ScannerGrouper: A Generalizable and Effective Scanning Organization Identification System Toward the Open World

Xin He
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Enhuan Dong
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Jiyuan Han
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Zhiliang Wang
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Hui Zhang[†]
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Liang Liu
Beihang University
Beijing, China

Lianyi Sun
Beijing Institute of Technology
Beijing, China

Supei Zhang
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Pengfei Xue
National University of Defense Technology
Changsha, China

Guanglei Song
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Han Li
360 Technology Co.
Beijing, China

Xiaowen Quan
WebRAY Tech (Beijing) Co., Ltd., Tsinghua University
Beijing, China

Jiahai Yang
Institute for Network Sciences and Cyberspace, Tsinghua University
Beijing, China

Abstract

In recent years, many scanning organizations deploy large numbers of scanners to actively probe the Internet. Identifying the organizations behind these scanners is of significant value. The problem of analyzing the sources of scanners has been investigated in various studies. However, as far as we know, the problem of effectively and generally identifying scanner organizations in real-world scenarios remains unsolved.

In this paper, we present ScannerGrouper, a darknet-independent system specifically designed to identify the organizations behind Internet scanners in real-world scenarios. ScannerGrouper leverages monitoring systems capable of capturing service probes, e.g., honeypots, to collect traffic for subsequent analysis. To address the robustness challenge, ScannerGrouper selects features from the payloads of the first service probes sent by scanners through statistical analysis, and aggregates the identification results from multiple service-specific classifiers. To tackle the open-world issue, ScannerGrouper customizes a state-of-the-art open-set model to our specific task, and updates the system incrementally. We conduct extensive experiments to validate ScannerGrouper effectiveness. ScannerGrouper outperforms baseline solutions in identification performance, achieving a weighted average F1-score that is 1.63 to 4.05 times higher. We also experimentally analyze the identification results of unattributed scanners, training time, the performance of possible alternative models of the core module, and the impact of hyperparameters, etc.

CCS Concepts

• **Security and privacy → Network security**.

Keywords

Scanner, Scanning Organization Identification, Scanning Activities

ACM Reference Format:
Xin He, Enhuan Dong, Jiyuan Han, Zhiliang Wang, Hui Zhang, Liang Liu, Lianyi Sun, Supei Zhang, Pengfei Xue, Guanglei Song, Han Li, Xiaowen Quan, and Jiahai Yang. 2018. ScannerGrouper: A Generalizable and Effective Scanning Organization Identification System Toward the Open World. In *Proceedings of ACM Conference (CCS'25).* ACM, New York, NY, USA, 25 pages. https://doi.org/XXXXXXX.XXXXXXX

[†]Also with Quan Cheng Laboratory, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS'25, Taipei, Taiwan
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Recent measurement studies reveal a substantial increase in Internet scanning activities, with a significant portion originating from scanning organizations that have emerged in recent years [20, 36, 38, 56, 58, 59]. Benefiting from the advances in scanning techniques [25, 35, 36, 40, 41] and the broad accessibility of cloud server rental services, the scanning organizations are now capable of conducting frequent and large-scale scanning of public address spaces.

Scanning organizations normally utilize multiple scanners to distribute scanning tasks. Based on a comprehensive investigation of existing research, we define a scanner as an entity or program identified by public IP addresses that sends probes / probing packets to one or more remote hosts with the aims of (1) detecting hosts that respond to the packets and (2) extracting information from the response packets sent by the hosts. Nowadays, a significant number of scanners are actively scanning the Internet [20, 33, 34, 36, 38, 44, 56, 58, 59, 64, 68].

Identifying the organizations of the scanners holds substantial value [19, 33, 34, 44, 64, 68]: (1) Guiding Cybersecurity Defense: Identifying scanner-affiliated organizations enables defenders to implement targeted strategies that mitigate or prevent potential network attacks. [68] is the first to comprehensively reveal the ethical concerns associated with certain scanning organizations, including their unauthorized access, exposure of service vulnerability information, disclosure of private data, concealment of identities, etc. IP identification of these organizations enables inline security defenses. (2) Supporting In-Depth Analysis of Scanning Behaviors: Scanner organization information enables deeper behavioral analysis, helping identify their data collection focus and preferred targets such as specific IPs, ports, services, or software. Studies such as [19, 28, 33, 34, 44] show that scanning traffic reflects target interests, aiding attack surface analysis, especially when scanner affiliation is known. (3) Enabling Deep Analysis of Scanning Organization Identities: Identifying organizations of the scanners helps uncover identities and scale of potential attack organizations. For example, [33] reports the discovery of Shadowserver, previously unknown to the authors[1]; [68] further notes that certain payload fields can help infer scanning organizations' identities.

The issue of analyzing scanner sources has been explored in multiple works. *However, the problem of effectively and generally identifying scanner organizations in real-world scenarios remains unresolved.* Most works on identifying scanner sources are unable to accurately determine the organizations behind scanners [28, 30, 31, 36, 38, 48, 58, 59, 65]. They typically rely on preliminary methods—such as WHOIS queries, IP geolocation, and reverse DNS resolution—that identify scanner sources typically at a coarse granularity. These approaches fail to attribute scanners without domain names or presence in official scanner lists at the organizational level, which we term unattributed scanners.

The remaining studies seek to identify the sources of a greater number of unattributed scanners at the organizational level [19, 33, 34, 44, 64, 68]. However, they are often not generalizable, ineffective in real-world scenarios, or both. First, the solutions proposed in [19, 33, 34, 44] are based on darknet (e.g., UCSD's /8 [17]), which requires large blocks of unused, routable public IP addresses [15, 53]—a privilege held by only a few, especially for IPv4. An alternative is to use scanner attribution intelligence from external darknets (e.g., DomainTools [23]), but this approach is ineffective for the increasingly common localized scanning. Studies [38, 56, 58] show that such scanning often targets most IPs within specific routed prefixes (e.g., enterprise networks). Overall, darknet-based solutions lack generalizability and are ineffective for localized scanning. Second, in the real-world scenarios, existing measurement studies [20, 36] reported that the temporal characteristics of the ports scanned by scanners and the statistical characteristics of scanner activities exhibit significant variability. However, the solutions proposed in [19, 33, 34, 44] treat these characteristics as features for identifying scanner organizations, making these solutions less robust in real-world scenarios. Third, in real-world scenarios, the solutions presented in [33, 34, 44] are unable to assess the identification results of unattributed scanners and instead depend on manual evaluation of these classification results. Finally, certain solutions [19, 64, 68] exhibit limitations in their scope of applicability. In real-world scenarios involving many diverse scanning organizations, they fail to demonstrate consistent effectiveness. The solution in [64] infers the organizations behind Modbus-targeting scanners based on honeypot traffic, extracting features specific to the Modbus protocol. However, it lacks a generalizable approach for handling other service types. The solution in [19] is the earliest to utilize machine learning for organizational-level scanner attribution on darknet traffic, but it only identified Censys scanners. It was later found to fail to converge on larger datasets with more classes [33]. The solution in [68] innovatively identifies scanner IPs by analyzing the search results of cyberspace search engines. However, its effectiveness depends on the availability of these search engines that provide sufficient query quotas and support batch-accessible search APIs, which are rare.

The primary limitation of previous studies in generalizability stems from their reliance on traffic collected from darknets. In this paper, we aim to design and implement a darknet-independent system that effectively identifies scanning organizations in real-world scenarios. The monitoring systems considered as alternatives to darknets include honeypots and network intrusion detection systems (NIDS), which can be deployed within targeted networks. As previously discussed, localized scanning typically involves probing most or all IP addresses within specific routed prefixes (e.g., enterprise networks). Given the limited IP addresses within these prefixes, honeypots or NIDS in these networks are likely to be hit. This darknet-independent approach lowers the barrier to identifying scanning organizations and facilitates the analysis of localized scanning activities.

However, it is challenging to design and implement such a system. First, it must be sufficiently robust. As mentioned previously, a major limitation of many existing works [19, 33, 34, 44] is the high variability of the features upon which they rely. Ensuring robustness of the entire system against dynamic scanner behaviors necessitates a thorough understanding of scanning activities and careful system design. Second, in real-world scenarios, identifying scanning organizations involves the open-world challenge [16, 32]; that is, there are unattributed scanners belonging to unknown scanning organizations, and the entire system needs incremental update

[1]This paper similarly uncovers a previously unknown organization: Palo Alto.

capabilities to continuously adapt to the possible changes in scanner features and to minimize the time required for each update. This requires the system to accurately and automatically identify the scanning organizations of unattributed scanners, regardless of whether they belong to known or unknown organizations, i.e., achieving open-set identification[16], while also supporting incremental updates. These are also the limitations of some of the existing works discussed earlier [33, 34, 44].

To address these challenges, we develop ScannerGrouper, the first darknet-independent system for identifying the organizations behind Internet scanners, designed to be generalizable, effective, and suitable for open-world scenarios. Our key ideas are: (1) to strength the system robustness, we conduct a statistical analysis of the scanning traffic collected by honeypots to select stable features capable of distinguishing scanning organizations, and then we improve the identification method by integrating the results derived from multiple classifiers; (2) to address the open-world challenge, we introduce modifications of a state-of-the-art (SOTA) open-set model tailored to our scanning organization identification problem, and we propose an incremental update method for the whole system.

To achieve our ideas, we carefully propose several novel methods and realize them into multiple modules of ScannerGrouper. First, based on our statistical analysis (§3.3.1), we select features from the payloads of the first service probe in each session sent by scanners. The analysis shows that these payloads are effective in distinguishing scanning organizations. Notably, to the best of our knowledge, no existing measurement studies have identified substantial variability in the payloads of service probes from the same organizations. Therefore, we select the payload fields of these first service probes as features.

Second, we propose a method to aggregate identification results from multiple service-specific classifiers. Our statistical analysis reveals that payloads of the first service probes in each session from different organizations may appear similar within a given service (e.g., HTTP) but exhibit significant differences across other services (e.g., TLS). To address this, a dedicated classifier is trained for each service, and the outputs of these classifiers are integrated via probe-level voting and ensemble learning.

Third, we tailor a SOTA image-based open-set model, IOMatch [46], to our task by designing a novel feature semantic encoding and refining the training process. We initially considered non-image open-set models but found their performance inadequate. Since recent progress in open-set recognition is mainly image-based, we adopt IOMatch for each service-specific classifier. Our encoding transforms payload semantics into image format, significantly reducing the number of distinct encoded values while preserving semantic meaning. We also enhance the training process to avoid manual evaluation. Instead of manually assessing unattributed scanner results [33, 34, 44], we treat probes from organizations with fully disclosed scanner IPs as unknown-class samples. This allows their classification outcomes to serve as indicators of the model performance on unattributed scanners.

Fourth, we incrementally update ScannerGrouper at regular intervals, such as weekly, to ensure adaptability to potential changes in scanner payload features and reduce update time. Inspired by [18, 21] and considering practical model update efficiency, we design dedicated update methods for both the IOMatch models and the ensemble learning model, enabling them to resume training with new datasets at the start of each time period. Additionally, we develop a method to dynamically extend the image encoding format during updates, ensuring compatibility with evolving data characteristics.

We implement ScannerGrouper and conduct comprehensive experiments to evaluate its effectiveness. Compared to several baseline approaches [33, 34, 44], ScannerGrouper achieves a weighted average F1-score that is 1.63 to 4.05 times higher. The incremental version of ScannerGrouper improves steadily through weekly updates, eventually approaching the performance of a one-time full-data training. Training time is acceptable. Through analysis of unknown-class scanner results, ScannerGrouper supports identifying known-class scanners among unattributed ones, discovering new scanning organizations via payload inspection, and enhancing analysts' understanding of emerging scanning activities for rapid defense response. Additionally, we compare IOMatch with Non-image open-set models. Results show that IOMatch significantly outperforms these baselines, validating the effectiveness of image-based payload encoding and its strong capability in learning payload features. Finally, we analyze the impact of hyperparameters and unknown-class scanner ratio on ScannerGrouper 's performance.

Our major contributions are summarized as follows:

- We propose ScannerGrouper, the first darknet-independent system—to the best of our knowledge—for identifying the organizations behind Internet scanners, generalizable and effective in open-world scenarios (§3).
- To enhance system robustness, we design modules (§§ 3.3 and 3.6) that extract features from the payloads of first service probes to distinguish scanning organizations and aggregate identification results from service-specific classifiers using ensemble learning.
- To address the open-world challenge, we devise modules (§§ 3.4 and 3.5) to adapt a SOTA open-set model to our problem, and develop an incremental system update module (§3.7).
- We conduct extensive experiments (§4) to demonstrate the performance of ScannerGrouper on multiple aspects.
- We open source our artifacts (e.g. source code, datasets) at https://github.com/lemonhx25/scannergrouper.

2 Background

Emerging Scanning Organizations: In recent years, scanning activities on the Internet have seen a significant rise [28, 36, 56, 58, 67]. Scanning can gather a wide range of critical information, such as active hosts, open ports, service software versions, and operating systems, etc [12, 25, 26, 35, 40, 49, 50]. The exposure of these information could represent a significant risk to cybersecurity. For example, scanning may disclose the version of a service's software on a target host. By correlating this information with a publicly accessible vulnerability database, it is possible to identify potential software vulnerabilities.

Approximately two decades ago, studies concentrated on detecting and preventing network scanning, as such activities frequently preceded cyberattacks. In recent years, however, the objectives and implications of scanning have become more intricate. Unlike earlier scanning organizations, some newer ones openly disclose their

scanning objectives on public websites. We term these as *acknowledged* scanning organizations[2], e.g., Censys [2, 27] and Shodan [9]. The cybersecurity community is increasingly appreciating the significance of the scanning from those acknowledged scanning organizations. Some of their users leverage the scanning results to evaluate the security posture of their network assets or to perform penetration testing, while others, such as researchers, employ them for large-scale cybersecurity analyses [27, 39, 42, 57, 62].

In addition to acknowledged scanning organizations, a substantial number of *unacknowledged* ones—those that do not disclose their objectives on any platforms—still engage in scanning activities. In contrast to *acknowledged* scanning organizations, unacknowledged ones are more traditional, which means they are more likely to exhibit malicious intent, with the scanned information potentially being used to facilitate subsequent attacks.

The scanning organizations typically employ multiple scanners, and the scanners normally send multiple types of probes / probing packets to the targets. In general, the probes typically may consist of ICMP probes for active host discovery, TCP SYN probes for open port scanning, and service probes to gather target service-related information, etc. The definition of a scanner was provided in §1. It is important to note that, according to our definition, some attackers may integrate scanners' behaviors into their broader activities; for example, they may attempt to log in after discovering an open SSH port (22) [29, 54]. In scanner-related research, these attackers are generally considered scanners and are not excluded from analysis.

Current Scanning Activities: Existing measurement studies have extensively characterized current scanning activities on the Internet. These studies reveal that the Internet scanning activities are highly complex, posing significant challenges for their analysis.

(1) All studies [20, 36, 38, 56, 58, 59, 68] that measure and analyze scanning traffic consistently reveal that the majority of scanners are unattributed and dynamic. Only a small number of scanners possess reverse DNS records that clearly identify their affiliated organizations, and, to the best of our knowledge, the sources of reliable scanner lists are limited to the official websites of only three acknowledged scanning organizations, i.e., Censys [2], Rapid7 [7], and Driftnet [3]. Moreover, the IP address set of the observed scanners changes dynamically over time.

(2) Recent studies [38, 56, 58] reveal that localized scanning is popular and differs from Internet-wide scanning in several aspects, such as the target ports and the number of probes sent by the scanners. Most existing works [19, 33, 34, 44] on identifying scanner organizations relies on darknets to collect scanning traffic. However, deploying darknets for localized scanning traffic is impractical due to the scarcity of public IP resources and the difficulty of allocating large IP blocks from operational networks.

(3) Griffioen et al. [36] and Collins [20] reported that the temporal characteristics of the ports scanned by scanners and the statistical characteristics of scanner activities are very dynamic. The scanner behaviors exhibit significant week-to-week fluctuations and are increasingly influenced by short-term trends. The number of ports targeted by scanners has increased over time, accompanied by a growing prevalence of vertical scanning behavior, where multiple ports on the same target IP are probed by one single scanner.

3 System Design

This section first outlines the design of ScannerGrouper, then details each component module.

3.1 Overview

The Architecture and Workflow: Fig. 1 shows the architecture and workflow of ScannerGrouper. ScannerGrouper consists of six modules: Traffic Collection Module, Feature Selection Module, Feature Encoding Module, Organization Identification Module, Result Aggregation Module, and System Update Module.

The Traffic Collection Module may include various monitoring systems, such as low-interaction honeypots, NIDS capable of filtering scanning traffic, reactive network telescopes[3] [38], and interactive cloud-based telescopes [56]. Any monitoring system capable of capturing service probes can be integrated. This study utilizes data from multiple low-interaction honeypots deployed by us, each exposing several common services. Additionally, two partner companies contributed supplementary honeypot data. A preliminary analysis of the datasets reveals that service-probing traffic associated with HTTP, TLS, and DNS encompasses most scanners from acknowledged organizations, warranting particular attention. Using the collected traffic data, the Feature Selection Module conducts an in-depth statistical analysis of each payload field within the first service probes for the HTTP, TLS, and DNS protocols. Based on this analysis, several payload fields are selected as features for identifying scanner organizations, utilizing a typical feature selection method.

Next, the Feature Encoding Module transforms the selected fields of each service probe into an image using our proposed payload semantic encoding method. These images serve as input to the Organization Identification Module, which trains an IOMatch model [46] for each service. The IOMatch model, designed as an open-set model, identifies the scanner's organization for each service probe and accommodates unknown organizations as part of its identification outcomes. The identification results are then passed to the Result Aggregation Module, which uses probe-level voting and ensemble learning to produce scanner-level results, subsequently stored in the scanner database.

When ScannerGrouper requires an update, the System Update Module builds a new training set using data from the Traffic Collection Module and updated ground truth, then updates the IOMatch models, the ensemble learning models, and the image encoding format across relevant modules.

Addressing the Challenges: The Feature Selection Module and the Result Aggregation Module are designed to solve *the robustness challenge* mentioned in §1. The statistical analysis conducted within the Feature Selection Module demonstrates that the payloads of the initial service probes in each session can effectively differentiate scanners from distinct organizations. However, our findings also reveal that various fields within the payloads exhibit differing levels

[2]We adopt the term "acknowledged" from [20] and use "known" interchangeably in this paper.

[3]For localized darknets, we suggest deploying Spoki-like darknets [38] to support ScannerGrouper, which can complete the TCP handshakes and capture the first service probes sent by scanners to the darknet after the handshakes [24].

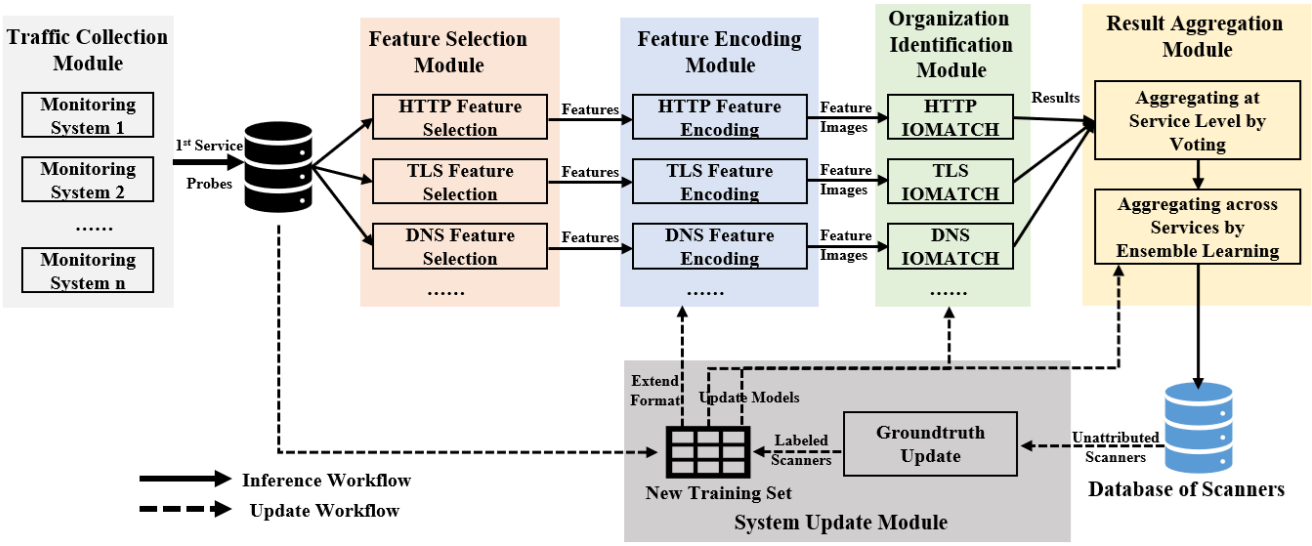


Figure 1: The architecture and workflow of ScannerGrouper.

of effectiveness in identifying the scanner organizations. Therefore, we select the fields for feature extraction from the initial probe of each service using the classic mutual information-based feature selection method [60]. Furthermore, the statistical analysis also shows probe payloads originating from different organizations may appear similar under certain services (e.g., HTTP) but display significant variation under others (e.g., TLS). To address this, in the Result Aggregation Module, we combine the classification results from classifiers trained on the probes across different services with probe-level voting and an ensemble learning method devised by us.

The Feature Encoding Module, Organization Identification Module and System Update Module are proposed to address *the open-world challenge* presented in §1. The challenge introduces two key requirements. First, ScannerGrouper must be capable of identifying unknown organizations of scanners, i.e., solving the open-set identification problem. Second, ScannerGrouper must support incremental updates. We first explored non-image-based open-set models but found their performance insufficient for our needs. Given that recent advances in open-set recognition are largely driven by image-based methods, we adapt the SOTA model IOMatch—originally designed for image data—to our task. To support its input requirements, we design a Feature Encoding Module to encode payload fields into image format. Initially, we try label encoding; however, we find it to be inefficient in representing semantic information and prone to generating an excessive number of encoded values. Then, we propose a novel payload semantic encoding method. Given that our dataset originates from real Internet traffic and includes probes sent by real, unattributed scanners, the inclusion of these unknown-class samples in the training process requires manual evaluation of the classification results, as is common in existing solutions [33, 34, 44]. To avoid manual evaluation, in the Organization Identification Module, we propose treating samples associated with organizations that have disclosed all scanner IPs as unknown-class samples during training, allowing for the automatic evaluation of the trained model's performance on unattributed scanners. In the System Update Module, we design efficient incremental update methods for ScannerGrouper, including model updates—drawing inspiration from [18, 21] while accounting for practical update efficiency—and dynamic extension of the image encoding format to accommodate evolving data.

3.2 Traffic Collection Module

This module collects traffic via monitoring systems capable of capturing service probes, including honeypots, NIDS, and interactive cloud-based telescopes like DScope [56]. Honeypots and DScope provide better generalizability than darknets in the cloud era, while honeypots and NIDS can be deployed in protected networks to monitor localized scanning activities effectively.

In this paper, we deploy low-interaction honeypots to capture service probes, focusing on five common services—HTTP (80), TLS (443), DNS (53), SSH (22), and RDP (3389)—on their standard ports. To ensure geographic diversity, one honeypot is deployed on each of five continents. In addition to our own honeypots, we incorporate data from two enterprises—Company A and Company B—both using low-interaction honeypots. Company A's honeypots support over 40 services, while Company B's support over 20. These services all open standard ports. We list the top 10 services based on the number of corresponding service scanners. For more information, please see Appendix B.

Table 1: Outline of Honeypot Datasets

Dataset	# of Honeypots	Format	Collection Time
SelfDeploy-24	5	PCAP	2024.02.01-2024.04.30
SelfDeploy-25	5	PCAP	2024.12.10-2025.03.17
SelfDeploy-25-S	5	PCAP	2025.03.18-2025.06.30
CompanyA	2	PCAP	2023.10.01-2023.10.26
CompanyB-22	5	CSV	2022.10
CompanyB-23	5	CSV	2023.10

Tab. 1 outlines the datasets. PCAP files capture all IP packets, while CSV files include only packets with payload, omitting ones like TCP SYN. The format depends on the honeypot deployment. Company B records only payload-carrying packets. A preliminary

analysis (§§ 3.2.1 and 3.2.2) shows that most scanners of acknowledged organizations primarily send HTTP, TLS, or DNS probes.

3.2.1 Scanner IP Labeling. To support our preliminary analysis and model training in subsequent sections, we label each probe with its scanner's affiliated organization using three sources. Probes unmatched by any method are labeled as unknown. In total, scanners from 17 known organizations are labeled. The labeling process is designed for easy automation.

- **Official websites of three acknowledged scanning organizations:** Censys [2], Rapid7 [7], and Driftnet [3] publicly disclose the IP addresses of all their scanners.
- **Reverse DNS resolution:** Using the public reverse DNS database [6], we resolve source IPs to domain names and label probes based on the organization names in those domains. While some scanning organizations may claim to configure PTR records for all their scanners, in reality only a subset of scanners have them, and such claims are difficult to verify. Moreover, reverse DNS resolution may yield only coarse-grained results—e.g., *.amazonaws.com points to AWS but not the specific scanning organization—making it unsuitable for labeling.
- **Our labeling method for cyberspace search engine scanners:** Cyberspace Search Engines are an important type of scanning organization. We propose a precise labeling method by crafting custom honeypot responses and correlating them with public Search Engine data. To support this, we additionally deploy three dedicated HTTP honeypots. Details are in Appendix A. The identification results are presented in Tab. 2. A similar approach appears in [68], but our encrypted response fields offer greater stealth.

Table 2: The identification results of our proposed method.

Cyberspace search engine	# of probes captured by the honeypots	# of scanners
censys	45	40
zoomeye	52	7
fofa	33	22
shodan	30	16
quake	4	3
x.threatbook	5	5
onyphe	3	3
hunter.qianxin	1	1

These three labeling methods were carefully selected for their reliability. While we also considered the approaches in [20] and [36], they were ultimately excluded due to concerns about their accuracy. Specifically, the list in [20] lacks thorough validation, and the intelligence provided in [36] is not transparent. It is worth noting that the selected labeling methods do not involve differentiating between known and unknown scanning traffic or isolating background traffic based on collected traffic.

3.2.2 Service Selection. To identify unattributed scanners potentially associated with known scanning organizations, we analyze the number of scanners and the services most frequently probed from known organizations, as summarized in the Tab. 3.

Over 85% of scanners from known organizations probe HTTP, TLS, or DNS. Therefore, in this paper, we focus on the payloads of these service probes to classify scanners from known organizations and identify unattributed scanners potentially linked to them.

Table 3: A preliminary analysis of the datasets.

Dataset	# of Service Probes	# of Scanners	# of Scanners from Known Orgs.	# of Scanners from Known Orgs. that Conducted HTTP, TLS, or DNS Service Probing
SelfDeploy-24	3,421,629	44,204	3,136	2,918 (93%)
SelfDeploy-25	3,724,696	47,498	5,296	4,555 (86%)
SelfDeploy-25-S	3,040,194	45,123	5,283	4,647 (88%)
CompanyA	11,659,368	29,752	2,005	1,939 (97%)
CompanyB-22	62,248,213	299,399	1,271	1,259 (99%)
CompanyB-23	78,234,409	497,134	2,538	2,488 (98%)

3.3 Feature Selection Module

This module presents an in-depth statistical analysis of each payload field in initial HTTP, TLS, and DNS service probes across individual sessions. We make two key observations: (1) Initial probe payloads are effective in distinguishing scanning organizations. To the best of our knowledge, no existing measurement studies report significant variability in the payloads of probes originating from the same organization, suggesting that the features derived from these initial probes are sufficiently stable. (2) While probes may appear similar across organizations within one service, they differ significantly across other services. Based on this, we select key fields from each service for feature extraction.

ScannerGrouper is an incremental solution, and it supports system incremental updates for each time period. During the initial usage of ScannerGrouper, the service field selection is initialized based on the method in this subsection. Subsequent updates do not alter the selected fields.

Though this paper uses low-interaction honeypot traffic as input, ScannerGrouper is not limited to this source. Any system that captures initial service probes—such as NIDS or cloud-based telescopes like DScope [56]—can be used. As long as first probes are available, ScannerGrouper remains applicable.

3.3.1 Statistical Analysis of the 1st Service Probes. We parse the initial service probes targeting the three services and analyze the payload fields by scanning organization, using the SelfDeploy-24 dataset as a representative example.

The statistical analysis of the HTTP User-Agent field is presented in Tab. 4. Organizations like quake, internettl, onyphe, internet_census, fofa, stretchoid, and criminalip can be effectively distinguished by this field, as nearly 90% of their probes share unique, consistent values.

The first service probes of TLS and DNS services also contain similar distinguishing fields. Appendix C presents the analysis of TLS cipher suites and DNS query names. TLS handshake cipher suites help differentiate organizations like quake, internettl, onyphe, ipip, shodan, driftnet, and intrinsec, while DNS query names distinguish shadowserver, cybergreen, censys, shodan, and tum.

Additionally, we observe that probe payloads from different organizations may be similar for one service but show significant differences in others. For example, shodan and intrinsec largely overlap in the HTTP User-Agent field—59.7% of shodan's probes and 100% of intrinsec's use the same User-Agent string. However, they can be distinguished using the TLS handshake cipher suites field: shodan's probes include 57 cipher suites, while intrinsec's include 109. Thus, TLS data enables effective differentiation between them.

3.3.2 Key Field Selection. HTTP and TLS contain many payload fields (e.g., dozens in HTTP), and using all of them may introduce

Table 4: Statistical Analysis of User-Agent Field of HTTP Service Probes

Org.	User-Agent	Ratio
quake	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36 Edg/120.0.0.0	100.0%
internettl	masscan/1.3 (https://github.com/robertdavidgraham/masscan)	100.0%
shadowserver	N/A	50.8%
onyphe	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0	100.0%
internet_census	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36	91.8%
fofa	Mozilla/5.0	96.3%
ipip	Mozilla/5.0 (Windows NT 10.0; WOW64)	34.2%
binaryedge	N/A	85.6%
shodan	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36	59.7%
driftnet	N/A	52.0%
intrinsec	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36	100.0%
stretchoid	Mozilla/5.0 zgrab/0.x	100.0%
criminalip	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190 Safari/537.36	100.0%

irrelevant features that hinder accurate scanner organization identification. To address this, we apply the mutual information (MI) algorithm [60] to select the most informative fields through the following steps:

- **MI Calculation:** Each field is encoded using label encoding, and the MI value between each field and the target variable, i.e., the organization labeled in §3.2.1, is calculated.
- **Field Ranking:** Fields are ranked based on their MI values. A higher MI value indicates that a field is more important for the classification task related to the target labels.
- **Field Selection:** Based on the ranking, the top n fields with the highest MI values are selected as features.

Using this method, we find that the top 6 fields in each dataset have significantly higher MI values, so we select them as features for both HTTP and TLS probes. Details are provided in Appendix D. ScannerGrouper supports incremental updates: the Feature Selection Module selects fields only during the initial training period, and the same field types are retained in subsequent updates. The fixed field selection is effective in practice, representing the best practices observed in our datasets. For both HTTP and TLS services, the top-6 fields identified in the first week remain unchanged in subsequent weeks (see Appendix E).

Compared to HTTP and TLS, the DNS probe contains fewer fields. Only the query name is variable in length (up to 255 bytes); others are fixed. We use all DNS fields and truncate the query name to the first 32 bytes for feature selection.

3.4 Feature Encoding Module

To adapt the IOMatch model, we encode the field features selected by the previous module into image format. We initially consider using the typical label encoding method. However, we find that label encoding is inefficient at representing semantic information. Moreover, when applied to fields with numerous option combinations (e.g., the HTTP User-Agent field), label encoding treats each unique combination as an independent value, resulting in an overwhelming number of encoded values. An illustrative example is provided in Appendix F.1.

To address these issues, we propose a semantic encoding method tailored to three field types: option, string, and integer fields. The issue mainly lies in option fields, which may contain multiple values—such as in HTTP User-Agent or TLS Hash Algorithms. For each, we enumerate all possible values and assign each to a pixel in the encoded image, setting it to 255 if present in the probe, and 0 otherwise. Detailed encoding methods are listed in Appendix F.2.

We encode the selected fields of each service probe payload as an image. To conform to conventional image dimensions, we define the image as a square with dimensions of 32×32 pixels. Fields are arranged row by row, left to right, with each placed in a predefined position. An example of HTTP probe encoding is shown in Appendix F.3. To accommodate the potentially large number of options in option-based fields, we design an encoding strategy that ensures the image size remains within 32×32 pixels, as detailed in Appendix F.4.

3.5 Organization Identification Module

In real-world scenarios, some scanners originate from unknown scanning organizations. Therefore, the Organization Identification Module must not only identify scanners as belonging to specific known organizations, but also possess the capability to recognize scanners from unknown ones. This problem essentially falls under the category of open-set identification. To address this issue, we employ IOMatch [46], an open-set recognition method that has achieved SOTA performance in the field of image recognition.

We initially explored non-image open-set models but found their performance lacking. As recent advances in open-set recognition are primarily in image recognition, we adopt IOMatch. Performance comparisons between IOMatch and non-image open-set models are provided in §4.5.

We train a separate IOMatch model for each service to learn features from the 1st probe's payload. Each sample in IOMatch corresponds to an image from the above module. The advantage of IOMatch lies in its ability to leverage both known-class and unknown-class samples for semi-supervised learning. Details are in Appendix G. The classification results of IOMatch consist of $K + 1$ classes, where the first K classes correspond to known classes, and the $(K + 1)$-th class represents the unknown class.

Our dataset, derived from real Internet traffic, includes probes from unattributed scanners. Typically, handling unknown-class samples requires manual evaluation, as seen in existing solutions [33, 34, 44]. To automate this, we treat samples from organizations [2, 3, 7] that disclose all scanner IPs as unknown-class during training. The identification results for these samples can then be used to evaluate the model's performance on unattributed scanners.

3.6 Result Aggregation Module

As shown in §3.3.1, probe payloads from different organizations may appear similar in one service but differ in others. To achieve accurate scanner-level identification, we aggregate identification results across multiple services within each time period.

The IOMatch models in §3.5 produce probe-level identification results. We group them by scanner IP (ip_i) and service type (s_j) within the same time period (W_t), creating sets $R^{W_t}_{ip_i,s_j}$. For multiple probes from the same IP and service, we use a voting mechanism: the organization with the highest vote proportion is selected as the final result: $r^{W_t}_{ip_i,s_j} = \arg\max_{org}(count(org \in R^{W_t}_{ip_i,s_j}))$.

We then use multiple logistic regression models for ensemble learning to approximate a weighted voting model that aggregates service-level identification results. The scanner-level result $r^{W_t}_{ip_i}$ is obtained by aggregating $r^{W_t}_{ip_i,s_j}$ across all service types ($j = 1, \ldots, J$).

Logistic regression estimates the probability P of an event based on independent variables X, using the logistic (Sigmoid) function. We extend this approach to our multi-class classification task by building K logistic regression models, one for each known organization class, to aggregate service-level identification results.

For the k-th model, the output is: $P_k = \sigma(w_k^T X + b_k)$, where σ is the Sigmoid function, and w_k and b_k are the weight vector and bias for the k-th model. The independent variable X consists of service-level results for the same IP within time period W_t, with each element $x_i = r^{W_t}_{ip_i,s_1}, \ldots, r^{W_t}_{ip_i,s_J}$, one-hot encoded with $K+1$ binary values, with only one value set to 1. The position of the 1 indicates the index corresponding to the identified organization. Therefore, the weight vector for each model has dimensions $J \times (K+1)$, and $K+1$ indicates K known classes and 1 unknown class. The weights reflect the model's confidence in identifying the organizational class for each service.

Due to the lack of accurate labels for unknown-class samples, training a dedicated model is impractical. Instead, we propose a two-case rule using K logistic regression models for known classes: if all models output probabilities below threshold θ, the sample is classified as unknown; otherwise, it is assigned to the class with the highest probability above θ. The rule is:

$$r^{W_t}_{ip_i} = \begin{cases} \text{unknown}, & \text{if } \forall k \in \{1, 2, \ldots, K\}, P_k < \theta, \\ \arg\max_k P_k, & \text{otherwise.} \end{cases} \quad (1)$$

Model Training and Threshold θ Selection: We train the logistic regression models using the One-Versus-Rest (OVR) method [10], treating scanner IPs of the target organization as positive samples and others as negative. Service-level identification results serve as features, and model weights are learned using cross-entropy loss. The threshold θ is automatically selected via equidistant search on the training set. Specifically, we enumerate possible thresholds from 0 to 1 with a fixed step size (we set the step size as 0.01 to accurately get an optimal threshold). The goal is to find the threshold that maximizes the scanner-level average F1-score. Then we apply the learned threshold on the testing set, balancing the performance of identifying known and unknown classes.

3.7 System Update Module

ScannerGrouper supports incremental updates for each time period. In this study, ScannerGrouper considers weekly system updates. In the initial phase, service field selection is initialized using the dataset and labels from the first week to identify the fields used as features. Once these fields are selected, they remain unchanged. Subsequent updates only expand the image encoding based on these fields and update the IOMatch and logistic regression models.

At the end of each time period, i.e., each week, ScannerGrouper processes new data (X_t) and updates the ground truth labels (L_t) for scanner organizations, as described in §3.2.1. In addition to the three labeling methods, threat intelligence from security vendors can also be used. Three components are updated sequentially:

- **Image Encoding Expansion:** Encoding is extended after the first week's dataset to handle new options in the HTTP and TLS fields. Starting with the top six critical fields selected from the first week's dataset, each field is encoded and filled into its designated position in the image, as shown in Appendix F.3. When new data X_t arrives, newly observed option values are identified and appended to the end of the image.
- **IOMatch Model Update**: Inspired by Section 2.5 in [18] and Section 3 in [21], we design an incremental update method for IOMatch. At the end of each training period, model parameters are saved to checkpoint files. During the next period, ScannerGrouper loads the previous parameters, then uses the current data X_t and labels L_t to update the model.
- **Logistic Regression Model Update**: Since training logistic regression models is time-efficient, we incorporate the service-level identification results from the current week with those from previous weeks, along with the updated labels, to retrain the models.

4 System Evaluation

We first describe the experimental setup in §4.1. We then evaluate ScannerGrouper in the following aspects:

- **Identification Performance (§4.2).** Compared to baseline solutions, ScannerGrouper achieves a weighted average F1-score that is 1.63 to 4.05 times higher. The incremental design can increase performance through weekly updates, eventually achieving performance close to one-time training on the entire dataset.
- **Identification Result Analysis of Unknown-Class Scanners (§4.3).** ScannerGrouper enables identifying known-class scanners among unattributed ones, discovering new scanning organizations through payload analysis, and enhancing analysts' understanding of unknown scanning activities to support rapid defense strategies.
- **Training Time (§4.4).** ScannerGrouper shows an acceptable training time for both weekly updates and one-time training.
- **IOMatch vs Non-image Open-set Models (§4.5).** Experimental results show that the IOMatch model significantly outperforms Non-image open-set models, highlighting the effectiveness of image-based payload encoding and demonstrating IOMatch's capability to learn meaningful payload features.
- **ScannerGrouper Hyperparameter Analysis (§4.6).** We assess the impact of varying update frequencies (3-day to 10-day intervals) on the performance of incremental learning. We also

analyze IOMatch's performance under different hyperparameters and examine the impact of the threshold θ used for identifying unknown-class scanners in the Result Aggregation Module.
- **Impact of Unknown-Class Scanner Ratio on Identification Performance (§4.7).** We evaluate the robustness of ScannerGrouper against varying proportions of unknown-class scanners.

4.1 Experimental Setup

Implementation: Using machine learning toolkit including sklearn and pytorch, we implement multiple versions of ScannerGrouper for different use cases. (1) ScannerGrouper-i denotes the incremental version, in which the dataset is periodically collected, and the system is updated at regular intervals, specifically weekly in the evaluation, and ScannerGrouper-i is the version we recommend to use in real-world scenarios. (2) ScannerGrouper-f is a one-time training version, which utilizes the entire training set in a single training session. This version is to assess the performance ceiling of our solution and facilitate comparisons with non-incremental baseline works. ScannerGrouper-f is also suitable for scenarios where do not require incremental training.

Datasets: The datasets we adopt include (1) SelfDeploy, (2) CompanyA, and (3) CompanyB, which are briefly introduced in Tab. 1. The SelfDeploy datasets are used to evaluate ScannerGrouper in incremental training experiments due to their longest traffic collection duration. All datasets are utilized to assess the performance under one-time training.

Training: For ScannerGrouper-f, we split the datasets by time in an 8:2 ratio for training and testing. For instance, for the SelfDeploy-24 dataset, the corresponding period of the training set is 02.01–04.12 and testing is 04.13–04.30. Taking the SelfDeploy datasets as an example, we analyze the number of scanners per organization in the training and testing sets. As shown in Appendix H, the testing set effectively represents the overall scanner distribution.

For ScannerGrouper-i, training and testing proceed weekly. At the end of each week, the system automatically collects that week's data and labels, then updates itself as described in §3.7. After an update (taking about one minute, see §4.4), the system identifies probes from the following week, treated as the test phase. Probe-level results are accumulated and aggregated at week's end (§3.6) to produce scanner-level results, which are used to evaluate weekly identification accuracy. In ScannerGrouper-i, the Feature Selection Module determines the selected fields for HTTP and TLS services exclusively during the training process of the first week. In subsequent updates, the types of selected fields remain fixed.

For both versions of ScannerGrouper, as mentioned in §3.5, we treat the samples associated with organizations that have disclosed all scanner IPs, including censys and driftnet[4], as the unknown class samples during training. Moreover, we primarily follow recommended hyperparameter settings—such as the learning rate for IOMatch—and adopt default values suggested by toolkits like scikit-learn and PyTorch. Detailed hyperparameter usage is provided in Appendix I. For each version of ScannerGrouper, we apply consistent hyperparameters across all relevant datasets.

[4]We do not add Rapid7 to the unknown class, because Rapid7 sends very few packets in all datasets.

Metrics: (1) Classification performance: Besides the K known class, we further evaluate the performance of the unknown class (the $(K+1)$-th class) by analyzing the results of censys and driftnet samples. For each class, we evaluate three metrics: Precision = $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$, and F1-score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$, where TP is true positives, FP is false positives, and FN is false negatives. (2) Model training time: We evaluate the training time of ScannerGrouper and all baseline solutions. The ScannerGrouper requires training for two models: IOMatch and logistic regression. We focus on recording the training time of IOMatch, as the logistic regression training is negligible, taking only a few seconds.

Baselines: Due to the alignment of research objectives, we compare ScannerGrouper with DarkVec [33], i-DarkVec [34], and Kallitsis's Framework [44]. In contrast, ScannerGrouper is not compared with Scanner-Hunter [64], DANTE [19], or Wu's Framework [68], as these studies pursue similar but different research objectives. These solutions either focus solely on identifying the organizations associated with Modbus scanners [64] or are only effective for a small number of scanned organizations [19, 68].
- **i-DarkVec [34]:** i-DarkVec applies Natural Language Processing (NLP) techniques, starting by constructing a corpus. i-Darkvec group the scanned ports into sets and treat the temporal sequences of scanner IPs reaching each set as word sequences. A Word2Vec model is then trained to embed each scanner IP into a feature vector. i-DarkVec uses k-Nearest-Neighbor (k-NN) classifier to identify scanners based on the majority labels of their k neighbors in the embedding feature space. Furthermore, i-Darkvec leverages an incremental learning approach.
- **DarkVec [33]:** DarkVec adopts the same approach as the i-DarkVec for collecting temporal features and constructs embeddings using the Word2Vec model. However, it lacks the capability for incremental updates.
- **Kallitsis's Framework [44]:** Kallitsis's framework analyzes daily scanning traffic to extract features for each scanner, focusing on statistics such as total packets, bytes sent, counts of targeted ports, and IPs. An Autoencoder is used to generate low-dimensional embeddings. Kallitsis's framework also adopts k-NN classifier to identify scanners within the embedding feature space. Since features are constructed daily, the same scanner IP may appear multiple times across days, resulting in repeated identification in our evaluation. To resolve this, a voting mechanism aggregates results for each IP.

The three baselines were originally designed for darknet traffic, which differs from honeypot traffic and, as a result, cannot be directly applied to our datasets. In darknet traffic, there are no responses to incoming packets, so each TCP connection contains only the initial SYN packet, and each UDP session includes only the first packet sent by the scanner. Therefore, when evaluating these baselines, we adapt our datasets to the darknet style by retaining only the first SYN packet of each TCP connection and the first packet of each UDP session.

4.2 Identification Performance

Incremental Training Experiments. ScannerGrouper-i conducts weekly incremental training to update the system continuously.

The performance in the final week of each dataset is considered representative of ScannerGrouper-i's overall performance. The results are presented in Tabs. 5 and 6. Scanning organizations with fewer than five scanners in the test set are excluded, as their results are not representative. The same filtering criterion is applied throughout §4.2. As shown in Tabs. 5 and 6, the weighted average F1-score of ScannerGrouper-i ranges from 0.82 to 0.85—3.41 to 4.05 times higher than that of i-Darkvec. While i-Darkvec excels at identifying unknown classes, it is less accurate in classifying known ones[5]. This is due to the large amount of unknown samples in the training set, which benefits k-NN in identifying unknown classes.

Table 5: Performance of incremental training on the last week of SelfDeploy-24 dataset.

	ScannerGrouper-i			i-DarkVec			# of scanners
	Precision	Recall	F1-score	Precision	Recall	F1-score	
binaryedge	0.52	0.83	0.64	0.10	0.02	0.03	65
criminalip	1.00	0.40	0.57	0.10	0.20	0.13	5
internet_census	1.00	1.00	1.00	0.00	0.00	0.00	40
ipip	1.00	0.80	0.89	0.00	0.00	0.00	5
onyphe	1.00	0.78	0.88	0.00	0.00	0.00	9
shadowserver	0.97	0.52	0.67	0.09	0.05	0.06	134
shodan	1.00	1.00	1.00	0.00	0.00	0.00	16
stretchoid	0.89	0.98	0.93	0.45	0.09	0.15	328
censys and driftnet (in unknown class)	0.83	0.76	0.79	0.37	0.79	0.50	339
weighted avg	0.86	0.81	0.82	0.31	0.32	0.24	941

Though our approach outperforms the baseline, ScannerGrouper-i underperforms for some organizations in two scenarios: (1) scanners of some organizations send only one type of service probe with payloads resembling those of another organization, and (2) the training sample size is small for some organizations. For scenario (1), consider the BinaryEdge identification in SelfDeploy-24: 11 BinaryEdge scanners are misclassified as Stretchoid. These IPs send only TLS probes (110 in total) during the test period, and their payloads—specifically the handshake_ciphersuites, hash_algorithms, and supported_groups fields—are identical to those in 230 TLS probes from Stretchoid (out of 489 in total). This payload similarity leads to misclassifications at the packet level, which are then aggregated to the scanner level.

For scenario (2), in the SelfDeploy-25 dataset, two small-sample organizations—ipip and Rapid7—achieve an F1-score of 0. Specifically, 5 of 6 ipip scanners and all 5 Rapid7 scanners are classified as unknown. This likely stems from insufficient training data, which causes the logistic regression models for these classes to output low confidence scores for their test samples—below the unknown-class threshold θ—resulting in misclassification as unknown.

In addition, we also present the weighted average F1-score over the last four weeks, highlighting the upward performance trend of ScannerGrouper-i. We further evaluate ScannerGrouper-i with one more dataset, i.e., the SelfDeploy25-S dataset. For more details, please refer to the Appendix J.

One-time Training Experiments. For the SelfDeploy datasets, the experimental results in Tabs. 7 and 8 show that our ScannerGrouper-f achieves the F1-score above 0.80, significantly outperforms Darkvec and Kallitsis's Framework. Notably, Kallitsis's Framework excels only in identifying unknown classes, likely because the training set contains numerous unknown samples. This abundance causes k-NN classification to favor unknown class samples, highlighting the limitations of traditional k-NN methods in scenarios with many unknown class samples. Next, we evaluate ScannerGrouper-f and the baselines on the CompanyA dataset and assess ScannerGrouper-f's performance on the CompanyB dataset (where the dataset lacks the packet header information necessary for baseline feature construction). The experimental results in Appendix K indicate that, similar to its performance on SelfDeploy, ScannerGrouper-f achieves a weighted average F1-score exceeding 0.8 on both datasets, consistently outperforming the baselines.

[5]Precision is marked as "–" when its denominator (TP + FP) is zero.

Table 6: Performance of incremental training on the last week of SelfDeploy-25 dataset.

	ScannerGrouper-i			i-DarkVec			# of scanners
	Precision	Recall	F1-score	Precision	Recall	F1-score	
binaryedge	1.00	0.65	0.79	0.78	0.15	0.25	92
criminalip	1.00	0.82	0.90	0.09	0.09	0.09	11
internet_census	1.00	0.96	0.98	0.58	0.10	0.17	72
internettl	0.54	1.00	0.70	0.00	0.00	0.00	7
ipip	-	0.00	0.00	-	0.00	0.00	6
onyphe	0.68	0.81	0.74	0.22	0.54	0.31	26
rapid7	-	0.00	0.00	-	0.00	0.00	5
shadowserver	0.92	0.51	0.66	0.09	0.01	0.02	118
shodan	1.00	1.00	1.00	-	0.00	0.00	10
stretchoid	0.91	0.98	0.95	0.72	0.08	0.14	400
censys and driftnet (in unknown class)	0.77	0.96	0.85	0.27	0.85	0.41	265
weighted avg	0.88	0.87	0.85	0.49	0.29	0.21	1012

Additionally, we evaluate ScannerGrouper-f using various training–test split ratios (7:3, 8:2, 9:1) and observe only minor differences in identification performance. Specifically, taking the 8:2 ratio as the baseline, performance fluctuations under other ratios do not exceed 5%. For details, see Appendix L.

We also evaluate ScannerGrouper-f using two scanner lists: one derived from darknet data [20], and another from a GitHub open repository [5]. Our system accurately identifies the organizations behind scanners overlapping with these lists and our honeypot datasets, achieving high F1-scores between 0.78 and 0.84. Detailed results are provided in Appendix M.

Similar to ScannerGrouper-i, ScannerGrouper-f also underperforms for certain scanning organizations, with analysis revealing similar issues: (1) scanners of some organizations send only one type of service probe whose payloads closely resemble those of another organization, and (2) the training sample size is insufficient for some organizations.

4.3 Identification Result Analysis of Unknown-Class Scanners

Using the SelfDeploy-24 testing set as an example, we analyze the identification results of unknown-class scanners based on ScannerGrouper-i's output, with similar observations on SelfDeploy-25. The outcomes fall into two categories: (1) unknown-class scanners identified as known classes, and (2) unknown-class scanners still recognized as unknown classes.

Case 1: unknown-class scanners identified as known classes. We compare unknown-class probes classified as known to those originally labeled known-class probes by examining their 128-dimensional feature vectors, extracted by the trained IOMatch model, representing each probe image. For visualization, we apply t-SNE to reduce these vectors to two dimensions for each service.

As shown in Fig. 2, probes from unknown-class scanners classified as known classes are close in feature space to those probes originally labeled as the same class, indicating payload similarity. This suggests that some unknown-class scanners—lacking labels

Table 7: Performance of one-time training on SelfDeploy-24 dataset

	ScannerGrouper-f			DarkVec			Kallitsis's Framework			# of scanners
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	
binaryedge	0.54	0.77	0.64	0.47	0.82	0.60	-	0.00	0.00	132
criminalip	1.00	0.33	0.50	0.00	0.00	0.00	-	0.00	0.00	6
internet_census	1.00	1.00	1.00	0.03	0.02	0.02	-	0.00	0.00	53
ipip	1.00	1.00	1.00	-	0.00	0.00	-	0.00	0.00	5
onyphe	1.00	0.88	0.93	0.10	0.13	0.11	-	0.00	0.00	16
shadowserver	1.00	0.54	0.70	0.44	0.50	0.47	0.00	0.00	0.00	243
shodan	1.00	1.00	1.00	0.00	0.00	0.00	-	0.00	0.00	22
stretchoid	0.90	0.91	0.91	0.62	0.54	0.58	-	0.00	0.00	518
zoomeye	0.20	0.17	0.18	-	0.00	0.00	-	0.00	0.00	6
censys and driftnet (in unknown class)	0.71	0.81	0.76	0.80	0.37	0.51	0.31	1.00	0.47	449
accuracy	-	-	0.80	-	-	0.47	-	-	0.31	1450
macro avg	0.89	0.70	0.73	0.20	0.15	0.13	0.44	0.09	0.07	1450
weighted avg	0.83	0.80	0.80	0.59	0.47	0.49	0.20	0.31	0.15	1450

Table 8: Performance of one-time training on SelfDeploy-25 dataset

	ScannerGrouper-f			DarkVec			Kallitsis's Framework			# of scanners
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	
binaryedge	0.85	0.91	0.88	0.42	0.51	0.46	-	0.00	0.00	148
criminalip	1.00	0.82	0.90	0.08	0.09	0.09	-	0.00	0.00	11
fofa	1.00	1.00	1.00	0.00	0.00	0.00	-	0.00	0.00	7
internet_census	1.00	0.94	0.97	0.29	0.61	0.39	-	0.00	0.00	124
internettl	0.49	1.00	0.66	0.04	0.06	0.05	-	0.00	0.00	18
ipip	1.00	0.50	0.67	0.00	0.00	0.00	-	0.00	0.00	6
onyphe	1.00	0.57	0.73	0.26	0.55	0.35	-	0.00	0.00	79
rapid7	1.00	1.00	1.00	0.00	0.00	0.00	-	0.00	0.00	5
shadowserver	0.99	0.55	0.71	0.45	0.71	0.55	-	0.00	0.00	282
shodan	1.00	1.00	1.00	0.00	0.00	0.00	-	0.00	0.00	18
stretchoid	0.91	0.97	0.94	0.62	0.08	0.14	-	0.00	0.00	793
censys and driftnet(in unknown class)	0.77	0.92	0.84	0.37	0.40	0.39	0.25	1.00	0.40	502
accuracy	-	-	0.87	-	-	0.33	-	-	0.25	1993
macro avg	0.93	0.72	0.72	0.16	0.18	0.13	0.25	0.06	0.02	1993
weighted avg	0.89	0.87	0.87	0.46	0.33	0.30	0.25	0.25	0.10	1993

from the three sources described in §3.2.1—may actually belong to known scanning organizations. ScannerGrouper thus helps expand analysts' knowledge of scanner ranges for known classes.

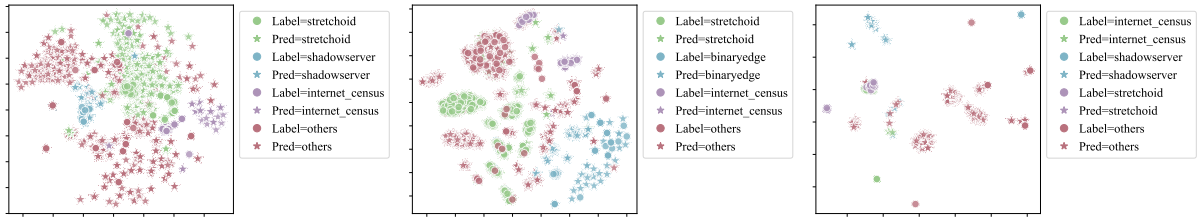
Case 2: unknown-class scanners still recognized as unknown classes. To evaluate payload similarity among probes from these scanners, we apply K-means clustering to their 128-dimensional feature vectors generated by IOMatch (performed separately for each service). The number of clusters K is automatically determined using the silhouette coefficient, as detailed in Appendix N. Clustering results for HTTP, TLS, and DNS services are visualized using 2D t-SNE in Fig. 3. We then perform statistical analysis on the clustered probes to summarize scanning characteristics of the unknown-class groups.

For HTTP probes, six clusters are identified, as summarized in Tab. 9. *Notably, HTTP probes from Palo Alto Networks—an organization not initially identified as a scanning organization by us—are clustered into a single group, demonstrating ScannerGrouper 's capability to uncover previously unknown scanning organizations.*

Table 9: Summary of still-unknown HTTP probe clusters

Cluster	Scanning Type	# of Probes	Description
0	N/A scanner	1210	92.1% of probes send "N/A" User-Agent. 92.4% of probes send "N/A" Accept. 95.7% of probes send "/" Url. All **censys** 464 HTTP probes in this cluster.
1	Scattered scanner	405	Most probes (71.1%) send "*/*" Accept. 32.6% of probes are from **driftnet**, sending User-Agent: "Mozilla/5.0 (compatible; InternetMeasurement/1.0…)". 9.1% of probes send "curl/7.64.1" User-Agent. 8.6% of probes send "Mozilla/5.0 (compatible; Odin…)" User-Agent
2	N/A and Go scanner	471	Most probes (96.6%) send "N/A"Accept. 54.4% of probes send "N/A" User-Agent. 44.2% of probes send "Go-http-client/1.1" User-Agent. For Url, all probes are requesting non-empty resourses: "google.com:443" (36.7%), "/nice%20ports%2C/Tri%6Eity.txt%2ebak" (13.6%)....
3	Zgrab scanner	81	92.6% of probes send "Mozilla/5.0 zgrab/0.x" User-Agent.
4	Go scanner	467	93.4% of probes send "Go-http-client/1.1" User-Agent.
5	Paloalto scanner	388	58.8% of probes send "Expanse, a Palo Alto Networks company, searches across the global IPv4 space…" User-Agent.

For TLS probes, eight clusters are identified, as summarized in Tab. 10. The main differences among clusters lie in the "Handshake Cipher Suites" field: clusters 1, 2, 5, and 7 primarily use single cipher-suite probes, while cluster 4 employs a wide range of cipher suites. We further analyze cluster 1, which contains 280 probes—all using the same cipher suite configuration (Suites-19A). Of these, 271 originate from IPs publicly attributed to Censys. According to [26, 27], Censys uses the open-source scanning tool ZGrab [12], which defaults to the Suites-19A configuration. Thus, these 271 probes can be confidently attributed to Censys. The remaining 9 probes in cluster 1 also use Suites-19A but originate from IPs not disclosed by Censys. These are likely sent by other scanning organizations also using ZGrab.

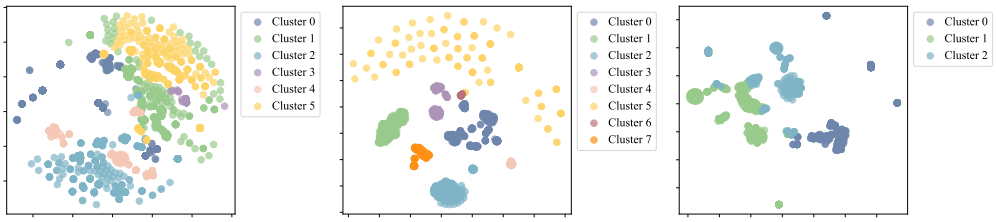


(a) HTTP known-class probes and unknown-class probes predicted to be known classes

(b) TLS known class probes and unknown-class probes predicted to be known classes

(c) DNS known class probes and unknown-class probes predicted to be known classes

Figure 2: t-SNE visualization of unknown-class probes classified as known classes. Each service figure shows the top 3 organizations by probe volume, marking the rest as "others". The star symbol represents unknown-class probes identified as known classes, and the circle symbol represents probes originally labeled as known classes.



(a) HTTP still-unknown probes

(b) TLS still-unknown probes

(c) DNS still-unknown probes

Figure 3: t-SNE visualization of clustering result for still-unknown probes, i.e., the probes sent from unknown-class scanners which are still classified as unknown-class by ScannerGrouper-i.

Table 10: Summary of still-unknown TLS probe clusters. The number after the word "suites" denotes the length of the value of Handshake Cipher Suites, which contains several names of ciphers.

Cluster	Scanning Type	# of Probes	Description
0	Scattered scanner	320	Most probes (69.1%) send empty Handshake Session ID. Common used ciphersuites: suites52 (26.6% probes), suites78 (20.6% probes), suites13-A (9.7% probes).
1	Suites19-A scanner	280	100% of probes use suites19-A, incluing all **censys** 271 TLS probes.
2	Suites19-B scanner	221	88.2% of probes use suites19-B.
3	Suites31 and suites43 scanner	131	Common used ciphersuites: suites31 (57.3% probes), suites43 (36.6% probes).
4	Large suites scanner	36	Common used ciphersuites: suites109 (44.4% probes), suites158 (25% probes), suites397 (11.1% probes).
5	Suites20 scanner	145	98.6% of probes use suites20-A, and those 98.6% probes send Handshake Session ID: b"\x84\b4,\x85 \xafn\xe3Y\xbbbhl\xff{=':\xa9\x82\xd9o\xc8\xa2\xd7\x93\x98\... ".
6	Scattered scanner	18	All (100%) probes send empty Handshake Session ID. Common used ciphersuites: suites20-B (38.9% probes), suites6 (33.3% probes), suites13-B (27.8% probes).
7	Suites16 scanner	117	75.2% of probes are using suites16.

For DNS probes, the results are similar and detailed in Appendix O, by clustering probes from unknown-class scanners still identified as unknown, ScannerGrouper effectively groups them by payload features, aiding security analysts in understanding ongoing scanning activities.

Takeaways: ScannerGrouper automatically clusters service probes from unknown-class scanners and reports payload statistics for each cluster. This facilitates: (1) identifying known-class scanners among unlabeled ones, (2) discovering new scanning organizations through payload analysis, and (3) expanding analysts' understanding of unknown scanning activities to support rapid defense strategies, such as developing cluster-specific payload-based filtering rules.

4.4 Training Time

Training time is the most time-consuming part of preparing the solutions, evaluated in this subsection, as shown in Tab. 11. The training time of ScannerGrouper-i is negligible compared to the time period of collecting datasets. For incremental training, both ScannerGrouper-i and i-DarkVec exhibit short training times, with ScannerGrouper-i requiring only 1.0 to 1.2 minutes for weekly data updates. In one-time training experiments, ScannerGrouper-f takes 17.7 to 18.5 minutes on the SelfDeploy dataset, longer than the baselines. It is worth noting that ScannerGrouper-f is intended to evaluate the performance ceiling of ScannerGrouper and where ScannerGrouper-i is preferred for deployment in real-world scenarios. Additionally, ScannerGrouper-f requires 192.2 to 202.1 minutes on the CompanyB datasets due to the large volume of probes collected, which increases the training time.

Table 11: Training Time Comparison

	Dataset (Collection Time)	ScannerGrouper-i	i-Darkvec	
Incremental Training	**SelfDeploy-24**	1.2 min/week	0.1 min/week	
	SelfDeploy-25	1.0 min/week	0.5 min/week	
	Dataset (Collection Time)	**ScannerGrouper-f**	**Darkvec**	**Kallitsis's Framework**
One-time Training	**SelfDeploy-24 (2.01-4.21)**	17.7 min	0.9 min	0.7 min
	SelfDeploy-25 (12.10-2.25)	18.5 min	6.2 min	3.1 min
	CompanyA (10.01-10.23)	28.7 min	2.3 min	1.7 min
	CompanyB-22 (10.01-10.28)	192.2 min	/	/
	CompanyB-23 (10.01-10.28)	202.1 min	/	/

4.5 IOMatch vs Non-image Open-set Models

We compare the performance of the IOMatch model with three representative non-image open-set models: (1) 1-vs-set[63], (2) PI-SVM[43]—both based on SVM—and (3) EVM[61], which uses extreme value theory. These models require vector-form inputs instead of image inputs, so we refer to them collectively as non-image models. For fair comparison, we follow the same dataset split as ScannerGrouper-f. We encode the features described in §3.3 into vectors using label encoding for input to the non-image models. In contrast, IOMatch directly processes image-form payload features

without vectorization. As shown in Tab. 12, IOMatch consistently outperforms the non-image open-set models[6] across all datasets and services. This highlights the effectiveness of image-based payload representations (§3.4) in capturing feature semantics, and demonstrates IOMatch's strength in learning payload patterns.

4.6 ScannerGrouper Hyperparameter Analysis

We evaluate the impact of different update frequencies on the identification performance of ScannerGrouper-i. In addition to the default 7-day update, we test 3-day, 5-day, and 10-day intervals using the SelfDeploy24 and SelfDeploy25 datasets. Identification performance is measured over the last four weeks of each dataset, as shown in Figs. 4 and 5. Updating every 3 days results in noticeable fluctuations in F1-score, likely due to overfitting or instability caused by overly frequent updates. In contrast, 5-day and 7-day updates yield higher and more stable F1-scores, indicating effective and robust learning. A 10-day update leads to reduced performance, possibly due to slower adaptation to new data.

We also analyze how training hyperparameters affect IOMatch performance. The key hyperparameters include the number of training epochs and batch size. Through analysis, we conclude that the parameter setting we use in both ScannerGrouper-i and ScannerGrouper-f yield stable performance across different datasets.

Moreover, we analyze the impact of different thresholds θ automatically selected via equidistant search in the Result Aggregation Module on scanner-level identification performance. The analysis results confirm that the θ obtained through the equidistant search on the training set can be reliably applied to the testing set.

The detailed analysis is presented in Appendix P.

4.7 Impact of Unknown-Class Scanner Ratio on Identification Performance

To examine the impact of unknown-class scanner proportions on model training, we use ScannerGrouper-f with random sampling to reduce the ratio of unknown scanners in the training set and retrain the model. As shown in Tab. 13, the F1-scores for different proportions (25%, 50%, 75%) remain close to the original setting (100%), with only a 4%–11% variation. This demonstrates the robustness of our system to imbalances in unknown-class scanner distribution. Additional results are provided in Appendix Q.

5 Related Work

Scanning Activity Identification for Network Intrusion Detection Systems. A large body of literature [22, 37, 47, 70] studied the identification problem of scanning activity for Network Intrusion Detection Systems (NIDS). The solutions proposed in these studies are capable of identifying various abnormal activities from network traffic, with scanning activity being one notable example. However, while these studies can identify the public IPs of scanners, they show no interest in the organizations to which the analyzed scanners belong.

Scanning Source Analysis in Measurement Studies. Recently, researchers have developed various monitoring systems to capture and analyze scanning traffic, enabling comprehensive measurement studies [28, 30, 31, 36, 38, 48, 58, 59, 65]. In these studies, various methods are utilized to analyze the origins of scanners (The first three methods have been employed in all these works.), but they fail to identify the affiliated organizations for a substantial number of scanners that lack domain names and are not included in any official publicly available scanner lists: (1) Querying WHOIS or IP geolocation databases to map scanner IPs to their sources at the level of cloud computing companies, Autonomous System (AS), country, etc. (2) Collecting publicly available scanner lists from the official websites of a few scanning organizations [2, 3, 7]. (3) Scanner domain names obtained through reverse domain name resolution are registered by various scanning organizations (e.g., Shodan [9]) or cloud computing companies (e.g., AWS [1]). (4) Griffioen et al. [36] used intelligence from specific security companies to identify organizations behind certain scanners. However, the intelligence generation methods lack transparency, and the approach's accuracy heavily depends on the credibility of the intelligence providers. (5) Griffioen et al. [36] also relied on third-party scanner lists [20] that are publicly available but not thoroughly validated.

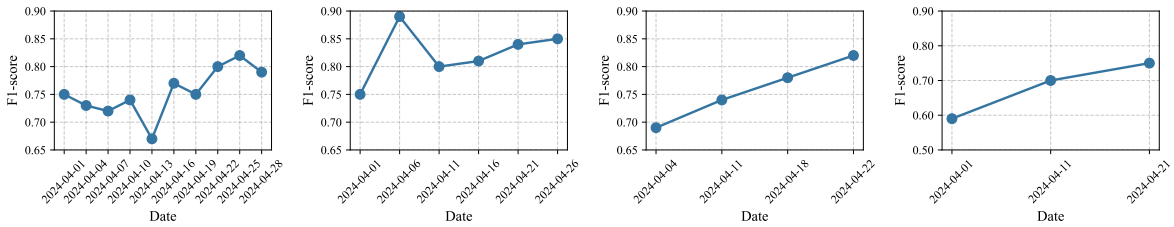
Scanning Organization Identification Studies. Several existing studies, including DarkVec [33], i-DarkVec [34], Kallitsis's Framework [44], Scanner-Hunter [64], DANTE [19], and Wu's Framework [68], aim to uncover the sources of a larger number of unattributed scanners at the organizational level. Nevertheless, they suffer from numerous shortcomings, making them either unsuitable for general use, ineffective in real-world scenarios, or both:

- Limitations of darknets: DarkVec, i-Darkvec, Kallitsis's Framework and DANTE identify scanner organizations from Darknet traffic. Darknets passively capture traffic directed to lots of routable but unused public IP addresses [15, 53]. These addresses neither host Internet services nor serve real Internet users. Those who wish to adopt darknet-based solutions must either invest in a large number of public IP addresses to build their own darknet, or rely on scanner attribution intelligence from external darknets (e.g., DomainTools [23]). However, the latter approach is ineffective for the increasingly common localized scanning. Prior studies [38, 56, 58] have demonstrated that scanners may only probe IP addresses within particular routed prefixes, such as those belonging to specific enterprise networks. Therefore, proponents of darknet-based solutions must resort to deploying costly darknets within the target networks to monitor localized scanning. However, acquiring a sufficient number of public IP addresses from an operational network to construct such a darknet remains a significant challenge. In contrast, honeypots and NIDS are better suited for capturing localized scanning traffic.
- Insufficiently robust features: Existing studies [20, 36] highlight variability in scanned ports' temporal characteristics and scanners' statistical activities. However, solutions like DarkVec, i-DarkVec, DANTE, and Kallitsis's Framework rely on such features [19, 33, 34, 44]. DarkVec and i-DarkVec group target ports and treat the temporal sequences of scanner IPs reaching each set as word sequences, training a single Word2Vec model [51, 52] to embed scanner IPs. DANTE uses port contact sequences within intervals, training per-port Word2Vec embeddings and averaging them for scanner representation. Kallitsis's Framework

[6] "N/A" indicates errors encountered in the initial stage of evaluation, likely due to large training data.

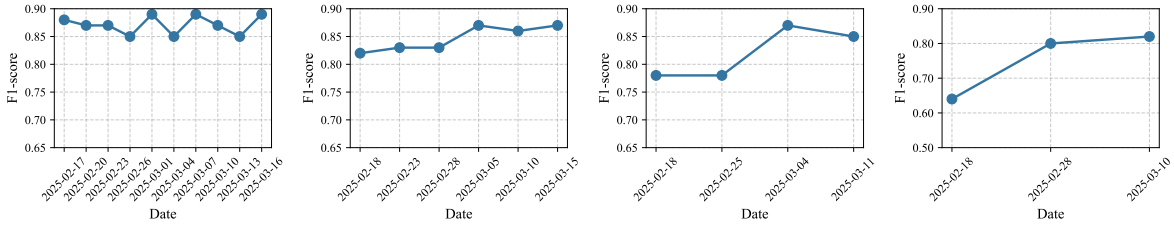
Table 12: IOMatch vs Non-image Open-set Models on probe level identification performance, we use F1-score as performance metric.

Dataset		SelfDeploy24			SelfDeploy25			Company A			Company B-22			Company B-23		
Classifier for each service		HTTP	TLS	DNS	HTTP	TLS	DNS	HTTP	TLS	DNS	HTTP	TLS	DNS	HTTP	TLS	DNS
Non-image open-set models	1-vs-set	0.62	0.29	0.12	0.25	0.04	0.12	0.12	0.01	0.22	0.05	0.78	0.20	0.13	0.01	0.05
	PI-SVM	0.13	<0.01	<0.01	0.01	0.01	0.03	0.02	0.02	0.02	<0.01	<0.01	0.04	<0.01	<0.01	0.01
	EVM	<0.01	0.04	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	0.25	<0.01	0.00	0.03	N/A	<0.01	0.80
SOTA Image-based open-set model	IOMatch	0.88	0.70	0.82	0.89	0.85	0.69	0.94	0.83	0.76	0.98	0.92	0.74	0.95	0.91	0.43



(a) 3-day update period **(b)** 5-day update period **(c)** 7-day update period **(d)** 10-day update period

Figure 4: Performance of incremental training with different update period on last four weeks of SelfDeploy24 Dataset



(a) 3-day update period **(b)** 5-day update period **(c)** 7-day update period **(d)** 10-day update period

Figure 5: Performance of incremental training with different update period on last four weeks of SelfDeploy25 Dataset

Table 13: ScannerGrouper-f identification performance (F1-score) on different unknown-class scanner ratio.

Dataset\Unknown-class scanner ratio	25%	50%	75%	100%
SelfDeploy24	0.77	0.76	0.81	0.80
SelfDeploy25	0.85	0.83	0.76	0.87

focuses on daily scanning statistics like total packets, counts of distinct targeted ports, and IP addresses.

- Manual evaluation of unattributed scanner results: DarkVec, i-DarkVec, and Kallitsis's Framework can classify certain unattributed scanners as known organizations but lack an automated method for verifying results. DarkVec and i-DarkVec use a semi-supervised k-NN model, to assign scanners to organizations based on their k-nearest neighbors. However, they require manual result verification (see Section 6.4 in [33]). Similarly, Kallitsis's Framework shares this limitation (see Section VI.B and VI.C in [44]).
- Limitations in the scope of applicability: Scanner-Hunter [64] leverages honeypot traffic and extracts features from multiple consecutive Modbus sessions, focusing on protocol-specific behaviors and fields. However, it does not propose a method for generalizing to other services. In contrast, our approach selects payload fields based on their mutual information with labels in the first service probe, making it extensible to service types beyond HTTP, TLS, and DNS. As far as we know, DANTE [19] is among the first to apply machine learning for organizational-level scanner attribution based on darknet traffic. However, it is limited to identifying scanners associated with Censys and was reported to fail to converge when applied to larger datasets with more diverse classes [33]. Wu's Framework [68] is the first to utilize "Mirror Services", whose search results from cyberspace search engines reveal scanner IPs. The framework extracts scanner IPs from only four cyberspace search engines—Censys [2], Shodan [9], FOFA [4], and ZoomEye [13]—as its effectiveness depends on the availability of search engines that provide sufficient query quotas and support batch-automatable APIs. However, such kind of search engines represent only a small subset of acknowledged scanning organizations.

6 Ethics Considerations

We collect scanning traffic from our honeypots and two partner security companies. All data is processed in secure environments to ensure no personally identifiable information (PII) is exposed. Since our research focuses on network measurement and analysis, we strictly adhere to ethical guidelines for network measurement [55, 66]. We also follow the Menlo Report [14] and other ethical guidelines [11, 45]. Moreover, all of our measurement activities have received approval from the academic committee of our department.

Our honeypots are low-interaction, responding only to initial service probes to minimize impact on hosting networks and the Internet. Deployment is carefully managed to maintain anonymity and avoid targeting. We disclose only high-level details—such as deployment continent and emulated services—while withholding information on hosting networks, honeypot counts per country, configurations, public IP addresses, and specific deployment locations to prevent potential exposure.

7 Discussion

ScannerGrouper and Imitation of Well-Known Scanning Organizations: After the release of ScannerGrouper, one might worry that attackers could gain additional insights into the scanning traffic of well-known scanning organizations, thereby facilitating imitation. However, ScannerGrouper does not introduce any new risks in this regard. The ability to impersonate well-known scanners already exists and can be achieved through well-established techniques, such as deploying honeypots and using reverse DNS lookups (also employed in §3.2.1) to analyze and mimic their scanning behavior. These techniques are independent of ScannerGrouper and have long been accessible to adversaries.

The Ability and Scalability of ScannerGrouper: ScannerGrouper classifies scanners using service probe features. Its clustering reveals payload features that help analyze their targets, techniques, tactics, and sometimes infer organizational identity. This is a useful first step toward identifying cybercriminal groups with multiple scanners. ScannerGrouper is capable of classifying any scanner that probes HTTP, TLS, or DNS services. As shown in Tab. 3, over 85% of scanners from known organizations target at least one of these services. For the minority of scanners that do not, ScannerGrouper can be adapted by incorporating other services as needed. Importantly, the design of ScannerGrouper is not limited to specific service types; its methodology remains applicable as long as suitable service traffic is available.

8 Conclusion

This paper presents ScannerGrouper, the first darknet-independent system—to the best of our knowledge—for identifying the organizations behind Internet scanners, generalizable and effective in open-world scenarios. ScannerGrouper achieves weighted average F1-score of 0.8-0.89, significantly higher than baselines. We also evaluate the identification of unattributed scanners, training efficiency, alternative core models, and the impact of hyperparameters.

Open Science

We have provided the artifacts at https://github.com/anonymous123-web/scannergrouper that contains no author-identifying information. The following artifacts will be openly shared: (1) the source code of ScannerGrouper; (2) the reimplementation of Kallitsis's framework [44], as the original code is not publicly available; (3) the honeypot traffic datasets we collected, with public honeypot IP addresses anonymized; and (4) the ground truth dataset used in §4. However, the honeypot datasets provided by two collaborating companies are proprietary and cannot be publicly released.

9 Acknowledgments

We would like to thank the anonymous reviewers for their constructive feedback. We would also like to thank Haoran Zhou for his help in re-implementing Kallitsis's framework [44]. This work is supported by the National Key R&D Program of China under Grant 2022YFB3102902, and the National Natural Science Foundation of China under Grant 62172251. Enhuan Dong and Jiahai Yang are the corresponding authors of this paper.

References

[1] AWS Cloud DNS service Website. https://aws.amazon.com/cn/route53/.
[2] Censys Search Engine. https://search.censys.io. Accessed on 2025-1.
[3] Driftnet Website. https://internet-measurement.com/. Accessed on 2025-1.
[4] FoFa Search Engine. https://en.fofa.info. Accessed on 2025-1.
[5] IP list of known scanners from Connie-Wild. https://github.com/Connie-Wild/scanner-ip-list.
[6] Ipsniper RDNS Dataset. https://ipsniper.info/archive/dns_data/. Accessed on 2025-1.
[7] Rapid7 Website. https://opendata.rapid7.com. Accessed on 2025-1.
[8] RFC2535. https://www.rfc-editor.org/rfc/rfc2535.html.
[9] Shodan Search Engine. https://www.shodan.io. Accessed on 2025-1.
[10] training of logistic regression model. https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression.
[11] USENIX Security '25 Ethics Guidelines. https://www.usenix.org/conference/usenixsecurity25/ethics-guidelines. Accessed on 2025-1.
[12] ZGrab 2.0. https://github.com/zmap/zgrab2. Accessed on 2025-1.
[13] ZoomEye Search Engine. https://www.zoomeye.org. Accessed on 2025-1.
[14] Michael Bailey, Aaron Burstein, KC Claffy, and et al. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. *Homeland Security: Science and Technology*, 2012.
[15] Michael Bailey, Evan Cooke, Farnam Jahanian, Andrew Myrick, and Sushant Sinha. Practical Darknet Measurement. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 1496–1501. IEEE, 2006.
[16] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.
[17] CAIDA. The UCSD Network Telescope. http://www.caida.org/projects/network_telescope/. Accessed on July 2025.
[18] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems*, 13, 2000.
[19] Dvir Cohen, Yisroel Mirsky, Manuel Kamp, Tobias Martin, Yuval Elovici, Rami Puzis, and Asaf Shabtai. DANTE: A framework for mining and monitoring darknet traffic. In *Computer Security–ESORICS, Proceedings, Part I 25*, pages 88–109. Springer, 2020.
[20] M. Collins, A. Hussain, and S. Schwab. Identifying and Differentiating Acknowledged Scanners in Network Traffic. In *IEEE EuroS&PW*, pages 567–574, 2023.
[21] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, and Manfred K Warmuth. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(3), 2006.
[22] H. Ding, Y. Sun, N. Huang, Z. Shen, and X. Cui. TMG-GAN: Generative Adversarial Networks-Based Imbalanced Learning for Network Intrusion Detection. *IEEE TIFS*, 19:1156–1167, 2023.
[23] Domain Tools. Farsight Security Information Exchange. https://www.domaintools.com/resources/user-guides/security-information-exchange-sie-darknet-channel/. Accessed on July 2025.
[24] Enhuan Dong, Jiahai Yang, Zhiliang Wang, Xin He, Jie Fang, Shenao Li, Supei Zhang, Zeyuan Guo, Hui Zhang, Zimu Li, et al. Internet Scan Source Identification: A Survey. *IEEE Network*, 2025.
[25] Z. Durumeric, E. Wustrow, and J. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security*, pages 605–620, 2013.
[26] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. A Search Engine backed by Internet-wide Scanning. In *ACM CCS*, pages 542–553, 2015.
[27] Zakir Durumeric, David Adrian, Phillip Stephens, Eric Wustrow, and J Alex Halderman. Ten Years of ZMap. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, pages 139–148, 2024.
[28] Zakir Durumeric, Michael Bailey, and J Alex Halderman. An Internet-Wide Wiew of Internet-Wide Scanning. In *USENIX Security*, pages 65–78, 2014.
[29] Claude Fachkha and Mourad Debbabi. Darknet as a Source of Cyber Intelligence: Survey, Taxonomy, and Characterization. *IEEE Communications Surveys & Tutorials*, 18(2):1197–1227, 2015.
[30] T. Favale, D. Giordano, I. Drago, and Mllia. What Scanners do at L7? Exploring Horizontal Honeypots for Security Monitoring. In *IEEE EuroS&PW*, pages 307–313, 2022.
[31] Kensuke Fukuda and John Heidemann. Who knocks at the ipv6 door? detecting ipv6 scanning. In *ACM IMC*, pages 231–237, 2018.
[32] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3614–3631, 2020.
[33] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi, and Dario Rossi. DarkVec: Automatic Analysis of Darknet Traffic with Word Embeddings. In *ACM CoNEXT*, pages 76–89, 2021.
[34] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi, and Dario Rossi. i-DarkVec: Incremental Embeddings for Darknet Traffic Analysis. *ACM TOIT*, 23(3):1–28, 2023.

[35] R. Graham. MASSCAN: Mass IP Port Scanner. https://github.com/robertdavidgraham/masscan. Accessed on 2025-1.
[36] H. Griffioen, G. Koursiounis, G. Smaragdakis, and C. Doerr. Have you SYN me? Characterizing Ten Years of Internet Scanning. In *ACM IMC*, 2024.
[37] Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar. Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 25(1):538–566, 2023.
[38] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C Schmidt, and Matthias Wählisch. Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope. In *USENIX Security*, pages 431–448, 2022.
[39] Szu-Chun Huang, Harm Griffoen, Max van der Horst, Georgios Smaragdakis, Michel van Eeten, and Yury Zhauniarovich. Trust but Verify: An Assessment of Vulnerability Tagging Services. In *USENIX Security*, Seattle, WA, 2025.
[40] L. Izhikevich, R. Teixeira, and Z. Durumeric. LZR: Identifying Unexpected Internet Services. In *USENIX Security*, pages 3111–3128, 2021.
[41] L. Izhikevich, R. Teixeira, and Z. Durumeric. Predicting IPv4 Services Across All Ports. In *ACM SIGCOMM*, pages 503–515, 2022.
[42] Liz Izhikevich, Manda Tran, Katherine Izhikevich, Gautam Akiwate, and Zakir Durumeric. Democratizing leo satellite network measurement. In *ACM SIGMETRICS*, 2024.
[43] Lalit P Jain, Walter J Scheirer, and Terrance E Boult. Multi-class open set recognition using probability of inclusion. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III 13*, pages 393–409. Springer, 2014.
[44] Michalis Kallitsis, Rupesh Prajapati, Vasant Honavar, Dinghao Wu, and John Yen. Detecting and Interpreting Changes in Scanning Behavior in Large Network Telescopes. *IEEE TIFS*, 17:3611–3625, 2022.
[45] Tadayoshi Kohno, Yasemin Acar, and Wulf Loh. Ethical frameworks and computer security trolley problems: Foundations for conversations. In *USENIX Security*, pages 5145–5162, 2023.
[46] Zekun Li, Lei Qi, Yinghuan Shi, and Yang Gao. IOMatch: Simplifying Open-set Semi-supervised Learning with Joint Inliers and Outliers Utilization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15870–15879, 2023.
[47] J. Liang, W. Guo, T. Luo, H. Vasant, G. Wang, and X. Xing. FARE: Enabling Fine-Grained Attack Categorization Under Low-Quality Labeled Data. In *NDSS*, 2021.
[48] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao, and G. Ahn. HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems. In *ACM CCS*, pages 279–291, 2020.
[49] Y. Luo, C. Li, Z. Wang, and J. Yang. IPREDS: Efficient Prediction System for Internet-wide Port and Service Scanning. In *ACM CoNEXT*, 2024.
[50] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, 2009.
[51] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781, 2013.
[52] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.
[53] David Moore, Colleen Shannon, Geoffrey M Voelker, and Stefan Savage. Network Telescopes: Technical Report. 2004.
[54] C. Munteanu, S. Saidi, O. Gasser, G. Smaragdakis, and A. Feldmann. Fifteen Months in the Life of a Honeyfarm. In *ACM IMC*, pages 282–296, 2023.
[55] Craig Partridge and Mark Allman. Ethical Considerations in Network Measurement Papers. *Communications of the ACM*, 59(10):58–64, 2016.
[56] Eric Pauley, Paul Barford, and Patrick McDaniel. DScope: A Cloud-Native Internet Telescope. In *USENIX Security*, pages 5989–6006, 2023.
[57] Ryan Pickren, Animesh Chhotaray, Frank Li, Saman Zonouz, and Raheem Beyah. Release the hounds! automated inference and empirical security evaluation of field-deployed plcs using active network data. In *ACM CCS*, pages 3674–3688, 2024.
[58] Philipp Richter and Arthur Berger. Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope. In *ACM IMC*, pages 144–157, 2019.
[59] Philipp Richter, Oliver Gasser, and Arthur Berger. Illuminating Large-scale IPv6 Scanning in the Internet. In *ACM IMC*, pages 410–418, 2022.
[60] B. C. Ross. Mutual Information between Discrete and Continuous Data Sets. *PLoS ONE*, 9(2), 2014.
[61] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):762–768, 2017.
[62] Said Jawad Saidi, Srdjan Matic, Oliver Gasser, Georgios Smaragdakis, and Anja Feldmann. Deep dive into the iot backend ecosystem. In *Proceedings of the 22nd ACM internet measurement conference*, pages 488–503, 2022.
[63] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.
[64] Chuan Sheng, Yu Yao, Lianxiang Zhao, Peng Zeng, and Jianming Zhao. ScannerHunter: An Effective ICS Scanning Group Identification System. *IEEE TIFS*, 2024.
[65] H. Tanveer, R. Singh, P. Pearce, and R. Nithyanand. Glowing in the Dark: Uncovering IPv6 Address Discovery and Scanning Strategies in the Wild. In *USENIX Security*, pages 6221–6237, 2023.
[66] Jeroen Van Der Ham. Ethics and Internet Measurements. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 247–251. IEEE, 2017.
[67] D. Wagner, S. Ranadive, H. Griffioen, M. Kallitsis, A. Dainotti, G. Smaragdakis, and A. Feldmann. How to Operate a Meta-Telescope in your Spare Time. In *ACM IMC*, pages 328–343, 2023.
[68] Mengying Wu, Geng Hong, Jinsong Chen, Qi Liu, Shujun Tang, Youhao Li, Baojun Liu, Haixin Duan, and Min Yang. Revealing the black box of device search engine: Scanning assets, strategies, and ethical consideration. In *NDSS*, 2025.
[69] Sergey Zagoruyko. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
[70] Ying Zhong, Zhiliang Wang, Xingang Shi, Jiahai Yang, and Keqin Li. RFG-HELAD: A Robust Fine-Grained Network Traffic Anomaly Detection Model Based on Heterogeneous Ensemble Learning. *IEEE TIFS*, 2024.

Appendices

A The Labeling Method for Cyberspace Search Engine Scanners in §3.2.1

Cyberspace search engines represent a significant category of scanning organizations. We propose an innovative and highly accurate method for labeling scanners associated with these search engines. This method is applied in §3.2.1 to label a subset of scanner IPs, further supporting our preliminary analysis. This section provides a detailed introduction to the proposed method, followed by an in-depth analysis of the identified search engine scanners.

A.1 The Proposed Labeling Method

The workflow of the proposed method is illustrated in Fig. 6. When a scanner from a cyberspace search engine sends an HTTP service probe to our honeypot, the honeypot generates a specially crafted HTTP response with an AES-encrypted timestamp inserted into the Set-Cookie field as a "tag". This response is sent back to the scanner, and the scanner's IP address, the timestamp response, and other relevant information are recorded in a database. Subsequently, we search for our honeypot IP across various cyberspace search engines using APIs provided by these engines or through web crawlers. We extract the encrypted timestamp from the search results and decrypt the tag stored in the Set-Cookie field. Finally, we query the records database to find the scanner's IP address.

There are two reasons for selecting the Set-Cookie field for tag insertion. First, most common fields in HTTP response packets have predefined contents and cannot be modified arbitrarily. Second, the Set-Cookie field is more flexible, with the ability to include numerous custom parameters.

A.2 Deployment and Analysis

We deployed three honeypots—two in Asia and one in Europe—hosting a simple HTTP application on ports 80 and 8080. The honeypots remained active for a duration of 50 days. The identification results are presented in Tab. 2. A total of 173 HTTP probes were successfully attributed to specific cyberspace search engines. These probes originated from 97 distinct scanners. The scanning frequencies of different search engines vary, leading to discrepancies in the number of scanners captured by the honeypots. We queried the WHOIS database and performed reverse domain name resolution on these scanner IPs. The censys scanners were associated with three ASes,

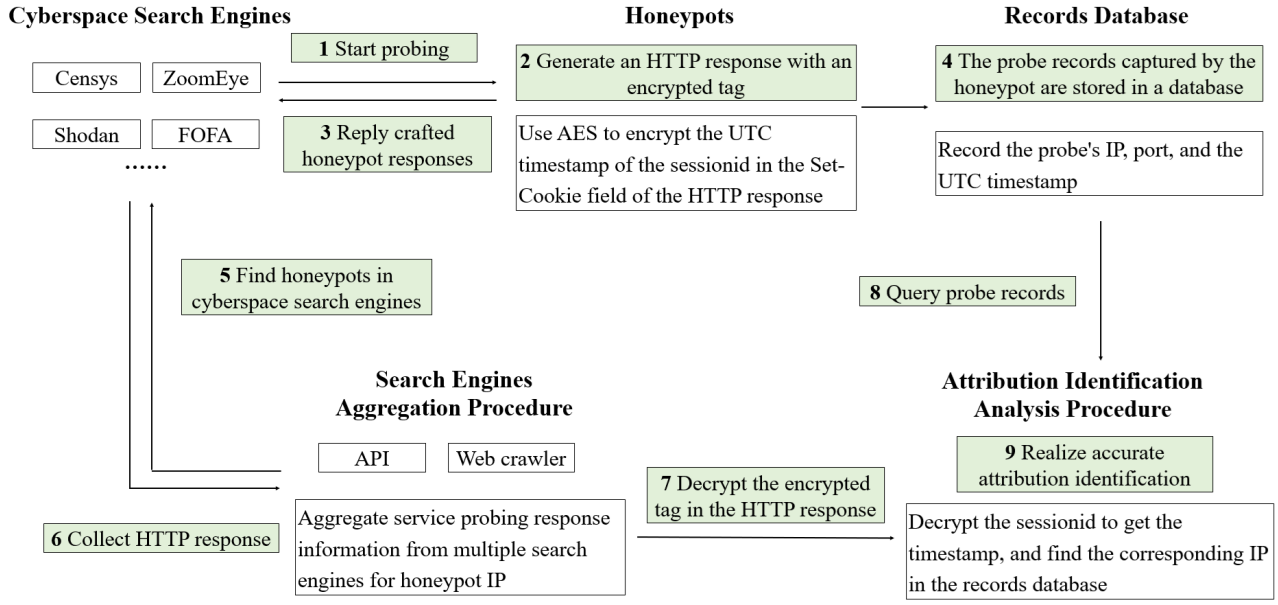


Figure 6: The workflow of the proposed labeling method applied in §3.2.1.

all of which had the domain name censys.io. ZoomEye probes were captured 52 times in total but originated from only 7 scanners. As for shodan, 8 of the 16 scanners had domain names indicating their association with shodan, while the remaining 8 had no domain name.

A.3 Ethics Considerations

The method for labeling cyberspace search engine scanners (§3.2.1, Appendix A) uses publicly available data collected via search engine APIs and web crawling. No PII was gathered, and all data was used solely for academic purposes. To minimize disruption, API calls and crawling were limited to fewer than one request per day per search engine.

B Top-10 Service of Company Datasets

We list top-10 services on two company datasets, sorted by the descending number of observed service scanners for each service, as shown in the Tab. 14.

Table 14: Top-10 service of company datasets.

CompanyA service	CompanyB service
HTTP	HTTP
TLS	SSL
SMTP	SMB
DNS	Telnet
SIP	Socks5
SSH	ADB
SNMP	SSH
SMB	RDP
Memcache	HTTP2
NTP	Redis

C Statistical Analysis of the Fields of DNS and TLS Probes

In §3.3.1, we perform a statistical analysis of the payload fields in the probes originating from each known scanning organization in the SelfDeploy dataset. Due to constraints on the length of the main body of the paper, we present the statistical analysis of the TLS handshake cipher suites (Tab. 15) and the DNS query name (Tab. 16) fields here.

D Field Selection Results for HTTP and TLS Probes

We analyze the first probes of HTTP and TLS using the mutual information method described in §3.3.2, with the results presented in Tabs. 17 to 20. In this paper, ScannerGrouper is designed as a solution that supports incremental updates. For the SelfDeploy dataset, incremental training is performed on a weekly basis, consistently using the fields selected during the first week throughout the entire training process. The results indicate that the MI values of the top six fields are significantly higher than those of other fields. Therefore, only the top six fields are used for training each dataset in this study.

E MI-Value Changes for HTTP and TLS Probes

To identify the best practices for field selection in our datasets, we calculate the MI values of the top-6 fields for HTTP and TLS services each week, using the first week's values as a reference. Results are shown in Tabs. 21 and 22.

The average MI change for the top-6 HTTP fields ranges from 4.7% to 25.1%, which is insufficient to alter the field ranking. For TLS, the average change is even smaller, ranging from 7.0% to 10.5%, and the top-6 fields remain unchanged.

In summary, the top-6 fields selected in the first week remain consistent across all subsequent weeks for both HTTP and TLS

Table 15: Statistical Analysis of the Field of Handshake Cipher Suites of TLS Service Probes

Org.	Handshake Cipher Suites	Ratio
quake	[TLS_ECDHE_RSA_WITH _CHACHA20_POLY1305_SHA256, ...]	100.0%
zoomeye	[TLS_ECDHE_ECDSA_WITH _AES_128_GCM_SHA256, TLS_...]	73.2%
internettl	[TLS_ECDHE_ECDSA_WITH _AES_256_CBC_SHA, TLS_ECD...]	100.0%
shadowserver	[TLS_ECDHE_RSA_WITH _AES_128_GCM_SHA256, TLS_EC...]	100.0%
onyphe	[TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305...]	100.0%
internet_census	[TLS_ECDHE_RSA_WITH _RC4_128_SHA, TLS_ECDHE_ECD...]	79.3%
censys	[TLS_ECDHE_RSA_WITH _CHACHA20_POLY1305_SHA256, ...]	100.0%
fofa	[TLS_ECDHE_ECDSA_WITH _AES_128_GCM_SHA256, TLS_...]	100.0%
ipip	[TLS_ECDHE_RSA_WITH _AES_128_GCM_SHA256, TLS_EC...]	100.0%
binaryedge	[TLS_ECDHE_ECDSA_WITH _AES_128_CBC_SHA, TLS_ECD...]	35.9%
shodan	[TLS_ECDHE_RSA_WITH _AES_256_GCM_SHA384, TLS_EC...]	100.0%
driftnet	[TLS_ECDHE_ECDSA_WITH _AES_128_GCM_SHA256, TLS_...]	100.0%
leakix	[TLS_ECDHE_ECDSA_WITH _AES_128_GCM_SHA256, TLS_...]	100.0%
intrinsec	[TLS_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_RC4_12...]	100.0%
stretchoid	[TLS_ECDHE_RSA_WITH _AES_128_GCM_SHA256, TLS_EC...]	29.0%
tum	[TLS_ECDHE_RSA_WITH _CHACHA20_POLY1305_SHA256, ...]	100.0%
criminalip	[TLS_ECDHE_RSA_WITH _AES_128_GCM_SHA256, TLS_EC...]	100.0%

Table 16: Statistical Analysis of the Query Name Field of DNS Service Probes

Org.	Query Name	Ratio
zoomeye	version.bind	100.0%
internettl	version.bind	100.0%
shadowserver	dnsscan.shadowserver.org	100.0%
onyphe	VERSION.BIND	100.0%
rapid7	version.bind	100.0%
cybergreen	www.cybergreen.net	100.0%
internet_census	VERSION.BIND	50.2%
censys	ip.parrotdns.com	100.0%
binaryedge	version.bind	100.0%
shodan	id.server	100.0%
stretchoid	N/A	67.8%
tum	www.google.com	100.0%
criminalip	VERSION.BIND	100.0%

services. This stability allows us to fix the selected fields over time in our datasets, reducing the update overhead for the encoding module. As a result, the same fields can be reused for long-term image encoding.

F Image Encoding

F.1 An Example of the Label Encoding Issue

In the SelfDeploy-24 dataset, the User-Agent field generates 943 distinct encoded values under label encoding. However, these values are actually derived from only 294 fundamental browser options. The 943 values of the User-Agent field in the SelfDeploy-24 dataset are combinations of these 294 base browser options. Consequently, label encoding fails to effectively capture the underlying options and their compositional relationships in option fields. Additionally, the large number of independent encoded values produced by label encoding leads to excessive granularity in pixel representation (e.g., splitting pixel intensity into 943 levels), which creates challenges for the IOMatch model to effectively learn the payload features.

F.2 Our Semantic Encoding Methods

- **Option fields:** The type of option fields may contain multiple options, with field values formed by combinations of these options, such as the HTTP User-Agent and TLS Hash Algorithms fields. For each option field, we begin by identifying all possible option values in the current dataset, with each option value represented by a single pixel in the encoded image. If the payload of the a probe contains a specific option value, the corresponding pixel's brightness is set to 255; otherwise, it is set to 0. In incremental learning scenarios, new training sets may introduce additional option values for these fields. When this occurs, we append these new option values to the end of the image representation, as detailed in §3.7.
- **String fields:** Fields consisting of a single string, which do not contain multiple options but provide semantic information through the string itself (e.g., the HTTP service's URL field, such as "/" or "/login.html"), are encoded using their ASCII values. Each character is mapped to one pixel, with brightness values ranging from 0 to 255. To minimize pixel usage, only the first 10 characters of each string field are encoded[7].
- **Integer fields:** Fields consisting of a fixed-length integer are commonly found in DNS services. The integer is converted into its binary representation with a fixed bit width. Each binary digit is then encoded as a pixel: a binary value of 1 corresponds to a brightness of 255, and a binary value of 0 corresponds to a brightness of 0.

F.3 Examples of Encoded Images

We apply the method described in §3.4 to encode the selected fields from each service probe into an image. To meet the IOMatch model's requirement for color image input, the filled rectangular image is replicated across all three RGB channels. An example is shown in Fig. 7a. As described in the module in §3.7, during the time periods following the first week, the image format may require

[7] Except for the query name field in the DNS payload, which is often much longer, 10 bytes are insufficient to capture the differences effectively.

Table 17: Top 10 probe fields for HTTP ranked by mutual information (MI) value on SelfDeploy dataset.

SelfDeploy24-1st Week		SelfDeploy24-Full		SelfDeploy25-1st Week		SelfDeploy25-full	
Field	MI Value	Field	MI Value	Field	MI Value	Field	MI Value
User-Agent	1.28	User-Agent	1.28	User-Agent	1.23	User-Agent	1.44
Accept	0.56	Accept	0.56	Accept	0.51	Accept	0.62
URL	0.52	URL	0.55	Version	0.39	Url	0.50
Accept-Encoding	0.52	Accept-Encoding	0.51	Accept-Encoding	0.38	Accept-Encoding	0.48
Version	0.25	Version	0.25	Url	0.38	Version	0.31
Connection	0.23	Connection	0.24	Connection	0.21	Connection	0.22
Method	0.10	Method	0.11	Method	0.14	Method	0.12
Accept-Language	0.05	Accept-Language	0.04	Accept-Language	0.13	Accept-Language	0.12
Pragma	0.02	Pragma	0.03	To	0.04	To	0.04
Host	0.00	Body	0.02	CSeq	0.04	CSeq	0.04

Table 18: Top 10 probe fields for HTTP ranked by mutual information (MI) value on CompanyA dataset and CompanyB datasets.

CompanyA		CompanyB-22		CompanyB-23	
Field	MI Value	Field	MI Value	Field	MI Value
User-Agent	0.98	User-Agent	0.59	User-Agent	1.32
Accept	0.56	URL	0.25	URL	0.56
Accept-Encoding	0.53	Accept	0.24	Accept	0.46
URL	0.14	Accept-Encoding	0.24	Connection	0.31
Connection	0.12	Connection	0.19	Accept-Encoding	0.27
Body	0.07	Accept-Language	0.14	Accept-Language	0.15
Accept-Language	0.04	Body	0.10	Method	0.10
Version	0.03	Cache-Control	0.07	Body	0.08
Method	0.03	Version	0.06	Version	0.05
Pragma	0.01	User-Agent	0.06	Proxy-Connection	0.02

Table 19: Top 10 probe fields for TLS ranked by mutual information (MI) value on SelfDeploy dataset.

SelfDeploy24-1st Week		SelfDeploy24-Full		SelfDeploy25-1st Week		SelfDeploy25-Full	
Field	MI Value	Field	MI Value	Field	MI Value	Field	MI Value
Handshake Random Bytes	1.63	Handshake Random Bytes	1.68	Handshake Random Bytes	1.61	Handshake Random Bytes	1.78
Handshake Time	1.63	Handshake Time	1.68	Handshake Time	1.61	Handshake Time	1.78
Handshake Cipher Suites	1.42	Handshake Cipher Suites	1.44	Handshake Ciphersuites	1.29	Handshake Ciphersuites	1.39
Extensions Hash Algs	1.14	Handshake Session ID	1.13	Extensions Hash Algs	1.08	Handshake Session ID	1.06
Handshake Session ID	1.06	Extensions Hash Algs	1.08	Handshake Session ID	0.92	Extensions Hash Algs	1.06
Extensions Supported Groups	0.47	Extensions Supported Groups	0.48	Extensions Supported Groups	0.53	Extensions Supported Groups	0.54
TLS Version	0.21	TLS Version	0.20	Extensions Ec Point Formats	0.22	TLS Version	0.19
Extensions Ec Point Formats	0.12	Extensions Ec Point Formats	0.14	TLS Version	0.20	Extensions Ec Point Formats	0.18
Handshake TLS Version	0.12	Handshake TLS Version	0.11	Handshake Tls Version	0.09	Handshake Tls Version	0.09
Handshake Comp Methods	0.00	Handshake Comp Methods	0.00	Handshake Comp Methods	0.00	Handshake Comp Methods	0.00

Table 20: Top 10 probe fields for TLS ranked by mutual information (MI) value on on CompanyA dataset and CompanyB datasets.

CompanyA		CompanyB-22		CompanyB-23	
Field	MI Value	Field	MI Value	Field	MI Value
Handshake Time	0.85	Handshake Random Bytes	0.91	Handshake Random Bytes	0.40
Handshake Random Bytes	0.85	Handshake Time	0.91	Handshake Time	0.40
Handshake Cipher Suites	0.71	Handshake Session ID	0.76	Handshake Session ID	0.36
Handshake Session ID	0.67	Handshake Cipher Suites	0.75	Handshake Cipher Suites	0.16
Extensions Hash Algs	0.59	Handshake Comp Methods	0.58	Handshake Comp Methods	0.08
Extensions Supported Groups	0.42	TLS Version	0.17	TLS Version	0.03
Extensions Ec Point Formats	0.09	Handshake TLS Version	0.07	Handshake TLS Version	0.02
TLS Version	0.05	Extensions Ec Point Formats	0.00	Extensions Ec Point Formats	0.00
Handshake TLS Version	0.02	Extensions Supported Groups	0.00	Extensions Supported Groups	0.00
Handshake Comp Methods	0.00	Extensions Hash Algs	0.00	Extensions Hash Algs	0.00

updates. After encoding the top six fields from the first week, any newly observed option values from the second week's data are appended to the image's tail. Similarly, for subsequent datasets, newly observed option values are identified and incorporated into the updated image encoding. An illustrative example is provided in Fig. 7b.

F.4 The Encoding Strategy to Limit Image Dimensions

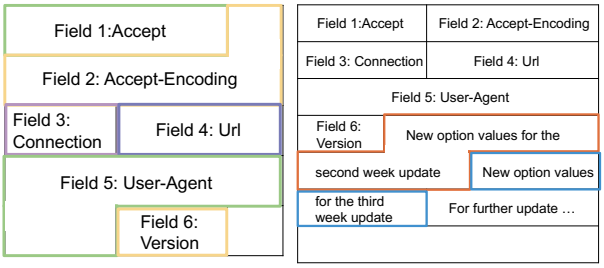
Given that the number of options in option-based fields can be large, we designed an encoding strategy to ensure that the image dimensions remain within 32×32 pixels. First, we exclude any options that appear less than 10 times in the current dataset. If the number of remaining options, plus the pixel usage for string and integer fields, exceeds 1024 pixels, we sort all the option values by

Table 21: The change in MI values for each week of the HTTP top-6 field compared to the first week.

HTTP	SelfDeploy24				SelfDeploy25		
Field	Week-1 MI	Peak MI difference vs. Week-1	Ratio of peak MI difference	Field	Week-1 MI	Peak MI difference vs. Week-1	Ratio of peak MI difference
User-Agent	1.28	-0.10	-7.6%	User-Agent	1.23	+0.33	26.9%
Accept	0.56	+0.06	10.8%	Accept	0.51	+0.18	35.2%
Url	0.52	-0.08	-15.2%	Version	0.39	-0.13	-32.9%
Accept-Encoding	0.52	-0.05	-9.2%	Accept-Encoding	0.38	+0.19	49.1%
Version	0.25	+0.05	20.6%	Url	0.38	+0.21	54.6%
Connection	0.23	+0.07	29.1%	Connection	0.21	+0.04	17.9%
Average			4.7%	Average			25.1%

Table 22: The change in MI values for each week of the TLS top-6 field compared to the first week.

TLS	SelfDeploy24				SelfDeploy25		
Field	Week-1 MI	Peak MI difference vs. Week-1	Ratio of peak MI difference	Field	Week-1 MI	Peak MI difference vs. Week-1	Ratio of peak MI difference
Handshake Random Bytes	1.63	+0.18	10.9%	Handshake Random Bytes	1.61	+0.22	13.5%
Handshake Time	1.63	+0.18	10.9%	Handshake Time	1.61	+0.22	13.5%
Handshake Cipher Suites	1.42	+0.09	6.3%	Handshake Cipher Suites	1.29	+0.15	12.0%
Extensions Hash Algs	1.14	-0.18	-15.9%	Extensions Hash Algs	1.08	-0.18	-16.6%
Handshake Session ID	1.06	+0.18	17.0%	Handshake Session ID	0.92	+0.22	23.4%
Extensions Supported Groups	0.47	+0.06	12.9%	Extensions Supported Groups	0.53	+0.09	17.0%
Average			7.0%	Average			10.5%



(a) Encoding the selected fields of an HTTP probe to an image.

(b) Updating image encoding for HTTP service probes.

Figure 7: Examples of encoding and updating image representations for HTTP probes.

frequency in descending order and select the top n options (where n = 1024 - pixels occupied by string and integer fields) to participate in the image construction.

G IOMatch Description

IOMatch [46] is equipped with a closed-world classifier that outputs the probability of a sample belonging to each known class. In addition to the closed-world classifier, IOMatch also employs a multi-binary classifier to predict the probability of a sample belonging to each known class. By combining the results from these two kinds of classifiers, IOMatch can determine the probabilities of a sample belonging to each known class as well as the unknown class. Furthermore, IOMatch improves model performance by utilizing unknown-class samples. It does so by generating recognition results for all unknown-class samples using the aforementioned classifiers, and these results are then treated as pseudo-labels (representing known or unknown-class samples) for training. Additionally, IOMatch simplifies the recognition process by constructing a unified open-set classifier that outputs probabilities for $K+1$ classes, where the first K classes correspond to known classes and the $K+1$ class represents the unknown class. This classifier is trained using samples with pseudo-labels as supervision signals.

H Number of Scanners on SelfDeploy Datasets

To demonstrate the proportionality of the training and testing sets, we take the SelfDeploy dataset as an example. As shown in Tab. 23,

Table 23: # of Scanners on SelfDeploy Datasets.

Scanning Org.	SelfDeploy24 # of scanners in training set	# of scanners in testing set	Scanning Org.	SelfDeploy25 # of scanners in training set	# of scanners in testing set
binaryedge	322	132	binaryedge	492	148
censys	240	265	censys	470	340
criminalip	7	6	criminalip	15	11
driftnet	254	184	driftnet	332	162
fofa	4	4	fofa	8	7
internet_census	106	53	internet_census	239	124
internettl	21	4	internettl	48	18
ipip	6	5	ipip	9	6
onyphe	27	16	onyphe	256	79
zoomeye	7	6	rapid7	10	5
shadowserver	499	243	shadowserver	553	282
shodan	32	22	shodan	31	18
stretchoid	991	518	stretchoid	967	793

the number of scanners from each organization in both sets under the 8:2 split remains on the same order of magnitude. This indicates that the testing set effectively represents the overall scanner distribution of the dataset.

I Hyperparameter Usage

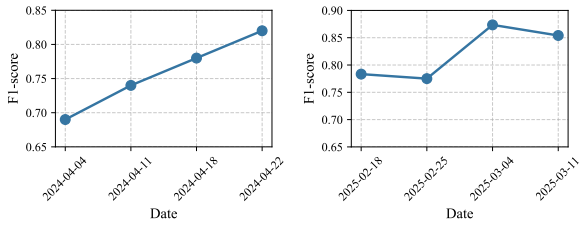
For each version of ScannerGrouper, we use consistent hyperparameters across all relevant datasets.

For the IOMatch model, the hyperparameter settings are as follows: (1) Backbone encoder: for all versions of ScannerGrouper, the backbone encoder is set to WRN-10-2 [69] (we follow the setting from the original IOMatch paper and chose the smallest hyperparameter version of the WRN network). (2) Learning rate: for all versions of ScannerGrouper, we set lr=0.03 (the same as the default setting of IOMatch). (3) Epoch: for ScannerGrouper-i, we train for 2 epochs when new data arrives to quickly update the model parameters. For ScannerGrouper-f, we set epoch=20. (4) Batch size: for ScannerGrouper-i, due to the small amount of incremental data, we set a smaller batchsize=32 for IOMatch. For ScannerGrouper-f, we set batchsize=64.

For the logistic regression model, we generally follow the default model setting as sklearn implemented. Additionally, to better learn the class with small samples, for all versions of ScannerGrouper, we use "balanced" mode in training (adjust weights inversely proportional to class frequencies), this mode to give large update weights to class with small samples when optimizing the hyperparameter of logistic regression model according to loss function. Following logistic regression, we apply an equidistant search to automatically

select the optimal threshold θ for scanner-level identification performance. To ensure precision, the step size for this search is set to 0.01.

J Performance of Incremental Training on Last Four Weeks of SelfDeploy Dataset



(a) SelfDeploy-24 **(b)** SelfDeploy-25

Figure 8: The performance of incremental training on last four weeks of SelfDeploy Dataset. In the plot, we use the 1st day of each week to indicate that week.

The ScannerGrouper-i performance for the last four weeks on both SelfDeploy-24 and SelfDeploy-25 datasets is shown in Fig. 8. The F1-score of the system increases with each weekly update.

We further evaluate ScannerGrouper-i on the supplemented SelfDeploy25-S dataset. As shown in Fig. 9, its identification performance exhibits a consistent upward trend over the last four weeks, reaching an F1-score of 0.89 in the final week. Detailed results for the final week are presented in Tab. 24.

Table 24: Performance of incremental training on the last week of SelfDeploy25-S dataset

	Precision	Recall	F1-score	# of scanners
binaryedge	0.86	0.95	0.9	97
criminalip	0.44	0.89	0.59	9
internet_census	1	0.86	0.92	69
ipip	1	0.17	0.29	6
leakix	-	0	0	5
onyphe	1	0.7	0.82	40
shadowserver	1	0.55	0.71	104
shodan	1	1	1	10
stretchoid	0.94	0.98	0.96	447
censys and driftnet(in unknown class)	0.83	0.94	0.89	288
weighted avg	0.91	0.9	0.89	1075

K Performance of One-time Training on CompanyA and CompanyB Datasets

Performance of One-time Training on CompanyA and CompanyB datasets are shown in Tabs. 25 and 26. Moreover, as noted in §3.2, the CompanyB dataset excludes packets without payloads, making it impossible to reconstruct darknet-style traffic, which consists solely of TCP SYN packets for each TCP connection. Consequently, baseline solutions based on darknet traffic cannot be applied to the CompanyB dataset.

L Performance of One-time Training under Different Training-Test Ratios

As shown in Tabs. 27 and 28, we evaluate ScannerGrouper-f under different training–test split ratios (7:3, 8:2, 9:1) on the SelfDeploy datasets. The identification performance remains relatively stable across these settings, with fluctuations within 5% when using the 8:2 split as the baseline.

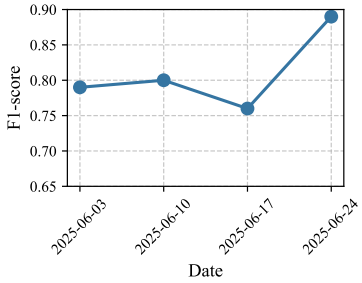


Figure 9: The performance of incremental training on last four weeks of SelfDeploy25-S Dataset. In the plot, we use the 1st day of each week to indicate that week.

However, the 7:3 split is not recommended, as the smaller training set may reduce identification accuracy—a trend observed in the SelfDeploy24 dataset. Likewise, the 9:1 split is suboptimal due to the limited number of scanners in the test set, which may not adequately represent overall performance. Based on this analysis, we recommend using an 8:2 training–test split.

M Performance of ScannerGrouper-f on Two Open Scanner Lists

We evaluate ScannerGrouper-f using two open scanner lists: one derived from darknet data [20], and another from a GitHub repository [5]. We first extract the overlapping IP addresses between these lists and the scanner IPs in our ScannerGrouper-f test set (8:2 split) respectively, which account for approximately 30% of the test set. We then assess how many of these overlapping IPs are correctly attributed to their organizations by ScannerGrouper-f and compute the corresponding F1-scores. The results are presented in Tabs. 29 and 30.

When evaluated on scanners overlapping with [20], ScannerGrouper-f achieves F1-scores of 0.78 and 0.80 on the SelfDeploy24 and SelfDeploy25 datasets, respectively. For scanners overlapping with the GitHub list [5], the scores reach 0.80 and 0.84. These results demonstrate that ScannerGrouper-f can reliably identify the organizations behind these IPs.

N K-means Clustering Parameter Automatic Selection

The parameter K (number of clusters) is automatically selected based on the silhouette coefficient. We calculate the maximum silhouette coefficient for each cluster and the total average silhouette coefficient. We then calculate the number of clusters whose maximum silhouette coefficient is greater than the total average silhouette coefficient, and divide this number by the total number of clusters K to obtain a ratio. A larger ratio indicates better clustering performance for each cluster. We iterate through K=3,...,10 and select the K with the largest ratio as the final number of clusters for clustering.

Table 25: Performance of one-time training on CompanyA dataset.

	ScannerGrouper-f			DarkVec			Kallitsis's Framework			# of scanners
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	
binaryedge	0.48	0.61	0.54	0.00	0.00	0.00	-	0.00	0.00	18
criminalip	1.00	0.88	0.93	0.00	0.00	0.00	1.00	0.04	0.08	24
intrinsec	1.00	1.00	1.00	0.10	0.07	0.08	-	0.00	0.00	56
leakix	1.00	1.00	1.00	0.06	0.06	0.06	-	0.00	0.00	18
onyphe	1.00	1.00	1.00	0.00	0.00	0.00	-	0.00	0.00	14
shadowserver	1.00	0.66	0.80	0.51	0.51	0.51	0.65	0.35	0.46	162
shodan	1.00	1.00	1.00	0.40	0.08	0.13	0.00	0.00	0.00	26
stretchoid	0.93	0.67	0.78	0.39	0.18	0.24	0.25	0.01	0.01	171
zoomeye	1.00	1.00	1.00	-	0.00	0.00	-	0.00	0.00	5
censys and driftnet (in unknown class)	0.86	0.83	0.84	0.53	0.69	0.60	0.22	0.99	0.36	115
weighted avg	0.94	0.77	0.84	0.37	0.32	0.33	0.40	0.28	0.19	609

Table 26: Performance of one-time training on CompanyB dataset.

	ScannerGrouper on CompanyB-22				ScannerGrouper on CompanyB-23			
	Precision	Recall	F1-score	# of scanners	Precision	Recall	F1-score	# of scanners
binaryedge	1.00	0.20	0.33	5	0.96	0.48	0.64	240
criminalip	0.96	0.69	0.80	35	1.00	0.82	0.90	34
internet_census	1.00	0.94	0.97	53	1.00	1.00	1.00	67
internettl	1.00	1.00	1.00	8	-	0.00	0.00	12
ipip	1.00	0.94	0.97	18	-	0.00	0.00	7
onyphe	1.00	0.86	0.92	7	1.00	1.00	1.00	14
shadowserver	0.98	0.94	0.96	338	1.00	0.92	0.96	295
shodan	1.00	1.00	1.00	29	1.00	0.98	0.99	40
stretchoid	0.81	0.35	0.49	49	1.00	0.94	0.97	447
zoomeye	1.00	1.00	1.00	1	1.00	1.00	1.00	5
censys and driftnet (in unknown class)	0.75	0.98	0.85	203	0.41	0.98	0.58	144
weighted avg	0.91	0.90	0.89	746	0.93	0.84	0.85	1305

Table 27: Performance of ScannerGrouper-f with different training-test ratios on the SelfDeploy24 dataset

	Train-Test Ratio7:3				Train-Test Ratio8:2				Train-Test Ratio9:1			
	Precision	Recall	F1-score	# of scanners	Precision	Recall	F1-score	# of scanners	Precision	Recall	F1-score	# of scanners
binaryedge	0.60	0.82	0.69	186	0.54	0.77	0.64	132	1.00	0.83	0.91	65
criminalip	1.00	0.33	0.50	6	1.00	0.33	0.50	6	1.00	0.40	0.57	5
internet_census	1.00	1.00	1.00	72	1.00	1.00	1.00	53	1.00	1.00	1.00	40
internettl	1.00	1.00	1.00	5	/	/	/	/	/	/	/	/
ipip	0.05	0.80	0.09	5	1.00	1.00	1.00	5	1.00	1.00	1.00	5
onyphe	1.00	0.86	0.93	22	1.00	0.88	0.93	16	1.00	0.78	0.88	9
shadowserver	1.00	0.54	0.70	318	1.00	0.54	0.70	243	1.00	0.52	0.68	134
shodan	1.00	1.00	1.00	27	1.00	1.00	1.00	22	1.00	1.00	1.00	16
stretchoid	0.93	0.87	0.90	666	0.90	0.91	0.91	518	0.91	0.88	0.90	328
zoomeye	0.29	0.33	0.31	6	0.20	0.17	0.18	6	/	/	/	/
censys and driftnet(in unknown class)	0.57	0.63	0.60	494	0.71	0.81	0.76	449	0.77	0.99	0.87	339
weighted avg	0.81	0.74	0.76	1807	0.83	0.80	0.80	1450	0.88	0.86	0.85	941

Table 28: Performance of ScannerGrouper-f with different training-test ratios on the SelfDeploy25 dataset

	Train-Test Ratio7:3				Train-Test Ratio8:2				Train-Test Ratio9:1			
	Precision	Recall	F1-score	# of scanners	Precision	Recall	F1-score	# of scanners	Precision	Recall	F1-score	# of scanners
binaryedge	0.84	0.87	0.86	172	0.85	0.91	0.88	148	0.72	0.86	0.78	98
criminalip	1.00	0.36	0.53	11	1.00	0.82	0.90	11	1.00	0.73	0.84	11
fofa	1.00	1.00	1.00	7	1.00	1.00	1.00	7	/	/	/	/
internet_census	1.00	0.92	0.96	148	1.00	0.94	0.97	124	1.00	0.95	0.97	74
internettl	0.44	0.96	0.61	24	0.49	1.00	0.66	18	1.00	1.00	1.00	8
ipip	1.00	0.57	0.73	7	1.00	0.50	0.67	6	1.00	0.50	0.67	6
onyphe	1.00	0.56	0.72	110	1.00	0.57	0.73	79	1.00	0.63	0.77	35
rapid7	1.00	1.00	1.00	5	1.00	1.00	1.00	5	1.00	1.00	1.00	5
shadowserver	1.00	0.60	0.75	358	0.99	0.55	0.71	282	1.00	0.54	0.70	146
shodan	1.00	1.00	1.00	19	1.00	1.00	1.00	18	1.00	1.00	1.00	11
stretchoid	0.93	0.98	0.95	1080	0.91	0.97	0.94	793	0.91	0.87	0.89	485
censys and driftnet(in unknown class)	0.75	0.91	0.82	596	0.77	0.92	0.84	502	0.68	0.89	0.78	310
weighted avg	0.89	0.87	0.87	2537	0.89	0.87	0.87	1993	0.86	0.83	0.83	1189

Table 29: Performance of ScannerGrouper-f on overlapping scanners of [20]

	SelfDeploy24					SelfDeploy25			
	Precision	Recall	F1-score	# of scanners		Precision	Recall	F1-score	# of scanners
internet_census	1.00	1.00	1.00	8	ipip	0.00	0.00	0.00	1
onyphe	1.00	0.83	0.91	12	onyphe	0.00	0.00	0.00	3
stretchoid	0.50	1.00	0.67	1	rapid7	1.00	1.00	1.00	5
shadowserver	1.00	0.45	0.62	67	shadowserver	1.00	0.25	0.40	60
shodan	1.00	1.00	1.00	16	shodan	1.00	1.00	1.00	12
censys and driftnet(in unknown class)	0.86	0.73	0.79	317	censys and driftnet(in unknown class)	0.87	0.87	0.87	323
weighted avg	0.89	0.70	0.78	421	weighted avg	0.88	0.78	0.80	404

Table 30: Performance of ScannerGrouper-f on overlapping scanners of [5]

	SelfDeploy24					SelfDeploy25			
	Precision	Recall	F1-score	# of scanners		Precision	Recall	F1-score	# of scanners
criminalip	1.00	0.33	0.50	6	criminalip	1.00	0.82	0.90	11
internet_census	1.00	1.00	1.00	37	internet_census	1.00	1.00	1.00	8
internettl	1.00	1.00	1.00	4	internettl	0.49	1.00	0.66	18
ipip	1.00	1.00	1.00	5	ipip	1.00	0.50	0.67	6
rapid7	/	/	/	/	rapid7	1.00	1.00	1.00	5
shadowserver	1.00	0.47	0.64	149	shadowserver	1.00	0.48	0.64	160
shodan	1.00	1.00	1.00	21	shodan	1.00	1.00	1.00	15
censys and driftnet(in unknown class)	0.84	0.81	0.83	449	censys and driftnet(in unknown class)	0.89	0.92	0.90	502
weighted avg	0.90	0.75	0.80	671	weighted avg	0.91	0.82	0.84	725

Table 31: Summary of still-unknown DNS probe clusters

Cluster	Scanning Type	# of Probes	Description
0	Ad1 scan	287	97.2% of probes with ad=1 (often used in DNS repsonse message).
1	Special queries	503	Common used Query Name: VERSION.BIND (37.6% probes), version.bind (20.7% probes), N/A (9.1% probes).
2	Scattered scanner	525	Common used Query Name: sl (68.4% probes), collectd.org (8.0% probes), higi.com (7.0% probes).

O Identification Result Analysis of DNS Probes from Unknown-Class Scanners

For DNS probes, three clusters are identified, as shown in Tab. 31. Cluster 0 is characterized by the 'ad' flag being set to 1, typically used in DNS responses to indicate authenticated data (as defined in RFC 2535 [8]). Cluster 1 contains special requests such as 'version.bind' (used to retrieve BIND version information) and empty domain names. Cluster 2 consists of scattered requests, with common domains including 'sl', 'collectd.org', and 'higi.com'. In summary, by clustering probes from unknown-class scanners still identified as unknown, ScannerGrouper effectively groups them by payload features, aiding security analysts in understanding ongoing scanning activities.

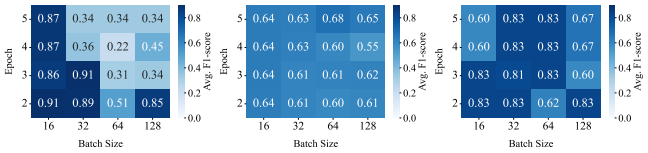
P Hyperparameter Analysis

P.1 IOMatch hyperparameter analysis

We use the IOMatch model for probe-level open-set identification and analyze how training hyperparameters affect its performance. The key hyperparameters include the number of training epochs (e.g., 15, 20, 25, 30) and batch size (e.g., 16, 32, 64, 128). We evaluate the performance of IOMatch separately for ScannerGrouper-i and ScannerGrouper-f.

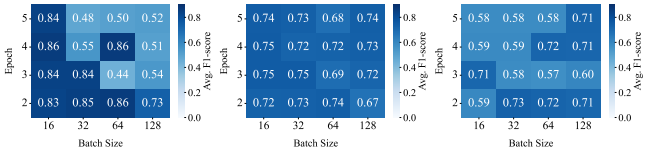
Hyperparameter Analysis for ScannerGrouper-i. "Epoch" refers to the number of training iterations per update. To reduce overfitting, we use a small number of epochs: 2, 3, 4, or 5. As shown in Figs. 10 and 11, models trained with fewer epochs (2 or 3) achieve higher F1-scores. Medium batch sizes (32 or 64) also yield consistently better performance. IOMatch performs similarly on the SelfDeploy-24 and SelfDeploy-25 datasets, and the setting epoch=2, batch size=32 used in ScannerGrouper-i provides stable performance across both.

Hyperparameter Analysis for ScannerGrouper-f. We conduct a similar analysis for ScannerGrouper-f across different datasets and observe trends consistent with those of ScannerGrouper-i: models trained with fewer epochs (e.g., 15 or 20) and medium batch sizes (32 or 64) achieve higher F1-scores. As shown in Figs. 12 and 13, the parameter setting epoch=20, batch size=64 used in ScannerGrouper-f provides stable performance across both the SelfDeploy-24 and SelfDeploy-25 datasets.



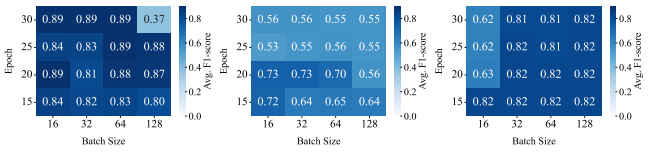
(a) HTTP service **(b)** TLS service **(c)** DNS service

Figure 10: Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-i (SelfDeploy-24).



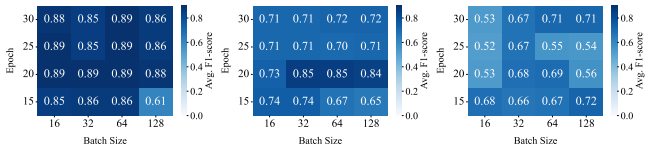
(a) HTTP service **(b)** TLS service **(c)** DNS service

Figure 11: Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-i (SelfDeploy-25).



(a) HTTP service **(b)** TLS service **(c)** DNS service

Figure 12: Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-f (SelfDeploy-24).

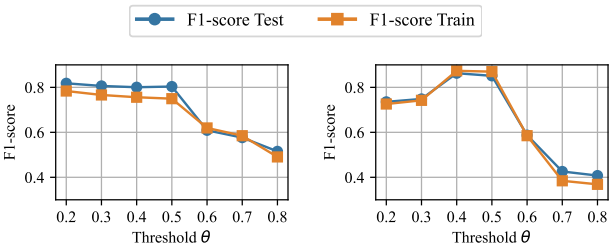


(a) HTTP service **(b)** TLS service **(c)** DNS service

Figure 13: Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-f (SelfDeploy-25).

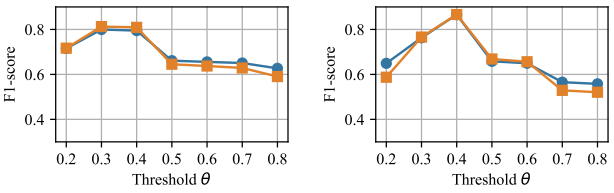
P.2 Threshold θ in unknown-class sample identification

In the Result Aggregation Module, we propose an equidistant search strategy on the training set to automatically determine the threshold for identifying unknown-class scanners. To validate its effectiveness, we evaluate whether the selected threshold generalizes to the testing set. We analyze the impact of different thresholds on scanner-level identification performance for both ScannerGrouper-i and ScannerGrouper-f. As shown in Figs. 14 and 15, performance remains consistent across training and testing sets when using the same threshold. These results confirm that the threshold θ determined from the training set generalizes well to the testing set.



(a) SelfDeploy-24 **(b)** SelfDeploy-25

Figure 14: ScannerGrouper-i performance with different threshold θ on the last week of SelfDeploy-24 and SelfDeploy-25 dataset.



(a) SelfDeploy-24 **(b)** SelfDeploy-25

Figure 15: ScannerGrouper-f performance with different threshold θ.

Q Performance of ScannerGrouper-f on Different Unknown-Class Scanner Ratios

To assess the impact of unknown scanner proportions on training performance, we construct training sets by randomly downsampling unknown scanners to 25%, 50%, and 75% of the original proportion. We then retrain ScannerGrouper-f on each set and evaluate its identification performance. As shown in Tabs. 32 and 33, the F1-scores across these settings vary by only 4%–11% compared to the baseline using 100% unknown-class scanners (Tabs. 7 and 8). These results demonstrate the robustness of our system to imbalanced distributions of unknown-class scanners.

Table 32: Performance of one-time training with different unknown-class scanner ratios on SelfDeploy24 dataset.

	Unknown-class scanner ratio of 0.25			Unknown-class scanner ratio of 0.50			Unknown-class scanner ratio of 0.75			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	# of scanners
binaryedge	1.00	0.77	0.87	0.54	0.77	0.64	1.00	0.77	0.87	132
criminalip	1.00	0.33	0.50	1.00	0.33	0.50	1.00	0.33	0.50	6
internet_census	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	53
ipip	0.06	1.00	0.12	1.00	1.00	1.00	1.00	1.00	1.00	5
onyphe	1.00	0.88	0.93	1.00	0.88	0.93	1.00	0.88	0.93	16
shadowserver	0.84	0.83	0.84	1.00	0.54	0.70	1.00	0.54	0.70	243
shodan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	22
stretchoid	0.89	0.95	0.92	0.93	0.80	0.86	0.94	0.77	0.85	518
zoomeye	1.00	0.83	0.91	1.00	0.83	0.91	1.00	0.83	0.91	6
censys and driftnet(in unknown class)	0.77	0.36	0.49	0.61	0.80	0.69	0.64	0.99	0.78	449
weighted avg	0.86	0.73	0.77	0.81	0.76	0.76	0.87	0.81	0.81	1450

Table 33: Performance of one-time training with different unknown-class scanner ratios on SelfDeploy25 dataset.

	Unknown-class scanner ratio of 0.25			Unknown-class scanner ratio of 0.50			Unknown-class scanner ratio of 0.75			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	# of scanners
binaryedge	0.75	0.90	0.82	0.66	0.89	0.76	0.83	0.89	0.86	148
criminalip	1.00	0.82	0.90	1.00	0.73	0.84	1.00	0.82	0.90	11
fofa	0.78	1.00	0.88	1.00	1.00	1.00	1.00	1.00	1.00	7
internet_census	1.00	0.94	0.97	1.00	0.94	0.97	1.00	0.94	0.97	124
internettl	0.46	1.00	0.63	0.69	1.00	0.82	1.00	1.00	1.00	18
ipip	1.00	0.50	0.67	1.00	0.50	0.67	1.00	0.50	0.67	6
onyphe	1.00	0.57	0.73	1.00	0.57	0.73	1.00	0.57	0.73	79
rapid7	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	5
shadowserver	1.00	0.53	0.69	0.79	0.55	0.65	0.99	0.54	0.70	282
shodan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	18
stretchoid	0.91	0.97	0.94	0.91	0.97	0.94	0.90	0.84	0.87	793
censys and driftnet(in unknown class)	0.75	0.85	0.80	0.73	0.77	0.75	0.50	0.52	0.51	502
weighted avg	0.88	0.85	0.85	0.84	0.83	0.83	0.82	0.72	0.76	1993