An Integrated Architecture for IoT Malware Analysis and Detection

Zhongjin Liu[1(✉)], Le Zhang[2], Qiuying Ni[2], Juntai Chen[2], Ru Wang[2], Ye Li[3], and Yueying He[1]

[1] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China
lzj@cert.org.cn
[2] Beijing University of Posts and Telecommunications, Beijing, China
[3] Beihang University, Beijing, China

Abstract. Along with the rapid development of the IoT, the security issue of the IoT devices has also been greatly challenged. The variants of the IoT malware are constantly emerging. However, there is lacking of an IoT malware analysis architecture to extract and detect the malware behaviors. This paper addresses the problem and propose an IoT behavior analysis and detection architecture. We integrate the static and dynamic behavior analysis and network traffic analysis to understand and evaluate the IoT malware's behaviors and spread range. The experiment on Mirai malware and several variants shows that the architecture is comprehensive and effective for the IoT malware behavior analysis as well as spread range monitoring.

Keywords: IoT malware · Behavior analysis · Mirai · Architecture

1 Introduction

With the rapid development of the Internet of things (IoT) technology, the number of smart devices has increased greatly. According to analysis reports, the number of smart devices (excluding smart phones, tablets and computers) will grow to 28.1 billion by 2020. The IoT devices have been used in a wide range of applications, such as smart grid, intelligent transportation, intelligent home, etc.

Along with the rapid development of the IoT, the security issue of the IoT devices has also been greatly challenged [1]. The vulnerability of IoT devices have drawn hackers' attention, especially for those who are interested in DDoS attacks. Considering the significant number of devices, IoT has been gradually becoming one of the weakest part of the computer network. The spread and evolution of the IoT malware, such as botany, worm, and malicious software, are both speeding up.

The most famous IoT malware "Mirai" has been used in some of the largest and most disruptive distributed denial of service (DDoS) attacks [2]. Since the source code of Mirai was published on the Internet, it has become an architecture for building new malware. The variants of the Mirai are constantly emerging.

However, there is lacking of an IoT malware analysis architecture to extract and detect the malware behaviors, which mainly lies in three aspects:

B. Li et al. (Eds.): IoTaaS 2018, LNICST 271, pp. 127–137, 2019.
https://doi.org/10.1007/978-3-030-14657-3_14

(1) Network security researchers have carried out extensive and in-depth research on PC and mobile malware analysis. Since IoT devices run embedded systems on different hardware architectures and platforms, such as ARM, PowerPC, etc. There is lacking of a cross-platform architecture for analyzing the malicious code.
(2) The analysis of malware is usually carried out case by case. These work are independent and short of association analysis. Therefore, there is lacking of an architecture for comparative analysis of different malware variants.
(3) Malware network behavior monitoring is an effective method for malware detection. How to extract network behaviors of IoT malware and detect them on wide area of Internet still needs to study.

A survey [3] show that The detection patterns used in static analysis include string signature, byte-sequence n-grams, syntactic library call, control flow graph and opcode (operational code) frequency distribution etc. In view of this problem that the extracted opcode sequences cannot represent the true behaviors of an executable. Ding presents a control flow-based method to extract executable opcode behaviors [4]. But they don't have much concern about the key behavior in malicious code, however the key behavior can reflect malicious behavior.

Researchers have less dynamic analysis of IoT malware and more dynamic analysis of mobile mal-ware. Because dynamic analysis tools require intensive computation power, which are inadaptable to IoT devices due to the resource-constraint problem [5]. Moreover, most advanced analysis techniques are highly dependent on the underlying system platform. Building these analysis techniques require ad-hoc development for different platforms in the diversified IoT environments [6, 7]. Besides it is difficult to make dynamic analysis due to the trouble in applying it in an actual environment and be-cause of the overhead of tracking data flow to a low level [8].

In [9, 10], techniques to cluster network traffic patterns associated with botnets are presented. The characteristics observed include the flow patterns between hosts, such as the number of connections and amount of data exchanged. Similarly, using machine learning and network traffic features, [11] presents an approach to detect malware related traffic.

Researchers have proposed a few approaches [12, 13] to detect the existence of botnets in monitored networks. Almost all of these approaches are designed for detecting botnets that use IRC or HTTP based C&C, but these solutions lack correlation analysis of traffic from two dimensions of time and space.

This paper proposes an integrated analysis architecture for the IoT malware. The architecture integrates static analysis, dynamic analysis, variant evolution analysis and network behavior detection to achieve the goal of extracting and detecting IoT malware in the large-scale network.

2 Architecture Design

To solve the problem, we propose an architecture to analyze the IoT malware behaviors in static and dynamic manner. Based on the behavior, we focus on the network behavior analysis and intend to detect IoT malware in large-scale network (Fig. 1).

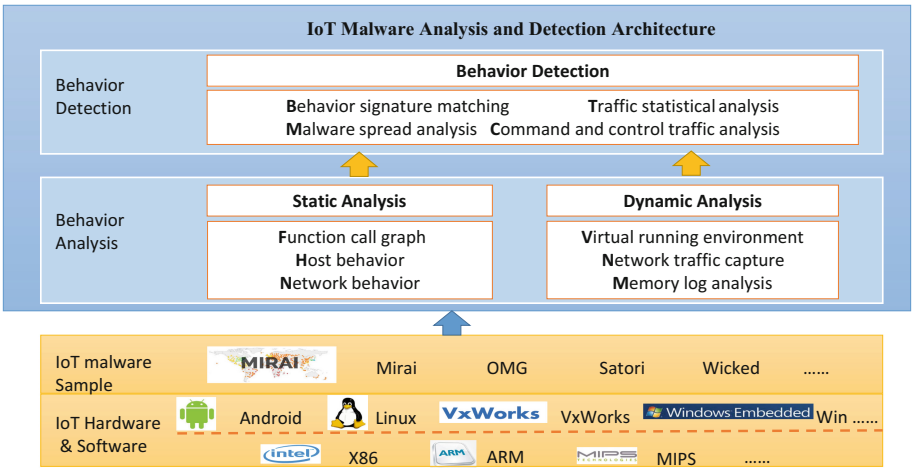

Fig. 1. IoT malware analysis and detection architecture

The architecture consists of two layer: Behavior Analysis layer and Behavior Detection layer.

The Behavior Analysis layer consists of two module, the static analysis module extracts function call graph, host behavior and network behavior by code reverse analysis; the dynamic analysis module captures network traffic and analyze memory log by running code in the virtual environment.

The Behavior detection layer matches the malware traffic with behavior signature. Based the traffic, the module is able to analysis the malware spreading and traffic pattern between bots and Command and control server (C&C).

3 Design Details

Based on the proposed architecture, we propose the method to derive and detect the static and dynamic behavior of different IoT malware. The analysis process is shown in Fig. 2.

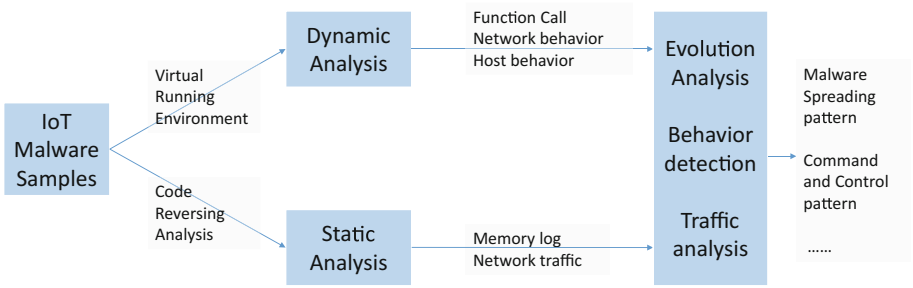

Fig. 2. The malware analysis process

Internet of things botnets virus attacks through more Internet of things devices, and traffic is huge in every attack. With the "Mirai" botnet attack as an example, we have a complete understanding of the embedded system attack on its static analysis, dynamic analysis, evolution analysis and traffic analysis, and our overall analysis process is as follows:

3.1 Static Analysis

The static analysis can extract the function call diagram, host-side behaviors and network-side behaviors, which helps to analyze the malicious code accurately. We are intend to analysis the malware from 3 aspects:

(1) **Extracting the function call graph.** Malicious code has a lot of behaviors, but in the process of malicious code variety, some specific function call processes are similar or even remain the same, such as: turning off the order of some processes in the host, occupying the resource of the system, and sending the specified information to the network server. Through the analysis of the function call process, it can be more easily determined from the whole structure of malicious code, and can provide the basis for the process of dynamic analysis to achieve the function of common verification.

(2) **Extracting the behaviors of host.** The host may be forced to perform some behavior after the host is infected with malicious code, which may prevent the running of the normal function of the host or open the new function without affecting the normal function of the host. The form or naming of malicious code may change, but the behavior of the final implementation will not change. The analysis of the behavior of the host end can help us correctly judge the behavior similarity and make up for the shortcomings of the simple analysis of the multiplexing function. In Mirai, for example, the host's behaviors: closing the watchdog process, preventing host to clear malicious code during reboot, closing the specific port and taking up it, deleting a specific file and kill corresponding to the process.

(3) **Extracting the behaviors of network.** The attacker in order to quickly infect the IoT equipment and effectively control of the host, the host usually performs certain specific network behavior operations, these network behavior operations maintain the connection between the attacker and the host, and the spread of the malicious code. Taking Mirai as an example, the behaviors of the network: the scanner module scans other potentially infected devices, and reports the infected devices to reach the goal of expanding the botnet; regularly sending messages to the C&C server, this host has been identified and stay active; upload and download some files.

3.2 Dynamic Analysis

The sandbox is a virtual environment which is often used to execute untested or untrusted programs or code without risking harm to the host machine or operating system. The sandbox is a powerful tool for dynamic analysis of IoT malware. However,

IoT malware has compilation formats for various platforms. Current sandbox is not able to run IoT malware directly.

We intend to handle this problem in two ways: First, run malicious code based on x86 sandbox directly by source code cross-platform compilation; second, integrate virtual machine, such as qemu, into sandbox, to simulate an embedded running environment for non-x86 instruction sets.

3.3 Evolution Analysis

Based on the extracted features from static and dynamic result, we are able to carry out evolution analysis. By comparing the extracted features of different malwares, such as function calls, traffic pattern, configuration table, we are able to analyze the evolution of the IoT malwares.

From the static result, take Mirai for example, if a set of functions that exist in Mirai sample also exists in the variant samples, then the Mirai and variant may belong to the same malware family. Therefore, we compare the similarity of different samples based on the reusable library. When we get a variant sample, we calculate the similarity of the function in the reuse function library, then divide the samples into different families.

For the dynamic result, we compared the similarities and differences between IoT malware by comparing host behaviors and network behaviors.

3.4 Detection and Traffic Analysis

Based on the extracted behaviors, we match the network traffic data by source IP, destination IP, source port number, destination port number and data packet signature in large-scale network. We can further extract relevant information based on these network traffic data.

We perform statistical analysis on network traffic data, including statistics on port numbers and the geographical situation of infected bots. Further, we perform cluster analysis on the Network traffic data and measure the similarities between different data sources to find out if there are multiple different variants in the data.

We analyze the behaviors of different variants from space and time dimensions, including the scope, time distribution, propagation method, and communication behavior, and then assess the degree of harm of different variants.

4 Experiment and Results

To evaluate the architecture, we integrate many tools and develop the entire system based on these tools. For static analysis, we use IDA disassembly tools to analyzes malicious code and extract key behaviors of malicious code; for dynamic analysis, we use cuckoo sandbox to extract the characteristics of host behaviors and traffic. To make comparison of different variant of malware, we obtain and test Mirai, satori, OMG and Wicked samples in the experiment. Furthermore, we capture malware related traffic in the large-scale network based on the static and dynamic signatures. The result is shown below.

4.1 Static Analysis

(1) Function flow chart analysis

The function control flow chart can be obtained by static disassembly of the Mirai code through IDA and other disassembler tools. As shown in the Fig. 3, we obtain the following modules.

Bot main module: Firstly, anti-debugging, disabling watchdog function to prevent the device restart remove malicious files, then making sure that an independent example is running in the device, then opening the killer module, attack module, scanner module, finally, keeping links between BOT and CNC server, accepting and sending necessary information.

Kill module: Closing the specific ports and occupying, deleting specific files and killing corresponding processes to achieve the function of occupying resources alone.

Attack module: Parsing attacked command and launching Dos attack.

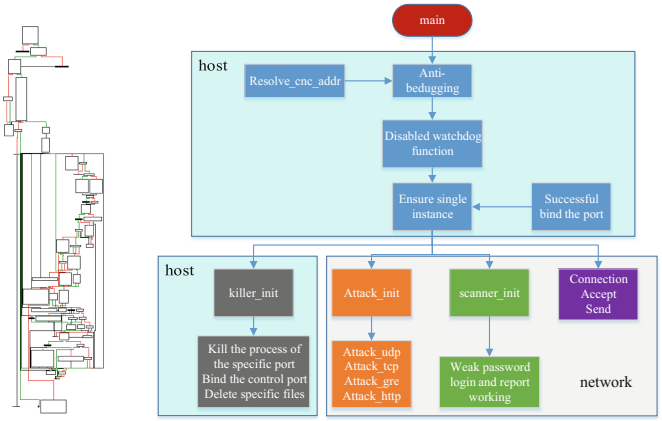

Fig. 3. Functions of the Mirai code

Scanner module: When bot scan other potentially infected devices, if bot can log in to a new bot with a weak password, the successful result will be reported to CNC server. The botnet expands rapidly through this pattern.

Connect, accept, and send module: Realizing links between BOT and CNC server, accepting and sending necessary information.

(2) Key behavioral operations

Each function represents the different functions, the key functions of many have the malicious behavior, and the key functions are divided into two classes: host behavior and network behavior, which can be better combined with dynamic analysis and flow analysis. The results of the classification are shown in the following Table 1:

Table 1. The results analyzed by IDA

Class	Function name	Means
Host behavior	anti_gdb_entry()	anti-debugging
	close_watchdog()	disabling watchdog
	ensure_single_instance()	ensure single instance
	killer_init()	killer module initialize
	bind()	bind the specific ports
	kill()	kill the specific process
Network behavior	attack_init()	attack module initialize
	scanner_init()	scanner module initialize
	establish_connection()	establish connection
	teardown_connection()	tear down connection
	accept()	accept messages from CNC
	send()	send messages to CNC

4.2 Dynamic Analysis

In the sandbox setting, both host and guest machine (virtual machine) use Ubuntu system. The sandbox is installed on the host machine and the recompiled IoT malware runs in the virtual machine environment. We analyzed the malicious code of several Mirai botnet variants and obtained the following results (Table 2).

Table 2. The results analyzed by sandbox

Source	Destination	Protocol	Source port	Destination port	Package number
192.168.56.101	224.0.0.251	MDNS	5353	5353	23
192.168.56.101	192.168.56.255	BJNP	8612	8612	16
192.168.56.101	192.168.56.255	BJNP	8612	8610	16

Analysis the packages of BJNP protocol (network protocol used by Canon printers and scanners), the malicious code can discover the network printers and scanners within the local area network (LAN) through send BJNP protocol discovery request package to broadcast address. Then find printers and scanners support BJNP protocol.

It is able to launch attacks to printers and scanners, such as Exhaustion of print consumables attack; read print log, access to private information; printer configuration changes (if the network printer has changed the administrator password, theoretically could firstly send the content to the attacker machine then send the content to the severs' printers and scanners by ARP cheat).

4.3 Evolution Analysis

According to the results of the static and dynamic analysis of the malware on the IoT, we analyzed the similarity and differences between Mirai and the Mirai variants (Table 3).

Table 3. The comparison between Mirai and variants

Name	Feature	For Mirai	For variants
Satori	Transmission Mode	A Telnet scanner component is downloaded in an attempt to scan to identify vulnerable devices and use the Mirai Trojan to infect after infecting an IoT device	Two embedded vulnerabilities are exploited in an attempt to infect remote devices connected to ports 37215 and 52869 in-stead of using the scanner component
	Target device	Scanning ports 2323 and 23	Connected to ports 37215 and 52869
OMG	Configuration table	Include killing processes, Telnet brute force logins, and launching DDoS attacks	Setting up a firewall to allow traffic to penetrate two random ports
IoTroop	Exploitation	Using the default credentials of the IoT device	Exploiting a wider range of vulnerabilities to target a wider range of products
	DDoS loader	Placing a Mirai-style DDoS engine on the device	Placing a loader that constantly communicates with the C2 server
Wicked	Persistence	Cannot persist and keep on device after restart	Downloading payloads on demand from C&C servers, and adding code to home router firmware to make malware lasting

4.4 Network Traffic Analysis

We collected traffic data of two IoT malwares. Therefore, we consider the traffic can be divided into two IoT events. Each IoT event is not related to each other. We compare the different port scanning behavior and evaluate the spread area of two malwares.

4.4.1 Port Number Statistical Analysis

We perform statistical analysis on the destination port numbers in the two events to obtain the following information (Fig. 4):

From the figure we can see that the Event 1 mainly scan port 7547, using the remote command execution vulnerability rather than the weak password Trojans are significantly different from the behavior of the previous Mirai [14]. From our statistical data, this variant is in an extremely active state, and the daily record of the active scan source is in the million level. The port number scanned in event 2 is much smaller than the port number in Event 1, and may also be related to the amount of data. Event 2 has the highest proportion of port number 23.

We can figure out that malware 1 prefer 7547 port to spread itself. However, malware 2 use many ports to spread itself, the ports include some widely used ports, such as 23, 22, 2000, 445, 80, 81, 3389, 8545, 2323.

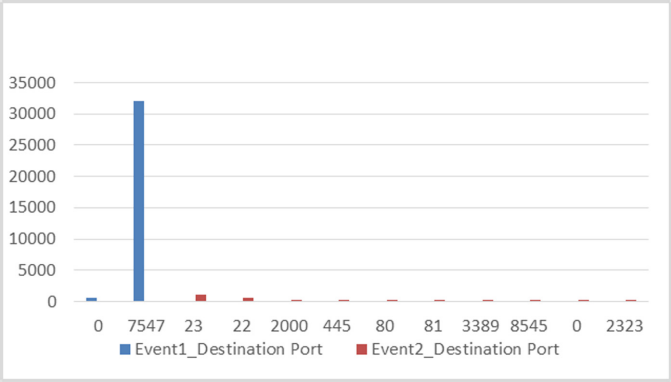

Fig. 4. Destination port numbers of different IoT malwares' interest

4.4.2 Distribution of Infected Areas

The color depth in the picture shows the severity of the infection. Focusing on the variant of scanning port number 7547, we can find that the infection of bots in China is the most serious (Fig. 5).

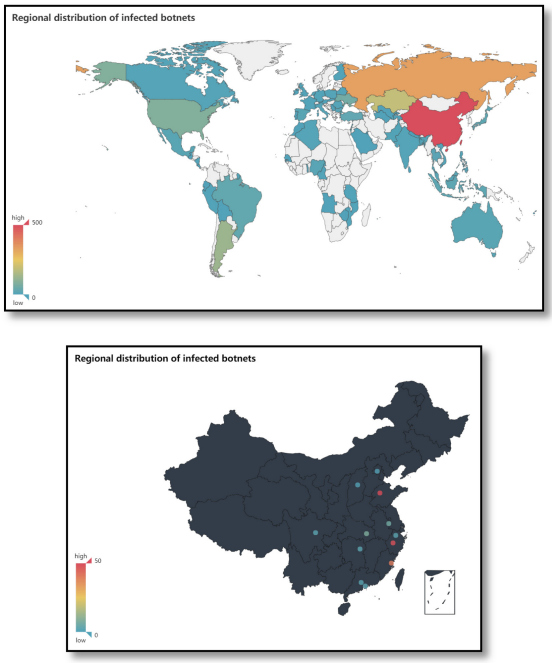

Fig. 5. Regional distribution of the infected botnets

After preliminary analysis, we analyze the distribution of infections in various cities in China. The distribution of data is not evenly distributed, but the focus is on variants of the scanning port number 23. The most serious infection area is in east area of China.

5 Conclusion and Future Work

In this paper, we propose an integrated architecture for IoT malware analysis and detection. By integrating IDA disassembly tool and cuckoo sandbox, we successfully extract the behaviors of the Mirai malware and several variants. We are also able to analysis the evolution and spread of IoT malwares through large-scale network traffic analysis.

Our future work is to make this architecture adapt to more novel variants of the IoT malware. We will apply existing static and dynamic analysis methods to different embedded systems. Simultaneously, we can perform cluster analysis on network traffic data to distinguish different traffic patterns of different variants.

References

1. Jing, Q., et al.: Security of the IoT: perspectives and challenges. Wirel. Netw. **20**(8), 2481–2501 (2014)
2. Kolias, C., et al.: DDoS in the IoT: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
3. Gandotra, E., Bansal, D., Sofat, S.: Malware analysis and classification: a survey. J. Inf. Secur. **5**(02), 56 (2014)
4. Ding, Y., et al.: Control flow-based opcode behavior analysis for Malware detection. Comput. Secur. **44**, 65–74 (2014)
5. Zhang, Z.K., Cho, M.C.Y., Wang, C.W., et al.: IoT security: ongoing challenges and research opportunities. In: IEEE International Conference on Service-Oriented Computing and Applications, pp. 230–234. IEEE (2014)
6. Davidson, D., Moench, B., Jha, S., et al.: FIE on firmware: finding vulnerabilities in embedded systems using symbolic execution. In: Usenix Conference on Security, pp. 463–478. USENIX Association (2013)
7. Zaddach, J., Bruno, L., Francillon, A., et al.: Avatar: a framework to support dynamic security analysis of embedded systems' firmwares. In: Network and Distributed System Security Symposium (2014)
8. Ham, H.S., Kim, H.H., Kim, M.S., et al.: Linear SVM-based android malware detection for reliable IoT services. J. Appl. Math. **2014**(4), 1–10 (2014)
9. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: Proceedings of USENIX Security Symposium (2008)
10. Strayer, W.T., Lapsely, D., Walsh, R., Livadas, C.: Botnet detection based on network behavior. In: Lee, W., Wang, C., Dagon, D. (eds.) Botnet detection. Advances in Information Security, vol. 36, pp. 1–24. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-68768-1_1

11. Bekerman, D., Shapira, B., Rokach, L., Bar, A.: Unknown malware detection using network traffic classification. In: Proceedings of IEEE Conference on Communications and Network Security (CNS) (2015)
12. Binkley, J.R., Singh, S.: An algorithm for anomaly-based botnet detection. In: Proceedings of USENIX SRUTI 2006, pp. 43–48, July 2006
13. Edwards, S., Profetis, I.: Hajime: analysis of a decentralized internet worm for IoT devices. Rapidity Netw. (2016)
14. Li, F.: Blog. https://blog.netlab.360.com/a-few-observations-of-the-new-mirai-variant-on-port-7547/