# ThingGate: A gateway for flexible operation of bare-metal IoT honeypot

Chun-Jung Wu[†], Katsunari Yoshioka[†, ††], and Tsutomu Matsumoto[†, ††]

***Abstract***: Internet of Things (IoT) malware keep evolving and exploit multiple vulnerabilities to infect IoT devices. Besides malware, human attackers also utilize various tools to access and collect variable information on the devices For instances, web UI of IP Cameras and routers are constantly searched and accessed if vulnerable. In order to observe and analyze such variety of attacks in depth, there is an increasing need for bare-metal IoT devices as a honeypot since it is costly to emulate device-specific vulnerabilities and complex functionalities from their dedicated services. However, operating bare-metal IoT honeypots has unique technical challenges mostly coming from their low configurability as an embedded system. A bare-metal honeypot needs proper access control while it is allowing attackers to access its inside to some degree, such as filter out bricking commands and changes of critical configuration. From this observation, we propose ThingGate, a gateway for flexible operation of bare-metal IoT honeypot. ThingGate employs a man-in-the-middle proxy to control and manage inbound and outbound traffic of the bare-metal IoT honeypot. We evaluate ThingGate with seven bare-metal IoT devices and show that it successfully blocks unwanted incoming critical attacks, masks wireless access point information of the devices while showing high observability of various attacks exploiting different vulnerabilities.

***Keywords***: IoT Honeypot, MITM, Transparent proxy, IoT devices

## 1. Introduction

In recent years, people have been connecting various things to the Internet for monitoring, collecting data, or remote manipulation. Backend applications collect and exchange data with these devices through the network. The network of this appearance is called the Internet of Things (IoT). However, an IoT Malware “Mirai” was used for conducting the massive Distributed Denial of Service (DDOS) attack against Dyn DNS. In October of 2016, about 100,000 Mirai IoT botnet nodes were enlisted in this incident and reported attack rates were up to 1.2 Tbps[1]. Therefore, cyber threats from IoT botnet have become a reality. To observe cyber attacks against such devices and analyze the threats from IoT malware, some researchers design new observation mechanisms and build various honeypots such as IoTPOT[2], SIPHON[3], IoTCandyJar[4], and real devices Honeypot for observing Web UI of IoT devices[5].

The competition between hackers and cybersecurity researchers is an endless war. IoT malware keeps evolving and exploits multiple vulnerabilities to infect IoT devices. Since May 2018, the Mirai and Gafgyt malware families that assimilate multiple known exploits affecting the Internet of Things (IoT) devices. These exploits come from 11 makers' devices over HTTP, UPnP, Telnet, and SOAP protocols [6]. a

Besides their well-known activities such as DDoS, recent IoT malware have diverse purposes including coin mining, click fraud, and sending spam emails. Nonetheless, human attackers also utilize various tools to access and collect variable information on the device. For example, WebUI of many IP Cameras and routers are constantly searched and accessed if vulnerable. In order to observe and analyze such variety of attacks in depth, there is an increasing need for a bare-metal IoT honeypot, namely a real IoT device as a honeypot, since it is costly to emulate device-specific vulnerabilities and complicated functionalities provided through their WebUI and other dedicated services, such as UPnP. However, it is worth noting that operating bare-metal IoT honeypots has unique technical challenges mostly coming from their low configurability as an embedded system. For example, honeypot operators may need to control the incoming traffic since there are critical attacks that may destroy firmware and/or change the network configuration of devices that could make the honeypot inoperational. Also, honeypot operators may need to mask and/or replace outgoing responses from the honeypot devices as they may contain information such as surrounding wireless access points, which could reveal the physical location of the honeypot devices.

### 1.1 Research questions

For the honeypot consisted of physical IoT devices, we want to figure out the following research questions.

1. If we build the honeypot with vulnerable devices, how to prevent critical and destructive attack vectors?
2. Some attackers may change the setting of devices which cause functional disorder in devices. Is there a convenient way to prevent it?
3. How to prevent information leak from the WebUI of devices?

† The author is with the Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama 240-8501, Japan
†† The author is with Institute of Advanced Sciences, Yokohama National University, Yokohama, Kanagawa 240-8501, Japan

### 1.2 Overview

The remainder of this paper is divided into six sections. Section 2 presents the literature review of this research. Section 3 clarifies the terminology used in the paper. Section 4 describes our method. Section 5 details the evaluation experiments and results. Section 6 discusses our experiment results. Section 7 briefly concludes our research.

## 2. Related work

In [3], Guarnizo et al. proposed the SIPHON architecture, which is a scalable high-interaction honeypot platform for IoT devices. Our architecture leverages IoT devices physically present at one location and connected to the Internet through so-called wormholes distributed worldwide. The resulting architecture allows the exposure of a few physical devices over numerous geographically distributed IP addresses.

Many embedded devices have WebUI for device management and operation, and some of them are open to the Internet with vulnerability and weak credentials. Ezawa et al. [5] proposed the use of a honeypot to monitor attacks against the WebUI of IoT devices by employing bare-metal devices. The observation results contained attacks against regular web servers and indicated that some attacks are automatically conducted through certain tools or types of malware. The observation also suggests that some attackers changed the DDNS, VPN, and network settings, resulting in the device becoming unavailable for other attackers.

Tamiya et al. [7] employed a decoy honeypot consisted of five IP Cameras to capture the behavior of human-like attackers. His research shows the behavior including extracting environment parameter of devices, downloading the snapshot of live streams, and long-term peeping live streams.

Tambe, et al. [8] developed a scalable VPN-forwarded Honeypots to solve the cloud IP problem in SIPHON. Tambe's honeypot consisted of physical IoT device and prevented Wi-Fi information leak by the physical way. They remote Wi-Fi chip or made use of an electromagnetically shielded enclosure to prevent nearby wireless signals from reaching those devices.

Compared to existing literature, we find the previous honeypot of physical IoT devices lacks abilities against destructive commands. Moreover, only Tambe's research prevented from sensitive information leaks. Our work focuses on the high interaction honeypot consisting of physical IoT devices. Our approach improves the security of the honeypot, including protecting sensitive data collecting by sensors. Besides, our program monitoring and manage the incoming traffic to avoid destructive commands.

## 3. Definitions

### 3.1 Man-in-the-middle

The man-in-the-middle (MITM) refers to an attack in which the attacker positions themselves between two communicating parties and secretly relays or alters the communication between these parties, who believe that they are engaging in direct communication with each other. Messages intended for the legitimate site are passed to the attacker instead, who saves valuable information, passes the messages to the legitimate site, and forwards the responses back to the user. The MITM way can lead to the web proxy attack, in which a malicious web proxy receives all web traffic from a compromised computer and relays it to a legitimate site. The proxy collects credentials and other confidential information from the traffic. MITM flows are difficult to detect because a legitimate site can appear to be functioning properly and the user may not be aware that something is wrong [9]. We utilize a web proxy attack to monitor and manage the flow between clients and our honeypot.

### 3.2 Transparent Proxy

In computer networks, a proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers [10]. A proxy server can fulfill the request from the client, filter out, or modify the request in a specific way. Transparent Proxying or a transparent proxy means we redirect traffic into a proxy at the network layer, without any client configuration [11]. The client is unaware that the response received originates from the proxy server and not from the source server. We conduct the flow forwarding through MITM proxy by `pf` of FreeBSD [12] and `socat` [13].

### 3.3 Cyber Attacks against WebUI of Physical IoT Devices

In 2017, Ezawa et al. [3] propose a Honeypot consisting of physical IoT devices to observe attacks against the WebUI of IoT devices. The devices include IP Cameras, routers, pocket routers, a printer, and a TV receiver. In 2018, Tamiya et al. [7] employed five IP Cameras to build a decoy honeypot to capture the behavior of peeping attackers. According to these two honeypots, we summarized four kinds of cyber attacks against these WebUI:

1. Configuration information theft attacks

   If the device contains vulnerabilities of information disclosure or weak credentials. The attacker can collect the configuration and parameters of devices by some URLs, such as get_status.cgi.

2. Modification of the configuration

   According to the observation of the two honeypots, attackers may modify the DDNS, VPN, credentials, and network configuration.

3. Snapshot attacks

Snapshot is a feature of IP Cameras and offers a current time image of the live stream to users. Once the clients send the HTTP request of the snapshot, the web server will provide the current time image in a JPG or PNG file.

4. Long term peeping

   This attack collected by IP Cameras when some clients access the URL of the live stream. Moreover, the clients stay on the web page of live streams for several hours.

## 4. ThingGate

### 4.1 Goal

ThingGate is a customized MITM proxy for managing flow between clients and the honeypot that consists of phycal IoT devices. To face the challenges from the physical IoT devices, we define the following two goals.

1. Incoming traffic management

   We wish to block the incoming flow of unwanted or deadly attack vectors.

2. Response information management

   Our program checks the HTTP response from IoT devices and prevents the leakage or exposure of sensitive information. Blocking Wi-Fi with an electromagnetic shielding container is costly. We hope to prevent leakage through a simple and light-weight method.

### 4.2 System Overview

Our design, which was inspired by SIPHON architecture [3], is displayed in Figure 1. Our honeypot consists solely of real IoT devices. Moreover, SIPHON's forwarder is improved with an MITM proxy to manage the forward traffic from wormholes to local physical IoT devices. These flows may target IoT devices other than ours.

**Wormhole**. The wormhole device contains some ports open to the general Internet on a public IP address. We transparently forward the incoming traffic toward these ports through an MITM proxy to a specific port on a remote physical IoT device. Forwarding is conducted through `socat` [13], which is a command-line-based utility that establishes two bidirectional byte streams.

**MITM Proxy**. The `socat` utility ensures that the traffic between the wormhole and the IoT device has managed to accomplish the protection and HTTP response rewriting tasks in real time. The proxy examines all the flows and decides to block, delegate to devices. The proxy conducts the modification of the flow through the MITM way.

**Data Storage**. The storage dumps traffic records from the wormholes and aggregates the data for offline analysis. For example, Wireshark is used to analyze the headers of HTTP requests in dumped traffic files.

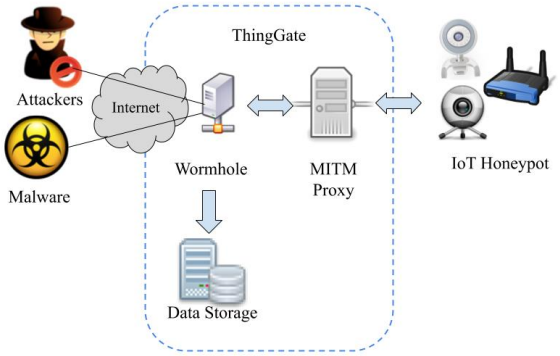


Figure 1 System overview of ThingGate.

### 4.3 Incoming traffic management

For protecting IoT devices from destructive attack or modification configuration, we prepared a blocking list of bricking and critical configuration URL. Figure 2 shows the workflow of filter out unwanted traffic. ThingGate checks incoming HTTP traffic if the URL of the HTTP request is in the blocking list. If the list contains the URL, ThingGate will filter out the form data and replace the URL with another safe URL. Moreover, then send the safe HTTP request to the targeted device.

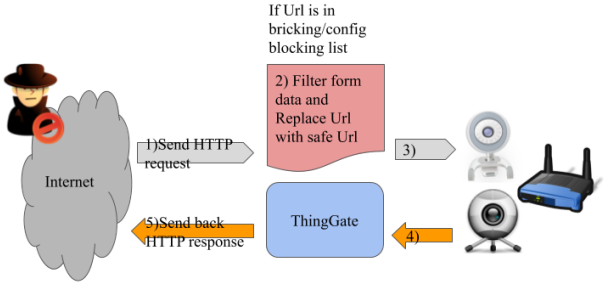


Figure 2 Filtering out unwanted traffic.

### 4.4 Response information management

Many IoT devices contain sensors and show the data in the WebUI. For example, an IP Camera shows the Wi-Fi access point (AP) list in the Wi-Fi setting web page shown in Figure 3. The list exposed not only the mac address of devices but also organization name, models of devices, or maker names.

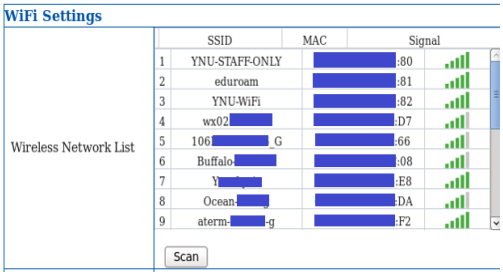


Figure 3 The example of Wi-Fi AP list sensed by IP camera.

In order to prevent information leak from sensed data of the device, ThingGate manages the output traffic of the IoT honeypot. Figure 4 shows the workflow of response information management. If the client visits the web page of Wi-Fi AP List, ThingGate will response a fabricated AP to the client.

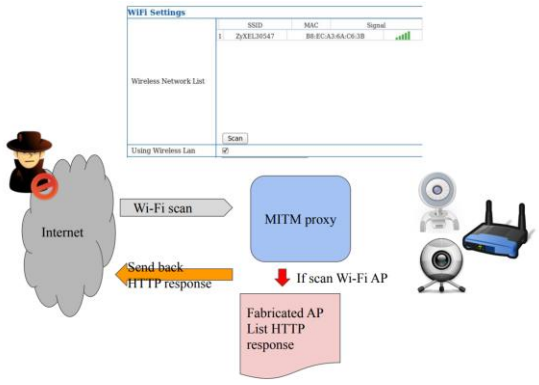


Figure 4 Protecting Wi-Fi information with ThingGate.

## 5. Evaluation

### 5.1 Prototype and data set

We developed a prototype of ThingGate using Python and the MITM proxy open-source software [11]. We performed four different experiments with seven physical IoT devices to evaluate the effectiveness of ThingGate. Table 1 presents the specification of our devices, all of which contained vulnerabilities that had been publicly disclosed. Besides, we installed ThingGate on a server with four cores Intel 3.10 GHz CPU, 16 GB RAM, and 1.8 Terabytes disk.

Table 1 IoT devices used in experiments.

| IoT device | Maker's country | CPU Arch. | price* (JPD) |
|---|---|---|---|
| Router A | Taiwan | MIPS | 26,000 |
| IP Camera A1 | China | ARM | 4,980 |
| IP Camera A2 | China | ARM | 4,980 |
| IP Camera A3 | China | ARM | 4,980 |
| IP Camera B | USA | ARM | 3,000 |
| IP Camera C | Taiwan | MIPS | 14,000 |
| IP Camera D | Taiwan | MIPS | 7,960 |

* We collected this price information from Amazon Japan on Oct. 1, 2018. IP Camera A1 ~A3 are the same mode devices

Table 2 presents the two data sets collected by our honeypot through ThingGate. From September 8 to October 13, 2018, we used seven devices and 32 IP addresses to collect the attack flow (data set 1). Moreover, we summarized the unwanted attack URLs to construct the blocking lists. Moreover, we analyzed the URL list of critical configurations and the URLs of destructive vulnerabilities from our IoT devices. We designed and implemented the prototype of ThingGate according to data set 1. From November 17, 2018, to June 31, 2019, we deployed ThingGate and forwarded 19 IP addresses to conduct the evaluation experiments. The collected flow for this period is labeled as data set 2.

Table 2 Data set for analysis.

| Data set | HTTP requests | honeypot IP | Duration | Analysis or evaluation subjects |
|---|---|---|---|---|
| 1 | 307,405 | 19 | 2018/09/08~ 2018/10/13 | Blocking Lists |
| 2 | 1,920,653 | 19 | 2018/11/17~ 2019/06/31 | Blocking unwanted flow. Fabricated sensor content. |

Table 3 shows the distribution of HTTP methods in data set2. The GET and POST accounted for the vast majority (97 %) which contain various cyber attacks against HTTP services. Moreover, some of the OPTION method flows come from the Real Time Streaming Protocol (RTSP) [14]. The RTSP traffic means some attackers or malware recognized our devices are IP Cameras and want to utilize our RTSP services. Besides, the M-SEARCH and NOTIFY traffic are based on Universal Plug and Play protocol (UPnP) [15]. Our devices disabled the UPnP port and services by default, but the clients try to attack our UPnP service. For the PROFIND flows, the clients blindly sent remote buffer overflow packets which against IIS 6.0 [16].

Table 3 HTTP method statistics for data set 2.

| HTTP method | count | Percentage (100%) |
|---|---|---|
| CONNECT | 420 | 0.022 |
| GET | 1,512,526 | 78.751 |
| HEAD | 7,062 | 0.368 |
| M-SEARCH | 41,961 | 2.185 |
| NOTIFY | 67 | 0.003 |
| OPTIONS | 264 | 0.013 |
| POST | 356,272 | 18.550 |
| PROPFIND | 1,938 | 0.101 |
| PUT | 132 | 0.006 |

Table 4 presents the statistics of HTTP requests, attackers' IP address and URLs. Because we forward fifteen IP address for IP Camera A1, A2, and A3, they got 51/% HTTP requests. However, IP Camera C got the most HTTP requests and URLs on condition

that forwarding only one IP traffic to the device.

Table 4 Statistics of cyber attacks. Observation of 7 months.

| IoT device | Honeypot IP counts | HTTP request counts | Unique attacker IP counts | Unique URL counts |
|---|---|---|---|---|
| Router A | 1 | 17,447 | 1,808 | 6,150 |
| IP Camera A1 | 5 | 340,316 | 22,336 | 2,300 |
| IP Camera A2 | 5 | 455,639 | 23,196 | 4,546 |
| IP Camera A3 | 5 | 193,941 | 13,642 | 1,573 |
| IP Camera B | 1 | 54,024 | 4,581 | 1,740 |
| IP Camera C | 1 | 782,645 | 4,291 | 2,8111 |
| IP Camera D | 1 | 76,641 | 4,422 | 1,395 |
| Total | 19 | 1,920,653 | 57,230 | 38,374 |

## 5.2 Blocking unwanted flow experiments

### 5.2.1 Design of experiment

We analyzed our devices and created a list of configuration URLs and destructive vulnerabilities. There 51 critical configuration URLs and one destructive URL in the list, and we extract the pathname of configuration URLs to build a blacklist. Further, we select target pages in devices for replacing the pathname in the blacklist. Table 5 presents the blacklist against IP Camera A1~A3. Moreover, we deployed the blacklist in ThingGate and redirected flow to the target pages if the incoming traffic matched the blacklist. The flow of one IP address was forwarded to all devices except for the three IP Cameras.

Table 5 Configuration blacklist and replaced pathnames against IP Camera A1~A3.

| Configuration pathname | Description of pathname | Replaced pathname | Description of pathname |
|---|---|---|---|
| /set_network.cgi | change network settings | /admin2.htm | show camera status |
| /reboot.cgi | reboot camera | /admin2.htm | show camera status |
| /set_upnp.cgi | change UPnP settings | /upnp.htm | show UPnP information |
| /set_wifi.cgi | set Wi-Fi network | /wireless.htm | show Wi-Fi settings |
| /set_ddns.cgi | change dynamic DNS settings | /ddns.htm | show dynamic DNS settings |
| /set_users.cgi | change user settings | /user.htm | show user account settings |
| /restore_factory.cgi | restore factory settings | /upgrade.htm | show upgrade functions |
| /upgrade_htmls.cgi, | upgrade system firmware | /upgrade.htm | show upgrade functions |
| /upgrade_htmls.cgi | upgrade WebUI | /upgrade.htm | show upgrade functions |

### 5.2.2 Experimental results

From data set 2, we found on June 7th, an attacker from American accessed our Wi-Fi router in the honeypot and modified the LAN DNS setting, point to a Vietnam server. ThingGate successfully blocked this HTTP request, filtered out the form data, and replace the URL with another URL in WebUI. Figure 5 shows the detail information of the HTTP request.

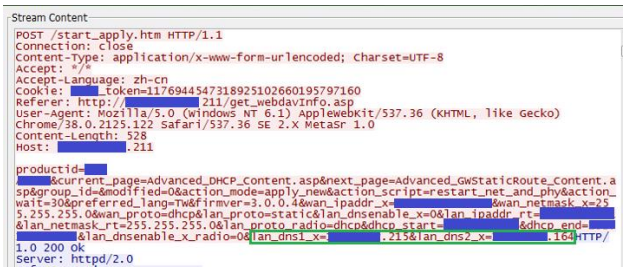


Figure 5 The HTTP request of a modifying configuration attack.

## 5.3 Fabricated sensor information experiment

### 5.3.1 Design of experiment

We selected the WebUI of all of the IP Cameras as victim devices that we would like to protect. ThingGate monitored the flow of 18 IPs forwarded to these cameras. If clients requested the web page of scanning Wi-Fi information, we replaced the information with fabricated information.

### 5.3.2 Experimental results

In data set 2, we found that ThingGate successful sent fabricated Wi-Fi information to 44 different clients in 80 HTTP response. Table 6 presents part of the attackers' geographical location, number of requests sent, and duration of visit to our honeypot. The Googlebot [17] client only sent 23 HTTP requests in one day.

Table 6 part of attackers who request Wi-Fi information. Observation of 7months from IP Camera A1~A3, B, C, and D.

| clients | Source IP | Country | Requests for Wi-Fi | Total request count | Attack Duration |
|---|---|---|---|---|---|
| Client A | aaa.aaa.202.28 | USA | 3 | 2704 | 2018/12/12~201 |

| | | | | | |
|---|---|---|---|---|---|
| Client B | bbb.bbb.169.38 | USA | 3 | 4167 | 8/12/12<br>2018/12/23~2018/12/23 |
| Client C | ccc.ccc.226.5 | USA | 6 | 3476 | 2018/12/17~2019/01/12 |
| Client D | ddd.ddd.89.58 | China | 1 | 537 | 2019/01/11~2019/01/11 |
| Client E | eee.eee.148.116 | China | 3 | 2333 | 2018/11/19~2018/11/19 |
| Client F | fff.fff.15.51 | France | 3 | 98 | 2019/01/07~2019/01/08 |
| Client G | ggg.249.79.85* | USA | 1 | 23 | 2018/11/17~2018/11/17 |

*The clients G is Googlebot

### 5.4 Observation of cyber attacks against the WebUI

According to data set 2, there are 1,920,653 cyber attacks employed HTTP requests to attack our honeypot. Some of these attacks are only able to be observed by physical devices. We collected similar attacks which present in Ezawa's and Tamiya's honeypot [5] [7]. We found attackers attempt to capture and modify the configuration of devices, remotely control direction and zoom of IP Camera, peep the live video, snapshot of IP Camera and utilized the remote code execution (RCE) vulnerability of devices [18]. In addition, after the RCE attack vector, the attacker download devices' live stream by access a hidden web application. The application "/video.cgi"did not appear in source code and can be customized by width and height parameters. Table 7 shows the statistic and description of the attack against our physical device.

There were 49 source IPs has watched the live stream of the camera. Among them, five IPs were peeking over an hour. The maximum time of peeking is about 18 hours. Moreover, some clients from 21 source IP addresses adjusted the directional and zoom of the camera. One client from American applied the RCE exploit code of IP Camera C to attack IP Camera C and D. The Live stream for long term peeping, the real-time response of control direction and zoom, and the whole scenario of RCE attack are hard to simulate by VM-based honeypot. Our physical devices behind ThingGate successfully observed these kinds of cyber attacks.

Table 7 Cyber attacks against WebUI of IoT devices. Observation of 7months from IP Camera A1~A3, B, C, and D.

| Category | Pathname | Description of URL | Victim devices | Request counts |
|---|---|---|---|---|
| Configuration information theft attacks | get_params.cgi | Show system variables | IP Camera A1~A3, B | 599 |
| | get_status.cgi | Show configuration of devices | IP Camera A1~A3, B, C, D | 1064 |
| modification of the configuration | /%5ccgi-bin/set_network.cgi | Set network configuration | IP Camera A3 | 83 |
| | decoder_control.cgi | Control directional and zoom | IP Camera A1~A3 | 153 |
| Snapshot attacks | snapshot.cgi | Show current image of live video stream | IP Camera A1~A3, B | 2,920 |
| Long term peeping | livestream.cgi | Show live video stream | IP Camera A1~A3, | 4560 |
| | videostream.cgi | Show live video stream | IP Camera A1~A3, B | 273 |
| Remote Command Execution | /setSystemCommand | Set OS Commands for execution | IP Camera C, D | 4 |

## 6. Discussion

From the observation of cyber attacks in data set 2, our honeypot successfully collected attacks against physical IoT devices Through ThingGate. These attacks, such as peeping video streams, control the direction of the camera, and RCE attacks first and then download live stream via hidden web application are hard to simulate by the virtual machine. We also found 44 clients request the Wi-Fi AP information web page from the fabricated sensor information experiment. From the 44 clients, 41 clients employed a predefined list to scan victims, two are human-like attackers, and Googlebot. Googlebot is web crawler software developed by Google, and it analyzed the JavaScript code in WebUI, traverse all web pages including the Wi-Fi configuration page. We verified Googlebot by using a reverse DNS lookup on the accessed IP address according to a Google document [19]. The results of observation show that many attackers collected Wi-Fi AP information from WebUI of IoT devices.

Compared with Tambe's research, they applied the physical way that remoted Wi-Fi chip or made use of an electromagnetically shielded enclosure. ThingGate is much easier and cheaper. However, there is a risk that attackers get root privilege and open terminal to scan Wi-Fi though command-line interface.

According to the result of block unwanted traffic experiment, we prove ThingGate can prevent devices from modifying the configuration. Hence, we can build the bare-metal IoT honeypots together with ThingGate to increase the efficiency and safety of

the honeypot.

## 7. Conclusion

We employ the transparent proxy to develop a supporting mechanism for honeypot consisted of physical IoT devices. ThingGate can improve the security of honeypot, manage the incoming traffic and output response content via MITM way. We evaluated ThingGate on the public internet, prove the effectiveness of ThingGate. In our observation, ThingGate did not yield the cyber attacks against physical devices, such as RCE attack and long term peeping. In our experimental result, we successfully defended the unwanted attack which changes configuration. Moreover, ThingGate fooled 44 clients who requested the Wi-Fi AP list in WebUI with fabricated AP.

## Acknowledgments

A part of this work was funded by MSec project, supported by the National Institute of Information and Communications Technology (NICT).

## Reference

[1] P., Loshin, “Details emerging on Dyn DNS DDoS attack, Mirai IoT botnet,” TechTarget network, http://searchsecurity.techtarget.com/ news/450401962/Details-emerging-on-Dyn-DNS-DDoS-attack-Mirai-IoT-botnet, accessed Feb. 06. 2019.

[2] Y.M.P., Pa, S., Suzuki, K., Yoshioka, T., Matsumoto, T. Kasama, and C., Rossow, “IoTPOT: A Novel Honeypot for Revealing Current IoT Threats,” Journal of Information Processing, Vol.24, No.3, pp.522–533, 2016.

[3] J. D., Guarnizo, A., Tambe, S. S., Bhunia, M., Ochoa, N. O., Tippenhauer, A., Shabtai, & Y., Elovici, “Siphon: Towards scalable high-interaction physical honeypots,” In Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security, pp. 57-68, April, 2017.

[4] T., Luo, Z., Xu, X., Jin, Y., Jia, & X., Ouyang, "Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices." Black Hat, 2017.

[5] Y., Ezawa, K., Tamiya, S., Nakayama, Y., Tie, K., Yoshioka, and T., Matsumoto, “An Analysis of Attacks Targeting WebUI of Embedded Devices by Bare-metal Honeypot,” In Computer Security Symposium 2017, Oct., 2017.

[6] R, Nigam, “Unit 42 Finds New Mirai and Gafgyt IoT/Linux Botnet Campaigns,” Unit42, https://unit42.paloaltonetworks.com/unit42-finds-new-mirai-gafgyt-iotlinux-botnet-campaigns/, accessed Feb. 06. 2019.

[7] K., Tamiya, Y., Ezawa, Y., Tie, S., Nakayama, K., Yoshioka, and T., Matsumoto, "Observation of Peeping using Decoy IP Camera," In Symposium on Cryptography and Information Security 2018, Jan., 2018.

[8] A., Tambe, L.A., Yan, R., Sridharan, M., Ochoa, N.O., Tippenhauer, A., Shabtai, and Y., Elovici. "Detection of Threats to IoT Devices using Scalable VPN-forwarded Honeypots." In Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, pp. 85-96. ACM, 2019.

[9] M., Jakobsson, and Z., Ramzan, "Crimeware: understanding new attacks and defenses," pp17-19, Addison-Wesley Professional, 2008.

[10] A., Luotonen, and k., Altis, "World-wide web proxies," Computer Networks and ISDN systems, 27(2), pp147-154, 1994.

[11] mitmproxy, https://mitmproxy.org/, accessed Feb. 06. 2019..

[12] PF (4), https://www.freebsd.org/cgi/man.cgi?pf, accessed Feb. 06. 2019.

[13] Rieger, G., socat (1) - Linux man page, https://linux.die.net/man/1/socat, accessed Feb. 06. 2019.

[14] S., Henning, A. Rao, and R., Lanphier, "Real time streaming protocol (RTSP)," https://www.ietf.org/rfc/rfc2326.txt, accessed Feb. 06. 2019.

[15] P., Alan, L., Farrell, D., Kemp, and W. Lupton. "Upnp device architecture 1.1." In UPnP Forum, vol. 22. 2008.

[16] Z., PENG and C., WU, "Microsoft IIS 6.0 - WebDAV 'ScStoragePathFromUrl' Remote Buffer Overflow," Exploit Database, https://www.exploit-db.com/exploits/41738, accessed Feb. 06. 2019.

[17] Google, “Googlebot,” https://support.google.com/webmasters/answer/182072?hl=en, accessed Feb. 06. 2019.

[18] METASPLOIT, “D-Link DCS-930L - (Authenticated) Remote Command Execution (Metasploit),” https://www.exploit-db.com/exploits/39437, accessed Feb. 06. 2019.

[19] Google, "Verifying Googlebot," https://support.google.com/webmasters/answer/80553, accessed Feb. 06. 2019