The Mirai Botnet and the IoT Zombie Armies

Georgios Kambourakis[a,b], Constantinos Kolias[a], Angelos Stavrou[a]
[a]Computer Science Department, George Mason University, Fairfax, VA 22030, USA
{gkampour, kkolias, astavrou}@gmu.edu
[b]Dept. of Information and Communication Systems Engineering, University of the Aegean, 83200, Greece

***Abstract*—The rapidly growing presence of Internet of Things (IoT) devices is becoming a continuously alluring playground for malicious actors who try to harness their vast numbers and diverse locations. One of their primary goals is to assemble botnets that can serve their nefarious purposes, ranging from Denial of Service (DoS) to spam and advertisement fraud. The most recent example that highlights the severity of the problem is the Mirai family of malware, which is accountable for a plethora of massive DDoS attacks of unprecedented volume and diversity. The aim of this paper is to offer a comprehensive state-of-the-art review of the IoT botnet landscape and the underlying reasons of its success with a particular focus on Mirai and major similar worms. To this end, we provide extensive details on the internal workings of IoT malware, examine their interrelationships, and elaborate on the possible strategies for defending against them.**

Keywords*—*Mirai, Hajime, DDoS, IoT, Botnet, Network Security.

I. Introduction

Without a doubt, the average household is amassing an increasing number of IoT devices [1]. A similar trend is observed on the use of IoT in industrial use case scenarios [2]. Unfortunately, a great number of IoT devices are build with only cost as the driving design tenet and, as a result, many of them are having poor configurations and open design. Of course, from a security standpoint, they have become the new low hanging fruit for hackers and have drawn the attention of botnet masters [3, 4, 5]. One of the most recent and prominent examples of the security threat that IoT devices pose is the Mirai botnet. The Mirai presence was unveiled in Aug 2016 by MalwareMustDie [6]. There is some speculation that Mirai is the successor to qBot (also known as Qakbot) [7]. However, the origins of the Mirai malware as well as the perpetrators' motivation remain still uncertain. Mirai is classified as an Executable and Linkable Format (ELF) multi-platform worm, and thus is also known as ELF Linux/Mirai. According to estimations [8], after its source code was disclosed on Sept. 30, 2016, Mirai botnets managed to control remotely nearly half a million IoT devices, assembling a mighty botnet. Up to this moment, Mirai and its variants are accounted for some of the largest and most catastrophic DDoS attacks, including those against computer security journalist Brian Krebs's web site, French web host OVH, and Dyn [9], showing an impressive peak of 620 Gbps in the first of the aforementioned incidents. A plethora of other relatively smaller scale DDoS attacks have been carried out using a hit-and-run tactic by Mirai botnets against other targets, including residential IP addresses and game servers. The attack signatures against the network domain www.incapsula.com is a characteristic example of this situation [10]. Specifically, the latter attack in terms of GRE floods had a peak of 280 Gbps attributed to about 50 thousand unique IPs dispersed in 164 countries. These IPs belong to DVRs, WebIP cameras on Busybox, other Busybox equipped Linux IoT boxes, and under-secured Linux servers. Note that Busybox packs stripped-down versions of many common Linux utilities into a small executable, with a focus on embedded OSs having very limited resources. Allegations about Mirai made by an individual under the pseudonymous Anna-Senpai [11] designated an arsenal of about 400 thousand bots. Mirai daily basis statistics and an overview of data pertaining to Mirai C&C servers are provided by NetLab at [12].

As detailed in the next section, until now, several pages, blogs and other sources on the Internet have been sporadically reported on Mirai and its variants. However, to our knowledge, none of them offers a well-rounded review of this particular ecosystem. In this respect, the present article aspires to fulfill this significant literature gap by providing a comprehensive but succinct analysis on the Mirai botnet, its variants, and the repelling tactics and countermeasures.

The remainder of the paper is structured as follows. The next section outlines relevant IoT botnets and describes their basic aspects. Section III presents a high level overview of Mirai, while section V focuses on methods to detect infected devices and repel future infections. The last section concludes and gives pointers to future research.

II. The evolving landscape of IoT botnets

By capitalizing on the vast number of poorly protected IoT devices out there, the release of Mirai's source code [13] (as well as those of other worms or trojans) was a stimulus to similar malware projects. In fact, the adaptation of Mirai's techniques have been already reported, and the number of discovered variants grows on a daily basis. This is also verified by a plethora of recent DDoS incidents against a variety of targets. The hefty 54-hour long app-layer DDoS attack against a US College at the end of Feb. 2017 by a Mirai strain is indicative example of this situation [14]. A Windows-based spreader for Mirai was recently revealed by Kaspersky [15], while nearly 900 thousand customers of Deutsche Telekom Internet Service Provider (ISP) were denied Internet access after their routers being enslaved by a Mirai variant [16]. Researchers at IBM's X-force [17] unearthed a Mirai strain that comes with a new impressive functionality; a bitcoin miner slave. Not surprisingly, Nexusguard's 2016 Q4 DDoS threat report [18] shows a staggering augment of Mirai-based bots number, from 213K to 493K only 2 months after the release of Mirai's source code.

It is believed that Mirai itself is an evolution of an older DDoS type of Trojan known as Bashlight, Bash0day, Bashdoor, Gafgyt, Lizkebab, or Torlus [6, 19]. Very similar to Mirai, Gafgyt connects to random public IP addresses and performs Telnet scanning. If a connection succeeds, it attempts to guess the login credentials from a hardcoded roster. If positive, acting in a raucous way, Gafgyt downloads and attempts to run bot binaries targeting multiple architectures in hopes that at least one will be prosperous. Recently, Mirai variants such as the one identified in Nov. 2016 by honeypots are also known to hit other TCP ports [20, 21]. For instance, TCP port 7547 is well-known to be used by ISPs to remotely manage their customers' broadband routers, and in that case, Bashlight and Mirai variants attack devices found prone to a Simple Object Access Protocol (SOAP) vulnerability [22, 23]. Other ports of interest to this kind of malware include 7547, 5555, 23231, 37777, 6789, 22, 2222, 32, and 19058. Nevertheless, at least up to this moment, the overwhelming amount of scans (97%) is done on ports 23 and 2323.

According to [21] in addition to the those included in the original Mirai code, Mirai variants use 2 new alternative services to resolve the IP addresses of the their C&Cs, namely securityupdates.us, timeserver.host, and 2 report servers, namely, rep.securityupdates.us, ntp.timeserver.host. This fact suggests that Mirai descendants share some of the botnet's infrastructure. Such variants also try to defend against competitors by closing the vulnerable port using iptables. On the other hand, opposite to Bashlight and older versions, newer Mirai malware strains, encrypt communications between the bot and the C&C server, making the defender's task much harder. One could say that today Mirai and Bashlight are two of the prevalent malware families being used to rapidly assemble multitudinous IoT-powered DDoS armies. According to Level3 Communications, Bashlight controls nearly a million IoT devices and is the cardinal competitor of Mirai botnets. Other similar Linux-based trojans that enslave IoT devices for assembling DDoS botnets include PNScan and Remaiten. The first one, targets devices on the x86 platform, and depending on its particular strain, tries to either brute-force router login based on a special dictionary, or/and establish a SSH connection by using only 3 couples of user credentials. The latter is multi-platform and syndicates the facilities of Tsunami (Kaiten) [24] and Gafgyt. An interesting IoT Trojan, the first written in the Lua programming language specifically for ARM architecture IoT Linux machines, was reported on late Aug. 2016 by MalwareMustDie [25]. This malware was named ELF Linux/LuaBot. According to the researchers, LuaBot's primary goal is to zombify devices for adding them to a centrally controlled botnet. It is also known to hijack cable modems with the aim to copy their configuration and loot private certificates, which are later sold to cloners. Lastly, LuaBot is capable of performing application layer DDoS attacks. The binary analyzed by the researchers was equipped with MatrixSSL's code libraries for encryption operation (HTTPS connections) and featured an encrypted C&C communication channel to make hijacking attempts against its C&C harder. After compromising a device, LuaBot restricts remote access to it via the use of tailor-made iptables rules. More interestingly, the Trojan embeds a JavaScript engine for executing scripts signed with the author's public key, is equipped with its own Lua resolver function for DNS queries, includes a SOCKS server, and several other networking capabilities along with the associated libraries. These qualities make LuaBot autonomous, meaning that it does not rely on the infected host to execute its tasks. LuaBot seems to act antagonistically to Mirai and its creators claim to be a white/grey-hat outfit; characteristically, when accessing http://miraitracker.net/ one reads: *"This bot isn't Mirai. Luabot has signature based memory scanner that kills all Mirai based bots running on router"*.

More recently, researchers from Radware [26] have identified in their honeypots another flavor of Busybox-based IoT malware they call "BrickerBot". According to them, the attacking bots were spread worldwide and all had their SSH port open, running an older version of the Dropbear SSH server. Our search using the CyberSpace search engine ZoomEye revealed 7,514,757 devices running the Dropbear SSH server. By leveraging on misconfigurations or security gaps, this ilk of malware aims at launching Permanent Denial-of-Service (PDoS) against Linux-powered IoT devices with the aim of rendering them practically unusable. This is done by defacing the device's firmware, erasing all files from its memory, re-configuring operating system kernel parameters related to network connectivity, removing the default gateway, and so on. As with Mirai, BrickerBot brute-forces Telnet credentials, but it seems that does not download a binary after breaking into the device. A second variant of BrickerBot seems to be more sneaky as the attacker does not rely on Busybox and exploits the Tor network to obfuscate the source IP address of its bots. Another interesting IoT-powered worm that presents very similar behavior to Mirai, at least to its spreading functionality, yet far more sophisticated, was identified on Oct. 2016 by researchers in Rapidity Networks [27]. They coined it as "Hajime", and according to their analysis, it seems that it kicked off a few days before the release of Mirai's source code. Therefore, the researchers presumed it is improbable that the two malwares share any common code. However, it is very likely that the one malware mimics the other in an effort to confuse network administrators and pass itself as another Mirai/Hajime strain. In any case, as it is explained further down, Hajime is more knotty than Mirai, and thus it is examined further in the context of this paper.

III. A 10,000 Foot View of the Mirai Operation

Like the majority of DDoS-malware, Mirai is comprised of two parts; the bots and the infrastructure. The bot part (coded in C) is responsible for unleashing one of several DDoS attacks and for exploring the IP space for new victims. Recall, that it mostly targets Linux-based IoT devices. The botnet's infrastructure depicted in figure 1 is composed of a C&C module (implemented in Go) that provides the herder with a management console, a "report" or "collector" server that gathers and maintains information about the active bots in the botnet, as well as “loader” devices that facilitate the propagation of the malware to newly-discovered victims. So far, various Mirai binaries destined to 18 diverse platforms, including ARM, MIPS, SPARC, Intel x86, and others have been spotted in the wild. For recruiting new members, every bot scans randomly the IP address space, but interestingly, in addition to loopback, internal networks and multicast addresses, it avoids to tinker with those belonging to a hard-coded list of IP addresses. These include the US Postal Service, the Department of Defense, the Internet Assigned Numbers Authority (IANA), General Electric, and Hewlett Packard. For each IP, the original version of the bot probes the standard Telnet 23 port and once in every ten attempts TCP port 2323. This happens in an attempt to discover online services running on the target that prompt for user credentials. Then, the malware employs a brute-force dictionary-based technique for “guessing” passwords based on a hard-coded list. That inventory contains 62 username/password dyads. Arguably, this is an indication that the malware coder targets specific devices rather than searching in the dark. From those pairs, 60 are unique. Also, there are 15 unique user names, and only 42 unique passwords; 2 entries contain no password. Apparently, such credentials exist on the IoT devices to allow remote access to the device by the provider for management purposes.

Upon having a working Telnet session, i.e., gaining shell access, the characteristics of the newly-discovered victim, including its IP address, port, and login credentials are reported back to the “report” or “collector” server listening on port 48101. Via a Tor circuit, the bot-herder is always able to connect to the report server to inspect the new prospective targets, and may decide to infect some or all of them. In that case, the victims’ IP is sent from the report server to some other proxies called “loaders”. The latter entity logs in to the victim and depending on its hardware architecture, instructs it to download (via TFTP or wget) and execute the appropriate binary. Then, the new bot can resolve cnc.changeme.com to learn the current IP of the C&C server listening on port TCP 23 (port 101 is used for management). Using IP fluxing, the latter IP changes over time. From that time on, the bot listens to instructions stemming from the C&C server. Starting 10 secs after establishing a connection, the bot and C&C transmit heartbeat packets to each other every 60 secs.

As already explained in further detail in section II, the malware strives to dominantly eliminate potentially existing competitive worms (such as the Anime, qBot, and Bashlight malwares) via a practise known as memory scraping. Towards the same end, it kills all processes (PIDs) that use SSH, Telnet, and HTTP ports, and binds an own socket to these ports. In some cases, to avoid detection, the worm is also self-deleted, but its process continues to run in memory. It also monitors the device’s watchdog timer to defend against system hangs and reboots (recall that the malware does not write itself to the device’s persistent filesystem). Mirai bot has both network and application layer DDoS attacks in its arsenal. The former category include a variety of floods, including SYN, ACK, UDP, GRE IP and ETH, STOMP, and DNS. For the latter, a Mirai bot can launch HTTP floods masquerading itself as one of several common user-agents.

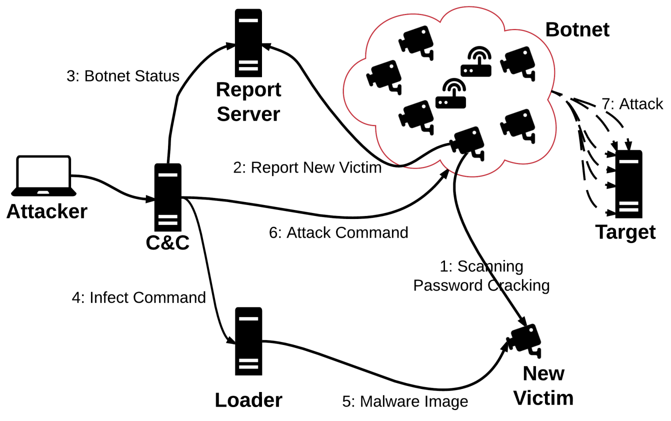

Fig. 1. Overview of Mirai communication and basic components

IV. Detailed Description & Comparison

In this section, we dive deeper into the mechanics of two of the most representative variants of this type of malware, namely the Mirai and Hajime botnets. A comprehensive comparison between the two in matters of their basic functionality is also offered.

A. *Infiltration*

As already mentioned, the first step towards infecting new victims is scanning IPs for open ports and testing for default credentials. Mirai conducts this process on TCP port 23, while Hajime attacks port 5358 as well. The latter port is used by the Web Service on Devices API (WSDAPI), offering a way for implementing Web services on resource constrained embedded devices. While the hardcoded user credential lists for Mirai and Hajime have many common records, Hajime does not examine the list sequentially. Actually, if the login banner is unknown to Hajime, credentials are fetched randomly. After the correct credentials are recovered, Mirai will attempt to gain a Linux shell. It does so by executing a set of commands e.g., *shell; enable; sh; /bin/busybox MIRAI* [6]. Notice, that the final command which is usually /bin/busybox MIRAI or /bin/busybox ECCHI actually refers to a dummy non-existent module. Therefore, the desideratum error message: *“ECCHI: applet not found”* is retrieved. This is a common practise for a) verifying that command sequences that do not produce output have completed, b) verify that busybox is installed, c) it helps the attacking bot to discern between a Cowrie SSH/Telnet honeypot (or other Linux distributions), which typically responds with a help screen if triggered with an erroneous module. Actually, analysis has shown that for all or Mirai variants, every valid command is followed by the execution of a dummy command, including “MIRAI”, “ECCHI”, “IHCCE” or “VDOSS”.

An additional observation is that after discovery of the credentials neither Mirai nor Hajime bots make an effort to reset the password of the infected device.

B. Infection

Upon acquiring a working shell, Mirai attempts to fingerprint the device by checking the system mounts for writeable paths (*cat /proc/mounts*) and exploring its hardware platform by executing *cat /proc/cpuinfo*. That is, for the first part, the malware tests whether a binary file can be created via "echo". Precisely, it echoes the string "kami" to the file /.nippon. The same test is done for all the available partitions. Using the commands *rm /.t; rm /.sh; rm /.human* (repeated for directories /dev, /run, /run/lock, /run/shm) the malware will erase the ".nippon" file and other similar ones that may have been left behind by rivals. The malware tests if TFTP and wget are available. Assuming wget is working, the malware is fetched. In the case neither of these tools is available, the malware will employ the echo command to build the binary. Note that contrary to former malware of this kind, where these exploits were carried out via bash, Perl, or Python scripts, or relatively hefty binaries, for Mirai and similar modern malware the binaries are in the order of few hundred bytes. The malware is executed and immediately after removed.

Hajime presents a similar fingerprinting behavior by selecting as its working path the first writeable directory excluding /proc, /sys, or /. After, it needs to download and execute a tiny binary (platform-specific assembly program, known as loader stub), which will in turn connect back to the attacking bot and download a larger executable. So, after Hajime checks that a stub is not already present at the victim, it inspects the header of the /bin/echo binary to learn the victim's hardware architecture. The latter information is needed to download the correct platform-specific binary. The IP and port (4636) of the attacking host are hardcoded into the loader stub. So far, Hajime supports binaries for the arm5, arm6, arm7, mipseb, and mipsel platforms. Note that the stub will delete itself as soon as the download of the larger binary is finished. Interestingly, as pointed out in [27], at least until now, anyone can connect to the attacking host's malware distribution port because the latter does not check if an incoming connection indeed stems from an infected host. It is also to be noted that Hajime's scan and load module supports the Universal Plug and Play (UPnP) Internet Gateway Device (IGD) protocol supported by a great variety of NAT-enabled routers. This enables it to punch pin-holes in such devices exposing the ports it requires for its operation even behind the gateway.

C. Operation

The new Mirai bot will start scanning for other vulnerable devices, initiating a maximum of 128 connections/sec. Once a minute, it sends any information it has collected to the report server on port 80. It also establishes a raw socket connection to C&C server to receive commands. A command includes the attack type, attack duration, attack id, target count, and a list of targets. On the other hand, Hajime relies on BitTorrent DHT protocol with dynamic, daily-changing info_hashes for peer discovery and uTorrent Transport Protocol (uTP) for data exchange [27]. These overlaid communications on top of BitTorrent P2P network are RC4 encrypted and signed using public and private keys. For this functionality, the worm opens up a random TCP high port and UDP port 1457. The first port enables the infection process to remotely download the malware to the newly enslaved device, while the second allows Hajime to use BitTorrent DHT and uTP. So, in opposite to Mirai which relies on a centralized infrastructure, Hajime demonstrates high resilience as its communications regarding configuration and software updates to its member bots happen over a tracker-less fully distributed overlay network. This means that Hajime is far more impervious than Mirai to takedown and hijack attempts by correspondingly ISPs and competitive botnet operators. Another major difference between the two malwares is that all present Hajime variants do not afford any DoS or other attack functionality. Instead of that, its sole aim is to propagate to other vulnerable devices and next ban access to their vulnerable TCP ports, namely 23, 7547, 5555, and 5358. It also removes any firewall rule pertaining to Customer Premises Equipment (CPE) WAN management protocol, blocking this way ISP access to the device. Such behavior is similar to BrickerBot and Linux/Wifatch [28] and basically targets at blocking Mirai and similar antagonistic IoT botnets. This however does not make Hajime a real defender against Mirai and similar worms. That is, it is not to be forgotten that Hajime secretly and without authorization installs a back entrance on the device, which is at least nefarious and mostly unlawful. And of course, no one can rule out the possibility of future Hajime versions being capable of conducting large-scale vulnerability scanning, massive surveillance, executing DDoS, etc.

V. Detection and Mitigation

Based on the analysis of the previous section, it is obvious that strong signatures can easily be deduced for network or host-based methods to detect the presence of Mirai, Hajime, Bashlight or other resembling family of botnets throughout the different phases of their lifecycle. First off, during the infection stage, one can passively monitor network traffic on standard ports 23, 2323, and 22, which are recursively bombarded with authorization attempts to gain access to the IoT device. For this ilk of malware, the number of alternative credentials being used is limited, thus foreseeable.

Moreover, Mirai bots frequently exchange traffic with the loader, even when in state of inertia. While the messages send by the bots are not always the same, a pattern of opening a TCP connection, followed by a sequence of packets of predictable size and then finalizing the connection can easily be extracted. Figure 2 is indicative of such a pattern.

A surge of egress traffic can be also observed throughout the course of an attack. Mirai bots follow a greedy approach with the supported attack types. Our experiments with a Mirai-infected Raspberry Pi 3 attest that as soon as the bot receives an attack command e.g., to execute SYN flooding, it starts producing more than 1,500,000 SYN packets in a minute, which translates in an average of 25,000 packets per second. Yet, most of the attacks do not make a serious attempt to randomize specific packet fields, thus the forged packets can easily be distinguished from the benign. For example, the HTTP flooding attack relies on five alternative specific user-agents similar to the: *Mozilla/5.0 (Macintosh; Intel Mac*

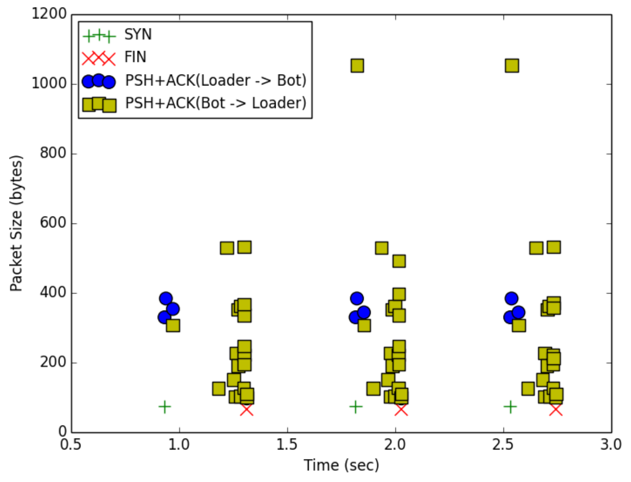

Fig. 2. Pattern of communication between Mirai bot and loader

OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko) Version/9.1.2 Safari/601.7.7

Having the above in mind, typical countermeasures to minimize the attack surface include the blocking of TCP ports used for probing and brute-forcing the device, and the dropping of TCP egress connections containing attack traffic, say, by applying egress and ingress filtering at router level and deploying rule-of-thumb countermeasures for preventing DDoS attacks. The well-known "keep security simple" principle must be also followed by closing and stopping nonessential ports and services running on the device. A more drastic measure, which however is not always desirable, is to ensure that no IoT device is accessible from the Internet and the organization's intranet is fully isolated. Recall that in some cases, modems do not implement NAT, meaning that every device connected to such a modem will receive a public IP address. Even worse, usually, the modem offers no friendly interface to manage the open/closed ports. If remote access to a device is mandatory, a Virtual Private Network (VPN) can be used, when applicable. Other solutions, much more preferable than Telnet, are https for GUI management, and SSH or TLS-based Telnet security for shell access.

Allowing Busybox execution only by a specific user can also be of help to the defender. This can be applied in combination with account lockout policies (e.g., fail2ban) to fight against brute-force attacks, and automatic logging out and re-authentication if the device is left unattended or idle for some time. Generally, the access to any device must be controlled based on an access list (least privilege). Because in the majority of cases this kind of malware resides only in memory, rebooting the gateway will remove the infection. However, this is futile without changing the default password to a secure value beforehand (preferably before first time deployment). As already pointed out in the previous section, UPnP on IoT devices should be also disabled. The operating system and applications running on an IoT device must be also updated with the latest patches and bug fixes. To promptly detect scan attempts against any system or device, a logging process on both the network perimeter and the devices themselves must be activated by default. In addition to all the above, detection and countermeasures specific to Hajime can focus on UDP packets carrying P2P traffic and dropping them using, say, an application layer firewall facility. Also, monitoring and possibly blocking TCP port 4636, as Hajime stub binary acquires the worm over that port.

VI. Conclusions

The sheer volume of Linux devices directly exposed to the Internet primarily for remote management purposes, along with the lack of frequent firmware updates, the ease to built IoT exploits, and the numerous of already known, scalable vulnerabilities found in IoT devices, equip malicious actors with ample ammo to overpower security measures via the assembly of powerful botnets. In addition to all the malware families examined in section II, the case of "Leet" botnet which seems to challenge Mirai in terms of DDoS attack power verifies this observation [29]. This threat is persistent given that even script-kiddie type of attackers are able to modify ready-to-run code as that of Mirai to built their own variant. Such modern malware is usually slim, multi-platform-destined, and in most cases does not even care of staying hidden. That is, as already explained, Mirai turns its victims into zombies hamstringing the device until the user notices it and performs a soft reboot. This however may happen after a long period of time, in which the enslaved device serves solely the botmaster. All in all, it would not come as a surprise if in the near future we witness new strains of such malware families, which in addition will be capable of writing themselves to the device's persistent filesystem. On top of that, it can be safely estimated that future strains will target an extended range of IoT devices. Especially for BrickerBot and similar malware, this undoubtedly means greater dissatisfaction and frustration for the end-user.

It seems that the problem is mainly attributed to the vendors for shipping their devices with active remote admin capabilities (Telnet, ssh, etc), which however are undocumented (so the end-user is neither aware of nor knows how to use them), rather to the end-users for not changing the default password or applying the latest software patches, if any. One could say that Linux is not even the most appropriate operating system for many of these IoT devices (think of a fridge, air-conditioner, or thermostat) as it is sure to augment the attack surface and render the device prone to the plethora of legacy attacks met in the desktop world. Arguably, a safer option would be the use of a slimmer, cumbersome to fingerprint OS, such as RIOT OS, Google's Brillo, ARM Mbed OS, Nucleus RTOS, FreeRTOS, and others. For the present time, firmware update appears to be an effective solution. However, this is rather cumbersome to be done on large-scale because it requires physical access, and it is unique per vendor/model. So, until this happens, an IoT device may "ping-pong" between different botnets. That is, following every reboot, the same device is potentially enslaved by different malware. In a way, this means two things. First, the lifetime of such a worm on a specific device depends on reboots, and second the "protection" provided by Hajime or similar worms against Mirai and others holds until the next reboot, given that the changes all these worms make are not persistent, but only in RAM.

REFERENCES

[1] Research Nester. *IoT Market: Global Demand, Growth Analysis and Opportunity Outlook 2023*. Tech. rep. Feb. 2017. URL: http://www.researchnester.com/reports/internet-of-things-iot-market-global-demand-growth-analysis-opportunity-outlook-2023/216.

[2] L. D. Xu, W. He, and S. Li. "Internet of Things in Industries: A Survey". In: *IEEE Transactions on Industrial Informatics* 10.4 (Nov. 2014), pp. 2233–2243.

[3] K. Angrishi. "Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT Botnets". In: *arXiv.org* arXiv:1702.03681v1 (Feb. 2017). URL: https://arxiv.org/abs/1702.03681v1.

[4] C. Kolias, A. Stavrou, and J. Voas. "Securely Making Things Right". In: *Computer* 48.9 (Sept. 2015), pp. 84–88. ISSN: 0018-9162. DOI: 10.1109/MC.2015.258.

[5] M. Anagnostopoulos, G. Kambourakis, and S. Gritzalis. "New facets of mobile botnet: architecture and evaluation". In: *International Journal of Information Security* 15.5 (Oct. 2016), pp. 455–473.

[6] MalwareMustDie! *Linux/Mirai, how an old ELF malcode is recycled*. Aug. 2016. URL: http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html.

[7] N. Falliere. *W32.Qakbot in Detail*. Symantec. 2009. URL: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_qakbot_in_detail.pdf.

[8] M. Mimoso. *Mirai Bots More Than Double Since Source Code Release*. Oct. 2016. URL: https://threatpost.com/mirai-bots-more-than-double-since-source-code-release/121368/.

[9] KrebsOnSecurity. *KrebsOnSecurity Hit With Record DDoS*. Sept. 2016. URL: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[10] I. Zeifman B. Herzberg D. Bekerman. *Breaking Down Mirai: An IoT DDoS Botnet Analysis*. Oct. 2016. URL: https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html.

[11] KrebsonSecurity. *Who is Anna-Senpai, the Mirai Worm Author?* Jan. 2017. URL: https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/.

[12] NetLab. *Mirai Scanner*. 2017. URL: http://data.netlab.360.com/mirai-scanner/.

[13] *Mirai-Source-Code*. Oct. 2016. URL: https://github.com/jgamblin/Mirai-Source-Code/tree/master/mirai/bot.

[14] D. Bekerman. *New Mirai Variant Launches 54 Hour DDoS Attack against US College*. Mar. 2017. URL: https://www.incapsula.com/blog/new-mirai-variant-ddos-us-college.html.

[15] Kaspersky Lab. *A Windows-based spreader for Mirai malware has been discovered*. Feb. 2017. URL: http://newsroom.kaspersky.eu/en/texts/detail/article/skilled-attacker-develops-advanced-windows-botnet-to-spread-infamous-mirai-malware.

[16] KrebsOnSecurity. *New Mirai Worm Knocks 900K Germans Offline*. Nov. 2016. URL: https://krebsonsecurity.com/2016/11/new-mirai-worm-knocks-900k-germans-offline/.

[17] M. Alvarez D. McMillen. *Mirai IoT Botnet: Mining for Bitcoins?* Apr. 2017. URL: https://securityintelligence.com/mirai-iot-botnet-mining-for-bitcoins/.

[18] NexusGuard. *Distributed Denial of Service (DDoS) Threat Report Q4 2016*. Dec. 2016. URL: https://news.nexusguard.com/threat-advisories/q4-2016-ddos-threat-report.

[19] MalwareMustDie! *Overview of "SkidDDoS" ELF++ IRC Botnet*. Feb. 2016. URL: http://blog.malwaremustdie.org/2016/02/mmd-0052-2016-skidddos-elf-distribution.html.

[20] US-CERT. *Heightened DDoS Threat Posed by Mirai and Other Botnets Alert (TA16-288A)*. Nov. 2016. URL: https://www.us-cert.gov/ncas/alerts/TA16-288A.

[21] NetLab. *A Few Observations of The New Mirai Variant on Port 7547*. Nov. 2016. URL: http://blog.netlab.360.com/a-few-observations-of-the-new-mirai-variant-on-port-7547/.

[22] radware. *Telecom & IT Service Providers Beware: New IoT Threat*. Dec. 2016. URL: https://security.radware.com/WorkArea/DownloadAsset.aspx?id=1309.

[23] L. Oppenheim S. Tal. *The Internet of TR-069 Things: One Exploit to Rule Them All*. Apr. 2015. URL: https://www.rsaconference.com/writable/presentations/file_upload/hta-r04-the-internet-of-tr-069-things-one-exploit-to-rule-them-all_final_copy1.pdf.

[24] MalwareMustDie! *Another story of Unix Trojan: Tsunami/Kaiten.c (IRC/Bot) w/ Flooder, Backdoor at a hacked xBSD*. May 2013. URL: http://blog.malwaremustdie.org/2013/05/story-of-unix-trojan-tsunami-ircbot-w.html.

[25] MalwareMustDie! *Linux/LuaBot - IoT botnet as service*. Sept. 2016. URL: http://blog.malwaremustdie.org/2016/09/mmd-0057-2016-new-elf-botnet-linuxluabot.html.

[26] radware. *"BrickerBot" Results In PDoS Attack*. Apr. 2017. URL: https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/.

[27] I. Profetis S. Edwards. *Hajime: Analysis of a decentralized internet worm for IoT devices*. RapidityNetworks. Oct. 2016. URL: https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf.

[28] M. Ballano. *Is there an Internet-of-Things vigilante out there?* Oct. 2015. URL: https://www.symantec.com/connect/blogs/there-internet-things-vigilante-out-there.

[29] D. Bekerman A. Zawoznik. *650Gbps DDoS Attack from the Leet Botnet*. Dec. 2016. URL: https://www.incapsula.com/blog/650gbps-ddos-attack-leet-botnet.html.