Chapter 21

Smart Internet Probing: Scanning Using Adaptive Machine Learning

Armin Sarabi,[1*] Kun Jin,[2] and Mingyan Liu[3]

[1]*Deparment of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109, USA*
[2]*Deparment of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109, USA*
[3]*Deparment of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109, USA*

*Corresponding author; arsarabi@umich.edu

Abstract: Network scanning is widely used to assess security postures of hosts/networks, discover vulnerabilities, and study Internet trends. However, scans can generate large amounts of traffic, and efficient probing of IPv6 hosts (where global scans are infeasible) is an outstanding problem. In this chapter, we develop a framework for efficient Internet scans using machine learning, by preemptively detecting and avoiding the scanning of inactive hosts. We evaluate this framework over global scans of the IPv4 space over 20 ports, and show that using location and ownership information we can reduce the bandwidth of scans by 26.7-72.0%, while discovering 90-99% of active hosts. We then evaluate a sequential method by gradually adding information obtained from scanned ports to adaptively predict the

remaining port responses, yielding 47.4-83.5% of bandwidth savings at the same true positive rates. Our framework can be used to lower the bandwidth consumption of scans and increase their hit rate, thereby reducing their intrusive nature and enabling efficient discovery of active devices.

Keywords: Network Scanning, Vulnerability Assessment, Prediction and Forecasting

21.1. Introduction

Network scanning is a widely studied topic, ranging from partial scans of the Internet [1, 2], to global scans of the IPv4 address space [3, 4, 5]. This has lead to the development of network scanning tools such as ZMap [3] and NMap [6], which have provided researchers with large amounts of information on arbitrary Internet hosts. Data resulting from network scans have been used in a wide range of security studies, e.g., to probe and characterize machines utilized in the Mirai botnet [7], to gauge the security posture of networks for cyber-risk forecasting [8], and to study hosts susceptible to the Heartbleed vulnerability [9]. Internet scanning is a crucial tool for giving visibility into the security of Internet-connected entities, as it can measure the attack surface of networks by revealing (potentially misconfigured/vulnerable) networked devices accessible on the public Internet. Additionally, network scanning has been used in many Internet measurement studies, including studies for examining trends and adoption rates of different technologies [10, 11, 12], to detect discoverable hosts and to categorize them (e.g., IoT devices) [13, 14, 15, 16], and to map network topologies [17, 18, 19].

However, the current approach to Internet scanning involves exhaustively sending probes to every scanned IP address (possibly the entire IPv4 address

space), regardless of whether the target host is reachable on the public Internet. Therefore, network scans strain the targeted networks/hosts, as they can produce large amounts of traffic, especially when multiple ports of a host are being probed. In addition, global scanning of the IPv6 address space is not feasible using such exhaustive methods, forcing researchers to come up with techniques for producing scan targets, in order to obtain a representative subset of publicly discoverable hosts for characterizing and studying IPv6 networks [2, 20].

Note that a large majority of probes sent during a scan will go unanswered, since most IP addresses are *inactive*, meaning that they are not running any Internet-facing service, or do not respond to outside probes. This gets more pronounced as multiple ports are scanned, since a single *active* IP address may only have a few number of active/open ports, i.e., ports that respond to probes. In fact, the Censys database [4] which contains global scans of IPv4 address space across 37 different port contains roughly 161 million records in its snapshots on 1/1/2019, meaning that ∼94.3% of the announced Border Gateway Protocol (BGP) prefixes (∼170 /8 blocks, or ∼2.8 billion addresses) are inactive, or do not respond to requests on any of the scanned ports. For active IP addresses, the corresponding hosts are only responding to requests for 1.8 ports on average.

Given the above context, a question arises as to whether probes can be sent in a more judicious manner, i.e., if an Internet scanner can preemptively predict whether a probe will be left unanswered, and thus refrain from sending it in the first place. This would then lead to an overall reduction in bandwidth for an Internet scan, reducing its intrusive nature. Moreover, reducing the bandwidth of Internet scans allows one to probe IP addresses at a faster rate, which in turn

increases the hit rate for discovering active hosts. This is important as more hosts are migrating to the IPv6 address space [21, 22], where global scanning is not feasible by existing methods, and increasing the rate at which hosts can be probed (e.g., using a hitlist [2, 20]) will lead to more discovered active devices in this space.

Motivated by the above, we develop a framework that leverages machine learning to predict whether a given host will respond to probes on different ports. In a machine learning setting, port responses can be seen as a set of binary labels and we can use classification models to perform these multi-label predictions. In situations where we can predict port responses accurately, we are able to save unnecessary probes and improve the efficiency of the scanner. Our first set of features for prediction include geolocation and autonomous system (AS) information of IP addresses. These features can provide a machine learning model with information about the underlying network, e.g., to distinguish between residential, educational, and hosting networks, which can help the scanner adjust its probes accordingly. As an example we observe that hosts belonging to residential networks are more likely to respond to requests on port 7547 (CWMP), a protocol commonly used by routers/modems.

Moreover, we observe dependencies between the responses of different ports, i.e., whether a host is responding to probes on the scanned ports. For instance, if we receive a response from a host on port 443 (HTTPS), then it is likely that it will also respond to probes on port 80 (HTTP), since both ports are used to serve web content to clients. We also measure and observe high correlation between ports associated with mail servers. We can then leverage these correlations to improve classification accuracy by scanning different ports of an IP address in sequence, and appending the obtained labels/features result-

ing from each probe to the features vectors for predicting the remaining port responses. We observe different levels of correlation between different ports, in other words, some port responses are highly dependent on one another, while others act more independently. Therefore the efficacy of the scanner is highly dependent on the order in which ports are scanned. For this we develop an adaptive technique for generating an optimal order for probing different ports of a host.

We evaluate our technique over scans collected over 20 different ports between January and May of 2019 by the Censys database and show that using only geolocation and autonomous system information, we can achieve bandwidth savings (the reduction in number of probes) of 26.7-72.0%, while still achieving 90-99% true positive rates for detecting active ports. We further train and evaluate a sequence of classifiers, gradually adding the information obtained from previous scans for predicting the responses of remaining ports, and show that we can increase bandwidth savings to 47.4-83.5% at the same true positive rates. We show that using only a single feature from probed ports, i.e., whether the host has responded on said port, is sufficient for achieving the aforementioned boost in performance, while minimizing the computational overhead for performing predictions. Adding more information, e.g., features resulting from a stateful, or application layer probe [23], only results in marginal benefits, while significantly increasing the computation required for performing and processing the information from such scans. Additionally, capturing the demographics of networks through geolocation and AS information is crucial for sequential scans, as conducting predictions based on only port responses results in poor performance.

Our main contribution are summarized as follows:

- We develop and evaluate two methods for conducting *smart* network scans by predicting whether a host will respond to probes on a certain port: (1) parallel scans using a priori (i.e., location and AS) attributes of hosts as features for prediction, and (2) sequential scans that take advantage of cross-protocol dependencies, using the result of one scan as features, in order to adaptively adjust the probes for consequent scans.
- For sequential scans, we develop a novel technique to find an optimal order for scanning ports. We achieve this by first training a set of classifiers, measuring the contribution of one port for predicting the responses of remaining ports, and probing ports by decreasing order of their importance.
- We evaluate this framework by simulating it over global scans of the public Internet conducted between January and May of 2019 over 20 ports, and show that we can reduce the number of probes by 26.7-72.0% (47.4-83.5%) using parallel (sequential) scans with negligible computational overhead, while maintaining a 90%-99% true positive rate for discovering active devices across all ports. We also examine the coverage of scans over vulnerable/misconfigured IP addresses, and observer high true positive rates (>98.5%) along these subpopulations of active IPs, suggesting that our method can be reliably used for discovering vulnerable devices and assessing networks' security posture.

The remainder of this chapter is organized as follows. In Section 21.2 we go over the data sets used in our study, and how we preprocess the data to prepare it for our analysis. In Section 21.3 we go over the models used in our study, and define metrics for evaluating the performance of our framework. Section 21.4 details our methodology for combining machine learning models with network scanners under different scenarios. In Section 21.5 we evaluate

the performance of our technique for reducing the number of probes sent by a network scanner. We discuss our results in Section 21.6, go over related works in Section 21.7, and conclude in Section 21.8.

21.2. Data Sets

In this section we go over the database used for obtaining global scans of the Internet, and explain in detail how we curate and process measurements for evaluating the performance of our technique in the real-world.

21.2.1. Global Internet scans

For obtaining scans of the public Internet, we use Censys [4], a database containing results from global scans of the IPv4 address space across 37 different ports for our observation window between January and May of 2019. Each snapshot in the Censys database contains records (stored using JSON documents) on discoverable hosts, i.e., hosts that respond to at least one of the sent probes. For this study we use snapshots corresponding to the following five dates from 2019: 1/1, 2/1, 3/1, 4/1, and 5/1. We mainly use the snapshot from 1/1/2019 to evaluate our framework, but use the more recent snapshots to measure the performance degradation of our models over time in Section 21.6.3.

Each record in a Censys snapshot contains attributes that have been extracted from port responses, such as headers and banners, SSL/TLS certificates, and even highly granular attributes such as the choice of ciphers for encryption and specific vulnerabilities probed for by Censys (e.g., Heartbleed).

In addition, Censys also reports geolocation and ownership (AS) information about hosts in their database provided by the MaxMind GeoLite2 database [24], Merit Network [25], and Team Cymru [26]. We use these records for extracting features of Internet hosts, and training/evaluating our machine learning models for predicting port responses.

21.2.2. Data curation

To evaluate our framework, we generate information for randomly drawn IP addresses in the following manner. We first select 17.5 million random IP address from announced Border Gateway Protocol (BGP) prefixes corresponding to each snapshot date, captured by CAIDA from Routeviews data [27], about 170 /8 blocks or ∼2.8 billion addresses. This is done to remove reserved and private IP addresses, as well as address spaces not announced on BGP. For each selected IP address, we then check whether it has a corresponding record in a Censys snapshot. For IP addresses that do have a Censys record (i.e., an *active* IP), we append the Censys record to our curated data set. For addresses that do not have a corresponding Censys record (i.e., an *inactive* IP), we query its geolocation and autonomous system information from Censys using the following technique. We first find the two closest active IPs in Censys to the inactive IP, i.e., one with a smaller IP address, and one with a larger IP address. We then find the smallest Classless Inter-Domain Routing (CIDR) blocks that contain the inactive IP address and each of its active neighbors. If the corresponding CIDR block for one neighbor is smaller than the other, we then decide that the inactive IP belongs to the same network as that neighbor, and use the AS and geolocation properties of the corresponding neighbor

for the inactive IP. If all three addresses are contained within the same CIDR block, then we copy AS and geolocation information from the closest neighbor, or the one with a larger IP address if both neighbors have the same distance to the inactive IP address.

The above procedure yields about one (16.5) million randomly drawn active (inactive) IP addresses from each snapshot (note that only ∼5.7% of all IP addresses are active according to Censys probes). We further sub-select one million addresses from the inactive IPs to obtain a more balanced data set, resulting in a curated data set containing roughly one million active and one million inactive IPs for each snapshot. We use these data sets for training and evaluating the performance of our scanning techniques.

21.2.3. Data processing

Records from the Censys database are stored using JSON documents with deeply nested fields, containing location and ownership (AS) properties, as well as attributes extracted from parsed responses, including headers, banners, certificate chains, and so on. However, while these documents contain a wide range of characteristics about Internet hosts, the information cannot be fed into a classification model out of the box, and we need to convert these documents to numerical feature vectors for analysis by a machine learning model.

JSON documents follow a tree-like structure, allowing different fields to be nested inside one another, e.g., properties regarding the location of a host, including country, city, latitude, and longitude. Therefore, simply extracting tokens from the string corresponding to a JSON document fails to recognize its structure, and does not provide any information about the field from which

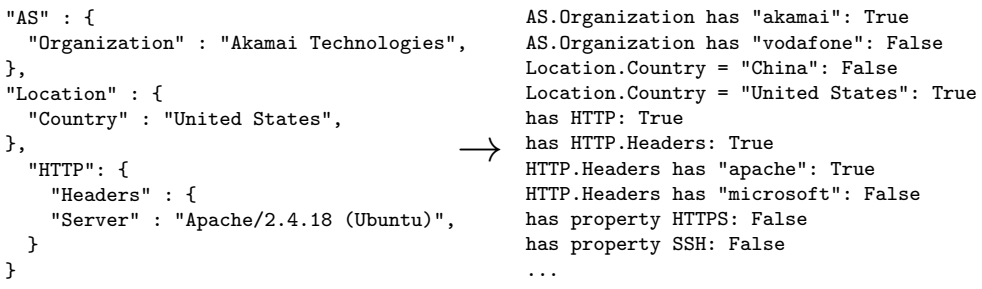


Figure 21.1: An example JSON document with nested fields (left), and binary features extracted for analysis in a machine learning model (right).

the token was extracted.

To address the above problem, we use the approach developed by Sarabi and Liu [28] to extract high-dimensional binary features vectors from these documents. This feature extraction algorithm first learns the schema [29] of JSON documents in the Censys database by inspecting a number of sample documents, and then extracts binary features from each field according to the learned schema. This then produces features that can be attributed to fields of the original JSON documents, and are extracted according to the data type of those fields (i.e., string, categorical, boolean, etc.). Furthermore, for optional fields we can also generate features that reflect their existence in a document, e.g., open ports, or if a host is returning headers/banners for different protocols.

Figure 21.1 shows an example of how a JSON document can be transformed into a binary vector representation using this approach. Note that each generated feature is assigned to a certain field of the original JSON document, allowing us to separate features extracted from location and ownership (AS) information, as well as features extracted from different port responses. This allows us to gradually add the information of scanned ports to our models for performing predictions of remaining ports.

Table 21.1: Sections from Censys documents, number of features generated from each section, and frequencies of active (open) ports among active IP addresses; note that 5.7% of all IPv4 addresses are active according to Censys measurements.

Section	# of features	Frequency
Geolocation	1513	N/A
Ownership (AS)	425	N/A
21/ftp	141	6.5%
22/ssh	622	11.6%
23/telnet	125	2.4%
25/smtp	875	4.1%
53/dns	41	5.1%
80/http	810	33.3%
110/pop3	790	3.0%
143/imap	827	2.8%
443/https	3263	35.3%
445/smb	1	1.8%
465/smtp	588	2.5%
587/smtp	993	3.4%
993/imaps	808	2.7%
995/pop3s	792	2.6%
2323/telnet	20	0.4%
3306/mysql	201	3.0%
5432/psql	68	0.4%
7547/cwmp	202	12.5%
8080/http	417	12.7%
8888/http	157	4.2%

We train the feature extraction model from [28] on one million randomly drawn records from the 1/1/2019 Censys snapshot (chosen independently from the dataset detailed in Section 21.2.2), producing 14 443 binary features extracted from 37 different ports. To control the number of generated features, we impose a limit of 0.05% on the sparsity of extracted features. These features are in the form of tags assigned to a host, e.g., if a host responds to probes on a certain port, if it belongs to a particular country, or if we observe certain tokens in fields inside the document, e.g., AS names, headers/banners, etc.

We exclude features that are extracted from Censys documents' metadata, which are added by Censys by processing the information gathered from all scanned ports and cannot be assigned to a certain port. We further remove 11 ports that have been observed on less than 0.3% of active IP addresses, since we cannot collect enough samples on these ports for training robust models. We also remove port 3389 (RDP protocol), observed on 1.9% of active IPs, due to poor prediction performance, indicating that our feature set is not effective in predicting responses for this port. After pruning the feature set we obtain 13 679 features from 20 ports, as well as location and AS properties, for training/evaluating our framework. Table 21.1 contains the fields/ports used for our analysis, as well as the number of features extracted from each field, and frequencies of active/open ports among active IP addresses.

21.3. Model and Metrics

In this section we go over the classification algorithm used for our proposed framework and the features used for training models, and define the metrics used for evaluating the performance of our classifiers.

21.3.1. Classification algorithm

Classification is one of the most well-studied machine learning tasks. A wide range of algorithms with various levels of complexity have been designed to tackle this problem, including logistic regression, support vector machines (SVM), neural networks [30], and decision tree-based models such as random forests [31] and gradient-boosted trees [32]. Simple linear algorithms such as logistic regression are fast to train, but often achieve less accuracy than more complex and recent models, especially over large data sets. Neural networks and deep learning models achieve state-of-the-art performance for many tasks such as image/text classification, but are often slow to train. With tabular data (or when not dealing with image, video, or natural language processing tasks) tree-based algorithms are often preferred to deep learning models, due to their superior speed/performance. For this study, we use a gradient-boosted trees model, more specifically XGBoost [32] for predicting whether a host will respond to requests on a certain port. We train one classifier for predicting the label assigned to each port, resulting in a collection/sequence of classifiers that can be used to predict the responses of all ports for a specific IP address.

21.3.2. Features for model training

For training a model, we first produce labels for each port of an IP address by observing whether Censys has reported a response under said port for its record of that IP address. Note that for an inactive IP, all the produced labels are zero, meaning that no port is responding to requests. We then use different subsets of the binary features discussed in Section 21.2.3 for training binary classifiers, as detailed below.

Pre-scan features. These include features extracted from location and AS properties, which are available before performing any scans. These features provide a priori information about each host, which can be used as initial attributes for predicting port responses. Location information can help detect patterns in the behavior of IPs in different regions, while AS properties can help predict labels based on the type/owner of the IP address. For instance, observing the word "university" in an AS name can indicate an educational network, while "cable" can help recognize residential/ISP networks.

Post-scan features. Assuming that probes are performed sequentially, classifiers can also leverage features extracted from previous probes of an IP address for predicting the responses of the remaining ports. These then provide a posteriori features for classification. Note that using a stateless scanner such as ZMap [3], we only record whether a host has responded on a certain port, resulting in a single binary feature. However, with a stateful scan such as ZGrab [23], a full handshake is completed with the server, and subsequent classifiers can also make use of parsed responses, resulting in a richer feature set. We evaluate both of these cases to determine the improvement provided by machine learning for stateless and stateful scans.

21.3.3. Metrics

In Section 1, we pointed out that our learning task can be transformed into a binary classification problem. Traditional metrics to measure the performance of classifiers include accuracy and the AUC score. Note that the latter can only be applied when the classifier is able to produce probabilities or scores, and not just a zero or one prediction, which is the case for gradient-boosted

trees. The AUC score is a good metric to measure a classifier's ability in rank ordering samples when labels are highly unbalanced. Indeed our task is a highly unbalanced classification problem, since most IP addresses are inactive or do not respond to probes. However, for our study we are focusing on reducing the number of probes for relatively high true positive rates, while the AUC score factors in the accuracy of the model over all operating points or true positive rates. Therefore, to evaluate the performance of trained classifiers, we instead estimate the probing rate at different true positive rates for discovering active ports. Take $y_i^k \in \{0, 1\}$ to denote the label for IP $i \in \{1, \ldots, N\}$ and port $k \in \{1, \ldots, M\}$ (i.e., whether IP i responds to probes on port k), $0 \leq \hat{y}_i^k \leq 1$ to be the prediction of the true label generated by a trained classifier, and $0 \leq t_r^k \leq 1$ to be the threshold corresponding to the true positive rate $0\% \leq r \leq 100\%$. Further assume that $\mathbb{S}_a$ denotes the set of active IP addresses in our data set, and p_a to be the percentage of active IPs in-the-wild (for Censys p_a is approximately 5.7%). We can then define the probing rate as follows:

$$PR_r^k = p_a \frac{\sum_{i \in \mathbb{S}_a} \mathbb{1}\{\hat{y}_i^k >= t_r^k\}}{|i \in \mathbb{S}_a|} + (1 - p_a) \frac{\sum_{i \notin \mathbb{S}_a} \mathbb{1}\{\hat{y}_i^k >= t_r^k\}}{|i \notin \mathbb{S}_a|} \tag{21.1}$$

Note that the probing rate is defined for a certain port (k), and a target true positive rate (r). In Equation 21.1, we are computing the weighted average of probing rates over active and inactive IPs, since in our curated data sets, detailed in Section 21.2.2, the proportions of active/inactive IPs do not reflect their proportions in-the-wild, i.e., we oversampled active IP addresses to obtained a more balanced data set for training. $1 - PR_r^k$ then denotes the

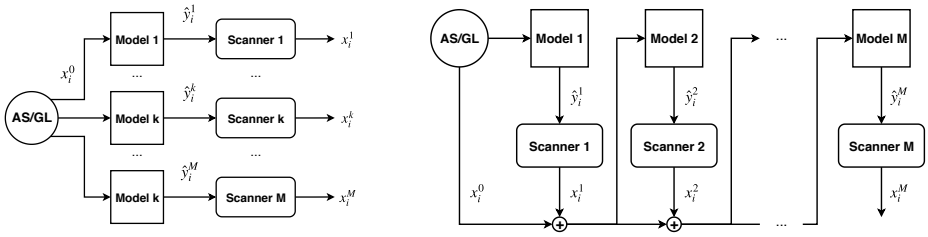
bandwidth savings for port k at the true positive rate r, which we will use to report the performance of our models in Section 21.5.

21.4. Methodology

In this section we propose two methods for combining machine learning with networks scans, namely parallel and sequential scanning. Parallel scans can be done independently on multiple ports, but use minimal information for predicting port responses. On the other hand, sequential scans use a richer feature set which in turn leads to more bandwidth savings, but require scanning multiple ports in a pre-determined order.

21.4.1. Parallel scanning

Currently, most Internet scans (e.g., scans in the Censys database) are performed separately and independently across different ports. In other words, the entire IPv4 address is sweeped multiple times, each time sending probes to all IP addresses on a certain port. This allows different ports to be scanned independently, possibly at different times, thereby reducing the amount of traffic sent to networks/hosts. In this scenario, our method can only use the location and AS properties of the targeted IP addresses for predicting the responses of hosts, as depicted in Figure 21.2a. In this diagram, the gelocation (GL) and AS features are fed to each trained model in order to produce the prediction $\hat{y}_i^k$ of the true label y_i^k for sample i and port k, i.e., the estimated likelihood that IP address i will respond to probes on port k. These predictions are then fed to the scanner, which will decide whether to scan different IP/port pairs



(a) Parallel scans

(b) Sequential scans

Figure 21.2: Diagram of scanning when all port labels are predicted using only geolocation and ownership (AS) properties (left), and when information obtained from port responses are used as features for subsequent probes (right). $\hat{y}_i^k$ denotes the predicted label for IP address i and port k from a trained classification model (model k), and x_i^k is the feature vector resulting from the probe of IP i on port k; x_i^0 is the feature vector for AS/GL features.

depending on the prediction of the model. In this study, we make decisions by thresholding $\hat{y}_i^k$; if $\hat{y}_i^k < t_r^k$ the scanner refrains from sending the probe. Note that t_r^k (specific to port k) is the threshold for reaching a target true positive rate r.

While this approach uses a minimal amount of information for prediction, applying machine learning to parallel scans is fairly straightforward, since the predictions of trained models can simply be translated into blacklists that can be fed to network scanners for refraining from sending probes to certain IP/port pairs. Moreover, due to the crude granularity of gelocation and AS features, we do not need to perform predictions for every IP address, but only for IP blocks in which all IP addresses share the same features, therefore reducing the computational overhead of our approach. We will discuss this point in more detail in Section 21.6.4.

21.4.2. Sequential scanning

In contrast to parallel scanning, one can also design a scanner to scan different ports in a sequential manner. In this setting, we can take advantage of the responses of previously scanned ports for predicting the remaining labels. Cross-protocol dependencies have been observed by Bano et al. [13], but were not directly used for bootstrapping network scans. This is due to the fact that cross-protocol correlations by themselves are not sufficient for predicting other port labels, as we will further discuss in Section 21.6.1. However, we show that when combined with pre-scan features, i.e., location and AS properties, cross-protocol information can help improve the efficiency of sequential scans, as compared to parallel scans.

Assume $x_i^k, k \in \{1, \ldots, M\}$ to denote the feature vector resulting from probing IP i on port k, and x_i^0 to denote pre-scan features. Then for sequential scanning, the classifier for port k is trained using $\{x_i^l, l < k\}$ as features, i.e., GL/AS features, as well as ports scanned earlier in the sequence. Note that for parallel scans in Section 21.4.1 we are only performing predictions using x_i^0 as features. We evaluate and compare this approach to parallel scanning in Section 21.5, resulting in more bandwidth savings. Figure 21.2b depicts the process used for sequential scans. Similar to parallel scanning, each model in this figure is generating a prediction $\hat{y}_i^k$ for an IP/port pair, which is then fed to the scanner for thresholding. The features resulting from each scan (i.e., the post-scan features in Section 21.3.2) are then appended to the model's input features and used for all subsequent models. This allows models toward the end of the sequence to make predictions based on a richer feature set, which can result in more bandwidth savings for their corresponding scans.

21.4.2.1. Finding an optimal scan order

Note that to achieve the most bandwidth savings, we first need to obtain an optimal order for scanning different ports of an IP address, since the dependency between port responses can vary between different pairs of ports. Moreover, the relationship is not necessarily symmetric, since a prominently used port such as port 80 (HTTP) can provide a lot of information regarding the responses of more uncommon ports, while the reverse might not be true.

To this end, we need to find an optimal scanning order to achieve the lowest possible average probing rate across all ports. Note, however, that exhaustively evaluating all permutations of ports is not feasible; instead, we use the following heuristic approach for quantifying port dependencies and finding a semi-optimal order. We first train a set of classifiers that use the responses of all remaining ports for prediction. For instance, if we want to predict the labels assigned to port 21 (FTP), we use features obtained from all the remaining 19 ports, as well as location/AS features, for training/evaluating our classifiers. We then compute the importance of each port for predicting the labels of all other ports. In ensembles of decision trees, such as gradient-boosted trees, this can be achieved by summing the improvement in accuracy brought by a feature in all the splits that it is used for, yielding its contribution to the predictions of the model. For each trained classifier, we then compute and normalize the contribution from all features used in the model (so that all feature importances sum up to one), and then compute the contribution of each port for the model's predictions.

Let $A_{M\times M}$, where M is the number of ports ($M = 20$ for this study), be a square matrix, where a_{ij} represent the importance of port i for predicting the label of port j. Our goal is to find an ordering of ports, so that ports that pose

high importance for other ports are scanned first. In other words, if a_{ij} is high, then we prefer to scan port i prior to port j. We can reformulate this problem to finding a permutation A_p of both rows and columns in A, such that the sum of elements in the upper triangle of A is maximized. We compare two techniques for finding an optimal ordering, namely sorting ports by decreasing order of their total contribution to other classifiers ($a_i = \sum_{j \neq i} a_{ij}$), and sorting by increasing order of total contribution received from other ports ($a_i = \sum_{i \neq j} a_{ji}$). The former prioritizes scanning ports that pose high importance for predicting other port response first, while the latter prioritizes scanning ports that are highly dependent on other ports last. In our experiments, we found that the first approach resulted in higher overall bandwidth savings, and therefore we report our results using this approach. While the proposed heuristic approach is not guaranteed to find the best possible order (which would require exhaustive evaluation of all port permutations), we found it to perform well in practice; we will further elaborate on this in Section 21.5.3.

21.4.2.2. Training the sequence of classifiers

Note that the aforementioned models are only used for finding the order in which ports are scanned, and are not used for running the actual probes. Once the order had been determined, we retrain our classifiers using the obtained sequence. Each classifier is then trained using gelocation and AS features, as well as features resulting from scans that are placed earlier in the sequence. It is worth mentioning that for false negatives (i.e., when an active port is not scanned due to an incorrect prediction) the true features of the corresponding port are not revealed, and subsequent models are making predictions based on partially masked features. Therefore when evaluating our classifiers, we

also mask the features corresponding to false negatives[1], to get an accurate estimate of our technique's performance; for training, we use the true features of each port without any masking.

21.5. Evaluation

In this section we evaluate the bandwidth savings of our framework for both parallel and sequential scans, and compare the performance of stateless sequential scans with stateful scans and scanning without using a priori (i.e., location and AS) information.

21.5.1. Setup

We use cross-validation to train, tune, and evaluate the performance of our framework. We split the curated data sets from each snapshot (detailed in Section 21.2) into a training set containing 60% of samples, used for training classifiers, and a test set containing 40% of samples for evaluating performance according to the metrics defined in Section 21.3.3.

For XGBoost models, we use 100 boosting rounds (i.e., the number of trees in the ensemble), a learning rate of 0.2, and a maximum depth of 20 for each tree in the ensemble; these parameters have been chosen by cross-validation. We use the logistic objective to produce probabilities between zero and one for each sample.

[1]Note that false negatives are dependent on the choice of the classifier's operating point, or the target true positive rate. Therefore, we generate different masks and rerun the sequence for evaluating each true positive rate.

Note that for curating data sets, we undersampled inactive IP addresses to prevent our data sets from being dominated by inactive samples. However, simply training a classifier on this dataset means that the training algorithm gives equal weight to predictions on active/inactive IPs, while in reality a false prediction on an inactive sample should be given higher weight than an active sample. Therefore, we adjust the weights for active and inactive samples to reflect their true population in the real world.

Finally, we weight positive labels for each port by computing the ratio between the sum of weights for IPs with an inactive port, and the sum of weights for IPs with an active port, i.e., $\sum_{y_i^k=0} w_i^k / \sum_{y_i^k=1} w_i^k$, where w_i^k is the weight assigned to sample i for port k. This is done to handle the imbalanced nature of our data set, where positive labels are scarce. Without this scaling, the model will be heavily biased toward predicting a label of zeros for all samples, resulting in poor performance over samples with an active port [33].

21.5.2. Parallel scanning

We first evaluate the performance of our technique for improving parallel scans, depicted in Figure 21.2a. Table 21.2 summarizes our results. Each row in Table 21.2 corresponds to a certain target true positive rate, reporting the bandwidth savings ($1 - PR_r^k$) for different ports, that is the percentage of probes that were not sent while achieving the target true positive rate. We have also reported the overall bandwidth savings by averaging bandwidth savings across all ports in the last row in Table 21.2.

Our evaluation results suggest that we can achieve an overall bandwidth savings of 26.7% while detecting 99% of IP/port pairs that respond to requests;

Table 21.2: Bandwidth savings for parallel scans using AS and geolocation features. Each cell reports the percentage of IPs for which we can refrain from sending a probe, while still achieving the corresponding true positive rate for detecting active ports. Overall bandwidth savings are computed by averaging over all ports. Ports 445 (SMB) and 7547 (CWMP) receive the most bandwidth savings.

Port/protocol	Target TPR				
	90%	95%	98%	99%	99.9%
21/ftp	60.8%	43.9%	27.7%	17.6%	2.9%
22/ssh	59.8%	44.5%	28.7%	20.2%	6.9%
23/telnet	51.7%	36.1%	21.5%	15.3%	4.3%
25/smtp	69.1%	50.3%	30.8%	21.4%	4.0%
53/dns	59.7%	43.7%	27.2%	16.9%	3.3%
80/http	56.6%	42.3%	27.0%	19.5%	7.7%
110/pop3	81.8%	64.2%	39.4%	23.1%	3.0%
143/imap	82.9%	66.0%	43.7%	27.3%	3.6%
443/https	53.9%	39.7%	26.0%	19.2%	7.5%
445/smb	92.3%	78.0%	56.9%	46.8%	24.4%
465/smtp	82.1%	67.3%	42.8%	29.5%	3.4%
587/smtp	77.2%	61.1%	37.6%	23.8%	5.0%
993/imaps	83.9%	69.1%	46.3%	31.6%	3.1%
995/pop3s	86.1%	71.4%	48.0%	31.5%	4.1%
2323/telnet	62.7%	46.0%	31.1%	24.1%	1.0%
3306/mysql	81.8%	63.3%	43.8%	28.1%	4.6%
5432/psql	66.4%	45.2%	20.8%	14.3%	1.0%
7547/cwmp	92.4%	86.7%	78.6%	70.0%	39.2%
8080/http	61.0%	48.5%	33.1%	23.9%	8.0%
8888/http	78.8%	63.1%	41.9%	30.5%	8.5%
Overall	72.0%	56.5%	37.6%	26.7%	7.3%

this can be further increased to 72.0% when using a lower true positive rate of 90%. Note that this true positive rate is consistent across all ports; for each port we are computing the threshold for achieving the target true positive rate by computing the ROC curve of samples in our test set, and choosing the corresponding operating point for the target true positive rate.

Our results suggest that location and AS properties of IP addresses can be effectively used to predict whether a host will respond to requests on a certain port. Interestingly, we observe that responses of some ports can be predicted more accurately, resulting in higher overall bandwidth savings. For instance, while the savings at 99% true positive rate in Table 21.2 are ∼20-30% for most ports, we can obtain higher savings of 70.0% for port 7547 (corresponding to the CWMP protocol). Note that the CWMP is protocol used by modems and router, which are more common on ISP networks. Indeed we observe that IP addresses that have the token "charter" (i.e., Charter Communications) in their AS description field are 15 times more likely to respond to probes on port 7547, while observing the token "amazon" makes a host more than 400 times less likely to have an open CWMP port.

21.5.3. Sequential scanning

We then evaluate the performance for sequential scans, depicted in Figure 21.2b. For this section, we only use a single feature from each port, i.e., whether it has responded to the scanner. This makes our technique compatible with stateless (IP layer) scanners such as ZMap, while at the same time minimizing the processing required for constructing features for subsequent probes.

As discussed in Section 21.4.2, we first need to to measure the importance of

Table 21.3: Bandwidth savings when using all port responses (except the one that is being predicted) as features. Note that these results provide a lower bound for the bandwidth savings for sequential scanning (Table 21.4).

Port/protocol	Target TPR				
	90%	95%	98%	99%	99.9%
21/ftp	95.2%	82.0%	58.5%	42.8%	13.8%
22/ssh	85.7%	66.5%	43.3%	31.4%	10.5%
23/telnet	73.4%	56.4%	37.1%	22.5%	5.0%
25/smtp	98.7%	94.7%	69.2%	45.2%	8.7%
53/dns	79.2%	63.3%	39.6%	25.9%	4.5%
80/http	87.6%	69.6%	48.8%	34.9%	11.6%
110/pop3	99.8%	99.8%	99.8%	99.5%	60.9%
143/imap	99.9%	99.8%	99.8%	99.7%	82.8%
443/https	78.7%	59.8%	39.1%	28.8%	9.8%
445/smb	94.0%	81.9%	63.6%	53.4%	30.1%
465/smtp	99.9%	99.8%	99.7%	96.7%	50.3%
587/smtp	99.2%	94.0%	72.7%	52.8%	12.9%
993/imaps	99.9%	99.9%	99.8%	99.7%	48.9%
995/pop3s	99.9%	99.9%	99.8%	99.8%	95.2%
2323/telnet	69.4%	52.3%	39.3%	28.9%	15.2%
3306/mysql	98.8%	96.8%	84.4%	67.3%	21.9%
5432/psql	90.1%	73.5%	34.4%	19.9%	5.3%
7547/cwmp	92.9%	87.2%	80.2%	72.6%	41.1%
8080/http	70.8%	58.5%	42.5%	31.3%	10.6%
8888/http	85.2%	72.4%	53.5%	41.2%	10.8%
Overall	89.9%	80.4%	65.3%	54.7%	27.5%

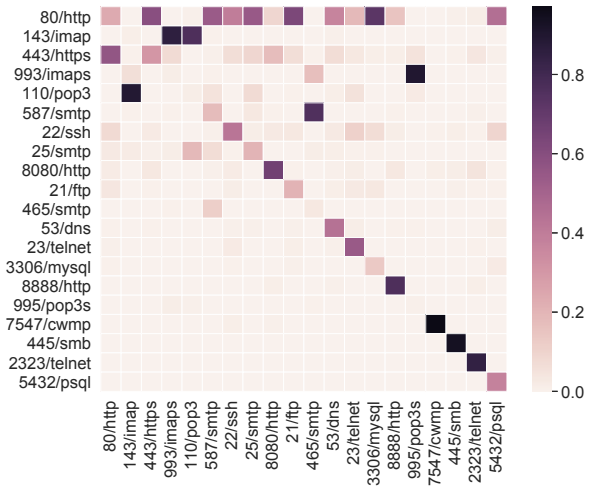


Figure 21.3: Contribution of port responses for predicting other port labels. Ports are scanned from left to right (top to bottom). Each column displays the importances of other ports for predicting the label of the corresponding port. The importances on the diagonal correspond to the contribution of location and AS features. We observe that high importance cells are placed in the upper triangle, enabling classifiers to use them for prediction.

each port for predicting other port labels. Therefore, we train a set of classifiers by using all port responses (except the one for which we are predicting labels) as features for prediction. Our results are reported in Table 21.3. We observe that the bandwidth savings are significantly higher that those in Table 21.2; note that the bandwidth savings in Table 21.3 are an upper bound for the savings we can achieve by predicting port responses, since each classifier is using the most amount of information that can be available to them for prediction. However in reality, some ports are inevitably scanned before others, meaning that they need to generate predictions based on a subset of the information that is used in Table 21.3.

We then compute the importance of each port to find an optimal ordering according to the process detailed in 21.4.2, so that ports that pose high impor-

tance for predicting the remaining labels are scanned first. Figure 21.3 displays the pairwise importance of ports. Ports that are scanned first are placed to the left (top) of the figure. We observe that after ordering, most of the cells with a large contribution are placed in the upper triangle. Note, however, that in some instances (e.g., ports 80 and 143, corresponding to the HTTP and IMAP protocols) we inevitably have to forgo some high importance cells, e.g., when two ports are mutually important for one another. Interestingly, we observe that ports corresponding to mail servers (IMAP/IMAPS, POP3/POP3S, and SMTP) are highly correlated with one another, which explains the high bandwidth savings for these ports (often more than 90%) in Table 21.3. The HTTP protocol also exhibits high importance for a wide range of ports, which is the main reason why it is scanned first, followed by the IMAP protcol, which provides major benefits for predicting the responses of other mail protocols. Ports such as 7547 (CWMP) and 445 (SMB) are isolated, meaning that they do not provide nor recieve any benefits for/from the labels of other ports, and are placed toward the end of the sequence.

We then train a sequence of classifiers using the obtained order from Figure 21.3, the bandwidth savings of our trained models are included in Table 21.4. Compared to Table 21.2 we observe between 10% to 20% more bandwidth savings for different true positive rates; this is largely due to the significant performance benefits over mail server ports. However, we also obtain more bandwidth savings across almost all ports, with notable savings on ports 21 (FTP), 3306 (MySQL), and 5432 (PSQL). Our results justify the use of the sequential structure in Figure 21.2b for reducing the probing rate of scans. For instance, at 99% true positive we can refrain from sending approximately 1/4 probes for parallel scans, while using sequential scans this increases to dropping

Table 21.4: Bandwidth savings when ports are sequentially scanned according to the diagram in 21.2b, yielding ∼10%-20% more bandwidth savings compared to Table 21.2. Mail server ports (IMAP/IMAPS, POP3/POP3S, and SMTP) receive the most benefits, reducing the number of probes by more than 90% in some instances. FTP, MySQL, and PSQL protocols also receive considerable boost compared to Table 21.2.

Port/protocol	Target TPR				
	90%	95%	98%	99%	99.9%
21/ftp	82.1%	68.7%	47.7%	36.3%	11.2%
22/ssh	69.3%	52.7%	35.3%	25.6%	9.7%
23/telnet	62.9%	49.0%	32.3%	22.3%	7.9%
25/smtp	85.8%	69.9%	48.8%	37.1%	10.1%
53/dns	72.1%	57.0%	36.5%	25.4%	5.9%
80/http	55.8%	41.4%	26.1%	18.8%	7.0%
110/pop3	99.1%	97.6%	87.3%	72.8%	27.8%
143/imap	98.0%	91.0%	68.0%	51.7%	6.3%
443/https	61.7%	45.3%	30.8%	22.5%	7.3%
445/smb	94.1%	82.9%	65.1%	53.1%	22.6%
465/smtp	99.8%	99.7%	97.0%	90.6%	50.1%
587/smtp	90.7%	75.6%	53.8%	39.9%	12.6%
993/imaps	99.5%	99.2%	97.8%	88.4%	37.1%
995/pop3s	99.9%	99.8%	99.8%	99.7%	94.0%
2323/telnet	68.1%	50.7%	38.6%	35.3%	12.0%
3306/mysql	98.2%	95.9%	80.6%	67.2%	23.4%
5432/psql	90.1%	70.2%	34.0%	21.6%	6.4%
7547/cwmp	92.6%	86.8%	79.3%	70.8%	37.9%
8080/http	65.6%	53.7%	37.7%	28.4%	10.8%
8888/http	84.3%	72.7%	53.2%	40.5%	14.9%
Overall	83.5%	73.0%	57.5%	47.4%	20.8%

slightly less than 1/2 probes. At 95% true positive rate, Table 21.4 suggests that we can drop roughly 3/4 probes, with a probing rate of 27.0%, while using parallel scans we achieve a probing rate of 43.5%.

We also observe that the obtained scan order puts our bandwidth savings close to the upper bounds in Table 21.3. Note that as mentioned before, we inevitably have to forgo the dependency of some pairs of ports (e.g., the dependency of port 80 on other ports, and port 143 on other mail server ports), in order to utilize their high predictive power for ports placed further in the classifier sequence. In fact, the difference between bandwidth savings for ports 80 and 143 in Tables 21.3 and 21.4 accounts for 44% of the difference in overall bandwidth saving at 99% true positive rate. This suggests that we are indeed taking full advantage of cross-protocol information for reducing probing rates, and further justifies our proposed technique for finding an optimal order for scanning ports.

21.6. Discussion

In this section, we provide further motivation for our proposed method for conducting smart scans by comparing our methodology to other approaches with different levels of information, inspecting coverage of scans over vulnerable and misconfigured IP addresses, discuss how often models need to be retrained to keep them up-to-date, as well as the practical utility of our framework and its computational performance.

21.6.1. Comparison with other approaches

Our results on parallel and sequential scans demonstrate the ability of using location, AS, and cross-protocol information for predicting active IP addresses. In previous work, Klick et al. [34] have shown that it is possible to reduce scan traffic by conducting full scans and identifying low density prefixes. However, this method relies on conducting full scans to be able to identify low density prefixes, especially when inspecting small prefixes or rarely active ports, where random sampling of a small subset would not yield an accurate estimation. We address this by conducting partial scans of 17.5 million IP addresses (∼0.6% of the address space announced on BGP) to train classification models based on location and ownership properties, combining them with cross-protocol information to further reduce the traffic generated by scans. Moreover, Klick et al. [34] report that their accuracy (true positive rate) drops at a rate of 0.3%-0.7% per month, while our method can guarantee coverage by retraining models, or readjusting thresholds as discussed in Section 21.6.3.

While the dependency between active ports has been observed in previous work [13], our method attempts to utilize this property for improving the efficiency of scans by combining them with a priori attributes. To examine the boost achieved by appending gelocation and AS properties to port responses, in this section we compute bandwidth savings when using only cross-protocol information for sequential scan.

Additionally, our results so far on sequential scans have been obtained by adding only a single binary feature for each scanned port: whether the probed host has responded to the request. While, this assumption makes our technique compatible with stateless scanners such as ZMap, it leads to the following question, can we achieve a better performance by completing a full

handshake with the probed host, and record and append the resulting features for prediction? Note that for active IP addresses, Censys also records details of stateful (application layer) scans conducted using ZGrab [23], which in turn is converted to a rich feature set as detailed in Table 21.1. These features include tokens extracted from headers/banners of different protocols, parsed SSL/TLS certificate chains served by secure protocols such as HTTPS, and even granular attributes such as the choice of encryption ciphers, misconfiguration such as open DNS resolvers, etc.

To answer the previous questions, we also train a sequence of classifiers without using geolocation and AS features, and another sequence utilizing the full set of features extracted from stateful scans. Our results are included in Table 21.5, where we have included average bandwidth savings across all ports. Note that for inactive IPs/ports, stateful and stateless scans provide the same level of information, and since the majority (94.3%) of IP addresses are inactive, the overall performance of both methods will be similar. Therefore for a more thorough comparison, we are also reporting the bandwidth savings of all three methods over only active IP addresses in Table 21.5. We have plotted the overall bandwidth savings of the examined scanning strategies in Figure 21.4.

Comparing stateless scans with and without a priori features, we observe that the latter achieves significantly less bandwidth savings, even lower than parallel scans. This demonstrates the importance of using gelocation and AS features for conducting machine learning enabled scans, i.e., characterizing the network that an IP address belongs to plays an important role for predicting active hosts. This also suggests that that pre-scan and post-scan features complement each other well, boosting the performance of our framework when

Table 21.5: Overall bandwidth savings across all ports for parallel, stateless sequential, and stateful sequential scans. Savings are reported over all IP addresses, as well as active IPs only. Conducting stateless scans without the use of location and AS feature results in poor performance. Additionally, we do not observe a significant improvement by using stateful scans.

Scan type		Target TPR				
		90%	95%	98%	99%	99.9%
Parallel	All	72.0%	56.5%	37.6%	26.7%	7.3%
	Active	57.6%	43.5%	27.9%	19.4%	5.6%
Stateless sequential	All	83.5%	73.0%	57.5%	47.4%	20.8%
	Active	60.9%	50.1%	37.2%	29.2%	12.0%
Stateless sequential (w/o pre-scan features)	All	49.7%	37.1%	25.0%	19.6%	3.5%
	Active	32.9%	25.0%	20.5%	14.2%	3.6%
Stateful sequential	All	83.9%	73.1%	58.2%	47.9%	20.2%
	Active	67.9%	58.0%	45.3%	36.7%	15.0%

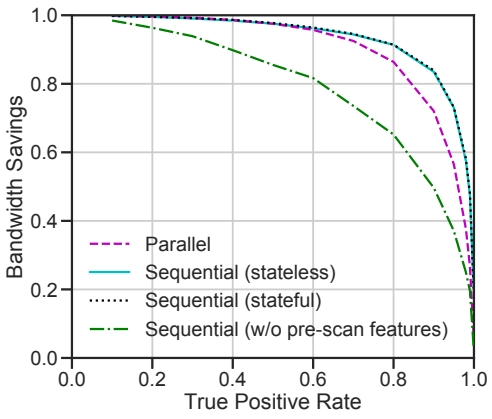


Figure 21.4: Overall bandwidth savings of different feature sets for bootstrapping scans. Pre-scan (location and AS) and cross-protocol information complement each other well, producing the largest savings in the sequential case.

using both feature sets. Cross-protocol information by itself does not provide good predictive power for most port responses, with the exception of mail protocols which are strongly correlated. In fact, mail protocols account for almost all of the bandwidth savings when not using any pre-scan features. On the other hand, location and AS properties offer bandwidth savings across all ports, even for isolated ports such as 7547 (CWMP) and 445 (SMB). Consequently, using both pre-scan and cross-protocol information, we can further reduce probes by leveraging all available information, including biases in different regions, and strong correlations between certain ports. Note that while cross-protocol dependencies have been observed in previous work [13], our framework leverages these correlations for bootstrapping scans by combining them with a priori attributes.

Comparing stateless and stateful scans in Table 21.5, we observe similar overall performance[2], and slightly higher bandwidth savings for active IPs in the stateful scenario. Note that applying our technique for stateful scans also leads to a significant computational overhead, since the results of each scan must be parsed, followed by feature extraction, and the increase in the number of features also leads to complex, and therefore slower models. Due to the small performance benefit offered by stateful scans, we conclude that using stateless scans is sufficient for our sequential approach in Figure 21.2b.

Note that another approach for reducing probes of an Internet scan is to refrain from sending probes to hosts that have not been responsive in previous scans. However, this method relies on periodically performing global scans,

[2]The small decrease in overall performance for the 99.9% true positive rates for stateful scans is possibly due to overfitting because of the large number of features, and variations due to randomness in the training process.

while our proposed method only needs a partial exhaustive scan for training the underlying classifiers. Moreover, using historical data fails to recognize new active IP addresses. On the five snapshot used for this study, we observe that an average of 19.7% of active IPs in each snapshot are not present in the previous month's snapshot. Previous work has also reported that the accuracy of this approach drops to 80% within one month [34]. We have also included a breakdown of new active ports in Table 21.6, where the percentage of new active ports can be as high as 50.6% for port 445 (SMB). This suggests that naively using historical data for producing scan target can lead to low true positive rates, while our proposed approach is also effective at detecting new active IPs. Nevertheless, it is interesting to examine whether adding historical data to our feature set can further boost the performance of classifiers, a problem that we leave for future work.

21.6.2. Coverage on vulnerable IP addresses

Network scanning is often used to give visibility into the security posture of networks by revealing vulnerable Internet-facing machines. While simply being accessible on the public Internet poses risk of being targeted by attackers, and some ports (e.g., port 7547 for routers) generally should not be left open, other protocols such as HTTP(S) and mail protocols are used to offer services to clients, and simply responding to probes is not necessarily an indication of a vulnerability. In this scenario, the scanner should be able to discover vulnerable subpopulations with good coverage (i.e., true positive rate) to be able to accurately assess the attack surface of networks. Therefore in this section, we examine the true positive rate of our scanning technique for three types of

Table 21.6: Average percentage of active ports that are inactive in the previous month's snapshot, averaged over all five snapshots in this study. Note that the average percentage of new active IPs over all snapshots is 19.7%.

Port	Protocol	Percentage
21	ftp	17.6%
22	ssh	20.4%
23	telnet	31.6%
25	smtp	12.2%
53	dns	28.8%
80	http	13.5%
110	pop3	6.9%
143	imap	5.2%
443	https	14.6%
445	smb	50.6%
465	smtp	7.2%
587	smtp	5.0%
993	imaps	7.8%
995	pop3s	5.2%
2323	telnet	28.5%
3306	mysql	12.2%
5432	psql	10.8%
7547	cwmp	33.3%
8080	http	17.8%
8888	http	29.7%

vulnerable and misconfigured machines, namely open DNS resolvers, HTTPS servers susceptible to the Heartbleed vulnerability, and Exim mail servers vulnerable to CVE-2018-6789 disclosed on 2/8/2018.

Open DNS resolvers are utilized in DNS amplification attacks, where an attacker turns small queries into large payloads, resulting in a reflection Distributed Denial of Service (DDoS) attack by eliciting responses from open DNS resolvers to spoofed IP addresses [35]. While an open DNS resolver is not a direct vulnerability, it endangers the security of the Internet, and previous work has shown that it is correlated with high cyber-risk due to being an indication of mismanangement and poor security policies [8]. We extract 3 520 442 (43.4% of all DNS servers) open DNS resolvers for 1/1/2019 as indicated by Censys. The Heartbleed vulnerability (CVE-2014-0160) [9] is a security bug in the OpenSSL library that was discovered in April 2014, allowing attackers to remotely read protected memory from HTTPS servers. While this vulnerability was promptly patched following its disclosure, Censys reports 101 083 (0.18% of HTTPS servers) vulnerable HTTPS sites in its 1/1/2019 snapshot. CVE-2018-6789 affecting Exim SMTP mail server versions ≤ 4.90, allows attackers to remotely execute arbitrary code. By parsing the banner of the SMTP protocol on ports 25, 465, and 587, we extract 400 199 (6.0% of all servers), 523 134 (13.0%), and 685 191 (12.4%) susceptible servers, respectively.

Table 21.7 displays the observed true positive rates for the aforementioned vulnerabilities at different operating points for parallel and sequential scans. We observe similar coverage to the overall target true positive rates for open resolvers, and higher coverage for Exim servers. For servers vulnerable to Heartbleed, coverage is lower ($\sim$75%) at 90% target TPR, however this improves for higher operating points. Overall, we do not observe a large bias that would im-

Table 21.7: Coverage (true positive rate) over vulnerable and misconfigured IP addresses at different target true positive rates.

Scan type		Open Resolver	Heartbleed	Exim $\leq$ 4.90		
		53/dns	443/https	25/smtp	465/smtp	587/smtp
Parallel	90%	87.9%	74.8%	96.1%	92.6%	95.0%
	95%	93.6%	91.3%	98.2%	97.7%	98.4%
	98%	97.4%	95.7%	99.4%	99.5%	99.6%
	99%	98.6%	98.5%	99.8%	99.8%	99.8%
	99.9%	99.9%	100.0%	99.9%	100.0%	99.9%
Sequential	90%	87.7%	75.2%	98.1%	93.6%	94.1%
	95%	94.0%	92.9%	99.2%	97.7%	97.8%
	98%	97.6%	98.4%	99.8%	99.3%	99.8%
	99%	98.7%	99.9%	100.0%	99.9%	99.9%
	99.9%	99.9%	100.0%	100.0%	100.0%	99.9%

pede our technique from discovering vulnerable subpopulations, even for rare vulnerabilities such as Heartbleed affecting 0.18% of HTTPS sites; this further justifies the efficacy of our technique for security applications. Note, however, that while we observe consistent discovery rates for the examined vulnerabilities, there may exist other subpopulations of interest with low coverage. Nevertheless, one can guarantee discovery rates by adjusting the threshold for sending out probes, or training classifiers specifically targeting said subpopulations by applying the same methodology.

21.6.3. Keeping models up-to-date

The Internet is an ever-changing ecosystem, where structures and patterns can change over time. For instance, ownerships of different networks, or the behavior of system administrators can change over time, varying the patterns which are utilized by our models for predicting port responses; this in turn

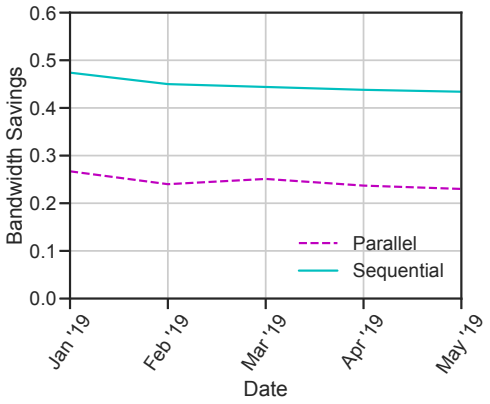


Figure 21.5: Performance of parallel and sequential models trained at 1/1/2019, over data from different dates. Bandwidth savings are reported for a target true positive rate of 99%. We observe gracefully degrading performance, suggesting that models only need to be retrained once every few months.

warrants retraining models in order to keep them up-to-date. To determine how often models should be retrained, and to evaluate the performance of trained models for predicting port responses in future scans, we evaluate the overall bandwidth savings over our data sets for 2/1, 3/1, 4/1, and 5/1 of 2019, when models are trained on data from 1/1/2019.

We have included our results in Figure 21.5; we are reporting bandwidth savings corresponding to a 99% true positive rate. To achieve the target true positive rate for each scan, we use partial exhaustive scans on 17.5 million addresses for each date (the same amount used for training the machine learning models), and readjust thresholds t_r^k detailed in Section 21.3.3; this allows us to guarantee overall coverage while removing the need to retrain models. We observe a graceful degradation of performance, suggesting that models only need to be retrained once every few months to keep them up-to-date.

21.6.4. Practical utility

We now discuss how our framework can be combined with existing scanners. The first step is to collect exhaustive measurements on a random subset of IP addresses for training classification models. Note that tools such as ZMap already traverse the IPv4 address space in a randomized order, in order to spread their traffic and reduce the strain on different networks; therefore we can simply perform a partial scan first, and then pause for training models. For this study, we used data corresponding to 17.5 million IP addresses, which only account for $\sim$0.6% of the utilized IPv4 space announced on BGP. For parallel scans, the predictions of trained models can simply be provided to the scanner as a blacklist, telling the scanner to avoid sending probes to IPs where it is fairly certain that the probe is not going to be answered. For sequential scans, the same process can be achieved in sequence, simply taking the results of one full scan, performing predictions, and providing the results as a blacklist for the subsequent scan.

One important aspect of our framework is its computational overhead for performing predictions. For our experiments, we used the GPU-accelerated algorithm for XGBoost for both training and evaluating our models, using a NVIDIA GeForce GTX 1080 Ti GPU. It takes 20-40 seconds to train a model for a single port; we observe similar times for both parallel and sequential models. Note, however, that according to our results in Section 21.6.3, we do not need to retrain models after each scan.

For a trained model it takes approximately 10 microseconds to perform a prediction on a single sample. While this would add up to a significant time if predictions are performed for every single IP address (i.e., 2.8 billion addresses for the entire IPv4 address space), note that AS/GL information are

typically constant across large networks, and there are a limited number of port configurations, leading to duplicate feature vectors. In fact, we observe that for an entire Censys snapshot, there are only ∼130 000 unique GL/AS feature vectors; adding the labels for all 20 ports we observe roughly 3 million unique configurations of both GL/AS features and port labels. This suggests that the entire computation for a parallel scan can be done in ∼1.3 seconds, while for a sequential scan we can perform the required predictions in approximately 30 seconds.[3] Therefore, the computational overhead of applying our proposed techniques is negligible; for comparison, ZMap scans the entire IPv4 address space for a single port in 45 (5) minutes with a (10) gigabit connection.

21.7. Related Work

There are numerous measurement studies that analyze Internet scan data. Security studies focus on measuring vulnerabilities, misconfiguration, and analyzing malicious hosts; these include, e.g., errors made by Certificate Authorities (CAs) when issuing certificates [12], a study on the Mirai botnet [7], and a measurement-based analysis of the Heartbleed vulnerability [9]. Studies on trends and adoption rates include measuring HTTPS adoption [10], TLS deployment [11], and a study of Certification Authority Authorization (CAA) [16]. Studies on discovering different types of Internet-facing devices include scanning for instances of the Robot Operating System (ROS) [14], and developing an engine for discovering IoT devices [15]. Beverly et al. [17], Claffy

[3]Note that this is a worst-case estimate. For ports toward the start of the sequence the number of unique feature vectors is closer to the unique vectors for GL/AS features, since only a fraction of port responses have been revealed and are being used for prediction.

et al. [18], Shavitt and Shir [19] use networks scans for mapping the topology of the Internet.

While there exist a wide range of measurement studies utilizing Internet scans, there are fewer works focusing on applications of machine learning for processing network scan data. Liu et al. [8] use symptoms of mismanagement (e.g., open resolvers and untrusted certificates) discovered on an organization network, to predict their risk of suffering from a data breach. Sarabi and Liu [28] develop a framework for processing Internet scan data in machine learning models. In this chapter, we use the feature extraction techniques from [28] to generate features for training classifiers.

Generating scanning targets and analyzing active (or live) IPs has also received attention in the past. Bano et al. [13] conduct an analysis of live (active) IPs, and (similar to our results) find cross-protocol correlations between the liveness of different ports. Klick et al. [34] periodically scan the entire IPv4 address space, omitting low density prefixes from future scans. In comparison with [34], our method does not require full scans by using machine learning to identify low density networks, and leverages cross-protocol information for further improving the efficiency of scans. Fan and Heidemann [1] develop an automated technique for generating a representative Internet hitlist for scanning. Murdock et al. [2] identify dense address space regions and generate candidates for IPv6 scanning. Our proposed framework can complement target generation techniques, using machine learning to probe hosts more prudently, which in turn results in a higher hit rate, allowing one to scan hitlists at a faster rate.

Note that for our problem we are trying to predict multiple (possibly correlated) labels, making our learning task a multi-label classification problem.

Read et al. [36] suggest training a chain of classifiers in sequence, appending the output of each model in the chain for subsequent predictions. We use a similar approach to predict port labels; however, instead of using the predictions of the model, we append the true label of each sample (only if the corresponding classifier tells us to perform a probe) to subsequent predictions, since probing then reveals the true label of the IP address. Another method for dealing with correlated labels for multi-label classification includes applying a label transformation, e.g., to produce uncorrelated labels [37, 38, 39]. Dealing with imbalanced classes has been studied by Chawla et al. [40], where the authors suggest a combination of undersampling the majority class with oversampling the minority class (by creating synthetic samples). In this study, we undersample the majority class (inactive IPs), and also weight the minority class (active IPs/ports) to deal with imbalanced classes.

21.8. Conclusions and Future Work

In this chapter, we developed and evaluated a framework for reducing the bandwidth of network scans by predicting whether a host will respond to requests on a certain port, using location and ownership (AS) properties, as well as cross-protocol information. We demonstrated that using only location and AS features we can achieve overall bandwidth savings of 26.7-72.0% at 90-99% true positive rates for detecting active/open ports, averaged over 20 port scans from the Censys database. Moreover, we developed a novel technique for finding an optimal order for scanning ports and training a sequence of classifiers, appending the responses of scanned ports for predicting active IPs over subsequent scans. We show that using this technique we can increase the bandwidth

savings of Internet scans to 47.4-83.5% at 90-99% coverage levels. This reduction in bandwidth is due to the high dependency between the responses of certain ports, for instance ports corresponding to mail servers. We further show that our technique can be applied on top of current scanning tools with little computational overhead, providing blacklists in order to refrain from sending probes to certain IP/port pairs.

We compared our methodology to other strategies for conducting machine learning enabled scans, concluding that ignoring location and AS properties results in poor performance, while using the full set of features from stateful scans only provides marginal benefits, while significantly increasing computational requirements. We also showed that scans have consistent coverage along vulnerable and misconfigured subpopulations, and are therefore appropriate for efficient and accurate assessment of the attack surface of networks.

We intend to apply our developed techniques to develop smart scanners that can efficiently scan IPv4 networks and IPv6 hitlists, increasing the hit rate for discovering active IPs. This allows scanners to scan IPv4 faster and less intrusively compared to exhaustive scans, while covering larger histlists for discovering more devices on IPv6. Additionally, using other sources of information, e.g., historical data, and local patterns in how devices are placed on the Internet (for example some networks might tend to put active devices at the start of their allocated IP blocks, while others might use random placement) can also help improve the efficiency of scans. Note that for sequential scans we are using a static order for probing different ports. However, it might be more efficient to change the order of scans for different networks, for instance scanning modem/router protocols first for consumer networks, while prioritizing web protocols for hosting networks. Using a dynamic order for scans is

another direction for future work.

21.9. Acknowledgments

This work is supported by the NSF under grants CNS1939006, CNS2012001, and by the ARO under contract W911NF1810208.

Bibliography

[1] X. Fan and J. Heidemann, "Selecting representative IP addresses for Internet topology studies," in *ACM SIGCOMM Conference on Internet Measurement*. ACM, 2010, pp. 411–423.

[2] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, "Target generation for Internet-wide IPv6 scanning," in *Internet Measurement Conference*. ACM, 2017, pp. 242–253.

[3] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide scanning and its security applications." in *USENIX Security Symposium*, vol. 8, 2013, pp. 47–53.

[4] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *ACM Conference on Computer and Communications Security*. ACM, 2015, pp. 542–553.

[5] D. Leonard and D. Loguinov, "Demystifying service discovery: Implementing an Internet-wide scanner," in *ACM SIGCOMM Conference on Internet Measurement*. ACM, 2010, pp. 109–122.

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

[7] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the Mirai botnet," in *USENIX Security Symposium*, 2017, pp. 1092–1110.

[8] Y. Liu, A. Sarabi, J. Zhang, P. Naghizadeh, M. Karir, M. Bailey, and M. Liu, "Cloudy with a chance of breach: Forecasting cyber security incidents," in *USENIX Security Symposium*, 2015, pp. 1009–1024.

[9] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey *et al.*, "The matter of heartbleed," in *Internet Measurement Conference*. ACM, 2014, pp. 475–488.

[10] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring HTTPS adoption on the web," in *USENIX Security Symposium*, 2017, pp. 1323–1338.

[11] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of age: A longitudinal study of TLS deployment," in *Internet Measurement Conference*. ACM, 2018, pp. 415–428.

[12] D. Kumar, Z. Wang, M. Hyder, J. Dickinson, G. Beck, D. Adrian, J. Mason, Z. Durumeric, J. A. Halderman, and M. Bailey, "Tracking certificate misissuance in the wild," in *IEEE Symposium on Security and Privacy*. IEEE, 2018, pp. 785–798.

[13] S. Bano, P. Richter, M. Javed, S. Sundaresan, Z. Durumeric, S. J. Murdoch, R. Mortier, and V. Paxson, "Scanning the Internet for liveness," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 2, pp. 2–9, 2018.

[14] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the Internet for ROS: A view of security in robotics research," *arXiv preprint arXiv:1808.03322*, 2018.

[15] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering Internet-of-Things devices," in *USENIX Security Symposium*, 2018, pp. 327–341.

[16] Q. Scheitle, T. Chung, J. Hiller, O. Gasser, J. Naab, R. van Rijswijk-Deij, O. Hohlfeld, R. Holz, D. Choffnes, A. Mislove *et al.*, "A first look at certification authority authorization (CAA)," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 2, pp. 10–23, 2018.

[17] R. Beverly, R. Durairajan, D. Plonka, and J. P. Rohrer, "In the IP of the beholder: Strategies for active IPv6 topology discovery," in *Internet Measurement Conference 2018*. ACM, 2018, pp. 308–321.

[18] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: From art to science," in *Cybersecurity Applications & Technology Conference for Homeland Security*. IEEE, 2009, pp. 205–211.

[19] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 71–74, 2005.

[20] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S. D. Strowes, L. Hendriks, and G. Carle, "Clusters in the expanse: Understanding and unbiasing IPv6 hitlists," in *Internet Measurement Conference*. ACM, 2018, pp. 364–378.

[21] Akamai, "State of the Internet IPv6 adoption visualization," https://www.akamai.com/uk/en/resources/our-thinking/state-of-the-internet-report/state-of-the-internet-ipv6-adoption-visualization.jsp.

[22] Internet Society, "State of IPv6 deployment 2018," https://www.internetsociety.org/resources/2018/state-of-ipv6-deployment-2018.

[23] The ZMap Project, "ZGrab 2.0," https://github.com/zmap/zgrab2.

[24] Maxmind, "GeoLite2 database," https://www.maxmind.com/en/geolite2-developer-package.

[25] Merit Network, https://www.merit.edu.

[26] Team Cymru, http://www.team-cymru.org.

[27] CAIDA, "Routeviews prefix to AS mappings dataset (pfx2as) for IPv4 and IPv6," https://www.caida.org/data/routing/routeviews-prefix2as.xml.

[28] A. Sarabi and M. Liu, "Characterizing the Internet host population using deep learning: A universal and lightweight numerical embedding," in *Internet Measurement Conference*. ACM, 2018, pp. 133–146.

[29] JSON Schema, http://json-schema.org.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[31] Scikit-learn, "Ensemble methods: Random forests," http://scikit-learn.org/stable/modules/ensemble.html#forest.

[32] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.

[33] XGBoost, "Parameter tuning," https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html.

[34] J. Klick, S. Lau, M. Wählisch, and V. Roth, "Towards better Internet citizenship: Reducing the footprint of Internet-wide scans by topology aware prefix selection," in *Internet Measurement Conference*. ACM, 2016, pp. 421–427.

[35] Cloudflare, "DNS amplification (DDoS) attack," https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack.

[36] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, p. 333, 2011.

[37] W. Bi and J. T. Kwok, "Multilabel classification with label correlations and missing labels," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[38] Y.-N. Chen and H.-T. Lin, "Feature-aware label space dimension reduction for multi-label classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 1529–1537.

[39] F. Tai and H.-T. Lin, "Multilabel classification with principal label space transformation," *Neural Computation*, vol. 24, no. 9, pp. 2508–2542, 2012.

[40] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.