PortFiler: Port-Level Network Profiling for Self-Propagating Malware Detection

Talha Ongun, Oliver Spohngellert, Benjamin Miller, Simona Boboila, Alina Oprea, Tina Eliassi-Rad
Northeastern University
Jason Hiser, Alastair Nottingham, Jack Davidson, Malathi Veeraraghavan
University of Virginia

Abstract*—Recent self-propagating malware (SPM) campaigns compromised hundred of thousands of victim machines on the Internet. It is challenging to detect these attacks in their early stages, as adversaries utilize common network services, use novel techniques, and can evade existing detection mechanisms. We propose PortFiler (PORT-Level Network Traffic ProFILER), a new machine learning system applied to network traffic for detecting SPM attacks. PortFiler extracts port-level features from the Zeek connection logs collected at a border of a monitored network, applies anomaly detection techniques to identify suspicious events, and ranks the alerts across ports for investigation by the Security Operations Center (SOC). We propose a novel ensemble methodology for aggregating individual models in PortFiler that increases resilience against several evasion strategies compared to standard ML baselines. We extensively evaluate PortFiler on traffic collected from two university networks, and show that it can detect SPM attacks with different patterns, such as WannaCry and Mirai, and performs well under evasion. Ranking across ports achieves precision over 0.94 and false positive rates below 8×10^{-4} in the top 100 highly ranked alerts. When deployed on the university networks, PortFiler detected anomalous SPM-like activity on one of the campus networks, confirmed by the university SOC as malicious. PortFiler also detected a Mirai attack recreated on the two university networks with higher precision and recall than deep-learning based autoencoder methods.

***Index Terms*—malware detection, security analytics, self-propagating malware, traffic profiling**

I. Introduction

Self-propagating malware (SPM) is a prevalent class of malware that has recently gained in popularity among adversaries. In 2016, Mirai [6] infected more than 600K IoT devices and launched devastating denial-of-service attacks against high-profile websites. In 2017, the WannaCry ransomware attack impacted more than 300K vulnerable devices in 150 countries in a few days [31]. SPM attacks have been seen in the wild since 2001, with famous attacks such as Code Red being successful at widely propagating and infecting machines on the Internet [34]. Their recent surge in popularity can be attributed to an increased number of vulnerabilities being discovered in network services (such as the EternalBlue vulnerability in the SMB protocol exploited by WannaCry and NotPetya), the increasing number of connected devices on the Internet, and the new monetization mechanisms offered by ransomware.

Defending against SPM campaigns is challenging for multiple reasons. A widely-adopted response to stopping WannaCry and NotPetya was to block port 445 allocated to the SMB file-sharing service, a solution adopted today by many ISPs. However, many other SPM attacks leverage common network services that cannot be blocked entirely. For instance, Mirai performed scanning over multiple network protocols, including Telnet, HTTPS, FTP, SSH, and CWMP [6]. As the scanning activities of SPM attacks blend in with large volumes of legitimate traffic, it becomes difficult to distinguish the malicious traffic and block it. Furthermore, attackers behind SPM could employ evasive methods to hinder detection. While the WannaCry and Mirai attacks progressed rapidly to have a significant global impact, future attacks can be more targeted or stealthier by propagating at slower rates. Thus, new techniques are needed to bridge the gap in the defense side against this new wave of automated, self-propagating attacks.

We design PortFiler, a machine learning system that uses network traffic captured at a border of an organizational network to detect and prioritize emerging SPM attacks. To the best of our knowledge, we are the first to design a robust ML system for SPM malware detection that uses network-level information. The critical insight in this work is that SPM requires remote probing of a large number of machines to propagate effectively. This behavior can be viewed as an invariant characteristic of this class of malware attacks since the success of the operation significantly depends on the deployed propagation mechanism. To capture this behavior, PortFiler extracts a set of 35 features from the Zeek connection logs, mapping communication of internal hosts in the network with external destinations. At training time, PortFiler learns the profile of the normal network communication on a set of monitored ports, without relying on traces of existing malware attacks. We design in PortFiler novel ensemble methods that train multiple unsupervised ML models and combine their anomaly scores into a single, unified score. The base models in the ensembles can be instantiated with density-based models such as Kernel Density Estimate (KDE), or unsupervised tree-based methods, such as Isolation Forest. At testing time, PortFiler detects traffic anomalies that exhibit different behavior from training observations, and prioritizes the anomalies via score ranking for investigation by security analysts. Security Operations Centers (SOC) have limited manual investigation budgets, and having a low false positive rate in the top-ranked alerts is a strict requirement for an ML model to be deployed.

Another important requirement in the design of PORTFILER is its resilience to evasion attacks. We experiment with two evasion strategies a motivated attacker might employ: reducing the rate of probing external destinations, and leveraging external destinations already visited by internal machines. We show that standard ML methods degrade in performance as the malware becomes more evasive, but our proposed ensembles are much more robust against evasion. We compare the performance of PORTFILER's ensembles with standard ML models and unsupervised deep learning models based on autoencoders, and show the advantages of our method.

Finally, we evaluate PORTFILER using Zeek [1] logs collected on two university networks. We use public malware traces for four SPM families [22] and generate our own WannaCry malware variants in a virtual environment. We show that both ensemble strategies result in increased resilience against evasion, and they have low false positive rates. We evaluate the top ranked alerts at one of the university networks, and confirm with the SOC that a detection on port 445 is malicious. In addition, PORTFILER detected a Mirai attack recreated on the two university networks with high precision and recall.

We summarize our contributions below:

- We design PORTFILER, a new machine learning system for detecting SPM attacks. PORTFILER learns the normal traffic profiles on selected ports in a network at training time without relying on traces of existing malware attacks, and can detect suspicious traffic anomalies generated by SPM campaigns in testing.
- We propose a novel ensemble methodology for increasing the resilience of PORTFILER against several malware evasion strategies. For example, the ensemble method maintains an Area Under the Precision-Recall Curve of 0.93 for a WannaCry variant 64x slower than the original, an improvement of 31% compared to standard KDE models.
- We develop a methodology for ranking PORTFILER alerts across ports and generating a unified list for investigation by SOC security experts. For the top 100 ranked alerts, the ensemble method achieves a precision above 0.94 with false positive rates below 8×10^{-4} at detecting WannaCry.
- We evaluate PORTFILER on more than 6 billion Zeek logs collected at two campus networks, and confirm our detections as malicious with the SOC. Additionally, we detected a recreated Mirai attack on the two university networks with high precision and recall.

II. PROBLEM DEFINITION AND THREAT MODEL

Self-Propagating Malware (SPM). In SPM attacks, an infected machine attempts to propagate indiscriminately on the Internet, with the goal of spreading the infection widely. SPM attacks involve two phases: (1) *probing*, in which a large number of IPs are probed to identify potential victims; and (2) *propagation*, in which the malware attempts to infect the identified targets. After each successful infection, the malware continues the propagation to other IPs, and also monetizes the infection through different mechanisms.

We describe two famous SPM attacks, WannaCry and Mirai, that have been extremely successful at reaching hundreds of thousands of victims on the Internet. During probing, WannaCry probes machines on the local network (*internal probing*), and also probes pseudorandomly-generated IPs on the Internet (*external probing*). WannaCry is mostly known for its ransomware behavior: after propagation, it encrypts the files on the victim machine and displays a ransom message to the user. The Mirai campaign is mostly known for its large-scale DDoS attacks performed after getting a large victim base of IoT devices. To get access to these IoT devices, Mirai performed rapid probing on ports 23 and 2323 (later expanded to other ports) using pseudorandom IP addresses [6]. After Mirai identifies a potential victim IP, it uses a brute-force login phase over Telnet, using common user names and passwords. If the login is successful, device-specific malware is downloaded to each victim. Once the campaign controls a large number of victim devices, they are instructed by the command-and-control center to launch DDoS attacks.

We highlight that while the end goal of these two campaigns is completely different – WannaCry is a ransomware, while Mirai launches DDoS attacks – both of them use self-propagating behavior to reach a large number of victims on the Internet. In fact, other SPM families we surveyed in Table I exhibit similar behavior as WannaCry and Mirai. We obtained public data for 5 different examples of SPM malware [22] and show the scanning rate (IPs probed per minute), as well as the entropy of the probed IP addresses (split into 4 octets) in the same table. We observe that all 5 malware families have entropy close to 8 and the entropy of the uniform distribution is $\log_2 256 = 8$, which implies that the IP addresses they probe are uniformly distributed. The main differences are in the scanning rate and ports, two properties that we vary in our experiments to capture a range of SPM behavior.

Problem definition and threat model. We aim to detect SPM attacks in their very early stages of operation, during the initial probing and propagation phases, to prevent the spread of SPM attacks on the Internet. We assume that the attack compromises one or several victims inside the monitored network and we aim to detect SPM behavior without knowledge of the port and service on which the malware propagates. We rely on Zeek network logs collected at the border of the monitored network, which we assume are not compromised by the attacker. Thus, the information recorded in the logs is reliable and can be used for analysis. Our goal is to handle advanced SPM attackers, even those that employ evasion strategies against our detection methods. We assume that attackers may have knowledge about the features and ML models used for detection. We design machine learning models for SPM detection that are robust against different evasion strategies.

Challenges. ML for malware detection introduces well-documented challenges, such as limited availability of ground truth for training supervised models, strict requirements on low false positive rates, and resilience to evasion [28]. Detecting new SPM on common network ports (80 or 443) is particularly

Malware	**Ports**	**Scan Rate (IPs per min)**	**Entropy (IP octets)**	**Infection Method**
Mirai [9]	23, 2323	50-300k	7.9829	Dictionary attack
Hajime [15]	23, 81	10k	7.9822	Vulnerabity Exploit: Dictionary attack, GoAhead-Webs credentials
Kenjiro [29]	80, 8080, 37215	5k	7.9832	Vulnerabity Exploit: CVE-2016-6563, CVE-2017-17215
WannaCry [31]	445	1750	7.9734	Vulnerabity Exploit: CVE-2017-0143
Hide and Seek [2]	23, 9527	260	7.9726	Dictionary Attack, Vulnerabity Exploit: CVE-2017-11634

TABLE I: Comparison of SPM scanning patterns. We surveyed 5 malware families exhibiting various scanning rates (number of IPs contacted by a victim per minute) on different ports. A common characteristic of SPM malware is the probing of randomly distributed IP addresses, demonstrated by the entropy values. In our experiments, these parameters (i.e., scanning rate and ports) are varied to capture a range of SPM behaviors.

difficult given the vast amounts of network traffic on these ports and the high variation of traffic patterns.

III. PortFiler System Overview and Methodology

A. Overview

Figure 1 shows an overview of our system. PortFiler trains newly-introduced unsupervised ensemble models that learn the legitimate network traffic distribution, and applies them at testing time to identify and rank anomalies. A prioritized list of alerts is presented to SOC for investigation.

Network data monitoring. Zeek [1] is a well-known network monitor that processes raw packet captures to generate network logs. For PortFiler, we use the Zeek connections logs (conn.log files) that record connection metadata for both TCP and UDP connections crossing the border of two university networks we monitor. The fields recorded in conn.log include a timestamp of connection start, source and destination IP and port, transport protocol, duration of connection, source and destination payload bytes, number of packets, as well as connection state.

Model training. For the ports of interest, PortFiler profiles the traffic and trains ensemble models for learning the regular communication patterns. We define a set of 35 features for each port and time window of fixed length. To learn the distribution of the traffic features per port, we introduce a new methodology for creating ensembles of anomaly detectors that combine anomaly scores produced by multiple ML models, increasing resilience against evasion. The individual models in the ensemble can be either density estimation models, such as Kernel Density Estimation (KDE) [26] or tree-based models, such as Isolation Forest (IF) [19].

Ranking alerts. At testing time, we apply the trained ensemble models on new Zeek conn.log data. We also develop methodology to rank the alerts across ports and prioritize them for investigation by SOC.

Ethical considerations. The IP addresses of the internal machines are anonymized in a consistent manner to protect personal information about the machines or users on the network. We performed all our analysis on servers within the university network, without downloading the data locally. The IRB office at one of the universities reviewed our data collection and anonymization process and determined that our research does not qualify as Human Subject Research.

B. Port-Based Traffic Features

We would like to capture the traffic patterns generated on the monitored network, and create features that distinguish SPM behavior from regular communication patterns. An important consideration when defining our set of features is to profile the traffic characteristics of different applications running in a network. As applications are assigned specific ports, we choose to define the PortFiler features at the port level. This also enables our system to be lightweight, instead of computing features for each individual host in the network, which would be computationally expensive. We consider five ports of interest: 445 (the SMB port, used originally by the WannaCry malware), 80 (HTTP), 443 (HTTPS), 22 (SSH), and 23 (Telnet, used by Mirai). We selected these ports after analyzing several SPM malware (see Table I), but our models can be easily extended to monitor a larger set of ports for SPM-like behavior.

We aggregate all logs for a fixed time window and extract a set of 35 features. These capture the following statistics on the set of connections on a particular port:

Traffic statistics features: We extract several traffic statistics features: number of distinct internal and external IPs communicating on that port, number of connections, and number of new distinct external IPs (that have not been contacted before) per port. We expect to observe an increase in these features during periods with high SPM activities.

Duration features: We extract max, min, variance and mean of connection duration values during each time window. SPM connections have a lower duration than most legitimate traffic.

Bytes and packets features: We extract max, variance, and mean of sent and received byte and packet values. The distribution of bytes sent and received during SPM infection might be different compared to normal periods, and similarly for the packet distribution. We also define "number of connections with no bytes received" as most of SPM connections are for non-responding IPs and result in zero bytes received.

Connection state: We define the number of connections for each state (e.g., S0, S1, OTH, etc.). In particular, a large number of failed connections might be indicative of a large number of SPM probing and propagation attempts. We also define the number of failed connections, a feature that aggregates multiple connection failure states.

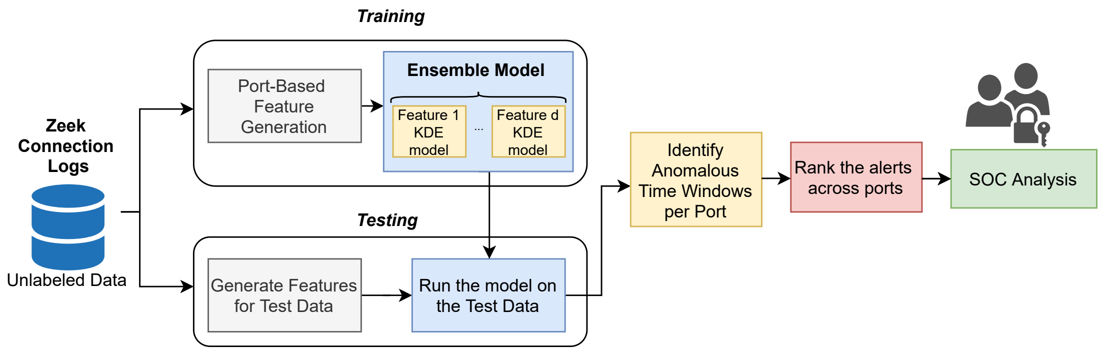
Training
Zeek Connection Logs
Unlabeled Data
Port-Based Feature Generation
Ensemble Model
Feature 1 KDE model
...
Feature d KDE model
Testing
Generate Features for Test Data
Run the model on the Test Data
Identify Anomalous Time Windows per Port
Rank the alerts across ports
SOC Analysis

Fig. 1: PORTFILER System Overview. PORTFILER extracts a set of 35 features for a fixed time window and port. During training, it learns the legitimate distribution of network traffic on each port, using our ensemble methodology. During testing, anomaly scores are generated on new network logs, and the most anomalous connections are prioritized for SOC investigation.

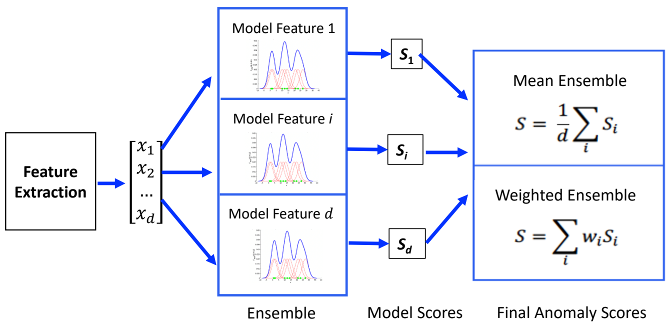

Fig. 2: PORTFILER overview of the novel ensemble methods we introduce. At training, features are extracted per port and time window. A separate ML model is trained on each of the d features. At testing, new data is evaluated against all the d models, to produce anomaly scores $S_1, \ldots, S_d$. The scores are combined into a final anomaly score S either by a mean ensemble (using equal weights) or weighted ensemble (using pre-defined weights).

We highlight an engineered feature that consistently shows highest feature importance in our evaluation. The *new external IPs* feature represents the number of new external IPs, (i.e., IPs that have not been seen before) contacted per port within each time interval. To compute this feature, we create a history of visited external IPs and update it over time. This feature captures the randomness in SPM propagation behavior, which results in many new, previously unvisited IP destinations.

C. Novel Unsupervised Ensemble Models

Network traffic profiling can be performed using a range of unsupervised ML models. One of our main design criteria, as mentioned, is the resilience of our models to advanced malware evasion strategies. As we show in our evaluation, standard ML models fail in the face of evasive SPM. This motivates us to introduce our new ensemble methodology for learning network traffic distribution, while achieving resilience against evasion. We implement and evaluate several existing ML models for comparison to PORTFILER's ensembles. These include a simple threshold-based baseline method, standard Kernel Density Estimate (KDE), Isolation Forest (IF), and an unsupervised deep learning method based on autoencoders.

Ensembles of multiple ML models have been used extensively in supervised learning tasks for improved generalization, but not as much in unsupervised learning tasks like ours. The distribution learning task is particularly challenging due to feature correlation, which can degrade performance of multi-feature models. When analyzing the network logs, we observed that many of PORTFILER features are naturally correlated. An ensemble of models trained on individual features addresses this issue, since each individual ML model learns a single-dimensional distribution, a much more tractable task. Similar to standard bagging methods for supervised learning, our proposed ensembles generalize better at testing time [8], which also implies more resilience under adversarial evasion, as shown in our experiments.

Figure 2 gives an overview of our proposed ensemble method for network traffic profiling. During training, individual ML models are trained on each traffic feature and an ensemble of all these models is generated. If training data samples $x_i = (x_{i1}, \ldots, x_{id})$ are d-dimensional for $i \in \{1, \ldots, N\}$, we train d unsupervised ML models: $f_1, \ldots, f_d$, each f_j taking feature j of a sample as input. In our best-performing ensemble, we instantiate the individual base models f_j with KDE models (but we experimented with ensembles of Isolation Forest models as well). Model f_j estimates the distribution's density by averaging the outputs of a kernel function K, applied to data centered at each point x_{ij} in the sample, and having the same bandwidth h. The estimated density f_j is: $f_j(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_{ij}}{h})$. A common choice for kernel functions is the Gaussian kernel, which uses the standard normal density as the kernel function.

At testing time, each model generates an anomaly score: $S_j = f_j(x)$ for a new sample x. If the probability density falls below a threshold T ($f_j(x) < T$), we consider x as an anomaly. The threshold T is determined at training time based on the legitimate data to fix the false positive rate, and we vary it in our experiments. Finally, we combine the anomaly scores generated by individual models into a single *ensemble anomaly score* using weights w_i:

$$\mathsf{AnomalyScore}(x) = \sum_{i=1}^{d} w_i S_i$$

We consider two ensemble methods, which differ in model weight assignments:

1) Mean Ensemble: Each ML model contributes equally to the final anomaly score. This is useful when no a priori information is known about the attack.

2) Weighted Ensemble: This uses a weighted combination of models for computing anomaly scores. This method has significant advantages in cases in which some information about the attack becomes available, and some features have higher relevance for a particular attack.

Model weight computation. An interesting property of weighted ensembles is that they can be adapted to different attacks, by assigning higher weights to more relevant features. The model weights can be computed with a variety of methods. For instance, domain experts could assign weights manually based on attack forensic analysis. Instead, we wanted to find an automated procedure to assign model weights. That led us to a supervised approach for weight computation, in which we assume the availability of an attack trace, which is a variant of the attack we are interested in (we generate multiple variants of WannaCry in our experiments by varying propagation rate and interval between probes). We use the labeled attack variant merged with one day of legitimate data and train a Random Forest classifier. We compute feature importance for the classifier and assign weights proportional to these values. This procedure identifies the features that have high importance for the attack, and assigns higher weights to the corresponding models in the ensemble.

D. Ranking

Motivated by constraints on SOC investigation time, our goal is to prioritize the alerts generated by our models and provide the highly ranked alerts to human experts. When each port is analyzed individually, ranking alerts is based directly on anomaly scores: the lower the score, the more anomalous a sample is. In reality, SOC analysts might not know in advance which ports are exploited by SPM, and analyzing alerts on each port is time consuming. To address this issue, we rank the alerts *across all ports* to generate a unified list of the most suspicious alerts an analyst should investigate. The standard KDE model returns probability densities, which are aggregated by the ensemble, but are not directly comparable across ports. To address this issue, we normalize the probability densities by computing the Complementary Cumulative Distribution Function (CCDF) for each sample x.

For ensembles using KDE as the base ML model, we compute weighted combinations of CCDF scores for ranking. If $f_1, \ldots, f_d$ are the d base models in the ensemble, the score of a data point $x = (x_1, \ldots, x_d)$ is:

$$C(x) = \sum_{i=1}^{d} w_i \mathsf{CCDF}(x_i) = \sum_{i=1}^{d} w_i (1 - \int_{-\infty}^{x_i} f_i(x))dx$$

Thus, we aggregate the CCDF values using the ensemble weights. We use this normalized anomaly score to rank across ports and prioritize the alerts with the lowest scores across all ports. For ensembles using Isolation Forest as the base ML model, normalization is not necessary, since the score represents the path length from the root node to the terminating node (rather than a probability density). The shorter the path, the more anomalous a sample is. Thus, we can compare and rank the weighted ensemble scores directly across ports.

IV. Experimental Setup

A. Datasets for Evaluation

Network logs at two university networks. The network traffic used for evaluation is the anonymized Zeek conn.log data provided by two universities: University of Virginia (Univ-1) and Virginia Tech (Univ-2). Our experimental setup consists of one week of training and one day of testing, resulting in a total of 1.5 TB of data and 9.64 billion events for Univ-1, and 1.2 TB of data and 9.69 billion events for Univ-2. For each day of training and testing, we extract the 35 features on the five ports of interest. Processing one day to extract features takes on average 1.7 hours for Univ-1 traffic and 0.4 hours for Univ-2 traffic.

Malicious Datasets. We obtained access to a public dataset including the SPM families from Table I [22]. We determined that all malware families have entropy of the external IPs close to that of the uniform distribution, confirming that the IP addresses they connect to are chosen pseudorandomly. The scanning rate and ports vary across families. In these traces, the attack is limited to a single infected IP and usually has small duration (less than an hour). We determined that WannaCry is a representative SPM family, with relatively slow scanning rate compared to others, and decided to use it to generate more realistic SPM samples. We set up a virtual network with 50 virtual machines running on three physical machines. Each VM ran a vulnerable version of Windows, which can be exploited by EternalBlue. We isolate the virtual environment by blocking the external network traffic, but we allow internal connections between the virtual machines. We run the original WannaCry attack for two hours and infect 48 machines on a subnet. We also generate other variants with different scanning rates to test the resilience of our methods to evasion. We capture full pcaps of the network traffic, from which we generate Zeek conn.log.

B. Evasion Strategies

In reality, advanced attackers can employ evasion strategies to make the detection of SPM traffic more difficult. We experiment with two evasion strategies:

Method 1: Slowing down the probing rate. An adversary with no knowledge about the ML models could attempt a straightforward strategy to lower the scanning rate and become more stealthy. The original WannaCry attack contacts on average 14K external IPs per minute and runs for two hours. We generate new variants, where each infected machine decreases its probing rate by a factor of 2, 4, 8, 16, 32, 64, or 128 by dropping a fraction of connections uniformly at random.

Method 2: Leveraging IP destinations from history. One of the most relevant features in our models is the number of "new IP" destinations, which are IP addresses not visited before on that particular port. SPM is likely to generate a large number of new IPs if the probed IPs are generated at random, as done by both WannaCry and Mirai. An attacker with knowledge of our feature set could attempt to evade this feature by probing already visited IP addresses. Starting from the malware variant 4 times slower than the original, we select variants that use a factor 2, 4, 8, 16, 32, 64, or 128 new IPs, while the rest are are previously-visited IPs from the history.

V. Evaluation

We perform an in-depth evaluation of our proposed ensemble methods against several baselines for detecting SPM and show the higher resilience of ensembles against evasion strategies. Given that our data is imbalanced, we use Area Under the Precision Recall Curve (PR-AUC) for evaluation. We are also interested in minimizing the false positive rate to account for the limited budget of SOC investigations.

A. Baselines and Standard ML Detectors

Setup. For these experiments, we merge the malicious traces at testing time to generate controlled experiments with attack ground truth. We overlay the malicious traces onto the network traffic from the testing day by selecting random internal IPs in the network, and merging the legitimate and malicious conn.log data. When merging the malicious traffic, we preserve all the attributes of the connections.

Baseline Threshold Detector. We first experiment with a threshold-based baseline detector that marks a time window as anomalous if the number of connections on the monitored port exceeds a threshold. The connection threshold is varied in order to obtain a full precision-recall (PR) curve. The threshold-based detector performs poorly even on the original WannaCry variant. For instance, on port 22 the PR-AUC is as low as 0.39, while for port 443 the PR-AUC is 0.5. The only port where the threshold-based approach performs well is 445 because it is blocked at Univ-1 and there is very little regular traffic. These numbers reduce considerably for the 8x slower variant, resulting in an PR-AUC of just 0.09 on port 23.

Dimensionality Reduction. Dimensionality reduction methods, such as PCA, UMAP, and t-SNE, are used extensively for anomaly detection. We performed experiments with both t-SNE and UMAP, but only present results for t-SNE. Figure 3 shows a scatterplot of the two principal components of malicious and background samples after t-SNE for slow variants (1/128 rate) of both WannaCry and Mirai on port 80. We observe that most of the malicious samples overlap with background samples, and there is no clear separation. The results were similar for other ports and SPM variants. These results indicate that off-the-shelf anomaly detection algorithms are less likely to find malicious samples with high accuracy, and more guided approaches are required for our system.

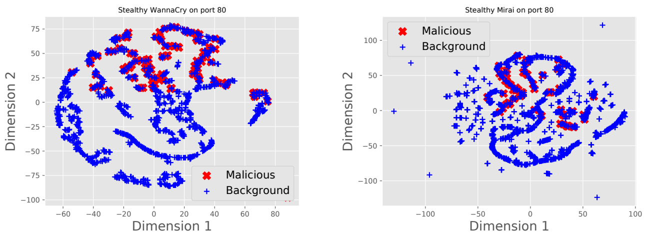

Fig. 3: t-SNE visualization of the 35-feature representation of slow WannaCry (left) and Mirai (right) at Univ-1 on different ports. Malicious samples overlap with background samples, indicating that off-the-shelf anomaly detection algorithms are less likely to find malicious samples with high accuracy.

Baseline ML Models. As simple methods are ineffective for our problem of detecting SPM attacks, we are motivated to employ ML-based approaches for learning the distribution of network traffic on each port. We evaluate the standard KDE and Isolation Forest (IF) ML models trained with our 35 features previously described, using the Univ-1 dataset, after doing an extensive hyper-parameter search. In all our experiments, we selected a time window for feature extraction of one minute (after experimenting with several values: 30 sec, 1 min, 2 min, 5 min, and 10 min). Thus each day of data has 1440 time windows per port. We discuss here results on Univ-1, but results on Univ-2 are similar.

We found that the KDE model performs better than IF, and both methods improve upon the threshold-based detector. IF has the lowest PR-AUC on port 443, which is the highest volume port, with PR-AUC reaching 0.58 for WannaCry and 0.62 for Mirai. KDE has PR-AUC scores higher than 0.99 on all ports for the original WannaCry and Mirai attacks. Similarly, we obtained PR-AUC score of 1.0 using the Kenjiro and Hajime malware samples described in Table I.

However, neither of these standard ML models are resilient against the evasion strategies described in Section IV-B: *Method 1: Slowing down the scanning rate*, and *Method 2: Leveraging IP destinations from history*. To evaluate evasion resilience, we use the representative WannaCry malware we generated and its evasive variants. Figure 4 shows the change in the KDE PR-AUC score as we decrease the probing rate of WannaCry (*Method 1*), and the new IP rate (*Method 2*). It is apparent that the standard KDE model is not resilient to these evasion methods. For example, by slowing down by a factor of 64, the PR-AUC score of KDE reaches around 0.2 on ports 23, 80, and 443. While ports 445 and 22 show less impact, the scores decrease more significantly on ports with large amount of traffic. We obtained similar results for Mirai, and we believe that other SPMs will exhibit similar behavior under evasion.

B. PortFiler: Ensemble Models

Resilience to Evasion. We designed ensembles of multiple models with the goal of increasing the system's resilience to evasion. Experiments in Figure 5 use weighted ensembles

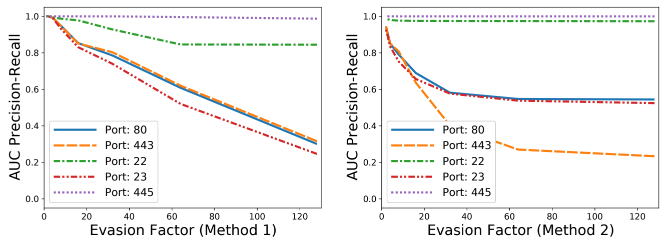

Fig. 4: Baseline KDE model. Robustness against evasion is generally low for Evasion Method 1 (slowing down the probing rate) and 2 (leveraging IP destinations from history). Performance drops significantly as the evasion factor increases.

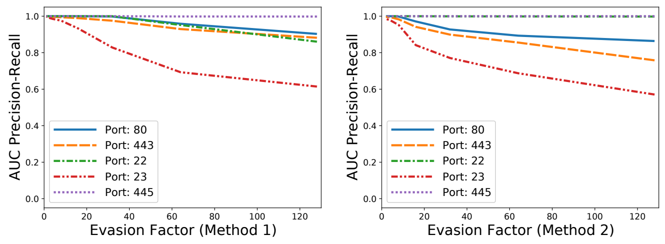

Fig. 5: Weighted Ensemble of KDEs. Ensemble model is more resilient against evasion as we learn how to weight each model properly. Evasion Method 1 (slowing down the probing rate) and 2 (leveraging IP destinations from history) are examined.

of KDEs, where the weights represent normalized feature importance coefficients. These coefficients are computed using a variant of WannaCry that we generated (see Section III-C). Figure 5 shows that the weighted ensemble is resilient against both evasion strategies on all ports. For instance, at a propagation rate 128 times slower than the original WannaCry, it still maintains PR-AUC above 0.8 for ports 80 and 443.

Figure 6 presents a direct comparison of weighted ensembles against the baseline KDE model. The PR curves demonstrate the significant improvement of weighted ensembles on port 443 using the WannaCry variant with the evasion factor 64. The weighted ensemble achieves PR-AUC of 0.93 and 0.86 for the two evasion methods, compared to values of 0.62 and 0.27 for baseline KDE. We also evaluated the mean ensemble method, which uses uniform weights for all models. Mean ensembles are more resilient than baseline KDE on ports 80 and 443, which are the most challenging to protect against evasion, given the large volume of legitimate traffic on these ports. However, of all methods explored, weighted ensembles are the most resilient against evasion. We obtained similar results for ensembles of IF models and for evasive Mirai variants, which we omitted here for lack of space.

Ranking Evaluation. We evaluate our method for ranking the alerts across all ports, as described in Section III-D. In this experiment, the attack sample is merged on one of the ports, and the most suspicious samples on all five ports are ranked to provide a unified alert list. Table II shows the statistics of the top-100 ranked alerts, which are ranked using the weighted

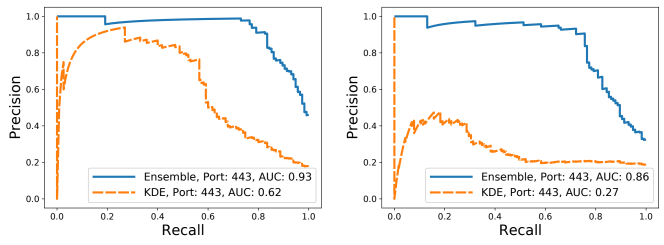

Fig. 6: Precision-Recall curves demonstrate the improved performance of weighted ensembles compared to the baseline KDE model on port 443 using the WannaCry variant with evasion factor 64. Graphs for both evasion methods are included.

	Original Wannacry		*Slow WannaCry*	
Port	**Prec**	**FPR**	**Prec**	**FPR**
80	1.0	0	0.93	0.0009
443	1.0	0	1.0	0
22	0.95	0.0007	0.94	0.0008
23	0.94	0.0008	0.73	0.005
445	1.0	0	0.99	0.0001

TABLE II: Precision (Prec) and False Positive Rate (FPR) in the top-100 ranked alerts across the ports for the original and slow WannaCry (1/8 rate), demonstrating the strength of weighted ensembles of KDEs with low FPR.

ensembles of KDE models to detect the original and 8x slower WannaCry variant. The metrics are defined based on 7200 samples over 5 ports, with 116 of them malicious. We observe that precision in the top-100 alerts is very high (100% on ports 80, 443, and 445), and the minimum is 94% on port 23. In addition, the False Positive Rate is lower than 8×10^{-4} for the original WannaCry. Results are similarly promising on the slower variant. We also experimented with weighted ensembles of IF models, and the results are slightly worse.

C. Deployment on Two University Networks

Malware Detection "in the Wild". We deployed PORTFILER on two university networks, UNIV-1 and UNIV-2, in order to evaluate our system in a real setting. No merging of malicious traffic was performed in this experiment. We rank the alerts, and consider the top 10 highest risk alerts on both networks. At UNIV-2, PORTFILER detected a series of anomalies on September 9 on port 445, between 11:29am and 12:51pm. During this time period, the new IP feature significantly increased. After additional investigation, we discovered that all the top 10 detections were part of the same attack. A single internal IP attempted to connect to over 15K external destinations, of which at least 14K failed. This probing behavior appears to be identical to what we observe in the WannaCry logs we collected (in terms of timing patterns, random IP selection, and packet sizes), and we thus suspect this internal IP was infected with WannaCry. The SOC at UNIV-2 confirmed that the activity was indeed malicious.

Mirai Attack Recreation and Detection. We extended our evaluation by leveraging an actual Mirai attack recreated on the two university networks that lasted 5 days. At UNIV-1, two Openstack networks were set up, consisting of 142 vulnerable VM nodes. At UNIV-2, 3 vulnerable VMs with public IPs were configured. In coordination with the security teams at

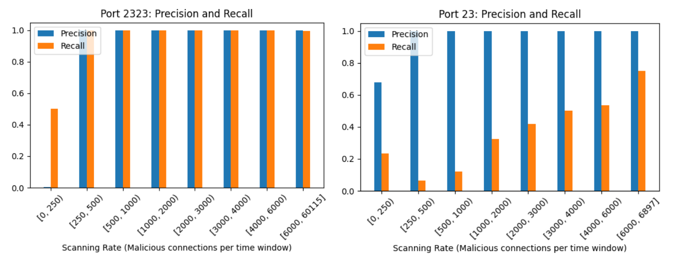

Fig. 7: Detection performance by scanning rate for ports 2323 and 23 using Mirai attack recreation. PORTFILER detects the attack successfully on port 2323 for scanning rates higher than 250 connections per minute. On port 23, it exhibits increased recall at higher scanning rates.

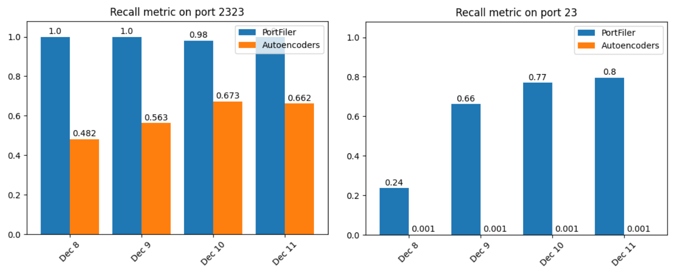

Fig. 8: Recall per day showing superior performance of PORTFILER (mean ensemble method) compared to an unsupervised deep learning approach (autoencoders) on ports 2323 and 23 using Mirai attack recreation.

both UNIV-1 and UNIV-2, the open source Mirai malware was modified to prevent any potential security side effects on the network by allowing infection only in the intended, vulnerable VMs. After infection, nodes communicated to a command-and-control server under our control and attempted to infect other nodes through bi-directional scanning between UNIV-1 and UNIV-2. The entire public IP space of 500K IPs at both networks was scanned during the attack. Mirai was configured to have a fast scanning behavior on port 2323 (90% of scanning) and a slow scanning behavior on port 23 (10% of scanning). The legitimate and malicious traffic was captured at the network edge via Zeek logs for the entire duration of the experiments. We added capability to Mirai to label the malicious scanning traffic to obtain ground truth for evaluation.

We deployed PORTFILER directly on UNIV-1 using the mean ensemble model, attempting to detect the recreated Mirai attack. The attack starts slowly with a single infected node and the scanning rate (total number of scanning requests per minute) increases as the attack progresses. Figure 7 shows the precision and recall metrics at various scanning rates. On port 2323, which has less legitimate traffic, PORTFILER is able to detect the attack very well throughout the experiment, even at low scanning rates. On port 23, performance improves as the scanning rate increases, reaching almost 0.8 recall at a scanning rate of 6000 connections per minute.

Comparison with Unsupervised Deep Learning Models. Using the recreated Mirai attack, we further deepened our analysis of unsupervised malware detection methods. To this end, we conducted several experiments to provide a baseline comparison between PORTFILER and an off-the-shelf unsupervised deep learning approach. We designed a feature space equivalent to PORTFILER's, by using the same fields from Zeek connection logs. Numerical fields (i.e., duration, bytes, packets) were imported directly and categorical fields (i.e., connection state) were one-hot encoded. IP fields were omitted given the difficulty of one-hot encoding the extremely large sample space. This resulted in 19 features per connection log. Following a standard autoencoder architecture, we vectorize the input by concatenating a number of feature rows (i.e., 100 connection logs) together, resulting in 1900 features. Through an extensive hyperparameter search we established that a one-layer architecture with 128 hidden nodes and no regularization leads to the smallest train/validation loss and reconstruction error. We compare this baseline deep learning approach against PORTFILER's mean ensemble method, which does not require any information about the attack. We assume a fixed budget of false positive alerts and evaluate the precision-recall metric for the two methods. We experimented with multiple false positive alert budgets, and the findings were consistent across the board: while precision is high for both methods, the recall metric is much lower for autoencoders due to a high number of misclassifications as false negatives. Figure 8 shows detection results for a fixed budget of 144 false positive alerts per day.

There are also practical limitations for deep learning approaches, including higher execution times and memory requirements. The input of autoencoders is not aggregated, in contrast to PORTFILER, which uses aggregated features. For ports with a high traffic volume, processing the entire train/test data in memory on a system with 64GB of RAM was not possible, which prompted us to employ batching strategies. These experiments indicate that a basic unsupervised deep learning model is not sufficient to effectively detect SPM attacks. Moreover, deep learning models are much more resource intensive compared to PORTFILER.

VI. RELATED WORK

ML malware detection. ML for security has been an active area of research, focusing on various attacks and malware types. Perdisci et al. [23] and Rafique et al. [24] investigate unsupervised network analysis to build a malware detection system. Botnet detection has been proposed based on network behavior [11], [12], [5] and C&C patterns [7], [14], [25]. Alahmadi et al. [5] use connection state patterns to detect bots using Markov models. Enterprise security analytics has been an active research area [16], [21], [33] and ML solutions have been deployed in the industry [3].

Self-Propagating Malware detection. Internet worms have been studied in the past as early examples of SPM [30]. Detection mechanisms based on payload signatures [27], [17], [20], and fine-grained host-level network monitoring [13], [32] have been proposed. Our system is designed independently of specific malware, and is more scalable compared to host-level monitoring. Recent work proposed detection models for SPM such as WannaCry [4], [10] and Mirai [18], focusing mostly on binary analysis, host-based signals or fingerprinting certain network behavior using DNS requests. Akbanov et al. [4] propose detection methods against WannaCry using software-defined networking to mitigate the threat. Kumar et al. [18] discuss detection of Mirai at the probing phase by leveraging Mirai traffic signatures. They sample the packets transmitted by IoT devices both across time and device to find probing activities matching Mirai's behavior.

VII. Conclusion

Self-propagating malware is a prevalent threat on the Internet. Recent campaigns demonstrate that SPM can propagate fast and cause global disruption. We propose PortFiler, an ML-based anomaly detection system on network traffic for detecting SPM. PortFiler extracts port-based features from network logs, profiles legitimate activity on network ports using newly designed unsupervised ensemble methods, and ranks anomalies for SOC investigation. We evaluate PortFiler using Zeek network logs from two university networks and several SPM families, and show its ability at detecting SPM with high precision and recall, and low false positives. We also show the resilience of our newly-introduced ensemble methods against evasion and its benefits compared to standard ML and deep learning methods.

Acknowledgements

This research was sponsored by the contract number W911NF-18-C0019 with the U.S. Army Contracting Command - Aberdeen Proving Ground (ACC-APG) and the Defense Advanced Research Projects Agency (DARPA), and by the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ACC-APG, DARPA, Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. This project was also funded by NSF under grant CNS-171763.

References

[1] The Zeek Network Security Monitor. https://zeek.org.
[2] Tracking the Hide and Seek Botnet - MalwareTech, 2019.
[3] Machine Learning: Symantec's Past, Present, and Future, Jun 2020. https://symantec-enterprise-blogs.security.com/blogs/feature-stories/machine-learning-symantecs-past-present-and-future [accessed 11. Jun. 2021].
[4] M. Akbanov, V. G. Vassilakis, and M. D. Logothetis. Ransomware detection and mitigation using Software-Defined Networking: The case of WannaCry. *Computers & Electrical Engineering*, 76:111–121, 2019.
[5] B. A. Alahmadi, E. Mariconti, R. Spolaor, G. Stringhini, and I. Martinovic. BOTection: Bot detection by building Markov Chain models of bots network behavior. In *ACM ASIA CCS*, 2020.
[6] 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 Symp.*, 2017.
[7] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel. DISCLOSURE: Detecting botnet command-and-control servers through large-scale NetFlow analysis. In *ACSAC*, pages 129–138, 2012.
[8] P. Bühlmann, B. Yu, et al. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.
[9] E. Bursztein. Inside the infamous Mirai IoT Botnet: A Retrospective Analysis. *Cloudflare Blog*, Aug 2020.
[10] Q. Chen and R. A. Bridges. Automated behavioral analysis of malware: A case study of WannaCry ransomware. In *ICMLA*. IEEE, 2017.
[11] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symp.*, 2008.
[12] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *USENIX Security Symp.*, volume 7, pages 1–16, 2007.
[13] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley. Worm detection, early warning and response based on local victim information. In *20th Annual Computer Security Applications Conference*, pages 136–145. IEEE, 2004.
[14] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *NDSS*, 2008.
[15] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin. Measurement and analysis of hajime, a peer-to-peer iot botnet. In *NDSS*, 2019.
[16] X. Hu, J. Jang, M. P. Stoecklin, T. Wang, D. L. Schales, D. Kirat, and J. R. Rao. Baywatch: robust beaconing detection to identify infected hosts in large-scale enterprise networks. In *DSN*. IEEE, 2016.
[17] H.-A. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX security symposium*, volume 286. San Diego, CA, 2004.
[18] A. Kumar and T. J. Lim. Early detection of mirai-like iot bots in large-scale networks through sub-sampled packet traffic analysis. In *Future of Information and Communication Conf.*, pages 847–867. Springer, 2019.
[19] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *8th IEEE Int'l. Conf. on Data Mining*, pages 413–422. IEEE, 2008.
[20] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 226–241. IEEE, 2005.
[21] A. Oprea, Z. Li, R. Norris, and K. Bowers. MADE: Security analytics for enterprise threat detection. In *ACSAC*, pages 124–136, 2018.
[22] A. Parmisano, S. Garcia, and M. J. Erquiaga. Stratosphere laboratory. a labeled dataset with malicious and benign IoT network traffic. *https://www.stratosphereips.org/datasets-iot23*, January 2019.
[23] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *NSDI*, 2010.
[24] M. Z. Rafique and J. Caballero. Firma: Malware clustering and network signature generation with mixed network behaviors. In *RAID*, 2013.
[25] C. Rossow and C. J. Dietrich. Provex: Detecting botnets with encrypted command and control channels. In *DIMVA*, pages 21–40. Springer, 2013.
[26] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.
[27] S. Singh, C. Estan, G. Varghese, and S. Savage. The earlybird system for realtime detection of unknown worms (technical report cs2003-0761). *University of California, San Diego, USA*, 2003.
[28] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symp. on security and privacy*, pages 305–316. IEEE, 2010.
[29] A. Spadafora. Hakai IoT botnet infects popular router brands. *ITProPortal*, Sep 2018.
[30] S. Staniford, V. Paxson, N. Weaver, et al. How to own the internet in your spare time. In *USENIX security symposium*, volume 2, pages 14–15, 2002.
[31] Symantec Security Response. What you need to know about the wannacry ransomware. https://symantec-blogs.broadcom.com/blogs/threat-intelligence/wannacry-ransomware-attack, 2017.
[32] J. Xia, S. Vangala, J. Wu, L. Gao, and K. Kwiat. Effective worm detection for various scan techniques. *Journal of Computer Security*, 14(4):359–387, 2006.
[33] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *ACSAC*, pages 199–208, 2013.
[34] C. C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *CCS*, page 138–147. ACM, 2002.