hardcoded list for the dictionary attack. The TCP SYN scanning technique is employed to probe the 23/2323 port of a device for the purpose of determining whether the port is open. As can be seen from the Mirai source code, 160 IP addresses are randomly selected as target for the bot to scan. The bot sends TCP SYN packets to those IP addresses by using a raw socket. Then the raw socket is checked so as to obtain SYN+ACK packets.

As long as the bot receives a SYN+ACK packet, it establishes a TCP connection with the IP address in the non-blocking mode and records the connection in a table. After checking the raw socket and recording the connections, the bot checks the status of all TCP connections in the table. If a TCP connection fails to be built within 5 s, the bot will give up the connection request.

Conversely, if the TCP connection to the telnet port of an IP is established successfully, the bot will select a pair of usernames/password

Algorithm 1: Mirai Scanning Algorithm

```

Require:
  (a)  $\mu$  - the number of SYN probes sent by Mirai each
  time,
  (b)  $m$  - the size of a table,
  (c) table - TCP connection table,
  (d) table[ $i$ ] - store the  $i$ th TCP socket in table,
  (e) table[ $i$ ].state - TCP connection state of the
   $i$ th entry of table, e.g., connecting or connected.
Ensure: Discover the vulnerable devices
1: while TRUE do
2:   if time interval since last SYN probe  $\geq 1s$  then
3:     Send  $\mu$  probes using a raw socket
4:   end if
5:   while TRUE do
6:     Read SYN+ACKs packets from the raw socket
7:     if error || no data for reading || no entries in
       table then
8:       break
9:     else if the  $i$ th entry of table is empty then
10:      Build a connection,
       table[ $i$ ].state = connecting
11:    end if
12:  end while
13:  for  $i = 1 : m$  do
14:    if the connection in table[ $i$ ] times out then
15:      if table[ $i$ ].state == connected then
16:        re-connect(try times < 10), otherwise
        free table[ $i$ ]
17:      else if table[ $i$ ].state == connecting
       then
18:        free table[ $i$ ]
19:      end if
20:      else if table[ $i$ ].state == connecting
       then
21:        Set the table[ $i$ ] in write file descriptor set
        (wr fdset)
22:        else if table[ $i$ ].state == connected
        then
23:          Set the table[ $i$ ] in read file descriptor set
        (rd fdset)
24:        end if
25:      end for
26:      Select the write and read file descriptors
27:      for  $i = 1 : m$  do
28:        if table[ $i$ ] in wr fdset then
29:          if connection error for table[ $i$ ] then
30:            free table[ $i$ ]
31:          else if connection success for table[ $i$ ]
           then
32:            table[ $i$ ].state = connected
33:          end if
34:        end if
35:        if table[ $i$ ] in rd fdset then
36:          while TRUE do
37:            if connection error for table[ $i$ ] then
38:              re-connect(try times < 10), otherwise
              free table[ $i$ ]
39:            break
40:            else if no data for reading then
41:              break
42:            else if data for the last command then
43:              if login success then
44:                report to scanListen server
45:              else if login fail then
46:                re-connect(try times < 10),
                otherwise free table[ $i$ ]
47:              end if
48:            else
49:              send one command
50:            end if
51:          end while
52:        end if
53:      end for
54:    end while

```

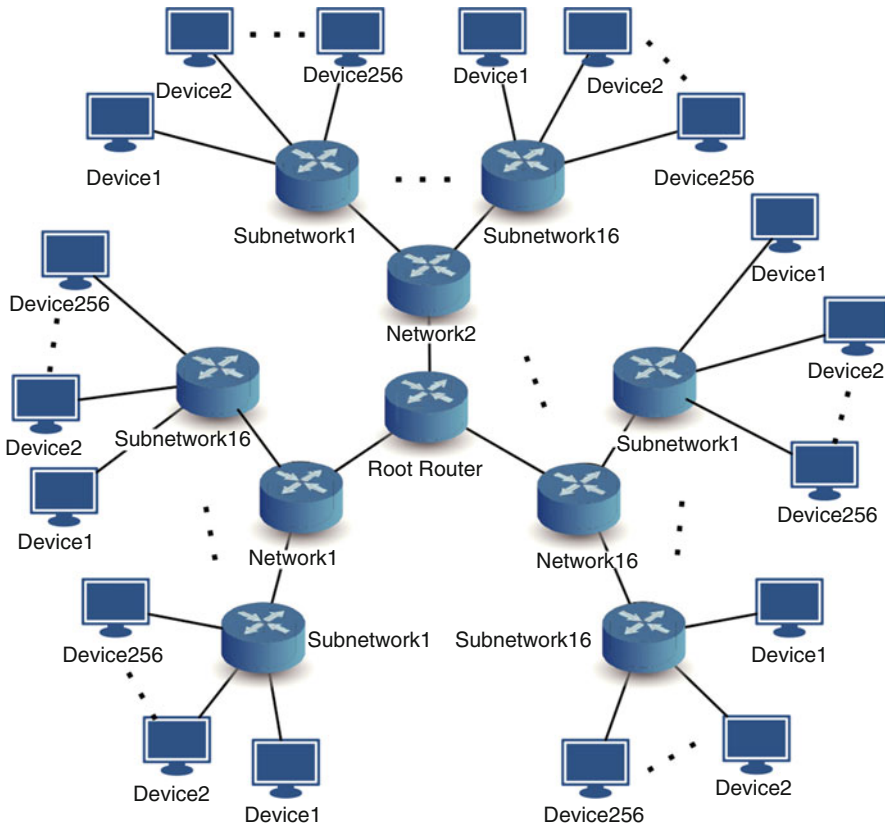
at random from the hardcoded list. Meanwhile, according to the feedback from the target device, the login can be verified as either a success or a failure. If the bot receives no response from the device in 30s, the login is deemed as a failure, and another trial goes on. The bot cuts off the unsuccessful connection and builds a new TCP connection to guess a random username/password again. However, such trials are only limited to ten times for Mirai to crack the username/password of a target IP.

Why Mirai Did Not Get Attention During Propagation

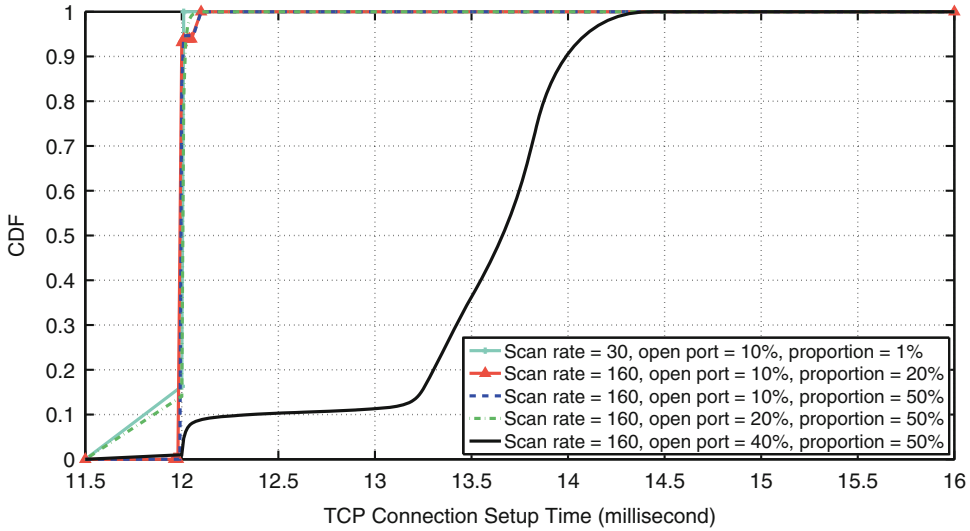
Since Mirai propagation did not cause the large-scale network congestion in the Internet in contrast to the propagation of Code Red worm (Zou et al. 2002), Mirai did not get attention during the propagation. To shed light on this phenomenon, we perform extensive simulations of Mirai propagation and analyze the time of TCP connection setup between a bot and a telnet port opened device to reflect the network state.

We simulate the original Mirai propagation using NS3. Recall that we divide the IP address space into three categories. We use empirical data to set the number of the devices in each category. Mirai grows to a peak of 600,000 infections at the end of November 2016 (Antonakakis et al. 2017), and the total number of IP addresses is 3,417,112,576 besides the IP addressed in the Mirai whitelists. Therefore, the proportion of vulnerable IoT devices is 0.0176%. Additionally, according to the results shown in ZoomEye (Zoo 2017), 11.1% online hosts open the telnet ports. To simplify the calculation, we round the proportions of the devices with open telnet ports and vulnerable telnet services to the nearest integer. Thus, in our simulation, we assume 10% of IoT devices in the IP space open the telnet port and 1% of the devices with the telnet service are vulnerable. Assume that the total number of devices in our simulated network is 65536.

By using Monte Carlo method, we select 6325 devices that open telnet port, and 64 devices can be compromised by Mirai malware. Due to the limitation of computing power of IoT devices, the empirical results from Figure 6 in Antonakakis et al. (2017) show that the scanning bandwidth of 80% bots is smaller than 2000 Bps. Since the size of a TCP SYN scanning packet is 74 bytes, most bots can only probe around 30 IP addresses to scan their telnet ports in each round, rather than 160 IP addresses used in the Mirai source code. Thus, we change the scanning rate to 30 in our simulation. Figure 2 illustrates the network topology used in the simulation experiment. We use the NS3 global routing by building a global routing database for the topology using a Dijkstra Shortest Path First (SPF) algorithm, and we set the throughput of devices and routers as 1000 Mbps. The TCP connection setup time between a bot and a telnet port opened device



New Variants of Mirai and Analysis, Fig. 2 Network topology used in the simulation experiment



New Variants of Mirai and Analysis, Fig. 3 TCP connection setup time distribution with different configurations

New Variants of Mirai and Analysis, Table 1 The proportions of devices with open telnet port and vulnerable telnet service

Devices with open telnet port	Devices with telnet service are vulnerable
10%	20%
10%	50%
20%	50%
40%	50%

is used to determine if the network is congested. Figure 3 illustrates that the Mirai does not cause network congestion by using empirical parameters in the simulation. Therefore, we can infer that the infected devices cannot generate a large amount of propagation traffic due to the limited computing power of vulnerable IoT devices.

Besides the limitation of IoT device computing power, another reason why Mirai does not cause network congestion is that the proportions of devices with open telnet port and vulnerable telnet service, respectively, are small. We assume the bot can probe 160 IP addresses in each round as the source code set and set two proportions in Table 1. The NS3 simulation results in Fig. 3 show that until we set 40% of IoT devices in the IP space with open telnet port and 50% of the devices with telnet service that are vulnerable, the

time of TCP connection setup becomes longer. At that point, the Mirai propagation has caused network congestion. Therefore, Mirai cannot cause network congestion by scanning and infecting a small number of devices with open telnet port and vulnerable telnet service.

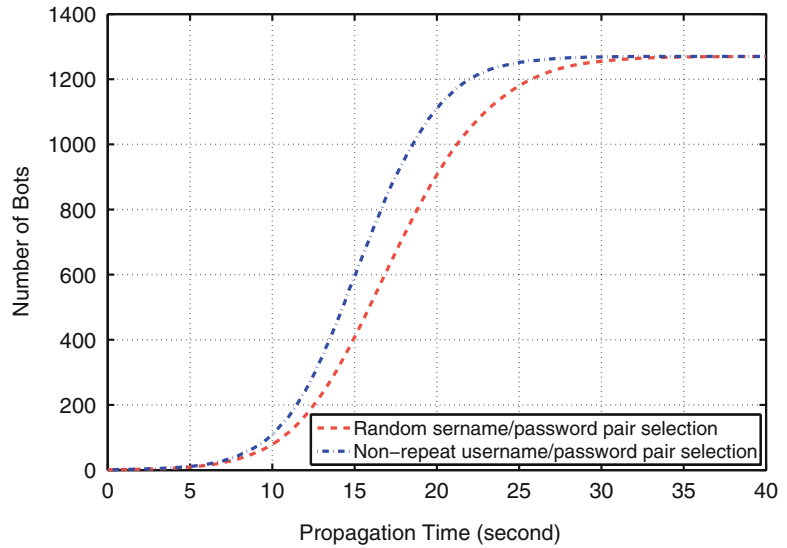
According to the real-world observation (Antonakakis et al. 2017), the proportion of devices with open telnet port and vulnerable telnet service are much lower than the 40% and 50%, respectively, and the limitation of IoT devices computing power cannot probe 160 IP addresses every round either. Therefore, Mirai did not cause network congestion and get attention during the propagation.

New Mirai Variants

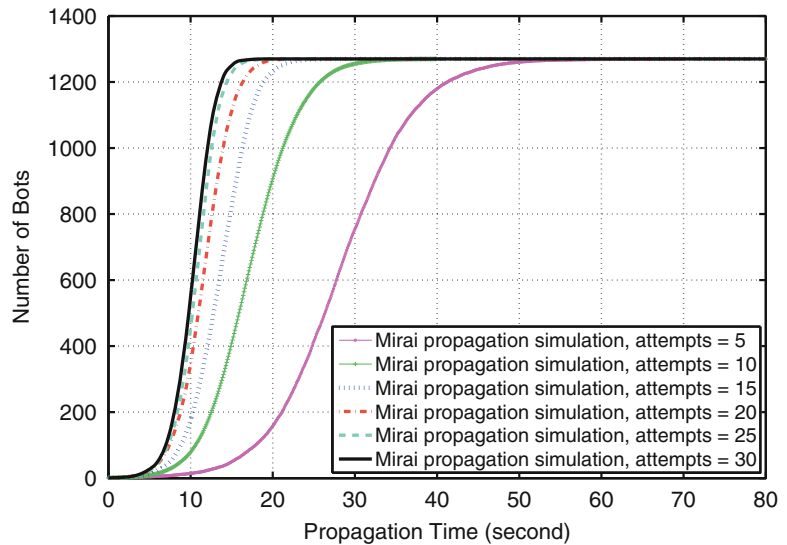
We design new scanning strategies in order to boost the propagation. In contrast to the malware such as the Code Red worm (Zou et al. 2002) that exploits software vulnerabilities, the Mirai malware not only scan the devices to discover the open telnet port but also crack the username/password using a hardcoded dictionary so as to successfully infect IoT devices. Therefore, the propagation speed of Mirai is much slower than that of the Code Red worm.

To optimize the scanning strategy, the probability that a bot successfully finds the right

New Variants of Mirai and Analysis, Fig. 4 The Mirai propagation speed using different cracking strategies



New Variants of Mirai and Analysis, Fig. 5 The Mirai propagation speed using different password cracking times



pair of username/password is the key parameter, we denote as q here. The original Mirai scanning strategy randomly selects a pair of username/password from a hardcoded list for the dictionary attack. Since there are 62 pairs of username/password in the dictionary and the bot tries at most 10 times on a target, we can derive $q = [1 - (61/62)^{10}]$. Our variant of Mirai does not try the same password/username pair more than once. If the bot tries at most ten times on a target as the original Mirai, q can be improved to $q = 10/62$. The simulation result in Fig. 4

shows that non-repeat username/password pair try strategy can improve the Mirai propagation speed.

The value of q varies as the username/password cracking times change. Based on the analysis of the original Mirai scanning strategy, we carry out Mirai propagation simulations on a scale of username/password cracking times. The cracking times are set at 5, 15, 20, 25, and 30, respectively. The simulation results are displayed in Fig. 5. It is observed that with the number of attempts increased (i.e., q increases), the rise of the

propagation speed ensues. Another observation is that as the number of attempts increases, the gain (acceleration) of the propagation speed falls. Thus we can devise a new variant of Mirai by keeping the username/password cracking times at 20 so as to fasten the Mirai propagation speed.

Future Directions

In this chapter, we study Mirai and propose new Mirai variants. We carefully introduce the propagation process of Mirai. Extensive NS3 simulations suggest that if the number of vulnerable devices is small, infected devices would not cause network congestion. This is why Mirai did not get attention during its propagation. To understand the potential impact of future Mirai attacks, we study new scanning strategies of Mirai that may adopt a non-repeating username/password cracking strategy and raise the number of cracking attempts so as to significantly boost the propagation speed. Our work in this chapter raises the alarm again for the IoT device manufacturers to better secure their products.

References

- Anna-senpai (2016) [FREE] world's largest net:Mirai botnet, client, echo loader, CNC source code release. <https://hackforums.net/showthread.php?tid=5420472>
- Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, Durumeric Z, Halderman JA, Invernizzi L, Kallitsis M, Kumar D, Lever C, Ma Z, Mason J, Menscher D, Seaman C, Sullivan N, Thomas K, Zhou Y (2017) Understanding the Mirai Botnet. In: Proceedings of the 26th USENIX security symposium (Security)
- Department of Justice (2017) Justice department announces charges and guilty pleas in three computer crime cases involving significant DDoS attacks. <https://www.justice.gov/opa/pr/justice-department-announces-charges-and-guilty-pleas-three-computer-crime-cases-involving>
- Department of Justice (2018) Hackers' cooperation with FBI leads to substantial assistance in other complex cybercrime investigations. <https://www.justice.gov/usa-ak/pr/hackers-cooperation-fbi-leads-substantial-assistance-other-complex-cybercrime>
- Herzberg B, Bekerman D, Zeifman I (2016) Breaking down Mirai: an IoT DDoS botnet analysis. <https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html>
- Hilton S (2016) Dyn analysis summary of Friday October 21 attack. <https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack>
- Kambourakis G, Koliass C, Stavrou A (2017) The Mirai botnet and the IoT zombie armies. In: Proceedings of the IEEE military communications conference (Milcom)
- Klaba O (2016) Octave klabla twitter. <https://twitter.com/olesovhcom/status/778830571677978624>
- Krebs B (2016) KrebsOnSecurity hit with record DDoS. <https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos>
- Ling Z, Liu K, Xu Y, Gao C, Jin Y, Zou C, Fu X, Zhao W (2018) IoT security: an end-to-end view and case study. <https://arxiv.org/pdf/1805.05853.pdf>
- Mal (2016) Linux/Mirai, how an old ELF malware is recycled. <http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html>
- Williams C (2016) Today the web was broken by countless hacked devices. https://www.theregister.co.uk/2016/10/21/dyn_dns_ddos_explained
- Zoo (2017) Zoomeye. <https://www.zoomeye.org/>
- Zou CC, Gong W, Towsley D (2002) Code red worm propagation modeling and analysis. In: Proceedings of the 9th ACM conference on computer and communication security (CCS)