Implementation of MQTT/CoAP Honeypots and Analysis of Observed Data

Hajime Shimada
Information Technology Center, Nagoya University
Nagoya-shi, Japan
Email: shimada@itc.nagoya-u.ac.jp

Katsutaka Ito
Meitetsucom Co., LTD
Nagoya-shi, Japan,
Email: itokatu@net.itc.nagoya-u.ac.jp

Hirokazu Hasegawa
Information Strategy Office, Nagoya University
Nagoya-shi, Japan
Email: hasegawa@icts.nagoya-u.ac.jp

Yukiko Yamaguchi
Information Technology Center, Nagoya University
Nagoya-shi, Japan
Email: yamaguchi@itc.nagoya-u.ac.jp

***Abstract*—Recently, there are many systems that utilize Internet of Things (IoT) effectively. Those systems often use simple IoT-aimed protocols, such as Message Queue Telemetry Transport (MQTT) or Constrained Application Protocol (CoAP). However, recent cyber-attacks have been targeting IoT systems (e.g., the "Mirai" malware) so we are concerned that malicious persons could also exploit IoT-aimed protocols in cyber-attacks. Thus, we proposed MQTT/CoAP honeypots to observe possible cyber-attack or scouting activities related to cyber-attack. To imitate real IoT systems, the proposed honeypots hold imitated sensing data which is updated periodically. Also, to avoid ill use by attackers, the proposed honeypot accepts update requests from the Internet, but the updated value is only visible to the same request source IP (Internet Protocol) address. The proposed honeypots were deployed in December 2016 (MQTT) and August 2017 (CoAP) and requests were observed from the Internet continuously. We observed several mysterious requests to both MQTT and CoAP honeypots. We observed that the MQTT honeypot received some non-MQTT protocol based requests to 1883/tcp and some of them are wrongly interpreted as MQTT protocol. We determined that an effective MQTT server must be robustly implemented to handle there types of requests.**

***Keywords*–Honeypot; Internet of Things; MQTT; CoAP.**

I. Introduction

In recent years, there are many systems that utilize Internet of Things (IoT) effectively. Many of those systems use common protocol to transfer data, such as HTTP (Hyper-Text Transfer Protocol), however, such protocols were developed for transferring comparatively rich content that include a large amount of overhead even when sending small amounts of data. In some cases, the size of the protocol header becomes larger than the size of the sending data. As a result, some systems use simple IoT-aimed protocols, such as Message Queue Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) to reduce overhead [1][2].

However, recent cyber-attacks have been targeting IoT systems so we are concerned that malicious persons could also exploit IoT-aimed protocols in cyber-attacks. The use of those protocols is comparatively small, so there may exist many insecure servers which use those protocols. A number of services gather the opened ports and services of a server and list them via a Web service for security notice (e.g., Shodan [3]). However, such services can also be used by an attacker to explore servers, including the servers using IoT-aimed protocols.

In this paper, we report on the results of our previously proposed MQTT/CoAP honeypots to observe possible cyber-attacks or scouting activities related to cyber-attacks. To imitate real IoT systems, the proposed honeypots hold fake sensing data that are updated periodically. To avoid ill use by attackers, the proposed honeypot accepts update requests from the Internet but the updated value is only visible to the same IP (Internet Protocol) address that sent the updating request. This function can avoid ill use for a malware's Command and Control server. Also, we set the data transfer limit per hour to prevent DoS (Denial of Service) attacks (registering attack destinations as subscribers).

The MQTT version of the proposed honeypot was deployed in December 2016, and we presented our initial report at a domestic meeting in March 2017 [4]. We deployed our CoAP version in August 2017 and continuously observed requests from the Internet. Unfortunately, we lost data recorded after November 15, 2017 due to both system failure and careless backup treatment. So, the data that we could analyze are from December 1, 2016 to October 30, 2017 from the MQTT honeypot and from August 1, 2017 to October 30, 2017 from the CoAP honeypot. The analyzed results show that requests for MQTT were several times larger than those of CoAP in a month, but many of them came from security companies. We observed mysterious requests originating from 3 different countries with the same payload in around 2000 second intervals without following the standards of the protocol. We considered these requests to be attempts for unauthorized access or to attack the server.

The rest of the paper is organized as follows. Section II introduces related works about honeypots for IoT systems. Section III introduces the characteristics of IoT-aimed protocols that we cover in our proposed honeypot. We introduce our proposal and implementations in Section IV. Section V shows gathered accesses with the proposed honeypot and analysis results. Finally, we conclude and introduce future works in Section VI.

II. Related Works

Some honeypots implement IoT-aimed protocols as a part of their functions. A honeypot named Dionaea[5][6] has a MQTT module as a third party implementation. A generic low-interaction honeypot named glutton [7] also has a MQTT module. However, these honeypots only gather requests arrived to opened ports and cannot imitate a real IoT system.

Some studies have focused on honeypots that mimic IoT devices. Some of those honeypots gather information from attacks via telnet (23/tcp). Pa et al. proposed IoTPot[8][9] which analyze attacks by emulating telnet connection of various IoT devices. They detected 106 distinct types of malware from 5 malware families. Their observations were only limited to telnet so that our research differs in that we cover multiple IoT-aimed protocols. Wang et al. proposed ThingPot [10] which emulates multiple protocol servers including several IoT-aimed protocols, such as MQTT, CoAP, and Advanced Message Queuing Protocol (AMQP). But their emulation is limited to the server level and not the system level so that attackers can easily determine that it is a honeypot by checking the topics in the honeypot. Luo et al. proposed IoTCandyJar [11] which emulates multiple protocol servers including several IoT-aimed protocols, similar to ThingPot.

In this paper, we introduce a proposal of IoT honeypot system that imitates real IoT systems by registering fake sensing values periodically. We operated the systems over a long period to gather data on MQTT and CoAP connections, which were treated as being in the “minor connection category” in prior studies.

III. Introduction of IoT-Aimed Protocols

A. MQTT: Message Queue Telemetry Protocol

Message Queue Telemetry Transport (MQTT) [12] is a broker-based publish/subscribe-type messaging protocol. It can use both 1883/tcp and 1883/udp for implementation, but TCP is widely used. A fixed header of MQTT is only 2 bytes so that it is suitable for IoT devices and networks with limitations on processor and network performance.

An outline of MQTT usage is shown in Figure 1. The MQTT server is called a broker and clients are separated into publishers and subscribers. A subscriber sends a request to the broker by indicating the topic, and the broker sends the topic to all subscribers when a publisher sends a corresponding topic to the broker. Topics in a MQTT system are organized with a hierarchical structure similar to a directory in a file system (e.g., /8F/room806/temperature). The subscriber can use the wildcard “+” to represent a hierarchy to indicate multiple topics (e.g., /7F/+/humidity). The wildcard “#” can be used to represent all lower hierarchies to indicate multiple topics (e.g., /6F/#) or by itself to obtain all topics. The published message is discarded after distribution to the subscriber in the MQTT basic configuration. However, the MQTT has a retaining function that keeps the latest published topic in the broker and sends it to a new subscriber if they request the topic.

MQTT has 3 levels of QoS (Quality of Service) which are described as at most once, at least once, and exactly once, respectively. The QoS is present on both the publisher’s and the subscriber’s sides, and they are only valid between publisher/subscriber and broker.

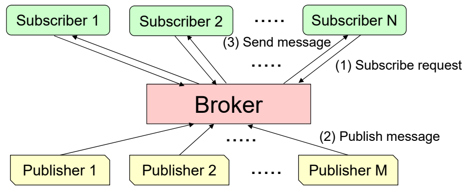


Figure 1. Outline of MQTT usage

TABLE I. METHODS OF CoAP AND CORRESPONDING RESPONSES

	Response from server	
Method	Resource exists	Resource does not exists
GET	2.05 Content	4.04 Not Found
PUT	2.04 Changed	4.04 Not Found
POST	2.04 Content	4.03 Created
DELETE	2.02 Deleted	4.04 Not Found

B. CoAP: Constrained Application Protocol

Constrained Application Protocol (CoAP) [13] is a simplified HTTP protocol based on UDP. Similar to MQTT, CoAP prepares a server to share messages between clients. Table I shows usable methods for CoAP clients and corresponding responses from the server. The client can manipulate resources by using both a URI (Uniform Resource Identifier) and method.

The client can obtain all resources with a “GET /.well-known/core” request. By adding the “observe” option to the GET method, the server automatically sends any updated data.

IV. Proposal of Honeypots

A. Concept of Implementation

Both MQTT and CoAP utilizes the server layer to gather data. So, a server with a fixed IP address becomes the target of a cyber-attack.

Figure 2 shows the concept of a honeypot implementation. The honeypot imitates a server using IoT-aimed protocols and accepts arbitrary requests from the Internet. To capture all requests received by the server, including out of standard requests (e.g., requests with other protocols), we capture whole packets that come to the honeypot at the layer 4 (TCP/UDP) level and send them to the analyzer module. To imitate a real IoT system, we prepare a fake data registration module that periodically registers fake IoT sensor node data (e.g., temperature, barometric pressure, CO_2 cardinality) to the honeypot.

The honeypot system also has to consider ill use of the honeypot by smart attackers. Possible malicious usages of the honeypot server and countermeasures are listed below.

a) Exploit honeypot process vulnerability to occupy server: By exploiting vulnerabilities in a server’s processes and utilizing privilege escalation methods, an attacker can sometimes occupy servers. To limit the occupation range and stop the occupied server easily, we executed the honeypot on a Virtual Machine (VM) that can easily be stopped from a VM host. The packet capture module is placed on another VM host to achieve continuous capture even if the honeypot VM has been occupied by the attacker.

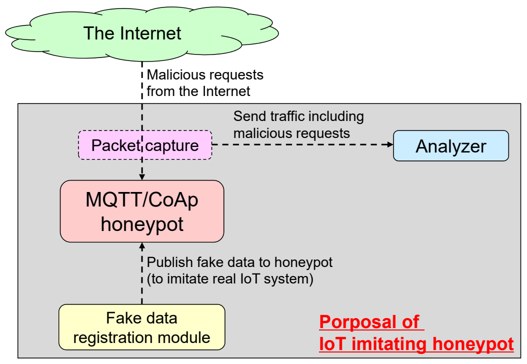

The Internet
Malicious requests from the Internet
Send traffic including malicious requests
Packet capture
Analyzer
MQTT/CoAP honeypot
Publish fake data to honeypot (to imitate real IoT system)
Fake data registration module
Porposal of IoT imitating honeypot

Figure 2. Concept of Implementation

b) Utilize as Command and Control server of malware: Command and Control server is the core server of a Botnet or Remote Administration Trojan type malware in which an attacker can use versatile servers for it even social networking services (e.g., Twitter, Slack). To avoid this type of ill use, we add a limitation to the server that an updated value is only visible from an IP address that sent the update request.

c) DoS attack to other host: Both MQTT and CoAP have a "distribute newly registered data to subscribers" mode which an attacker can use for DoS attacks by registering attack destinations as subscribers. To avoid this type of ill use, we modified the server source code to not distribute newly registered data even if those options have been enabled. Furthermore, we limit the outbound traffic rate at the VM host side.

d) Malicious message to IoT clients: There is possibility that some attacker may exploit other clients (fake data publisher module) by registering some malicious data to the honeypot. We implemented a fake data publisher module as an independent VM and only execute registration actions to the broker VM (no read action).

B. Detailed implementation of MQTT Honeypot

Figure 3 shows a detailed implementation of the MQTT honeypot. As discussed in Section IV-A, we separated the honeypot VM and fake data publisher VM. Furthermore, captured traffic is sent to the other host, to avoid suffering from VM host aiming attacks. Thus, we utilized 3 network domains, such as an Internet connection domain (133.x.x.x/26), fake data publisher connection domain (192.168.10.0/24), and analysis host connection domain (192.168.5.0/24).

We utilized mosquitto [14] which is a famous open source MQTT broker implementation, with the modifications described in Section IV-A. The fake data publisher and analyzer were implemented as a Python script with the paho-mqtt [15] module. We used tcpdump for the packet capture module.

The fake data publisher module publishes data to imitate a sensor network for a building energy management system. The topic format is organized as "/floor_number/ room_number/sensing_data." The floor number has 5 variations as represented by the "8F" notation. The room number has 17 variations as represented by the "708" notation. The sensing data has 3 variations, such as temperature, humidity, and pressure. An example topic is represented by the

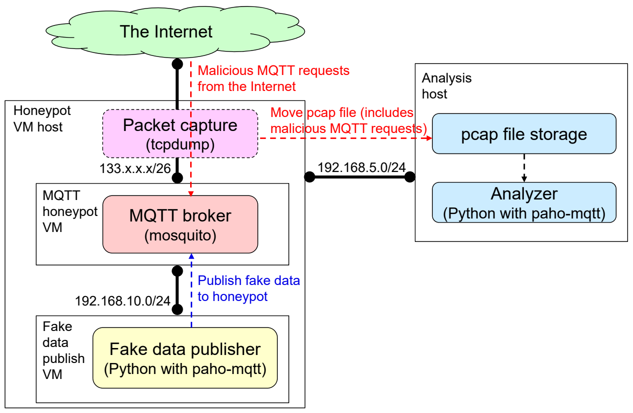

The Internet
Malicious MQTT requests from the Internet
Honeypot VM host
Packet capture (tcpdump)
133.x.x.x/26
Move pcap file (includes malicious MQTT requests)
Analysis host
pcap file storage
192.168.5.0/24
Analyzer (Python with paho-mqtt)
MQTT honeypot VM
MQTT broker (mosquito)
Publish fake data to honeypot
192.168.10.0/24
Fake data publish VM
Fake data publisher (Python with paho-mqtt)

Figure 3. Detailed implementation of MQTT honeypot

TABLE II. HONEYPOT HOSTS

Host name	Type	Deployment day
IBmonitor.net....	MQTT	December 1, 2016
freezermonitor.net....	CoAP	August 1, 2017
co2monitor.net....	CoAP	August 1, 2017
furnacemonitor.net....	CoAP	August 1, 2017

"/8F/708/temperature" notation. All published data are registered with a retaining notification so that the attacker can see the fake data anytime.

C. Detailed implementation of CoAP Honeypot

The implementation of the CoAP honeypot is very similar to that of the MQTT honeypot so that the basic organization is identical to the one shown in Figure 3. We implemented "one topic to one CoAP server" so that the number of honeypot VMs and fake date register VMs are multiplied from those shown in Figure 3. This enable us to increase the number of honeypots to capture requests from different IP addresses. Note that either "multiple topic to one server" or "one topic to one server" can be selected from both the MQTT and CoAP honeypot. Instead of using the paho-mqtt module, as shown in Figure 3, we utilized Python with the aiocoap [16] module in the CoAP server, fake data registration client, and analyzer.

V. Analysis

A. Honeypot Setup

We created 1 MQTT honeypot and 3 CoAP honeypots, as shown in Table II. The MQTT honeypot utilized topics of the MQTT server to aggregate several fake sensor results into one MQTT server. Detailed topics on the MQTT server are shown in Section IV-B. We placed one topic on each CoAP server so that we prepared 3 CoAP servers. We added the word "monitor" to a part of the hostname, as shown in Table II. These hostnames emphasises that the hosts are monitor servers of an IoT system. The hostnames are registered to DNS before the observation term.

We deployed the MQTT and CoAP honeypots on different dates. This is why the deployment days differ. Unfortunately, due to both honeypot VM host failure and careless initialization of the backup host, we lost data after November 15, 2017 so that the following analysis was performed with data up until October 30, 2017.

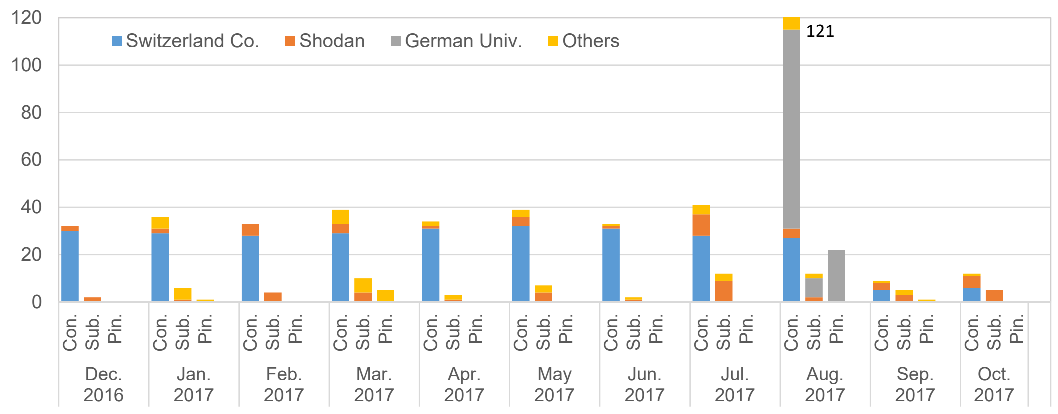


Figure 4. Number of MQTT requests per month

TABLE III. MQTT ACCESS SOURCES AT TCP SYN LEVEL (MORE THAN 3 TIMES)

Rank	IP address block or domain	Num of access	Organization
1	185.35.63.0/24	274	Security research team of Switzerland company
2	185.35.6.0/24	264	Security research team of Switzerland company
3	census.shodan.io	79	Shodan
4	134.147.202.0/24	76	Security research team of German university
5	members.linode.com	43	Hosting service A
6	zare.com	27	Hosting service B
7	180.149.126.0/24	20	Mongolian ISP A
8	180.149.125.0/24	15	Mongolian ISP B
9	research-scanner-dfn86.syssec.rub.de	12	Security research team of German university
10	150.100.253.0/24	10	Academic Network
11	188.166.165.0/24	6	Cloud service A
12	106.75.81.0/24	4	Cloud service B
13	182.86.142.0/24	3	Chinese ISP A
13	vmobile.jp	3	Japanese ISP A

B. *Analysis of MQTT*

We analyzed gathered MQTT requests minutely with net-mqtt-trace version 1.14 and Wireshark version 2.4.1.

1) Number of Requests: Figure 4 shows the number of the MQTT requests per month. The vertical axis shows the number of requests and the horizontal axis shows the months and types of request, such as MQTT connect requests (Con.), MQTT subscribe requests (Sub.), and MQTT ping requests (Pin.), respectively. The four layers per bar graph show a breakdown of the top 3 access sources shown in Section V-B2 and others. As shown in Figure 4, the request counts increased largely in August 2017. This is a result of a sudden large amount of requests from a German university, which is also described in Section V-B2. However, those request counts suddenly decreases in September 2017 including from other sources. This is possibly as result of the honeypots suddenly being omitted from Shodan's search result.

2) Source of Accesses: Table III shows slightly anonymized access sources attempting to access the MQTT honeypots at the TCP (Transmission Control Protocol) level more than 3 times. We counted TCP SYN packets from the Internet so that the counts listed in Table III include non-MQTT requests (e.g., send data of other protocols after the TCP handshake has been established). Thus, the sum of requests is a larger number than that shown in Figure 4. As shown at ranks 1, 2, 3, 4, and 9, security companies, services, and researchers frequently accessed the honeypot. We also plotted a breakdown of the top 3 access sources in Figure 4. As shown in the figure, the German university frequently accessed in August 2017, but did not access them at any other time. A Switzerland company sent continuous MQTT connect requests, but finished in September 2017 when the honeypots were omitted from Shodan. Shodan sent both continuous connect and subscribe requests.

However, as shown from lower ranks, tens of accesses came from ISPs (Internet Service Provider), hosting services, and cloud services. We also received mysterious accesses to the CoAP honeypots from ISPs (see Section V-C) so that those accesses may contain accesses from persons interested in exploiting MQTT or having a remotely dominated PC or server. There are 7 sources that accessed the honeypots twice and 28 sources that accessed them only once. Most of them were ISPs and so on, which indicates that a number of them were possible malicious sources.

3) Notable Accesses in MQTT: Below, we present the notable accesses to MQTT including minor visitors. As described below, a MQTT server that is exposed to the Internet receives non-MQTT requests to the 1883/tcp port and some of them were wrongly interpreted as MQTT requests. An effective MQTT server must be robustly implemented to handle these types of requests.

a) HTTP request after TCP establishment: We observed sent HTTP requests (starts from "GET HTTP/1.x...") after TCP establishment 7 times. 3 of them came from the vmobile.jp domain and the remaining 4 came from members.linode.com domain. In this case, character "G (0x47)" of "GET" word is treated as "MQTT Publish Ack Flag (0x47)" on the MQTT server so that we wrongly interpreted it as a mysterious MQTT connection.

b) SOAP request with MQTT Publish flag: We observed the following SOAP (Simple Object Access Protocol) request with the MQTT publish flag from the members.linode.com domain 3 times and from Cloud service A once. In this case, the character "<(0x3c)" of the "<soap:" notation is treated as "MQTT Publish Message Flag (0x3c)" on the MQTT server. This request also causes "crash at decode module in Publish.pm" error under the net-mqtt-trace program based analysis. Similar problems are likely to occur on the server side if the MQTT server has the same vulnerability.

```
<soap:Envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
<operationID>00000001-00000001</operationID>
</soap:Header><soap:Body>
<RetrieveServiceContent xmlns="urn:internalvim25">
<_this xsi:type="ManagedObjectReference"
type="ServiceInstance">ServiceInstance</_this>
</RetrieveServiceContent>
</soap:Body></soap:Envelope>
```

c) Multiple different protocol based requests recorded simultaneously: We observed the arrival of multiple different TCP-based protocol requests to 1883/tcp port from same IP address (zare.com) in the same day. The requests consisted of at least the following requests.

- HTTP protocols that started with "GET HTTP/1.0" or"GET /nice ports,/Tri\nity.txt.bak HTTP/1.0".
- SMB protocol that started with "PC NETWORK PROGRAM 1.0...".
- SIP protocol that started with "OPTIONS sip:nm SIP/2.0 Via: SIP/2.0/TCP nm;branch=foo...".
- Possible streaming protocol that started with "MMSNSPlayer/9.0.0.2980;...".
- Mysterious protocol that started with "dobjectClass0".
- Mysterious protocol that started with "(CONNECT_DATA=(COMMAND=version))".
- Mysterious protocol that started with "random1random2random3random4/".

d) Simultaneous requests from 2 domains: We observed that several domain pairs sent simultaneous requests. The detail of the requests is as follows. First, a part of the domain pair executes a TCP level observation that only sends SYN (does not reply to ACK even if the honeypot sends SYN/ACK) or sends SYN and RST (sends RST after SYN/ACK has arrived). Then, around 2000 seconds later, the other domain of the domain pair establishes a TCP connection and sends a MQTT connection request. Below, we present the pair of request sources. Source 3) achieved short intervals between requests, such as 7 or 8 seconds.

1) 185.35.62.0/24 and 185.35.63.0/24: 257 times
2) Mongolian ISP A and Mongolian ISP B: 13 times
3) amazonaws.com and 71.6.216.0/24: 2 times
4) 106.75.5.0/24 and sendclould.org: 1 time
5) 112.193.170.0/24 and 125.76.61.0/24: 1 time
6) 163data.com.cn and 175.152.30.0/24: 1 time

e) UDP MQTT request: Although MQTT permits UDP (User Datagram Protocol) requests, we recorded very few of them. We received 2 UDP MQTT requests from Chinese ISP A domain. The content of both requests are the same 41 bytes but we could not understand their purpose. The requested IP address also sent a TCP request on the same day.

```
30:27:02:01:00:04:06:70:75:62:6c:69:63:a0:1a:02:02:
6f:0c:02:01:00:02:01:00:30:0e:30:0c:06:08:2b:06:01:
02:01:01:01:00:05:00)
```

f) QoS of MQTT: MQTT has 3 levels of QoS which are described as at most once, at least once, and exactly once, respectively. In observation, all connections and MQTT ping requests came with at most once and all subscribe requests came at least one.

g) Read topics of MQTT: We observed read-all requests which are the simplest request, 70 times. However, we also observed some cross topic type requests indicating that some access source recognized a topic and sent requests about corresponding topic. The number of cross topic type requests for /humidity, /pressure, and /temperature were 6, 19, and 7, respectively. Such a MQTT interaction based analysis is cannot obtain prior works listed in Section II.

C. Analysis of CoAP

We analyzed gathered CoAP requests minutely with Wireshark version 2.4.1.

1) Number of Requests: Table IV shows the number of requests to the 3 CoAP honeypots. The number of requests are the aggregated result of the 3 CoAP honeypots because the number of requests were limited. Thus, the average number of requests to one honeypot becomes one-third of the recorded value. The number of requests are also limited because the observation term was only 3 months.

TABLE IV. NUMBER OF CoAP REQUESTS PER MONTH

Month	Number of requests
August 2017	42
September 2017	45
October 2017	51

2) Source of Accesses and Request Pattern: Table V shows slightly shaded all access sources to CoAP honeypots. They are separated into 3 large amount ones and 3 small amount ones. In large amount ones, Shodan is also seen in MQTT honeypot, but the left 2 access source were not seen in MQTT honeypot.

Table VI shows the request patterns in CoAP request packets from the analysis result of Wireshark. The large amount access sources have the same characteristics to the observed data on the MQTT honeypot, in which attempts were made to read data on the server. However, the last row of Table VI shows quite strange requests. We could not analyze the content of those requests, and their sizes were 542 bytes, which is comparatively larger than that of the frequent requests shown in large amount accesses.

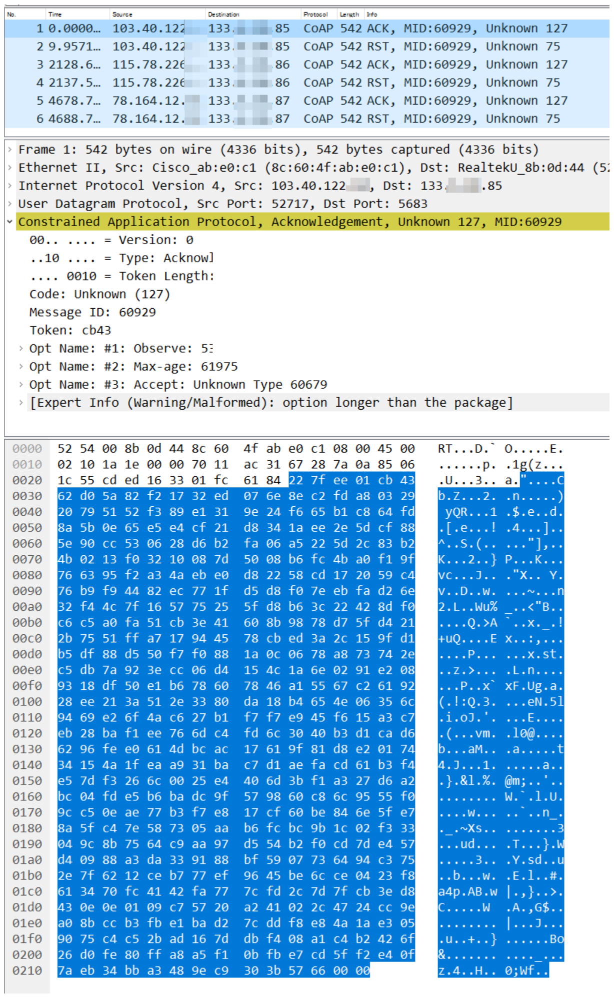
Figure 5 shows the detail of the requests categorized as "Unknown 127" by Wireshark. There are three requests from different IP addresses to different CoAP honeypots. These 3 source IP addresses are also listed as rank 4, 5 and 6 in Table V. As shown in Table V, the IP addresses are existing ISPs of different countries so that this series of requests are quite dubious. We estimated that those accesses were attempting to exploit vulnerabilities of some devices.

TABLE V. ALL CoAP ACCESS SOURCES

Rank	IP address block or domain	Num of access	Request size (bytes)	Organization
1	customer.tdc.net	93	63	Danish Telecommunication
2	census.shodan.io	30	65	Shodan
3	185.121.173.0/24	9	60	In & Datacenter service
4	jogjaringan.net.id	2	542	Indonesian ISP
5	115.78.226.0/24	2	542	Vietnamese ISP
6	78.164.12.0/24	2	542	Turkish ISP

TABLE VI. REQUEST PATTERN OF CoAP

Request	Source domain
CON, GET, /.well-known/core	Danish Telecommunication
CON, GET, End of Block #0, /.well-known/core	Shodan
CON, GET, /	In & Datacenter service
Unknown 127	3 ISPs



```
No. Time      Source          Destination      Protocol Length Info
  1 0.0000…   103.40.122[redacted] 133.[redacted].85  CoAP 542 ACK, MID:60929, Unknown 127
  2 9.9571…   103.40.122[redacted] 133.[redacted].85  CoAP 542 RST, MID:60929, Unknown 75
  3 2128.6…   115.78.226[redacted] 133.[redacted].86  CoAP 542 ACK, MID:60929, Unknown 127
  4 2137.5…   115.78.226[redacted] 133.[redacted].86  CoAP 542 RST, MID:60929, Unknown 75
  5 4678.7…   78.164.12.[redacted] 133.[redacted].87  CoAP 542 ACK, MID:60929, Unknown 127
  6 4688.7…   78.164.12.[redacted] 133.[redacted].87  CoAP 542 RST, MID:60929, Unknown 75
```

```
> Frame 1: 542 bytes on wire (4336 bits), 542 bytes captured (4336 bits)
> Ethernet II, Src: Cisco_ab:e0:c1 (8c:60:4f:ab:e0:c1), Dst: RealtekU_8b:0d:44 (52
> Internet Protocol Version 4, Src: 103.40.122[redacted], Dst: 133.[redacted].85
> User Datagram Protocol, Src Port: 52717, Dst Port: 5683
v Constrained Application Protocol, Acknowledgement, Unknown 127, MID:60929
    00.. .... = Version: 0
    ..10 .... = Type: Acknowl
    .... 0010 = Token Length:
    Code: Unknown (127)
    Message ID: 60929
    Token: cb43
  > Opt Name: #1: Observe: 5:
  > Opt Name: #2: Max-age: 61975
  > Opt Name: #3: Accept: Unknown Type 60679
  > [Expert Info (Warning/Malformed): option longer than the package]
```

```
0000  52 54 00 8b 0d 44 8c 60  4f ab e0 c1 08 00 45 00   RT...D.` O.....E.
0010  02 10 1a 1e 00 00 70 11  ac 31 67 28 7a 0a 85 06   ......p. .1g(z...
0020  1c 55 cd ed 16 33 01 fc  61 84 22 7f ee 01 cb 43   .U...3.. a."....C
0030  62 d0 5a 82 f2 17 32 ed  07 6e 8e c2 fd a8 03 29   b.Z...2. .n.....)
0040  20 79 51 52 f3 89 e1 31  9e 24 f6 65 b1 c8 64 fd    yQR...1 .$.e..d.
0050  8a 5b 0e 65 e5 e4 cf 21  d8 34 1a ee 2e 5d cf 88   .[.e...! .4...]..
0060  5e 90 cc 53 06 28 d6 b2  fa 06 a5 22 5d 2c 83 b2   ^..S.(.. ..."],..
0070  4b 02 13 f0 32 10 08 7d  50 08 b6 fc 4b a0 f1 9f   K...2..} P...K...
0080  76 63 95 f2 a3 4a eb e0  d8 22 58 cd 17 20 59 c4   vc...J.. ."X.. Y.
0090  76 b9 f9 44 82 ec 77 1f  d5 d8 f0 7e eb fa d2 6e   v..D..w. ...~...n
00a0  32 f4 4c 7f 16 57 75 25  5f d8 b6 3c 22 42 8d f0   2.L..Wu% _..<"B..
00b0  c6 c5 a0 fa 51 cb 3e 41  60 8b 98 78 d7 5f d4 21   ....Q.>A `..x._.!
00c0  2b 75 51 ff a7 17 94 45  78 cb ed 3a 2c 15 9f d1   +uQ....E x..:,...
00d0  b5 df 88 d5 50 f7 f0 88  1a 0c 06 78 a8 73 74 2e   ....P... ...x.st.
00e0  c5 db 7a 92 3e cc 06 d4  15 4c 1a 6e 02 91 e2 08   ..z.>... .L.n....
00f0  93 18 df 50 e1 b6 78 60  78 46 a1 55 67 c2 61 92   ...P..x` xF.Ug.a.
0100  28 ee 21 3a 51 2e 33 80  da 18 b4 65 4e 06 35 6c   (.!:Q.3. ...eN.5l
0110  94 69 e2 6f 4a c6 27 b1  f7 f7 e9 45 f6 15 a3 c7   .i.oJ.'. ...E....
0120  eb 28 ba f1 ee 76 6d c4  fd 6c 30 40 b3 d1 ca d6   .(...vm. .l0@....
0130  62 96 fe e0 61 4d bc ac  17 61 9f 81 d8 e2 01 74   b...aM.. .a.....t
0140  34 15 4a 1f ea a9 31 ba  c7 d1 ae fa cd 61 b3 f4   4.J...1. .....a..
0150  e5 7d f3 26 6c 00 25 e4  40 6d 3b f1 a3 27 d6 a2   .}.&l.%. @m;..'..
0160  bc 04 fd e5 b6 ba dc 9f  57 98 60 c8 6c 95 55 f0   ........ W.`.l.U.
0170  9c c5 0e ae 77 b3 f7 e8  17 cf 60 be 84 6e 5f e7   ....w... ..`..n_.
0180  8a 5f c4 7e 58 73 05 aa  b6 fc bc 9b 1c 02 f3 33   ._.~Xs.. .......3
0190  04 9c 8b 75 64 c9 aa 97  d5 54 b2 f0 cd 7d e4 57   ...ud... .T...}.W
01a0  d4 09 88 a3 da 33 91 88  bf 59 07 73 64 94 c3 75   .....3.. .Y.sd..u
01b0  2e 7f 62 12 ce b7 77 ef  96 45 be 6c ce 04 23 f8   ..b...w. .E.l..#.
01c0  61 34 70 fc 41 42 fa 77  7c fd 2c 7d 7f cb 3e d8   a4p.AB.w |.,}..>.
01d0  43 0e 0e 01 09 c7 57 20  a2 41 02 2c 47 24 cc 9e   C.....W  .A.,G$..
01e0  a0 8b cc b3 fb e1 ba d2  7c dd f8 e8 4a 1a e3 05   ........ |...J...
01f0  90 75 c4 c5 2b ad 16 7d  db f4 08 a1 c4 b2 42 6f   .u..+..} ......Bo
0200  26 d0 fe 80 ff a8 a5 f1  0b fb e7 cd 5f f2 e4 0f   &....... ...._...
0210  7a eb 34 bb a3 48 9e c9  30 3b 57 66 00 00         z.4..H.. 0;Wf..
```

Figure 5. Mysterious CoAP requests from 3 different country ISPs to the 3 CoAP honeypots on the same day

VI. CONCLUSION

We introduced how to create and operate honeypots that observe IoT-aimed protocols in this paper. We also presented the analysis result of 11 months of MQTT honeypot observations and 3 months of CoAP honeypot observations. The observation results show that MQTT seems to get greater interest than CoAP on the basis of access count. However, we also observed mysterious CoAP requests so that we believe that we have to take care for both protocols. Furthermore, we observed that the honeypot received some non-MQTT protocol based requests to 1883/tcp and some of them are wrongly interpreted as the MQTT protocol, which indicates that an effective MQTT server must be robustly implemented to handle these types of requests.

For future works, we first have to restart the observation and continue them for the long term because we lost more than a year's worth of data, as explained in Section I. Second, there are many other IoT-aimed protocols (e.g., Advanced Message Queuing Protocol that uses 5672/TCP) so that we are planning to implement them as honeypots.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Numbers 16K00071 and 19H04108.

REFERENCES

[1] D. H. Kim, J. B. Park, J. H. Shin, and J. D. Kim, "Design and Implementation of Object Tracking System Based on LoRa," *ICOIN 2017*, pp. 463–467, Jan. 2017.

[2] J. Joshi et al. "Performance Enhancement and IoT Based Monitoring for Smart Home," *ICOIN 2017*, pp. 468–473, Jan. 2017.

[3] https://shodan.io/ (Accessed on Sep. 10, 2019)

[4] K. Ito, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Primary Discussion about a Honeypot System for IoT Aied Protocols (in Japanese)," *IEICE Tech. Rep.*, Vol. 116, No. 522, pp. 103–108, Mar. 2017.

[5] "DinoTools/dionaea: Home of the dionaea honeypot" https://github.com/DinoTools/dionaea/ (Accessed on Sep. 10, 2019)

[6] "ysmal/dionaea: MQTT module" https://github.com/ysmal/dionaea/tree/master/modules/python/dionaea/mqtt/ (Accessed on Sep. 10, 2019)

[7] "mushorg/glutton: Generic Low Interaction Honeypot" https://github.com/mushorg/glutton/ (Accessed on Sep. 10, 2019)

[8] Y. M. P. Pa, S. Suzuki, K. Yoshioka, and T. Matsumoto, "IoTPOT: Analysing the Rise of IoT Compromises," *WOOT 15*, Aug. 2015.

[9] 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, Mar. 2016.

[10] M. Wang, J. Santillan, and F. Kuipers, "ThingPot: an interactive Internet-of-Things honeypot," arXiv: 1807.04114, pp. 1-8, Jul. 2018.

[11] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, "IoTCandyJar: Towards an Intelligent-Interaction Honeypot for IoT Devices," Black Hat 2017, pp. 1-11, Aug. 2017.

[12] OASIS. "MQTT V3.1.1 OASIS Standard," Oct. 2014.

[13] IETF. "The Constrained Application Protocol (CoAP)," RFC7252, Jun. 2014.

[14] "Eclipse Mosquitto^{TM} - An open source MQTT broker," https://mosquitto.org/ (Accessed on Sep. 10, 2019)

[15] "Eclipse Paho - MQTT and MQTT-SN software," https://www.eclipse.org/paho/ (Accessed on Sep. 10, 2019)

[16] "chrysn/aiocoap: The Python CoAP liary," https://github.com/chrysn/aiocoap/ (Accessed on Sep. 10, 2019)