The Mirai Botnet – How does it operate, what did it achieve, and how can IoT Botnets be mitigated?

Samuel Boyer

October 2019

Contents

1 Introduction

Distributed Denial-of-service (DDoS) attacks are the latest trend in cyber-warfare. With thousands - or millions - of zombie machines flooding an Internet server with requests, a large enough botnet can easily cripple the availability of a service. Christopher Wray, Director of the FBI, describes botnets as a 'Threat to the Homeland', stating attacks in recent years have "been responsible for billions of dollars in damages". (Wray, 2017) Attacks can have many motives - bragging rights between hacker groups, for paid contracts, or even for censorship by a government[1].

Botnets have been around since as early as 2004 (Mashevsky, 2005) and have amassed millions of zombie machines through various zero-day attacks, though recent years have offered a new source of network communication power: the Internet of Things (IoT). Many physical appliances are being redesigned for interaction with computer systems for various reasons, which is rapidly increasing the number of internet-connected devices (Intel, 2014). The boom in the IoT industry is encouraging start-ups and other businesses to quickly release new 'smart' devices, but this is resulting in many cases of under-developed and insecure software solutions within the embedded systems of the appliances.

Combining the need for maximum bandwidth in DDoS attacks with insecure internet-facing devices on millions of unique IP addresses explains why the security industry is so concerned about botnets. The first botnet system to exploit the IoT goldmine, Mirai, gained infamy in 2016 for disabling access to a large number of internet services, with almost effortless malware infection.

2 Constructing a botnet

Botnets are often grown through exploiting an oversight in an operating system. This is true of early IoT botnets such as Mirai, although the oversight is in the devices' configuration rather than a programming vulnerability. Many IoT devices' operating systems are very hastily built, and steps such as changing passwords and disabling unnecessary services are simply skipped. In some cases, this results in a Telnet service for remote administration being left open, oftentimes secured with default credentials. This makes it easy for a hacker (or infection server) to connect to the device over Telnet (on port TCP/23), attempt to authenticate with a small dictionary of default credentials, and - if successful - download and run a malicious binary.

Mirai isn't the first botnet malware to exploit weak credentials. Throughout 2014, the hacking group Lizard Squad built a botnet of home internet routers and some IoT devices through default Telnet logins, and amassed network bandwidth capable of DDoS attacks of up to 400Gbps. (Bing, 2016) It's unclear whether Mirai's greater success was due to a larger credential dictionary, or just

[1] It is often claimed that the Chinese government have launched DDoS attacks on services promoting free speech - such as the attack on Telegram in June 2019, coinciding with the Hong Kong protests.(Durov, 2019)

better timing with the IoT explosion, but the disruption Lizard Squad caused could have been inspiration for this botnet's development.

In September 2016, the author of the Mirai system released the source code on hackforums.net, seemingly to allow other hackers to create their own botnets with the same code[2]. (Krebs, 2016b) It was subsequently analysed by many security professionals and hackers.

The system is split into three subsystems: the Mirai malware, referred to in the source as 'Bot', which runs on vulnerable IoT devices; the Mirai Command and Control server ('CNC'), which distributes attack commands to the Bots; finally a second server, 'Loader', is used to aid in the spread of the malware.

It's worth noting that additional vulnerable devices are found by Bot instances, rather than a single infection machine, meaning the malware can also be considered a worm (self replicating computer program). This allows the malware to spread at an exponential rate, and explains how Mirai (according to the owner) reached a peak of 380,000 unique devices. (Krebs, 2016b) Once the bot is manually installed on one machine, it can begin to spread throughout the internet.

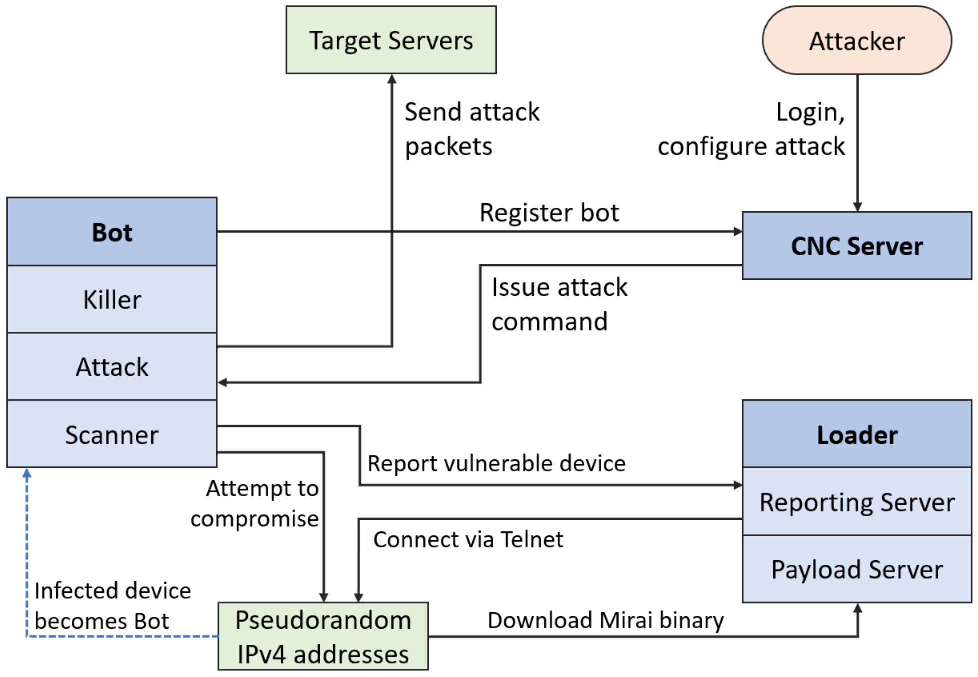

Figure 1: A flowchart describing the interactions between the components of Mirai, based on Hamdija Sinanović's analysis (Boyer, 2019)

[2]The original files were removed shortly after their release by the author; it is now hosted on GitHub by Jerry Gamblin (Gamblin, 2016) and most research points to this source.

2.1 The malware (Bot)

The Bot is a C program containing just over 5000 lines, separated into three modules: Attack, Killer and Scanner. These are started in separate threads once the initial setup is done.

As with most malware, a lot of effort is undertaken to conceal the program's presence. For instance, the program uses `util_strcpy` calls to replace its process name and arguments with random characters, so references to the malware are removed from the process table.

In particular, one of the earliest operations of the Bot (found in main.c) is to delete its binary (executable) file, meaning the program only resides in memory. Removing the original executable is a common malware feature, but the program is usually moved elsewhere first (such DLL Injection on Windows machines). This isn't done in Mirai, but it's also not necessary, since its targets are embedded systems that aren't often switched off. This prediction is reinforced later in `main()`, where the linux watchdog is disabled, in order to prevent system reboots.

Before registering the device with the CNC server, it checks if another instance of Mirai is running on this machine. There's no incentive to have two bots running on the same machine, as a single instance will already use as much network bandwidth that the machine has available, and multiple instances will conflict when opening sockets for the CNC server. Since the process names are now randomised, it detects another instance by checking port 48101; if another process is claiming this port, it kills that process.

The Killer module handles some unusual behaviour for malware; it actually improves the security of the device by closing ports 22 (SSH), 23 (Telnet) and 80 (HTTP)[3]. Furthermore, it then continually scans the processes listed in `/proc` to detect running instances of other popular IoT malware (such as Anime, Qbot and Zollard), by scanning the process' memory for certain fingerprints. If it finds a match, the process is killed. Such behaviour is likely done both to fully monopolize the individual device's network bandwidth, and in attempt to weaken the rival botnets on a global scale.

2.2 The Command and Control (CNC) Server

The Command and Control server is connected to by both the attacker and the Bots, and coordinates the DDoS attacks. The CNC opens a Telnet service (on TCP/23) for user login and bot reporting, and an API service on TCP/101 for automated attack requests. It connects to a MySQL server to load and store user credentials and a log of attacks.

This may seem like an over-engineered solution to simply tell the bots what to do, but this allows the botnet administrator to sell portions of the botnet to their clients, giving them each a login to coordinate the attack themselves. Clients are allocated a maximum number of bots they can use, likely based on

[3]This is done by killing the process currently using a port, then itself binding to the port so a restarted process can't reopen its port.

how much they've paid. Once logged in, clients stage an attack, specifying how many bots they wish to participate and how long the attack should last for. The CNC then connects to the required number of bots (stored in a FIFO queue, so multiple attacks can occur at once) and sends over the attack details.

Since CNC servers can be seen as a single point of failure for botnets, much effort is taken by the security industry to disable these servers, through 'take-down' requests to the hosting provider. To combat this, the bots first use a DNS query to find the IP address of the current CNC server, so it can move to a new IP as often as necessary [4].

2.3 Attack methods

When an attack instruction is received from the CNC server, the bot begins its attack in the Attack module. The malware has ten different attack methods, which are offered to clients logged into the CNC:

```
Available attack list
udp: UDP flood
dns: DNS resolver flood using the targets domain, input IP is ignored
greip: GRE IP flood
stomp: TCP stomp flood
greeth: GRE Ethernet flood
udpplain: UDP flood with less options. optimized for higher PPS
http: HTTP flood
vse: Valve source engine specific flood
syn: SYN flood
ack: ACK flood
```

Listing 1: Attack methods and descriptions, as printed by the CNC and defined in cnc/attack.go

Each of these attack methods attempt to flood the target machine with a high volume of requests, but using different packets to target different services. The simple methods, such as UDP, SYN and ACK floods affect lower-levels of the OSI model; their small packet sizes mean more can be sent per second, but also means the recipient can process and discard these packets faster. The larger packet floods, such as DNS and HTTP, will reach higher levels of the OSI model, meaning their lower transmission rate is compensated with greater memory consumption.

The remaining methods employ lesser-used protocols such as GRE (Generic Routing Encapsulation, a tunneling protocol) and TCP STOMP (Simple Text Oriented Messaging Protocol), and floods the server with packets from these respective protocols. Mirai appears to be the first malware to employ such methods, and Dan Breslaw of Imperva suggests they're used to target "an architectural soft spot in hybrid mitigation deployments". imperva2016stomp

[4]Some modifications of Mirai utilize the Tor network and host the CNC as a hidden service (`.onion` domain), to keep its IP address secret.

2.4 Self-replication and the Loader

The third module in the Mirai Bot, 'Scanner', handles part of the system's self-replication. The Bot generates a random IPv4 address and attempts to connect to it via Telnet (on TCP/23). It firstly checks if the port is open at all by sending a SYN packet and awaiting a SYN-ACK response. If a Telnet service is found, it then begins its login brute-force with a hard-coded dictionary of 62 username/password pairs. If a successful login is performed, the connection is closed, and the vulnerable device is reported to the Loader server, along with the correct login to use. When the Loader learns of a new vulnerable device, it connects again via Telnet, downloads the correct binary for this architecture via TFTP (Trivial File Transfer Protocol; a lightweight variant of FTP), executes it and disconnects.

```c
do
{
    tmp = rand_next();

    o1 = tmp & 0xff;
    o2 = (tmp >> 8) & 0xff;
    o3 = (tmp >> 16) & 0xff;
    o4 = (tmp >> 24) & 0xff;
}
while (o1 == 127 ||                             // 127.0.0.0/8      - Loopback
      (o1 == 0) ||                              // 0.0.0.0/8        - Invalid address space
      (o1 == 3) ||                              // 3.0.0.0/8        - General Electric Company
      (o1 == 15 || o1 == 16) ||                 // 15.0.0.0/7       - Hewlett-Packard Company
      (o1 == 56) ||                             // 56.0.0.0/8       - US Postal Service
      (o1 == 10) ||                             // 10.0.0.0/8       - Internal network
      (o1 == 192 && o2 == 168) ||               // 192.168.0.0/16   - Internal network
      (o1 == 172 && o2 >= 16 && o2 < 32) ||     // 172.16.0.0/14    - Internal network
      (o1 == 100 && o2 >= 64 && o2 < 127) ||    // 100.64.0.0/10    - IANA NAT reserved
      (o1 == 169 && o2 > 254) ||                // 169.254.0.0/16   - IANA NAT reserved
      (o1 == 198 && o2 >= 18 && o2 < 20) ||     // 198.18.0.0/15    - IANA Special use
      (o1 >= 224) ||                            // 224.*.*.*+       - Multicast
      (o1 == 6 || o1 == 7 || o1 == 11 || o1 == 21 || o1 == 22 || o1 == 26 || o1 == 28 || o1 ==
    29 || o1 == 30 || o1 == 33 || o1 == 55 || o1 == 214 || o1 == 215) // Department of Defense
);
```

Listing 2: An excerpt of mirai/bot/scanner.c

Above is the method that generates a random IPv4 address for checking. An interesting quirk of the scanner is its deliberate avoidance of some IPv4 blocks, particularly those of USPS and the US Department of Defence. It's likely this was done to evade detection and analysis by these groups, allowing the botnet to remain undiscovered for longer. In contrast to this stealth feature, the program contains no Anti-VM techniques. This is a common occurrence in modern malware where the program can stop operating maliciously if it believes it's under dynamic analysis in a virtual machine. (Assor, 2016)

3 Mirai's impact on Web services

Since portions of the botnet were rented to various clients, it's unclear exactly how many attacks were conducted with the original Mirai botnet throughout its operation, though the major attacks are well-documented.

On the 20th September 2016, Brian Krebs' security research blog was hit with a DDoS attack, believed to be the first major Mirai attack. (Krebs, 2016a) The attack lasted around four days, and peaked at 620Gbps, almost double the size of largest attack his hosting provider had previously seen. The attack appeared to comprise of a combination of SYN, HTTP and GRE floods, the latter of which being a major clue that Mirai was used. It's suspected this attack was done in retaliation for revealing the identities of two major botnet authors a week prior - which Krebs would go on to repeat for Mirai's author, Paras Jha. (Krebs, 2017)

At around the same time (from the 19th September onward), the french hosting provider OVH was targeted by the Mirai botnet. The attack continued with varying intensity until as late as the 29th, with bandwidth that peaked at 1.1Tbps, a unprecedented bandwidth for the time. (Goodin, 2016)

The final major attack targeted Dyn, a major DNS provider in the US and some of Europe. Throughout the day on the 21st October, Dyn's DNS servers were intermittently forced offline by a Mirai botnet[5]. (Hilton, 2016) This resulted in many popular websites being inaccessible via their web address, such as Amazon, Netflix and Twitter, among many others. The peak size of this attack wasn't initially disclosed, but was later revealed to be around 1.2Tbps, even greater than the OVH attack and raising the record further.

4 Mitigation

In order to prevent IoT botnets from growing even larger, these devices must be better-secured. However, as long as there's *any* devices with vulnerabilities, botnets will always exist in some form, so it's equally important to protect web services from DoS attacks.

4.1 Protecting IoT devices

An obvious first step would be to encourage IoT device manufacturers to have better security as default. This includes changing default passwords and removing remote admin services unless absolutely necessary. If a more secure protocol were used, such as ssh, authentication keys could be used instead of passwords, so the chances of reverse engineering a device and recovering a password is minimized.

Even if security is taken more seriously in future device iterations, this will still leave millions of insecure devices vulnerable to attack. Some have speculated and developed solutions to 'sanitize' these devices through the use of Anti-worms; these infect devices in a similar manner to regular malware, but instead intend to help the owner by patching the device and improving its overall security. Research by Michele De Donno *et al.* show that such anti-worms are already in the wild. (De Donno et al., 2017)

[5]Although Mirai's source code was released on the 30th September, allowing this attack to be performed by a new botnet administrator, it was later revealed to be executed by Jha.

This solution, however, has its issues. Releasing any worm into the wild has a risk of creating unexpected issues in the devices, which could turn the program into a threat itself. A notable example of this is the Welchia worm from 2003, which attempted to protect systems from the MSBlaster worm by forcing a system update. This created great disruption through automatic restarts and massive ICMP network traffic, as Gene Bransfield describes in (Bransfield, 2003).

There's also the legal and ethical implications of tampering with others' devices without consent. It's unclear how severely such an anti-worm would breach Computer Misuse laws - it'd likely break Section 1 of the UK's Computer Misuse Act 1990 ('Unauthorised access to computer material'), but Section 3 was changed in 2007 to read 'Unauthorised acts with intent to impair, or with recklessness as to impairing, operation of computer, etc.', which seems less cut-and-dry about such an intrusion[6] (*Computer Misuse Act 1990* 2015). Though as De Donno alludes, some may consider it equally unlawful of device owners to be hosting a vulnerable device:

> Nevertheless, we can not ignore that, accordingly to various legislations, also the very action of failing to protect your own device and unwillingly participating to a malicious action could be considered illegal.

Clearly the value - and legality - of such a solution would be the subject for much debate.

4.2 Protecting servers

The first step in protecting web services is to install a reverse proxy before the web server, to which all requests are made and forwarded to the server. This allows for the introduction of load balancing and rate limiting of requests, both of which can help to tackle DDoS attacks, as long as the proxy itself can handle the incoming traffic.

To mitigate against higher-level attacks, a Web Application Firewall (WAF) can sometimes be used on the proxy to identify deliberate attacks and block these requests. WAFs analyse the request headers and usually check for injection attempts, but can also detect strings that would trigger an application-specific DoS. Furthermore, some service providers such as Cloudflare and FortiGuard offer IP Repuation services. A large list of IP addresses and blocks known to have participated in malicious activity is used to categorise requests, allowing those from safe IPs to pass through, and those from suspicious IPs to be better scrutinised or dropped altogether.

Even if such mitigation tactics are used, the firewall servers must still be able to handle the amount of bandwidth we've seen in recent DDoS attacks (reaching over 1Tbps); commercial Ethernet connections are only beginning to

[6]The original Section 3 was titled 'Unauthorised modification of computer material', which doesn't appear to distinguish malicious modification from 'helpful' modification.

reach 100Gbps (Lustig, 2017), so the only current solution appears to be a distributed collection of data centres to diffuse the attack.

5 The future of botnets and DDoS

It's unlikely that the excitement around the IoT industry will dissipate in the coming years, so I don't believe device manufacturers will alter their priorities to better their security practises. To quote Nermin Hajdarbegovic in 2015 (before the Mirai attacks even began), "In the rush to bring new products and services to market, many companies are likely to overlook long-term support". (Hajdarbegovic, 2015) He also believes the low-power nature of the devices means that encryption and standard security measures can't be implemented without a CPU upgrade, which would obviously upset entrepreneurs.

The 'success' of the original Mirai botnet and the release of its source code inspired many hackers to build botnets of their own. This, combined with the increased adoption of IoT devices means botnet threats - and exploit traffic - are only growing, as shown by F-Secure's honeypots detecting 760 million Telnet requests in H1 2019. (Michael, 2019) Furthermore, many hackers are producing variants of the original Mirai malware, featuring more attack mechanisms to target a greater range of IoT devices. Some variants have also been found to feature web proxies and cryptocurrency miners, more akin to traditional botnets, though the focus is likely still on DDoS, an equally lucrative service. (Joven and Yang, 2018)

5.1 Amplification attacks

As well as an attacker's botnet, recent DDoS attacks have increasingly made use of other servers to amplify their attacks, through various UDP services. Since UDP services require no session or state for a request to be made, it's easy to spoof the source IP address in a request to, say, a DNS or NTP server. If a request to such services responds with a larger packet than the request packet, and the source IP is spoofed to that of a victim server, the UDP server sends a packet to the victim server that's potentially orders of magnitude larger than the attacker's bot could have sent (for bandwidth reasons).

In February 2018, GitHub experienced a DDoS attack of a previously unseen scale - ingress traffic reached a peak of 1.35Tbps, but only around ten thousand unique endpoints were detected. (Kottler, 2018) The subsequent report from GitHub's engineering team found the requests were the result of an amplification attack via the `memcached` key/value store server. This vulnerability was described only a week earlier by Cloudflare and provided an amplification factor of over 50,000, so only a few publicly-exposed `memcached` instances were needed to cripple the website. (Majkowski, 2018)

This is another example of latent insecure internet services causing great risk for all other Internet users. While most system administrators have updated and secured such services (Kerner, 2018), many instances - and servers in general -

are badly-maintained and will remain vulnerable, potentially causing disruption for many years to come.

References

Assor, Y. (2016). *Anti-VM and Anti-Sandbox Explained.* URL: https://www.cyberbit.com/blog/endpoint-security/anti-vm-and-anti-sandbox-explained/ (visited on 18/09/2019).

Bing, M. (2016). *The Lizard Brain of LizardStresser.* URL: https://web.archive.org/web/20181129102135/https://asert.arbornetworks.com/lizard-brain-lizardstresser/ (visited on 23/09/2019).

Boyer, S. (2019). *Self-produced diagrams.*

Bransfield, G. (2003). "The Welchia Worm". In: URL: https://www.giac.org/paper/gcih/517/welchia-worm/105720 (visited on 21/09/2019).

Breslaw, D. and D. Bekerman (2016). *A Wicked Family of Bots.* Imperva. URL: https://www.imperva.com/blog/mirai-stomp-protocol-ddos/ (visited on 26/09/2019).

Computer Misuse Act 1990 (2015). The National Archives. URL: https://www.legislation.gov.uk/ukpga/1990/18 (visited on 26/09/2019).

De Donno, M. et al. (2017). "AntibIoTic: Protecting IoT Devices Against DDoS Attacks". In: URL: https://arxiv.org/pdf/1708.05050.pdf (visited on 21/09/2019).

Durov, P. (2019). *Pavel Durov (Telegram's CEO) on Twitter.* URL: https://twitter.com/durov/status/1138942773430804480 (visited on 10/09/2019). "IP addresses coming mostly from China. Historically, all state actor-sized DDoS (200-400 Gb/s of junk) we experienced coincided in time with protests in Hong Kong (coordinated on @telegram). This case was not an exception."

Gamblin, J. (2016). *Leaked Mirai Source Code for Research/IoC Development Purposes.* URL: https://github.com/jgamblin/Mirai-Source-Code (visited on 05/09/2019).

Goodin, D. (2016). *Record-breaking DDoS reportedly delivered by ¿145k hacked cameras.* Ars Technica. URL: https://arstechnica.com/information-technology/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/ (visited on 30/09/2019).

Hajdarbegovic, N. (2015). "Are We Creating An Insecure Internet of Things (IoT)? Security Challenges and Concerns". In: URL: https://www.toptal.com/it/are-we-creating-an-insecure-internet-of-things (visited on 22/09/2019).

Hilton, S. (2016). *Dyn Analysis Summary Of Friday October 21 Attack.* Dyn. URL: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/ (visited on 30/09/2019).

Intel (2014). *A Guide to the Intenet of Things Infographic.* URL: https://www.intel.co.uk/content/www/uk/en/internet-of-things/infographics/guide-to-iot.html (visited on 16/09/2019).

Joven, R. and K. Yang (2018). *A Wicked Family of Bots*. Fortinet. URL: https://www.fortinet.com/blog/threat-research/a-wicked-family-of-bots.html (visited on 23/09/2019).

Kerner, S. M. (2018). *Memcached DDoS Attacks Slow Down as Patching Ramps Up*. eWeek. URL: https://www.eweek.com/security/memcached-ddos-attacks-slow-down-as-patching-ramps-up (visited on 29/09/2019).

Kottler, S. (2018). *February 28th DDoS Incident Report*. Github. URL: https://github.blog/2018-03-01-ddos-incident-report/ (visited on 23/09/2019).

Krebs, B. (2016a). *KrebsOnSecurity Hit With Record DDoS*. URL: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (visited on 30/09/2019).

— (2016b). *Source Code for IoT Botnet 'Mirai' Released*. URL: https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/ (visited on 05/09/2019).

— (2017). *Who is Anna-Senpai, the Mirai Worm Author?* URL: https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/ (visited on 30/09/2019).

Lustig, T. (2017). *25/50 and 100Gb Ethernet Soon to be Most Deployed Ethernet Bandwidth*. Mellanox. URL: https://blog.mellanox.com/2017/01/2550-and-100gb-ethernet-soon-to-be-most-deployed-ethernet-bandwidth/ (visited on 29/09/2019).

Majkowski, M. (2018). *Memcrashed - Major amplification attacks from UDP port 11211*. Cloudflare. URL: https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/ (visited on 23/09/2019).

Mashevsky, Y. (2005). *The Bagle botnet*. URL: https://securelist.com/the-bagle-botnet/36046/ (visited on 18/09/2019).

Michael, M. (2019). *Attack Landscape H1 2019: IoT, SMB traffic abound*. F-Secure. URL: https://blog.f-secure.com/attack-landscape-h1-2019-iot-smb-traffic-abound/ (visited on 23/09/2019).

Wray, C. (2017). *Current Threats to the Homeland*. URL: https://www.fbi.gov/news/testimony/current-threats-to-the-homeland (visited on 15/09/2019).

Bibliography

FortiGuard IP Reputation Service (2019). FortiNet. URL: https://help.fortinet.com/fddos/4-1-5/index.html#page/FortiDDoS_Handbook/enabling_ip_reputation.html (visited on 25/09/2019).

Generic Routing Encapsulation (GRE) (2019). Techopedia. URL: https://www.techopedia.com/definition/21757/generic-routing-encapsulation-gre (visited on 27/09/2019).

NTP Amplification DDoS Attack (2019). Cloudflare. URL: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/ (visited on 25/09/2019).

Sinanović, H. and S. Mrdovic (2017). "Analysis of Mirai Malicious Software". In: URL: `http://people.etf.unsa.ba/~smrdovic/publications/SoftCOM2017_Sinanovic_Mrdovic.pdf`.

Trivial File Transfer Protocol (TFTP) (2019). Techopedia. URL: `https://www.techopedia.com/definition/1881/trivial-file-transfer-protocol-tftp` (visited on 27/09/2019).