Revealing the Black Box of Device Search Engine: Scanning Assets, Strategies, and Ethical Consideration

Mengying Wu[†*], Geng Hong[†*], Jinsong Chen[†], Qi Liu[†], Shujun Tang[‡§], Youhao Li[‡], Baojun Liu[§], Haixin Duan[§¶] and Min Yang[†]
[†]Fudan University, China, {wumy21,jschen23,qiliu21}@m.fudan.edu.cn, {ghong,m_yang}@fudan.edu.cn
[‡]QI-ANXIN Technology Research Institute, China, liyouhao@qianxin.com
[§]Tsinghua University, China, tsj23@mails.tsinghua.edu.cn, {lbj, duanhx}@tsinghua.edu.cn
[¶]Quancheng Laboratory, China

***Abstract*—In the digital age, device search engines such as Censys and Shodan play crucial roles by scanning the internet to catalog online devices, aiding in the understanding and mitigation of network security risks. While previous research has used these tools to detect devices and assess vulnerabilities, there remains uncertainty regarding the assets they scan, the strategies they employ, and whether they adhere to ethical guidelines.**

This study presents the first comprehensive examination of these engines' operational and ethical dimensions. We developed a novel framework to trace the IP addresses utilized by these engines and collected 1,407 scanner IPs. By uncovering their IPs, we gain deep insights into the actions of device search engines for the first time and gain original findings. By employing 28 honeypots to monitor their scanning activities extensively in one year, we demonstrate that users can hardly evade scans by blocklisting scanner IPs or migrating service ports. Our findings reveal significant ethical concerns, including a lack of transparency, harmlessness, and anonymity. Notably, these engines often fail to provide transparency and do not allow users to opt out of scans. Further, the engines send malformed requests, attempt to access excessive details without authorization, and even publish personally identifiable information(PII) and screenshots on search results. These practices compromise user privacy and expose devices to further risks by potentially aiding malicious entities. This paper emphasizes the urgent need for stricter ethical standards and enhanced transparency in the operations of device search engines, offering crucial insights into safeguarding against invasive scanning practices and protecting digital infrastructures.

I. Introduction

Device search engines like Censys[1] and Shodan[2] scan the entire Internet to catalog online devices, maintaining up-to-date records of hosts and services within the public IPv4 address space. These engines are crucial for helping engineers understand network security risks by offering comprehensive and robust data support. Researchers frequently utilize device search engines to build a data-driven view of device landscape and surging vulnerabilities impact. For instance, prior studies employed these engines to collect data on resident IP addresses [3], electric vehicle charging management systems [4], and insecure industrial control systems (ICS) [5].

Attackers can abuse the powerful scanning capabilities of such engines to identify vulnerable devices and establish zombie networks for malicious activities like cryptocurrency mining [6]. It is estimated that the over-collection of data by Shodan-like services led to a loss of approximately $3.86 million in 2020 alone [7]. Moreover, it remains uncertain whether these engines consider ethical implications while striving to provide competitive network assessment reports. Users who care about security and privacy have started to take action, including reporting abusive scanning IPs to AbuseIPDB [8], a public IP blocklist, and moving services from default ports to other ports. To the best of our knowledge, there has been limited effort to thoroughly examine the operational strategies, and potential ethical violations associated with these engines.

To fill this gap, this paper presents the first measurement study on the working strategies of device search engines and reveals their potential aggressive behavior and privacy issues. Our study is driven by the following research questions (RQs):

- RQ1 Can users block the IPs of device search engines to avoid being scanned?
- RQ2 Can users migrate service ports to avoid being scanned?
- RQ3 Will the scanning of the device search engine introduce any security or privacy concerns to the services being scanned?

Challenges. Prior works [9, 10] used User-Agent (UA) headers to identify traffic from search engines, as the Robots Exclusion Protocol [11] is commonly adopted by these engines. However, to profile the behavior of device search engines, *the main difficulty lies in differentiating device search engine scanning activities from others*, largely due to most device search engines' hesitance to disclose their IP lists and only web requests containing UA as labels. Even getting their IPs, there was no ready-made system available to comprehensively understand scanning strategies and detect potential ethical violations.

Insights. To find the IPs of device search engines, we identified a unique service, termed *Mirror Service*, where responses contain the IP of the request sender, as shown in Figure 1.

*These authors contributed equally to this work.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.241924
www.ndss-symposium.org

Fig. 1: IP Mirror Service. Interacting with a SIP server by `sipsak -v -s {IP}` will reply with the sender's IP. When querying `{"SIP" "received"}` in Shodan, its *ScanIP* is shown under the red mark.

Network services may include the visitor's IP for debugging, error prompting, or log metadata purposes. For example, when communicating with SIP (Session Initiation Protocol) [12], the visitor's IP address is shown when a proxy receives the request from a different address than the one specified in the header. *When device search engines scan those services, their IP addresses (ScanIP) are inevitably logged.*

Getting the *ScanIPs* helps to distill engines' action. Specifically, we use honeypots to capture in-depth behavior effectively. While simulating every device type for monitoring scans is impractical, we focused on IoT devices due to their ubiquity. Device search engines are widely used for discovering IoT devices, each offering complex services for identification. This targeted approach allowed us to capture in-depth behaviors of device search engines effectively.

Our work. We develop a systematic framework to retrieve Mirror Services from engines' search results and collect *ScanIPs* based on the Mirror Service banners. We applied our framework to four engines : Censys [13], Shodan [2], FOFA [14], and ZoomEye [15]. We also deployed honeypots to learn scan strategies and evaluate their ethical consideration of scanning. Through our innovative methodology, we gained deep insights into the actions of device search engines for the first time, ensuring that our findings are original contributions to the field, not mere reiterations of publicly available information from the device search engines themselves.

Results. Using data records in device search engines collected between March 2023 and March 2024, we collected 106,132 Mirror Services and 1,407 *ScanIPs*. FOFA has the most *ScanIPs* (665), followed by ZoomEye (166), Censys (140), and Shodan (91).

Scan Strategy. We deployed 28 honeypots across the different countries and captured 7.4 million requests from 839 *ScanIPs* from March 2023 to March 2024, totaling 4.6GB of raw logs. We found that FOFA and ZoomEye did not use fixed scanning IPs, with *FOFA typically rotating its ScanIPs every three months*. As 665 IPs we found are reported abusive in AbuseIPDB by users, the rotation may aim to avoid being blocklisted by users (see Section V-A). The port preference among the engines differs. ZoomEye primarily scanned high-risk DDoS ports, while other engines focused on common service ports like HTTPS, SSH, and Telnet (see Section V-B).

Protocol Identification. For identifying services on open ports, we found that *device search engines probe services not only on default ports but also on neighbor ports* (see Section V-C), this indicates that users who migrate the ports of services cannot conceal the service being indexed by device search engine effectively. For example, RDP is probed on ports 3388 to 3390. When engines fail to identify the default protocol, they adopt fallback strategies: most prefer HTTP and HTTPS, while FOFA switches to FTP and ZoomEye to RDP.

Ethical Scanning. Various countries have enacted cybersecurity laws [16, 17, 18] and personal information privacy laws [19, 20, 21] to safeguard people's rights. Guidelines also exist to regulate scanning and crawling behaviors. However, scanning and indexing device and service information may violate the principles of transparency, harmlessness, and anonymity. To assess the potential violation (see Section VI), we summarize the guidelines based on best practices from popular scanning tools [1, 22, 23], crawler standards [24, 25], and ethical principles [26].

As for transparent scanning, device search engines should inform individuals about who is collecting their data, why it is being collected, and how to opt-out. Notably, *users cannot discern whether scans originate from FOFA or ZoomEye* through IP homepages, WHOIS, Reverse DNS, or public listings. Apart from Censys, none of the engines provide opt-out options, and most conceal their identity in the User-Agent. Additionally, we observed that Censys does not adhere to its recommended practice [1] of explaining the scanning purpose on every probe.

For harmless scanning, device search engines should only send standard requests and access public resources. However, we observed that they send *malformed requests, attempt unauthorized data collection*, and exploit vulnerabilities, risking user privacy and security. In our investigation of 12 popular services, all four engines excessively attempted anonymous logins, retrieved system details, and enumerated database contents, exposing 214,862 Redis hosts and 135,599 FTP services that lack authentication and are vulnerable to arbitrary access.

For anonymity, we witness device search engines publishing unanonymized sensitive data in search results, including PII (name, email, avatar, screenshots, *etc.*) and database entries. Specifically, Specifically, Shodan lists data entries for 68,543 Redis hosts, while FOFA and ZoomEye publish 145,310 database indices of Elasticsearch. Notably, 904,303 snapshots of IP cameras and screenshots of remote desktops are displayed in Shodan, *as a paid service*.

Contributions. This paper makes the following contributions:

• We proposed a semi-automated framework for discovering services that can reflect *ScanIPs* of device search engines and uncover 1,407 *ScanIPs*.

• We conduct the first comprehensive analysis of the scan strategy of device search engines, demonstrating that users cannot evade scans by blocklisting scanner IPs.

• We unveil how device search engines identify protocol on ports, offering insights into how users can hide their services.

• We conducted an ethical analysis of device search engine scanning behaviors, uncovering instances where engines conceal their identities, engage in unauthorized access, and expose user camera interfaces.

II. Background and Related Work

Facing the rising requirement of internet analysis, there has been an increasing number of device search engines in recent years, as listed in Table I. These engines are specialized scanning tools that index information about internet-connected devices. They provide Internet threat intelligence, consisting of device types, running services, and potential vulnerabilities. This data is utilized by security researchers, network administrators, and even cyber attackers to locate weaknesses.

These engines collect detailed asset records, including IP addresses, ports, timestamps, geographical locations, and banner content. They also offer advanced features such as service version labeling, protocol identification, honeypot detection, certificate analysis, and vulnerability detection.

To facilitate result querying, device search engines maintain up-to-date snapshots of hosts and offer a user interface (UI) and APIs. They typically index responses and develop engine-specific search syntax, allowing users to filter and access targeted assets effectively.

Previous device search engine research has primarily focused on developing scanning techniques and toolchains, as well as analyzing internet behavior facilitated by these tools.

Scanning Tools. Internet-scale scanning tools like nmap [30] and ZMap [22] are fundamental components of device search engines, used to initiate host discovery within the address space. Censys [1], which employs ZMap to conduct single-packet host discovery scans across the IPv4 address space in 45 minutes, effectively mapping out reachable hosts. Other tools, such as IRLscanner [31] and MASSCAN [32], can scan the entire Internet in under five minutes, while Zippier ZMap [33] dramatically improves scanning speed to 4.5 minutes by parallelizing address generation and utilizing zero-copy NIC access.

Behavior Analysis with Device Search Engines. Researchers have used the indexed results from device search engines to detect potential vulnerabilities and assess their severity, particularly in IoT devices. For instance, prior works utilize device search engines to collect resident proxy IPs [3], electric vehicle charging management systems [4] and search for Mirai bots from HTTPS, FTP, SSH, Telnet, and CWMP [34], determining the types of infected devices [35]. Srinivasa et al. [36] unveiled 1.8 million misconfigured IoT devices without authority that may be exploited to perform large-scale attacks, Sasaki et al. [5] detected 890 insecure ICS devices in Japan via their WebUI and discovered 13 0-day vulnerabilities. These works highlight the significant risk of over-sharing device information.

Bot Analysis. The most relevant works of ours are [9], [10], [37], and [38]. Sun et al. [9] first measured web crawler ethicality and found most search engines respect robots.txt but misinterpret certain rules, while Li et al. [10] uncovered the behavior and features of bots, particularly exposing the extensive activity of malicious bots. However, their reliance on user agents can not tell the behavior of device search engines. Bodenheim et al. [37] evaluated Shodan's indexing and querying capabilities on ICS, while Zhao et al. [38] evaluated the vulnerability surface of IoT devices and utilized 60 days to learn the scanning period of engines. Both used records from a few servers (four and seven) to analyze engine scans on IP level. In contrast, our paper introduces a method to discover mirror services reflecting *ScanIPs*, which hasn't been reported before. Armed with these unique viewpoints, we are able to analyze previously unknown device search engine assets for the first time, analyze scanning strategies using a one-year dataset, and conduct an ethical analysis, revealing unethical practices.

Even though the evolution of network-level scanning techniques has accelerated the ability of device search engines to index Internet assets, questions remain regarding the ethics of their scanning practices. Can users blocklist their IPs or hide the services to avoid their scanning? Do device search engines conduct ethical scanning? Do they give users any ways to originate their scanning? There is concern about whether their scanning would harm devices or expose hidden vulnerabilities and sensitive data, requiring further study.

III. Preliminary Study

To understand the behaviors of device search engines, the main challenge lies in differentiating the actions of these engines from other bots or scanners, since most device search engines did not publish their ScanIP lists nor announce in User-Agent when accessing web services. This section introduces the Mirror Service, a service that can contain the requester's IP address, and demonstrates how it provides us with an opportunity to analyze device search engines.

A. Mirror Service

In network services, it's common for responses to include the IP information of the request sender, a phenomenon we refer to as "Mirror Services". These services may include the visitor's IP address in their responses for various reasons in design, such as debugging, error message, or log metadata. For example, MySQL [39] responds to illegal connection attempts by notifying the attempting IP address that it does not have permission to connect to the server, showing *"Host {IP} is not allowed to connect to this MySQL server"*. Similarly, SIP (Session Initiation Protocol) [12], a communication protocol used to establish, modify, and terminate multimedia sessions across networks, reveals the sender's IP when a User Agent (UA) or proxy receives a request from a different address than the one specified in the top `Via` header field.

B. Mirror Service in Device Search Engine

Typically, Mirror Services do not compromise security assumptions, as only the request receiver can log the sender's IP. However, the situation changes when device search engines scan these Mirror Services and display the services' responses. This inevitably exposes *ScanIP* and also provides us with an opportunity to analyze the behavior of device search engines.

To systematically survey the Mirror Services in device search engines, we identified 13 device search engines by using

TABLE I: *ScanIPs* across different services of device search engines in the preliminary study. ○ represents the *ScanIPs* are hid, ● represents the *ScanIPs* are shown in a standard form, ◑ represents the *ScanIPs* are shown in a reverse form, ◐ represents the *ScanIPs* are encoded in URL, and - represents we did not find records containing that attribute for the specified service.

Engine	Country	Year	HTTP X-Forward-For	MySql ERR_HOST	SIP Received	SMTP No Valid PTR	HTTP Location
Shodan[2]	USA	2009	●	○	○	◑	◐
ZoomEye[15]	China	2013	●	●	●	◑	◐
Censys[13]	USA	2015	●	●	-	◑	◐
FOFA[14]	China	2015	●	○	○	◑	◐
BinaryEdge[27]	Switzerland	2015	●	●	●	◑	◐
Netlas[28]	Armenia	2021	●	●	-	-	-
Hunter [29]	China	2021	●	●	●	◑	-

keywords such as "cyber asset search engine" and "device search engine" in search engines. We successfully registered accounts and accessed device data from seven of them. We manually inspected their search results concerning web services, MySQL [40], SIP [12], and SMTP [41], checking whether and how their *ScanIP* is presented.

Table I shows that Mirror Services are widely scanned and logged by device search engines. By searching engine records with specific Mirror Service traits (*e.g.,* `service:"SIP" + banner:"received"` for the SIP protocol in ZoomEye) and employing regular expressions to match IPv4 formats, IP addresses can be uncovered.

Formats of IPs. We found that IPs can be reflected in three formats—standard, reverse, and encoded—based on different service designs and requirements, as shown in Table I.

- **Standard IP**, an IPv4 address represented using dotted-decimal notation, such as "1.2.3.4".
- **Reverse IP**, *i.e., "4.3.2.1.in-addr.arpa"*, is utilized in reverse DNS queries to find the domain name associated with an IP. Besides, SMTP servers may raise exceptions with the sender's IP in reverse form if no valid PTR (Pointer) record is found.
- **URL encoding IP**, ensures that special characters are converted in URLs, preventing conflicts with URL structure and syntax. For instance, the standard form IP *"1.2.3.4"* becomes *"1%2E2%2E3%2E4"*. This encoding is often used when transmitting IPs as parameters, such as in the Location header for redirects.

Sanitized Mirror Services. Interestingly, device search engines are aware that the Mirror Services can leak their scanning assets, so they mask or replace their scanner IPs. Specifically, ZoomEye and FOFA substituted the *ScanIPs* with placeholders, *i.e., "xxx.xxx.xxx.xxx"* and *"*.*.*.*"*, respectively. Shodan uses a more advanced method, mapping its *ScanIPs* one-to-one to multicast addresses (224.0.0.0/4) [42]. Despite these measures, IP addresses can still appear in various protocols and forms, leaving many *ScanIPs* visible in search engine results.

IV. METHODOLOGY

A. Overview

Based on the preliminary study, we build a two-part framework, including *ScanIP* collection and behavior monitoring modules in this section. Figure 2 shows the overall architecture of our framework.

Scanner IP Collecting. To find the scanner IP addresses (*ScanIP*) of device search engines, our preliminary study reveals that the Mirror Service leak visiting IP information in their responses, which can inadvertently reflect the *ScanIPs*. Thus, we leverage Mirror Services to systematically collect *ScanIPs* (Section IV-B).

Search Engine Analyzing. To systematically understand scanning strategies and capture potential ethical violations by device search engines, no off-the-shelf system was available to provide such functionality. Therefore, although it is nearly impossible to simulate every type of device to monitor scanning activities, we instead focused on a highly targeted target: IoT devices, to analyze device search engines. The behavior monitoring module utilizes honeypots to collect *ScanIP* behaviors, which trap and collect the behaviors of *ScanIPs* based on IoT web honeypots and sniffer honeypots (Section IV-C).

B. Scanner IP Collecting

Figure 8 in the Appendix illustrates our scanner IP collecting module, which consists of three components: 1) Mirror Service finder for matching records by Mirror Service type patterns; 2) *ScanIP* collector for collecting *ScanIPs* from Mirror Service records; 3) Mirror type expansion module for distill new types of mirror service from *ScanIP* records.

1) Mirror Service Finder: To discover and collect services that can reflect IP (*i.e.,*Mirror Service), we propose a methodology that relies on the Mirror Service type pattern. By leveraging the collected Mirror Service type pattern, we efficiently search for relevant records on the engine and subsequently validate the authenticity of the Mirror Service, ensuring accurate and reliable results.

Candidate Mirror Service Collection. We begin collecting candidate Mirror Service instances based on known Mirror Service type identified in Section III-B. As shown in Figure 1, we first leverage service attributes (*e.g.,* `SIP`) and invariable keywords (*e.g.,* `received`) within records to query and retrieve relevant Mirror Service data. These keywords are manually defined when a new kind of Mirror Service is found and subsequently translated into engine-specific syntax, such as `protocol="sip" && banner="received"` for FOFA and `service:"SIP"+banner:"received"` for ZoomEye. Getting the queries, we executed efficient searches through APIs to gather matching host records on each engine.

Mirror Service Verification. However, not all hosts that meet the pattern requirements are Mirror Services, as some servers

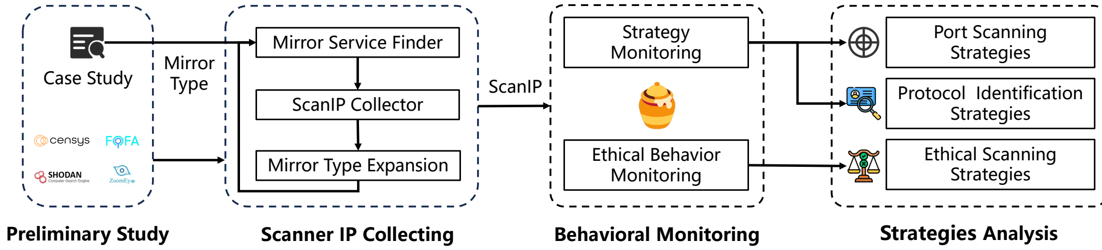


Fig. 2: Methodology Overview

may counterfeit their response (*e.g.*,, honeypots). To address this, we propose verifying the records through both static and dynamic methods. Static filtering is based on two observations: (1) a valid *ScanIP* must be a legitimate public IPv4 address, rather than a private, multicast, or reserved address, and (2) a server typically does not crawl itself using its own IP, meaning the *ScanIP* should differ from the host IP. Dynamic verification involves actively probing the host of the candidate Mirror Service. If the response contains the sender's IP, the server is confirmed as a valid Mirror Service, and the IP in the response record represents the engine's scanner IP, i.e., *ScanIP*.

2) ScanIP Collector: As device search engines periodically scan the same host to identify new services and update records, we can acquire *ScanIPs* by monitoring the results of Mirror Services on different device search engines periodically. Leveraging the search API provided by device search engines, we daily query all Mirror Services from each device search engine and extract *ScanIPs*.

3) Mirror Service Type Expansion: As device search engines may use the same *ScanIPs* to probe different services, the same *ScanIP* can appear in the records of multiple Mirror Services. By analyzing records containing these *ScanIPs* and filtering out known Mirror Services, we group similar records based on their context and semantics. Through manual inspection of these groups, we identify new types of Mirror Services and establish their service queries for further exploration and discovery. This approach enables us to expand the scope of Mirror Service from known seeds to new types based on observed patterns in collected data.

C. Behavioral Monitoring

To understand potential attack vectors and study behavioral patterns in network security, researchers frequently deploy honeypots [43]. These controlled environments allow for the observation and analysis of device search engines, bots, and potential attackers' actions. By monitoring interactions with these honeypots, valuable insights into their behaviors and strategies can be obtained.

To comprehensively understand device search engine behaviors from multiple perspectives, we enhanced our honeypot infrastructure with two tailored designs. This included a full-port closed honeypot and a popular-port open honeypot to unveil port scanning and protocol identification techniques. Further enhancing a web honeypot with IoT device emulation and comprehensive files, we aimed to attract more in-depth scanning sessions to thoroughly evaluate ethical behavior and real-world impact assessments.

1) Strategy monitoring: To gain insights into the scanning strategies employed by device search engines, we utilized the following two honeypots.

To understand the port scanning strategies, we use a full-port closed honeypot to capture the port scanning activities. Interactions between services on different ports can introduce biases in data packet counts. To ensure equal scanning across all ports, we closed them on the honeypot, allowing each port to receive only one data packet per scan.

To delve into how device search engines discern protocols on open ports, we established a honeypot with commonly used ports open. Due to resource constraints, we concentrated on the top 100 high-traffic ports based on the results from our full-port closed honeypot. Our honeypot passively acknowledged packets without other active responses at the application layer.

Additionally, we implemented a traffic monitoring function in the honeypots to capture and analyze incoming data packets comprehensively. This enables detailed analysis of scanning patterns and behaviors exhibited by device search engines.

2) Ethical behavior monitoring: As unethical behavior could potentially exist across various services, the exhaustive simulation of all services to capture such behavior is impractical. Thus, we leverage web honeypots as they offer heightened customization and facilitate the emulation of a broad array of web-based services, making them an efficient choice for capturing engines' ethical behavior.

Customized default pages for IoT devices. In the countless web services, we opted to focus on IoT devices, which are abundant in number and often riddled with vulnerabilities. Targeting IoT devices increases the likelihood of capturing anomalous scanning behavior by device search engines. IoT devices come in various types with significant differences in functionality. Therefore, we embedded the fingerprints of the IoT device management page into the default homepages of our web honeypots to simulate these devices. To ensure consistent data collection, we configured it to respond to all unknown path access requests uniformly.

Decoy paths. To gain insights into whether engines attempt to access sensitive data from hosts without proper authentication and their handling of such content, we constructed a series of decoy paths. Specifically, given that IP cameras represent another common type of IoT device with web services, we

TABLE II: Overview of Mirror Services and *ScanIPs* detected across four device search engines, with data collection from March 2023 to March 2024. The total number of Mirrors is the union of Mirrors from four engines.

Engine	# of Mirror Services	# of *ScanIPs*	# of *ScanIPs* in Honeypot
Censys [13]	45,580	481	140
Shodan [2]	611	91	81
FOFA [14]	58,671	668	579
ZoomEye [15]	3,197	167	39
Total	106,132	1,407	839

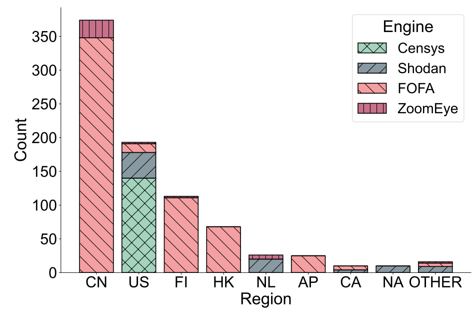


Fig. 3: *ScanIP* Region Distribution.

referred to various generic configurations of IP cameras and selected 21 typical paths for simulation. These paths can return sensitive information such as camera snapshots and device configuration files. To enhance the authenticity of the simulation, the camera snapshot paths also include dynamic timestamps to simulate real-time monitoring scenarios.

Dynamic trackable links. To delve deeper into the scanning behavior boundaries of *ScanIPs*, we implemented a dynamic linking strategy within our honeypot. Specifically, we encoded information such as client IP, port, honeypot IP, timestamp, and others for each access and embedded them as clickable links within the page body. This approach introduces variability to the links displayed with each page load, thereby increasing the complexity and uncertainty of the scanning process.

D. Implementation and Result

Based on our preliminary study of the seven successfully accessed engines, we selected all of those that offer sufficient queries and batch-automatable search API capabilities to collect the Scanner IP. As a result, we focused on four engines: Censys, Shodan, FOFA, and ZoomEye. While our findings may not apply to all engines, our study provides valuable insights into these four search engines, which are widely used in academic research such as [4, 5, 38], and are likely representative of broader industry trends. The details of implementation can be found in Appendix A.

Dataset. We deploy our honeypots in 4 cities, including Tokyo, Singapore, Beijing, and Shenzhen. In each city, we deploy five web honeypots, one closed honeypot and one open honeypot, with the same settings, to gain further insights into variations across different regions. Our data was collected from 28 honeypots deployed across 3 different countries from March 2023 to March 2024.

In the one-year dataset, in addition to the five seed mirror types in the preliminary study, 74 new mirror types were discovered after mirror type expansion, such as illegal visitor warnings in Redis and the welcome banner of ZXFS FTP. This helped us discover 1183 new *ScanIPs*. As shown in Table II, we found 106,132 Mirror Services and 1,407 *ScanIPs* in the four device search engines. FOFA boasts a significant number of *ScanIPs*, totaling up to 665, and maintains the largest number of Mirror Services. In contrast, the remaining three device search engines only possess 91 to 481 *ScanIPs*[1].

Our honeypots captured 7,362,701 requests, with 32,035 in full-port closed honeypots, 347,784 in popular-port open honeypots, and 6,982,882 in web honeypots, from 839 unique *ScanIPs* totaling 4.6GB in size. Here we define a request as a transport-layer TCP/UDP packet. Once we have identified these *ScanIPs*, we employ the earliest and latest timestamps of their records on device search engines to establish their active duration. We then filter and retain only the behaviors exhibited by these *ScanIPs* within our honeypot during this designated period, ensuring that the data we collect indeed originates from device search engines activities. Due to the different scan strategies across various engines, our honeypot can hardly capture behaviors from all *ScanIPs*. Consequently, all subsequent behavioral analysis will be based solely on the subset of *ScanIPs* observable within our honeypot.

E. Discussion

Currently, device search engines are unable to handle special IP formats in responses that they are not aware of when attempting to mask their IPs. We admit that our paper will remind the engines with mirror service in the three IP formats in Section III-B, but our methodology remains robust. Mirror service can always generate new and diverse methods to encode IPs into the response, such as 1.2.3.4 to 4-3-2-1 or 4%3%2%1, making engines hard/fail to sanitize scanner IPs. This will lead to an ongoing iterative battle between device search engines and mirror services.

V. Scan Strategy

In this section, we report the scan strategy of device search engines according to the dataset acquired from 28 honeypots from March 2023 to March 2024.

A. Landscape

1) Geographic distribution: Figure 3 shows the regions of *ScanIPs* used by each device search engine. Overall, device search engines prefer to use their own country's IPs. For example, 67% and 72% of ZoomEye and FOFA *ScanIPs* are in China, with FOFA relying on cloud ISPs and ZoomEye using multiple consumer ISPs. In the case of Shodan, 46.91%

[1]While Censys publishes its IP ranges [44], these ranges lack specificity and may introduce false positives in traffic analysis. Therefore, we collected Censys IPs using our own methodology.

of its *ScanIPs* are from the US, utilizing a combination of both enterprise and cloud ISPs for broader coverage. Censys uses enterprise ISP, brings all its *ScanIPs* come from the US. Additionally, we observed that IPs from Finland and the Netherlands are highly preferred. Specifically, 19.17% of FOFA IPs and 5.13% of ZoomEye IPs originate from Finland, while 15.38% of ZoomEye IPs and 24.69% of Shodan IPs come from the Netherlands. This may arise from their minimal restrictions, competitive prices, and hosting facilities offering high-speed large-bandwidth Internet access [45].

2) Server rotation strategy: The *ScanIPs* usage duration in device search engines reveals insights into their operational strategies. We introduce *ScanIPs* lifespan as a metric, capturing the time between their first and last appearance in honeypots. Figure 4 shows the lifespan of *ScanIPs*. We observed the lifespan of different scanners overlapping targeting the same mirror service, which indicates that one mirror service is scanned by IPs randomly selected from the *ScanIPs* pool.

All device search engines engage in the bulk activation of *ScanIPs*. For instance, Censys activated 7, 9, and 16 scanners on July 28th, August 24th, and October 24th, 2023, respectively. Similarly, we observed Shodan activate 18 *ScanIPs* across 11 network segments on October 20th.

Both ZoomEye and FOFA demonstrate patterns of IP abandonment and rotation. In FOFA, a consistent pattern emerges, characterized by four instances of mass *ScanIP* activation: occurring in mid-May, mid-August, and mid-November 2023, as well as early January 2024, followed by their abandonment around the same periods. In contrast, ZoomEye's IP changes lack periodicity, as shown in Figure 4d. Notably, in our communications, ZoomEye informed us that their IPs are dynamically assigned by ISPs, indicating that their *ScanIPs* are not fixed and may change at any moment, which aligns with the lack of periodicity. In our one-year monitoring, we did not observe Censys and Shodan significantly retiring *ScanIPs*.

We further checked AbuseIPDB [8], a blocklist where users report malicious IPs, and found 665 *ScanIPs* have been labeled with "Port Scan", "Hacking" and "Brute-Force" tags. Rotating *ScanIPs* makes scanning activities more resilient against being blocklisted by IPs.

Finding I: *FOFA and ZoomEye do not use fixed scanning assets, with FOFA typically rotating its IPs every three months, making it hard for users to avoid being scanned by blocklisting device search engine IPs.*

B. Port Scanning Strategy

Port scanning is a crucial function of device search engines, allowing users to identify open ports and their associated services. Although modern scanning tools can efficiently scan IPv4 addresses, due to resource constraints, we find that device search engines do not scan all ports of the entire IPv4 space once a day, and make trade-offs between different ports. Based on our port-closing honeypot, we can analyze the scanning preferences of different device search engines.

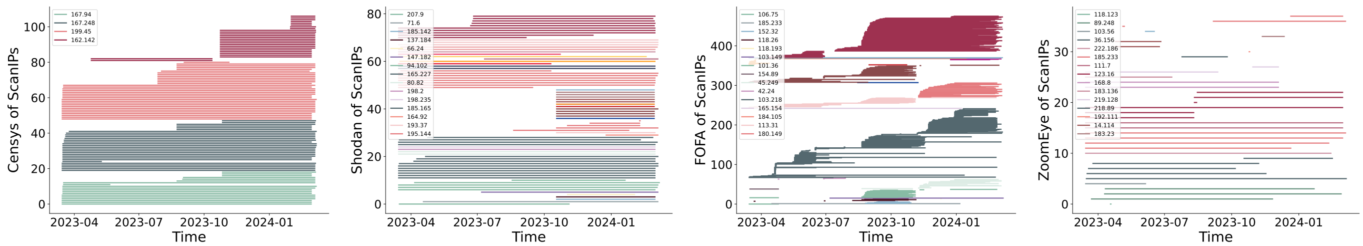
We first examined the packet setting of their port scanning. The device search engines use different TCP settings when scanning. TTL (Time to Live) indicates the packet's lifespan in the network. ZoomEye stood out with SYN packets having TTL values approaching 240, significantly higher than Shodan(110), Censys(50), and FOFA(50), which is also higher than the default TTL values of Linux/MacOS (64) and Windows (128). While a higher TTL increased the probability of packets reaching their destination, it also burdened routers, potentially leading to waste in scenarios with poor network conditions or faults, especially when there are routing loops in the network. As for TCP window size, Shodan dynamically adjusts its size between 1,024 and 65,535, while others use fixed sizes, including FOFA(1,400), Censys(42,340), and ZoomEye(63,540). A large window can facilitate faster data transmission in unstable network environments. However, considering that engines need to continuously send scanning packets, a large window may increase network load.

TABLE III: Top 10 ports scanned by each device search engines and all visitors except device search engines.

Rank	Device Search Engine				Others
	Censys	Shodan	FOFA	ZoomEye	
1	443	443	443	443	23
2	3306	2222	22	2222	3389
3	22	22	23	500	445
4	23	23	3306	53	22
5	2222	3306	2222	161	80
6	139	3389	123	5683	6379
7	32080	53	53	9001	443
8	43080	19	21	587	8088
9	21	161	8443	5060	8080
10	2323	2087	5060	123	1433

Scanning range. Despite device search engines scanning all 65,536 ports extensively, only the frequently scanned ports represent their interest. We found significant variation in the number of ports targeted by different engines. For example, 20% of Shodan and ZoomEye's traffic targets 29 ports, whereas Censys scans 49 ports, and FOFA targets seven ports.

We compare the top 10 ports they scanned and all other visitors (excluding the device search engines) in Table III and found that device search engines have unique preferences compared to the nature, usually real attackers. Ports like 445(SMB), 80/8088/8080(HTTP), 6379(Redis) and 1433(SQL server) are commonly targeted outside the device search engines. Notably, port 443 attracts the most attention from all four engines, as it is the standard port for HTTPS, which has the most possibility to catch web services. Among common HTTP/HTTPS ports, aside from 443, ZoomEye considers port 9001, FOFA and Censys consider port 8443, and Shodan includes ports 10001 and 8009. As for the other protocols, Shodan, Censys, and FOFA focus on the common services exposed to the public network. The prioritized scanning ports include SSH (22/2222), Telnet (23/2323), MySQL (3306), and NTP (123). However, ZoomEye's scanning focuses on high-risk targets. When looking at a wide range of scanning ports, 14 of the ports frequently scanned by ZoomEye ports are not preferred as the top 30 by other engines, such as CoAP (5683), game server (27015), and BitTorrent (6881). These services are frequently abused for reflective amplification DDoS attack [46, 47], highlighting ZoomEye's unique scanning behavior. This focus on less commonly monitored ports provides valuable insights



(a) Censys (b) Shodan (c) FOFA (d) ZoomEye

Fig. 4: The lifespan of *ScanIPs* in Censys and FOFA. Each line represents the lifespan of one *ScanIP*, and *ScanIPs* in the same network segment (with a /16 subnet mask) are marked with the same color.

into potential vulnerabilities and emerging DDoS threats.

> **Finding II**: *ZoomEye prefers to scan ports with a high risk of DDoS attack, while other engines focus on the most common ports on the internet.*

C. Protocol Identification Strategy

Details of running protocols on network assets are critical threat intelligence, as they help pinpoint potential vulnerabilities and targets for attacks. However, the methods employed by device search engines to effectively identify these protocols remain unclear. This study examines how device search engine identifies protocol services on the host. To solve this problem, we first developed a two-step method to identify the probes used by device search engines and subsequently conducted an analysis of their probing strategies.

1) Methodology: Identifying protocol-specific probes is not straightforward due to two challenges: (1) the probes sent by device search engines are intended to identify a wide range of protocols, leading to diverse probes with different payloads in the traffic, and (2) even within the same protocol, variations occur due to different versions or configurations, resulting in inconsistent probes of one protocol.

Rule generation. We developed a comprehensive set of rules that can encompass a broader range of protocols by using the existing common rule list and manually adding more rules.

Firstly, we utilized the rule list from nmap-service-probes[48], which contains probes for querying various services and matching expressions to recognize and parse responses. Figure 5 shows an example of "GetRequest" probe identifying an HTTP service. Besides, we employ existing network package parsers to identify other services, such as the Scapy library [49] for the TLS protocol.

Fuzzy matching. To handle the various probe variants, we refined the matching method using domain-specific protocol knowledge, enhancing the generality of the matching process. We first calculate the edit distance between unmatched probes and the acquired rule list. Then, we select the rule with the smallest distance and determine if the corresponding protocol is related to the variant. For instance, the probe for Oracle TNS in nmap rule list is "`\x00Z\x00\x00\x01\x00\x00\x00...`". However, according to the design of Oracle TNS[50], the first two bytes (*i.e.,* `\x00Z`) indicate the packet length, which is a dynamic value across packets. We improved it by "`\x00*\x00\x00\x01\x00\x00\x00...`", where $\star$ represents a wildcard.

Since off-the-shelf rule lists cannot cover all probes, we also manually survey the remaining unmatched probes. Specifically, we searched for unmatched payloads in the form of hexadecimal escape characters on Google, then inferred the purpose of the probe based on the query results.

2) Results: Analyzing the traffic captured by our popular-port open honeypots, we identified 60 types of TCP probes and 67 types of UDP probes targeting 42 protocols, covering 94.8% of the packets, as illustrated in Table IV. We summarize three different strategies as shown in Figure 7.

Probe Types. The probes we collected can be classified into two categories based on their corresponding protocol: *Specific Probe* and *Generic Probe*.

Specific Probes are effective for a specific protocol, generally the handshake messages of a particular protocol, such as "`\x6C\x00\x0B\x00\x00\x00...`", which is a hello message of X11 [51]. These probes are primarily used to detect default port numbers associated with specific protocols.

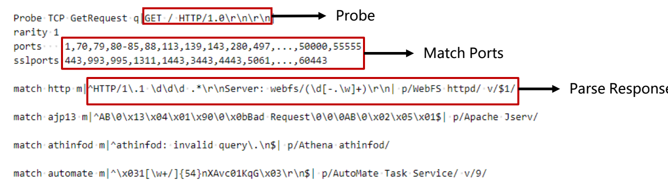


Fig. 5: A typical probe rule in nmap-service-probe [48], including service probe, match ports, and the response to parse. We utilize the probes in this list to identify part of the payloads from device search engines.

TABLE IV: The number of protocols and ports of the identified probes in different device search engines.

Engine	# of TCP_Protocol	# of TCP_Port	# of UDP_Protocol	# of UDP_Port
Censys	20	72	23	372
Shodan	21	80	34	51
FOFA	26	72	6	6
ZoomEye	16	52	32	118

On the other hand, we found three *Generic Probes* that are designed to be effective across multiple protocols, which share the same command or handshake method. For instance, the probe "help\r\n" is applicable to various services, including ident[52], SMTP[41], NNTP[53] and so on.

Neighbor strategy. Table X in the Appendix shows the identified protocol payloads alongside their corresponding port numbers. There is no doubt that device search engines prioritize utilizing Specific Probe to identify port services associated with default services, such as requesting DNS[54] probes on port 53. Additionally, beyond default service ports, device search engines also attempt to probe services on certain neighbor ports. For instance, although the default port for the X11 protocol [51] is 6000, we observed X11 probes being received on ports ranging from 6000 to 6002. Similarly, we observe RDP [55] being probed on ports 3388 to 3390, despite 3389 being the default port. Neighbor ports also include jumping ones, such as 5673 VS 5683(CoAP), and 6666/7000 VS 6379(Redis). Service deployers who wish to avoid identification should refrain from using default ports of protocols, as well as neighbor ports we listed in Table X.

Finding III: *Users cannot evade scans by migrating the ports of services they wish to hide because device search engines probe protocols not only on default ports but also on neighbor ports.*

Shared strategy. Some ports are used by multiple protocols, instead of one specific protocol, leading to potential collisions. Therefore, multiple probes from various potential protocols are sent to the same port. For example, probes for both the adb [56](Android Debug Bridge) and socks5 [57] protocols were received on TCP port 5555.

Fallback strategy. When device search engines fail to identify the protocol on specific ports, they employ a fallback strategy to explore alternative protocols, as shown in Figure 7. Consequently, multiple probes are observed across a majority of ports. All four device search engines employ a combination of `GET HTTP` and `TLS` handshake to enhance web service detection. Moreover, FOFA and ZoomEye have incorporated FTP and RDP probes into their fallback strategies, respectively.

Protocol preference. According to the probe's aim to protocol, we can learn the different preferences of device search engines in identifying protocols. Figure 6 shows the top 10 protocols/services that have the highest proportions across the four engines. Although 443 is the favorite port of all engines, Shodan is trying to find more HTTP(62.6%) services, compared to HTTPS(23.8%). FTP for FOFA(20.3%) and RDP for ZoomEye(17.4%) stand out, matching the specific generic fallback strategy unique to each engine while having a much smaller presence in the other engines.

VI. Ethical Scanning

In this section, we critically evaluate the ethical practices of device search engines from three aspects: transparency, harmlessness, and anonymity.

The operations of these engines involve accessing computer systems and collecting sensitive data, raising important ethical considerations. To safeguard citizens' computers and data, various countries have enacted robust cybersecurity and personal information privacy laws, such as the European Union's GDPR [19] and NIS2 [16], the USA's CFAA [20] and CCPA [17], as well as regulations in China [18, 21, 58], Japan [59, 60], and Singapore [61, 62]. They apply to the countries[2] where the device search engine companies are registered and where our honeypots are deployed.

Although there are currently no specific legal interpretations or industry standards explicitly applicable to device search engines, we propose a set of ethical principles aimed at safeguarding users' rights. These principles draw from best practices established by notable tools and engines such as ZMap, Censys, and Onyphe [1, 22, 23], guidelines for search engine crawlers [24, 25], and foundational ethical frameworks like the Menlo Report[26]. Key practices include transparency, harmlessness, and anonymity. Our evaluation results in these three areas are summarized in Table V.

A. Transparency

In search engine crawler standards, transparency about crawler identity is crucial, since crawlers are required to clearly inform users about data collection practices and purposes, meanwhile, users can protect opt-out rights by *robots.txt*. As device search engines cover a broader scope than search engines, they also bear responsibility for clear disclosure to signal benign scanning intent. We summarized the five best actions for transparency. In total, Censys and Shodan have made conscious efforts to make their identities and activities transparent to users, while FOFA and ZoomEye are not.

Explain the purpose of every probe. Network administrators may be wary of unauthorized scans, but if they understand the purpose is to identify vulnerabilities and offer security recommendations, they are more likely to permit such activities. The best practice involves hosting a website on port 80 of each *ScanIP* to describe the purpose and nature of the scan, recommended by three tools/engines that proposed best practices. As an alternative approach, declaring identity in the User-Agent header during HTTP scans can also signal intent, however, only HTTP scanning can be informed.

Unfortunately, testing revealed that none of the *ScanIPs* from the four engines provided such information, even Censys said it in its paper [1]. Specifically, we found that 83 FOFA IPs and 9 ZoomEye IPs had port 80 open, but the content varied significantly, ranging from nginx test pages and device login pages to WordPress websites, indicating that these IPs may be resold and used by others.

Finding IV: *Although Censys proposed the best practice of hosting a website on port 80 of each ScanIP to describe the scan's purpose and nature, no engines, including Censys itself, fully follow it in implementation.*

[2]The inclusion of European Union regulations is essential due to the international nature of data flow and the stringent requirements of GDPR. Article 3, paragraph 2(b) of the GDPR [19] stipulates that the regulation applies to any entity processing and monitoring the data of EU citizens, regardless of the entity's location.

Fig. 6: Top 10 protocol/services that have the highest proportions across the four engines.

TABLE V: Ethical violation of device search engines.

Type	Action		Censys	Shodan	FOFA	ZoomEye
Transparency[1]	Explain the purpose on every probe.		😐	😈	😈	😈
	Publish probes IP address list for opt-out.		😊	😈	😈	😈
	Use fixed IP addresses instead of trashable ones.		😊	😊	😈	😈
	Set whois records with organization and abuse email.		😊	😐	😈	😈
	Reverse DNS pointing to the company.		😐	😐	😈	😈
Harmlessness[2]	Malformed requests		😊	😊	😊	😈
	Unauthorized Access Service	**Minimized Probe**				
	FTP	Null Probe	😊	😈	😈	😈
	Redis	Command: ping	😈	😈	😈	😈
	ZooKeeper	Command: ruok	😈	😈	😈	😈
	ElasticSearch	Path: /	😈	😈	😈	😊
	MongoDB	Command: mongo	😈	😈	😈	😈
	RDP	RDP Handshake	😊	😈	😈	😈
	LDAP	LDAP Handshake	😈	😈	😊	😊
	Memcached	Command: stats	😊	😈	😊	😊
	CouchDB	Path: /	😊	😈	😈	😊
	IP Camera(Web Service)	Path: /	😊	😈	😊	😊
	OpenWrt Router(Web Service)	Path: /	😊	😊	😊	😈
	Prometheus(Web Service)	Path: /	😈	😊	😊	😊
Anonymity[3]	FTP		😐	😐	😐	😐
	Redis		😐	😈	😐	😐
	ZooKeeper		😐	😐	😐	😐
	ElasticSearch		😐	😈	😈	😐
	MongoDB		😐	😈	😈	😐
	RDP		😊	😈	😈	😈
	LDAP		😈	😈	😊	😊
	Memcached		😐	😐	😐	😐
	CouchDB		😐	😈	😈	😐
	IP Camera		😊	😈	😊	😊

[1] 😊 indicates scanners obey guidance, 😐 indicates scanners obey the guidance partially, and 😈 indicates all scanner violate transparency principle.
[2] 😊 indicates that the engine only sends standard and minimized probes, and 😈 indicates the use of malformed or infiltrated requests.
[3] 😊 indicates that PII has been fully anonymized, 😐 indicates only the assist software version, which may facilitate attacker infiltration, and 😈 indicates sensitive PII has not been anonymized and leaked.

Only Censys identifies itself as "Mozilla/5.0 (compatible; CensysInspect/1.1; +https://about.Censys.io/)" in User-Agent, with 31% of scans targeting the root directory lack a UA. Other engines claim to be users of Chrome or Firefox on Windows, Linux, or macOS, as shown in Table IX.

Publish scanner IP address list for opt-out. To respect users' privacy and information security, an opt-out option should be provided, as required by major privacy regulations [17, 19, 60]. Unfortunately, only Censys offers explicit instructions for opting out of scanning activities. We observed that FOFA responds aggressively to users who do not want to be scanned, advising them not to place their devices on the external network [63].

Censys takes a proactive approach by publishing IP ranges and suggesting users block their access via firewalls. They also inform users about filtering scans using the User-Agent. This transparency reflects Censys' commitment to ethical scanning practices and respect for user privacy.

Use fixed IP addresses instead of tractable ones. As we discovered in Section V-A2, both FOFA and ZoomEye rotated their *ScanIPs* in one year, with FOFA specifically replacing its IP pool every three months. This practice poses a challenge for users attempting to evade scans by configuring their firewalls.

Set whois records with organization and abuse email. This helps users to easily identify and contact the engines in case of any abuse or issues related to their IP addresses. Among the four engines, Censys is the only one that set its *ScanIPs* with its own abuse email and organization. In contrast, FOFA and ZoomEye both utilize the whois information of their cloud service providers, rather than maintaining their own information. Shodan, similarly, only has an IP segment with 11 *ScanIPs* with whois information associated directly with Shodan, while 81.7% of its ScanIPs use the Whois details provided by its cloud service providers.

Reverse DNS pointing to the company. Among the four

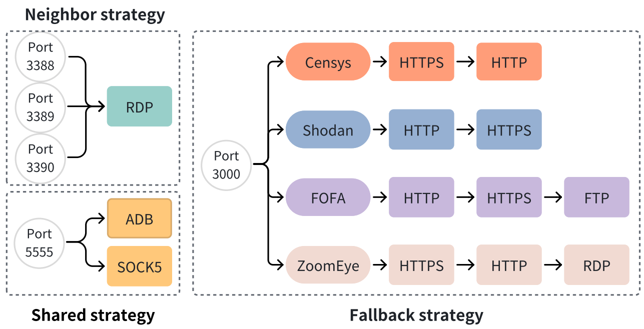


Fig. 7: Three probe strategies across the four engines. The order of fallback probes is sorted according to the sequence of probes.

engines, only Shodan and Censys have reverse DNS records associated with their scanning IPs. Shodan's reverse DNS records point to scanf.shodan.io or census.shodan.io, while Censys' records point to censys-scanner.com. Notably, there are 23 *ScanIPs* from Shodan and 24 *ScanIPs* from Censys without corresponding reverse DNS records. Interestingly, the IPs lacking RDNS records from Censys are within their publicly announced IP ranges. Regarding Shodan, the *ScanIPs* come from the same subnet, suggesting their association with Shodan. In contrast, neither ZoomEye nor FOFA assigns reverse DNS records to their *ScanIPs*.

Finding V: *Through the analysis of 1,407 ScanIPs, users cannot identify whether the scans originate from FOFA or ZoomEye. This makes users hard to identify and evade scanning.*

B. Harmlessness

Cybersecurity-related laws protect computers attacks, including intentionally accessing a computer without authorization or exceeding authorized access to obtain information or recklessly causing damage. Device search engines ensure that their scanning activities are harmless, such as only sending standard requests and accessing permitted resources. However, we observed harmful scanning in our honeypots, from all device search engines, including malformed requests and attempts at unauthorized access, which may lead to system error, data breaches, or vulnerability exposure.

We first investigate the default scanning paths of them, as shown in Table VII. All engines commonly request the root path (/) and the icon (/favicon.ico), which indicate the web server's existence and functionality. Additionally, Shodan and ZoomEye access robots.txt, security.txt, and sitemap.xml files, providing supplementary website information. Notably, security.txt files provide pathways for reporting security issues, facilitating communication between researchers and administrators, and highlighting the engines' proactive role in enhancing cybersecurity practices.

1) Malformed requests: Sending malformed data packets or protocol requests can potentially lead to abnormal behaviors in target systems or network devices. We found that ZoomEye, by default, employs a malformed probe in the form of "GET /nice%20ports%2C/Tri%6Eity.txt%2ebak HTTP/1.0\r\n\r\n" for all web services, which we can decode to a more friendly "/nice ports,/Trinity.txt.bak". This probe comes from Nmap's service detection [64], uses ASCII escaped characters in an attempt to generate an HTTP 404 error message to probe a web server, which is one of the top four web services exploits in 2019 [65]. A successful scan can reveal crucial details about the web server's codebase and potentially even expose vulnerabilities through response headers and error messages. As a result, this technique is often exploited by attackers.

2) Unauthorized access: Unauthorized access involves bypassing security measures, exploiting vulnerabilities, and leveraging weak authentication. An ethical device search engine should adhere to data minimization principles during scanning, avoiding unauthorized access to sensitive paths on a user's host to prevent potential privacy breaches. Specifically, these engines attempt to access paths requiring authentication but are often left insecure. This behavior indicates that some engines may view user data as a key component of their commercial value, without users' knowledge or consent.

Minimized scanning. To clarify whether engines acquire data unethically when finding a service, we first define the minimized actions and the infiltration actions in scanning.

Given that device search engines provide service tags as part of their functionality, we define the minimized action as probing to confirm a service on a port and ceasing further scanning, aligning with data minimization principles. In contrast, infiltration probes aim to get more detailed and private service information once a service has been identified.

For instance, a minimized probe, i.e., *"GET /"* request is enough to verify ElasticSearch web server. However, using "/_cat/indices" will over-collect database indices. Besides, some services, such as MongoDB, connecting successfully with specific tools can confirm the existence of the MongoDB service. Any subsequent interaction probes are considered infiltration. Similarly, fetching the FTP welcome banner after the handshake can know an FTP server, just like opening a webpage. Attempting anonymous FTP login is like infiltrating the webpage's login system, exposing weakly protected hosts and aiding attackers in identifying potential victims.

We selected ten common services vulnerable to unauthorized access and deployed them as response templates. We used interaction tools to probe and understand the requisite actions needed to elicit various responses and then defined the minimized probe for each service. We found that some services even do not require specific clients for confirmation, for example, the `PING` command is sufficient to determine the presence of Redis on a host, evidenced by the response "PONG". Also, send `ruok` to ZooKeeper will receive `imok`.

We use engine records and traffic captured by our honeypots to determine whether the engines attempt unauthorized access to infiltrate services. Specifically, we searched the host records containing these services in each device search engine and manually inspected the first 100 entries to check for potential excessive data acquisition, based on our defined minimized probe. For web services, we learn from our honeypots.

Infiltration of device search engines. The result in Table V shows that unauthorized access is widely attempted among the device search engines. Six services are infiltrated by at least three engines. Engines connect and then send additional commands. This includes anonymously logging in to FTP, getting system details of ZooKeeper(`stat`) and RDP, and enumerating the databases of ElasticSearch(`/_cat/indices`), MongoDB(`show dbs`), and Redis(`keys *`).

The probe, leveraging by three engines (except for Censys), used for RDP is exploiting a vulnerability with a CVSS3 score of 9.8 [66]. The script `rdp-ntlm-info` in Nmap sends an incomplete CredSSP (NTLM) authentication request with null credentials, which causes the remote service to respond with an NTLMSSP message of CredSSP (NTLM).

Successful infiltrations exposed weakly protected hosts lacking authentication. For instance, the non-error response of the INFO command on Redis hosts granted unauthorized access to Redis information, revealing that 74.97%(59,725/79,664) of Redis hosts listed on Shodan and 182,137 hosts on Fofa are vulnerable to arbitrary access. Similarly, the success in logging into FTP service exposes 135,599 FTP hosts listed in ZoomEye do not need authentication. Only Censys, who does not infiltrate FTP host, will not tell the authentication information. The issue surrounding ZooKeeper is even more critical. While a `ruok` request or client handshake can confirm the service's existence, the status response to the "stat" command reveals 99.91% (369,552/369,881) of the recorded hosts are vulnerable to unauthenticated access, with only 0.09% of the hosts resisting these probes.

Shodan is the most serious among the four engines, almost infiltrating all services. Memcached [67], and IP Cameras are only infiltrated by Shodan. For Memcached, Shodan sent a "stats" command followed by "stats settings" to retrieve additional configuration information. Our honeypots detected that Shodan attempted to access and retain 25 sensitive paths for IP camera configuration details and real-time feeds, violating user privacy as outlined in Table VIII. Notably, some hosts provided by Shodan did not offer real-time images in their root directories, suggesting Shodan probed deeper paths, confirming our honeypot findings. Also, this helps the attackers locate and exploit IP cameras that are accessible without authentication, who can abuse it for illicit camera spying and exacerbating the sale of voyeuristic content. Evidence of this trend lies in Shodan's Explore module, where seven of the top 10 voted queries focus on seeking live webcam feeds, with one even titled "live sex cam".

Although Censys claims that they never try to log into any service, read any database, or gain authenticated access to any system, we still find Censys infiltrated six services, such as getting server detail of ElasticSearch(`/_nodes`) and MongoDB(`isMaster` and `Buildinfo`). Also, Censys extracts the user's information including email, company, department and telephone in the Lightweight Directory Access Protocol (LDAP) [68] service. As for web services, Censys scanned nine paths of the Prometheus server, which facilitates monitoring system metrics and alerting. However, Censys stated that it introduced granular recognition for Prometheus in 2019 [69], allowing users to search for exposed Prometheus endpoints.

Besides, Zoomeye enumerates files using `LIST` command after logging in FTP, and accesses the sensitive path "/cgi-bin/luci/" of OpenWrt routers, a web interface that allows users to configure and manage the router through the browser. FOFA exhibits a preference for enumerating indices and extracting database information within database-like services, such as ElasticSearch [70] and CouchDB [71]. We also witness FOFA acquiring "/_cat/indices" for ElasticSearch in our honeypots.

Interestingly, in our communications with these device search engines, they all claimed to have only scanned the root directory or paths like robots.txt, explicitly denying any scanning of sensitive paths. However, when asked about discrepancies between their behavior and claims, they refused to answer and ended communication.

Finding VI: *The device search engines send malformed requests, attempt to access excessive details without authorization, and even exploit vulnerabilities, posing risks to user privacy and security.*

C. Anonymity

Anonymity refers to hiding personal information when displaying search results. The privacy laws require that published data cannot identify specific natural persons and cannot be reversed or reconstructed, to prevent user data from leakage. Even when engines use minimized probes in Section VI-B, certain responses can still contain private information. Failure to anonymize the privacy before displaying on search results can lead to privacy leakage risks.

Privacy data in device search engine result. Due to the variety of types of private data and their different definitions, we specifically focus on privacy data in the ten services' responses, includes host names, user names, avatars, emails, geographical location, screenshots, *etc.* Such information might be abused by network attackers, such as launching social engineering. In addition, the leakage of database information may lead to theft of valuable user information.

Leaking software versions has caused huge risks. The OWASP top 10: 2021 [72] highlights "Vulnerable and outdated components", indicating that many hosts have not been updated to the latest version and remain susceptible to security threats. What's more, security companies [73, 74, 75] treat version disclosure as a vulnerability, as attackers can exploit known vulnerabilities associated with disclosed versions.

Privacy leakage. Here we use the same assessment method with Section VI-B, and manually inspect the first 100 search records of 10 services. Considering that engines may not categorize software versions as privacy and widely publish this information, we isolated it in our experiments to avoid influencing the results of other privacy leaks, using a yellow face in Table V. The result shows that Shodan exposed database or PII for 7 out of 10 services. Furthermore, version information is widely exposed in the records.

Database services' data indices are being exposed and displayed, as intentionally customized features tailored for these services. We found 145,310 database indices of Elasticsearch, 178,879 indices of MongoDB, and 2,306 databases of CouchDB are showing on Shodan and FOFA. What's more, Shodan lists 68,543 Redis hosts with their keys.

What's worse, Shodan provides an image display platform on https://images.shodan.io/, displaying images of IP cameras they discovered and log-in screenshots of RDP services with the avatars and usernames. 65,042 camera snapshots and 791,333 remote desktop screenshots are displayed upon submission of this paper.

In contrast, Censys strives to mitigate privacy risks by displaying only relevant fields from responses, successfully avoiding leaking any private information on RDP and ZooKeeper. However, they still inadvertently leaked 230 LDAP user data (including name, email, company, address, and phone), along with Elasticsearch node and MongoDB device configurations.

ZooKeeper [76] typically reveals all connected clients by default. Notably, FOFA masks all client IP addresses when displaying ZooKeeper results, while ZoomEye only masks its own IP. This highlights the different approaches various engines take in handling sensitive information.

Finding VII: *The device search engines fail to anonymize asset sensitive data, including PII (735 phone numbers, 65,042 cameras, 791,333 remote desktop screenshots, etc.), 326,495 database index and entries, before publishing on their search results.*

VII. Discussion

A. Suggestions

By uncovering the scanner IPs of device search engines, our original findings expose significant ethical considerations in the engines' scanning activities, *i.e.,* lack of transparency, harmlessness, and anonymity. Our findings underscore the pressing need for stringent ethical standards and regulatory oversight in the use of these engines. Therefore, based on the behavior of device search engines, we propose the following suggestions for both users and engines.

To avoid being scanned, users can use WHOIS and reverse DNS records to find and block IPs from transparent engines. For those engines that do not use fixed IPs, users can leverage public blocklists such as AbuseIPDB, as 47.26%(665/1,407) scanner IPs we found have been reported and labeled. Users can also report suspicious scanning IPs to help others. If users have to expose services on the public network, we recommend concealing them by migrating default ports to random ports, rather than neighbor ports or the ports we show in Table X.

Our research reveals a substantial number of unauthenticated services exposed to attackers due to excessive infiltration by the device search engines. Users should check if their services are left unauthenticated, as we have found that at least 48 CVEs associated with 10 services require unauthenticated access as an entry point, leading to potential risks such as arbitrary code execution and denial-of-service attacks.

We suggest that device search engines enhance their ethical scanning practices. To improve transparency, they should clearly explain the purpose of each scan and provide users with an opt-out option by publishing a list of scanner IP addresses. Additionally, using fixed IP addresses instead of disposable ones can further improve trust. Moreover, device search engines should minimize potential harm to hosts by sending only standard, minimal probes and should refrain from exploiting any vulnerabilities or unauthorized services. They must also avoid excessive probing of user devices to enhance functionality, particularly when it involves accessing private data. Finally, to protect user privacy when displaying data, engines should anonymize any potentially sensitive information. We recommend that engines report vulnerabilities and privacy leaks to the appropriate stakeholders rather than disclosing them publicly in search results or other open channels.

B. Limitations

Although our work has uncovered 1,407 *ScanIPs* and their strategies, we have some limitations. First, since our *ScanIPs* are collected based on TCP services, any device search engine that differentiates between TCP and UDP scanning may miss *ScanIPs* dedicated solely to UDP services and their corresponding behaviors. Also, our method only captures exposed *ScanIPs*; many scanner IPs may remain undiscovered if they have not scanned easily exposed services.

Secondly, the artificial nature of honeypots may bring potential bias. Honeypots are commonly used in cybersecurity research, such as [5], [36], [35]), reflecting real-world attack patterns and offering valuable insights into network threats. To minimize potential bias, our honeypots are deployed and configured to closely mimic real system behavior, reducing discrepancies with real-world environments. For example, we decoy camera snapshots with dynamic timestamps to simulate real-time monitoring scenarios. Additionally, we deployed honeypots across multiple countries and collected data over a year to ensure diversity and representativeness.

However, given our honeypot number and monitoring period, certain aspects may lack statistical conclusions, such as scanning preferences across geographical regions and the periodicity of full-port scans. We believe that deploying more honeypots over an extended duration may give the conclusions.

Due to limitations in computing resources and manpower, our understanding of device search engines' behavior is primarily focused on web services representative of IoT devices. Expanding the services simulated on our honeypots could provide insights into a broader range of engines' behaviors. For services beyond the web, we also manually examined their privacy leakage behaviors in Section VI-C.

C. Ethics and Disclosure

Ethic concerns. In our research, we adhere to ethical guidelines by utilizing publicly available data provided by device search engines to locate their *ScanIPs*, without engaging in any database attacks against these engines. Additionally, during our periodic searches for IP Mirror Services, we strictly abide by the limitations of our purchased membership account, responsibly using their query API.

Disclosure. During the process of collecting traffic in our honeypots, we actively engage with device search engines. Each engine responded positively when we reported suspicious scanning IPs. For example, upon discovering 14 IPs using Censys' user agent but not listed in Censys' public IP range, we promptly reported this to Censys and confirmed that they were indeed fake. Additionally, we encountered instances where attackers exploited discarded *ScanIPs* from FOFA and ZoomEye to conduct malicious scans against us. After reporting these incidents, both FOFA and ZoomEye confirmed that the IPs in question had been abandoned.

VIII. Conclusion

This study presents the first comprehensive assessment of the assets, operational strategies, and ethical concerns of device search engines, providing original findings rather than reiterating existing information. Through innovative methodologies, we collected 1,407 scanner IPs and demonstrated that users could hardly evade scans by blocklisting scanner IPs or migrating service ports. Our research exposes significant ethical breaches—primarily the lack of transparency, harmlessness, and anonymity in their scanning activities. These findings underscore the pressing need for stringent ethical standards and regulatory oversight in the use of these engines, which are pivotal in network security but also pose risks to user privacy.

Given these issues, we advocate for the formulation of clear ethical guidelines and the establishment of robust regulatory frameworks to govern the operations of device search engines. In conclusion, while device search engines are invaluable for network security, their responsible use is paramount. Our study calls for a balanced approach that aligns the powerful capabilities of these tools with stringent ethical practices, thus protecting users and strengthening the security landscape.

Acknowledgment

We would like to thank Bingjing Song for providing technical and data assistance for this paper. We also thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by National Natural Science Foundation of China (62302101). Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China.

References

[1] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 542–553.

[2] Shodan. (2013) Shodan.io. https://www.shodan.io/.

[3] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu, "Resident Evil: Understanding Residential IP Proxy as a Dark Service," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1185–1201, iSSN: 2375-1207.

[4] T. Nasr, S. Torabi, E. Bou-Harb, C. Fachkha, and C. Assi, "ChargePrint: A Framework for Internet-Scale Discovery and Security Analysis of EV Charging Management Systems," in *Proceedings 2023 Network and Distributed System Security Symposium*. San Diego, CA, USA: Internet Society, 2023.

[5] T. Sasaki, A. Fujita, C. H. Gañán, M. van Eeten, K. Yoshioka, and T. Matsumoto, "Exposed Infrastructures: Discovery, Attacks and Remediation of Insecure ICS Remote Management Devices," in *2022 IEEE Symposium on Security and Privacy (SP)*, May 2022, pp. 2379–2396, iSSN: 2375-1207.

[6] A. C. Security. (2019) Xulu: Cryptojacking leveraging shodan, tor, and malicious docker container. https://www.alibabacloud.com/blog/594869.

[7] M. Lutz. (2023) Shodan: The most terrifying, dangerous search engine in the world. https://www.linkedin.com/pulse/shodan-most-terrifying-dangerous-search-engine-world-martin-lutz.

[8] A. LLC. (2024) Abuseipdb - ip address abuse reports. https://www.abuseipdb.com/.

[9] Y. Sun, I. G. Councill, and C. L. Giles, "The ethicality of web crawlers," in *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1. IEEE, 2010, pp. 668–675.

[10] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *2021 IEEE symposium on security and privacy (sp)*. IEEE, 2021, pp. 1589–1605.

[11] M. Koster. (1994) A standard for robot exclusion. http://www.robotstxt.org/orig.html.

[12] E. Schooler, J. Rosenberg, H. Schulzrinne, A. Johnston, G. Camarillo, J. Peterson, R. Sparks, and M. J. Handley, "SIP: Session Initiation Protocol," RFC 3261, Jul. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3261

[13] Censys. (2015) Censys search. https://search.censys.io/.

[14] FOFA. (2024) Fofa search engine. https://fofa.info/.

[15] ZoomEye. (2013) Zoomeye - cyberspace search engine. https://dev.zoomeye.org/.

[16] E. Parliament and C. of the European Union, "Directive (eu) 2022/2555 of the european parliament and of the council of 14 december 2022 on measures for a high common level of cybersecurity across the union, amending regulation (eu) no 910/2014 and directive (eu) 2018/1972 (nis2 directive)," https://eur-lex.europa.eu/eli/dir/2022/2555, 2022, accessed: 2024-06-26.

[17] S. O. C. D. O. JUSTICE. (2024) California consumer privacy act (ccpa). https://oag.ca.gov/privacy/ccpa.

[18] T. S. C. of the National People's Congress, "Cybersecurity law of the people's republic of china," https://www.gov.cn/xinwen/2016-11/07/content_5129723.htm, 2016, accessed: 2024-06-26.

[19] E. Parliament. (2016) General data protection regulation. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[20] U. S. Congress, "Computer fraud and abuse act," https://www.law.cornell.edu/uscode/text/18/1030, 1986, accessed: 2024-06-26.

[21] T. S. C. of the National People's Congress, "Data security law of the people's republic of china," http://www.npc.gov.cn/npc/c30834/202106/bfbbcf742f2849e79db79fd9c1c824e1.shtml, 2021, accessed: 2024-06-26.

[22] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," 2013, pp. 605–620.

[23] ONYPHE. (2022) Our 10 commandments for ethical internet scanning. https://www.onyphe.io/docs/write-ups/our-10-commandments-for-ethical-internet-scanning.

[24] M. Koster, G. Illyes, H. Zeller, and L. Sassman, "Robots Exclusion Protocol," RFC 9309, Sep. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9309

[25] M. Koster. (1993) Guidelines for robot writers. https://www.robotstxt.org/guidelines.html.
[26] D. Dittrich, E. Kenneally *et al.*, "The menlo report: Ethical principles guiding information and communication technology research," US Department of Homeland Security, Tech. Rep., 2012.
[27] BinaryEdge. (2024) Binaryedge. https://www.binaryedge.io/.
[28] netlas. (2024) Netlas: Comprehensive internet-wide scanning & osint platform. https://netlas.io/.
[29] QIANXIN. hunter.qianxin.com. https://hunter.qianxin.com/.
[30] G. Lyon. (2024) Nmap: the network mapper - free security scanner. https://nmap.org/.
[31] D. Leonard and D. Loguinov, "Demystifying service discovery: implementing an internet-wide scanner," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 109–122.
[32] R. D. Graham, "Masscan: Mass ip port scanner," *URL: https://github. com/robertdavidgraham/masscan*, 2014.
[33] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman, "Zippier zmap: Internet-wide scanning at 10 gbps," in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.
[34] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *the Proceedings of the 26th USENIX Security Symposium*, Vancouver, BC, Canada, Aug. 2017.
[35] O. Cetin, C. Ganan, L. Altena, T. Kasama, D. Inoue, K. Tamiya, Y. Tie, K. Yoshioka, and M. van Eeten, "Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai," in *Proceedings 2019 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2019.
[36] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Open for hire: Attack trends and misconfiguration pitfalls of iot devices," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 195–215.
[37] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548214000213
[38] B. Zhao, S. Ji, W.-H. Lee, C. Lin, H. Weng, J. Wu, P. Zhou, L. Fang, and R. Beyah, "A large-scale empirical study on the vulnerability of deployed iot devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1826–1840, 2020.
[39] MySQL. (2024) 8.2.22 troubleshooting problems connecting to mysql. https://dev.mysql.com/doc/refman/8.0/en/problems-connecting.html.
[40] Oracle. (2000) "mysql." mysql.com, oracle. www.mysql.com/.
[41] D. J. C. Klensin, "Simple Mail Transfer Protocol," RFC 5321, Oct. 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5321
[42] D. Meyer, M. Cotton, and L. Vegoda, "IANA Guidelines for IPv4 Multicast Address Assignments," RFC 5771, Mar. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5771
[43] W. Contributors. (2018) Honeypot (computing). wikipedia. https://en.wikipedia.org/wiki/Honeypot_(computing).
[44] Censys. (2024) Opt out of data collection. https://support.censys.io/hc/en-us/articles/360043177092-Opt-Out-of-Data-Collection.
[45] googiehost. (April 13, 2024) best-vps-hosting-netherlands. https://googiehost.com/blog/best-vps-hosting-netherlands/.
[46] Netscout. (2019) Coap attacks in the wild. https://www.netscout.com/blog/asert/coap-attacks-wild.
[47] K. El Defrawy, M. Gjoka, and A. Markopoulou, "Bottorrent: Misusing bittorrent to launch ddos attacks." *SRUTI*, vol. 7, pp. 1–6, 2007.
[48] Nmap. (2024) nmap-service-probes. https://svn.nmap.org/nmap/nmap-service-probes.
[49] secdev. (2024) Github - secdev/scapy: Scapy: the python-based interactive packet manipulation program & library. supports python 2 & python 3. https://github.com/secdev/scapy.
[50] Oreilly. (2024) The oracle® hacker's handbook: Hacking and defending oracle by david litchfield. https://www.oreilly.com/library/view/the-oracle-hackers/9780470080221/9780470080221_the_tns_protocol.html.
[51] R. Scheifler, "X Window System Protocol, version 11: Alpha update April 1987," RFC 1013, Jun. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1013
[52] M. StJohns, "Identification Protocol," RFC 1413, Feb. 1993. [Online]. Available: https://www.rfc-editor.org/info/rfc1413
[53] C. Feather, "Network News Transfer Protocol (NNTP)," RFC 3977, Oct. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc3977
[54] "Domain names - implementation and specification," RFC 1035, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1035
[55] Deland-Han. (2023) Understanding the remote desktop protocol (rdp). https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol.
[56] A. shell. (2024) adb connect - adb shell. https://adbshell.com/commands/adb-connect.
[57] M. D. Leech, "SOCKS Protocol Version 5," RFC 1928, Mar. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1928
[58] T. S. C. of the National People's Congress, "Personal information protection law of the people's republic of china," http://www.npc.gov.cn/npc/c30834/202108/55d2e3b9f2e84d5db37f0e30f43b1821.shtml, 2021, accessed: 2024-06-26.
[59] T. J. Parliament, "Act on prohibition of unauthorized computer access," https://www.japaneselawtranslation.go.jp/en/laws/view/2033, 1999, accessed: 2024-06-26.
[60] ——, "Act on the protection of personal information," https://www.ppc.go.jp/en/legal/, 2003, accessed: 2024-06-26.
[61] P. of Singapore, "Personal data protection act 2012 (no.

26 of 2012),” https://sso.agc.gov.sg/Act/PDPA2012, 2012, accessed: 2024-06-26.

[62] ——, “Computer misuse act (revised edition 2007),” https://sso.agc.gov.sg/Act/CMA1993, 1993, accessed: 2024-06-26.

[63] fofa_help_center. (2024) Help center - fofa. https://fofa.info/help_articles/list#%E5%9F%BA%E6%9C%AC%E9%97%AE%E9%A2%98.

[64] Doug. (2024) Re: More service detection notes: Http, ftp, dns, etc. https://seclists.org/nmap-dev/2006/q2/207.

[65] S. Levi and N. Binyaminov. (2020) Top 10 web service exploits in 2019. https://www.radware.com/blog/security/2020/04/top-10-web-service-exploits-in-2019/.

[66] V. Inc. (2019) rdp-ntlm-info nse script. https://vulners.com/nmap/NMAP:RDP-NTLM-INFO.NSE.

[67] Dormando. (2024) memcached - a distributed memory object caching system. https://memcached.org/.

[68] J. Sermersheim, “Lightweight Directory Access Protocol (LDAP): The Protocol,” RFC 4511, Jun. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4511

[69] Censys. (2019) New! search censys for prometheus endpoints. https://censys.com/new-search-censys-for-prometheus-endpoints/.

[70] Elasticsearch. (2024) Elasticsearch platform — find real-time answers at scale — elastic. https://www.elastic.co/.

[71] A. CouchDB. (2024) Apache couchdb. https://couchdb.apache.org/.

[72] OWASP. (2021) A06 vulnerable and outdated components. https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/.

[73] turingpoint GmbH. (2024) Version disclosure — turingsecure. https://turingsecure.com/knowledge-base/issues/version-disclosure/.

[74] I. S. Corp. (2024) Cwe-205 web vulnerability & security checks — invicti. https://www.invicti.com/web-vulnerability-scanner/vulnerabilities/classification/cwe/.

[75] TheSmartScanner.com. (2024) Server version disclosure. https://www.thesmartscanner.com/vulnerability-list/server-version-disclosure.

[76] A. ZooKeeper. (2024) Welcome to apache zookeeper™. https://www.elastic.co/.

[77] nils ohlmeier. (2021) sipsak. https://github.com/nils-ohlmeier/sipsak.

[78] Flask. (2010) Welcome to flask — flask documentation (3.0.x). flask.palletsprojects.com. https://flask.palletsprojects.com/en/3.0.x/.

[79] Tcpdump.org. (2017) Tcpdump/libpcap public repository. https://www.tcpdump.org/.

Appendix

TABLE VI: Search syntax for search host 1.1.1.1:443 in different device search engines.

Engine	Syntax
Censys	(ip=”1.1.1.1”) and services.port=‘443‘
Shodan	ip:1.1.1.1 port:443
FOFA	ip=”1.1.1.1” && port=”443”
ZoomEye	ip:”1.1.1.1”+port:443

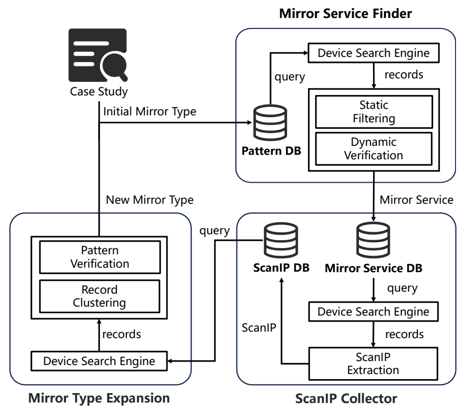


Fig. 8: Overview of *ScanIP* collection module.

A. ScanIP and Mirror Service Collection

Based on our preliminary study, we selected four engines that offer sufficient queries and batch-automatable search API capabilities: Censys, Shodan, FOFA, and ZoomEye. To mitigate the impact of *ScanIP* ownership changes and potential Mirror Service downtime, we limited our analysis to data records in device search engines collected between March 2023 and March 2024.

Candidate Mirror Service Collection. We initiated our approach by extracting search patterns for Mirror Service identified in the preliminary study (Section III-B). We used the text in the records which are unrelative with the variable factors (IP and environment variables) and the protocol type as its pattern, denoted as *MirrorPattern*. For instance, the pattern for Mirror Service reflecting *ScanIP* via the SIP protocol is defined as “protocol:sip && banner:received=”. Meanwhile, we designated the patterns that match the candidate *ScanIP* in the responses as *IPPattern*. For example, the *IPPattern* corresponding to the SIP protocol is “received=${ipv4}”. Then we defined the tuple <*MirrorPattern*, *IPPattern*> as a pattern capable of detecting Mirror Services and mining *ScanIPs*. We then converted *MirrorPattern* into the corresponding syntax of the device search engine and obtained matching host records by search APIs.

Mirror Service Verification. After filtering invalid IPs and those that were the same as the host’s IP, we used tools such as Nmap, Netcat, telnet, and sipsak[77], to probe the host and check if its response including our testing IP. If it does, we confirm it as a valid Mirror Service.

Mirror Service Type Expansion. We took the *ScanIPs* we have collected as seeds and queried the records of the three types of scanned IPs on device search engine. For instance, on ZoomEye, the query statement retrieving records whose responses contain the three variations of 1.2.3.4 is `banner: ``1.2.3.4'' banner: ``1%2E2%2E3%2E4'' banner: ``4.3.2.1''`.

Then, considering the records may come from various services, including those known Mirror Service types, we filtered out matching patterns of known Mirror Service types. For the remaining records, we utilized a clustering approach to extract new patterns for Mirror Service. Specifically, given that the device search engine records provide service tags, we initially categorized the records according to their services. Then we recognized that these records could contain variable information in digital formats, such as timestamps or service versions. This information could notably influence the clustering process, leading us to remove all numeric data from the response records. Subsequently, we computed the text similarity among the remaining records based on cosine distance and categorized those with a similarity score above 0.9 within the same service as responses originating from the same reason. In the end, we manually reviewed each category to identify potential new Mirror Service types. Leveraging domain knowledge, these are subsequently transformed into query statements compatible with various device search engines. Additionally, regular expressions to match *ScanIPs* were formulated and integrated into our pattern database.

B. Behavior monitoring

Using Flask [78], we developed an HTTP service in our web honeypot and integrated incorporated fingerprints of router device configuration pages, creating a low-interaction IoT honeypot. We extracted 443 fingerprints from devices identified by device search engines, and encompassed major router manufacturers such as TPLINK, DLINK, and Tenda. We also embedded our dynamic links in the default page, which is encoded by a unique visit record as ID, including visiting IP, port, honeypot IP, and timestamp.

In terms of paths, we set common web files including robots.txt, sitemap.xml, and security.txt. Within robots.txt, we defined paths of varying depths that are allowed for crawler access, deliberately including some forbidden paths related to known web vulnerabilities to observe whether device search engines would intentionally conduct scans. sitemap.xml provides directory paths of different depths to test the scanning depth and breadth of crawlers. In security.txt, we deliberately included a path in the contact section to observe whether crawlers would read this information. We also embed the unique IDs in the dir of the paths.

Our full-port closed honeypot and popular-port open honeypot are both sniffer honeypots, with the only difference being whether ports are open or not. It utilizes the tcpdump [79] to monitor traffic across all ports and combines Berkeley Packet Filter (BPF) syntax for precise packet filtering, enabling effective capture of specific types of traffic. To eliminate potential interference, we migrated the SSH port to a less commonly used service port and blocked traffic on it. This prevented unnecessary data interactions that could skew results. To passively acknowledge TCP packets without actively responding at the application layer, we use `nc -lk` and `nc -luk` to listen on TCP and UDP ports, respectively.

TABLE VII: Default scanning path of device search engines in web service scanning.

Path	Censys	Shodan	FOFA	ZoomEye
/	✓	✓	✓	✓
/robots.txt	✗	✓	✗	✓
/favicon.ico	✓	✓	✓	✓
/.well-known/security.txt	✗	✓	✗	✓
/sitemap.xml	✗	✓	✗	✗
/nice ports,/Trinity.txt.bak	✗	✗	✓	✗

TABLE VIII: Sensitive paths scanned by each device search engine. "Ratio" refers to the percentage of total scanning traffic conducted by the engine that targets the path.

Engine	Type	Path	Request Times	Ratio
Censys	Web(Prometheus)	/api/v1/label/goversion/values	26,242	1.45%
		/api/v1/label/goversion/values	26,242	1.45%
		/api/v1/query	26,195	1.45%
		/api/v1/labels	26,141	1.44%
		/api/v1/label/__name__/values	26,118	1.44%
		/api/v1/targets	25,648	1.42%
		/api/v1/label/version/values	25,619	1.42%
		/api/v1/status/config	13,015	0.72%
		/tr064dev.xml	4,927	0.27%
		/api/json	287	0.02%
Shodan	IoT(IP Camera)	/cgi-bin/authLogin.cgi	5,459	1.31%
		/filestation/wfm2Login.cgi	5,099	1.22%
		/photo	4,933	1.18%
		/video	4,878	1.17%
		/snapshot.cgi	750	0.18%
		/cgi-bin/viewer/video.jpg	528	0.13%
		/cgi-bin/snapshot.cgi	519	0.12%
		/snapshot.jpg	485	0.12%
		/tmpfs/auto.jpg	465	0.11%
		/cgi-bin/view/image	276	0.07%
		/axis-cgi/jpg/image.cgi	273	0.07%
		/ipcam/jpeg.cgi	272	0.07%
		/ISAPI/Streaming/channels/101/picture	268	0.06%
		/jpg/image.jpg	266	0.06%
		/Streaming/channels/1/picture	265	0.06%
		/Streaming/channels/101	261	0.06%
		/image/jpeg.cgi	258	0.06%
		/img/snapshot.cgi	253	0.06%
		/-wvhttp-01-/GetLiveImage	251	0.06%
		/-wvhttp-01-/GetOneShot	250	0.06%
		/videostream.cgi	223	0.05%
		/get_status.cgi	219	0.05%
		/videostream.asf	218	0.05%
		/cgi-bin/video_snapshot.cgi	217	0.05%
		/snap.jpg	212	0.05%
FOFA	Web(Elasticsearch)	/_cat/indices	199	0.23%
ZoomEye	IoT(OpenWrt Router)	/cgi-bin/luci/	3,059	4.89%
		/studio/index.html	895	1.43%

C. Web Scanning Strategy

From 2,503,761 requests our web honeypot captured, we observed that on average, Shodan sent 17.69 requests, Censys sent 60.25 requests, Fofa made 5.87 requests, and ZoomEye sent 3.07 requests per day to each honeypot, revealing distinct patterns. Their path access strategies also differed significantly. ZoomEye's *ScanIPs* occasionally appended an extra slash ('/') to paths, such as scanning both '/favicon.ico' and '/favicon.ico/', suggesting a URL normalization strategy to handle trailing slashes across different web servers.

We also observed that device search engines perform scanning in multiple phases. Shodan, for example, begins with a broad scan using the User-Agent "Mozilla/5.0 ... Safari/537.36" for default paths. Once it discovers information of interest, it launches a focused scan, switches to Python-based User-Agents (such as `` `python-requests/2.27.1'' `` and `` `python-requests/2.23.0'' ``) and scans multiple specific paths. This change in User-Agent suggests that Shodan potentially employs diverse strategies and techniques during the scanning process.

TABLE IX: The main User-Agents used by device search engines.

Engine	Path	User-Agent	Ratio
Censys	all	Mozilla/5.0 (compatible; CensysInspect/1.1; +https://about.Censys.io/)	69%
		-	31%
Shodan	/	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36	31%
	/favicon.ico	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36 Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:80.0) Gecko/20100101 Firefox/80.0	31%
	camera	python-requests	15%
	other	-	36%
FOFA	/favicon.ico	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112 Safari/537.36	17%
	other	-	81%
ZoomEye	all	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4 240.111 Safari/537.36	29%
		-	55%

TABLE X: The Multi-port identified protocols and their corresponding number of probe types and the list of ports. Since the number of ports corresponding to fallback probes is vast, their results are not displayed here.

Service	# of Types	Target Ports
Secure Shell	2	[22, 2222]
Network Basic Input/Output System	7	[25, 137, 139, 7587, 11382, 23915, 29844, 31530, 34125, 34303, 40013, 44893, 47415]
OpenVPN	2	[443, 500, 1194]
Socket Secure	4	[1080, 5555, 5678, 7777, 7788, 7890, 8888]
Microsoft SQL Server	5	[427,1433,1434,7025,10001,16592,20748,21429,22637,28864,31980,41372,51668,55010, 61870]
Mikrotik Router	7	[111,2000,4478,7215,8728,10151,23810,24285,27527,32400,38676,40454,41787,49122]
Session Initiation Protocol	6	[4871,5060,5061,6060,6788,8320,10325,10326,14396,16319,19867,25841,27650,31492, 34182,35042,37997,39510,39849,46321,46837,49699,50929,54023,58038]
NAT Port Mapping Protocol	6	[69,80,520,1812,1877,2869,3389,3600,3786,5351,5432,6340,6604,6969, 7001,7320,7398,8000,8290,8835,8945,9999,10690,11211,12205,16397,17180,23205, 23209,23627,24046,24588,24921,28348,30718,32626,34425,34664,35494,37834,40257,40891, 41145,41216,41407,45127,45567,46062,47868,51168,51261,53413,53878,54232,57385,58682, 59478,64738,64940,65501]
X Window System	4	[6000, 6001, 6002]
Redis	5	[6379, 6666, 7000]
Ubiquiti Discovery Protocol	4	[19,382,3745,4095,5094,9185,10001,11977,18798,19132,20004,22153,22834,24669, 27464,32157,32521,32889,34344,36712,38130,39396,39509,42481,44045,47395,51887]
Domain Name System	7	[53,69,174,1967,2967,5353,9646,10001,20104,21301,28159,29997,30855,32276, 37165,47268,48409]
Network Time Protocol	8	[123,1632,2112,9577,14983,23708,33270,36503,42507,51759,52315,53075,61172,65037]
X Display Manager Control Protocol	2	[69,177,1910,12816,13495,13636,14694,15330,15742,17790,25622,30397,32888,36997, 38792,40538,45197,47122,50647,59675]
Negotiation of NAT-Traversal in the IKE	1	[500,1194,1891,3997,4304,4500,6154,7928,8209,12390,12429,14973,16160,20969, 22993,24512,25270,26680,28200,31788,33172,34949,34956,38381,38538,40126,40224,40727, 42850,42910,44568,44806,45708,46061,49109,49147,51822,54015,59491,63038,63284,64367]
Routing Information Protocol	6	[520, 2222, 4301, 17948, 23103, 27305, 35315, 35405, 36333, 38527, 64648]
Universal Plug and Play	3	[1474, 1900, 16435, 21721, 24695, 32410, 32414, 37215, 38412, 38599, 45913, 56721]
Citrix MetaFrame application	2	[1604, 23168, 23261, 33352, 38205, 38890, 41912, 46508, 58206, 58344, 58686]
RADIUS	2	[1645, 1812, 6574, 16531, 20899, 26701, 29322, 48794, 52452, 54347]
Simple Object Access Protocol	3	[370, 2191, 3702, 8446, 21229, 35830, 56006]
Apple Remote Desktop	4	[3283, 9334, 13853, 14434, 17847, 43041, 47851, 52327, 55123, 56498, 62279, 63176]
A2S Query protocol	3	[4131,8626,12893,18745,21025,22767,24018,27015,27016,27105,28015,32165,41700,57896]
VxWorks Wind DeBug agents	3	[4210,12819,14567,14771,17185,18265,20379,26764,28940,31339,48717,49530,49661,51202, 57125,57175,57381,57609,62151,63735]
Datagram Transport Layer Security	2	[5061,5257,5684,5738,6625,7243,11920,19604,20374,20720,21406,28845,31436,31966, 33703,38765,39434,39783,50338,52540,52668,52685,53405,59168,63340]
DNS-Based Service Discovery	2	[5353, 18235, 18529, 24173, 24626, 25301, 26081, 29939, 45293, 62663, 65337]
Building Automation and Control Networks	2	[5407, 6833, 7642, 9140, 18427, 25337, 31513, 33728, 42168, 47808]
PC Anywhere	4	[5001, 5632, 10522, 31348, 39939, 41650, 42730, 50388, 57664]
Distributed hash table	3	[6881, 13001, 24530, 29579, 29899, 44629, 44633, 47199, 48097, 48688]
Simple Mail Transfer Protocol	2	[25, 587]
GPRS Tunneling Protocol	2	[2123, 2152, 3386]
Session Traversal Utilities for NAT	3	[3478, 8088, 37833]
Constrained Application Protocol	2	[5673, 5683]
Android Debug Bridge	2	[5555, 9001]
Java Remote Method Invocation	1	[6000, 10001]
Java Debug Wire Protocol	1	[8000, 9000]
All-Seeing Eye	1	[8000, 11211]