Landing Reinforcement Learning onto Smart Scanning of The Internet of Things

Jian Qu[†§], Xiaobo Ma[†§*], Wenmao Liu[‡], Hongqing Sang[‡], Jianfeng Li[‡‡], Lei Xue[‡‡], Xiapu Luo[‡‡], Zhenhua Li[‖], Li Feng[††], Xiaohong Guan[†§]
[†]MOE Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China
[§]Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China
[‡]NSFOCUS Inc., China
[‡‡]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[‖]School of Software, Tsinghua University, Beijing, China
[††]Center of Dependable and Secure Computing (CDSC) of Wuhan Digital Engineering Institute (WDEI), Wuhan 430074, China

***Abstract*—Cyber search engines, such as Shodan and Censys, have gained popularity due to their strong capability of indexing the Internet of Things (IoT). They actively scan and fingerprint IoT devices for unearthing IP-device mapping. Because of the large address space of the Internet and the mapping's mutative nature, efficiently tracking the evolution of IP-device mapping with a limited budget of scans is essential for building timely cyber search engines. An intuitive solution is to use reinforcement learning to schedule more scans to networks with high churn rates of IP-device mapping. However, such an intuitive solution has never been systematically studied. In this paper, we take the first step toward demystifying this problem based on our experiences in maintaining a global IoT scanning platform. Inspired by the measurement study of large-scale real-world IoT scan records, we land reinforcement learning onto a system capable of smartly scanning IoT devices in a principled way. We disclose key parameters affecting the effectiveness of different scanning strategies, and find that our system would achieve growing advantages with the proliferation of IoT devices.**

I. Introduction

In recent years, cyber search engines, such as Shodan [1], Censys [2], [3], and ZoomEye [4], have gained popularity among the security community due to their strong capability of indexing the Internet of Things (IoT) like webcams and routers. They actively scan IoT devices with fingerprints of various devices for unearthing IP-device mapping, offering publicly available search engine services. One can simply access these services using a browser, and obtain IP-device mapping results by host names, IP addresses, certificates, or device-specific keywords. These search engines can also be used to effectively find IoT devices with certain (possible) vulnerabilities on the Internet [5]–[7]. In the short term, they render large-scale attacks against IoT devices easier, while simultaneously offering a public channel for devices' owners (especially those with higher security requirements) to be informed of the exposure. More importantly, in the long run, they would force IoT device manufacturers into making the best efforts to improve the security of their devices [8], [9].

*Corresponding author. Email: xma.cs@xjtu.edu.cn

Because of the large address space of the Internet and the mutative nature of IP-device mappings, efficiently tracking the evolution of IP-device mapping with a limited budget of scans is essential for building timely cyber search engines. High-rate scanning, in spite of the timeliness, usually induces excessive noises and thus may be blocked by firewalls. On the contrary, low-rate scanning is not noisy, but not timely. As a result, the data obtained from cyber search engines would be generated a long time ago, rather than the up-to-date IoT devices on the Internet that are far more valuable.

Scanning IoT devices in a principled way so to meet the timeliness requirement with a limited budget of scans depends on two major aspects. One is the resource investment, and the other is the scanning strategy. The former includes the number of servers, the servers' processing power, the bandwidth, etc. The latter concerns how to schedule scans encapsulating fingerprints of IoT devices across all IP addresses from the temporal perspective. Since the resource investment for cyber search engines is relatively stable, we only focus on the latter aspect. Existing cyber search engines, such as Censys, divide the protocol into different categories, and for each category the scanning frequency is manually defined [3]. For example, Censys scans HTTP daily and SSH biweekly. Apparently, manually scheduling scans cannot maximize the timeliness of cyber search engines because it does not consider IP-device mapping dynamics in different networks.

Despite its conceptual simplicity, how to design a system capable of smartly scheduling scans for IoT devices has never been systematically investigated. An intuitive solution is to use reinforcement learning to flexibly schedule more scans to networks with high churn rates of IP-device mapping, as can be learned from historical scanning records. However, there are two major challenges to apply reinforcement learning into designing the system. First, an immediate challenge is that designing such a system necessitates large-scale real-world IoT scanning records so to have insights of IP-device mapping dynamics all over the Internet. However, there is no publicly available data for gaining insights and facilitating the design.

Second, the IP-device mapping dynamics are driven by hidden factors that are hard to infer (e.g., which networks have the same IP assignment policy and can be characterized by similar mapping dynamics). This hurdle is further compounded by the Internet's tremendous and time-evolving nature, making it challenging to design the system.

To address these challenges, we carry out both measurement study and system design for smartly scheduling scans for IoT devices. First, we perform large-scale measurements of IP-device mapping dynamics using our global IoT scanning platform. The measurement enables us to collect real-world IoT scanning records, quantify the IP-device mapping dynamics, and analyze factors affecting the dynamics. Second, inspired by the measurement study, we design a novel system that lands reinforcement learning onto guiding the smart scanning by exploiting the observation that the IP-device mapping dynamics in different networks may vary significantly.

Our system makes automatically learning IP-device mapping dynamics across different networks as a built-in feature for continuous scanning decision making, enabling the encouragement of scans to networks with high churn rates of IP-device mapping dynamic mapping and the discouragement of scans to those with low churn rates. To our best knowledge, we are the *first* to explore principled ways to scan IoT devices based on real-world measurement study. Our major contributions are summarized as follows.

- We perform measurements based on large-scale real-world IoT scanning records (consisting of 5,241,566 IP cameras) by scanning the entire IPv4 space for about 40 days, and quantify the IP-device mapping dynamics. The results reveal that both the IoT device types and IP address pools affect the dynamics.
- We land reinforcement learning onto a system capable of smartly scanning IoT devices. The system can encourage scans to networks with more dynamic IP-device mapping while impeding scans to those with less dynamic mapping. It consists of two novel strategies for scheduling scans based on online learning and batch learning. It could temporally schedule scans through continuous scanning decision making in consideration of historic IP-device mapping dynamics, as well as the hierarchically learned (spatial) IP address pools.
- Through extensive experiments, we demonstrate that our system could generally capture more IP-device mapping mutations than random and sequential scanning. We disclose the two key parameters affecting the effectiveness of different scanning strategies, i.e., the scan rate and the proportion of IoT devices to IP addresses. We also find that, as the number of IoT devices on the Internet grows, our system would grow far more advantageous than random and sequential scanning.

Roadmap. Sec. II performs measurement study using real-world data. Sec. III introduces system overview, Sec. IV details system design, and Sec. V conducts the evaluation. Finally, we survey the literature in Sec. VI, and conclude in Sec. VII.

II. Understanding IP-device Mapping Dynamics

A. Background

There are usually three ways to configure the IP address for a device, namely, Dynamic Host Configuration Protocol (DHCP), Point-to-Point Protocol (PPP), and static IP configuration [10]–[12]. Both DHCP and PPP will cause IP address changes frequently.

The DHCP server controls a pool of IP addresses. A client, when connecting to the network, can be automatically assigned an IP address from the pool by the DHCP server. The client can keep the IP address within the lease duration (configured by the network manager). Upon the lease duration expires, the client can send a message to the DHCP server to extend its lease for the same IP address [10]; if the client does nothing, the address will be revoked.

PPP can be encapsulated in data link layer protocols like PPP over Ethernet (PPPoE) and PPP over Asynchronous Transfer Mode (PPPoA). PPP first establishes a session between the client and the server. Then, the Internet Protocol Control Protocol (IPCP) is used to configure the client device's IP address. IPCP does not have a lease duration. The IP address is released when the PPP session ends [11]–[13].

In both DHCP and PPP, if the session of the device continues, the IP address will not change. IP address mutations can be triggered by both the client and the server. On the client side, once the device re-establishes the PPP session, a new IP address will be assigned to the device; if a DHCP client is offline for a period of time that exceeds the lease duration and then reconnects the DHCP server, its IP address will change. On the server side, Padmanabhan et al. found some ISPs may limit the session duration (typically a multiple of 24 hours), and the IP address will change periodically [14].

B. Measuring IP-device Mapping Dynamics

To understand IP-device mapping dynamics, one needs to perform large-scale scans encapsulating fingerprints of IoT devices globally, and collect detailed scanning records. Despite the prevalence of IoT fingerprinting techniques [15]–[17], no public scanning record is available to facilitate the study.

To this end, we take the first step to measure real-world IP-device mapping dynamics by performing large-scale scanning campaigns using our globally deployed (commercial) IoT scanning platform. Specifically, we scanned the entire IPv4 space for identifying IP cameras, the most popular type of IoT devices on the Internet. The scanning campaign was repeated four times, starting on June 5, June 15, June 26, and July 6, 2021, respectively. At each time, the scanning campaign lasted about 10 days, resulting in 2,896,824, 3,089,436, 3,093,510, and 3,076,343 successful scanning records, respectively.

Note that the traffic fingerprints of these IP cameras are extracted from their banner text that commonly describe device types explicitly. We have been manually maintaining and labeling a database of IoT fingerprints with the aid of machine learning. For more details, the reader can refer to [18].

The four time scanning campaigns allow us to measure IP-device mapping mutations by comparing scanning records.

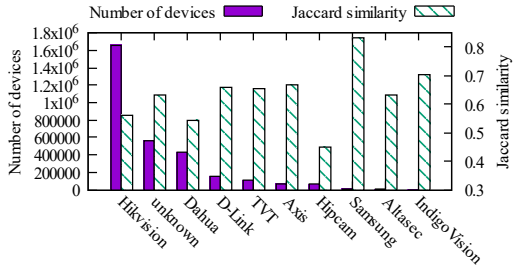


Fig. 1. The number of cameras from different manufacturers and Jaccard similarity between the two scans.

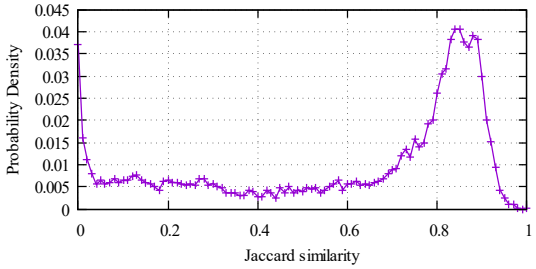


Fig. 2. The probability distribution of the Jaccard similarity between two scans across all IP address pools.

Suppose we perform a scanning campaign at time t_1 and the result is S_{t_1}, the set of (successfully scanned) mapping between IP addresses and device types. For example, $S_{t_1}[\alpha] =$ "D-Link Camera", where α is an IP address. S_{t_2} is the result of scanning campaign at time t_2. We employ Jaccard similarity to measure the mapping mutations between t_1 and t_2:

$$J(t_1, t_2) = \frac{|S_{t_1} \cap S_{t_2}|}{|S_{t_1} \cup S_{t_2}|}. \quad (1)$$

If there are no IP-device mapping mutations, $J(t_1, t_2)$ will equal 1. As the mutations become significant, $J(t_1, t_2)$ will approach 0. Note that, when $J(t_1, t_2)$ equals 1, we cannot be 100% sure that there are no mapping mutations, due to the possibility of an IP address being re-assigned to a device of the same type as the device that originally owns the IP address. However, in such a case, the probability of mapping mutations would be extremely small and ignorable.

1) Device Types: In our scanning campaigns, we successfully find a cumulative number of 5,241,566 IP cameras. Fig. 1 shows the average number of devices of the top 10 popular IP camera types across all campaigns. For a certain device type, we calculate the mapping mutations between every two successive campaigns using (1). Consequently, three values of Jaccard similarity measuring mapping mutations are derived, and we plot the average value in Fig. 1. It can be seen that the Jaccard similarity between two successive scanning campaigns differs significantly across IP camera types. For example, the Jaccard similarity of the Samsung camera is about 0.83, but that of the Hipcam camera is less than 0.45. This indicates that the IP-device mapping dynamics are device-type-specific.

2) IP Pools: We consider an IP address pool as a set of IP addresses (possibly) under the same IP management policy. Different IP address pools may have different IP-device mapping mutation intensities because each IP address pool has its configuration, capacity, and occupancy. Padmanabhan et al. found that the frequency of IP address changes was related to geographic location [14]. In other words, the IP-device mapping mutations are related to the IP address pools.

To gain (coarse-grained) insights into IP-device mapping mutations of large-sized IP pools, we define an IP address pool as a class B IP space, and the IPv4 space is divided into 65,536 IP address pools. For each pool, we derive (1) between every two adjacent scanning campaigns and calculate the average Jaccard similarity. In our calculation, we neglect the IP address pools with less than 50 identified IP cameras for statistical validity. Fig. 2 presents the probability distribution of the average Jaccard similarity across all IP address pools. Although there are two peaks when the Jaccard similarity approaches 0.85 or slightly exceeds 0, the overall distribution is dispersed. This implies that the IP-device mapping dynamics are also dependent on IP address pools.

III. System Overview

Since the IP-device mapping dynamics are related to device types and IP address pools, we exploit this observation to design a system capable of smartly scheduling scans for IoT devices. Fig. 3 shows the architecture of our proposed system.

First of all, a *priority matrix* describing the IP-device scanning tasks is fed into the system. Each element of this matrix is a priority value (i.e., 1, 2, 3, ...) specifying the order to execute the corresponding scanning task defined by (device type, IP address). Initially, the priority matrix is randomly or sequentially defined. As the scanning proceeds, we will collect more historical records consisting of (IP, device, last scan time). Then, we derive a *probability matrix* quantifying the probability of IP-device mutation. Each element of this matrix denotes the probability that the corresponding IP-device mapping mutations in the next scheduled scan. We keep updating the probability matrix as new scanning records arrive, while simultaneously returning probability ranking as feedback to refresh the priority matrix. Finally, the tasks with larger values of mutation probability would be assigned higher priorities in the task queue.

When we derive the probability matrix, not only do we use scanning records but also an *intensity matrix*. The rationale is that the IP-device mutation probability in the next scheduled scan depends on both temporal and spatial factors. The temporal factor is the time interval between the current time and the last scan time (in scanning records). As the time interval increases, the mutation probability grows because of occurrences of events such as device replacement, and IP reconfiguration. The spatial factor is the intensity matrix that characterizes the overall likelihood of IP-device mutation in individual IP address pools. Each individual pool is expected to be under the management of the same IP assignment policy, and hence devices in that range are statistically coherent in terms of IP-device mutation. IP-device mappings belonging to IP address pools with stronger mutation intensity tend to have higher mutation probability.

The intensity matrix is estimated using scanning records. Originally, each IP address pool is defined as the IP address

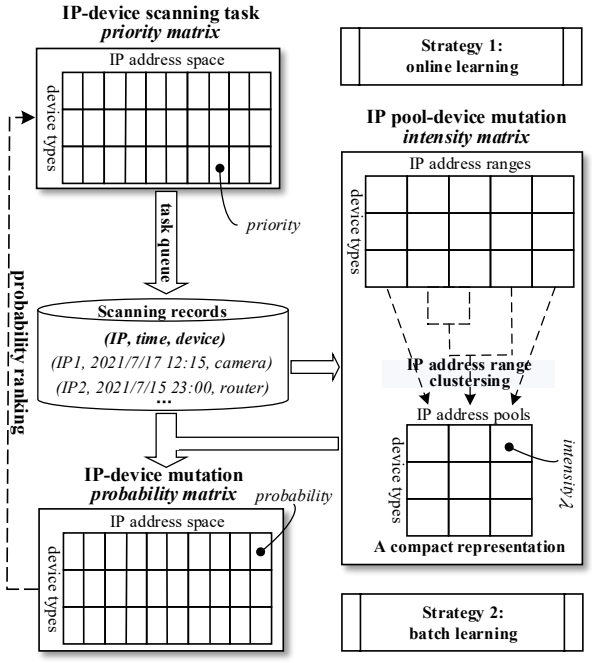


Fig. 3. The architecture of the proposed IoT scanning system.

range of a small network, say a class C network, to ensure its high probability under the management of the same IP assignment policy. As scanning records accumulate, our system will hierarchically cluster small IP address ranges into large IP address pools, resulting in a compact representation of the intensity matrix. The compact representation leads to a more computational effective estimation of the intensity matrix. More importantly, as the size of an IP address pool increases, the estimation accuracy for the intensity matrix increases due to the growing number of scanning records in that pool.

The above architecture of our system constitutes a continuous scanning decision making process. Under such an architecture, we propose two reinforcement learning-based scanning strategies, namely, online learning and batch learning, according to the way to build the intensity matrix. The online learning strategy incrementally updates the intensity matrix while performing scanning, and directly enters the continuous scanning decision making process. The batch learning strategy, however, proactively scans abundant information to build the intensity matrix before entering the continuous scanning decision making process.

IV. System Design

Following the proposed architecture, we detail our system design. First, we present the designing goal and system model. Then, we design online learning and batch learning scanning strategies, and the IP address pool estimation technique. Table I summarizes major notations in our system.

A. Designing Goal

We model the IoT scanning process as a continuous decision-making process. For a certain type of IoT device, the basic task of the decision-making is to select one IP address, scan it, and get the result. That is, the task can be regarded as a cycle of selection and scan. The cycle will be repeated continuously, hence constituting the entire scanning process.

From the perspective of IoT devices, our goal is to optimize the scanning strategy to capture as many IP address changes as possible. Such a goal is equivalent to capture as many IoT device changes as possible from the perspective of IP addresses. To sum up, we aim to capture as many IP-device mapping mutations as possible. Note that we treat an IP address without hosting any device as a special IP-device mapping. During the scanning process, we score the reward as 1 upon capturing a mutation. Given a fixed number of scans and a period of time, we aim to maximize the total reward.

TABLE I
Summary of major notations.

Notation	Definition
d	a device type
r	an IP address range
A	the set of scanning tasks
$\lhd(t_1,d_1),(t_2,d_2)\rhd$	a 2-gram scanning record
$R(d)$	all 2-gram scanning records with $d_1 = d$
S_t	the set of the latest scanning records at time t
$\pi(S_t)$	the next scheduled scanning tasks given S_t
λ	the IP pool-device mapping mutation intensity

B. System Model

1) Scanning Process Modeling: The scanning can be modeled as a process of continuously making decisions on the priorities of IP-device scanning tasks. All the scanning tasks are organized in the priority matrix. At a high level, the scanning process keeps refreshing the priority matrix regularly after executing a bunch of scanning tasks. The refreshing is actually to fine-tune the priority values based on the set of the latest scanning records of all IP addresses.

Let $S_t(t = t_0, t_1, t_2, \ldots)$ be the set of the latest scanning records of all IP addresses at time t, and A denote the set of scanning tasks. Then, the problem becomes fine-tuning the priority value for each scanning task in A based on S_t. Intuitively, scanning tasks with larger IP-device mutation probabilities (during the next scheduled scan) should be assigned higher priorities. More precisely, given S_t, top-k scanning tasks in terms of mutation probability ranking, denoted by $\pi(S_t)$, joins the queue of the next scheduled scan. $\pi(S_t)$ is formally expressed as

$$\pi(S_t) = \{a | \text{if } P(a|S_t) \in \text{top-k}(P(a|S_t)), a \in A\}, \quad (2)$$

where $P(a|S_t)$ is the IP-device mapping mutation probability when we choose a to scan at state S_t. Apparently, modeling IP-device mapping mutation and estimate the value of $P(a|S_t)$ is crucial in the scanning process.

We would like to point out that, in real-world scanning, scanning tasks associated with one IP address but multiple device types could be performed at the same time to improve the scanning efficiency, especially when the device types share the same port. For example, many types of commercial IP cameras open TCP port 554 by default. In this case, a scan

using a single TCP connection destined to TCP port 554 of the target IP address will identify the camera type.

2) IP-device Mapping Mutation Modeling: Consider an event as one IP-device mapping mutation. Naturally, for a certain device, the event arrivals can be modeled as a non-homogeneous Poisson process [19]. The reasons are twofold. First, the events of a device's IP address changes are generally independent of each other. For example, in the DHCP configuration mentioned in Sec. II-A, the IP address changes are independent since they are attributed to many random factors, such as device online-offline dynamics and human intervention; in the static IP configuration, the two typical events, namely, initial IP assignment and final IP release, are also independent. Second, in different periods of time, the rate of IP address changes is likely to be different. In other words, the rate of IP address changes is time-dependent. Formally, the event arrivals have the following properties.

$$\begin{cases} P[N(t+\Delta t)-N(t)=1]=\lambda(t)\Delta t+o(\Delta t), \\ P[N(t+\Delta t)-N(t)\geq 2]=o(\Delta t), \end{cases} \tag{3}$$

where $\lambda(t)$, a non-negative function of time t and device type d, denotes the mapping mutation intensity (i.e., the rate of IP address changes at t), and $N(t)$ represents the number of IP address changes during $(t, t+\Delta t]$, of a certain device.

Suppose t_1 is the last time to scan the IP address a, and the current time is t_2. At t_1, we find that a hosts a device d, which is recorded in the state s. Assume that the mapping mutation intensity of device d is $\lambda(t)$. Then, the probability of capturing the IP-device mapping mutation at t_2 is approximated as

$$P(a|S_t)=1-e^{\int_{t_1}^{t_2}\lambda(t)dt}. \tag{4}$$

The reason why it is approximate rather than strictly equal is as follows. After the device's IP address changes, other devices of the same type may be assigned the IP address again with a very low (almost ignorable) probability.

Recall that both the IoT device types and IP address pools affect the IP-device mapping dynamics, as is revealed in Sec. II. We maintain and estimate $\lambda(t)$ separately for each pair of device type and IP address pools in the intensity matrix demonstrated in Fig. 3.

3) Mapping Mutation Intensity Estimation: If mapping mutation intensity, i.e., $\lambda(t)$, can be estimated, the mutation probability $P(a|S_t)$ would be calculated. We can estimate $\lambda(t)$ using historic scanning records. Maximum likelihood estimation and least-squares methods can be used for parameter estimation. However, they are very difficult to solve.

Let us take maximum likelihood estimation as an example. In the simplest case, $\lambda(t)$ is a constant, which means λ is independent of time and only related to the device type. We define a *2-gram scanning record* as a (successive) subsequence of the scanning record sequence of an IP address. A 2-gram scanning record can be expressed as $<(t_1,d_1),(t_2,d_2)>$, where t_1 is a scanning time, t_2 is the scanning time following t_1, d_1 is the device type at t_1, and d_2 is the device type at t_2. The probability that a 2-gram scanning record occurs is

$$P(<(t_1,d_1),(t_2,d_2)>|\lambda)=\begin{cases} e^{-\lambda(t_2-t_1)} & \text{if } d_1{=}d_2, \\ 1{-}e^{-\lambda(t_2-t_1)} & \text{if } d_1{\neq}d_2. \end{cases} \tag{5}$$

Algorithm 1: Scanning using Online Learning Strategy

Input: scanning task set $A = \{\text{IP addresses}\} \times \{\text{device types}\}$
Output: scanning records
initialization:
$\lambda(d,r,t) = y(d,r,t) = n(d,r,t) = 0$;
$S_t(a) = (t_0, d_0)$ for each a;
while *True* **do**
  $\lambda(d,r,t) = \frac{y(d,r,t)+1}{y(d,r,t)+n(d,r,t)+1}$;
  scan $\pi(S_t)$ and get a set of 2-gram scanning records E;
  for *each* $[a, <(t_1,d_1),(t_2,d_2)>]$ *in* $[\pi(S_t), E]$ **do**
    if $d_1 \neq d_2$ **then**
      $y(d_1, r_a, t)$ += $\frac{|T(\lambda(d,r_a,t))\cap(t_1,t_2]|}{t_2-t_1}$;
    else
      $n(d_1, r_a, t)$ += $\frac{|T(\lambda(d,r_a,t))\cap(t_1,t_2]|}{|T(\lambda(d,r_a,t))|}$;
    end
    update $S_t(a) = (t_2, d_2)$;
  end
end

λ is the IP-device mapping mutation intensity of device type d_1. Let $R(d_1)$ denote the set of every 2-gram scanning record with the first device type equal to d_1 across all IP addresses. The likelihood function of d_1 changing to other device types across all IP addresses is

$$L(\lambda|R(d_1))=\prod_{r\in R(d_1)} P(r|\lambda). \tag{6}$$

Then, we maximize the likelihood function to estimate λ. However, the maximal value of this function is difficult to solve by calculating the zero point of the derivative, even if the log-likelihood function is used. To make things exacerbated, λ may not be a constant, and parameter estimation would be challenging. We need a solution to estimate λ.

Padmanabhan et al. revealed that the address changing intensity is dependent on the time during a 24-hour day [14]. Therefore, we model $\lambda(t)$ as a periodic function with a period of 24 hours for each device type. To estimate this continuous function, we can discretize it into small intervals. Specifically, we divide $\lambda(t)$ into 24 intervals, and the problem becomes estimating 24 discrete variables. We use $T(\lambda(t))$ to represent any one of the intervals, i.e., $[0,1)$, $[1,2)$,..., and $[23,24)$. We notice that the probability $\mathrm{P}(d_1 \neq d_2 | T(\lambda) \subseteq (t_1,t_2])$ grows as $\lambda(t)$ increases. We use this probability to approximate $\lambda(t)$.

Next, we design two strategies for estimating $\lambda(t)$, namely, the online learning strategy and the batch learning strategy.

C. Online Learning Strategy

The online learning strategy incrementally updates the estimation of $\lambda(t)$ as the scanning records arrive. The strategy is detailed in Algorithm 1.

In Algorithm 1, $\lambda(d,r,t)$ is represented by a three-dimensional array. The first dimension is the device type, the second dimension means the IP address range, and the third dimension refers to the 24 discrete time intervals. r_a represents the IP address range index of address a. $\lambda(d,r,t)$ equals $(y(d,r,t)+1)/(y(d,r,t)+n(d,r,t)+1)$ in the algorithm, where $y(d,r,t)$ and $n(d,r,t)$ represent the number of times

the scanning records change and does not change within the time period of $T(\lambda(d,r,t))$, respectively.

Specifically, for a record $< (t_1,d_1),(t_2,d_2) >$, if $d_1 \neq d_2$, $|T(\lambda(d,r_a,t)) \cap [t_1,t_2)|/(t_2-t_1)$ is added to $y(d,r_a,t)$, where $|T|$ refers to the time duration of T. This means that we split the contribution of one scanning record into multiple time intervals, since the mutation could happen at any time interval between t_1 and t_2. Similarly, $|T(\lambda(d,r_a,t)) \cap (t_1,t_2]|/|T(\lambda(d,r_a,t))|$ is added to $n(d,r_a,t)$ if $d_1 = d_2$, because no mutation between t_1 and t_2 in the scanning record indicates no mutation in all time intervals between t_1 and t_2.

The reason why we make $\lambda(d,r,t)$ equal to $(y(d,r,t)+1)/(y(d,r,t)+n(d,r,t)+1)$ instead of $y(d,r,t)/(y(d,r,t)+n(d,r,t))$ is to balance exploration and exploitation in online reinforcement learning [20]. Our solution is a combination of *upper confidence bound* and *Laplace smoothing*. In our solution, all the values of $\lambda(d,r,t)$ are equal to one at the beginning of the scan, thereby allowing us to randomly explore different IP addresses at the initial period. When the time tends to infinity, $\lambda(d,r,t)$ gradually stabilizes.

D. Batch Learning Strategy

In the online learning strategy, when the average scanning interval of each IP address exceeds 24 hours, the value of $\lambda(d,r,t)$ will be very similar, making segmentation of time meaningless. For example, suppose we have a 2-gram record $< (t_1,d_1),(t_2,d_2) >$, Algorithm 1 works well when t_2-t_1 is small. However, if t_2-t_1 surpasses 24 hours, the contribution of this 2-gram record to estimating $\lambda(t)$ will be limited.

To address the limitation of the online learning strategy, we propose the batch learning strategy in Algorithm 2. The basic idea is to conduct a special scan to collect information (i.e., batch learning), and then use the collected information to perform scanning (i.e., delayed scanning). That is, we efficiently and proactively collect needed information before entering a continuous decision-making scanning process.

In Algorithm 2, the entire scanning process is divided into two stages. Stage 1 performs batching learning. Specifically, we set a fixed interval for sequential scans to estimate $\lambda(t)$ efficiently. The estimation algorithm is the same as that in Algorithm 1, except that we calculate $\lambda(t)$ after Stage 1 (rather than at the time of every scan). Stage 2 conducts delayed scanning. We calculate $P(a|s)$ for each IP address and scan the IP address with the highest value of $P(a|s)$. Note that $\lambda(t)$ does not change any longer once Stage 2 starts.

The advantage of the batch learning scanning is the capability of accurately estimating $\lambda(t)$. Therefore, the final scanning performance could be improved, yet at the cost of extra scanning investment at the batching learning stage.

E. IP Pool Estimation Based on IP Address Range Clustering

After several rounds of scanning, the IP address range clustering algorithm can be used to estimate the IP address pools, resulting in a compact representation of the intensity matrix. During the scanning process, we gradually obtain statistics about IP-device mapping dynamics. Besides helping us perform scans more efficiently, such statistics allow us to estimate IP address pools. If some addresses have the same characteristics, they are likely to belong to the same IP address pool. The key to this problem is to find some features so that the features of the IP address ranges in the same pool are as similar as possible, and the features of the IP address ranges in different IP address pools differ from each other.

Algorithm 2: Scanning using Batch Learning Strategy

Input: scanning task set $A = \{\text{IP addresses}\} \times \{\text{device types}\}$
Output: scanning records
initialization:
$\lambda(d,r,t) = y(d,r,t) = n(d,r,t) = 0$;
$S_t(a) = (t_0,d_0)$ for each a;
—Stage 1: batch learning
while *True* **do**
  perform sequential scanning and get $< (t_1,d_1),(t_2,d_2) >$;
  if $d_1 \neq d_2$ **then**
    $y(d_1,r_a,t)$ += $\frac{|T(\lambda(d,r_a,t)) \cap (t_1,t_2]|}{t_2-t_1}$;
  else
    $n(d_1,r_a,t)$ += $\frac{|T(\lambda(d,r_a,t)) \cap (t_1,t_2]|}{|T(\lambda(d,r_a,t))|}$;
  end
  update $S_t(a) = (t_2,d_2)$;
end
$\lambda(d,r,t) = \frac{y(d,r,t)+1}{y(d,r,t)+n(d,r,t)+1}$;
—Stage 2: delayed scanning
while *True* **do**
  calculate $P(a|s)$ for each address a using (4);
  scan address a' that maximize $P(a'|s)$;
end

To exploit device distribution for estimating IP address pools, we employ a clustering-based method. Specifically, we use 256 IP addresses as the smallest unit of IP address ranges to perform clustering, since many network administrators also consider the IP address range comprising 256 addresses as the smallest unit. For each block of IP addresses, we calculate the population distribution of different types of devices using the scanning records, and cluster different IP address ranges based on the distances of the population distribution. We use hierarchical clustering to perform the clustering. The reason for is that the hierarchical clustering does not require to predefine the number of clusters.

V. Evaluation

A. Experiment Settings

To evaluate the performance of different scanning strategies, the ground truth regarding the IP-device mapping mutation records (e.g., DHCP records) is required in large-scale networks. However, such ground truth can be hardly available

from ISPs due to privacy concerns. Therefore, we set up a simulation environment to simulate the scenario where many devices change their IP addresses in large-scale networks. The simulation enables us to set various parameters and discover potential factors influencing the performance. It also allows us to use random seeds to ensure that different scanning strategies work in exactly the same environment for fair comparison.

The simulation environment is built based on two components. One is the simulation of individual devices. The other is the simulation of large-scale autonomous networks. In the simulation of individual devices, we make the IP-device mapping mutations as a nonhomogeneous Poisson process, and $\lambda(t)$ is a periodic function with a period of 24 hours for each device. For each device, we assume that the IP-device mapping mutations occur instantaneously, and the target IP address that a device transfers to is randomly selected in the remaining IP addresses of the IP address pool where the device resides.

Note that an autonomous network may have multiple IP address pools, and each pool has its own configuration. For simplicity, we assume that all devices' IP addresses are dynamically assigned. This assumption does not hinder the practical use of our strategy, as it is easy to know which IP addresses are statically configured. To be more realistic, for each IP address pool, the IP address assignment is set continuously and the number of IP addresses is a multiple of 256.

We set the number of devices and the device distribution to be different across IP address pools. We randomly generate the number of each type of device for each IP address pool, and randomly assign an initial IP address for each device. Under these settings, we mainly test two typical scenarios below:
Scenario A. All the IP-device mapping mutations are subject to the *homogeneous* Poisson process. For each combination of device type and IP address range, we assign a random λ.
Scenario B. All the IP-device mapping mutations follow a *non-homogeneous* Poisson process. Each type of device changes its address at a certain time of each day.

Scenario A can be considered as the worst case scenario for our scanning strategies due to the maximized randomness of IP-device mapping dynamics (hence little information to exploit), while scenario B is a more general and best case scenario. Accordingly, the experiments under these two scenarios can represent the lower bound and upper bound performance of the proposed strategies, respectively.

Table II details our parameter settings, including the total number of IP addresses, the total number of IP pools, the scan rate, etc. We define a round of scan as scanning all addresses once, and the scanning will end after a certain number of rounds. Note that both λ and t in Table II depend on device types and IP address ranges.

B. Scanning Performance using Different Strategies

To evaluate the performance of our online learning scanning and batch learning scanning, we compare them with naive strategies including random scanning and sequential scanning. In addition, we use a "God's view scanning" strategy, which can know the specific time when a device changes its IP address in Scenario B. Therefore, the God's view strategy can be used as the upper bound of all the scanning strategies.

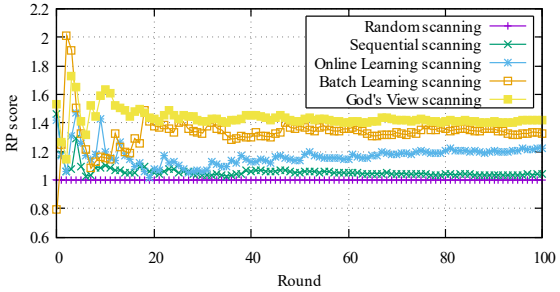


Fig. 4. A scanning example using different strategies in Scenario B. The default parameters in Table II are used.

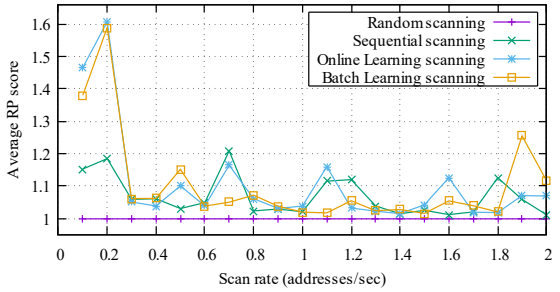
TABLE II
DEFAULT PARAMETER SETTINGS.

Parameters Name	Value
Total number of IP addresses	8,192 (32 class C networks)
Total number of IP pools	10
Scan rate	0.5 IP addresses per second
The proportion of devices to addresses	0.8
Number of device types	20
Number of scanning rounds	100
λ in Scenario A	$1/\lambda \sim$ U(0h,24h)
Address change time in Scenario B	$t \sim$ U(0h,24h)

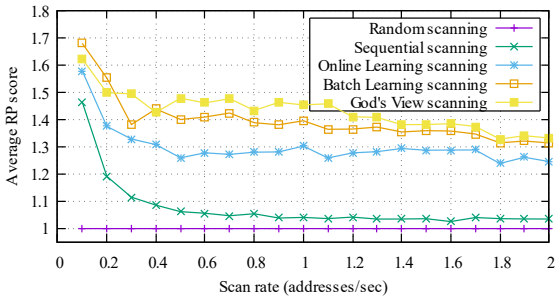
To eliminate the influence of the absolute number of IP-device mapping mutations and make a fair comparison, we define the relative performance (RP) score to measure the performance of each strategy. The RP score is defined as

$$\text{RP score} = \frac{N_i(t)}{N_r(t)}, \tag{7}$$

where $N_i(t)$ and $N_r(t)$ represent the number of IP-device mapping mutations at time t using scanning strategies i and

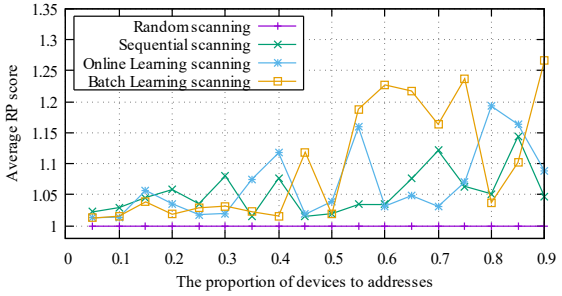


(a) Scenario A

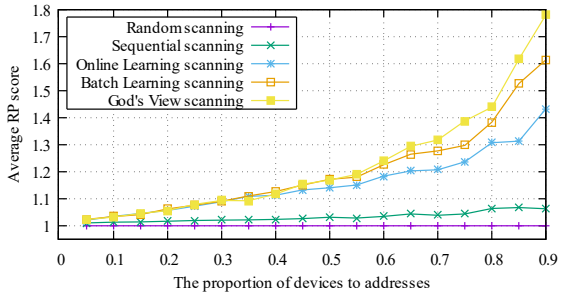


(b) Scenario B

Fig. 5. Average RP score under different scan rates. Each data point is the average of 100 experiments under the same parameters.



(a) Scenario A



(b) Scenario B

Fig. 6. Average RP score over varying proportion of devices to addresses. Each data point is the average of 100 experiments under the same parameters.

"random scanning", respectively. Particularly, when i is the random scanning, RP score (relative performance of a strategy over random scanning) is always equal to 1.

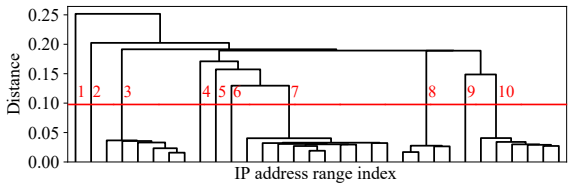
Fig. 4 shows an experiment using different strategies in Scenario B. In this experiment, for a fair comparison, we set the number of IP-device mapping mutations of each device to be the same across different strategies. The x-axis represents the round number of scanning, and one round of scanning means that the strategy finishes scanning all IP addresses. The y-axis represents the RP score of each strategy.

In Fig. 4, we see that the curve is not stable at the beginning, especially within the first 10 rounds, because of the large variance caused by insufficient samples. As the round number increases, the curve becomes more stable. At the 100th round when the scan ends, the RP scores of the strategies "God's view scanning", "batch learning scanning", "online learning scanning", "sequential scanning" and "random scanning" are 1.42, 1.32, 1.23, 1.04, and 1.00, respectively.

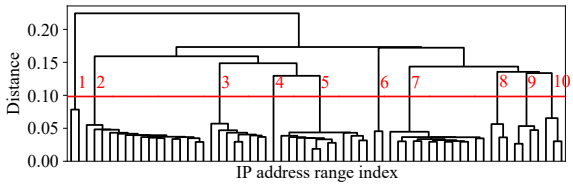
Among all scanning strategies other than the God's view, the batch learning scanning performs the best. However, such performance comes at the cost of conducting 100 rounds of scanning to collect information (i.e, the batch learning stage) before entering the continuous decision-making-based scanning. The RP score of the online learning scanning increases as the scanning proceeds. In summary, the experiment shows that our strategies significantly outperform random scanning and sequential scanning under the parameters in Table II.

C. Performance Sensitivity

Previous experiments are conducted using the parameter settings in Table II. To gain insight into the performance under different parameters settings, we next examine the impact of different parameter settings on scanning strategies.



(a) Setting 1: the number of IP address ranges=32, scan rate=0.5



(b) Setting 2: the number of IP address ranges=64, scan rate=0.25

Fig. 7. IP pool estimation using hierarchical clustering. When a threshold of 0.1 is selected, the IP address ranges (i.e., class C networks) are accurately clustered into 10 IP pools in both settings (following Table II unless specified).

Through varying a certain parameter and keeping the remaining parameters constant, we find that there are mainly two parameters that have a significant impact on the RP score of different strategies, namely, the scan rate and the proportion of devices to IP addresses. To understand the influence of these parameters on the scanning performance, we conduct experiments, and the results are shown in Fig. 5 and Fig. 6.

Consider that, even if each experiment is conducted with the same parameters, the varying λ of different device types may result in fluctuations in the experiment results. Therefore, we use the average results of multiple experiments. Each point in Fig. 5 and Fig. 6 represents the average RP score across 100 experiments with the same parameters. In each experiment, the scan is conducted 100 rounds. In Scenario A, we compare four scanning strategies, and five strategies in Scenario B. Note that the God's view strategy only works in Scenario B.

1) Scan Rate: Fig. 5(a) and Fig. 5(b) represent the performance of scanning strategies with different scan rates in Scenario A and Scenario B. We find that, as the scan rate increases, the average RP score gradually decreases. This indicates that as the scan rate tends to be positive infinity, all IP-device mapping mutations will be captured by any strategy.

The temporal fluctuation of the average RP score in Scenario B is smaller than that in Scenario A. Meanwhile, the average RP score improvement that our proposed scanning strategies make over random scanning in Scenario B is higher than that in Scenario A. We believe the reason is that the IP-device mapping mutations in Scenario A are more uncertain than that in Scenario B, thereby restricting the capability of our strategies. Particularly, when the scan rate is small, the performance improvement over sequential scanning is not significant. However, when the scan rate grows large, the performance improvement becomes significant.

2) The Proportion of Devices to IP Addresses: Fig. 6(a) and Fig. 6(b) depict the performance using different proportions of devices to IP addresses in Scenario A and Scenario B. In these two figures, the temporal fluctuation of the average RP score in Scenario A is still larger than that in Scenario B.

As the proportion of devices to IP addresses increases, the absolute number of IP address mutations captured by all strategies grows larger, but the growth rates of different strategies differ from each other. Specifically, the growth rates of our strategies are larger than those of random and sequential scanning. This implies that, as the number of IoT devices on the Internet grows, our strategies would become far more advantageous than random and sequential scanning.

3) IP Pool Estimation: We use hierarchical clustering to demonstrate the effect of IP address pool estimation. The advantage of hierarchical clustering is that we can observe the whole clustering process and choose a more realistic number of clusters. We use our proposed scanning strategies to collect scanning records in the clustering experiments. We find that the clustering results are not sensitive to the distance function, and hence we use the single-linkage clustering [21].

Fig. 7(a) shows the clustering result using the parameter settings in Table II. We initialize 8,192 IP addresses from 10 IP pools with 20 different types of devices. The smallest unit of clustering is an IP address range consisting of 256 IP addresses, i.e., a class C network, and the number of IP address ranges to cluster is 32. If correctly clustered, 10 different clusters will be obtained. Fig. 7(a) is plotted after 100 rounds of scanning. The distance threshold interval for correct clustering is [0.041,0.129] in Fig. 7(a), and [0.079,0.129] in Fig. 7(b). If a threshold of 0.1 is selected, the IP address ranges would be clustered into 10 pools, corresponding to the 10 different IP pools that we initialized. After clustering, we derive a compact representation of the IP pool-device mutation intensity matrix. The size of the compact matrix is 69.7% smaller than that of the original matrix. Consequently, each element of the compact matrix has on average 3.2 times the number of IP addresses of the original matrix, resulting in more samples for accurate intensity estimation.

VI. Related Work

In the past few years, there has been a lot of research on Internet scanning. Salient scanning tools include Nmap, Zmap, Masscan, etc [3], [15]–[17], [22]–[29]. Nmap is rich in functions, such as port scanning and device fingerprinting, but it is heavyweight and thus not suitable for large-scale scanning [15]. Zmap and Masscan use stateless scanning (e.g., no TCP three-way handshakes), thereby achieving fast scanning [27].

Several studies focused on device identification. Nmap OS fingerprinting works by sending up to 16 specially designed packets to find the ambiguities in the standard protocol [15]. Kohno et al. fingerprint devices using clock skew in device hardware [16]. Feng et al. use automatically generated rules to inspect the application layer and identify devices [17].

In the last decade, there have been cyber search engines using scanning technologies, such as Shodan and Censys [3], [25], [26]. Generally, a cyber search engine scans the IPv4 space periodically for collecting device information. Users can search device information using keywords like hostnames, IP addresses, certificates. Shodan, published in 2009, provides information of about 500 million devices every month, including operating system, hostname, version, and so forth [26].

Censys is another cyber search engine similar to Shodan. It first uses ZMap to scan the entire IPv4 address space, and then uses an application scanner ZGrab to collect the handshake information of various protocols. Censys artificially defines protocol-dependent scanning frequencies and posts a timetable of the scheduling strategy on its website [3], [27]. Some studies show that Censys has a faster scanning speed and a website update speed than Shodan [5], [30]. Both Shodan and Censys use distributed servers for scanning [26].

Using cyber search engines, several studies have analyzed a wide range of device vulnerabilities on the Internet. Durumeric et al. revealed that there are a large number of vulnerable RSA and DSA keys due to the widely used insecure random number generators [23]. Genge et al. developed a vulnerability assessment function based on Shodan and revealed high accuracy of the result with 3,922 known vulnerabilities on 1,501 services. O'Hare et al. found 12,967 potential known vulnerabilities on 2,571 services [31]. Wan et al. revealed the impact of location origins in Internet-wide scans [32]. Unlike existing studies, we focus on scanning scheduling strategies based on modeling IP-device mapping dynamics so to smartly scanning IoT devices in a principled way. Our work complements existing studies and could be incorporated into a wide range of scanning tools.

VII. Conclusion

Scanning IoT devices in a principled way is an important problem. We made the first step toward investigating this problem based on a real-world global IoT scanning platform. Our large-scale measurement study revealed that both the device type and IP address pools are related to the IP-device mapping dynamics. Inspired by this observation, we designed a system capable of smartly scheduling scans for IoT devices. The proposed system can achieve a reinforcement learning-based continuous scanning decision making process using both online learning and batch learning strategies.

Through extensive experiments, we demonstrated that our system could generally capture significantly more IP-device mapping mutations than random and sequential scanning, and approach the God's view strategy. We revealed the two key parameters affecting the performance of different strategies, i.e., the scan rate and the proportion of devices to IP addresses. We found that, as the number of IoT devices grows, our system would become far more advantageous than random and sequential scanning. Therefore, it is expected that our system would be promising as the proliferation of IoT devices.

Acknowledgement

This work was supported in part by National Natural Science Foundation (61972313, 62002306), Postdoctoral Science Foundation (2019M663725, 2021T140543), Hong Kong RGC Project (No. PolyU15223918), CCF-NSFOCUS KunPeng Research Fund, Research Fund from Huawei Technology, PolyU startup fund (ZVU7), and the Fundamental Research Funds for the Central Universities, of China. Xiaobo Ma is an XJTU Tang Scholar supported by Cyrus Tang Foundation.

REFERENCES

[1] Shodan, "The search engine for the Internet of Things," [Online], 2021, Available:https://www.shodan.io/.

[2] Censys, "Censys," [Online], 2021, Available:https://censys.io/.

[3] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *Proc. ACM CCS*, 2015.

[4] ZoomEye, "Zoomeye - Cyberspace Search Engine," [Online], 2021, Available:https://www.zoomeye.org/.

[5] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014.

[6] H. Al-Alami, A. Hadi, and H. Al-Bahadili, "Vulnerability scanning of IoT devices in Jordan using Shodan," in *Proc. IEEE IT-DREPS*, 2017.

[7] S. Torabi, E. Bou-Harb, C. Assi, E. B. Karbab, A. Boukhtouta, and M. Debbabi, "Inferring and investigating IoT-generated scanning campaigns targeting a large network telescope," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[8] X. Ma, J. Qu, J. Li, J. C. Lui, Z. Li, and X. Guan, "Pinpointing Hidden IoT Devices via Spatial-temporal Traffic Fingerprinting," in *Proc. IEEE INFOCOM*, 2020.

[9] X. Ma, J. Qu, J. Li, J. C. S. Lui, Z. Li, W. Liu, and X. Guan, "Inferring Hidden IoT Devices and User Interactions via Spatial-Temporal Traffic Fingerprinting," *IEEE/ACM Transactions on Networking*, 2021.

[10] R. Droms, "Dynamic host configuration protocol," 1997.

[11] L. Mamakos, D. Simone, R. Wheeler, D. Carrel, J. Evarts, and K. Lidl, "A method for transmitting PPP over Ethernet (PPPoE)," 1999.

[12] W. Simpson, *RFC1661: the point-to-point protocol (PPP)*. RFC Editor, 1994.

[13] G. McGregor, "The PPP Internet protocol control protocol (IPCP)," 1992.

[14] R. Padmanabhan, A. Dhamdhere, E. Aben, N. Spring *et al.*, "Reasons dynamic addresses change," in *Proc. IMC*, 2016.

[15] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[16] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[17] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-things devices," in *Proc. USENIX Security*, 2018.

[18] NSFOCUS, "Reports," [Online], 2021, Available:https://nsfocusglobal.com/company-overview/resources/#reports.

[19] G. C. Moura, C. Ganán, Q. Lone, P. Poursaied, H. Asghari, and M. van Eeten, "How dynamic is the isps address space? towards internet-wide dhcp churn estimation," in *Proc. IEEE IFIP Networking*, 2015.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[21] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[22] M. Hastings, J. Fried, and N. Heninger, "Weak keys remain widespread in network devices," in *Proc. IMC*, 2016.

[23] Z. Durumeric, "Fast internet-wide scanning: A new security perspective," Ph.D. dissertation, 2017.

[24] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS certificate ecosystem," in *Proc. IMC*, 2013.

[25] B. Genge and C. Enăchescu, "ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services," *Security and Communication Networks*, vol. 9, no. 15, pp. 2696–2714, 2016.

[26] S. Lee, S.-H. Shin, and B.-h. Roh, "Abnormal behavior-based detection of Shodan and Censys-like scanning," in *Proc. IEEE ICUFN*, 2017.

[27] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide scanning and its security applications," in *Proc. USENIX Security*, 2013.

[28] Z. Shamsi, D. B. Cline, and D. Loguinov, "Faulds: A non-parametric iterative classifier for Internet-wide OS fingerprinting," in *Proc. ACM CCS*, 2017.

[29] S. Gordeychik, D. Kolegov, and A. Nikolaev, "SD-WAN Internet Census," *arXiv preprint arXiv:1808.09027*, 2018.

[30] O. Soyer, K.-Y. Park, N. H. Kim, and T.-s. Kim, "An Approach to Fast Protocol Information Retrieval from IoT Systems," in *Advanced Multimedia and Ubiquitous Engineering*. Springer, 2017, pp. 226–232.

[31] J. O'Hare, R. Macfarlane, and O. Lo, "Identifying Vulnerabilities Using Internet-Wide Scanning Data," in *Proc. IEEE ICGS3*, 2019.

[32] G. Wan, L. Izhikevich, D. Adrian, K. Yoshioka, R. Holz, C. Rossow, and Z. Durumeric, "On the origin of scanning: The impact of location on Internet-wide scans," in *Proc. ACM IMC*, 2020.