Electronic Preprint for Journal of Information Processing Vol.28

Regular Paper

# ThingGate: A Gateway for Managing Traffic of Bare-metal IoT Honeypot

Chun-Jung Wu[1,a)] Katsunari Yoshioka[1,2] Tsutomu Matsumoto[1,2]

Received: November 25, 2019, Accepted: June 1, 2020

**Abstract:** The Internet of Things (IoT) malware keep evolving and utilize multiple vulnerabilities to infect IoT devices. Besides malware, human attackers also utilize various tools to access and collect variable information on the device. For instances, web UI of IP Cameras and routers are constantly searched and accessed if vulnerable. In order to observe and analyze such a 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 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. Moreover, it adds the functionality of web tracking, which is not provided by the web UI of the original devices. We evaluate ThingGate with seven bare-metal IoT devices and show that it successfully blocks unwanted incoming attacks, masks wireless access point information of the devices, and tracks attackers on the device web UI while showing high observability of various attacks exploiting different vulnerabilities.

**Keywords:** IoT honeypot, MITM, transparent proxy, IoT devices, fingerprinting

## 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. This network of this appearance is called the Internet's Internet (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. These honeypots include, for example, 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]. Besides well-known activities such as DDoS, recent IoT malware 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. The functions provided through the WebUI and other dedicated services, such as UPnP.

However, it is worth noting that operating bare-metal IoT honeypots have 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 disconnect the honeypot devices. 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.

In this research, we focus on the honeypot observing cyber attacks against the WebUI of IoT devices. Many web honeypots record the source IP, Cookies, and HTTP header information as to identity. However, attackers have already developed countermeasures to evade tracking. Encrypted proxy and anony-

[1] Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[2] Institute of Advanced Sciences, Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[a)] cklonger@gmail.com

mous networks, such as Tor [7], have been widely used to avoid network tracking. Therefore, people develop new fingerprinting from browsers. The browser fingerprinting technique can generate more information and track attackers. Even they might have used proxies or anonymous networks to hide [8]. Therefore, we utilize browser fingerprinting to identify attackers who use browsers visiting our honeypot.

### 1.1 Research Questions

For the honeypot of physical IoT devices which contain WebUI, 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?

## 2. Definitions

### 2.1 Man-in-the-middle

The man-in-the-middle (MITM) refers to an attack in which the attacker positions himself 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.

### 2.2 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].

### 2.3 Browser Fingerprinting

Browser fingerprinting involves making a recognizable subset of users unique. The fingerprint is primarily used as a global identifier for those users. Furthermore, we can utilize a global identifier to create a web tracking mechanism for user browsers [12]. In 2012, Mowery and Shacham presented canvas fingerprinting, which is a web fingerprinting algorithm [15]. They demonstrated that the new HTML5 feature could be used to generate a relatively unique fingerprint that could be used to track users.

Canvas fingerprinting uses the browser's Canvas API to draw invisible images and extract a persistent, long-term fingerprint without the user's knowledge [16]. Tracking mechanisms have advanced such that these mechanisms are difficult to control and detect and are resilient to blocking or removing. Another feature of canvas fingerprinting is that the resulting fingerprint may differ from one browser to another on the same machine [17]. In this study, we use fingerprintjs2 [18], which is an open-source library of canvas fingerprinting, to achieve the web tracking function.

### 2.4 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. [19] employed 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.

## 3. ThingGate

### 3.1 System Goal

The use of conventional IoT devices for building new honeypots raises the following challenges:

(1) Inconvenient reset or restore mechanism
The reset or recovery process need some manual operation on devices. Many devices place the reset button on the control panel, and users have to press the button for a while to trigger the reset function.
(2) Threat of service segmentation fault attacks or brickering command
Some attack vectors, such as BrickerBot, can impair devices [20]. BrickerBot prevents devices from working again even with a factory reset. Moreover, some vulnerabilities, such as CVE-2017-17020, may cause a service shutdown. These types of attack vectors require the employment of human resource for maintenance [21].

(3) Misplaced IoT cyber-attacks flow
For analyzing attack vectors against IoT devices, purchasing all of physical IoT devices to build honeypot is not affordable. We only can utilize a limited number of devices in a honeypot. If the devices' weakness does not fit the incoming attack vector, this attack fails and devices cannot capture the further flow or binaries from clients

(4) Exposure risk from the sensor information
In SIPHoN [3], Guarnizo et al. indicated that scanning for Wi-Fi networks is a feature often offered in the admin interfaces of IP Cameras. The goal of SIPHON is to collect world-wide cyber attacks against IoT via a few devices deployed locally. However, their research did not mention if the Wi-Fi Access Point (AP) name may expose location or not. The Wi-Fi AP list may dynamically show any scanned AP, include a Personal Hotspot from a passerby's mobile devices. The name of AP in WebUI may include personal or organization information to exposing the physical location of honeypot. For video and audio information leakage risk, we place our devices in a close room to prevent sound and image information leakage.

ThingGate is a customized MITM proxy for managing flow between clients and the honeypot that consists of physical IoT devices. To face the challenges from the physical IoT devices, we define the following 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. The attackers might use browsers, and we inject browser fingerprinting JavaScript codes to track attackers.

(3) Real-time analysis of misplaced cyberattack
IoT malware employs various vulnerabilities from WebUI of devices, and injects OS command in the URL. However, some malware didn't check targeted devices before they send malicious HTTP requests. For the misplaced command injection URL (CI-URL) attack target is not in our physical IoT devices, we can conduct real-time analysis and download tasks.

### 3.2 System Overview

Our design, which was inspired by SIPHON architecture [3], is displayed in **Fig. 1**. Our honeypot consists solely of real IoT devices. Moreover, SIPHON's forwarder is improved with MITM proxy to manage the forward traffic from wormholes to local physical IoT devices. We design a module to analyze some CI-URLs. 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 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.

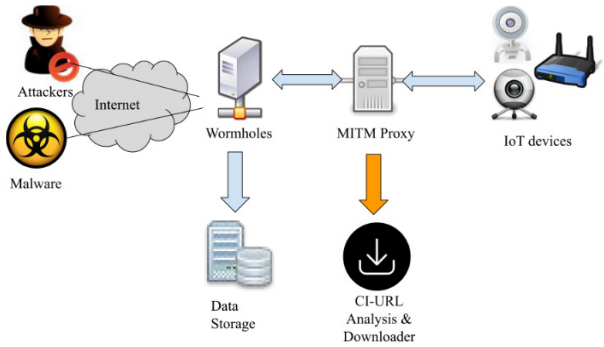


**Fig. 1** System overview of ThingGate.

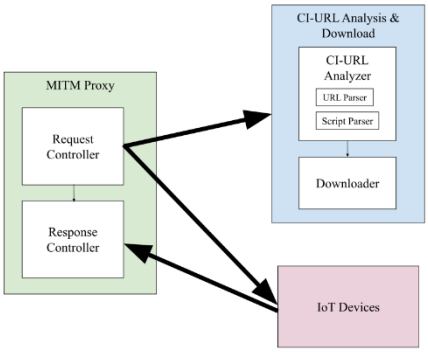


**Fig. 2** System Architecture of ThingGate.

**CI-URL Analysis and Downloader (CAD).** If the flow contains features of CI-URLs, then we redirect the HTTP request to CAD. CAD provides 200 response codes to the client and analyzes the CI-URL. If CAD successfully extracts download links from the flow, real-time download tasks of links can be conducted.

**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, or redirect the flow to the CI-URL analysis and downloader (CAD). The proxy conducts the modification of the flow through the MITM way.

**IoT Devices.** IoT devices are typical commercial off-the-shelf devices that contain vulnerabilities. In this research, we focus on cyber attacks against the WebUI of IoT devices. Thus, we only forward incoming HTTP flow to its HTTP service ports.

**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.

**System Architecture and Modules.** The system architecture of the ThingGate system is displayed in **Fig. 2**. The MITM proxy transparently manages the input and output flow of the honeypot constructed with physical IoT devices in real time. IoT devices answer the flow delegated by the proxy and send the HTTP response back to the client through the MITM proxy. Moreover, the proxy redirects specific HTTP requests that contain CI-URLs to the CAD. Then CAD extracts links and downloads malware

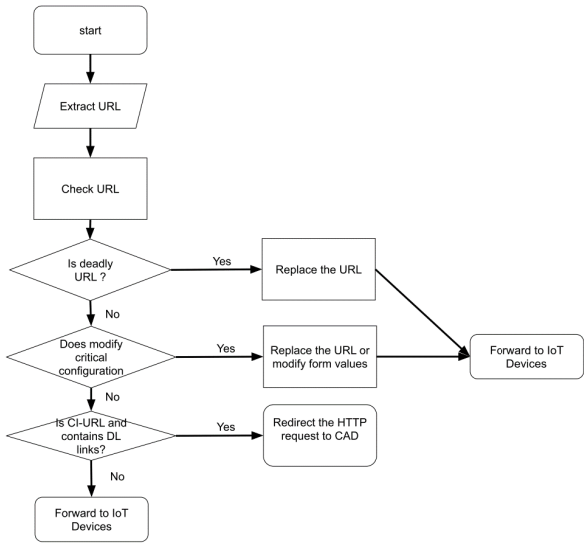


**Fig. 3** The processing flow of Request controller.

binaries. Moreover, our proxy injects fingerprinting JavaScript codes into the HTTP response content and replaces sensitive information with fabricated material.

**The details of the modules are as follows**:

**Request Controller.** The request controller is in charge of incoming HTTP requests. The request controller reviews every request and determines whether the flow should be directly forwarded to IoT devices. The process of URL checking is illustrated in **Fig. 3**. First, we examine whether the URLs utilize the dangerous vulnerability of our IoT devices. For example, D-Link's IP Camera, DCS-5020L, contains vulnerabilities in its WebUI. If attackers post a long string value to the URL "/setSystemNetwork" in the form parameters, then the HTTP request causes the web service to crash [21]. Therefore, the request controller replaces this URL with another valid URL. Second, according to Ezawa's study, some attackers change the DDNS, VPN, or network settings of IoT devices [6] to prevent other clients from accessing the device. These attacks may incur the necessity of performing manual tasks such as rebooting or resetting devices. Therefore, we must protect these critical configurations. The request controller compares the URLs of the incoming request, filters out the requests that cause unwanted configuration changes, and replaces these URLs with other valid URLs of the WebUI. Third, our program verifies the operating system (OS) commands and different URLs embedded in the URL. The request controller redirects these CI-URLs to the CAD. Finally, the request controller forwards the remaining HTTP requests to IoT devices.

**Response Controller.** The response controller is in charge of the HTTP responses from IoT devices. Two conditions trigger action by the response controller.

(1) The HTTP response from IoT devices contains a body tag.
The response controller injects fingerprinting JavaScript codes into the body tag. The JavaScript library creates a hash fingerprint if the client can support the JavaScript code.

(2) The HTTP response includes sensitive information
In this research, we focus on the Wi-Fi AP list. The name of the Wi-Fi AP may consist of a username or information concerning the organization or location. The response controller replaces all APs with fabricated AP information.

Encoded CI-URL:
http://xxx.xxx.156.202/login.cgi?cli=aa%20aa%27;wget%20http://yyy.yyy.173.159/d%20-O%20-%3E%20/tmp/.shinka;sh%20/tmp/.shinka%27$
Decoded CI-URL:
http://xxx.xxx.156.202/login.cgi?cli=aa aa';wget http://yyy.yyy.173.159/d -O -> /tmp/.shinka;sh /tmp/.shinka'$

**Fig. 4** The encoded URL of CI-URL and decoded results.

```
#!/bin/sh
u="asgknskjdgn"
bin_names="mips mipsel arm arm7 powerpc x86_64 x86_32"
http_server="yyy.yyy.173.159"
http_port=80
cd /tmp/||cd /var/
for name in $bin_names
    do
    rm -rf $u
    cp $SHELL $u
    chmod 777 $u
    >$u
    wget http://$http_server:$http_port/$name -O -> $u
    ./$u $name
```

**Fig. 5** Download Scripts from CI-URL.

For supporting more protocols in the future, such as HTTPS, Telnet, and UPnP, we will upgrade the Response controller to generate corresponding traffic according to our rules and protocols.

**CI-URL Analyzer.** is responsible for the two analysis of extracting download links from the CI-URLs and downloaded scripts. The CI-URL analyzer includes two components, namely the URL parser and script parser. The URL parser decodes the CI-URLs and transforms them into OS commands (**Fig. 4**). The CI-URL in Fig. 4 utilizes the vulnerability of the D-Link router DSL-2750B [22]. The URL parser decodes this CI-URL and extracts the link from the OS commands, "http://yyy.yyy.173.159/d." The CI-URL analyzer also passes the link to the downloader.

If we successfully download the file and the file is a shell script file (e.g., the script displayed in **Fig. 5**), then the script parser analyzes the content, traverses all parameters, and extracts the links of malware. Finally, the script parser passes the links of malware to the downloader.

**Downloader** is responsible for malware binaries download tasks. We found the header parameters' value in HTTP requests conducted by IoT devices may be distinguished from Unix/Linux server operating system. For example, the user-agent value conducted by macOS Mojave 10.14.2's wget command is "Wget/1.13.4 (darwin13.1.0)". The "darwin13.1.0." is a library name of macOS packages [23]. In contrast to the user-agent value produced by macOS, the produced by the router A in **Table 1** is "Wget./1.16 (linux-gnu)." Therefore, the user-agent in HTTP header may expose the information of the download client. Therefore, we customized our header values appear as similar as possible to IoT devices.

## 4. Evaluation

### 4.1 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

**Table 1** IoT devices used in experiments.

| IoT device | Maker's country | CPU Arch. | Price* (JPY) |
|---|---|---|---|
| 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** Data set for analysis.

| Data set | Number of HTTP requests | Number of honeypot IP | Time interval | Analysis subjects |
|---|---|---|---|---|
| 1 | 307,405 | 19 | 2018/09/08~ 2018/10/13 | Blocking list, CI-URL, and CAD |
| 2 | 1,920,653 | 19 | 2018/11/17~ 2019/06/30 | **Blocking unwanted flow, Web tracking, Handle misplaced attack, Fabricated sensor content** |

our devices, all of which contained vulnerabilities that had been publicly disclosed. Besides, we installed ThingGate on a server with Intel Core i5 i5-2400 Quad-core (4 Core) 3.10 GHz Processor, 16 GB RAM, and 1.8 Terabytes disk.

**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 19 IP addresses to collect the attack flow (data set 1). ThingGate only forwarded and recorded traffic to devices. Moreover, we analyzed the URL list of critical configurations and the URLs of deadly vulnerabilities from our IoT devices. We designed and implemented the prototype of ThingGate according to data set 1. And then, 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.

There are 58,923 different attacker IPs from data set 2. We use autonomous system number (ASN) to extract the background information of attacker IP address. An autonomous system (AS) [24] is a collection of connected IP routing prefixes under the control of one or more network operators on behalf of a single administrative entity or domain. The AS presents a common, clearly defined routing policy to the interne. An ISP must have an officially registered ASN. Therefore, we can use ASN of IP address to verify the ISP of IP address. Moreover, the database, "ipinfo" provides type data of all ASNs [25]. The database defines four kinds of ASN, including ISP, EDUCATION, HOSTING, and BUSINESS. For the ASN registers by security-related companies, such as Google are labelled as BUSINESS type. We use the database to categorize our attackers' IP address.

**Table 3** Statistics of ASN type for data set 2.

| ASN type | Count | Percentage (100%) |
|---|---|---|
| ISP | 45,938 | 77.96 |
| BUSINESS | 5,440 | 9.23 |
| EDUCATION | 2,402 | 4.08 |
| HOSTING | 4,672 | 7.93 |
| NONE* | 471 | 0.80 |

NONE* is no mapping ASN of given IP address

**Table 4** Top 10 ASN of attackers.

| Rank | ASN | ASN name | Count | Type |
|---|---|---|---|---|
| 1 | AS27699 | TELEFÔNICA BRASIL S.A, BR | 7061 | ISP |
| 2 | AS58224 | TCI, IR | 1332 | ISP |
| 3 | AS3462 | HINET Data Communication Business Group, TW | 1041 | ISP |
| 4 | AS45899 | VNPT-AS-VN VNPT Corp, VN | 1024 | ISP |
| 5 | AS9121 | TTNET, TR | 918 | ISP |
| 6 | AS45090 | CNNIC-TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited, CN | 905 | HOSTING |
| 7 | AS4134 | CHINANET-BACKBONE No.31,Jin-rong Street, CN | 864 | ISP |
| 8 | AS7713 | TELKOMNET-AS-AP PT Telekomunikasi Indonesia, ID | 825 | ISP |
| 9 | AS14061 | DIGITALOCEAN-ASN, US | 767 | HOSTING |
| 10 | AS4837 | CHINA169-BACKBONE CHINA UNICOM China169 Backbone, CN | 751 | ISP |

**Table 3** shows the distribution of ASN type, ISP is the top one type of our attackers (77.96%). Besides, the sum of EDUCATION and BUSINESS is about 13.31%. Therefore, the attacks conducted by traditional hackers. **Table 4** shows the top 10 ASN of attackers' IP address. Top one is an ISP in Brazil. There are eight ISPs and two HOSTING companies in the top 10.

**Table 5** 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) [26]. 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) [27]. 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 target IIS 6.0 [28].

**Table 6** presents the statistics of HTTP requests, attackers' IP a and URLs. Because we forward fifteen IP for IP Camera A1, A2, and A3, they got the most HTTP requests. However, IP Camera C got the most HTTP requests and clients' IP on condition forwarding only one IP traffic to each device.

**Table 5** 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 6** 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 |

### 4.2 Cyber Attacks Against the WebUI of Physical IoT Devices

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 presented in Ezawa's and Tamiya's honeypot [5], [19]. 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 [29]. In addition, after the RCE attack vector, the attacker download devices' live stream by 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 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 American client 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 7 months 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 |

**Table 8** 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 |

### 4.3 Blocking Unwanted Flow Experiments

#### 4.3.1 Design of Experiment

We analyzed our devices and created a list of configuration URLs and dangerous vulnerabilities. From the WebUI and device manuals, we choose 51 critical configuration URLs. Moreover, finding one bricking URL from the security report of IP Camera C. 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 8** presents the nine blacklist and mapping rules against IP Camera A1~A3. Moreover, we define twelve rules for Router A, ten rules for IP camera B, ten

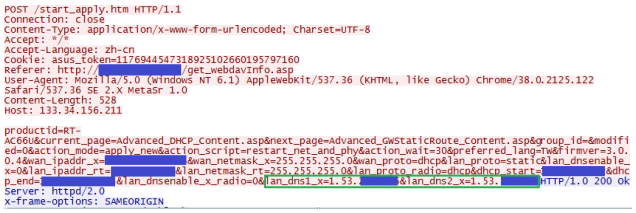


**Fig. 6** The HTTP request of a modifying configuration attack.

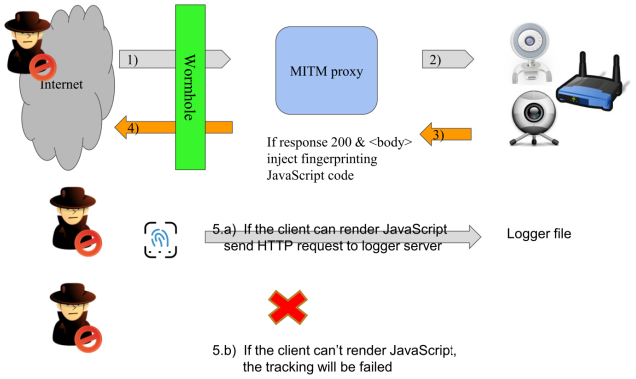


**Fig. 7** Web tracking flow of ThingGate.

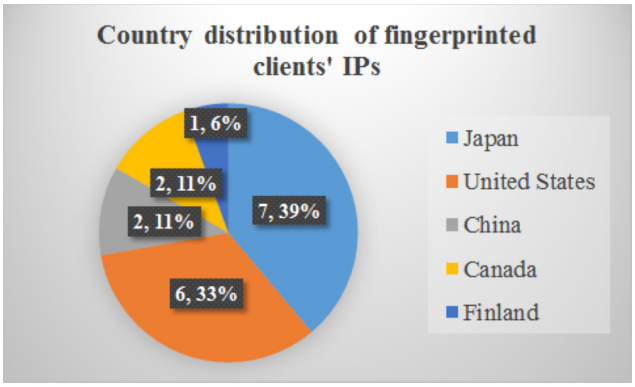


**Fig. 8** Country distribution of fingerprinted clients.

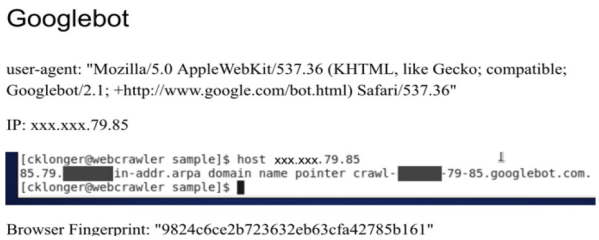


**Fig. 9** Googlebot's user-agent and the verifying result.

rules for IP camera C, and ten rules for IP camera D. According to these rules, ThingGate redirects 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.

#### 4.3.2 Experimental Results

From data set 2, we found on June 7th, an American attacker accessed our Wi-Fi router in the honeypot and modified the LAN DNS setting, point to a Vietnam server. ThingGate successfully blocked the HTTP request, filtered out the form data, and replace the URL with another URL in WebUI. **Figure 6** shows the detail information of the HTTP request. The green rectangle marks the parameters about LAN DNS setting.

### 4.4 Web Tracking Experiments

#### 4.4.1 Design of Experiment

We conducted this experiment on all devices in our honeypot.

As illustrated in **Fig. 7**, ThingGate examined the HTTP response content from all of the devices. If the response code equals 200 and the HTML contains the body tag, then the proxy injects fingerprinting JavaScript codes in response. If the client can render our JavaScript codes, then the client generates a canvas fingerprint and sends it back to ThingGate. However, if the client tool can't render our JavaScript, the tracking will be failed.

#### 4.4.2 Experimental Results

From data set 2, we found that clients from 18 different source IPs successfully sent their fingerprint values to ThingGate. We collected 26 different fingerprint values from these clients. The geographic information on the IPs of the fingerprinted clients is displayed in **Fig. 8**. Among 18 clients, seven were from Japan (39%) and six were from the United States (33%). In total, 72% of the clients were from these two countries. Four clients provided only one fingerprint value, whereas the other 14 clients provided two or more fingerprint values. Moreover, we discovered that one of the four single-fingerprinted clients was Googlebot [30]. We verified Googlebot by using a reverse DNS lookup on the accessed IP address according to a Google document [31] (**Fig. 9**). Googlebot attempts to access the IP Camera C and sends requests against 18 different URLs of the WebUI. These URLs contain the snapshot, parameters of the camera, DDNS, and Wi-Fi setting pages. Googlebot successfully collected the configuration information of the devices, including our fabricated Wi-Fi AP list.

Among the fingerprinted clients, three America clients sent the same two fingerprint values back to ThingGate. **Table 9** presents the attack features of the three clients. They almost traversed the forwarding IP of the honeypot. Moreover, more than 27% of HTTP requests were utilized in the HEAD method to attack our devices, and 83% of the URLs between the three clients were common among them. The identical features and fingerprint values implied that the three clients belonged to the same attacker.

### 4.5 Managing Misplaced Attacks Experiments

#### 4.5.1 Design of Experiment

ThingGate examines all of the incoming flow against 19 IP addresses. If any different site with OS commands is embedded in the URL, our program redirects the flow to CAD through an MITM way. Next, the CI-URL analyzer analyzes the URLs and scripts downloaded from the URLs. The downloader handles all downloading tasks if our parsers extract any link during the analysis.

#### 4.5.2 Experimental Results

The attack flow of data set 2 revealed that ThingGate redirected the HTTP requests of 411 CI-URLs to CAD. These CI-URLs contained 50 different URLs that exploited seven vulnerabilities. **Figure 10** depicts the vulnerability distribution of the URLs. A total of 76% of the CI-URLs used the top two vulnerabilities from products of D-Link and ThinkPHP. The usage of these two vulnerabilities was three times that of other vulnerabilities. **Table 10**

**Table 9** Features of the fingerprinted clients.

| Source IP address | Victim devices | Unique URL count | HEAD URLs count | Common URLs with 1st IP | Attack Duration |
|---|---|---|---|---|---|
| xxx.xxx.226.109 | IP Camera A1~A3, B, C, and D | 128 | 44 | N/A | 2018/12/05~ 2019/01/11 |
| xxx.xxx.32.101 | IP Camera A1~A3, B, C, and D | 74 | 23 | 62 | 2018/12/14~ 2019/01/23 |
| xxx.xxx.30.101 | IP Camera A1~A3, B, C, and D | 33 | 9 | 32 | 2018/12/28~ 2019/01/11 |

**Table 10** Information of Vulnerabilities. Observation of 7 months from IP Camera A1~A3, B, C, and D.

| Maker | CVE/ Exploit DB | Type | model/ver. | URL pattern of vulnerability |
|---|---|---|---|---|
| D-Link | OS Command Injection (Metasploit) [22] | Router | DSL-2750B | /login.cgi?cli=aa%20aa%27;wget%20 |
| ThinkPHP | Remote Code Execution [32] | Web app framework | V5.X | /index.php?s=/index/¥think¥app/invokefunction&function |
| AVTECH | 2015-2280 [33] | Camera/NVR/DVR | all version | /cgibin/nobody/Search.cgi?action=cgi_query&ip=google.com&port=80&queryb64str=Lw==&username=admin%20;XmlAp%20r%20Account.User1.Password%3E$ |
| AirLink | 2015-2280 [34] | Camera | SkyIPCam1620W | /maker/snwrite.cgi?mac=1234& |
| Fastweb | 2018-11336 [35] | Modem | V0.0067 | /status.cgi_=15269046001 31&cmd=3&nvget=login_confirm &password |
| MikroTik | 2018-14847 [36] | RouterOS | Before V6.38.4 | /jsproxy? |
| TUTOS | 'cmd.php' Remote Command Execution [37] | Software | V1.3 | /tutos/php/admin/cmd.php?cmd= |

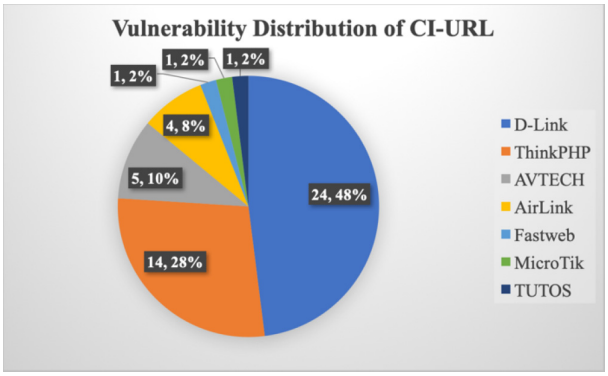


**Fig. 10** Vulnerability distribution of CI-URL.

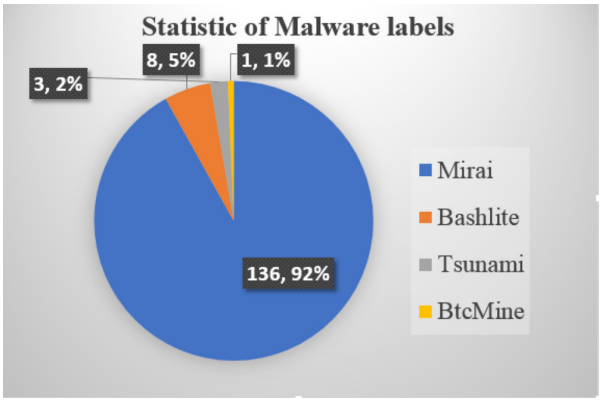


**Fig. 11** Statistic of malware labels.

presents information on the seven vulnerabilities, including the maker, model, version, and path of the WebUI.

From the 411 CI-URLs, the CAD downloaded 150 different malware binaries and 23 scripts. Therefore, we searched for an optimal solution for labeling these malware binaries. VirusTotal [38] was the platform used to obtain scan results from 66 antivirus engines. We sent 12,821 unique malware MD5s from IoTPOT in 2017 and selected the most common malware family name as the representative malware category from the VirusTotal reports. We also discovered that Kaspersky, DrWeb, and ESET-NOD32 are the top three antivirus engines because of their high detection ratio and consistency. We conducted a local scan of 40,203 different IoT malware binaries and found that DrWeb could label 39,245 of them, which comprises 97.61% of the submitted malware. The labeling performance of DrWeb surpassed that of both Kaspersky (69.82%) and ESET-NOD32 (74.57%). Therefore, we employed DrWeb to label the IoT malware collected by the CAD in data set 2.

DrWeb successfully marked 148 binaries. **Figure 11** illustrates the statistics of malware labels. Mirai malware accounts for the vast majority of binary files (92%). We discovered that 18 Mirai binaries employed ThinkPHP's vulnerability to infect victim sites. Moreover, the BTCMine malware (one binary) is a mining trojan. The attacker of the BTCMine malware also utilized the vulnerability of ThinkPHP to attack our honeypot.

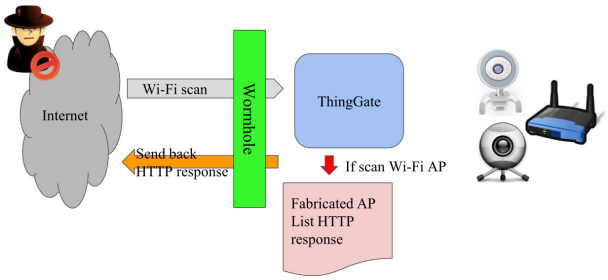


**Fig. 12** Fabricated Wi-Fi AP list.

**Table 11** Attacker who request Wi-Fi information. Observation of 7 months 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 | 2018/12/12~2018/12/12 |
| Client B | bbb.bbb.169.38 | USA | 3 | 4167 | 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 client G is Googlebot

### 4.6 Fabricated Sensor Information Experiment

#### 4.6.1 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. **Figure 12** depicts the webpages before and after modification with ThingGate.

#### 4.6.2 Experimental Results

In data set 2, we found that ThingGate sent fabricated Wi-Fi information to 44 different clients in 80 HTTP response. **Table 11** presents part of the attackers' geographical location, number of requests sent, and duration of visit to our honeypot. The Googlebot client only sent 23 HTTP requests in one day.

### 4.7 Stress Testing against IoT Devices

#### 4.7.1 Design of Experiment

IP Camera B only can offer four clients to view live stream video. Therefore, we assume up to five users may watch the live stream of IP Cameras concurrently. Our testing employs five Chrome browsers (v72.0.3626.121) on five computers to login IP Cameras and to view the pages contain live streams. We both conduct the testing through ThingGate or access WebUI directly. Moverover, examining each condition for ten times.

**Table 12** IoT devices used in experiments.

| IoT Device | Path | Clients download video through TingGate | Clients download video. |
|---|---|---|---|
| IP Camera A1 | /monitor2.htm | 50 | 50 |
| IP Camera B | /main.htm | 40/50* | 40/50 |
| IP Camera C | /video.cgi** | 50 | 50 |
| IP Camera D | /top.htm | 50 | 50 |

*IP Camera B only allows four clients to download live stream data. Therefore, our clients sent 50 requests on each condition and only got 40 responses with live streams.

**The certificates in the default firmware of IP Camera C is out-of-date. Hence, browsers including IE, Firefox, Chrome, and Safari block the rendering function of live stream of default web pages. Hence, we employed the hidden live stream application (/video.cgi) which employed by the attacker in 4.2 to evaluation.

#### 4.7.2 Experimental Results

**Table 12** shows the statistic of testing results between ThingGate and directly forwarding flow. Our results show attackers would receive the same rendering video under the two conditions, with or without ThingGate.

## 5. Related Work

In Ref. [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. [19] employed a decoy honeypot 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.

Compared to existing literature, we find the previous honeypot of physical IoT devices lacks abilities against sensitive information leaks and dangerous commands. 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 dangerous commands. Moreover, we extended the web tracking function to WebUI of physical devices. Further, our setup allows us to cap-

ture and analyze some misplaced attacks across different remote code execution vulnerabilities in real-time.

## 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 streams via hidden web applications, are hard to simulate by the virtual machine. The difference between general web applications and the WebUI of IoT devices is sensor data. The sensor data includes numeric, text, and streaming video, which depends on the type of sensor. The streaming data and the reaction of manipulating the sensor, such as rotating the camera, are hard to emulating by traditional honeypots. There are number of attacks on IP cameras including actually peeping the video by logging into the WebUI of the camera. There are also number of attacks on routers where configurations of router (such as DNS setting, VPN, etc.) are altered by attackers. It is hard to emulate these functionalities of Camera and routers by low-interaction honeypot and therefore, bare-metal honeypot is necessary.

From the unwanted block experiment, the results show that ThinkGate can block the attack, which changes critical configuration. We confirm that ThingGate can protect devices from misconfiguration.

In addition, we also found 44 clients request 80 times for the Wi-Fi AP information web page. ThingGate sent back fabricated sensor information to these clients and successfully prevent information leakage. Of the 44 clients, 41 clients employed a predefined list to scan victims; two are human-like attackers and Googlebot. From the web tracking experiment, we successfully extended a tracking function to IoT devices and tracked an attacker employed three American IP addresses to visit our honeypot. ThingGate added the browser fingerprinting functionality to WebUI of physical IoT devices.

About the misplaced attacks, ThingGate extracts 411 CI-URL and download 149 different malware binaries and 23 scripts. Moreover, we found 18 binaries utilized the ThinkPHP vulnerability, which is not an IoT device but a web application framework. The abuse of HTTP 80 port becomes much serious. From the stress testing results, attackers can get the same rendering live stream from IP cameras through ThingGate. Hence, we can build the bare-metal IoT honeypots together with ThingGate.

By protecting the configuration of devices, the attack vectors targeted the configuration would fail. The failure might make attackers perceive that the devices are unusual. Moreover, the injected JavaScript code sends a special HTTP request contains fingerprinting data to ThingGate. An attentive attacker may aware of the MITM attack and realize the target is a honeypot.

### 6.1 Limitations

ThingGate does have some limitation. Many of the limitations come from the design of CI-URL analyzer. First, the URL parser only analyzes the CI-URL whose OS commands the attacker embedded in URL. Our program did not check other header field or form data yet. Second, the script parser only was able to handle several kinds of shell scripts. A Linux sandbox can resolve more types of scripts. However, the sandbox must be monitored and implemented with the high-security design because of the Brickerbot. Thirdly, our web tracking function relies on JavaScript and Canvas fingerprint. Therefore, if the clients cannot render the JavaScript code, the client cannot trigger fingerprint function.

## 7. Conclusion and Future Works

### 7.1 Conclusion

We combine the ability of the transparent proxy and web tracking library, develop a supporting mechanism for honeypot of physical IoT devices. ThingGate can improve the security of honeypot, extend the functionality of web tracking, manage the incoming traffic, and output response content via MITM way. We evaluated ThingGate on the public internet, examined 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 track one American attacker use multiple IP addresses to visit our honeypot. To handle the unwanted incoming flow, we confirm that ThingGate can block traffic, which changes the critical configuration. Moreover, ThingGate collected 149 malware binaries and 23 scripts from 411 misplaced CI-URL, which employed seven vulnerabilities. Furthermore, ThingGate fooled seven clients who requested the Wi-Fi AP list in WebUI with fabricated AP.

### 7.2 Future Works

In this research, we only exam HTTP traffic. We think that extending the proxy scope for more protocols is essential, such as HTTPS, Telnet, and UPnP. Therefore, future works should focus on how to improve existing methodologies for building advanced honeypot.

**Acknowledgments** A part of this work was obtained from EU-Japan collaboration project “Multi-layered Security technologies to ensure hyperconnected smart cities with Blockchain, Big Data, Cloud and IoT (MSEC).” Special thanks go to Dr. Erwan Le Malecot for the critical comments and help on the network infrastructure.

### References

[1] Loshin, P.: Details emerging on Dyn DNS DDoS attack, Mirai IoT botnet, TechTarget network, available from ⟨http://searchsecurity.techtarget.com/news/450401962/Details-emerging-on-Dyn-DNS-DDoS-attack-Mirai-IoT-botnet⟩ (accessed 2019-02-06).

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

[3] Guarnizo, J.D., Tambe, A., Bhunia, S.S., Ochoa, M., Tippenhauer, N.O., Shabtai, A. and Elovici, Y.: Siphon: Towards scalable high-interaction physical honeypots, *Proc. 3rd ACM Workshop on Cyber-Physical System Security*, pp.57–68 (Apr. 2017).

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

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

[6] Nigam, R.: Unit 42 Finds New Mirai and Gafgyt IoT/Linux Botnet Campaigns, Unit42, available from ⟨https://unit42.paloaltonetworks.com/unit42-finds-new-mirai-gafgyt-iotlinux-botnet-campaigns/⟩ (accessed 2019-02-06).
[7] Dingledine, R., Mathewson, N. and Syverson, P.: Tor: The second-generation onion router, *Proc. 13th Conference on USENIX Security Symposium* (2004).
[8] Zhaopeng, J., Cui, X., Liu, Q., Wang, X. and Liu, C.: MicroHoneypot: Using Browser Fingerprinting to Track Attackers, *2018 IEEE 3rd International Conference on Data Science in Cyberspace* (*DSC*), pp.197–204 (2018).
[9] Jakobsson, M. and Ramzan, Z.: *Crimeware: Understanding new attacks and defenses*, pp.17–19, Addison-Wesley Professional (2008).
[10] Luotonen, A. and Altis, K.: World-wide web proxies, *Computer Networks and ISDN Systems*, Vol.27, No.2, pp.147–154 (1994).
[11] mitmproxy, available from ⟨https://mitmproxy.org/⟩ (accessed 2019-02-06).
[12] PF (4), available from ⟨https://www.freebsd.org/cgi/man.cgi?pf⟩ (accessed 2019-02-06).
[13] Rieger, G.: socat (1) - Linux man page, available from ⟨https://linux.die.net/man/1/socat⟩ (accessed 2019-02-06).
[14] Eckersley, P.: How unique is your web browser?, *International Symposium on Privacy Enhancing Technologies Symposium*, pp.1–18, Springer, Berlin, Heidelberg (July 2010).
[15] Mowery, K. and Shacham, H.: Pixel perfect: Fingerprinting canvas in HTML5, *Proc. W2SP*, pp.1–12 (2012).
[16] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A. and Diaz, C.: The web never forgets: Persistent tracking mechanisms in the wild, *Proc. 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp.674–689 (Nov. 2014).
[17] Raschke, P. and Küpper, A.: Uncovering Canvas Fingerprinting in Real-Time and Analyzing ist Usage for Web-Tracking, *Workshops der INFORMATIK 2018-Architekturen*, Prozesse, Sicherheit und Nachhaltigkeit, Köllen Druck+ Verlag GmbH (2018).
[18] Valve/fingerprintjs2, available from ⟨https://github.com/Valve/fingerprintjs2⟩ (accessed 2019-02-06).
[19] Tamiya, K., Ezawa, Y., Tie, Y., Nakayama, S., Yoshioka, K. and Matsumoto, T.: Observation of Peeping using Decoy IP Camera, Symposium on Cryptography and Information Security 2018 (Jan. 2018).
[20] Goodin, D.: BrickerBot, the permanent denial-of-service botnet, is back with a vengeance, ARS TECHNICA, available from ⟨https://arstechnica.com/information-technology/2017/04/brickerbot-the-permanent-denial-of-service-botnet-is-back-with-a-vengeance/⟩ (accessed 2019-02-06).
[21] Fidus: DLINK DCS-5020L DAY N' NIGHT CAMERA REMOTE CODE EXECUTION WALKTHROUGH, Fidus, available from ⟨https://fidusinfosec.com/dlink-dcs-5030l-remote-code-execution-cve-2017-17020/⟩ (accessed 2019-02-06).
[22] METASPLOIT: D-Link DSL-2750B - OS Command Injection (Metasploit), available from ⟨https://www.exploit-db.com/exploits/44760⟩, (accessed 2019-02-06).
[23] Kernel and Device Drivers Layer, available from ⟨https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html⟩ (accessed 2019-02-06).
[24] Hawkinson, J. and Bates, T.: Guidelines for creation, selection, and registration of an Autonomous System (AS) (1996).
[25] ipinfo.ip: Full Response, available from ⟨https://ipinfo.io/developers/responses#asn-details⟩ (accessed 2019-02-06).
[26] Henning, S., Rao, A. and Lanphier, R.: Real time streaming protocol (RTSP), available from ⟨https://www.ietf.org/rfc/rfc2326.txt⟩ (accessed 2019-02-06).
[27] Alan, P., Farrell L., Kemp, D. and Lupton, W.: Upnp device architecture 1.1., In UPnP Forum, Vol.22 (2008).
[28] Peng, Z. and Wu, C.: Microsoft IIS 6.0 - WebDAV 'ScStoragePathFromUrl' Remote Buffer Overflow, Exploit Database, available from ⟨https://www.exploit-db.com/exploits/41738⟩ (accessed 2019-02-06).
[29] METASPLOIT: D-Link DCS-930L - (Authenticated) Remote Command Execution (Metasploit), available from ⟨https://www.exploit-db.com/exploits/39437⟩ (accessed 2019-02-06).
[30] Google, available from ⟨https://support.google.com/webmasters/answer/182072?hl=en⟩ (accessed 2019-02-06).
[31] Google: Verifying Googlebot, available from ⟨https://support.google.com/webmasters/answer/80553⟩ (accessed 2019-02-06).
[32] VULNSPY, ThinkPHP 5.0.23/5.1.31 - Remote Code Execution, available from ⟨https://www.exploit-db.com/exploits/45978⟩ (accessed 2019-02-06).
[33] Eberhardt, G.: AVTECH IP Camera / NVR / DVR Devices - Multiple Vulnerabilities, available from ⟨https://www.exploit-db.com/exploits/40500⟩ (accessed 2019-02-06).
[34] CORE SECURITY: AirLink101 SkyIPCam1620W - OS Command Injection, available from ⟨https://www.exploit-db.com/exploits/37527⟩ (accessed 2019-02-06).
[35] Procode701: Fastweb FASTGate - 0.00.67 RCE Vulnerability, available from ⟨https://cxsecurity.com/issue/WLB-2018100117⟩ (accessed 2019-02-06).
[36] CVE-2018-14847, available from ⟨https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-14847⟩ (accessed 2019-02-06).
[37] HOUSSAMIX: TUTOS 1.3 - 'cmd.php' Remote Command Execution, available from ⟨https://www.exploit-db.com/exploits/4861⟩ (accessed 2019-02-06).
[38] Total, V.: Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community, available from ⟨https://www.virustotal.com/#/home/upload⟩ (accessed 2019-02-06).

**Chun-Jung Wu** received his B.S. in mathematics from National Taiwan University, Taiwan in 2003 and M.S. in computer Science from National Taiwan University of Science and Technology, Taiwan in 2007. From 2008 to 2016, he was an engineer at Institute for Information Industry, Taiwan. He received his Ph.D. in information sciences from Yokohama National University in 2019 respectively. His research interest is IoT Security.

**Katsunari Yoshioka** is an Associate Professor at Yokohama National University since 2011. Before that, he was a researcher at National Institute of Information and Communications Technology, Japan. His research interests cover wide area of system security and network security including malware analysis and IoT security. He received the commendation for science and technology by the minister of MEXT, Japan in 2009, the award for contribution to Industry-Academia-Government Collaboration by the minister of MIC, Japan in 2016, and the Culture of Information Security Award in 2017.

**Tsutomu Matsumoto** is a professor of the Faculty of Environment and Information Sciences, Yokohama National University, and directing the Research Unit for Information and Physical Security at the Institute of Advanced Sciences. Prof. Matsumoto also serves as the Director of the Cyber Physical Security Research Center (CPSEC) at the National Institute of Advanced Industrial Science and Technology (AIST). Starting from Cryptography in the early '80s, he has opened up the field of security measuring for logical and physical security mechanisms. He received a Doctor of Engineering degree from the University of Tokyo in 1986. Currently, he is interested in research and education of Embedded Security Systems such as IoT Devices, Cryptographic Hardware, In-vehicle Networks, Instrumentation and Control Security, Tamper Resistance, Biometrics, Artifact-metrics, and Countermeasure against Cyber-Physical Attacks. He serves as the chair of the Japanese National Body for ISO/TC68 (Financial Services) and the Cryptography Research and Evaluation Committees (CRYPTREC) and as an associate member of the Science Council of Japan (SCJ). He was a director of the International Association for Cryptologic Research (IACR) and the chair of the IEICE Technical Committees on Information Security, Biometrics, and Hardware Security. He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.