LIST: Lightweight Solutions for Securing IoT Devices against Mirai Malware Attack

1st Pallavi Kaliyar
Department of IIK
NTNU
Gjøvik, Norway
pallavi.kaliyar@ntnu.no

2nd Laszlo Erdodi
Department of IIK
NTNU
Trondheim, Norway
laszlo.erdodi@ntnu.no

3rd Sokratis Katsikas
Department of IIK
NTNU
Gjøvik, Norway
sokratis.katsikas@ntnu.no

***Abstract*—Recently, the number of Internet of Things (IoT) devices has increased significantly, as they have become affordable to most people. This spread has highlighted a critical security threat, namely the increasing number of Distributed Denial of Service (DDoS) attacks. As these resource-constrained IoT devices are built to be cost-efficient, their security measures are limited. Moreover, most users are not aware of the security measures that they must apply. Nowadays, almost every IoT device (e.g., fridge, air conditioner, thermostat, toaster) is able to connect to the internet, and this allows the user to access and control it with its own smartphone application. The lack of security measures in these devices was highlighted in September 2016, when a large-scale DDoS attack was launched using a botnet of compromised IoT devices. This type of attack has been since used in different forms and has been classified as *Mirai DDoS Botnet Attack*. This paper presents a detailed analysis of the Mirai attack and of the source code of the Mirai malware, reports on the implementation of the attack in a controlled environment, and proposes possible solutions that could help in mitigating the attack.**

***Index Terms*—Mirai Attack; Authentication; Internet of Things; malware; security.**

I. Introduction

Since its discovery in 2016, the Mirai's diffusion has been rapid and dramatic at the same time [1]. In August 2016, a new trojan that preyed on Unix Operating System's Executable and Linkable Format (ELF) files was discovered by a "MalwareMustDie" whitehat group [2]. The trojan aimed to send telnet attacks to other systems. During 2016 only, Mirai infected thousands of IoT devices. The power of this malware, which works with BASHLITE to carry out a DDoS attack, was clear to everyone in September 2016, when a huge DDoS attack took down Brian Krebs's [3] website with traffic of 620 Gbits/s. The attack was carried out with a huge number of bots that were located all around the world. In the same month, the French cloud and web hosting company OVH [4] became victim of another DDoS attack with a bigger traffic than the previous attack, i.e., 1 Tb/s. In this case, it was reported that the botnet was composed of 145,607 different devices from 8 different regions around the world, and they mainly were IoT devices like IP cameras and Digital Video Recorders.

In October 2016, the code of the Mirai was released so anyone can retrieve it from the internet [5] for analysis purposes. This inevitably led to a bigger diffusion of the code that other parties modified and improved. Due to this, the number of compromised Internet of Things (IoT) devices in 2016 varied from 213,000 to 493,000. In the same month, Dyn [6], a core ISP was hit by a massive DDoS attack against its DNS (Domain Name Server) infrastructure on the east coast of America, and this brought down some of the websites for which it provided services such as Twitter, Spotify, and Reddit. In November 2016, the Mirai took down almost all of Liberia's [7] websites, as the African state has only one internet cable, which provides a single point of failure for internet access. In the same month, a botnet of 400,000 IoT devices was up for rent on the deep web. The price was $2,000 for 20,000 compromised nodes. Furthermore, in December 2016, the British ISP TalkTalk [8] reported that Mirai had targeted customers using its Dlink DSL-3780 router.

In February 2017, Kaspersky Lab [9] researchers found that a hacker had created a variant of Mirai based on the Windows operating system. The researchers claim that the ability of this malware to spread across different Operating Systems is very limited. However, it was a sign that the Mirai power increased after releasing its source code. Later, in December 2017, two suspects admitted their guilt in developing and deploying the Mirai botnet. Obviously, this is not the end of Mirai's history as the vulnerabilities of the IoT devices that the malware uses are still present. If we cannot address this threat, the Mirai will become more powerful as vulnerable devices increase. Some other variants of Mirai are listed in Table I.

The main goal of this work is to suggest lightweight solutions for securing IoT devices against the Mirai malware attacks. We propose three lightweight solutions that can be used to secure resource-constrained vulnerable IoT devices with negligible overhead on the manufacturing cost. The key contributions of this work are:

- First, we present a detailed analysis of the Mirai source code, which is publicly available on the git repository [5] at GitHub since 2017. This analysis is important as it provides a more detailed description of the Mirai attack source code, i.e., what is the role of each Mirai source file in the execution of the Mirai attack.
- Our second contribution is the implementation of the Mirai code in a controlled environment, to show that although the Mirai attack has been long known, it is

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

TABLE I
DIFFERENT VARIANTS OF MIRAI

No.	Name	First Appearance	Exploit
1	Mirai original [5]	August 2016	Telnet 23/2323, brute force
2	Satori [10]	December 2017	Telnet 23/2323, Port 37215/52869, 2 exploits CVE-2014-8361 and CVE-2017-17215
3	Hajime [11]	March 2017	Telnet 23/2323, brute force, later closes the open ports
4	IoTroop [12]	October 2017	Vulnerability scanning instead of password brute-force
5	Okiru [13]	January 2018	IoT with RISC architecture, telnet default passwords 4 types of router exploits
6	Masuta, PureMasuta [14]	January 2018	EDB 38722 D-Link exploit
7	Jenx [15]	January 2018	2 exploits, CVE-2014-8361 and CVE-2017–17215
8	OMG [16]	March 2018	Make IoT a proxy server
9	Wicked [17]	June 2018	Port 80,81,8080,84433, new exploits, router exploits, cctv rce, CVE-2016-6277 command injection
10	Satori / 2018 [18]	July 2018	Android Debug Bridge (ADB) commands
11	Torii [19]	September 2018	Rich set of features for exfiltration of (sensitive) information, modular architecture capable of fetching and executing other commands and executables
12	Hakai, Yowai [20]	January 2019	Several hard coded exploits, ThinkPHP
13	Covid Mirai [21]	March 2020	TeamSpeak, Huawei default passwords
14	Satori – 2021 [22]	February 2021	Vantage Velocity field, Python script
15	Matryosh [23]	February 2021	Android Debug Bridge, TOR network is used

still very relevant, as a large number of devices are still vulnerable to it.
- Our third contribution is to propose three lightweight solutions to improve the security of IoT devices against Mirai and Mirai-like attacks. Contrary to previously proposed, state-of-the-art solutions, our solutions are applicable to both new and existing IoT devices, they do not require increased computational power, storage capability, or battery capacity, and they do not add any extra manufacturing cost to the devices.

The rest of the paper is organized as follows: Section II briefly discusses Mirai's evolution and analyzes related works that propose solutions to mitigate it. Section III describes how the Mirai attack works, focusing in particular on the Mirai source code. This is followed by the description of a real-world experiment that we conducted to gain remote access to an IoT device by launching a Mirai attack. The last part of Section III discusses three proposed solutions to limit the damage caused by the lack of security measures in IoT devices. Finally, we conclude our work in Section IV.

II. BACKGROUND AND RELATED WORK

In this section, first we briefly present (in sub-section II-A) different variants of the Mirai malware that have appeared in the last six years. Next, we briefly summarize (in sub-section II-B) security solutions proposed by other researchers.

A. Background: Evolution of the Mirai malware

The first variant of the Mirai that appeared in 2016 had separate loader and scanner modules. First it looked for open *telnet* ports and then used the default username, passwords, and password brute-force on port 23/2323. After this first variant, many other variants of Mirai appeared in the last six years. A list with a few of these key variants is shown in Table I. Below we list some of the changes identified in the working methodology of these different variants.

- The bot is able to do the scanning too, no separate scanning module is needed anymore.
- Several new exploits, such as router *http* interface vulnerabilities, Android Debug Bridge remote code execution, are added, in addition to the *telnet* as default.
- Some variants (e.g., IoTroop [12]) can do vulnerability scanning besides finding predictable credentials.
- In addition to being capable of carrying out DDOS attacks, some variants provide extra functionality, such as providing proxying functionality (using IoTs to forward packets in order to hide the source of the packet origin) [16].
- The number of devices involved has been increased, e.g., RISC processors [13].

B. Related work

The increasing number of botnets created using the Mirai attack has motivated cybersecurity researchers to develop efficient solutions to this problem. The solution used in the past proposes to analyze the Mirai traffic to find specific patterns that will allow the identification of the attack. This is the typical procedure that is used to analyse a malware, and it is done by using a *honeypot*. This strategy has been used in [24] [25]. A honeypot works as an IoT device and accepts all the attacker requests and replies with the intended commands. In particular, the honeypot used in [25] is made of two parts, as shown in Figure 1. The front-end part, which interacts with the Mirai malware, replies with the intended *telnet* commands. The back-end part records all the commands used by the malware to compromise the fake device and all the incoming traffic. The authors discovered that in the initial phase, the Mirai executes many *telnet* commands intended to

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

gain control of the device's shell. The possibility to discover if the Mirai malware is attacking a device is an interesting solution, but it does not offer any protection against this type of attack. It would be very difficult to store information for all different IoT devices and the pattern that identifies the malware on the devices, and instruct them to check each access attempt. This approach is infeasible, as we know that most IoT devices have very low computational power and even lower storage capability and battery capacity.

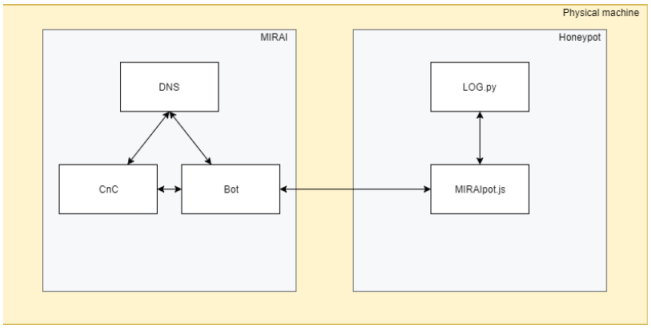

Fig. 1. Multi-component honeypot structure [25]

Another security solution to the Mirai botnet builds a whitelist-based intrusion detection technique for IoT devices [26]. This solution uses a gateway router which acts as a firewall for a set of IoT devices belonging to the local network that the router serves. The proposed mechanism is called *Heimdall*. It uses the gateway router to build a profile for each responsible device. A profile contains mainly a whitelist of the destinations (IP addresses) that a specific device can legitimately reach to perform its functions. Moreover, the profile also stores the typical traffic pattern of that device, including some statistics, e.g., number of TCP, UDP packets of the incoming and outgoing traffic. This is useful to prevent both the device from being attacked by Mirai (incoming traffic) and carrying out the attack (outgoing traffic). The profile is dynamic, so it is continuously updated, increasing the traffic pattern's precision. This approach seems attractive, as it does not require additional resources from the IoT device, because the Heidmall router does all the work. However, the solution faces some serious problems: First, the list of a single device's destinations may change very frequently, as usually the backend services of IoT devices are hosted on some public cloud infrastructure due to the devices' limited capabilities. Accordingly, the IP address of the destination servers may change very often. Secondly, this mechanism can become victim of a DNS poisoning attack. An attacker modifies the IP address from which the traffic is coming into a malicious one, so that the gateway router will reject some legitimate traffic.

After the public release of the Mirai source code, another countermeasure was developed, in the form of a worm called *Hajime* [11] [27]. This piece of "benign" malware works basically as Mirai; it uses the default logins to control IoT devices. It even uses the same username-password dictionary of Mirai. The purpose of Hajime is to gain access to the vulnerable devices to close their open ports, e.g., ports 23, 7547, 5553, and 5358, so that an attacker will not be able to use them. Thus, this code is like an anti-Mirai, as it hacks the devices to secure them. The problem with this approach is that the code is not persistent, as it is loaded on the device's RAM, and therefore it is deleted after each reboot.

Later in [28], the authors employed a static analysis to audit firmware of IoT devices to check its susceptibility against Mirai, which is not a feasible solution considering all scenarios of IoT applications. The authors in [29] propose a model made of a transformer-encode and use a hierarchical structure to extract semantic features from the information and functions to classify the malware. In [30], the authors proposed a Fog Computing-based IoT-DDoS defense framework for contemporary real-time IoT traffic to identify the presence of Mirai malware in the network. The authors in [31] proposed a Machine Learning (ML)-based mechanism for detecting Mirai Botnet attacks in IoT-based networks. The ML-based detection mechanism was detecting the attack using a real traffic dataset of IoT devices.

Having considered all these existing solutions [11], [24]–[31] with their advantages and drawbacks, we propose three new lightweight solutions against the Mirai attack that use an external device to authenticate the user to the IoT device, without the need to increase the capabilities of the device itself.

III. Proposed Approach

In this section, we first analyze the Mirai source code and discuss the Mirai attack on vulnerable, resource-constrained IoT devices. We then propose our solutions for securing the IoT devices against Mirai-like attacks.

A. Mirai source code analysis

The main idea behind the Mirai attack (see Figure 3 is to create a botnet made of IoT devices that a BotMaster/attacker can control to carry out a DDoS attack [32]. Initially, the Command and Control (CNC) server starts scanning for IP addresses with port 23 (*telnet*) open to control the bots. When such devices are found, the attacker injects the malware code as it controls the shell, and then the bot's information (IP address, port, and authentication credentials) are stored in a list. Once the malware is installed on the devices, it hides. The device continues its regular activities without knowing that it has been infected.

The Mirai malware source code can be found on the git repository [5] at GitHub. The code is mainly written in two programming languages, namely *Go* and *C*. The *Go* programming language is used to implement the part of the code which is used to control the CNC server [33]. The script named "admin.go" implements the primary administration interface that issues commands from the CNC server. The code script named "clientList.go" keeps track of the data needed to execute an attack, including a map/hashtable of the bots which are charged to carry out a specific attack. This code is also responsible for recording and checking the state of the bots before and after the attack. The attack requests initiated

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

Username	Password	Username	Password	Username	Password	Username	Password
root	xc3511	admin	meinsm	root	12345	root	7ujMko0vizxv
root	vizxv	guest	12345	user	user	root	7ujMko0admin
root	admin	tech	tech	admin	(none)	root	system
admin	admin	admin1	password	root	pass	root	ikwb
root	88888	administrator	1234	admin	admin1234	root	dreambox
root	xmhdipc	666666	666666	root	1111	root	user
root	default	888888	888888	admin	smcadmin	root	realtek
root	juantech	ubnt	ubnt	admin	1111	root	1010101
root	123456	root	klv1234	root	666666	admin	11111111
root	54321	root	Zte521	root	password	admin	1234
support	support	root	hi3518	root	1234	admin	12345
root	(none)	root	jvbzd	root	klv123	admin	54321
admin	password	root	anko	Administrator	admin	admin	123456
root	root	root	zlxx.	service	service	admin	7ujMko0admin
guest	guest	mother	fucker	supervisor	supervisor	admin	pass

Fig. 2. Default username and passwords in scanner dictionary

by the CNC server are executed by the code script named “attack.go”. This part of the code parses and formats the commands received, and sends them to the appropriate bots via code script named “api.go”. The code script “attack.go” can also set the attack duration, and the attack command is sent to the individual bot through the code script “api.go”.

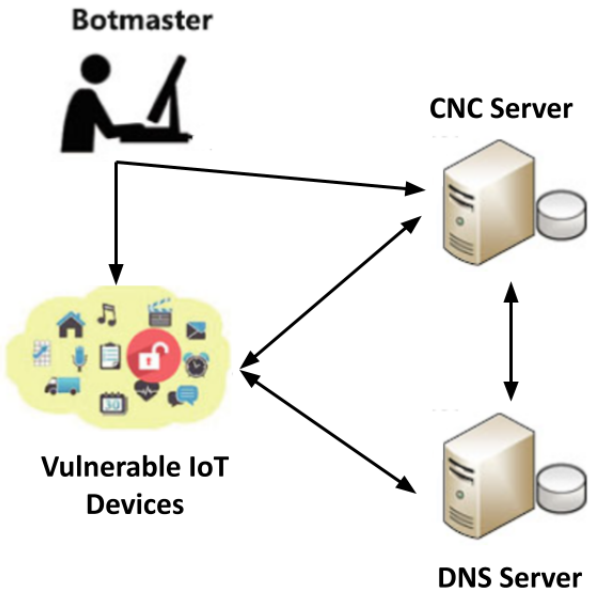

Fig. 3. Mirai Attack procedure

The most crucial piece of code in the CNC server is “main.go”, which regularly listens for connections on ports 23 (*telnet*) and 101 (*apibot* responses). If a connection to port 23 is found, the device is acquired and its credentials are stored. In the case of a port 101 connection, the control is handed over to the code script “api.go” which deals with an individual bot.

The code for the bot is written in the C programming language. It includes different functions built explicitly for different types of attack. The script “attack_udp.c” is able to carry out various types of UDP DDoS attacks, such as Generic Routing Encapsulation (GRE), Reflective Denial of Service (bandwidth amplification), DNS Flood via Query of type *A* record (map hostname to IP address), and Flooding of random bytes via plain packets, under specific commands. The code scripts “attack_tcp.c” and “attack_app.c” work similarly and can realize different types of attacks. The code script named “scanner.c” is used by the bots to do a brute force scanning on a range of IP addresses using a port scan (SYN scan) and trying to access vulnerable devices using a dictionary of default usernames and passwords, which is shown in Figure 2. This scanning aims to gain access to other vulnerable IoT devices and add them to the pool of botnets. If accessing a new device is successful, the bot reports to the CNC server information on the victim, i.e., IP address, port number, and authentication credentials. The Mirai also uses the code script “killer.c” which is responsible for killing various processes like *telnet* and *ssh* inside the bot. All the executable of the bot is controlled by the code script “main.c”, which establishes the connection to the CNC server, starts the attack, kills processes, and even scans for additional bots to add to the botnet by making use of the other pieces of code as described above.

B. Experimental setup

We performed a real-world experiment to prove how vulnerable are the resource constrained IoT devices which we are using in our daily lives. This experiment (see Figures 4 and 5) was conducted using the Mirai Botnet DDoS attack technique. First, IoT device IP addresses were collected from

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

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

var clientList *ClientList = NewClientList()
var database *Database = NewDatabase(DatabaseAddr, DatabaseUser, DatabasePass, DatabaseTable)

func main() {
    tel, err := net.Listen("tcp", "0.0.0.0:23")
    if err != nil {
        fmt.Println(err)
        return
    }

    api, err := net.Listen("tcp", "0.0.0.0:101")
    if err != nil {
        fmt.Println(err)
        return
```

Fig. 4. Mirai scanning code

```
#define ATK_VEC_UDP         0  /* Straight up UDP flood */
#define ATK_VEC_VSE         1  /* Valve Source Engine query flood */
#define ATK_VEC_DNS         2  /* DNS water torture */
#define ATK_VEC_SYN         3  /* SYN flood with options */
#define ATK_VEC_ACK         4  /* ACK flood */
#define ATK_VEC_STOMP       5  /* ACK flood to bypass mitigation devices */
#define ATK_VEC_GREIP       6  /* GRE IP flood */
#define ATK_VEC_GREETH      7  /* GRE Ethernet flood */
//#define ATK_VEC_PROXY       8  /* Proxy knockback connection */
#define ATK_VEC_UDP_PLAIN   9  /* Plain UDP flood optimized for speed */
#define ATK_VEC_HTTP        10 /* HTTP layer 7 flood */
```

Fig. 5. Mirai DDoS attacks

an open access database. When such a device is found, we look on the website *shodan.io* [34] for the type of device. We used 20 IoT devices with public IP address; 18 of these were found on *shodan.io* database. Based on this, we created an emulated network with all the devices we collected and launched the Mirai Botnet attack. In our emulated network, 20 different devices were placed, with open port 23. For 18 of them, we set up default credentials according to the *shodan.io* database. Since we chose the devices randomly we believe this experiment was a small scale but realistic one. We observed that the botnets were able to carry out the scanning and infection.

C. Lightweight security solutions to mitigate Mirai Attack

In our proposed approach for addressing the Mirai attack, we categorize the different resource-constrained IoT devices in three different levels, which differ with respect to the security features provided by the manufacturer. The details of these levels are as follows:

- Level 0: No Security, i.e., no security measures have been taken or applied to the IoT device.
- Level 1: Medium Security, i.e., few measures have been taken and applied to the IoT device.
- Level 2: Full Security, i.e., continuous security service and monitoring through a service provided by a specialized service provider or by the manufacturer.

We propose lightweight security solutions for the Level 0 (i.e., No security) IoT devices, as these are most commonly used in various real-world applications. The following subsections provide details about our proposed solutions that could help mitigate the different variants of the Mirai attack. Our proposals are based on the assumption that the security solutions must not affect the cost of the devices. This is because the vendors are developing devices that are getting cheaper day by day to make them affordable to many consumers. Hence, our solutions must increase the security of these devices without significantly affecting their cost. Additionally, we provide security not only to the devices that will be built in the future but also to those already in use.

1) Secure Authentication: Despite the solutions which are already presented to mitigate the Mirai attack (refer to Section II-B), we propose to improve the security of IoT devices by protecting them with more secure and hardly guessable login credentials. The malware to control and access these devices is injected only after the device has been compromised due to its weak login credentials. To make the prediction of username & password (authentication process) more complex, an idea is to generate a periodically random username & password for accessing the device. These random values must be built combining more random numbers generated through a pseudo-random number generator (PRNG) such as *Blum Blum Shub* [35].

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

This approach is somewhat complicated and insufficient because the credentials will be challenging to guess not only by the attacker but even by the owner of the device. Because of this, the random values created periodically must be stored in a different device, such as a smartphone. The idea is to build an Android application that will store only the current values for accessing the device, deleting the older ones, and will keep them secure by using encryption at another layer, e.g., using a password chosen by the user to access the application. Obviously, the communication between the external device and the IoT device for sharing the values must be encrypted so that an attacker cannot sniff the credentials during the data sharing. This leads to the necessity of adding a cryptographic suite to the IoT device. The solution does not significantly affect the manufacturing cost of the device, as the only requirement is to build a PRNG application and cryptographic functions to encrypt the data shared with the android application in the device; these can be implemented in hardware or firmware. Furthermore, the proposed solution is also applicable to existing devices, by means of a firmware update.

2) Biometric Authentication: The second solution aims to reduce the cost of the device even more, as it uses some features that are already present in the externally linked devices, which again can be a smartphone. We propose to utilize the biometric features e.g., digital fingerprint of the user as authentication parameters. Nowadays, it is very common for all smartphones to offer biometric authentication hence we can use these already existing mechanisms for IoT devices. The idea is to link a specific external device to one or more IoT devices so that we do not have direct communication between the IoT device and the database where its credentials are stored. In fact, the external device itself communicates with the Server with its ID, and provides the biometric authentication parameters. If such an external device is authorized for the specific IoT device to which it requires access and the credentials are the same as those stored in the server database, the access is granted. This idea is similar to the OAuth (Open Standard Authorization) concept where the authentication is provided by a third party component and the IoT device is only asking for authorization [36].

To better explain how this process works (see Figure 6), we initially need to define the first part of the authentication phase. When a new IoT device is booted, it must specify which external device will be used for authentication. In order to do this, the IoT device will be provided with some temporary credentials that the user must use to authenticate in a specific android application. Once the access has been completed, the user must create new credentials based on some biometric parameters used for future authentications. During this phase, the server to which the IoT device refers will store the device's serial number from which the application/ios has been used. For some specific devices, the serial number is the Unique Device Identifier (UDID), along with the ID number of the IoT device and the new authentication parameters. So each time a user wants to access an IoT device, it will use the android/ios application that will directly communicate with the server to check the credentials. If everything is correct, it will communicate both with the smartphone to notify the success of the authentication operation and with the IoT device to unlock it.

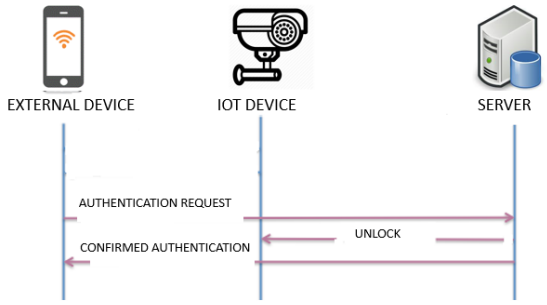

Fig. 6. Authentication process

3) Using One Time Password: The third solution is more convenient and cost-effective than the previous ones, as it does not inflict any additional cost on the IoT device, because the external device does all the computational work. The only weakness in the system is in the first authentication phase, in which the credentials are default credentials and can be easily predicted by an attacker. An initial username and password written in the instruction booklet can be sold with the device to overcome this problem. These can be used only once and only for accessing the application. An even more secure approach is to use a QR code for the first authentication as a One Time Password (OTP). Moreover, the frequency of the OTP based authentications could be optimised to improve the usability of this approach.

IV. Conclusion

We discussed how vulnerabilities of IoT devices can be exploited by a class of malware called Mirai, which creates a botnet of IoT devices. We presented a detailed analysis of the Mirai malware source code, and we implemented the Mirai attack using the same code, that is available on the Github. Our conclusion was that the attack is still very relevant and that resource-constrained IoT devices are vulnerable to it. We reviewed existing security solutions, whose take up in practice presents a number of difficulties, and we proposed three new ones, that provide security without increasing the manufacturing cost of the devices. Our future research will focus on validating these solutions by means of extensive experimentation.

References

[1] M. A. et. al, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

[2] “Mmd-0052-2016 - overview of ”skidddos” elf++ irc botnet,” https://blog.malwaremustdie.org/2016/02/mmd-0052-2016-skidddos-elf-distribution.html, February 2016.
[3] “Security man krebs’ website ddos was powered by hacked internet of things botnet,” https://www.theregister.com/2016/09/26/brian_krebs_site_ddos_was_powered_by_hacked_internet_of_things_botnet/, September 2016.
[4] “Inside the infamous mirai iot botnet: A retrospective analysis,” https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/, September 2016.
[5] “Mirai-source-code,” https://github.com/jgamblin/Mirai-Source-Code, accessed: 2010-02-20.
[6] “Ddos attack that disrupted internet was largest of its kind in history, experts say,” https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, October 2016.
[7] “Did the mirai botnet really take liberia offline?” https://krebsonsecurity.com/2016/11/did-the-mirai-botnet-really-take-liberia-offline/, November 2016.
[8] “Uk isp talktalk confirm loss of 101,000 subscribers after cyber-attack,” https://www.ispreview.co.uk/index.php/2016/02/isp-talktalk-suffers-sharp-fall-in-broadband-users-to-3-9-million.html, December 2016.
[9] “Kaspersky lab research shows ddos devastation on organizations continues to climb,” https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-research-shows-ddos-devastation-on-organizations-continues-to-climb, February 2017.
[10] “Source code of iot botnet satori publicly released on pastebin,” https://www.trendmicro.com/vinfo/it/security/news/internet-of-things/source-code-of-iot-botnet-satori-publicly-released-on-pastebin, January 2017.
[11] “Hajime malware: How does it differ from the mirai worm?” https://www.techtarget.com/searchsecurity/answer/Hajime-malware-How-does-it-differ-from-the-Mirai-worm, March 2017.
[12] “Iotroop botnet: The full investigation,” https://research.checkpoint.com/2017/iotroop-botnet-full-investigation/, October 2017.
[13] “Mirai okiru: The first new linux elf malware designed to infect arc cpus,” https://securityonline.info/mirai-okiru-the-first-new-linux-elf-malware-designed-to-infect-arc-cpus/, January 2018.
[14] “Mirai-based masuta botnet weaponizes old router vulnerability,” https://www.securityweek.com/mirai-based-masuta-botnet-weaponizes-old-router-vulnerability, January 2018.
[15] “Jenx: A new botnet threatening all,” https://www.radware.com/security/ddos-threats-attacks/threat-advisories-attack-reports/jenx/, January 2018.
[16] “Omg - mirai minions are wicked,” https://www.netscout.com/blog/asert/omg-mirai-minions-are-wicked, January 2018.
[17] “Wicked botnet uses passel of exploits to target iot,” https://threatpost.com/wicked-botnet-uses-passel-of-exploits-to-target-iot/132125/, June 2018.
[18] “Open adb ports used to spread possible satori variant,” https://www.trendmicro.com/en_gb/research/18/g/open-adb-ports-being-exploited-to-spread-possible-satori-variant-in-android-devices.html, July 2018.
[19] “Torii botnet, probably the most sophisticated iot botnet of ever,” https://securityaffairs.co/wordpress/76659/malware/torii-iot-botnet.html, September 2018.
[20] “Thinkphp vulnerability abused by botnets hakai and yowai,” https://malware.news/t/thinkphp-vulnerability-abused-by-botnets-hakai-and-yowai/26724, January 2019.
[21] “Mirai “covid” variant disregards stay-at-home orders,” https://www.f5.com/labs/articles/threat-intelligence/mirai-covid-variant-disregards-stay-at-home-orders, March 2020.
[22] “Satori: Mirai botnet variant targeting vantage velocity field unit rce vulnerability,” https://unit42.paloaltonetworks.com/satori-mirai-botnet-variant-targeting-vantage-velocity-field-unit-rce-vulnerability/, March 2021.
[23] “Satori: Mirai botnet variant targeting vantage velocity field unit rce vulnerability,” https://unit42.paloaltonetworks.com/satori-mirai-botnet-variant-targeting-vantage-velocity-field-unit-rce-vulnerability/, February 2021.
[24] Y. M. P. P. et. al, “Iotpot: Analysing the rise of iot compromises,” in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Washington, D.C.: USENIX Association, 2015. [Online]. Available: https://www.usenix.org/conference/woot15/workshop-program/presentation/pa
[25] H. Šemić and S. Mrdovic, “Iot honeypot: A multi-component solution for handling manual and mirai-based attacks,” in *2017 25th Telecommunication Forum (TELFOR)*, Nov 2017, pp. 1–4.
[26] J. Habibi, D. Midi, A. Mudgerikar, and E. Bertino, “Heimdall: Mitigating the internet of insecure things,” *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 968–978, Aug 2017.
[27] H. Tanaka, S. Yamaguchi, and M. Mikami, “Quantitative evaluation of hajime with secondary infectivity in response to mirai’s infection situation,” in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019, pp. 961–964.
[28] Z. Ahmed, I. Nadir, H. Mahmood, A. Hammad Akbar, and G. Asadullah Shah, “Identifying mirai-exploitable vulnerabilities in iot firmware through static analysis,” in *2020 International Conference on Cyber Warfare and Security (ICCWS)*, 2020, pp. 1–5.
[29] X. Hu, R. Sun, K. Xu, Y. Zhang, and P. Chang, “Exploit internal structural information for iot malware detection based on hierarchical transformer model,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 927–934.
[30] M. Snehi and A. Bhandari, “Apprehending mirai botnet philosophy and smart learning models for iot-ddos detection,” in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2021, pp. 501–505.
[31] A. R. S. Araujo Cruz, R. L. Gomes, and M. P. Fernandez, “An intelligent mechanism to detect cyberattacks of mirai botnet in iot networks,” in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2021, pp. 236–243.
[32] J. A. Jerkins, “Motivating a market or regulatory solution to iot insecurity with the mirai botnet code,” in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–5.
[33] “Mirai (ddos) source code review,” https://medium.com/@cjbarker/mirai-ddos-source-code-review-57269c4a68f, February 2016.
[34] “Shodan,” https://www.shodan.io/, accessed: 2010-02-20.
[35] D. Boneh, *Blum–Blum–Shub Pseudorandom Bit Generator*. Boston, MA: Springer US, 2005, pp. 50–51. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_37
[36] “Oauth 2.0,” https://nordicapis.com/why-oauth-2-0-is-vital-to-iot-security/, March 2017.

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0