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

This appendix provides...

ACM Reference Format:
. 2025. Appendix — ScannerGrouper: A Generalizable and Effective Scanning Organization Identification System Toward the Open World. 1, 1 (September 2025), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 The Labeling Method for Cyberspace Search Engine Scanners in ??

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 **??** 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.

1.1 The Proposed Labeling Method

The workflow of the proposed method is illustrated in Fig. 1. 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.

1.2 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 **??**. 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, all of which had the domain name censys.io.

Author’s Contact Information:

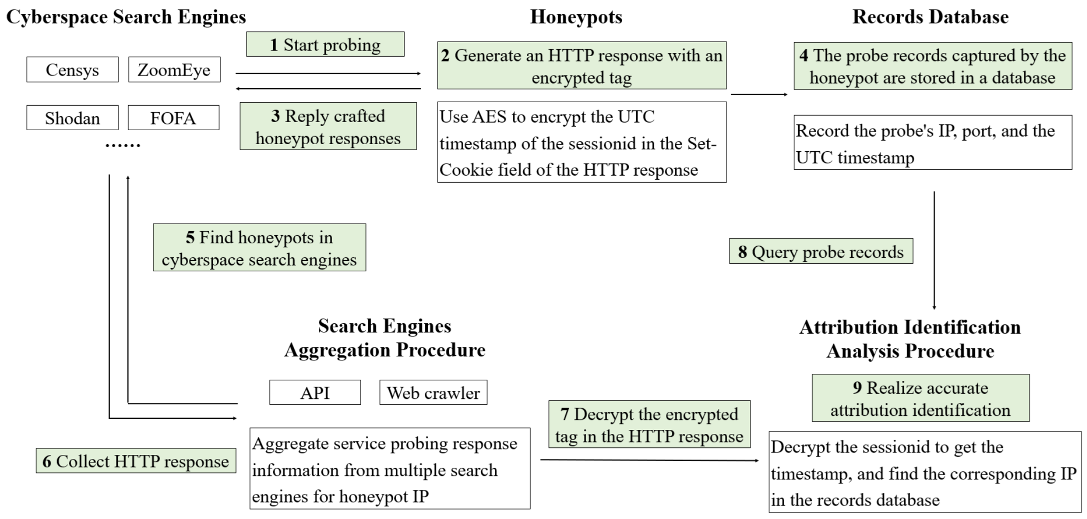

Cyberspace Search Engines
Censys
ZoomEye
Shodan
FOFA
......
1 Start probing
3 Reply crafted honeypot responses
Honeypots
2 Generate an HTTP response with an encrypted tag
Use AES to encrypt the UTC timestamp of the sessionid in the Set-Cookie field of the HTTP response
Records Database
4 The probe records captured by the honeypot are stored in a database
Record the probe's IP, port, and the UTC timestamp
5 Find honeypots in cyberspace search engines
8 Query probe records
Search Engines Aggregation Procedure
API
Web crawler
Aggregate service probing response information from multiple search engines for honeypot IP
6 Collect HTTP response
7 Decrypt the encrypted tag in the HTTP response
Attribution Identification Analysis Procedure
9 Realize accurate attribution identification
Decrypt the sessionid to get the timestamp, and find the corresponding IP in the records database

Fig. 1. The workflow of the proposed labeling method applied in **??**.

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.

1.3 Ethics Considerations

The method for labeling cyberspace search engine scanners (**??**, Section 1) 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.

2 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 Table 1.

Table 1. 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

3 Statistical Analysis of the Fields of DNS and TLS Probes

In **??**, 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

Table 2. 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 3. 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%

statistical analysis of the TLS handshake cipher suites (Table 2) and the DNS query name (Table 3) fields here. As shown in the tables, the first service probes of TLS and DNS services also contain similar distinguishing fields. 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.

4 Field Selection Results for HTTP and TLS Probes

We analyze the first probes of HTTP and TLS using the mutual information method described in **??**, with the results presented in Tables 4 to 7. 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.

Table 4. 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 5. 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 6. 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

5 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 Tables 8 and 9.

Table 7. 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

Table 8. 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 9. 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%

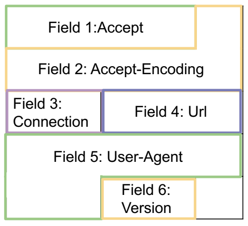
The average MI change for the top-6 HTTP fields ranges from 4.7% to 25.1%, which is insufficient to alter the field ranking. For TLS, the average change is even smaller, ranging from 7.0% to 10.5%, and the top-6 fields remain unchanged.

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

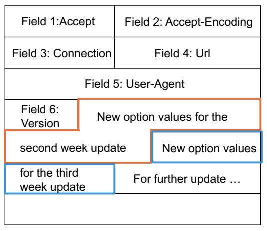
6 Image Encoding

6.1 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.



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



(b) Updating image encoding for HTTP service probes.

Fig. 2. Examples of encoding and updating image representations for HTTP probes.

6.2 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 **??**.
- **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[1].
- **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.

6.3 Examples of Encoded Images

We apply the method described in **??** 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. 2a. As described in the module in **??**, during the time periods following the first week, the image format may require 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. 2b.

6.4 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

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

pixels, we sort all the option values by 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.

7 IOMatch Description

IOMatch [7] 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.

8 Number of Scanners on SelfDeploy Datasets

Table 10. # of Scanners on SelfDeploy Datasets.

	SelfDeploy24			SelfDeploy25	
Scanning Org.	# of scanners in training set	# of scanners in testing set	Scanning Org.	# 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

To demonstrate the proportionality of the training and testing sets, we take the SelfDeploy dataset as an example. As shown in Table 10, 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.

9 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 [8] (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.

10 Identification Performance Analysis in Incremental Training Experiments

Table 11. 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 drift-net (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

Table 12. Performance of incremental training on the last week of SelfDeploy-25 dataset.

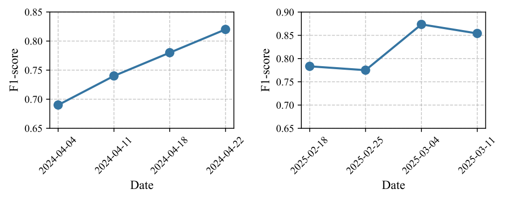
	ScannerGrouper-i			i-DarkVec			# of scanners
	Precision	Recall	F1-score	Precision	Recall	F1-score	
binaryedge	1.00	0.65	0.79	0.78	0.15	0.25	92
criminalip	1.00	0.82	0.90	0.09	0.09	0.09	11
internet_census	1.00	0.96	0.98	0.58	0.10	0.17	72
internettl	0.54	1.00	0.70	0.00	0.00	0.00	7
ipip	-	0.00	0.00	-	0.00	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 drift-net (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

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 Tables 11 and 12.

Here, we give detailed explanation of why some organizations underperform. As mentioned in original paper, 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.



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

Fig. 3. 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.

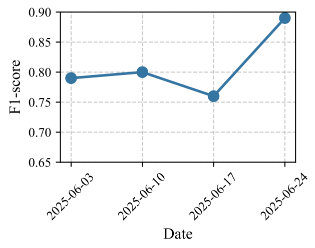


Fig. 4. The performance of incremental training on last four weeks of SelfDeploy25-S Dataset. In the plot, we use the 1st day of each week to indicate that week.

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

The ScannerGrouper-i performance for the last four weeks on both SelfDeploy-24 and SelfDeploy-25 datasets is shown in Fig. 3. 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. 4, 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 Table 13.

Table 13. 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

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

Performance of One-time Training on CompanyA and CompanyB datasets are shown in Tables 14 and 15. Moreover, as noted in **??**, 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.

Table 14. 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 drift-net (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 15. 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 drift-net (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

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

Table 16. 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

As shown in Tables 16 and 17, 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.

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.

Table 17. 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

14 Performance of ScannerGrouper-f on Two Open Scanner Lists

We evaluate ScannerGrouper-f using two open scanner lists: one derived from darknet data [4], and another from a GitHub repository [1]. 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 Tables 18 and 19.

When evaluated on scanners overlapping with [4], 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 [1], the scores reach 0.80 and 0.84. These results demonstrate that ScannerGrouper-f can reliably identify the organizations behind these IPs.

Table 18. Performance of ScannerGrouper-f on overlapping scanners of [4]

	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 19. Performance of ScannerGrouper-f on overlapping scanners of [1]

	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

15 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.

16 Identification Result Analysis of TLS Probes from Unknown-Class Scanners

Table 20. 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, inclduing 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 TLS probes, eight clusters are identified, as summarized in Table 20. 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 [5, 6], Censys uses the open-source scanning tool ZGrab [3], 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.

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

For DNS probes, three clusters are identified, as shown in Table 21. 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 [2]). 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.

Table 21. 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).

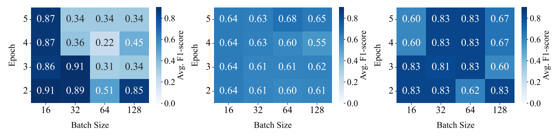
18 Hyperparameter Analysis

18.1 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.

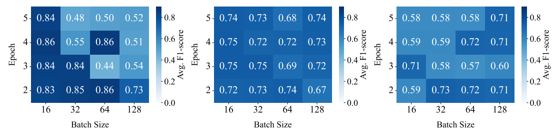
Hyperparameter Analysis for ScannerGrouper-i. "Epoch" refers to the number of training iterations per update. To reduce overfitting, we use a small number of epochs: 2, 3, 4, or 5. As shown in Figs. 5 and 6, models trained with fewer epochs (2 or 3) achieve higher F1-scores. Medium batch sizes (32 or 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. 7 and 8, the parameter setting epoch=20, batch size=64 used in ScannerGrouper-f provides stable performance across both the SelfDeploy-24 and SelfDeploy-25 datasets.



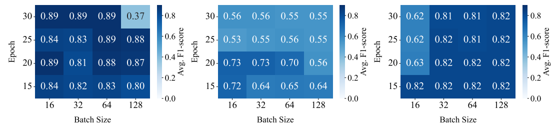
(a) HTTP service **(b)** TLS service **(c)** DNS service

Fig. 5. Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-i (SelfDeploy-24).



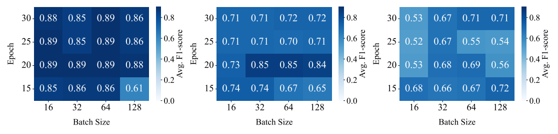
(a) HTTP service **(b)** TLS service **(c)** DNS service

Fig. 6. 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

Fig. 7. Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-f (SelfDeploy-24).

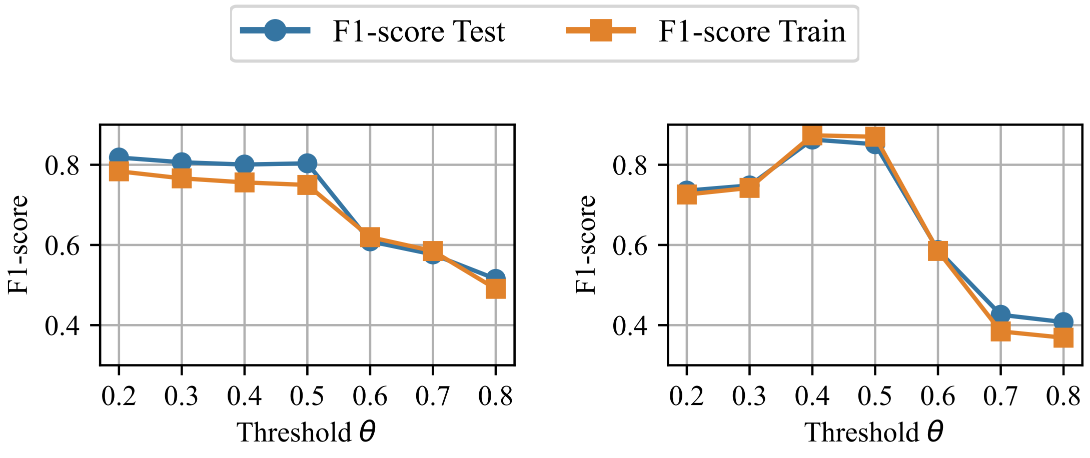


(a) HTTP service **(b)** TLS service **(c)** DNS service

Fig. 8. Performance with different values of epoch and batch size for IOMatch in ScannerGrouper-f (SelfDeploy-25).

18.2 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. 9 and 10, 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.

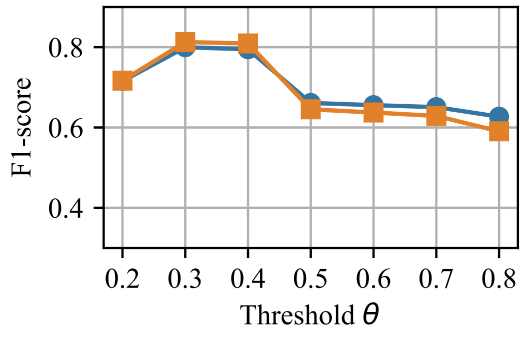


(a) SelfDeploy-24 **(b)** SelfDeploy-25

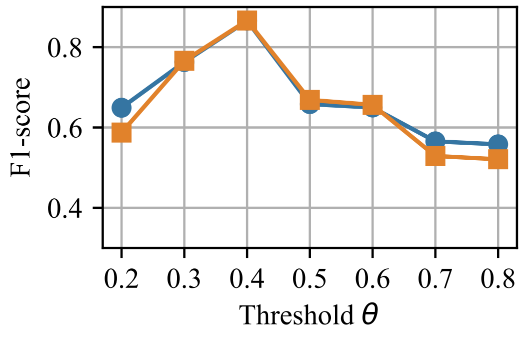
Fig. 9. ScannerGrouper-i performance with different threshold θ on the last week of SelfDeploy-24 and SelfDeploy-25 dataset.

19 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 Tables 22 and 23, the F1-scores across these settings vary by only 4%–11% compared to the baseline using 100% unknown-class scanners (**??**). These results demonstrate the robustness of our system to imbalanced distributions of unknown-class scanners.



(a) SelfDeploy-24



(b) SelfDeploy-25

Fig. 10. ScannerGrouper-f performance with different threshold θ.

Table 22. 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 23. 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

References

[1] IP list of known scanners from Connie-Wild. https://github.com/Connie-Wild/scanner-ip-list.

[2] RFC2535. https://www.rfc-editor.org/rfc/rfc2535.html.

[3] ZGrab 2.0. https://github.com/zmap/zgrab2. Accessed on 2025-1.

[4] M. Collins, A. Hussain, and S. Schwab. Identifying and Differentiating Acknowledged Scanners in Network Traffic. In *IEEE EuroS&PW*, pages 567–574, 2023.

[5] 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.

[6] 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.

[7] 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.

[8] Sergey Zagoruyko. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.