Behavioral Anomaly Detection of Malware on Home Routers

Ni An*, Alexander Duff†, Gaurav Naik†, Michalis Faloutsos‡, Steven Weber*, Spiros Mancoridis†
*Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA
†Department of Computer Science, Drexel University, Philadelphia, PA
‡Department of Computer Science and Engineering, University of California Riverside, Riverside, CA

***Abstract*—The Internet of Things (IoT) introduced new targets and attack vectors for malicious actors who infect insecure devices with malware in order to form large botnets that can launch distributed denial of service (DDoS) attacks. These botnets comprise various infected devices such as Internet-connected cameras and home routers. This paper focuses on the unsolved problem of creating robust malware detection to secure home routers. This research compares the effectiveness of three different approaches to behavioral malware detection on home endpoint routers through the observation of kernel-level system calls on these routers: i) one-class support vector machines, ii) principal component analysis, and iii) a naive anomaly detector based on unseen n-grams.**

I. Introduction and motivation

The term *Internet of Things* (IoT) refers to the growing network of "smart objects," which are computer systems embedded into everyday things to give them sensing and actuating capabilities in order to perform specific functions and share data remotely [1]. As more devices include Internet connectivity, digital storage, and processing capability, more daily tasks now include the use of computers in the form of these embedded systems. The increased convenience and efficiency provided by IoT devices have made them commonplace in recent years, but the adoption of IoT devices by the general public has also led to the rise of new security concerns as devices that traditionally were not computers are now vectors for malware. Over the last few years, malware-infected IoT devices have been increasingly used for launching DDoS attacks, often without their owners knowing that their devices have been compromised [2].

Recently, malicious actors have used botnets comprised of malware-infected IoT devices, such as Internet-connected appliances and home routers, to great effect. These devices are attractive targets for malware due to their lack of cryptographic encryption, or weak default authentication [2]. Among attacks that have exploited IoT devices is a botnet malware family named *Mirai*, which has caused notorious consequences, and brought the most massive DDoS attack to date [2]. On September 20, 2016, the website of security blogger Brain Krebs was taken offline by a massive DDoS attack on the content delivery and cloud service provider *Akamai*, which was forced to cease protection services for Krebs' website due to this attack. This attack was massive in terms of volume with respect to data transfer rate, transferring around 665 Gbps (Gigabits per second) [3]. One month later, *Dyn DNS*, a popular DNS service provider, fell victim to a similar attack. Many of their servers were brought offline by a DDoS attack, with roughly twice of the data transmission rate in the attack on Krebs, and thus caused many websites and services to become unreachable for several hours, including *reddit*, *GitHub*, and *Fox News*. The botnets in these attacks primarily consisted of IoT devices such as home routers, webcams, and other Internet-connected appliances [4].

With so many devices with diverse hardware resources, capabilities, and uses becoming ubiquitous, traditional antivirus techniques are not adequate countermeasures to the invasions of modern malware. Furthermore, most existing malware detection techniques rely on identifying signatures of known malware and, as such, have limited effectiveness for detecting polymorphic malware and are not effective at preventing zero-day attacks [5]. Therefore, to secure the IoT, there must be a solution that provides behavioral malware detection, flexible mitigation strategies, and portability across diverse platforms. The aforementioned IoT attacks demonstrate the immediate need for an effective defense against malware that seek to exploit the ubiquitous model of computing provided by the IoT.

A. Contributions

In this work, we employ three semi-supervised host-based anomaly detection algorithms to detect botnet malware that compromise home routers:

1) PCA-based anomaly detector;
2) one-class support vector machines (SVM);
3) a naive anomaly detector based on unseen n-grams.

This is the first rigorous exploration of detecting malware on end-point routers. Semi-supervised approaches are advantageous in that they only require normal data (i.e., clean data that is free of the influence of malware) for training, and are capable of detecting new attacks. First we demonstrate that using the bag-of-2-grams model, all three algorithms can achieve extraordinarily high detection rates of unseen malware without incurring any false alarm. Second we analyze the distributions of system call sequences of traces infected by various malware, with a comparison to the distribution of the clean traces. The rest of this paper is organized as follows. §II introduces related work. §III presents the collection method of system call traces for malware detection. §IV overviews the botnet malware considered in this work. §V introduces the three anomaly detection algorithms, and also analyzes the statistics of system call sequences from hosts infected by malware. §VI presents the experimental results. §VII concludes the paper.

II. Related work

There is a group of supervised behavioral analysis approaches, which require labeled data from both normal and infected hosts for training, to perform host-based intrusion detection [6], [7], [8]. Canzanese *et al.* [7] use the bag-of-n-gram model to process system call traces and then apply several supervised classifiers to detect malware on Microsoft Windows hosts. Moskovitch *et al.* [6] describe how to extract various features collected by the Windows performance counter and VTrace, and employ decision trees, and a Naive Bayes classifier, to detect worms. Mehdi *et al.* [8] propose hyper-gram, which is a novel feature extraction method, for malware detection on Linux virtual machines (VMs).

Several unsupervised behavioral analysis based detection approaches can be found in [9], [10], [11]. Kim *et al.* employ a long short-term memory language model of system call representations and then use an ensemble method that combines several simple thresholding classifiers to form a single stronger classifier [10]. Besides the system calls, some previous work also makes use of the system call arguments to detect malware [12], [9]. These techniques are all more resource intensive than the three simpler techniques we consider in this paper.

III. Data preparation

System call trace logs were collected from a virtualized home router through which real but anonymized Internet traffic was replayed. To ensure that the system call data used in the experiment reflected actual home router use, the test bed consists of real home router firmware running on VMs. System call trace logs were taken from a program built into the kernel of the router operating system (OS). The program was designed to interfere minimally with the operation of the device and the sequence of system calls generated during operation.

Userspace processes make *system calls* to request services from the OS's kernel. Different processor architectures have different sets of system calls and the number of system calls varies from one architecture to another. On Unix-like systems, implementations of the C library provide an *Application Programming Interface (API)* including wrapper functions for system calls. The *Application Binary Interface (ABI)* for a system defines how an application should make system calls and the system call numbers for its architecture. The compiler is responsible for adhering to a system's ABI. Thus, if a program's source code is changed or obfuscated (e.g., to avoid being detected by signature-based detection methods), the sequence of system calls it makes will be similar to the sequence of calls it made prior to obfuscation assuming it provides the same functionality.

The tracing framework `ftrace` provides the ability to record and monitor information about system calls as they occur on a system. `ftrace` is also highly configurable to exclude specific calls or events [13]. As such, `ftrace` can be used on systems built upon the Linux kernel by enabling the option `CONFIG_FUNCTION_TRACER` in the kernel configuration options. Once enabled, a new file system `debugfs` is mounted at `/sys/kernel/debug/`. The `tracing` directory resides in this file system and contains the configuration files for `ftrace` as well as the files in which tracing data are logged. For this work, the *syscall-sensor* utility from the *Heimdall* project [14] was used to collect trace data used by `ftrace`. This utility allows users to select the output format and to decide which fields from the records provided by `ftrace` to record.

The sequence of system calls recorded during data collection comprises those made by every process running on the device (with the exception of the sensor recording the calls and its subprocesses) in the order in which the system calls occur. The feature space of this experiment considers the full set of ARM architecture system calls should they appear in the sequence observed during the operation of the system under simulation.

Sample Internet traffic was obtained from the *MAWI* Working Group of the *WIDE* Project [15]. *MAWI* provides daily samples of anonymized Internet traffic (in the form of pcap files) captured from the transit link of *WIDE* to the upstream ISP. These samples typically contain hundreds of millions of packets sent between millions of servers and hosts. As such, these samples cannot be used directly as sample traffic to emulate typical home router use. However, they contain traffic from end users that can be filtered from the samples and used to represent traffic generated by an individual user.

In order to find suitable host traffic, the pcap files were searched for packets representing Internet browsing by considering only traffic sent to or from TCP ports 80 or 443 (the port numbers for HTTP and HTTPS, respectively). The IP addresses at the opposite end of traffic from these IP addresses represent hosts and if the host appears consistently throughout the entire sample, all traffic involving that host's IP is pulled from the full sample and written to a new pcap file containing only that host's traffic. Traffic from 9 unique hosts over 15 minutes were isolated into their own pcap files. To generate many unique traffic scenarios that are representative of traffic by hosts on a home network, the pcap files of each unique combination of up to 4 of these hosts were merged, resulting in $\sum_{k=1}^{4}\binom{9}{k} = 255$ unique samples.

IV. Malware

This work considers two families of malware that compromise embedded Linux-based systems in order to conduct DDoS attacks: MrBlack and Mirai. Both of them target vulnerable IoT devices that use weak security credentials [4]. This work uses one variant of MrBlack [1] and three variants of Mirai (enumerated as Mirai-v1 [2], Mirai-v2 [3], and Mirai-v3 [4]) for evaluating the effectiveness of the malware detection methods.

The older of the two malware families, MrBlack, was first observed in 2014 and infected devices located across 109 countries. The botnet was notable because it largely consisted of small office/home routers, most of which were ARM-based devices. MrBlack exploits lax security practices employed by the users of these devices and spreads by scanning the Internet for devices accepting remote connections over SSH or HTTP and attempting to authenticate with them using default credentials [16].

Mirai is the botnet malware that was responsible for the DDoS attack targeting the DNS provider *Dyn* in 2016 introduced in §I. Mirai created a large network of compromised devices by scanning the Internet for unsecured IoT devices and routers and attempting to authenticate with them using their default credentials [4]. Since Mirai's source code was publicly released [17], at least 493,000 devices have been infected by Mirai [18].

Executables for MrBlack and Mirai were obtained from *VirusShare* [19]. All of the sample malware executables are 32-bit ELF binaries compiled for ARM architecture.

V. Malware detection algorithms

This section first introduces two data preprocessing techniques employed on the raw system call traces before applying the anomaly detection algorithms, and then gives an overview of the three anomaly detection algorithms for detecting malware on home routers. This section also gives an analysis of the 2-gram distributions of infected system-call traces.

A. Data preprocessing

This section presents how we preprocess the raw system call traces before anomaly detection. §V-A1 introduces the bag-of-n-grams model and §V-A2 presents a transformation method to properly scale the features.

1) Bag-of-n-grams: Each raw system call trace is a sequence of system calls, and every system call can be uniquely identified by a *system call number*, which is an integer. As an essential representation in natural language processing (NLP), the bag-of-n-grams model can be used to represent a system call trace as a sum of the one-hot vectors of n-grams appearing in the trace [7]. A n-gram is a sequence of system call numbers appearing in a small window of length n in a trace. Suppose there are p distinct types of n-gram that are taken into consideration, a one-hot

vector of a n-gram is a length p vector with all entries set to zero except for a single entry set to one, which uniquely identifies this n-gram. Therefore, the bag-of-n-grams model represents each system call trace as a vector $X \in \mathbb{R}^p$, each entry of which is the number of occurrences of the corresponding n-gram.

2) TF-IDF transformation: TF-IDF is a widely-used transformation method in NLP. Built under the assumption that terms occurring more frequently in a single document, but less frequently across different documents, are more meaningful. The TF-IDF transformation puts larger weights on n-grams that occur frequently in a single trace but occur in just a few traces. Consider we have a bag-of-n-gram matrix $\mathbf{X} \in \mathbb{R}^{N \times p}$, each row of which represents a trace sample. The TF-IDF transformation is the product of the term frequency (TF) with the inverse document frequency (IDF) [7]:

$$\text{TF-IDF}(i, j) = \text{TF}(i, j) \times \text{IDF}(j), \qquad (1)$$

[1] MD5: 3a7ebd108a645d5d40abff6a05b67b82

[2] MD5: 0c34afe208edc82ba5dcdf0be14b0133

[3] MD5: 2ef11d7dbc34c0ace06197a9f8224c21

[4] MD5: 081262b472c629a1739f328f8b1dd15c

for i the trace index and j the n-gram index,

$$\mathrm{TF}(i,j) = \begin{cases} 1 + \ln \mathbf{X}_{i,j}, & \text{if } \mathbf{X}_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases},$$
$$\mathrm{IDF}(j) = \ln \frac{1+N}{1+\mathrm{DF}(j)} + 1,$$
$$\mathrm{DF}(j) = \left|\{i' : \mathbf{X}_{i',j} > 0\}\right|,$$

$\mathbf{X}_{i,j}$ denotes the (i,j)-th entry of $\mathbf{X}$, and $|\cdot|$ denotes the cardinality of a set. Each sample vector after the TF-IDF transformation is normalized to have the unit l_2 norm.

B. Approach # 1: PCA-based anomaly detection

Principal component analysis (PCA) based statistical anomaly detection (SAD) is widely used in detecting network traffic anomalies in backbone networks [20]. The main idea is to identify a low-dimensional subspace that captures most of the variance in a traffic matrix.

Assume that $\mathbf{X}^{N\times p}$ is a data matrix obtained after the preprocessing introduced in §V-A and mean subtraction, so that the mean of each column of $\mathbf{X}$ is zero. PCA-SAD finds the *best* k-dimensional subspace ($k \ll p$), i.e., the principal subspace, in the sense that this subspace can minimize the projection error in the Frobenius norm [21]. It can be proved that the basis vectors of the principal subspace are the eigenvectors corresponding to the several leading eigenvalues of the sample covariance matrix of $\mathbf{X}$. First, compute the sample covariance matrix of $\mathbf{X}$: $\mathbf{S} \equiv \frac{1}{N}\mathbf{X}^\intercal\mathbf{X}$. Then perform the eigen-decomposition: $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\intercal$, where $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, ..., \lambda_p\}$ is a diagonal matrix with eigenvalues as its diagonal entries sorted in decreasing order, and columns of $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p]$ are the eigenvectors corresponding to the eigenvalues. We can take the leading k eigenvectors $\mathbf{V}^{(k)} = [\mathbf{v}_1, ..., \mathbf{v}_k]$ as the basis for the principal subspace. To test if a new sample $\mathbf{x} \in \mathbb{R}^p$ (after mean subtraction) is an anomaly or not, we can compute its squared distance to the principal subspace: $Q_\mathbf{x} \equiv \mathbf{x}^\intercal(\mathbf{I} - \mathbf{V}^{(k)}\mathbf{V}^{(k)\intercal})\mathbf{x}$, and call this the *Q-statistic* of $\mathbf{x}$. PCA-SAD assumes that normal patterns mainly lie in the principal subspace, and an excessively large distance to this subspace is an indication of anomalousness [20]. Fixing a detection threshold to q, if $Q_\mathbf{x} > q$, we label $\mathbf{x}$ as anomalous; otherwise, we label it as normal.

There are many methods for choosing the dimension of principal subspaces (see [22]), we use a classic method that chooses the dimension k such that that a specific fraction β (e.g., $\beta = 0.9$) of variance is included in the principal subspace, i.e., $k = \min\left\{i \middle| (\sum_{j=1}^{i}\lambda_j)/(\sum_{j=1}^{p}\lambda_j) \geq \beta\right\}$.

C. Approach #2: One-class SVM

The one-class SVM proposed by Schölkopf [23] is a widely-used anomaly detection algorithm that has been applied to document classification, host-based intrusion detection, and other problems. Its objective is to find a hyperplane that can best separate the training set from the origin. The objective function is [23]:

$$\begin{aligned} &\min_{\mathbf{w},\boldsymbol{\xi},\rho} \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \\ &\text{subject to} \quad \mathbf{w}^\intercal\phi(\mathbf{x}_i) \geq \rho - \xi_i \quad \text{and} \quad \xi_i \geq 0, \end{aligned} \tag{2}$$

where $\mathbf{x}_i \in \mathbb{R}^p$ is a sample in the training set, $\phi(\cdot)$ is a mapping from the original p-dimensional feature space to a inner product space, $\mathbf{w}$ is the weight vector of the hyperplane in the inner product space, ξ_i's are penalty terms for error, ρ is a bias term, and $\nu \in (0,1]$ is a tunable parameter that poses an upper bound on the fraction of outliers in the training set. The decision hyperplane can be represented by $g(\mathbf{x}) \equiv \mathbf{w}^\intercal\phi(\mathbf{x}) - \rho = 0$. The convex optimization problem in (2) can be transformed to a Lagrangian dual problem:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &\text{subject to} \;\; 0 \leq \alpha_i \leq 1/(\nu N) \;\; , \quad \textstyle\sum_{i=1}^{N}\alpha_i = 1, \end{aligned} \tag{3}$$

where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\intercal\phi(\mathbf{x}_j)$ is the kernel function, and α_i's are the Lagarangian multipliers. Solving this dual problem finds a decision hyperplane: $g(\mathbf{x}) = \mathbf{w}^\intercal\phi(\mathbf{x}) - \rho = \sum_{i=1}^{N}\alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho = 0$. One-class SVM's decision function $f(\mathbf{x})$ is:

$$f(\mathbf{x}) = \begin{cases} \sum_{i=1}^{N}\alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho \geq 0, & \text{if } \mathbf{x} \text{ is normal} \\ \sum_{i=1}^{N}\alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho < 0, & \text{otherwise} \end{cases} \tag{4}$$

In this work, the linear kernel is applied, which means that $\phi(\mathbf{x}) = \mathbf{x}$. In the testing phase, the threshold 0 in $f(\mathbf{x})$ can be tuned to obtain a better tradeoff between the false alarm rate and the detection rate.

D. Approach #3: Naive anomaly detector using unseen n-grams

Due to their specialized responsibilities, the operations of IoT devices, including home routers are very limited and have quite predictable behaviors [4]. Thus, we can utilize a naive anomaly detection algorithm based on unseen n-grams. First, in the training phase, the naive anomaly detector learns a *normal dictionary*, denoted by $\mathcal{D}$, that consists of all occurring n-grams from the clean training system call traces. $\mathcal{D}$ serves as a normal operation profile. Under the assumption that $\mathcal{D}$ is trained by enough clean traces that can presumably characterize most operations of the device, an occurrence of a new n-gram that is not in $\mathcal{D}$ is an indicator of anomaly. During the detection process, for a length L trace fragment, the anomaly detector counts the number N_A of n-grams in this trace fragment that are not in $\mathcal{D}$, and uses this number N_A as the *anomaly degree*. If $N_A > \tau_A$, where τ_A is a tunable detection threshold, we label this fragment as "anomalous"; otherwise, we label it as "normal". Previous algorithms like STIDE [24] also use unknown contiguous subsequences, however the naive anomaly detector presented in this work follows a simplified procedure. Although its principle is simple, this algorithm is shown to be quite effective in detecting botnet malwares on routers in §VI-C.

E. Distributions of n-grams

This section investigates the distribution difference between n-grams of clean traces and those of infected traces. We calculate the empirical probability mass function (PMF) of n-grams of the clean traces, the traces infected by MrBlack, and the traces infected by 3 variants of Mirai, with $n = 2$. Fig. 1 shows the distributions of 2-grams of clean traces and infected traces. The 2-grams are ranked in decreasing order of their frequencies in the clean traces, and all the x-axes of plots in Fig. 1 are the same 2-gram ranking indices.

We use the Jensen-Shannon divergence (JSD) [25] to quantify the difference of the distribution of n-gram in infected traces and the distribution of n-gram in clean traces. Being derived from the Kullback-Leibler divergence (KLD), the JSD is a common distance metric of probability distributions. JSD has advantages over KLD on that it is symmetric and is bounded in $[0, 1]$. The JSD between empirical PMFs $\mathbf{p}$ and $\mathbf{q}$ is:

$$\mathrm{JSD}(\mathbf{p}, \mathbf{q}) \equiv \frac{1}{2}\mathrm{KLD}(\mathbf{p}||\mathbf{z}) + \frac{1}{2}\mathrm{KLD}(\mathbf{q}||\mathbf{z}), \qquad (5)$$

where $\mathbf{z} \equiv \frac{1}{2}(\mathbf{p} + \mathbf{q})$, and $\mathrm{KLD}(\mathbf{p}||\mathbf{z}) \equiv \sum_i \mathbf{p}(i) \ln \frac{\mathbf{p}(i)}{\mathbf{z}(i)}$, and $\mathbf{p}(i)$ denotes the i-th element of $\mathbf{p}$. Table I shows the calculated JSD's between the PMFs of infected trace and the PMF of the clean traces. From Table I we can see that the JSD of the malware Mr. Black is more than six times larger than that of Mirai variants, each of which is very similar in terms of JSD. This shows that the behavior of Mirai is more "stealthy" than that of MrBlack.

	MrBlack	Mirai-v1	Mirai-v2	Mirai-v3
JSD	0.0946	0.0093	0.0099	0.0137

TABLE I: Distribution distances of the 2-gram PMFs

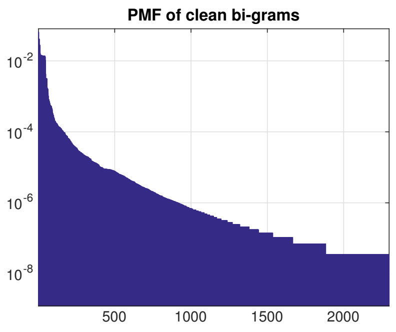
VI. Experimental results

In the experiment, only system call traces that are not infected by malware are used to train the anomaly detectors. Each system call trace contains a sequence of system calls occurring in a time window of 15 minutes. Every system call trace is partitioned into fragments, each of which has exactly L system calls (we call L the *fragment length*). There are 255 clean system call traces in total. The average number of system calls in a clean trace is $113,464$, the minimum is $46,052$, and the maximum is $127,484$.

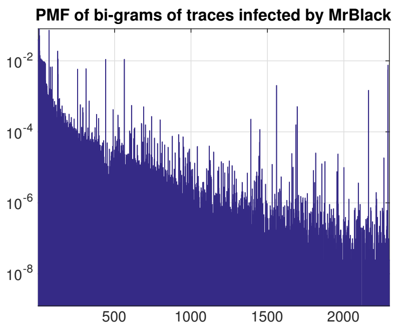
For the detection results shown in this section, all error rates are averaged over 5-fold cross-validation. We randomly divided clean traces into 5 equal-sized samples (i.e., each sample contains 51 system call traces). Each time we use 4 of the samples to train the anomaly detector, and 1 sample for testing. In the testing set, we also add traces infected by botnet malware (e.g., MrBlack, Mirai variants); for each malware, there are 51 infected traces. Then we repeat this process 5 times, and compute the average detection error rate. To test if a system call trace is anomalous or not, we adopt the following scheme: if at least one fragment of the system trace is labeled as anomalous by the anomaly detector, we label the whole system call trace as an anomalous trace; if all fragments of the system call trace are labeled as normal, we label the whole system call trace as normal. We adopt the true positive rate (TPR) and false alarm rate (FAR) as evaluation metrics. The FAR is defined as the ratio of clean traces that are labeled as "normal" over all clean traces in the testing set; the TPR is defined as the ratio of infected traces that are labeled as "abnormal" over all infected traces.

A. Detection results of PCA-SAD

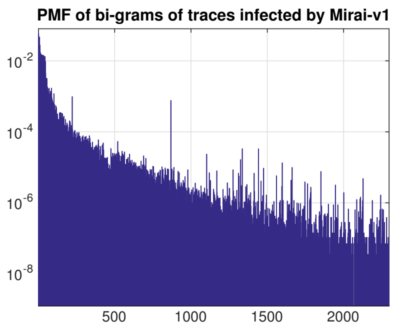
For PCA-SAD, we only consider the n-grams that appear in the training dataset. In the experiment, the principal subspace dimension k is chosen to include 90% of the variance. Fig. 6 shows a comparison of the *overall* TPR, i.e., the TPR of detecting all malware, at $\mathrm{FAR} = 0$ between PCA-SAD with TF-IDF and without TF-IDF. We can see that the detection accuracy of PCA-SAD without TF-IDF is higher than that of PCA-SAD with TF-IDF when $L \geq 3000$; while PCA-SAD with TF-IDF achieve higher detection accuracy when $L < 1000$. When $L = 6000$, PCA-SAD without TF-IDF can achieve 100% detection accuracy with zero FAR. Fig. 2 presents the ROC curves of detecting MrBlack and Mirai variants using PCA-SAD without TF-IDF transformation. From Fig. 2 we can see that for PCA-SAD without TF-IDF,



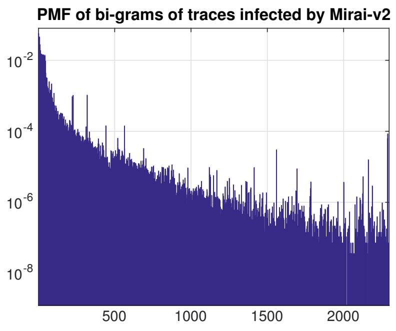
(a) Training traces



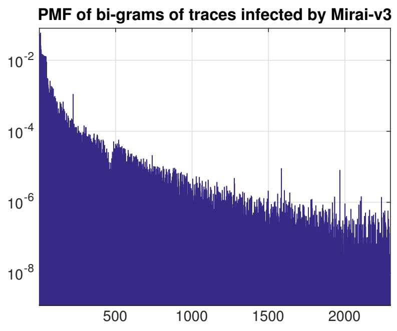
(b) Traces infected by MrBlack



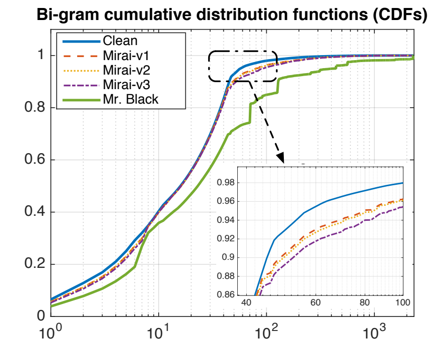
(c) Traces infected by Mirai (variant 1)



(d) Traces infected by Mirai (variant 2)



(e) Traces infected by Mirai (variant 3)



(f) Bi-gram CDFs

Fig. 1: Distributions of 2-grams

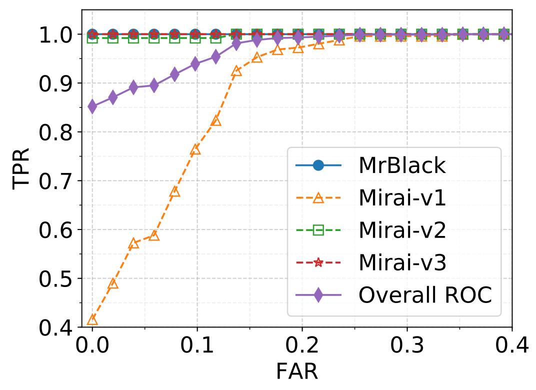
Mirai-v1 is harder to be detect than the other two Mirai variants; MrBlack is the most detectable and always has 100% detection rate with zero FAR. In order to study the impact of training size on the detection performance of PCA-SAD, we train PCA-SAD with different numbers of clean trace logs, and Fig. 3 shows the result. We can see that for large fragment lengths, $L = 5000$, and $L = 6000$, the decrease of training size degrades the PCA-SAD's performance.

B. Detection results of one-class SVM

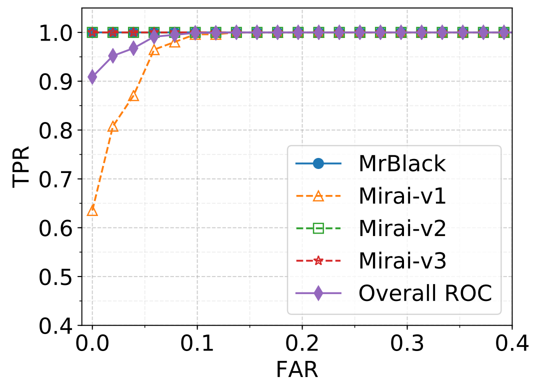
The linear kernel is used by our one-class SVM since: $i)$ it is computationally efficient compared with non-linear kernels; $ii)$ the number of features p is large (e.g., over 2000 distinct 2-grams are observed in the training set), there is little need to map it to a higher dimensional space [26]. This work only uses 2-grams, and there are two different approaches: $i)$ we use the features learned from the training set (i.e., we only consider 2-grams that occur in training phase), and call this the partial 2-gram space; $ii)$ we use all possible 2-grams as features, i.e., the complete 2-gram space. The feature size is $p = 379 \times 379 = 143,641$ for the second approach. We set the parameter $\nu = 0.001$, which imposes an upper bound on the ratio of training points that are classified as outliers. We can see from Fig. 6 that the one-class SVM using the complete 2-gram space achieves better performance than the one only using 2-grams in the training set. Fig. 4 shows the ROC curve of the one-class SVM using the partial 2-gram space when $L = 3000$; the one-class SVM using the complete 2-gram space has an almost perfect ROC curve and thus it is omitted here.

C. Detection results of naive anomaly detector using unseen n-grams

The ROC curves of the naive anomaly detector are shown in Fig. 5 using 2-grams. The ROC curves are created by sweeping the anomaly degree threshold τ_A (i.e., threshold for the number of unseen n-grams). For $L = 250$ with $\tau_A = 3$, and $L = 1000$ with $\tau_A = 5$, the anomaly detector can achieve an overall TPR 99.80% with FAR $= 0$. As Fig. 6 shows, the performance of the naive anomaly detector is not very sensitive to the change of fragment length L. However, one drawback of the naive anomaly detector is that it is not easy for it to choose an appropriate detection threshold τ_A that can have a large detection rate but can also guarantee a low FAR, since the anomaly degree N_A (i.e., the number of unseen n-grams in training) is completely unobservable and unpredictable in the training set.



(a) $L = 1000$



(b) $L = 3000$

Fig. 2: ROC curves of PCA-SAD using 2-grams (without TF-IDF)

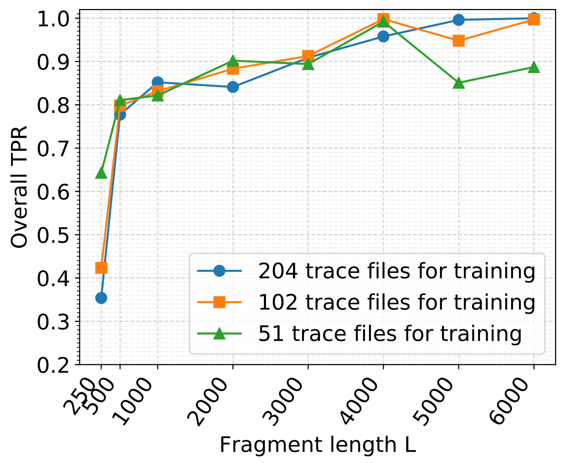


Fig. 3: Overall TPR of PCA-SAD with various number of training traces

Fig. 6 compares the detection accuracy of PCA-SAD, the naive anomaly detector using unseen n-grams, and one-class SVM using 2-grams. In Fig. 6, the y-axis is the maximum overall TPR corresponding to FAR $= 0$ averaged over 5-fold cross-validation. Fig. 6 shows that

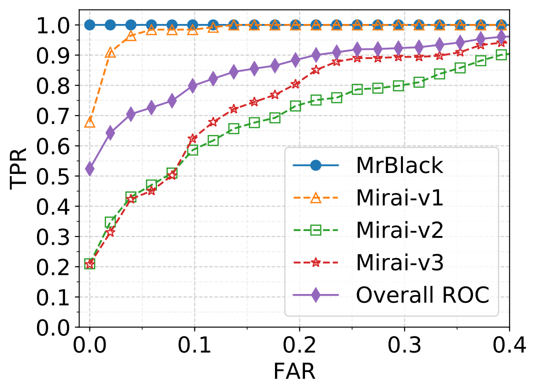


Fig. 4: ROC curves of the one-class SVM using 2-grams learnt from the training set ($L = 3000$)

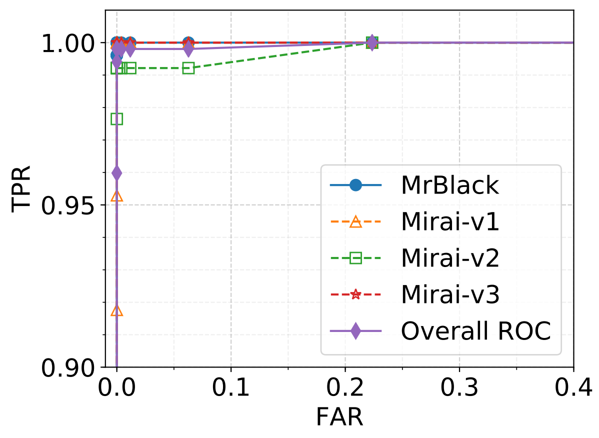


Fig. 5: ROC curves of the naive anomaly detector using unseen 2-grams ($L = 1000$)

PCA-SAD and one-class SVM with partial 2-gram space are sensitive to the change of fragment length L, and generally their detection accuracy increases as L grows large. However, L has trivial impact on the one-SVM using the complete 2-gram space and the naive anomaly detector. In general, the one-SVM using complete 2-gram space and the naive anomaly detector outperform the other anomaly detectors regardless of the choice of L. However, if $L = 6000$, PCA-SAD without TF-IDF can achieve the same TPR as the one-SVM with complete 2-gram space and the naive anomaly detector. One commonality of the one-SVM using the complete 2-gram space and the naive anomaly detector is that they both use information of 2-grams that are unseen in the training set, while the others just completely ignore 2-gram features that are not in the training set. The superiority of these two algorithms demonstrates the importance of the new n-grams caused by the malware's operation for anomaly detection.

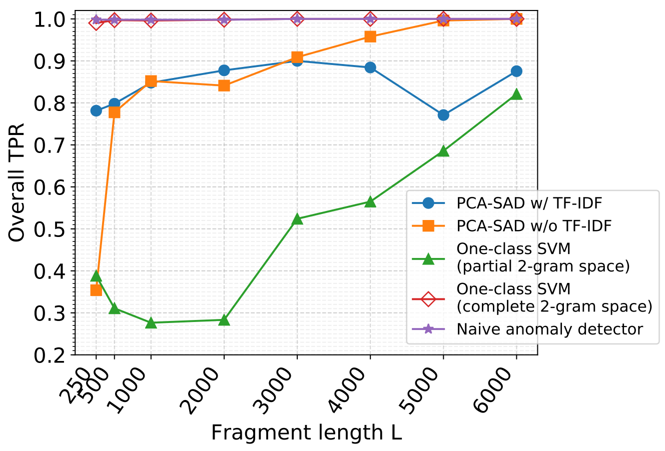


Fig. 6: Comparision of anomaly detectors' maximum overall TPR corresponding to FAR $= 0$ with 2-grams and various fragment length L

VII. Conclusion

This work demonstrates that system call based behavior analysis with semi-supervised anomaly detection algorithms can effectively detect previously unknown malwares on home routers, which is a special and essential component to the IoT, with high accuracy and low or no false alarms. Because the functionality and operation of IoT devices like routers are very specialized and regular, it is easy for anomaly detection algorithms to learn the normal patterns of home routers, and malicious programs can be detected if their operations cause a significant deviation from the devices' normal patterns of execution. Experimental results show that the one-class SVM using the complete 2-gram space, and the naive anomaly classifier based on unseen n-grams outperform the PCA-based anomaly detector when the fragment length is relatively short; while all three anomaly detectors achieve a 100% detection rate and close to zero false alarms when the fragment length is sufficiently large.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] (2016, Sep) IoT devices being increasingly used for DDoS attacks. [Online]. Available: http://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-attacks

[3] S. Mansfield-Devine, "DDoS goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare," *Network Security*, vol. 2016, no. 11, pp. 7–13, 2016.

[4] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb 2017.

[5] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1520–1549, Third 2014.

[6] R. Moskovitch, Y. Elovici, and L. Rokach, "Detection of unknown computer worms based on behavioral classification of the host," *Computational Statistics & Data Analysis*, vol. 52, no. 9, pp. 4544–4566, 2008.

[7] R. Canzanese, S. Mancoridis, and M. Kam, "Run-time classification of malicious processes using system call analysis," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, Oct 2015, pp. 21–28.

[8] B. Mehdi, F. Ahmed, S. A. Khayyam, and M. Farooq, "Towards a theory of generalizing system call representation for in-execution malware detection," in *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1–5.

[9] F. Maggi, M. Matteucci, and S. Zanero, "Detecting intrusions through system call sequence and argument analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 381–395, Oct 2010.

[10] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," *arXiv preprint arXiv:1611.01726*, 2016.

[11] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and M. Couture, "A host-based anomaly detection approach by representing system calls as states of kernel modules," in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, Nov 2013, pp. 431–440.

[12] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, "Anomalous system call detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 61–93, 2006.

[13] R. Steven. (2008) ftrace - function tracer. [Online; accessed 2-July-2017]. [Online]. Available: https://www.kernel.org/doc/Documentation/trace/ftrace.txt

[14] A. Duff. (2015) Heimdall. [Online]. Available: https://github.com/ronin-zero/heimdall

[15] MAWI data. [Online]. Available: http://mawi.wide.ad.jp/mawi/

[16] O. Gayer, R. Atias, and I. Zeifman. (2015, May) Lax security opens the door for mass-scale abuse of SOHO routers. [Online]. Available: https://www.incapsula.com/blog/ddos-botnet-soho-router.html

[17] (2016, October) Mirai-source-code. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code

[18] M. Kan, "Hackers create more IoT botnets with mirai source code," *ITworld*, October 2016.

[19] Virusshare.com. [Online]. Available: https://virusshare.com/

[20] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 217–228, 2005.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] G. Diana and C. Tommasi, "Cross-validation methods in principal component analysis: a comparison," *Statistical Methods & Applications*, vol. 11, no. 1, pp. 71–82, 2002.

[23] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[24] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in *Proceedings 1996 IEEE Symposium on Security and Privacy*, May 1996, pp. 120–128.

[25] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[26] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," National Taiwan University, Tech. Rep., 2003.