Computers and Electrical Engineering 102 (2022) 108212

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Using honeypots to model botnet attacks on the internet of medical things[☆]

Huanran Wang[a], Hui He[a,*], Weizhe Zhang[a,b], Wenmao Liu[c], Peng Liu[d], Amir Javadpour[e]

[a] *School of Cyberspace Science, Harbin Institute of Technology, Harbin, China*
[b] *Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China*
[c] *NSFOCUS inc., Beijing, China*
[d] *Pennsylvania State University, United States*
[e] *Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China*

ARTICLE INFO

Keywords:
Internet of things
Botnet
Attack pattern
Control period
Internet of medical things

ABSTRACT

Corona Virus Disease 2019 (COVID-19) has led to an increase in attacks targeting widespread smart devices. A vulnerable device can join multiple botnets simultaneously or sequentially. When different attack patterns are mixed with attack records, the security analyst produces an inaccurate report. There are numerous studies on botnet detection, but there is no publicly available solution to classify attack patterns based on the control periods. To fill this gap, we propose a novel data-driven method based on an intuitive hypothesis: bots tend to show time-related attack patterns within the same botnet control period. We deploy 462 honeypots in 22 countries to capture real-world attack activities and propose an algorithm to identify control periods. Experiments have demonstrated our method's efficacy. Besides, we present eight interesting findings that will help the security community better understand and fight botnet attacks now and in the future.

1. Introduction

The rapid development of the Internet of Things (IoT) and the Internet of Medical Things (IoMT) can provide fast, cost-effective, and safe solutions for many applications, such as drones and Smart City [1]. However, numerous vulnerable IoT devices are low-hanging targets with sensitive user privacy for botnet organizers to launch large-scale botnets [2].

Report [3] revealed that most botnets have no idle period. No sooner is one botnet extinguished than other botnets replace it. The intense competition between attackers generates vulnerable devices (also called bots) destined to be involved in different botnets sequentially or simultaneously. The captured attack records contain multiple botnet behaviors. Therefore, different botnet control periods separate the attack event sequence into temporal fragments. Besides, Internet users generate a massive amount of legitimate traffic, which becomes a sanctuary for malicious online activities from being detected. Network administrators could not effectively recognize these botnet "landscape changes" or respond quickly in many defense situations. The above dilemma is defined as the "temporal fragment problem" in Section 2.1.

☆ This paper is for special section VSI-iomt. Reviews were processed by Guest Editor Dr. Ashish Singh and recommended for publication.
* Corresponding author.
E-mail addresses: wanghuanran@hit.edu.cn (H. Wang), hehui@hit.edu.cn (H. He), wzzhang@hit.edu.cn (W. Zhang), liuwenmao@nsfocus.com (W. Liu), pliu@ist.psu.edu (P. Liu), a_javadpour@e.gzhu.edu.cn (A. Javadpour).
URLs: http://homepage.hit.edu.cn/huihe?lang=en (H. He), http://homepage.hit.edu.cn/wzzhang?lang=en (W. Zhang).

https://doi.org/10.1016/j.compeleceng.2022.108212
Received 16 January 2022; Received in revised form 21 June 2022; Accepted 29 June 2022
Available online 8 July 2022

1.1. Limitations of current approaches

Many solutions have been proposed for botnet detection, broadly divided into two categories: Signature-based methods [4] are mainly applied for large-scale analysis in practice. Feature-based solutions are widely explored in the academic community [5]. For instance, Li et al. [6] proposed a method that automatically detects unknown HTTP botnets with traffic features. They extract features from multiple attack surfaces and train classifiers using machine learning-based algorithms. However, no one can prove whether the selected features are enough for training.

As a remedy, Sun et al. [7] relieved the feature engineering burden by utilizing the historical events. They first established the attack model temporal based on an intuitive observation: bots tend to show time-related attack patterns within the same botnet. Their prerequisite is that one bot can only be controlled by one botnet during the observation period. However, there is a growing trend for multiple botnets contesting the same victim, which may mislead further security solutions.

1.2. Challenge and motivation

Honeypot technology is widely used for modeling attacker behavior and reducing botnet attacks on IoMT and is an essential motivation for this research. In the context of IoMT, devices can join several botnets simultaneously. Security analysts produce inaccurate reports when different attack patterns are combined with attack records. Botnet detection has been studied extensively, but no publicly available methods for classifying attack patterns based on control period information exist.

The above discussions provoke three urgent real-world research questions not addressed by prior work:

Q1: How to efficiently distinguish botnet attacks from legitimate traffic and relieve the burden of analysts from numerous IoT botnet activities for their tremendous volume and diversity?

Q2: How to infer the botnet patterns in the perspective of control periods and evaluate the botnet analysis method when most of the attack records are unlabeled?

Q3: What statistical findings can be found from IoT botnet inferring results and the collected real-world attack records?

To fill these gaps, we develop a data-driven study to detect botnets with four objectives: The first goal (to answer Q1) is to construct a set of honeypots that captures enough real-world attack activities. Our second goal (to further answer Q1) is to reduce the massive noise records by proposing a noise data filtering algorithm based on attack density. Our third goal (to answer Q2) is to infer control periods and group similar attack patterns of IoT botnets so that administrators can better understand the relationship and botnet control structures. Our fourth goal (to answer Q3) is to reveal data-driven findings that help the security community better understand IoT botnet attacks now and in the future.

1.3. Main contributions

In this paper, we investigate the threat of IoT device compromises in the masses and design three kinds of IoT-based honeypots to present the well-collected attacks data collected in the wild. We propose a noise data filtering algorithm to reduce the massive noise records based on attack density. In this way, the simplified attack data is initialized as session sequences of each bot. Then, we initially model the directly attack temporal information as Multivariate Hawkes Process. To infer different control periods of each bot, we propose a dynamic sliding window-based control period inferring algorithm based on an intuitive hypothesis: **botnets tend to show time-related attack patterns**. We first investigate the correlation between random variables' distribution and the botnet inferring result in the evaluation phase. We then design a set of controllable simulation experiments to verify the effectiveness of our unsupervised learning method. Besides, we reveal data-driven findings that help the security community better understand IoT botnet attacks now and in the future.

To our best knowledge, this is the first data-driven study focusing on IoT botnets from the perspective of a control period. Corresponding to the three research questions mentioned above, our contributions to this work are as follows:

1. We design three kinds of lightweight honeypots to simplify the legitimate traffic distinction process. We release our well-collected attack data in the wild (from March 2019 to September 2021) to the security community for further study after publishment.
2. We propose a novel botnet inferring model that resolves the "temporal fragment problem" without the extra feature engineering endeavor. To further verify the effectiveness of our unsupervised learning method, we present the distribution patterns of our botnet inferring results and then design a set of controllable simulation experiments.
3. Based on the analysis results, we conclude eight interesting findings of the real-world IoT botnet attacks. Some highlights include: (1) there is a wide difference in cyber threats from countries over the world; (2) weak passwords are the most commonly exploited attack surface in IoT botnets; (3) observation shows that attackers tend to ensure a successful hit by repetitive attempts; (4) scannings with TCP forwarding actions have DDoS intent; (5) most botnet organizers can achieve their goals within 15 commands; (6) there is a high correlation between IoT cyberattacks and the COVID-19 epidemic; (7) weak IoT devices are controlled by multiple botnets in turn or simultaneously; and (8) IoT botnet attacks are highly correlated with bots under the same network segment.

The rest of the paper is organized as follows. Section 3 surveys the related works. Section 2 explains preliminaries. Section 4 explicates the attacker pattern model and the clustering algorithm. We evaluate our model and analyze the result in Section 5. In Section 6, we present the conclusion and future research.

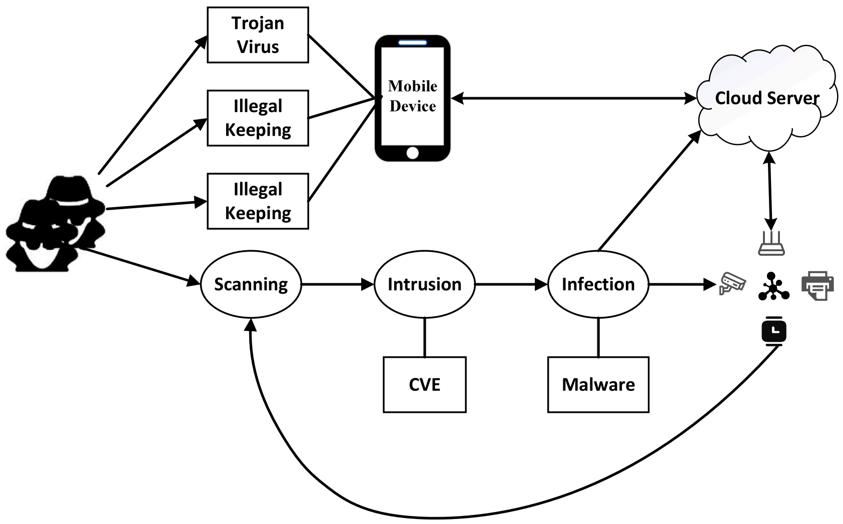


Fig. 1. IoT botnet attack phase.

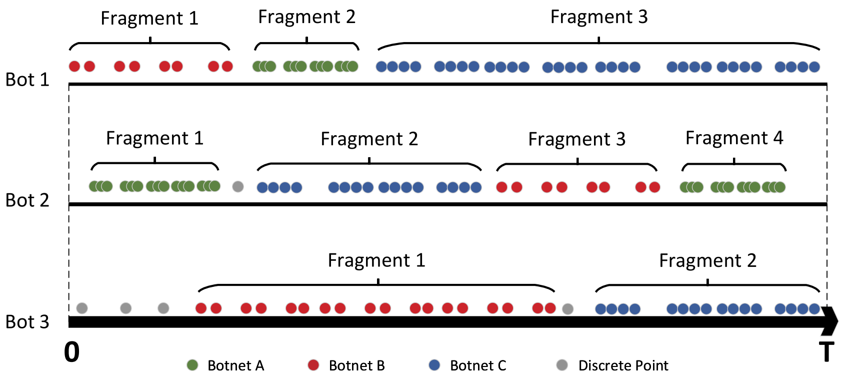


Fig. 2. Schematic diagram of temporal fragments. Each node represents an attack session.

2. Preliminaries

2.1. Temporal fragment problem

As defined in [8], an IoT botnet is a network of compromised IoT devices (called "Bots") under the control of a remote attacker (also called "Botmaster"). Multilayer command and control (C&C) is widely used for anonymity. Literature [4] summarizes the life cycle of a botnet into five stages: initial infection, secondary injection, connection, malicious command and control, update, and maintenance. Since the temporal information of the botnet attacks is our main focus, we provide a more macroscopical view by dividing the IoT attack integrity into three stages (shown in Fig. 1).

The first stage is scanning. Hackers attempt to incorporate devices that meet the invasion conditions into a list to be hacked through a wide range of port scans. The second stage is intrusion, in which attackers determine the common vulnerabilities and exposures (hereafter CVEs) that can be exploited for privilege escalation and remote login via the information received in the first stage. In the third stage, the attackers send tentative requests (such as Telnet sniffing) to confirm the connection and download the malicious scripts. Depending on each host CPU architecture, the code will trigger the corresponding executable file and exploit vulnerabilities in the device to turn it into a bot. These bots are then used in mining, performing DDoS attacks, propagating maliciously, and compromising privacy.

The vulnerability of these bots on the public network makes it possible for anyone to access them, leading to competition among attackers for resources. Different botnet control periods separate the sequence of attack events into temporal fragments.

Fig. 2 shows an intuitive schematic diagram of temporal fragments. During our observation period $[0, T]$, given three bots (controlled devices) which involved in three botnets. Given the control period of botnet bt_i and bot b_j, b_j is controlled by bt_i in period $[start, stop]$. The bot fragment of this control period are defined as: $f_{i,j} = (b_j, bt_i, [start, stop])$. We take temporal fragment as the basic unit of botnets, based on the following intuitive observations: **bots tend to show time-related attack patterns within**

the same botnet control period. In this paper, the term "attack pattern" refers to the temporal similarity of attack activities during the control period of the same organization or individual.

2.2. Attack pattern Hawkes process

The Poisson counting process is a statistical tool that has been proven to be greatly successful in modeling temporal events without capturing interactions between events [9]. Hawkes process is a particular case of point process which models the time intervals of random events. Given that a Hawkes process consists of K point processes (in this paper K bots) in the observation period, T are assumed as conditionally Poisson processes. Each bot includes a sequence of N attack events which are networking sessions essentially. An attack event is defined as $(t_n, a_n)_{n=1}^{N}$, where t_n denotes timestamp of attack n, and $a_n \in [1, K]$ denotes the bot to which the nth attack event belongs. In total, each set (t_n, a_n) represents an attack launched by an attacker a_n at time t_n. More specifically, following [9], intensity function (1) is used to characterize a Hawkes Process of each attack event n.

$$\begin{aligned}\lambda_n(t|t_n : t_n < t) &= \lambda_n(H_t)\\ &= \lambda_n^0 + \sum_{m=1}^{N} h_{m,n}(t - t_m)\\ &= \lambda_n^0 + \sum_{m=1}^{N} A_{m,n} \cdot W_{m,n} \cdot g_{\theta_{k,k'}}(\Delta t)\end{aligned} \tag{1}$$

where H_t denotes the event history before t, λ_n^0 denotes the background rate of n, which is intuitively described as random events that are not related to others. $h_{m,n}(t-t_m)$ is the time decayed influence between m and n, which can be decomposed by the product of $A_{m,n}$, $W_{m,n}$ and $\theta_{k,k'}$. $A_{m,n}$ is a binary adjacency matrix which is used to denote a directed edge from session m to session n. A Gamma prior on the weights, $W_{m,n}$ is used to denote the weight of edge between m and n. Then, $\theta_{k,k'}$ denotes the time-decayed influence between attacker n and m. Overall, $\lambda_n(t|t_n : t_n < t)$ measures the probability for the occurrence of a new session with the given history H_t based on random graph model. Therefore, we select $\lambda_n(t|t_n : t_n < t)$ to quantificat the temporal similarity between two attack session sequences. Establishing and using such a probability model requires a series of complex integrals, which is intractable. Therefore, we adopt the Gibbs sampling procedure followed literature [9] for estimation. For completeness and clarification, we simplistically restate the posterior distribution of $W_{m,n}$ in Eq. (2) and λ_n in Eq. (3).

Sampling weight matrix $W_{m,n}$:

$$\begin{aligned}&W_{p,q} \mid (t_n, a_n)_{n=1}^{N}, \theta_{p,q} \sim Gamma(\alpha_{p,q}, \beta_{p,q}),\\ &where\\ &\alpha_{p,q} = \alpha_W^0 + \sum_{n=1}^{N}\sum_{n'=1}^{N} \delta_{c_n,p}\delta_{c_{n'},q}\\ &\beta_{p,q} = \beta_W^0 + \sum_{n=1}^{N} \delta_{c_n,p}\end{aligned} \tag{2}$$

Sampling spontaneous attack rate λ_n:

$$\begin{aligned}&\lambda_n \mid (t_n, a_n)_{n=1}^{N}, \sim \text{Gamma}\left(\alpha_\lambda, \beta_\lambda\right),\\ &where\\ &\alpha_\lambda = \alpha_\lambda^n + \sum_n \delta_{c_n}\\ &\beta_\lambda = \beta_\lambda^n + T\end{aligned} \tag{3}$$

δ denotes the Kronecker delta function, which has been proven to work on this issue [9]. We use the inverse-scale parameterization of the gamma distribution.

3. Related work

Many endeavors have been done on Botnet analysis. We discuss the related works from three aspects according to our emphasis.

3.1. Honeypot study

Critical infrastructure relies on networks in a digitized world. These sensitive infrastructures are highly attractive for botnet organizers, who are frequently a step ahead of defenders [10]. Honeypots are decoys that lure cyber attackers into users' networks to counterbalance this situation. Honeypots can be classified into Low-Interaction Honeypots, Medium-Interaction Honeypots, and High-Interaction Honeypots.

Low-interaction honeypot [11] has a very limited set of interaction rules available for a visitor. Previous research about low-interaction honeypot includes generating intrusion and detection signatures of known attacks. They feature lightweight deployment and update routines without offering the complete functionality of the original device. While not much information is available about the 0 day vulnerability, the risk of compromising production systems by a successful honeypot attack is also low.

High-interaction honeypot [12] is a real, vulnerable device used to detect 0 day attacks. Due to high-interaction honeypots, the system services can be crashed by malicious commands. Therefore, an extremely resilient monitoring system is necessary for destructive readout.

Medium interaction honeypot [13] such as Cowrie provides a virtual file system with a fake shell to the attacker, which has proved to be efficient in the collection of autonomously spreading malware. Medium interaction honeypot is an ideal environment for tracking the state of IoT botnets because they try to combine the benefits of both Low-Interaction and High-Interaction Honeypots. Therefore, we developed a lightweight medium-interaction honeypot system by simulating vulnerable IoT-related targets with fidelity maintaining strategies. These strategies provide sufficient responses that known exploits await on certain services, ports, or protocols that trick attackers into sending their corresponding payloads.

3.2. Botnet detection

There are many methods proposed to detect a botnet. Signature-based methods are widely used for known botnets. For instance, Roesch et al. [14] proposed Snort, a lightweight network intrusion method by configuring a set of rules in advance. Snort is configured to run on infrastructures for long periods without administrative maintenance.

However, their method is highly dependent on the signature base and is useless for unknown attacks. Some works focus on machine-learning-based solutions employed to distinguish the botnet [15]. For instance, Li [6] proposed a method that automatically detects unknown HTTP botnets with traffic-based features. Preliminary experimental results show that the method can accurately detect botnets with low false-positive rates and automatically generate their signatures. Singh et al. [16] applied a random forest model to build a Peer-to-Peer botnet detection model in quasi-real-time. They provide a scalable implementation of real-time intrusion detection on open-source tools. PeerClean [17] utilizes the behavior-based features together with structured graph analysis. Their effectiveness is more robust and distinguishable than the host-level connection mode. In the experimental stage, they evaluate their work on the real traffic records, and the results show that PeerClean achieves high detection rates with few false positives. These methods tend to find the botnet behavior patterns from specific fine-grained features, which may only be effective for specific datasets.

As proved by Sun [7], bots that belong to the same botnet tend to launch temporally close attacks. Besides, the performance of the above methods is highly related to the feature quality because one never knows whether the used features are enough in the real world. As is well-known, detecting IoT attacks in the wild is an open issue because the real attack traffic will submerge in massive production activities.

Instead of distinguishing and isolating the botnet from the legitimate network, we first develop a set of IoT-related honeypots to collect the network events in the wild. Since our developed honeypots capture attacks with no actual or production value, any activity or traffic to the honeypot can be interpreted as an intrusion, fileless attack, or a probing effort [18]. We then infer the attacker pattern and group structure based on the collected data.

3.3. Attack pattern analysis

Classifying and clustering the attack behaviors is challenging due to the tremendous volume and diversity of collected honeypot data [19]. Attack pattern level study [20] can provide the researchers with a better understanding of emerging security trends. Study [21] utilized honeypots on a global scale to collect attack events within a framework for modeling and clustering attacker patterns. The results of these models reveal different interaction patterns from honeypots. They evaluated the proposed methods in a large dataset, including 167 million attacks.

To our best knowledge, related works on control period analysis are lacking. For example, Sun et al. [7] combine a Bayesian probabilistic graphical model and a graph-based clustering algorithm to identify latent influence between attackers. In this way, they could cluster the attacker activities and reasonably predict attacker behavior, detecting attacks based on previous behaviors captured by honeypots. The significant distinction between our work with them is that they lack the consideration of the botnet control period, which will cause incomplete analysis results. To help bridge this gap, we propose a data simplification method based on attack density and present a dynamic sliding window-based matching algorithm to infer the control intervals.

4. Attacker activities collection and modeling

Fig. 3 shows the workflow of our solution, including three steps: First, we develop worldwide scale honeypots for data collection. Second, we remove the repetitive records to collect a noiseless and well-formatted dataset. In the last step, we divide the attack sequence of each bot by partial similarity of temporal influence. We then cluster the control period-based attack patterns into botnets.

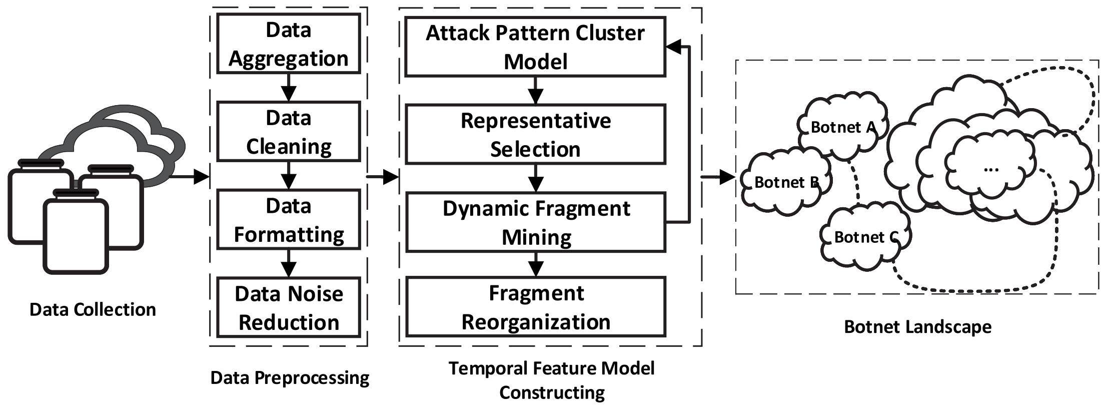


Fig. 3. Workflow of the framework.

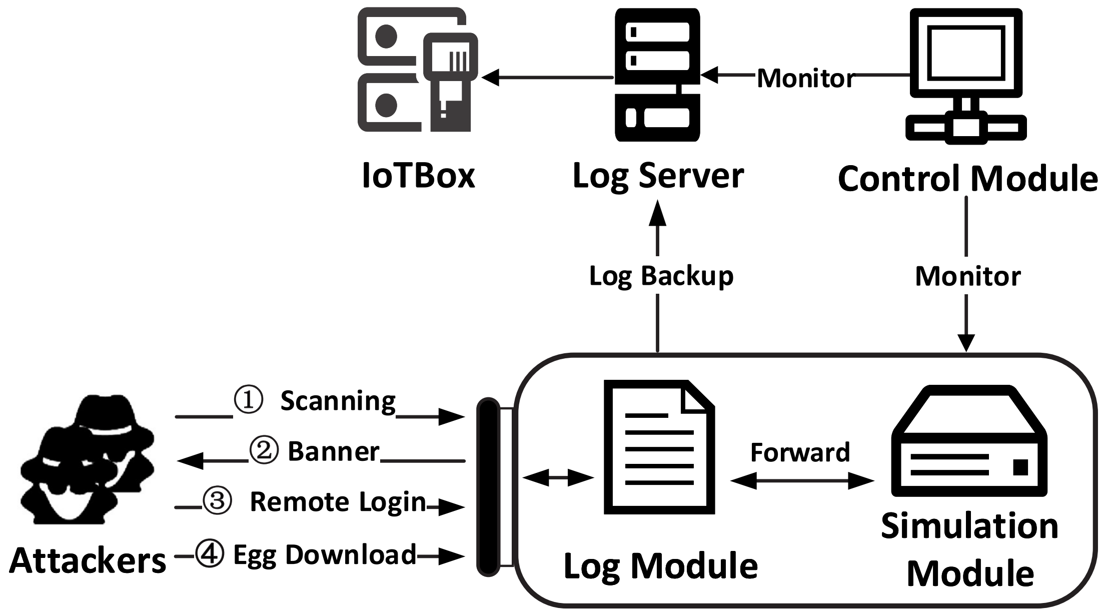


Fig. 4. Overview of honeypot system.

4.1. Honeypot deployment and data collection

This section answers the Q1 raised in Section 1.2 by designing a series of honeypots and proposing a data simplification algorithm. For convenience, we regard each network session as a single attack activity. Honeypots are decoys that lure cyber attackers into the network so that researchers can detect, deflect, and study hacking attempts to gain unauthorized access. Typically, a honeypot will present itself on the internet as a potential target for attackers, such as a server or other high-value asset, and it will inform and notify the defenders if unauthorized access attempts are made. Honeypots detect intruders before damaging or stealing data by taking proactive measures on the internal network. Traditional botnet capture methods include on-site forensics, user reporting, and vendor exchanges. The traditional approaches are typically manual.

Due to this, we developed a lightweight medium-interaction honeypot system comprising four components: remote control module, simulation module, log module, and IoT-based sandbox. We collect multiple attack features from our honeypots, including attack time, weak password attempts, protocol, commands, etc. Fig. 4 shows the overall structure of the honeypot system.

The CPUs of today can operate at different speeds thanks to many technologies, including Dynamic Frequency Scaling (DFS), Dynamic Voltage Scaling (DVS), and Voltage and Dynamic Frequency Scaling. We adopt DVFS to recommend suitable clusters within the cloud network. We can approximate the power-frequency relationship with a nonlinear and each level of productivity for Honeypots [22].

We formalized the honeypot service deployment process: The honeypots are sorted based on DVFS. We consider several virtual machines (VMs) and PMs in the data center, including N hosts. The initial VM is created on these hosts. Q is a set of m vectors

Table 1
Three types of honeypots.

Type	Simulation target
Protocol-based	Universal plug and play Server message block Open network video interface forum (ONVIF) Network time protocol (NTP) SSH/Telnet Transmission control protocol User datagram protocol Connection-less lightweight directory access protocol GPRS tunnel protocol Ubiquiti discovery protocol Advanced device discovery protocol
Vulnerability-based	CVE-2013-6117 (DHDVR) Netis backdoor of 53413 port
Application-based	Domain name system (DNS) OpenVPN Mail Microsoft SQL server (MSSQL) MySQL Redis Memcache

(m denotes the resources per host). N consists of hosts such as n_i, where $q_i = (q_{i1}, q_{i2}, \ldots, q_{im})$ represents the Quality of Service (QoS) values of the VM resources requested. $W = (w_1, w_2, \ldots, w_m)$ is the priority vector over the resources. The VM allocation unit determines the R_i classification value for each n_i after receiving the priority vector. Also, we create the initial VM on the host and have the appropriate amount of usability, with no more than two primary virtual machines created on it.

4.1.1. *Simulation module*

As discussed in Sections 2.1 and 3.1, the main attack surfaces in an IoT environment include obsolete versions of protocols, vulnerable applications, and firmware. To this end, we design medium-interaction honeypots with corresponding vulnerability attack surfaces. The simulation module is developed to obtain the trust of attackers. In general, we present three honeypots to simulate different vulnerable IoT-related targets with fidelity maintaining strategies. Table 1 shows the details of each simulation type.

The protocol-based honeypots simulate the IoT-related protocols. For example, UPnP (Universal Plug and Play) includes a set of network protocols (such as SSDP and SOAP) for home network discovery and device connection. Attackers first exploit the SSDP rules by requesting the SOAP port *1900* for accessible URLs to remote login. Therefore, we adopt WSGI to parse the requests and extract the *ST* field (used to store device type). We then recurrence the vulnerability of SSDP and SOAP to simulate UPnP. The vulnerability-based honeypots simulate vulnerable devices by deploying the execution environment with exposed vulnerabilities. For instance, *CVE-2013-6117* is exposed on Dahua web-enabled DVRs. By default, these devices communicate with an administrative service on TCP port *37777*. To enhance the fidelity of honeypots, we simulate it by deploying the firmware of Dahua DVR of version *v2.608* in the QEMU virtual environment. In this way, attackers can send a set of crafted Proof Of Concepts (POCs) to trigger the remote commands. Application-based honeypots utilize specific applications to attract IoT attacks. These honeypots can also capture the unknown CVEs because they deploy real-world vulnerable applications with embedded web services.

Besides, we present several strategies to prevent hackers from identifying our developed honeypots: First, we customize the QEMU configurations on each honeypot to ensure the hardware profiles (such as CPU and memory) similar to the real IoT devices. Second, some attackers probe whether the current environment is virtual. For example, they use Linux commands such as *dmidecode -s system-product-name* and *dmesg | grep -i virtual* to check the system attributes, which could expose the virtual machine information. Therefore, we redefine aliases of these commands in */etc/profile* to change the returned information. Third, we modify the system files (such as */proc/cpuinfo* and */etc/modules.conf*).

As we said, we have considered the simulation by deploying the firmware of Dahua DVR v2.608 on QEMU. Honeypots that are application-based are used to attract IoT attacks. Due to the embedded web services in these honeypots, it is possible to capture unknown CVEs. The following tasks have been considered:

- Through deploying an execution environment with exposed vulnerabilities, vulnerability-based honeypots simulate vulnerable devices.
- On web-enabled Dahua DVRs, for example, CVE-2013-6117 is exploitable. A TCP port 37777 is used by default for communication with administrative services.
- We simulate honeypots in QEMU using firmware of Dahua DVR v2.608 to enhance their fidelity.
- As these honeypots deploy real-world vulnerabilities with embedded web services, they can detect unknown CVEs.

4.1.2. *Log module*

We assume that any connections to our honeypots are suspicious because the honeypot has no production value. Therefore, all network traffic to and from the honeypots is recorded. To this end, we deploy an Apache server on each honeypot as a log module. To prevent it from being controlled by attackers, the simulated service port is modified to a random idle port, and the Apache public port is modified to the default port of the simulated service. Any attempt to connect the public port will be sent to the simulated server and saved as log files. The local logs are sent to the remote server every 10 minutes for backup.

According to our observation, the log server stores at most 200,000 logs per hour for each honeypot to achieve a trade-off between performance and efficiency. Specifically, we collect the system information (including CPU usage and memory usage), suspicious process (removing the necessary processes of honeypot), and traffic information (captured by tcpdump commands).

4.1.3. *IoT-based sandbox*

Although we only focus on the temporal features of the attack activities, we build a sandbox environment called IoTBox in the honeypot system to provide a dynamic analysis environment of the executable samples in future work. We implement it based on Detux, an open-source project. We use the QEMU hypervisor to emulate various CPU architectures, e.g., x86, x86-64, ARM, MIPS, MIPSEL.

To date, we can execute the file type of *elf* and *exe*. We implement a background program to record the system calls and runtime traffic. The traffic information is saved into *.pcap* files.

4.1.4. *Remote control module*

Honeypots deployed on the public network risk being invaded by attackers. To avoid this, the remote control module implements a communication monitoring mechanism to avoid becoming a new botnet incubator.

The communication monitoring mechanism sends heartbeat packets to the simulation module every 10 minutes to check honeypot survival status. A heartbeat packet includes the timestamp, a honeypot's IP source, and servers' CPU usage. We assume a threat has intruded on the honeypot if any exception occurs. The control module will terminate all active sessions and recover the honeypot to the original version. Then, the suspicious IPs are added to our block list for 30 minutes (because most malware-based attacks could finish in 30 minutes).

4.2. *Data preprocessing*

There are two challenges in processing attack records. Firstly, the massive raw records can rapidly overwhelm security analysts. Secondly, many repeated and incomplete attack events are captured in the wild. According to our observation, 16.42% of attack sessions are incomplete due to network or unexpected interruptions. 74.88% of all are less informative bots (IPs) with less than 30 occurrences. Therefore, we first integrate the captured data by standardizing the temporal and geographic information. We then remove the bots with less than 1000 records.

Statistically, about 23% of attackers attempt to detect whether the host is controlled by other attackers and terminate other control processes. For example: an attack record discovered on February 2, 2021: *ps -ef | grep -v 'ssh' | grep '[M]iner' | grep -v 'perl' | awk'if ($3 == 1) print $2' | xargs kill -9;/bin/busybox MIRAI*. These behaviors indicate that a bot exhibits different attack patterns from organizations or individuals. Therefore, we treat the infected devices that access multiple honeypots as different bots.

Algorithm 1 Attack Event Simplification Algorithm

```
Require: raw attack log base $B_R$
Ensure: simplified bot attack log base $B_S$
 1: for all $bot$ in $B_R$ do
 2:     s ← get the session sequence of $bot$
 3:     L ← get the interval list of $s$
 4:     D ← fit the interval distribution of $L$
 5:     mean_duration ← get the mean value of $D$
 6:     model ← DBSCAN(eps = mean_duration, minPts = 1)
 7:     clusters ← model.fit(s)
 8:     for all c in clusters do
 9:         $centroid_c$ ← get the centroid of $c$
10:         add centroid to simplified sequence $s'$ of bot
11:         add $s'$ to $B_S$
12:     end for
13: end for
14: return $B_s$
```

According to our observations, 95% attackers launch numerous repeated attacks to ensure a successful hit at short notice. Therefore, all attack activities in a botnet are mutually density-connected. Besides, if an attack activity is density-reachable from some cluster activities, it is part of the cluster as well. We propose an improved DBScan algorithm shown in Algorithm 1 to simplify the attack events. This algorithm works as follows: The session sequence of each bot is translated from the raw data by aligning on the timeline. We calculate the interval values between each adjacent session. Because network attacks are simultaneously affected by multiple random factors (such as attack cost, device vulnerability, etc.), the attack interval follows a normal distribution. We fit the session interval sequence to a normal distribution by Eq. (4) and compute its mean value μ.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{4}$$

The DBScan algorithm determines whether the nodes are density-reachable by customizing the *eps* and *minPts* parameters. The *eps* parameter is used as the distance threshold in neighborhood judgment. The *minPts* parameter is used to limit the minimum session number a cluster contains. We dynamically assign μ to *eps* and 1 to $minPts$.

4.3. Temporal feature modeling

After our botnet clustering endeavors, each cluster represents a botnet with a specific attack pattern, consisting of the bot sequence fragments (defined in Section 2.1) separated by control periods. Following the intuitive observations: **bots tend to show time-related attack patterns within the same botnet.** We propose a botnet clustering algorithm 2 based on control period detection to address the Q2 raised in Section 1.2, including four steps:

The first step (lines 2–4) presents the initial cluster procedure. Following literature [7], we first train the Bayesian probabilistic network model with Gibbs sampling for 1000 iterations to infer the model parameters, W, A, λ_n^0 (defined in Eq. (1)). We then obtain the latent influence between every two bots. Finally, we use the obtained intensity as the cluster weight to obtain the initial botnet clusters. Two sequences are similar when the intensity value exceeds 0.95 (set by user). Note that the *botClusters* in the third line contains bots with the temporal close patterns.

The second step (lines 6–9) aims to simplify the burden of data analysis. To this end, we select a representative from each initial cluster. We used a high threshold to limit the tightness of each cluster in the first step. Therefore, the representative selected at this step can present the attack characteristics of the whole cluster. The first step's clustering process ensures that each cluster's bot is highly consistent. The intensity function (defined in Eq. (1)) quantifies the temporal relationship between two bots, which we use as weights. We calculate the weight between every two bots in the same cluster. Finally, the maximum average weight bot is selected as the cluster's representative.

The third step (lines 11–21) aims to recursively mine the similarity among temporal fragments. We combine the representative in pairs and find the combination with the longest matching period. To this end, we map this problem to the longest common continuous substring (LCS) problem [23]. We adopt a dynamic sliding window strategy: The window size is initialized as X days, and the intensity value of each temporal subsequence window is recorded each time. The window expands Y days for each iteration until the entire observation window is traversed. Then, the window is initialized as X and moves Y days along the time axis. In this way, we find the longest similar periods among all the similar subsequence pairs. Based on the referring result, we divide the attack event sequence of each bot in the cluster. When the length of a sequence fragment is less than Z minutes, we believe that the length is not enough to express the long-term attack mode. The division procedure ends. After executing the third step, the algorithm will jump to the second step (the fifth line) and repeat all the above procedures until there is no division sequence. Note that *X*, *Y*, and *Z* are respectively initialized to *30*, *10*, and *50* in this case (or another value defined by network administrators).

The fourth step (lines 23–24) aims to recombine the mined fragments that should be time continuous. Since the above three steps will be repeated multiple times, the continuous-time sequences may be separated. Whether two fragments can be recombined is based on three rules: (1) if these two fragments belong to the same IP; (2) if these two fragments access the same honeypot; (3) if there are no other cluster members between their timestamps in the timeline.

Computational concerns. Our proposed method consists of five parts: data simplification in Algorithm 1 and the four steps in Algorithm 2. The first and fourth parts are serial relations, and the last four parts recursively according to the threshold condition (X, Y, Z). Since our proposed method is oriented to numerous and uncertain real attack records, only the approximate computational complexity is given here. Furthermore, assuming that the total data size is N, the space complexity of our proposed algorithm is approximate $O(N)$.

According to the above algorithm description, the first part has the same complexity as DBSCAN, which needs to find the eps neighbors of a specified data from the database. The scanning process can be completed in log time through optimization strategies (such as R trees), and the time complexity reaches $O(N\log(N))$. The second part is to estimate the Hawkes process parameters by Bayesian sampling. Assuming that the amount of data after simplification is N', it has been proved in Reference 3 that the time complexity can reach $O(N'\log N')$ after parallelization. In the third part, we here select a graph-based clustering approach following [7] which is based on the influence weight matrix inferred by the attacker activity model. The links between the considered attackers and the remaining ones need to be traversed through W in the clustering process. The worst case is that every two attackers are not in a cluster, and the complexity reaches $O((N')^2)$. In the fourth part, the algorithm generalizes the control period inferring the problem as an LCS problem, and the time complexity of the attack sequence comparison for each window is $N'\log N'$. Since the number of windows is reduced to a constant K by the constraint of the given threshold X, Y, Z, the overall time complexity is still $O(N'\log N')$. In the fifth part, we need to recombine the attack time fragments that have been clustered. The number of clusters and fragment lengths can be regarded as constants.

In summary, the approximate time complexity of our proposed method is defined in Eq. (5):

$$F(N) \approx O(N\cdot\log(N)) + T[2\cdot O(N'\cdot\log N') + O((N')^2)] \tag{5}$$

where T is the complexity required for recursion. Experiment 1 shows that N is an order of magnitude larger than N', so $F(N)$ is approximately $O(N\cdot\log(N))$.

Algorithm 2 Dynamic Sliding Window-based Control Period Inferring Algorithm

```
Require: attack matrix seqIpMat,
Ensure: the final fragment in the dictionary is tagged as part of a botnet fs final fragment result marked with botnet label in dictionary fs
 1: // Step 1: Attack Pattern Cluster Model
 2: $A, W$ ← train parameters of Bayesian model by Gibbs sampling with Eqs. (2) and (3)
 3: $botClusters$ ← the Bayesian model can be used to obtain the cluster results
 4: function InferFragment(botCluster)
 5:     // Step 2: Representative Selection
 6:     for all cluster $c$ in $botClusters$ do
 7:         Find the bot $b$ in $c$ whose mean intensity value with other bots is the largest
 8:         $add\ b\ to\ rs$
 9:     end for
10:     // Step 3: Dynamic Window Sliding
11:     for each two representatives combination $(b_i, b_j)$ in $rs$ do
12:         $(b_i.sequence, b_j.sequence)$ ← get the sequences of $(b_i, b_j)$
13:         $(f_i, f_j)$ ← get the longest similar subsequence pair of $(b_i.sequence, b_j.sequence)$
14:         add $(f_i, f_j)$ to fragment candidate list $frag\_candiList$
15:     end for
16:     $f_{start,stop}$ ← find the longest fragment in $frag\_candiList$ //$[start, stop]$ is the period of $f$
17:     if the length of $f_{start,stop} > Z$ then //$Z$ is the minimum value of fragment
18:         divide $rs$ based on time period $[start, stop]$ as $rs_{new}$
19:         $fs$ ← mark $rs_{new}$ with botnet labels
20:         InferFragment($rs_{new}$)
21:     else
22:         // Step 4: Merge with Condition
23:         $fs$ ← merge fragments which are time contiguous
24:         return fs
25:     end if
26: end function
```

5. Results and findings

In this section, we first present a data-driven analysis of our observation period. By analyzing the experimental results, we present some interesting findings. We evaluate our method on a large-scale dataset. All experiments are run on an Intel(R) Xeon(R) CPU X5650 @ 2.67 GHz with four clusters of 128 GB memory.

5.1. Dataset statistics and insights

In this section, we answer the Q3 raised in Section 1.2. To collect enough IoT botnet records for analysis, we developed 20 honeypots on 56 physical servers to continuously capture attack activities from November 2019 to September 2021, which will be updated in the future. Due to the limited equipment resources, we deploy all types of honeypots on each server by docker virtualization technology. In total, we deployed 462 honeypots distributed in 22 countries. To ensure that honeypots can collect attack activities worldwide, we deploy honeypots in Asia, Oceania, America, and Europe. The geographic attack distribution is shown in Fig. 5.

According to our observation, our deployed honeypots are targeted by bots worldwide. For example, on average, each honeypot captured attacks from 162 countries in March 2020. China faces the most severe security threat globally, accounting for 23.5% of all bots, followed by America (11.4%), Brazil (7.6%), and Burma (5.1%). **Finding 1: China faces the most cyber threats all over the world, followed by America, Brazil, and Burma.**

To exclude the inoffensive sniffing behavior from other security organizations, we search the bot IPs in VirusTotal and find that 86% of them have been marked as malicious. The rest of 14% benign traffic is removed from our collection. All IoT botnet attacks perform weak password attempts as the first step of intrusion. Included among the hacked (and vulnerable) passwords are default passwords used by manufacturers that give the appearance of IoT security layers. Table 2 shows the top 5 ranked username-password combinations and the related devices.

According to our statistics, the actual outcome in these situations is a larger attack surface of poorly defended endpoints for malicious actors to penetrate easily. Different botnets have precise requirements for the target device type. A clear target can significantly reduce the cost of vulnerability exploitation. We observe that bots on the same network segment try specific username-password combinations. For example, hosts under the 144.47.34.1/24 network segment only tried the password combination

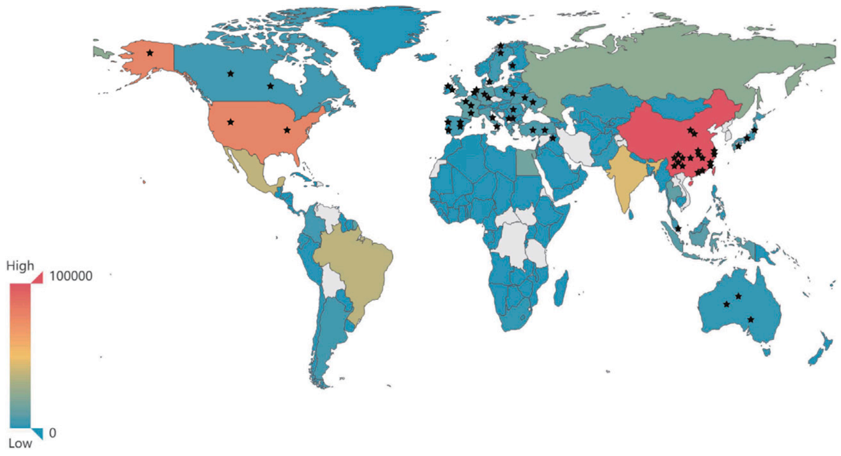


Fig. 5. Distribution of attacker countries. The black star in this figure represents the location where we deploy the honeypot servers.

Table 2
Top 5 ranked username-password attempts.

Rank	Username-password	Related target (Partial)
1	root-admin	Zcom wireless
2	root-	CellPlex, ADSL, AirStation, Equaliser4, Cyclades-TS800, Shiva, Juniper, Arkeia
3	root-default	Micronet Access Point
4	root-vizxv	Dahua DVR
5	root-t0talc0ntr0l4!	ThinkPHP, Control 4

root/t0talc0ntr0l4! in January 2020, which is the default password of Control 4 smart devices. **Finding 2: Weak password is the primary attack surface for IoT attacks, providing essential guidelines for analysts to classify botnets.**

39.5% of attack records are incomplete due to network reasons or unexpected interruptions. After removing the incomplete sessions, we obtain over 768, 742, 999 attack sessions during the 22-month observation period. Besides, according to our observations, 72% of attackers launch numerous repeated attacks to ensure a successful hit at short notice. For example, we captured attack traffic from *5.188.86.0 /23* on September 29, 2019. The attacker utilized 13 hosts on this network segment to launch 8,374,200 repeated SSH access requests within 10 min, with a success rate of only 4%. The highest frequency reaches 4450 visits per second. Because honeypots are not productive, we believe such behavior contains no DDoS intentions. **Finding 3: The attackers of the Internet of Things tend to perform repeated attacks in massive numbers to ensure a successful hit within a short period.**

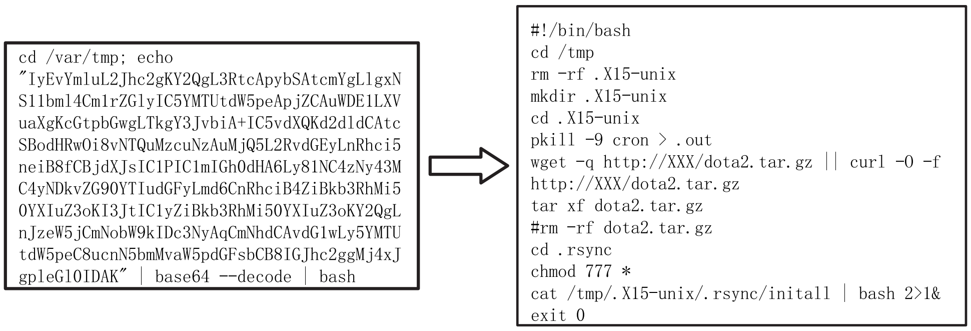
Based on the attack integrity discussed in Section 2.1, we observe that 95.7% of the complete attacks only performed scanning behaviors. Among these scanning behaviors, 38% of them utilize our honeypots as redirectors to access another attack target by TCP port forwarding. Most redirect targets have clear production value, including public facilities IPs, pornographic websites, official websites of public platforms, etc. Therefore, we believe that this attack is a DDoS behavior with apparent benefits. A typical example is that we captured 320,000 visits redirected to Yandex on June 21, 2021. In the same year, the attack was successively exposed [24], which verified our findings impliedly. **Finding 4: IoT scanning behaviors are often DDoS-driven. By tunneling TCP, IoT botnet organizers expand DDoS attacks and avoid traceability.**

After the bot logs into the honeypot, only 2.9% of them reach the command execution stage for further control. Among these executed commands, after excluding more than 87% of repetitive command sequences and invalid commands containing only "shell" or "/bin/busybox", the length of 98% remaining commands are in $[7, 15]$.

Besides, there are mainly two kinds of malware-related attacks. The first kind is to download samples directly from the C&C host using the *wget* command. Most C&C servers are provided by well-known suppliers, such as AWS or Vultr. The second is the fileless attack. Attackers send the base64 source code of malware to victims and then decrypt it before execution. Fig. 6 is an example of this attack. **Finding 5: With 15 Linux-based commands, most malicious attempts are implemented as automated scripts.**

The COVID-19 began to spread in December 2019, coinciding with this paper's observation period. In Fig. 7, we count the COVID-19 case number and our captured attack activities over time. A hacker tries to monitor (and capture) all the system's network traffic during an attack analysis. The hacker can gather more information about the network by analyzing that traffic. In other words, breaching users' systems or cracking passwords is not a primary objective for a hacker.

Obviously, with the two epidemic outbreaks in 2020 December and 2021 March, the number of attacks reached a peak almost simultaneously. These statistical results imply that the epidemic outbreak suddenly increased attack surfaces on government services, online education, online shopping, and healthcare as the global workforce transitioned to a largely remote operations model, botnet organizers. Not only did the number of attacks increase, but so did their speed and scale. As the global workforce transitioned to a largely remote operations model, attackers get more opportunities to invade. **Finding 6: A positive correlation exists between IoT attacks and the trend of COVID-19.**



```
cd /var/tmp; echo
"IyEvYmluL2Jhc2gKY2QgL3RtcApybSAtcmYgLlgxN
S11bml4Cm1rZGlyIC5YMTUtdW5peApjZCAuWDE1LXV
uaXgKcGtpbGwgLTkgY3JvbiA+IC5vdXQKd2dldCAtc
SBodHRwOi8vNTQuMzcuNzAuMjQ5L2RvdGEyLnRhci5
neiB8fCBjdXJsIC1PIC1mIGh0dHA6Ly81NC4zNy43M
C4yNDkvZG90YTIudGFyLmd6CnRhciB4ZiBkb3RhMi5
0YXIuZ3oKI3JtIC1yZiBkb3RhMi50YXIuZ3oKY2QgL
nJzeW5jCmNobW9kIDc3NyAqCmNhdCAvdG1wLy5YMTU
tdW5peC8ucnN5bmMvaW5pdGFsbCB8IGJhc2ggMj4xJ
gpleGl0IDAK" | base64 --decode | bash
```

```
#!/bin/bash
cd /tmp
rm -rf .X15-unix
mkdir .X15-unix
cd .X15-unix
pkill -9 cron > .out
wget -q http://XXX/dota2.tar.gz || curl -O -f
http://XXX/dota2.tar.gz
tar xf dota2.tar.gz
#rm -rf dota2.tar.gz
cd .rsync
chmod 777 *
cat /tmp/.X15-unix/.rsync/initall | bash 2>1&
exit 0
```

Fig. 6. An example of fileless attack.

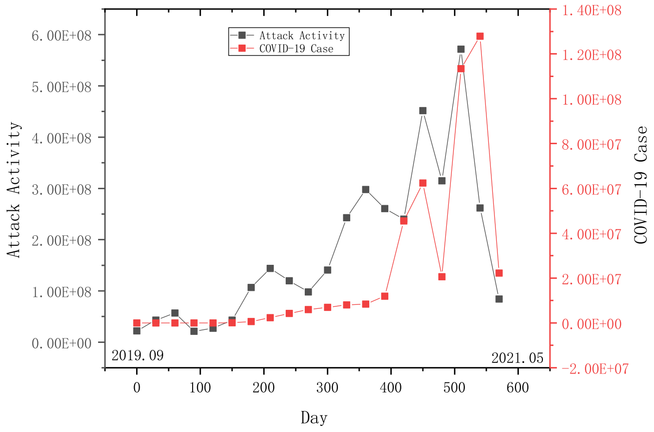


Fig. 7. Comparison of trends in capturing attacks and COVID-19.

The captured data includes 2, 920, 862 different bot IPs, of which 66% accessed more than one honeypot. The most frequent bot IP is captured by 104 honeypots in 5,295,301 sessions throughout the observation period. Therefore, without considering the control period of different attackers, the botnet analysis will bring misleading results to researchers. Note that we treat bots controlled by different botnets as different bots. **Finding 7: The compromised IoT devices will be controlled in succession or simultaneously by multiple botnets.**

5.2. Evaluation of botnet inference

To evaluate our method and address the Q2 raised in Section 1.2 by the following perspectives with three experiments.
RQ 1: *How effective is our proposed attack event simplification algorithm?*
RQ 2: *How does our performance compare with traditional methods?*
RQ 3: *Are there any representative random variables that can verify the effectiveness of our botnet analysis result?*

Experiment 1 (*RQ 1*). *The evaluation criteria in this set of experiments is the simplification rate r_T defined in Eq. (6).*

$$r_T = \frac{\sum_C^{n=1} c_i}{N} \tag{6}$$

where N donates the total session number, T donates the observation period (day), C donates the cluster set of the algorithm result, and c_i donates the removed session number of the ith cluster. Note that we manually verify that the sessions contained in c_i are indeed duplicates of data within a short period of time. Our honeypot distribution is relatively geographical dispersive. Therefore, the IPs captured by more than one honeypot tend to be controlled by multiple botnets. Besides, analyzing large-scale data has a tremendous computational burden on the server. Therefore, we filter out IPs captured by only one honeypot whose cumulative attack number is less than 5000. In this way, we obtain 200, 094, 875 *attack sessions that meet the above conditions to evaluate our algorithms.*

We use Algorithm 1 to filter repetitive attacks within a short period. Note that this procedure cannot be simply achieved by de-duplication because our goal is to remove the extra repetitive behaviors of the attacker in a short period to ensure a successful hit. Therefore, to avoid undermining the integrity of attack behaviors, we treat those not repeated in a short period as different

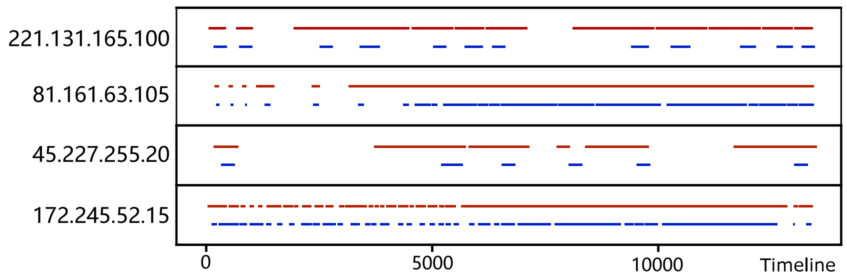
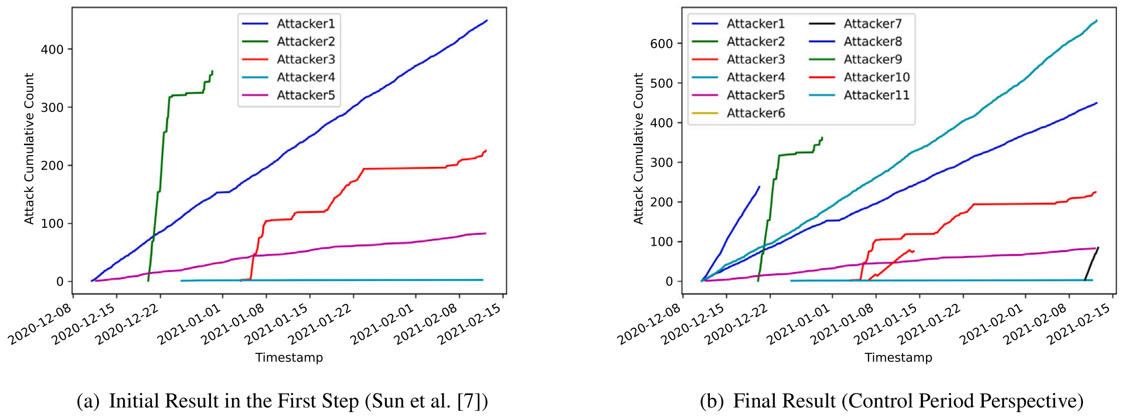


Fig. 8. Comparison of four experimental examples of algorithm 1. The red bar is before simplification processing, and the blue side is after processing.



(a) Initial Result in the First Step (Sun et al. [7])

(b) Final Result (Control Period Perspective)

Fig. 9. Comparison between attack cumulative counts of clusters in the initial step and the final result.

attacks. After simplifying the records, we removed 87% of repetitive attack sessions. Furthermore, manual verification shows that r_{100} reaches 97%. Manual validation shows that the filtered records are identical password attempts or port scans within 3 s. Similar behaviors separated by a longer interval (such as identical password attempts over 10 min) are preserved. We randomly select the attack sequence of 4 IPs (partial) and present the experimental results in Fig. 8.

Answer: *Experimental results show that the algorithm reduces the noise data by 87%, without destroying the original attack patterns.*

Experiment 2 (*RQ2*)**.** *How does our performance compare with traditional methods? In this experiment, we evaluate in two ways: (1) by intuitively observing the distribution of attack counts of different botnets in the results. (2) by evaluating the classification accuracy of simulated experiments in a controlled environment to verify the unsupervised real-world scenarios clustering effectiveness.*

After data simplification, we conduct our proposed algorithm 2 on the well-organized dataset. Due to limited computing resources, we verify the effectiveness of our algorithm by randomly selecting data from a honeypot. The final cluster result is the botnet referring result, consisting of bot fragments.

In the first step of our algorithm, we initially divided 128,737 attack sessions into 47 clusters through the Hawkes model. Since we select representatives of each cluster to represent the attack pattern of each botnet, the division of the control period of the representative bot can offer a significant guide for other bots in the cluster. After 11 iterations of segmentation, we successfully clustered these fragments into 121 clusters. Statistically, 89% of bots exist in different botnets. 74% bots are divided into less than 10 temporal fragments by different control periods. A Dutch bot (5.188.86.165) has 21 temporal fragments, which is the most. Fig. 9 shows the comparison of five examples' cumulative counts. The figure shows that our method presents botnet inference based on the existing cluster results of the traditional method. The other cluster botnets exhibit significantly different cumulative count growth rates.

Since the captured attack data is unlabeled, we design a set of controllable experiments to verify the effectiveness of our unsupervised learning method. According to the distribution of captured records, we design the following five random variables to simulate the uncertainty in the real-world attack behaviors of different botnet volumes for 30 days: (1) DCnt: attacks number launched per day; (2) DuAt: duration of each attack (minute); (3) VstFre: number of visits in a second per attack (every hundred times) (4) CelPro: the cancel probability of each visit; (5) Delay: delay time per visit (seconds). The attributes of the five simulated botnets are shown in Table 3. Note that the value range can be changed according to different user needs.

We assume that the control periods of different botnets are not overlapped in this experiment. Therefore, we randomly remove the overlap activities and initialize the starting timestamp to Unix time. Fig. 10 shows five simulated botnets' activities on timeline.

Table 3
Simulated botnet properties.

	Scale	DCnt	DuAt	VstFre	CelPro	Delay
A	1000	[24, 48]	[30, 60]	[10, 90]	0.1	[1, 2]
B	1000	[4, 8]	[10, 60]	[1, 3]	0.6	[1, 10]
C	100	[0, 2]	[30, 60]	[5, 10]	0.3	[0, 1]
D	10	[0, 2]	[30, 60]	[5, 10]	0.3	[0, 1]
E	10	[0, 1]	[30, 60]	[5, 10]	0.3	[0, 1]

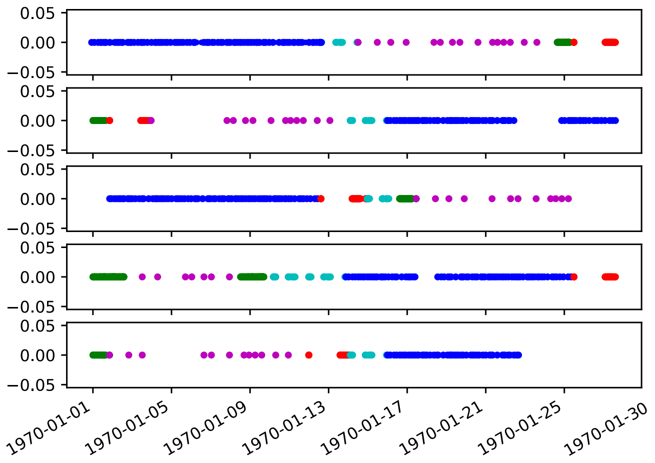


Fig. 10. Simulated botnet activities.

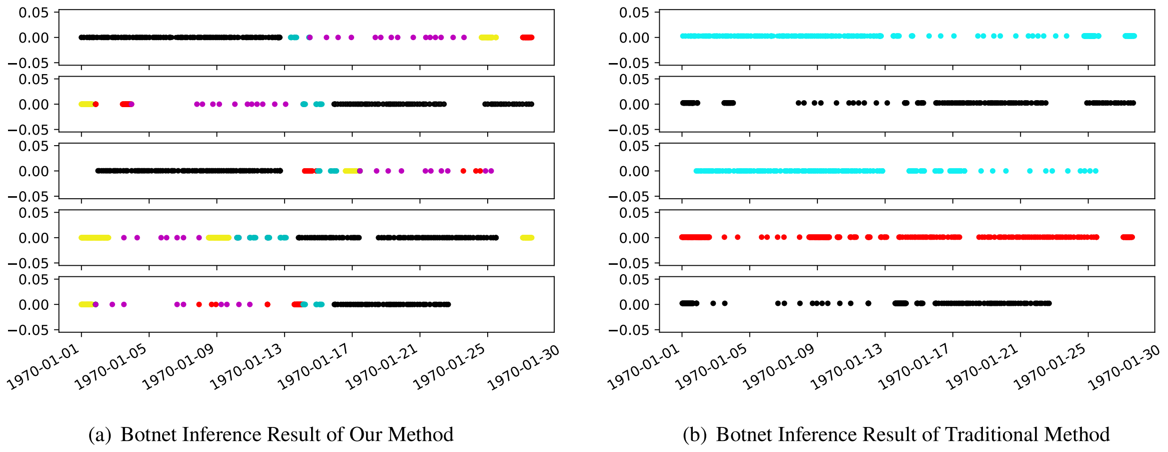
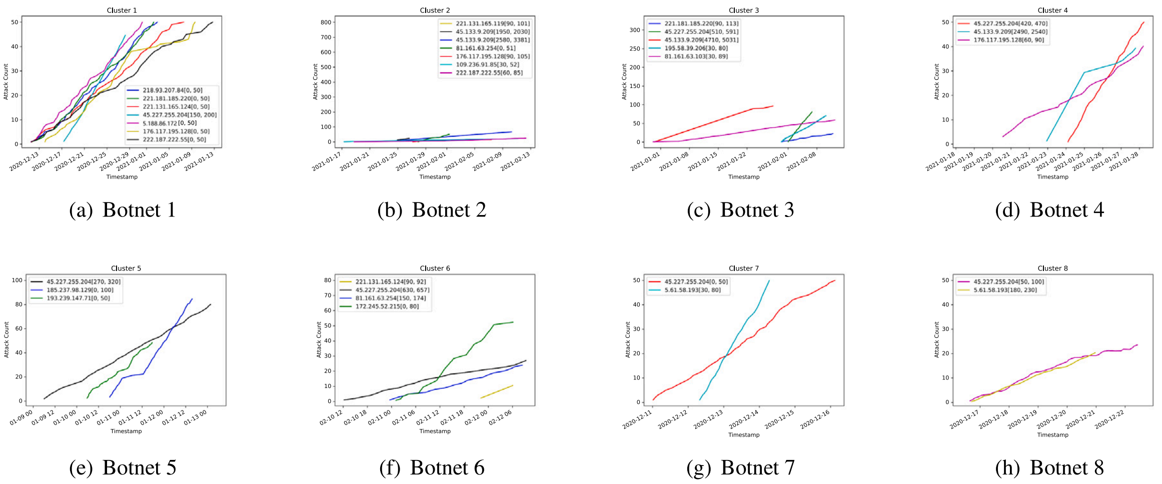


Fig. 11. Comparison result in controlled experiments. Different colors indicate different botnets.

In this way, all attack records in this controlled experiment are assigned known botnet labels. Experimental results show that our final clustering accuracy can reach 92%. As a comparison, the traditional method can only cluster the entire controlled host to get a macro-similar result without distinguishing different botnet control periods. Therefore, their description of the botnet attack patterns is inaccurate. The cluster results shown in Fig. 11 demonstrate that our algorithm can distinguish most control periods.
Answer: *Comparison result shows that the traditional work can effectively distinguish macroscopically attack patterns. However, our algorithm presents a more fine-grained inference from the perspective of the control period. Due to the objective of removing the attacker's extra repetitive behavior in a short amount of time, this procedure cannot be accomplished through de-duplication scripts. When a behavior does not repeat in a short period, it is treated as a different attack. Furthermore, Attacks increased in both number and speed, as well as in scale. As most of the global workforce is now conducted remotely, attackers have more opportunities to infiltrate.*

Experiment 3 (*RQ3*). *Are there any random variables that can prove the effectiveness of our botnet analysis result?*

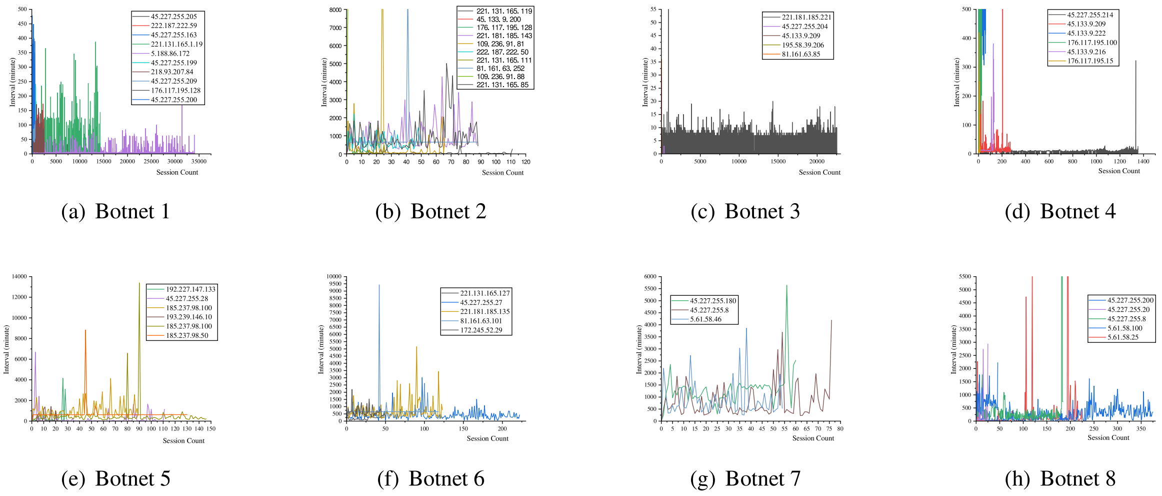
In this set of experiments, we measure the clustering effect by comparing the patterns of several variables in different clusters. These variables can be used as essential guidelines for botnet identification because they meet the following two conditions:



(a) Botnet 1 (b) Botnet 2 (c) Botnet 3 (d) Botnet 4

(e) Botnet 5 (f) Botnet 6 (g) Botnet 7 (h) Botnet 8

Fig. 12. The cumulative attacks number distribution of botnet clusters (Partial).



(a) Botnet 1 (b) Botnet 2 (c) Botnet 3 (d) Botnet 4

(e) Botnet 5 (f) Botnet 6 (g) Botnet 7 (h) Botnet 8

Fig. 13. Session interval of bot (Partial).

1. The same random variable results in different distributions across two different botnets;
2. The same random variable results in highly similar distributions across two different hosts employed by the same botnet.

We observe numerous bots on the same network segment in each cluster, and the attacks of these bots are almost all repetitive behaviors. We randomly select one bot in each network segment to present in the following figures to show the results clearly. We enumerate the distribution of outcomes for the following four variables:

Cumulative number of attacks is used to represent the total number of a bot attacking a honeypot in a period. Fig. 12 shows the cumulative attack number of eight clusters.

The slopes of monotonically increasing broken lines represent attack frequency changes over a while. The experimental results show that different clusters have intuitively distinct distribution patterns. The attack frequency change rate of bots in the same cluster is the same in the same period. Besides, different control periods for the same IP also show significantly different patterns. For example, the attack frequency of *45.227.255.204* when controlled by Botnet 8 12(h) is slower than when it is controlled by Botnet 7 12(g).

Session Interval of Bot indicates the compactness of the bot's attack in the same botnet, which is defined as follows: (1) assume Session A and Session B on host H belong to the same botnet. (2) Session A ends at time t_1, while session B starts at time t_2. (3) $t_2 - t_1$ is a random variable, defined as the Session Interval of Bot. Fig. 13 shows the distribution of Session Interval of Bot of eight clusters.

The experimental results show that the period for bots in the same network segment to be occupied is consistent, implying that the attacker uses similar methods to control after invading the devices under the unified LAN. Different network conditions result in different distribution patterns. Besides, different controlled periods of one bot show different variable distributions.

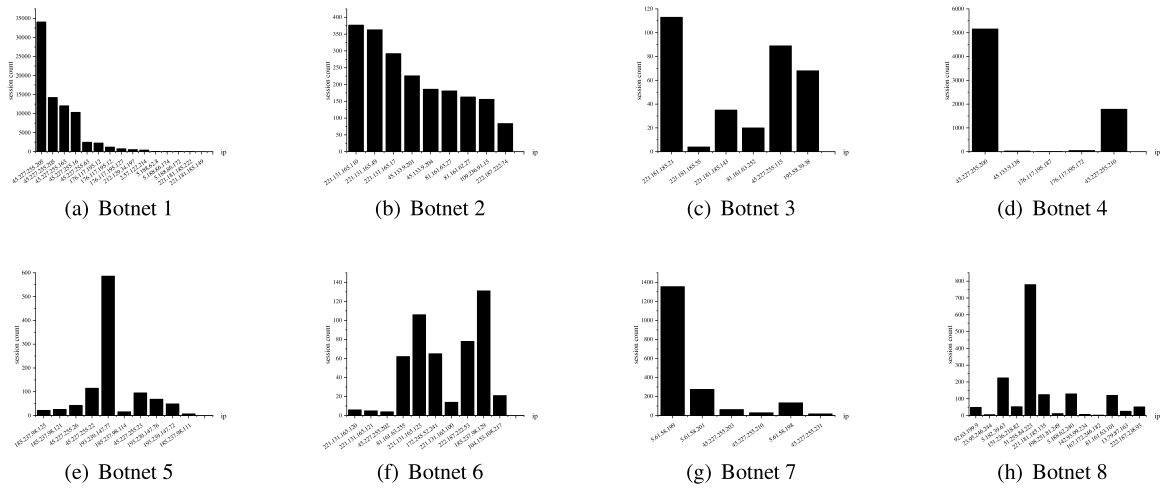


Fig. 14. Session count of bot (Partial).

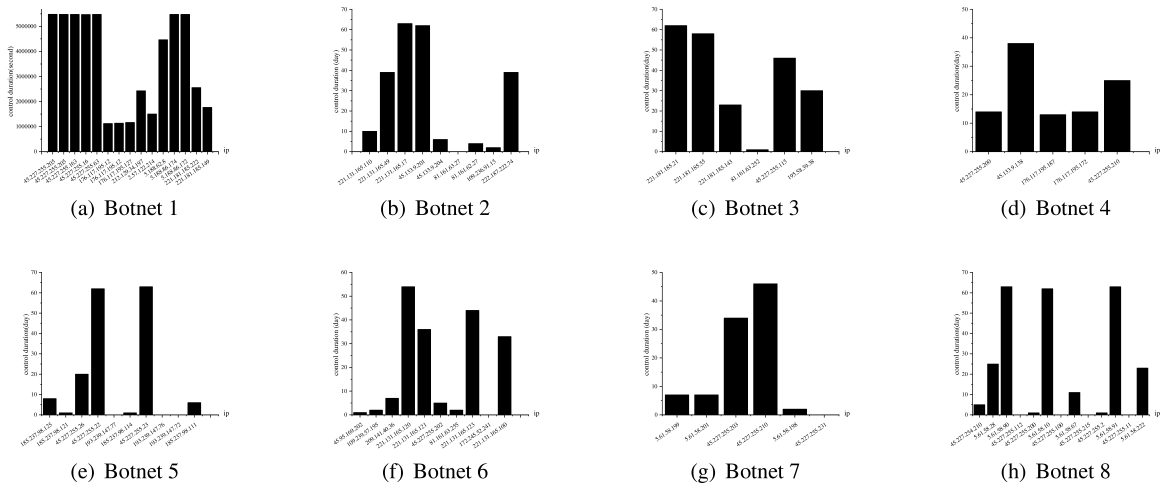


Fig. 15. Control duration of bot (Partial).

Session Count expresses the adequacy of the botnet's utilization of bots. We can intuitively believe that bots with more sessions consist of more behaviors. Fig. 14 shows the distribution of the Session Count of Bots.

Experimental results show that some attackers launch more sessions to achieve their goals, such as botnet 2 in Fig. 14(b). The distribution of this variable in each network segment still shows apparent regularity, implying that the attacker's intrusion strategies tend to traverse the available devices under the entire LAN.

Control Duration of Bot represents the control time of each bot under their botnets. Since our observation period is long enough (22 months), our experimental results can be considered to cover most of the botnet's life cycle. The results show that most botnets control bots for 1–70 days.

The results shown in Fig. 15 indicate a gap in the ability of different botnets to control bots. As mentioned earlier, botnet organizers have competitive relationships in the face of attractive and fragile devices. Therefore, this variable can be an essential guide for botnet tracking and classification.

Answer: *Clustering results show that the same variable (Cumulative number of attacks, Session Interval of Bot, and Control Duration of Bot) results in different distributions across two different botnets. Furthermore, the same variable results in highly similar distributions across two different bots controlled by the same botnet.*

Finding 8: IoT botnet attacks toward different LANs present different patterns.

6. Conclusion and future work

IoT botnet attacks are frequently launched in the real world. With the outbreak of COVID-19, attackers tend to organize different botnets that target the same victim host competitively or cooperatively. Furthermore, monitoring and analyzing large and complicated networks is still challenging. This study presents a data-driven study to analyze IoT botnet and attack patterns. To this end, we first deploy 462 honeypots and simplify the burden of large-scale data processing. Besides, we propose a recursion-based novel algorithm, which identifies botnets based on control periods. Experimental results show that our method outperforms the traditional method to infer the control period based on the temporal features. Furthermore, we reveal eight exciting findings that can provide a critical guideline for botnet analysis. The limitation of this study is that the overlapping control period might affect the accuracy. Therefore, fine-grained solutions should be designed to identify the attacks in an unsynchronous manner in future work. Our data-driven research mainly focuses on producing new findings based on a control period perspective rather than a real-time system. Therefore, our next step will consider system throughput, utilization, overhead, and sensitive analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was partly supported by the Key-Area Research and Development Program of Guangdong Province, China 2020B0101360001, the National Key Research and Development Program of China under Grant 2020YFB1406902, the Shenzhen Science and Technology Research and Development Foundation, China No. JCYJ20190806143418198, the National Natural Science Foundation of China (NSFC) (61872110), the Fundamental Research Funds for the Central Universities, China (Grant No. HIT.OCEF.2021007), and the Peng Cheng Laboratory Project, China (PCL2021A02). (Corresponding author: Hui He.)

References

[1] Al-Hamar Yuosuf, Kolivand Hoshang, Tajdini Mostafa, Saba Tanzila, Ramachandran Varatharajan. Enterprise credential spear-phishing attack detection. Comput Electr Eng 2021;94:107363.
[2] Zhou Wei, Jia Yan, Peng Anni, Zhang Yuqing, Liu Peng. The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. IEEE Internet Things J 2018;6(2):1606–16.
[3] Spamhaus Malware Team. Spamhaus botnet threat update: Q1-2021, https://www.spamhaus.org/news/article/809/spamhaus-botnet-threat-update-q1-2021.
[4] Feily Maryam, Shahrestani Alireza, Ramadass Sureswaran. A survey of botnet and botnet detection. In: 2009 third international conference on emerging security information, systems and technologies. IEEE; 2009, p. 268–73.
[5] Higuera Juan Ramón Bermejo, Higuera Javier Bermejo, García Juan Luis Tébar, Montalvo Juan Antonio Sicilia, Rubio Manuel Sánchez. Building a dataset through attack pattern modeling and analysis system. Comput Electr Eng 2022;97:107614.
[6] Li Ke, Liu Chaoge, Cui Xiang. Poster: A lightweight unknown http botnets detecting and characterizing system. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. 2014. p. 1454–6.
[7] Sun Peiyuan, Li Jianxin, Bhuiyan Md Zakirul Alam, Wang Lihong, Li Bo. Modeling and clustering attacker activities in IoT through machine learning techniques. Inform Sci 2019;479:456–71.
[8] Cheng Hongbing, Regedzai Gloria Rumbidzai, et al. A survey on botnet attacks. Am Sci Res J Eng Technol Sci 2021;77(1):76–89.
[9] Linderman Scott, Adams Ryan. Discovering latent network structure in point process data. In: International conference on machine learning. PMLR; 2014, p. 1413–21.
[10] Lackner Paul. How to mock a bear: Honeypot, honeynet, honeywall & honeytoken: A survey. In: Filipe Joaquim, Smialek Michal, Brodsky Alexander, Hammoudi Slimane, editors. Proceedings of the 23rd international conference on enterprise information systems, ICEIS 2021, online streaming, April 26-28, 2021, Vol. 2. SCITEPRESS; 2021, p. 181–8.
[11] Rahmatullah Dandy Kalma, Nasution Suiya Michrandi, Azmi Fairuz. Implementation of low interaction web server honeypot using cubieboard. In: 2016 international conference on control, electronics, renewable energy and communications. IEEE; 2016, p. 127–31.
[12] Scholten CPB. Automatic detection of zero-day attacks in high-interaction IoT honeypots using static analysis techniques (Masters thesis), University of Twente; 2021.
[13] Fraunholz Daniel, Krohmer Daniel, Anton Simon Duque, Schotten Hans Dieter. Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot. In: 2017 international conference on cyber security and protection of digital services. IEEE; 2017, p. 1–7.
[14] Roesch Martin, et al. Snort: Lightweight intrusion detection for networks. In: Lisa, vol. 99, 1999, p. 229–38.
[15] García Sebastián, Zunino Alejandro, Campo Marcelo. Survey on network-based botnet detection methods. Secur Commun Netw 2014;7(5):878–903.
[16] Singh Kamaldeep, Guntuku Sharath Chandra, Thakur Abhishek, Hota Chittaranjan. Big data analytics framework for peer-to-peer botnet detection using random forests. Inform Sci 2014;278:488–97.
[17] Yan Qiben, Zheng Yao, Jiang Tingting, Lou Wenjing, Hou Y Thomas. Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis. In: 2015 IEEE conference on computer communications. IEEE; 2015, p. 316–24.
[18] Amal MR, Venkadesh P. Review of cyber attack detection: Honeypot system. Webology 2022;19(1).
[19] Matin Iik Muhamad Malik, Rahardjo Budi. The use of honeypot in machine learning based on malware detection: A review. In: 2020 8th international conference on cyber and it service management. IEEE; 2020, p. 1–6.
[20] Jhaveri Rutvij H, Patel Narendra M, Zhong Yubin, Sangaiah Arun Kumar. Sensitivity analysis of an attack-pattern discovery based trusted routing scheme for mobile ad-hoc networks in industrial IoT. IEEE Access 2018;6:20085–103.
[21] Bar Ariel, Shapira Bracha, Rokach Lior, Unger Moshe. Identifying attack propagation patterns in honeypots using Markov chains modeling and complex networks analysis. In: 2016 IEEE international conference on software science, technology and engineering. IEEE; 2016, p. 28–36.

[22] Javadpour Amir, Wang Guojun, Rezaei Samira, Chend Shuhong. Power curtailment in cloud environment utilising load balancing machine allocation. In: 2018 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE; 2018, p. 1364–70.
[23] wikipedia. Longest common substring problem. 2021, https://en.wikipedia.org/wiki/Longest_common_substring_problem.
[24] BarakAdama. Mirai botnet: Investigating the largest ddos attack in the history of the internet. 2021, https://habr.com/ru/company/yandex/blog/577026/.

Huanran Wang received the BS degree in software engineering from Harbin Engineering University, China, in 2016. He is working toward the Ph.D. degree in the School of Computer Science and Technology, Harbin Institute of Technology. His research interests include Mobile Security, secure Internet of Things, and Cyberspace Security.

Hui He is supervisor in the School of Computer Science and Technology,Harbin Institute of Technology. She is a member of the IEEE, ACM and CCF. She conducts research in network and information technology, big data processing and analysis. She has published more than eighty scientific papers. She has accomplished many projects such as National High Technology Research.

Weizhe Zhang is currently a professor in the School of Computer Science and Technology at Harbin Institute of Technology, Harbin, China, and director in the Cyberspace Security Research Center, Pengcheng Laboratory, Shenzhen, China. He received his B.Eng, M.Eng and Ph.D. degree of Engineering in computer science and technology in 1999, 2001, and 2006 respectively from Harbin Institute of Technology.

Wenmao Liu is the director of Innovation Center, and also the leader of XingYun Lab of NSFOCUS Inc., Co-Chair of Cloud Security Service WG, CSA. He received his Ph.D. in Information Security from the Harbin Institute of Technology in 2013. After completion of his degree, Dr. Liu served as a researcher at NSFOCUS Inc.

Peng Liu received his Ph.D. from George Mason University in 1999. Dr. Liu is the Raymond G. Tronzo, MD Professor of Cybersecurity, founding Director of the Center for Cyber-Security, Information Privacy, and Trust at Penn State University. His research interests are in all areas of computer security. He has published over 300 technical papers.

Amir Javadpour obtained his M.Sc. degree in Medical Information Technology Engineering from University of Tehran, Iran, in 2014. From Guangzhou University, China, he received a Ph.D. in Computer Science/Mathematics/Cybersecurity. In addition, he has published papers with his colleagues in highly ranked journals and several ranked conferences on several topics, including Cloud Computing, Big Data, and the Internet of Things.