Adaptive Observation of Emerging Cyber Attacks targeting Various IoT Devices

1st Seiya Kato
Yokohama National University
Yokohama, Japan

2nd Rui Tanabe
Yokohama National University
Yokohama, Japan

3rd Katsunari Yoshioka
Yokohama National University
Yokohama, Japan

4th Tsutomu Matsumoto
Yokohama National University
Yokohama, Japan

***Abstract*—For years, honeypots have been a valuable tool for observing cyber attacks. But in the age of Internet-of-Things (IoT), where various kinds of devices are being connected to the Internet, honeypots need to achieve diversity and interactivity. In this paper, we propose X-POT, an adaptive honeypot framework that emulates various IoT devices while maintaining a certain level of interactivity. We use components to observe attacks on all TCP ports and by conducting an Internet-wide scan of relevant hosts, we collect responses from real devices. We then selectively choose them as honeypot responses and observe attacks targeting vulnerable IoT devices. We implemented an HTTP honeypot with X-POT framework and exposed it on the Internet for 2 months. We observed 4,729,097 HTTP requests on 64,912 ports and captured 1,276 malware samples. Moreover, we were successful in observing attacks targeting different services such as Docker API and CouchDB. We compared our system with well-known monitoring systems and obtained 669 types of attack defined by an open source Intrusion Detection System, which is up to 2.2 times higher than other systems, and collected 284 malware samples, which is up to 3.5 times higher than other systems. To this end, we reveal our malware datasets for interested researchers.**

***Index Terms*—Internet of Things (IoT), Honeypot, Network Monitoring, Internet-wide scan**

I. Introduction

With the rise of Internet of Things (IoT), billions of devices are being connected to the Internet. They cover a variety of protocols and applications that, IoT devices are used in various fields [1]. Not only in the consumer space but also industries have started to develop IoT applications and the whole trend seems to continue [2]. Consequently, cyber attacks targeting IoT devices are emerging and yet IoT malware families, such as Bashlite [3], Mirai [4] and their variants [5] have already compromised a large number of devices.

To tackle this problem, security researchers have shed light into the field of IoT security. Honeypot have long been a vital component for network monitoring. Various honeypots for observing attacks targeting IoT devices have been proposed [6]–[9]. However, due to the variety of devices seen on the Internet and their device-specific vulnerabilities, honeypots face several challenges. First is *diversity*: the ability to observe attacks targeting different devices. More and more devices are being connected to the Internet and yet, devices that run services on different ports are under attack. Unless monitoring sensors have a certain level of diversity, it would be difficult to observe trends in the threat landscape. Second is *interactivity*: the ability to emulate a specific device with accuracy. For a while, IoT malware had been targeting devices that run Telnet services with weak or default password. Recently, attacks targeting device-specific vulnerabilities have increased [10]–[12]. In case of network monitoring, an attack may not be observed unless sensors have a certain level of interactivity.

For these reasons, honeypots that can observe attacks in both variety and detail are required. Prior work has already followed this idea by accumulating the responses of various devices on the Internet and used machine learning techniques to learn behavioral knowledge [13]. We were inspired by this work and present a simple adaptive honeypot framework that scans related hosts and select characteristic responses to achieve interactivity. We also introduce the concept of darknet monitoring into honeypots in order to improve diversity. Moreover, the objective of our study is to investigate the emerging attacks targeting IoT devices and reveal what is going on.

In this paper, we propose X-POT, an adaptive honeypot framework that emulates various IoT devices and services while maintaining a certain level of interactivity. The word "X" stands for unknown device or service, meaning that X-POT is a honeypot that mimics various devices and services without any prior knowledge about them. We first use components that monitor attacks on all Transmission Control Protocol (TCP) ports to detect the increase of connections on specific ports. By performing an Internet-wide scan, we then collect responses from devices on the Internet. We selectively adapt them as honeypot responses to emulate the interaction between the targeted device. Concretely, responses are tested and evaluated based on the attackers' reaction to see which response would attract more attacks. This way, honeypot operators can start X-POT even with zero knowledge about the targeted devices.

During the experiment, we implemented a Hypertext Transfer Protocol (HTTP) honeypot with X-POT framework (henceforce HTTP X-POT). In the first experiment, we first deployed a low-interaction honeypot that simply responds on all TCP ports at 3 Internet Protocol (IP) addresses on Amazon Web Service (AWS). By seeing their incoming packets, we conducted an Internet-wide scan and collected response candidates during July 22th 2019 to March 15th 2020. We then deployed HTTP X-POT using the collected responses and by further scanning the Internet during March 15th 2020 to May 20th 2020 at 23 IP addresses. By applying 17,672 responses to

978-3-903176-32-4 © 2021 IFIP

HTTP X-POT, we were able to observe 4,729,097 HTTP requests on 64,912 ports. We were successful to emulate vulnerable services and observe attacks against famous services such as Docker Application Programming Interface (API) and CouchDB. We also captured 1,276 malware samples, most found to be targeting IoT devices.

In the second experiment, we compared the performance of HTTP X-POT with three different monitoring systems (henceforce sensors). These sensors observe or respond on all TCP ports. Each sensor, including HTTP X-POT, were deployed at 3 IP addresses on AWS, DigitalOcean, and Vultr and observed attacks during June 22nd 2020 to July 21th 2020. We also used the response candidates found in the first experiment. As a result, by using a open source Intrusion Detection System (IDS), HTTP X-POT observed 669 types of attacks, including six unique Remote Code Execution (RCE) attacks that were not observed in other sensors. The number of attack types were up to 2.2 times higher than other sensors. Moreover, HTTP X-POT collected 284 malware samples, which was up to 3.5 times higher than other sensors. These samples were divided into 14 different malware families using a well known labeling tool. Our main goal is to find out the trends in the raging threat and show insight on attacks targeting IoT devices. For this reason, we reveal our malware datasets on our homepage "https://ipsr.ynu.ac.jp/iot/index.html", hoping to provide hints on developing countermeasures.

The contributions of this paper are:

1) We propose X-POT, an adaptive honeypot framework that emulates various IoT devices. By monitoring attacks on all TCP ports and performing Internet-wide scans, we use their responses to observe attacks targeting vulnerabilities and maintain a certain level of interactivity.
2) During March 15th 2020 to May 20th 2020, we deployed an HTTP honeypot with X-POT framework. We observed 4,729,097 HTTP requests on 64,912 ports and captured 1,276 different malware samples. By adapting 17,672 unique responses, we were successful to emulate several services along with their vulnerabilities.
3) By comparing HTTP X-POT against three different well-known monitoring systems, HTTP X-POT observed attack alerts up to 2.2 times higher and captured malware samples up to 3.5 times higher than other sensors. We reveal our malware datasets for interested researchers.

The remainder of this paper is structured as follows. In Section 2, we introduce related work. In Section 3, we propose X-POT. In Section 4, we show Internet-wide scan results and observation results of HTTP X-POT and in Section 5, we compare HTTP X-POT with other existing monitoring systems. In Section 6, we describe ethical considerations and limitations of our work. Finally, we conclude in Section 7.

II. Related Work

Honeypots are widely used to obtain threat information, capture malicious binaries, and detect previously-unseen attacks. Honeypots are mainly classified as Low/High based on the degree of interaction, or even as server honeypots or client honeypots based on the direction of interaction [14]–[16]. In this study, we focus on the server honeypots, especially systems used for observing IoT related attacks.

In recent years, seeing the increasing number of attacks targeting IoT devices, several IoT honeypots have been introduced. IoTPOT [6] is the first low-interaction honeypot aimed to observe IoT related attacks over telnet services. However, low-interaction honeypot including dianoea [17] and cowrie [18] cannot fully simulate device specific behaviors that, it is difficult to observe attacks that target device-specific vulnerabilities. Moreover, Darknet [19] and similar monitoring systems [20]–[22] have clear limitation due to their low-interactivity and can not observe such attacks.

On the other hand, SIPHON [7] is a high-interaction honeypot that uses a proxy server called Wormhole to forward packets to IoT devices and observe attacks on multiple devices. Virtual Private Network (VPN) forwarded Honeypots [8] is another high-interaction honeypot that uses VPN to forward packets to IoT devices. Honware [9] is also a high-interaction honeypot that uses customized kernels to launch multiple firmware and observe attacks on various devices. High-interaction honeypots using actual devices and firmware can observe attacks in depth. However, high-interaction honeypots can be implemented only when the devices or their firmware are available for honeypot use [23]. Yet, it is difficult to prepare the devices or firmware without knowing the target devices. To deal with such unknown attacks, systems that automatically generate effective responses have been developed [24], [25]. Still, these systems cannot cope with a large number of devices because they need the traffic of real devices to generate responses. IoTCandyjar [13] is an intelligent honeypot for observing attacks in quality and quantity. By collecting real host responses and using machine learning techniques, it can adapt to unknown attacks and responds meaningfully to the attacker using Markov decision process.

Our proposed method is similar to IoTCandyjar in that using real host scan results as honeypot responses. However, we scan network related hosts and use simple methods to select characteristic responses. Furthermore, our proposed method is more adaptive to the emerging attacks because it continuously observes attack trends on all TCP ports, and conducts network scans when an increase of connection attempts is observed. To put it differently, rather than improving existing honeypot techniques, we focus on observing attacks targeting IoT devices and show insight on what is going on.

III. X-POT

In this section, we propose X-POT, an adaptive honeypot framework that emulates various IoT devices by using responses collected by Internet-wide scans. In section III-A, we outline the basic design of X-POT, and in section III-B we explain how we collect and adapt responses. In section III-C, we introduce HTTP X-POT which is a honeypot observing on all TCP ports implemented by the X-POT framework.

A. X-POT Design

Attackers often scan a wide range of IP addresses in order to spread their infection. These IP addresses are often chosen randomly and that such activities can be observed through honeypots sensors. However, we need to improve the observation capability of honeypots because of the increasing variation of ports used by devices on the Internet. Attackers are also targeting such devices and yet to their variety, it has become difficult to identify what kind of attack occurs on which port. Thus, we propose X-POT, a method to emulate the behavior of devices on the Internet by updating the response of a honeypot using the results of an Internet-wide scan. Figure 1 shows the basic idea of our approach. X-POT consists of an Inner Unit (Attack Monitor and Response Collector), which observes all TCP ports and collects responses by scanning the network, and Outer Unit (Service Emulator), which emulates the attack target using collected responses. The main goal of X-POT is to improve the observation ability of the honeypot. We now explain each component in detail as follows.

Attack Monitor: In order to understand the reality of the attacks on the Internet, it is better to actually observe the attacks. We can infer what kind of devices and software are targeted and what kind of vulnerabilities are exploited by analyzing the observed attacks. Although the attacks are becoming diverse, in many cases, attackers discover target hosts by network scans and check the responses of the hosts in order to see if the discovered hosts are indeed the target device. Therefore, we use an all-port honeypot to observe various attacks that occur on the Internet.

Response Collector: Some attacks are indiscriminate, while others are carried out after checking the response of the device and identifying it as the attack target [26], [27]. In order to adapt to these attacks, we use a high-speed network scanning tool to collect the responses from devices in the wild and adapt these responses to the honeypot. Before conducting network scans, we sanitize the scan payload by eliminating the parameters from the request and also manually confirm the payload of our scans in order to avoid spreading malicious payloads on the Internet. However, we note that it may not be a successful request and may not get an accurate response.

Service Emulator: The response obtained by the Response Collector is adapted to Outer Unit (Service Emulator), which is used for observing further attacks. This way, it is possible to emulate the target's behavior by using the target's response as a honeypot response. Attacks include backdoor tools, coin mining scripts, ransomware and other malicious binaries are downloaded. Therefore, the Service Emulator has a component that downloads files at that time from the download server if a Uniform Resource Locator (URL) with a command, such as *wget* and *curl*, are included in the received packet.

B. Honeypot Response

While the Internet-wide scan collects multiple responses from real devices, it is still uncertain which collected response would attract more attacks. There are various tactics to decide which response should be applied as honeypot responses based on their observation criteria. To solve this problem, X-POT randomly returns a response to each incoming request and determines the score of the response based on the client's reaction. A high score is given to a honeypot response when the client reacts to it. The responses and their scores are managed in the response value table (henceforce RV-table), and responses with higher score or those never applied as honeypot response are given higher priority. We will describe the response selection in details in the next section.

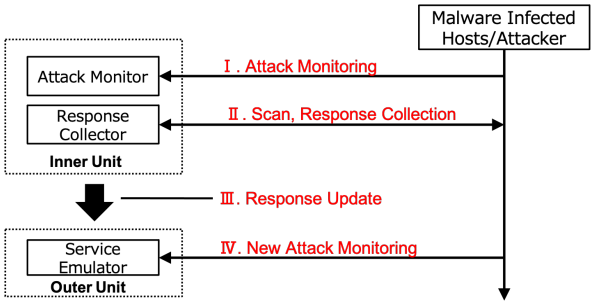

Fig. 1. Basic Idea of X-POT.

C. X-POT Implementation

In this section, we describe HTTP X-POT, an implementation example of an HTTP honeypot with X-POT framework. For the experiments in section IV and V, we implemented Attack Monitor and Service Emulator on the same machine and Response Collector on a different machine for efficient experiments. That is, HTTP X-POT consists of Web Honeypot, a group of honeypots that observe attacks on the Internet. Along with an analysis server that analyzes the observed attacks, scans the source network, and updates the responses of the honeypots. Figure 2 shows the system configuration of HTTP X-POT. We describe each component below.

Web Honeypot: Web Honeypot consists of multiple low-interaction HTTP honeypots distributed over the Internet that responds to incoming requests on all TCP ports. Each honeypot is a simple web server using python3 and they respond with a status code "200" and the string "It Works!" at an initial state. In addition, Web Honeypot has a function to transfer the daily observation log to the analysis server and has a function to receive and reflect the analysis results from the analysis server. During the experiment, we used an AWS EC2 t2.micro server as for the Web honeypot. The cost of using this type of instance was $10.26 per month, calculated by the AWS Pricing Calculator. We note that the average CPU usage was less than 10%. The response of Web Honeypot is managed using RV-table. Table I shows an example of RV-table for HTTP. RV-table consists of the response header (HTTP method, path, and status code) and the response body collected by Response Collector, as well as the port numbers on which the responses are collected and HTTP method used to collect the response. Moreover, a score is given to each response based on the reaction of the accessing hosts. When a response is newly collected, its score is initially set to 0.

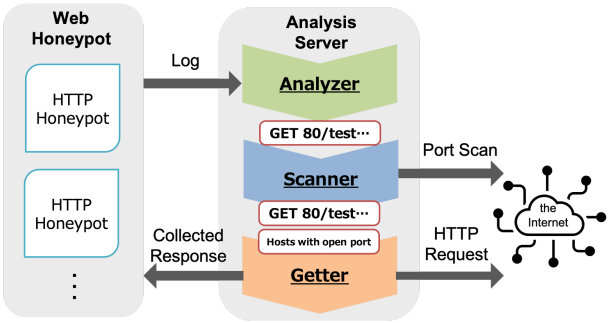

Fig. 2. Overview of HTTP X-POT.

TABLE I
EXAMPLE OF RESPONSE VALUE TABLE FOR HTTP.

Port	Method	Path	Header	Status_code	Body	Score
8888	GET	/hoge	md5_hash	200	md5_hash	0
8888	POST	/fuga	md5_hash	404	md5_hash	1
9000	GET	/	md5_hash	403	md5_hash	0
...						

The score becomes 1 if some reaction from the accessing host to the honeypot response is observed and the score becomes -1 if there is no reaction. When a port and path (hereafter called Access-URL) stored in the RV-table are accessed on Web Honeypot, a response with corresponding Access-URL is randomly chosen from all stored responses where score is 0 or 1. A response with a score of -1 will never be used in this case. Therefore, each collected response has possibilities to be tested as a honeypot response. We note that further evaluation on the response selection is our future work.

Analysis Server: Analysis Server consists of Analyzer, which analyzes the logs of honeypots, Scanner, which scans the Internet, and Getter, which collects response of hosts.

Analyzer: Analyzer collects logs and malware samples from Web Honeypot once a day. The HTTP requests with the same Access-URL are extracted from the one-day log, and the number of source IP addresses accessing to the Access-URL is calculated. Those HTTP requests observed from more than a threshold number of source IP addresses are considered emerging and sent to Scanner for collecting responses. The hash values of collected malware samples are sent to VirusTotal [28] and the results are stored. In addition, it synchronizes the RV-table of the honeypot and updates the scores.

Scanner: When an emerging HTTP request is reported from Analyzer, Scanner starts network scans. The scans are conducted using masscan [29] against all source IP addresses and their /16 network from which the HTTP requests were observed. Moreover, the /16 networks where the honeypots are located are also scanned. We note that only the corresponding port of the requests are scanned. The reason why we scan the senders of the emerging HTTP requests is that those hosts may have also been compromised and used as a stepping stone. Therefore we may be able to collect responses from the victim devices. Moreover, we scan the network around our honeypots. The fact that the HTTP requests are observed by the honeypot may indicate that the actual target devices can be around. Beforehand, we tested 1,294 IoT malware samples collected by IoTPOT [6] and confirmed that only 16% of them closed ports of vulnerable services while the original Mirai did close telnet and HTTP ports. We consider our approach of obtaining responses from compromised devices seems to be working at a certain degree.

Getter: Once a host listening on the target port is discovered by network scans, an HTTP request is sent to the host to collect HTTP response. This request is made from the Access-URL of the Analyzer. Seeing the corresponding result, we store the response consisting of the response header, response body and status code, and simultaneously the response header and body are converted to a hash value (md5) and the RV-table is updated. We note that if the response header is used as it is as Web Honeypot response, some field information, such as Content-Length, is deleted in order to avoid the possibility of a failure to respond correctly. Also, there is a possibility that we reproduce the attacks by scanning over the network because HTTP requests used in Getter are created based on the attacks observed in honeypots. Therefore, a header and data of a POST request is replaced with the values set by ourselves. We also manually check the Access-URL to ensure that the exploit is not included. Although we performed a strong sanitization process, the experimental results show that we were able to obtain adequate responses to emulate the target. Also, during the experiments, there were no gray cases where we needed detailed manual analysis, and it took us only a few minutes to examine each payload.

IV. OBSERVATION EXPERIMENTS

A. Attack Monitoring

By deploying HTTP X-POT on the Internet, we observed attacks from March 15th, 2020 to May 20th, 2020. During the experiment, we prepared HTTP X-POT on AWS, DigitalOcean, Vultr, Conoha, and Sakura Virtual Private Server (VPS), and observed the attack at 23 IP addresses. Table II shows the detail of observation periods and table III show the detail of their location. We prepared the Analysis Server on a commercial Internet Service Provider (ISP), and started scanning through a host with the web page stating that we will scan and obtain the response along with our contact information. We first deployed a low-interaction honeypot that simply responds with a certain message on all TCP ports at 3 IP addresses on AWS. By seeing their incoming packets, we conducted Internet-wide scans and collected response candidates during July 22th 2019 to March 15th 2020 (eight months). During this time we manually checked the Access-URL included in the request, while it took us only a few minutes to complete the process. We then deployed HTTP X-POT using the collected responses and by further scanning the Internet during March 15th 2020 to May 20th 2020 (two months). We modified response whenever there are accesses from multiple sources (roughly 5 IP addresses) that access to the same Access-URL in the same day.

As a result, HTTP X-POT observed 4,729,097 HTTP requests on 64,912 ports, from 85,849 unique source IP ad-

TABLE II
MONITORING SYSTEM.

	Monitoring System	#IP	Period
Observation Experiments	HTTP X-POT	23	March,2020 - May,2020
	(Response Collector)	3	July,2019 - March,2020
Comparison Experiments	No-Interaction	3	June,2020 - July,2020
	Handshaker	3	June,2020 - July,2020
	WebHoney	3	June,2020 - July,2020
	HTTP X-POT	3	June,2020 - July,2020
	(Response Collector)	23	March,2020 - May,2020

TABLE III
HTTP X-POT LOCATION.

VPS/Cloud	Country Code	# IP
AWS	AU,BH,BR,DE,HK,IE,IN,JP,KR,SE,SG,US	12
DigitalOcean	IN,NL,SG	3
Vultr	CA,FR,GB,JP,SG	5
Sakura	JP	1
Conoha	JP,SG	2
Total		23

dresses and captured 1,276 unique malware samples. There were 287 ports scanned and 17,672 unique responses obtained for 397 Access-URL that, we were able to observe emerging attacks including various Web user interface of IoT devices, such as routers, IP cameras, and capture IoT malware samples like Mirai and gafgyt. Some of the collected responses even contained device-specific values that, we observed characteristic attacks and malware samples. Other responses were also suspicious with multiple vulnerable service names.

B. Response Collecting

We were successful to collect a number of responses. Table IV shows the example of the collected responses; ports, methods, paths, the number of hosts with open ports, the number of obtained unique responses and scan date. The response of these Access-URL were changed during the observation period. The scan target was the port that had many requests from multiple hosts, such as 8000/TCP port related to the default port of web server and tmUnblock.cgi which is used in Cisco/Linksys router firmware and other targeted attacks. The number of hosts with open ports shows that there were many hosts with open port not on well-known port, and the number of unique responses shows that some hosts were running a web server on ports not on default ports.

The request and response information are useful for us to identify the device that attackers are targeting. For example, HTTP request information, such as *2375 GET /v1.16/version* and *200 GET /_search*, presume attack target devices are Docker API or Elasticsearch. Moreover, collected responses of *1200 GET /* indicates that Acess-URL related to RSShub. Although we have not observed any RCE for RSShub during the experiment, it is useful for implementing high-interaction honeypots to understand a potential attack target based on the content of the request or the obtained response.

There were also some unusual responses that contained more than 300 lines of different service and device names, such as GPON Home Gateway, Jenkins, and D-Link. We

TABLE IV
EXAMPLE OF COLLECTED RESPONSES (OBSERVATION EXPERIMENT).

Port	Method	Path	# Host	# Response (Unique)	Scan Date
85	GET	/	2,928	47	2019/12/15
999	GET	/	898	21	2019/11/09
1200	GET	/	953	20	2020/01/06
2375	GET	/v1.16/version	423	9	2019/09/17
8080	GET	/manager/html	9,544	245	2019/07/29
8083	POST	/manager_dev_ping_t.gch	5,388	160	2020/05/19
9200	GET	/_search	1217	179	2020/02/17
10000	POST	/session_login.cgi	7,453	2,565	2020/03/20
21080	GET	/	1,806	20	2020/04/24
50070	GET	/dfshealth.html	1,458	15	2020/04/29
55555	POST	/tmUnblock.cgi	1,273	4	2019/07/24

believe these responses are specially crafted for honeypot use to attract automated attacks that check for signatures of target devices. While we could use these crafted responses in our honeypot, it would decrease the capability to distinguish the target device as signatures of hundreds of different devices are contained in these responses.

C. Service Emulation

By using responses collected from real hosts, we were successful to observe several attacks that were not seen in the previous honeypot. This indicates that HTTP X-POT can emulate a devices that would be the attack target, and that the X-POT framework works well. The score of these responses were higher than those of others, confirming that the RV-table was working well. The detail of the attacks observed by changing the response are shown below.

Docker API: Figure 3 shows the attack observed on 2375/TCP port. Although the default response could only observe the first request (*2375 GET /containers/json*), the second request can be observed by changing the response to the one collected from the real host. The collected responses included Api-Version header, server header containing "Docker/18.06.1-ce (linux)" and the container ID, indicating that was the Docker API response. HTTP X-POT observed the request including the command and the container ID when sending this response to the attack packet. In this request, wget command was used to download a file, and the subsequent request was sent with a command to execute the downloaded file. The file was a high probability of a DDoS malware based on the report of VirusTotal. No requests were made to the container ID that executed the command, implying that the malware may have a function to connect to the C&C server.

Other attacks on 2375/TCP have also been observed to create a Docker container and download malware samples. The flow of attacks and information about the malware indicates that this attack was caused by Kaiji which is a XORDDoS malware family targeting IoT devices [30]. Since there are IoT devices that support Docker, such as OpenBlocks [31], and Docker can be used across a large range of IoT domains [32], it is likely that Docker is becoming the target of IoT malware.

CouchDB: Figure 4 shows that the attack observed on 5984/TCP port. When the HTTP X-POT returned a response with CouchDB in the server header, the client sent a request to create a user account using the PUT method. We confirmed

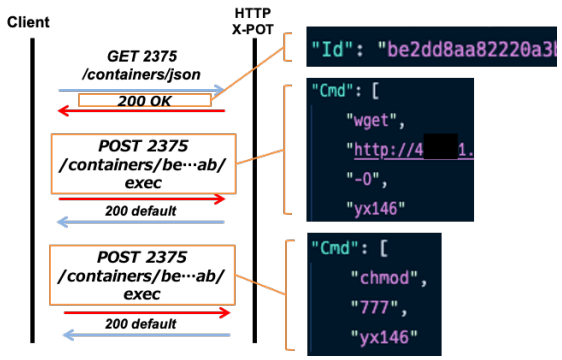

Fig. 3. Docker API Attack Flow (Observation Experiment).

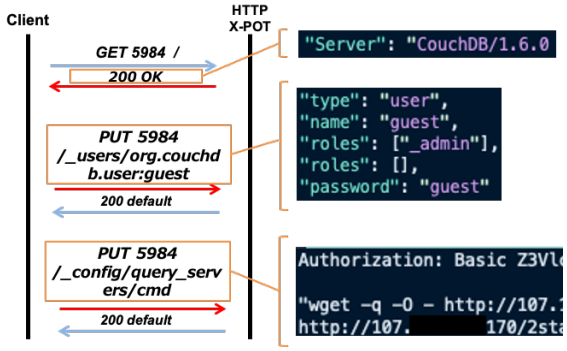

Fig. 4. CouchDB Attack Flow (Observation Experiment).

that the request to create a user were only observed when it returns a response with CouchDB version 2.1.1 or 2.2.0, and seems version 2.3.0 and 2.3.1 were not a target of the attack. After then, the username and password were used for basic authentication and the attacker tried to execute commands. The attacker used wget command to download a file (”2start.jpg”) and attempted to execute it. The downloaded file was a Bash script that tried to install a coin mining tool. While files with the same file names were regularly observed from April 20, 2020 to May 20, 2020, we confirmed that the attackers adjusted the scripts they used because of the different hash values of the files. After observing the first script on April 20, we confirmed that the script was modified to support both 32bit and 64bit Linux architectures on May 5. In addition, on May 13, we confirmed that the addition of download and execution commands for python scripts using urllib modules encoded in base64, and on May 20, the addition of scripts to retrieve network interface information using ifconfig and IP commands. This transition of script modification indicates that the attacker was trying to obtain more information from malware infected hosts by periodically updating the script.

D. Collecting Malware Samples

A total of 1,276 samples were obtained during the observation period, of which 797 could be reported by VirusTotal on May 30th, 2020. Table V shows the malware family names extracted using AVCLASS [33], their file names when downloaded, and the number of downloaded samples. Most of the malware samples were classified as Mirai or Gafgyt. In addition, many of the captured samples had file extensions of Central Processing Unit (CPU) architectures such as .x86 and .mips, which suggest that attackers tend to target IoT devices with various Operating System (OS) and architectures. The next most common family name was shell. Some scripts installed coin mining tools and others attempted to download and execute files from a download server using curl or wget command. Many of these scripts included multiple architectures strings, such as arm and mipsel. Some of the malware were simply ”Trojan-Downloader”, ”Coinminer.Miner”, or ”PowerShell.Gen”. Since the names of these samples were named quite differently by multiple anti-viruses, we estimated that the common family name could not be extracted. There were

TABLE V
MALWARE INFORMATION (OBSERVATION EXPERIMENT).

Malware Family	File Name	# Samples
mirai	sora.x86, LOTMOT.x86, dark.mips, ...	545
gafgyt	pXdN91.x68, x86_64, hell.x86, ...	116
shell	lilin.sh, axisbins.sh, Gbotbins.sh, ...	70
tsunami	arm7, x86_64, x86	22
shellbot	s.pdf	6
xorddos	L, 555	3
miraia	lmaoWTF/ZTE.sh, Chicken	2
miancha	download.exe	2
wget	sh	2
vobfus	no2.exe	1
valyria	222.hta	1
siscos	download.exe	1
netwalker	iguana2	1
juwp	arm.sh	1
ddostf	java	1
agentb	init.sh	1
No family name	-	22
Total		797

479 samples with no VirusTotal report. These samples were obtained when some attacks observed, such as the Realtek and the LILIN’s video recorder vulnerability. Thus, these samples may be relatively new samples in the threat landscape.

V. COMPARISON EXPERIMENTS

In this section, we compare the performance of HTTP X-POT with other well-known monitoring systems such as (1) ”No-Interaction” sensor that does not reply to any incoming requests, (2) ”Handshaker” sensor that sends SYN-ACK to any incoming requests on any TCP port, and (3) ”WebHoney” sensor that sends the same pre-defined response back to any incoming HTTP requests on any TCP port. We installed these sensor in Singapore region of AWS, DigitalOcean, and Vultr, and observed attacks from June 22nd, 2020 to July 21st, 2020. Table II show the detail of observation periods and their IP addresses. In this experiment, we changed the response on 297 ports as explained in section IV. We conducted an evaluation experiment with Suricata [34] which is a famous open source intrusion detection system in order to understand the types of observed attacks. We used Suricata version 5.0.3 and 31,812 rules, such as Emerging Threats Open Ruleset [35].

A. Comparison to Existing Monitoring Systems

We first show the overview of our observation results. Table VI shows the number of packets, the number of unique source IP addresses, the number of destination ports that received one or more packets, the number of unique alerts

detected by Suricata, and the number of samples collected by each sensors. While (1) No-Interaction sensor received packets from many hosts, the number of alerts in Suricata indicate that it did not observe as many attacks as other sensors. Also, since the No-Interaction sensor does not respond and establish a connection with the client, the number of obtained samples were zero. (2) Handshaker received a certain number of payloads and collected some samples, and the number of observable attacks was higher than that of No-Interaction. Because the packets contained strings, such as "SMB 2.002", "Cookie:mstshash=Administr", and "Windows for Workgroups", most of the observed payloads were considered to be targeting Windows systems, such as Remote Desktop Protocol and Server Message Block. Handshaker captured samples when observing an attack against Cisco/LinkSys routers using HTTP with */tmUnblock* and */ctrlt/DeviceUpgrade_1* as paths. These attacks are related to IoT malware, such as Mirai and Bashlite, and also observed in the HTTP X-POT. (3) WebHoney received more packets and the number of alert types were higher. On the other hand, the number of the collected samples was not so different from that of Handshaker, which indicates that WebHoney did not observe many attacks leading to malware download. To this end, HTTP X-POT was able to observe 669 types of attacks, 2.2 times higher than No-Interaction sensor, 1.4 times higher than Handshaker, and 1.1 times times higher than WebHoney. Also captured 284 malware samples, 3.5 and 3.2 times higher than those captured by Handshaker and WebHoney.

B. Comparison with Suricata Alerts

Seeing the Suricata alerts, we were successful to observe a variety of attacks. Table VII shows the example of attacks detected only by individual sensors. (1) No-Interaction sensor had no unique alert but had scan alerts, such as Microsoft SQL Server and Virtual Network Computing. (2) Handshaker had alerts related to 3-way-handshake, Domain Name System (DNS) cache poisoning and login attempts. We found 7 unique cases, including a brute-force attack on Internet Message Access Protocol (IMAP) and Post Office Protocol (POP3), which were not observed by other setups. The number of alert of (3) WebHoney increased by nearly 100 compared to Handshaker. WebHoney was able to observe some alerts related to a web server. WebHoney observed 17 unique attacks, indicating that simply setting up a fixed-response web server would improve the observation capability to some extent. HTTP X-POT observed 70 unique alerts related to HTTP including six RCE attacks, which shows that using real host responses leads to attract more attacks.

C. Comparison with Collected Malware Samples

Finally, we explain the malware samples captured from each sensor. Besides (1) No-Interaction sensors, (2) Handshaker captured 82 samples that, we speculate that some clients send packets containing the exploit immediately after the 3-way-handshake is established. Returning a fixed-response did not improve the observation performance of attacks leading to exploits because the number of samples obtained by (3) WebHoney was 88, which was not so different from that of Handshaker. HTTP X-POT captured 284 malware samples and the number of samples detected by more than one anti-virus software in VirusTotal was 197. Similarly, the number of samples detected in VirusTotal was 64 for Handshaker and 56 for WebHoney. The number of unique label which was classified by AVCLASS was 9 for Handshaker, 9 for WebHoney, and 14 for HTTP X-POT, indicating that the our system was capable of collecting a variety of malware samples. A lot of samples were classified as Mirai and Gafgyt that, attacker were targeting IoT devices. These samples were observed on ports, such as 8088/TCP and 9200/TCP, where the responses were changed and not seen on WebHoney.

TABLE VI
OVERVIEW OF OBSERVATION RESULT (COMPARISON EXPERIMENT).

Monitoring System	# Packets	# IP	# Ports	# Alerts	# Samples
No-Interaction	**13,563,246**	**294,082**	**61,996**	**300**	**-**
Handshaker	**77,108,904**	**241,669**	**65,535**	**472**	**82**
WebHoney	**353,539,775**	**98,363**	**65,535**	**596**	**88**
HTTP X-POT	**651,749,242**	**137,852**	**65,535**	**669**	**284**

VI. ETHICAL CONSIDERATIONS AND LIMITATIONS

A. Ethical Considerations

HTTP requests: Some of the attack observed in the honeypot tried to login or executed arbitrary commands to affect the server. We need to be very careful about sending these requests to a real host as-is we might be considered as an attacker. It is possible to detect a certain number of attack requests by using existing IDS, but it is impossible to detect all of them due to the variety of the targets and attack methods. Therefore, in this study, we used manual confirmation to send a request to a real host based on the attack information we have observed. In addition, we checked to make sure that the request did not become an illegal request by deleting parameters. We note that these processes take only a few minutes.

The impact of network scan: It is possible to stress the network because we conduct Internet-wide scans. Therefore, we limited the scan rate to 10,000 pps and used a server with static IP address for the scan. Along with the network scanner, we also prepared a web server that clearly states the purpose of the scan and our contact information. Whenever contacted, we can remove the specific network from further scan.

B. Limitations

Honeypot Detection: Attack Monitor that listen on all ports with the same response is likely to be detected by an attacker as a honeypot because the behavior is very different from a general web server. On the other hand, Service Simulator is less likely to be detected as a honeypot because the response changes according to the Access-URL. In this experiment, since Atatck Monitor and Service Simulator were installed in the same server, we investigated how well the server is recognized as a honeypot by using Honeyscore [36]. Honeyscore

TABLE VII
EXAMPLE OF SURICATA ALERTS (COMPARISON EXPERIMENT).

Monitoring System	Suricata Alert
No-Interaction	None
Handshaker	ET INFO SOCKSv4 HTTP Proxy Inbound Request (Windows Source) ET SCAN Rapid IMAP Connections - Possible Brute Force Attack ET SCAN Rapid POP3 Connections - Possible Brute Force Attack
WebHoney	ATTACK [PTsecurity] CoronaBlue/SMBGhost DOS/RCE Attempt (CVE-2020-0796) ET EXPLOIT Oracle WebLogic - wls-wsat Component Deserialization Remote Code Execution Windows ET EXPLOIT Tomcat File Upload Payload Request (CVE-2017-12615) ET WEB_SERVER JAWS Webserver Unauthenticated Shell Command Execution ET WEB_SPECIFIC_APPS Apache Tomcat Possible CVE-2017-12617 JSP Upload Bypass Attempt
HTTP X-POT	ATTACK [PTsecurity] Oracle Weblogic _async deserialization RCE Attempt (CVE-2019-2725) ATTACK [PTsecurity] Confluence <6.14.2,6.13.3,6.12.3 Unauthorized RCE (CVE-2019-3396) ATTACK [PTsecurity] Drupalgeddon2 <7.5.9 <8.4.8 <8.5.3 RCE (CVE-2018-7602) ATTACK [PTsecurity] Drupalgeddon2 <8.3.9 <8.4.6 <8.5.1 RCE through registration form (CVE-2018-7600) ET EXPLOIT Linear eMerge E3 Unauthenticated Command Injection Inbound (CVE-2019-7256) ET EXPLOIT bin bash base64 encoded Remote Code Execution 2 ET EXPLOIT file_put_contents php base64 encoded Remote Code Execution 1/2 ET SCAN ELF/Mirai Variant User-Agent (Inbound) ET SCAN Hikvision IP Camera 5.4.0 Information Disclosure ET WEB_CLIENT Possible Confluence SSTI Exploitation Attempt - Leads to RCE/LFI (CVE-2019-3396) ET WEB_SERVER HTTP POST Generic eval of base64_decode ET WEB_SERVER Possible IIS Integer Overflow DoS (CVE-2015-1635) ET WEB_SPECIFIC_APPS Drupalgeddon2 <8.3.9 <8.4.6 <8.5.1 RCE Through Registration Form (CVE-2018-7600) TGI HUNT directory traversal chars in HTTP Request Header

analyze the behavior of the host, and then calculating the likelihood of a honeypot on a score of 0.0 to 1.0. From March 15th, 2020 to May 20th, 2020, we regularly surveyed the Web Honeypot which is the component of HTTP X-POT by using Honeyscore and found that 5 IPs were 0.0, 17 IPs were 0.3 and 1 IP was 0.8, implying only one host was recognized as a honeypot. We assume that this result caused by returning the response which is known as honeypot during Honeyscore investigation. Therefore, it is necessary to remove suspicious responses that may be considered honeypots.

Emulation Limit: Although HTTP X-POT focuses on HTTP, it is possible to emulate the target device as well as HTTP if the protocol is a string-based. However, it may not be possible to emulate the behavior of the target device if the attacker may check not only the response, but also device and software special behaviors, such as moving the direction of IP camera or page transitions and execution results of arbitrary commands. For our evaluation experiments, also, we limited the number of networks to be scanned, but it is possible that there are no hosts that could be attack targets. Therefore, more responses can be observed by more extensive scanning, while taking into account the stress on the target network. It is also possible to use public scanning services, such as Shodan and Censys, to obtain and use the results of the scans on specific port.

Validity of Response Value: The value of the response to a request is determined from the client's reaction in the proposed system. However, some hosts send a large number of requests regardless of the response and validate multiple vulnerabilities at the same time, which causes the response evaluated incorrectly. Further study is needed on how to evaluate the response. For example, a method that sets a high score when multiple hosts take reaction, or a method that uses a dynamic analysis system to forward from the malware to the proposed system and observe the behavior of the malware, would make it possible to investigate responses in accuracy.

Undetectable Attacks: In this paper, we evaluated the increase in the number of attacks in the proposed system used by using the number of alerts in Suricata, but we did not evaluate zero-day attacks or attacks with no signature. However, the number of alerts of Suricata and the number of samples were increased, and the ability to emulate the target devices, which indicate that the proposed system has a high possibility to respond to unknown attacks.

VII. CONCLUSION

As more and more IoT devices are connected to the Internet and device-specific vulnerabilities are attacked, we expect that honeypots that can adapt to various situations will be necessary. We therefore proposed X-POT, an adaptive honeypot framework to improve the observation capability of honeypots by using the responses collected from the host through Internet-wide scan without any prior knowledge of the target. We deployed HTTP X-POT on the Internet and observed attacks targeting various IoT devices with integrity and captured several malware samples. We evaluated that X-POT had a higher observation capability than other well-known monitoring systems. We further reveal our malware datasets for interested researchers, hoping to provide hints for developing countermeasures.

ACKNOWLEDGMENT

A part of these research results was obtained from the commissioned research by National Institute of Information and Communications Technology (NICT) , JAPAN. This research was partly conducted under a contract of "Research and development on IoT malware removal / make it non-functional technologies for effective use of the radio spectrum" among "Research and Development for Expansion of Radio Wave Resources(JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

REFERENCES

[1] A. Ghasempour, "Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges," *Inventions*, vol. 4, no. 1, 2019. [Online]. Available: https://www.mdpi.com/2411-5134/4/1/22

[2] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] T. Spring, K. Carpenter, and M. Mimoso, "BASHLITE family of Malware Infects 1 Million IoT devices," *Threat Post*, 2016.

[4] Antonakakis, Manos and April, Tim and Bailey, Michael and Bernhard, Matt and Bursztein, Elie and Cochran, Jaime and Durumeric, Zakir and Halderman, J Alex and Invernizzi, Luca and Kallitsis, Michalis and others, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017, p. 1093–1110.

[5] Akamai, "NEW TSUNAMI/KAITEN VARIANT: PROPAGATION STATUS," https://blogs.akamai.com/sitr/2018/09/new-tsunamikaiten-variant-propagation-status.html.

[6] Yin, Minn, Pa, Pa and Shogo, Suzuki and Katsunari, Yoshioka and Tsutomu, Matsumoto and Takahiro, Kasama and Christian, Rossow, "IoTPOT: Analysing the Rise of IoT Compromises," in *Proceedings of the 9th USENIX Conference on Offensive Technologies*, ser. WOOT, 2015.

[7] Juan David Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martin Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovicici, "SIPHON: Towards Scalable High-Interaction Physical Honeypots," in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, ser. ACM, 2017, p. 57–68.

[8] A. Tambe, Y. L. Aung, R. Sridharan, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, "Detection of Threats to IoT Devices Using Scalable VPN-Forwarded Honeypots," in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 85–96.

[9] A. Vetterl and R. Clayton, "Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Zero Days," in *Proceedings of the 2019 APWG Symposium on Electronic Crime Research*, ser. eCrime, 2019, pp. 1–13.

[10] Claud Xiao and Cong Zheng, "New IoT/Linux Malware Targets DVRs, Forms Botnet," https://unit42.paloaltonetworks.com/unit42-new-iotlinux-malware-targets-dvrs-forms-botnet/, 2017.

[11] A. T. R. Team, "The eCh0raix Ransomware," https://www.anomali.com/blog/the-ech0raix-ransomware, 2019.

[12] Ken Hsu, Zhibin Zhang and Ruchna Nigam, "New Mirai Variant Targets Zyxel Network-Attached Storage Devices," https://unit42.paloaltonetworks.com/new-mirai-variant-mukashi/, 2020.

[13] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia and Xin Ouyang, "Towards an Intelligent-Interaction Honeypot for IoT Devices," in *Black Hat USA*, ser. Black Hat, 2017, p. 1–11.

[14] N. Marcin, W. Matthias, S. Thomas, C., K. Christian, and S. Jochen, "A Survey on Honeypot Software and Data Analysis," 2016.

[15] Mokube, Iyatiti and Adams, Michele, "Honeypots: concepts, approaches, and challenges," in *Proceedings of the 45th Annual Southeast Regional Conference*, ser. ACM, 2007, pp. 321–326.

[16] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá, "Enabling an Anatomic View to Investigate Honeypot Systems: A Survey," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3906–3919, 2018.

[17] GitHub, "Dionaea," https://github.com/rep/dionaea.

[18] ——, "Cowrie," https://github.com/micheloosterhof/cowrie.

[19] Eto, Masashi and Inoue, Daisuke and Song, Jungsuk and Nakazato, Junji and Ohtaka, Kazuhiro and Nakao, Koji, "Nicter: A Large-Scale Network Incident Analysis System: Case Studies for Understanding Threat Landscape," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 2011, p. 37–45.

[20] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet Background Radiation," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. Association for Computing Machinery, 2004, p. 27–40.

[21] C. Fachkha and M. Debbabi, "Darknet as a Source of Cyber Intelligence: Survey, Taxonomy and Characterization," *IEEE Communications Surveys and Tutorials*, vol. 18, pp. 1–1, 01 2015.

[22] Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson, "The Internet Motion Sensor: A distributed blackhole monitoring system," in *Proceedings of Network and Distributed System Security Symposium*, ser. NDSS, 2005.

[23] Tomasz Grudziecki, Paweł Jacewicz, Łukasz Juszczyk, Piotr Kijewski and Paweł Pawliński, "Proactive Detection of Security Incidents Honeypots," ENISA, Tech. Rep., 2012.

[24] Leita, Corrado and Dacier, Marc and Massicotte, Frederic,, "Automatic Handling of Protocol Dependencies and Reaction to 0-Day Attacks with ScriptGen Based Honeypots," in *Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg, 2006, pp. 185–205.

[25] C. Leita, K. Mermoud, and M. Dacier, "ScriptGen: an automated script generation tool for Honeyd," in *21st Annual Computer Security Applications Conference (ACSAC'05)*, 2005, pp. 12 pp.–214.

[26] "Exploit database," https://www.exploit-db.com/exploits/48531.

[27] "Exploit database," https://www.exploit-db.com/exploits/37169.

[28] VirusTotal, "Free online virus, malware and URL scanner," https://www.virustotal.com/en, 2019.

[29] Robert Graham, "Masscan," https://github.com/robertdavidgraham/masscan.

[30] Augusto Remillano II, Patrick Noel Collado and Karen Ivy Titiwa, "XORDDoS, Kaiji Variants Target Exposed Docker Servers," https://www.trendmicro.com/en_us/research/20/f/xorddos-kaiji-botnet-malware-variants-target-exposed-docker-servers.html, 2020.

[31] Plat'Home, "OpenBlocks IoT VX2 - Plat'Home Co.,Ltd." https://www.plathome.com/products/openblocks-iot-vx2.

[32] M. S. Abdul, S. M. Sam, N. Mohamed, K. Kamardin, and R. A. Dziyauddin, "Docker Containers Usage in the Internet of Things: A Surveyn," *OIJI SPECIAL ISSUE ON TECHNOLOGY AND INFORMATICS*, vol. 7, 12 2019.

[33] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVclass: A Tool for Massive Malware Labeling," in *Research in Attacks, Intrusions, and Defenses*, F. Monrose, M. Dacier, G. Blanc, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2016, pp. 230–253.

[34] Suricata, "Open Source IDS / IPS / NSM engine," https://suricata-ids.org.

[35] Emerging Threats Labs, "Emerging Threats rule," https://rules.emergingthreats.net/.

[36] Shodan, "Honeyscore," https://honeyscore.shodan.io.