Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants

Karel Bartos
Cisco Systems, Inc.
Czech Technical University in Prague, Faculty of Electrical Engineering

Michal Sofka
Cisco Systems, Inc.
Czech Technical University in Prague, Faculty of Electrical Engineering

Vojtech Franc
Czech Technical University in Prague, Faculty of Electrical Engineering

Abstract

New and unseen polymorphic malware, zero-day attacks, or other types of advanced persistent threats are usually not detected by signature-based security devices, firewalls, or anti-viruses. *This represents a challenge to the network security industry as the amount and variability of incidents has been increasing.* Consequently, this complicates the design of learning-based detection systems relying on features extracted from network data. The problem is caused by different joint distribution of observation (features) and labels in the training and testing data sets. This paper proposes a classification system designed to detect both known as well as previously-unseen security threats. The classifiers use statistical feature representation computed from the network traffic and learn to recognize malicious behavior. The representation is designed and optimized to be invariant to the most common changes of malware behaviors. This is achieved in part by a feature histogram constructed for each group of HTTP flows (proxy log records) of a user visiting a particular hostname and in part by a feature self-similarity matrix computed for each group. The parameters of the representation (histogram bins) are optimized and learned based on the training samples along with the classifiers. The proposed classification system was deployed on large corporate networks, where it detected 2,090 new and unseen variants of malware samples with 90% precision (9 of 10 alerts were malicious), which is a considerable improvement when compared to the current flow-based approaches or existing signature-based web security devices.

1 Introduction

Current network security devices classify large amounts of the malicious network traffic and report the results in many individually-identified incidents, some of which are false alerts. On the other hand, a lot of malicious traffic remains undetected due to the increasing variability of malware attacks. As a result, security analysts might miss severe complex attacks because the incidents are not correctly prioritized or reported.

The network traffic can be classified at different levels of detail. Approaches based on packet inspection and signature matching [15] rely on a database of known malware samples. These techniques are able to achieve results with high precision (low number of false alerts), but their detection ability is limited only to the known samples and patterns included in the database (limited recall). Moreover, due to the continuous improvements of network bandwidth, analyzing individual packets is becoming intractable on high-speed network links. It is more efficient to classify network traffic based on *flows* representing groups of packets (e.g. NetFlow [1] or proxy logs [26]). While this approach has typically lower precision, it uses statistical modeling and behavioral analysis [8] to find new and previously unseen malicious threats (higher recall).

Statistical features calculated from flows can be used for unsupervised anomaly detection, or in supervised classification to train data-driven classifiers of malicious traffic. While the former approach is typically used to detect new threats, it suffers from lower precision which limits its practical usefulness due to large amount of false alerts. Data-driven classifiers trained on known malicious samples achieve better efficacy results, but the results are directly dependent on the samples used in the training. Once a malware changes the behavior, the system needs to be retrained. With continuously rising number of malware variants, this becomes a major bottleneck in modern malware detection systems. Therefore, the robustness and invariance of features extracted from raw data plays the key role when classifying new malware.

The problem of changing malware behavior can be formalized by recognizing that a joint distribution of the malware samples (or features) differs for already known training (source) and yet unseen testing (target) data.

This can happen as a result of target evolving after the initial classifier or detector has been trained. In supervised learning, this problem is solved by domain adaptation. Under the assumption that the source and target distributions do not change arbitrarily, the goal of the domain adaptation is to leverage the knowledge in the source domain and transfer it to the target domain. In this work, we focus on the case where the conditional distribution of the observation given labels is different, also called a conditional shift.

The domain adaptation (or knowledge transfer) can be achieved by adapting the detector using importance weighting such that training instances from the source distribution match the target distribution [37]. Another approach is to transform the training instances to the domain of the testing data or to create a new data representation with the same joint distribution of observation and labels [4]. The challenging part is to design a meaningful transformation that transfers the knowledge from the source domain and improves the robustness of the detector on the target domain.

In this paper, we present a new optimized invariant representation of network traffic data that enables domain adaptation under conditional shift. The representation is computed for bags of samples, each of which consists of features computed from network traffic logs. The bags are constructed for each user and contain all network communication with a particular hostname/domain. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. This is achieved by combining bag histograms with an invariant self similarity matrix for each bag. All parameters of the representation are learned automatically for the training data using the proposed optimization approach.

The proposed invariant representation is applied to detect malicious HTTP traffic. We will show that the classifier trained on malware samples from one category can successfully detect new samples from a different category. This way, the knowledge of the malware behavior is correctly transferred to the new domain. Compared to the baseline flow-based representation or widely-used security device, the proposed approach shows considerable improvements and correctly classifies new types of network threats that were not part of the training data.

This paper has the following major contributions:

- **Classifying new malware categories** – we propose a supervised method that is able to detect new types of malware categories from a limited amount of training samples. Unlike classifying each category separately, which limits the robustness, we propose an invariant training from malware samples of multiple categories.
- **Bag representation of samples** – Instead of classifying flows individually, we propose to group flows into bags, where each bag contains flows that are related to each other (e.g. having the same user and target domain). Even though the concept of grouping flows together has been already introduced in the previously published work (e.g. in [32]), these approaches rely on a sequence of flow-based features rather than on more complex representation.
- **Features describing the dynamics of the samples** – To enforce the invariant properties of the representation, we propose to use a novel approach, where the features are derived from the self-similarity of flows within a bag. These features describe the dynamics of each bag and have many invariant properties that are useful when finding new malware variants and categories.
- **Learning the representation from the training data** – To optimize the parameters of the representation, we propose a novel method that combines the process of learning the representation with the process of learning the classifier. The resulting representation ensures easier separation of malicious and legitimate communication and at the same time controls the complexity of the classifier.
- **Large scale evaluation** – We evaluated the proposed representation on real network traffic of multiple companies. Unlike most of the previously published work, we performed the evaluation on highly imbalanced datasets as they appear in practice (considering the number of malicious samples), with most of the traffic being legitimate, to show the potential of the approach in practice. This makes the classification problem much harder. We provided a comparison with state-of-the-art approaches and a widely-used signature-based web security device to show the advantages of the proposed approach.

2 Related Work

Network perimeter can be secured by a large variety of network security devices and mechanisms, such as host-based or network-based Intrusion Detection Systems (IDS) [36]. We briefly review both systems, focusing our discussion on network-based IDS, which are the most relevant to the presented work.

Host-based IDS systems analyze malicious code and processes and system calls related to OS information. Traditional and widely-used anti-virus software or spyware scanners can be easily evaded by simple transformations of malware code. To address this weakness, methods of static analysis [30], [38] were proposed.

Static analysis, relying on semantic signatures, concentrates on pure investigation of code snippets without actually executing them. These methods are more resilient to changes in malware codes, however they can be easily evaded by obfuscation techniques. Methods of dynamic analysis [29], [34], [42] were proposed to deal with the weaknesses of static analysis, focusing on obtaining reliable information on execution of malicious programs. The downside of the dynamic analysis is the necessity to run the codes in a restricted environment which may influence malware behavior or difficulty of the analysis and tracing the problem back to the exact code location. Recently, a combination of static and dynamic analysis was used to analyze malicious browser extensions [20].

Network-based IDS systems are typically deployed on the key points of the network infrastructure and monitor incoming and outgoing network traffic by using static signature matching [15] or dynamic anomaly detection methods [8]. Signature-based IDS systems evaluate each network connection according to the predefined malware signatures regardless of the context. They are capable of detecting well-known attacks, but with limited amount of detected novel intrusions. On the other hand, anomaly-based IDS systems are designed to detect wide range of network anomalies including yet undiscovered attacks, but at the expense of higher false alarm rates [8].

Network-based approaches are designed to detect malicious communication by processing network packets or logs. An overview of the existing state-of-the-art approaches is shown in Table 1. The focus has been on the traffic classification from packet traces [5], [28], [39], [41], as this source provides detailed information about the underlying network communication. Due to the still increasing demands for larger bandwidth, analyzing individual packets is becoming intractable on high-speed network links. Moreover, some environments with highly confidential data transfers such as banks or government organizations do not allow deployment of packet inspection devices due to the legal or privacy reasons. The alternative approach is the classification based on network traffic logs, e.g. NetFlow [1], DNS records, or proxy logs. The logs are extracted at the transport layer and contain information only from packet headers.

Methods introduced in [12] and [23] apply features extracted from NetFlow data to classify network traffic into general classes, such as P2P, IMAP, FTP, POP3, DNS, IRC, etc. A comparison and evaluation of these approaches can be found in a comprehensive survey [24]. A combination of host-based statistics with SNORT rules to detect botnets was introduced in [16]. The authors showed that it is possible to detect malicious traffic using statistical features computed from NetFlow data, which motivated further research in this field. An alternative approach for classification of botnets from NetFlow features was proposed in [6]. The authors of [33] have used normalized NetFlow features to cluster flow-based samples of network traffic into four predefined categories. As opposed to our approach, the normalization was performed to be able to compare individual features with each other. In our approach, we extended this idea and use normalization to be able to compare various malware categories. While all these approaches represent relevant state-of-the-art, network threats evolve so rapidly that these methods are becoming less effective due to the choice of features and the way they are used.

One of the largest changes in the network security landscape is the fact that HTTP(S) traffic is being used not only for web browsing, but also for other types of services and applications (TOR, multimedia streaming, remote desktop) including lots of malicious attacks. According to recent analysis [18], majority of malware samples communicate via HTTP. This change has drawn more attention to classifying malware from web traffic. In [25], the authors proposed an anomaly detection system composed of several techniques to detect attacks against web servers. They divide URIs into groups, where each group contains URIs with the same resource path. URIs without a query string or with return code outside of interval [200, 300] are considered as irrelevant. The system showed the ability to detect unseen malware samples and the recall will be compared with our proposed approach in Section 8. In [40], the authors introduced a method for predicting compromised websites using features extracted from page content and Alexa Web Information Service.

Having sufficient amount of labeled malware samples at disposal, numerous approaches proposed supervised learning methods to achieve better efficacy. Clasifying DGA malware from DNS records based on connections to non-existent domains (NXDomains) was proposed in [2]. Even though several other data sources were used to detect malware (such as malware executions [3] or JavaScript analysis [22]), the most relevant work to our approach uses proxy logs [9], [17], [27], [44], [32].

In all these methods, proxy log features are extracted from real legitimate and malicious samples to train a data-driven classifier, which is used to find new malicious samples from the testing set. There are five core differences between these approaches and our approach: (1) we do not classify individual flows (in our case proxy log records), but sets of related flows called bags, (2) we propose a novel representation based on features describing the dynamics of each bag, (3) the features are computed from the bags and are invariant against various changes an attacker could implement to evade detection, (4) parameters of the proposed representation are learned automatically from the input data to maximize the detection performance, (5) the proposed classification system

Approach	Type	Method	Features	Target class	Testing Data			Malicious samples	Mal:All ratio
					Type	Year	All samples		
Wang [41]	U	anomaly detection	packet payload	worms, exploits	packets	2003	531,117	N/A	N/A
Kruegel [25]	U	anomaly detection	URL query parameters	web malware	proxy logs	2003	1,212,197	11	1:100k
Gu [16]	U	clustering	host statistics+SNORT	botnet	NetFlow	2007	100,000k	5,842k	1:17
Bilge [6]	S	random forest	flow size, time	botnets	NefFlow	2011	78,000,000	36	1:2.2M
Antonakakis [2]	S	multiple	NXDomains	dga malware	DNS data	2011	360,700	8008	1:45
Bailey [3]	S	hierarch. clustering	state changes	malware	executions	2007	4,591	4,591	1:1
Kapravelos [22]	S	similarity of trees	abstract syntax tree	web malware	JavaScript	2012	20,918,798	186,032	1:112
Choi [9]	S	SVM + RAkEL	URL lexical, host, dns	malicious flows	proxy logs	2009	72,000	32,000	1:2
Zhao [44]	S	active learning	URL lexical + host	malicious flows	proxy logs	2009	1,000,000	10,000	1:100
Huang [17]	S	SVM	URL lexical	phishing	proxy logs	2011	12,193	10,094	1:1
Ma [27]	S	multiple	URL lexical + host	malicious flows	proxy logs	2011	2,000,000	6,000	1:333
Invernizzi [18]	U	graph clustering	proxy log fields	mw downloads	proxy logs	2012	1,219	324	1:4
Soska [40]	S	random forests	content of web pages	infected websites	web pages	2014	386,018	49,347	1:8
Nelms [32]	S	heuristics	web paths	mw downloads	proxy logs	2014	N/A	150	N/A
Our approach	S	learned repr.+SVM	learned bag dynamics	malicious flows	proxy logs	2015	15,379,466	43,380	1:355

Table 1: Overview of the existing state-of-the-art approaches focusing on classification of malicious traffic (U = unsupervised, S = supervised). In contrast to the existing work, our approach proposes novel and optimized representation of bags, describing the dynamics of each legitimate or malicious sample. The approach is evaluated on latest real datasets with a realistic ratio of malicious and background flows (proxy log records).

was deployed on corporate networks and evaluated on imbalanced datasets (see Table 1) as they appear in practice to show the expected efficacy on these networks.

3 Formalization of the Problem

The paper deals with the problem of creating a robust representation of network communication that would be invariant against modifications an attacker can implement to evade the detection systems. The representation is used to classify network traffic into positive (malicious) or negative (legitimate) category. The labels for positive and negative samples are often very expensive to obtain. Moreover, sample distribution typically evolves in time, so the probability distribution of training data differs from the probability distribution of test data. This complicates the training of classifiers which assume that the distributions are the same. In the following, the problem is described in more detail.

Each sample is represented as an n-dimensional feature vector $x \in \mathbb{R}^n$. Samples are grouped into bags, with every bag represented as a matrix $X = (x_1, \ldots, x_m) \in \mathbb{R}^{n \times m}$, where m is the number of samples in the bag and n is the number of features. The bags may have different number of samples. A single category y_i can be assigned to each bag from the set $\mathscr{Y} = \{y_1, \ldots, y_N\}$. Only a few categories are included in the training set. The probability distribution on training and testing bags for category y_j will be denoted as $P^L(X|y_j)$ and $P^T(X|y_j)$, respectively. Moreover, the probability distribution of the training data differs from the probability distribution of the testing data, i.e. there is a domain adaptation problem [7] (also called a conditional shift [43]):

$$P^L(X|y_j) \neq P^T(X|y_j), \ \forall y_j \in \mathscr{Y}. \quad (1)$$

The purpose of the domain adaptation is to apply knowledge acquired from the training (source) domain into test (target) domain. The relation between $P^L(X|y_i)$ and $P^T(X|y_i)$ is not arbitrary, otherwise it would not be possible to transfer any knowledge. Therefore there is a transformation τ, which transforms the feature values of the bags onto a representation, in which $P^L(\tau(X)|y_i) \approx P^T(\tau(X)|y_i)$. The goal is to find this representation, allowing to classify individual bag represented as X into categories $\mathscr{Y} = \{y_1, \ldots, y_N\}$ under the above mentioned conditional shift.

Numerous methods for transfer learning have been proposed (since the traditional machine learning methods cannot be used effectively in this case), including kernel mean matching [14], kernel learning approaches [11], maximum mean discrepancy [19], or boosting [10]. These methods try to solve a general data transfer with relaxed conditions on the similarity of the distributions during the transfer. The downside of these methods is the necessity to specify the target loss function and availability of large amount of labeled data.

This paper proposes an effective invariant representation that solves the classification problem with a covariate shift (see Equation 1). Once the data are transformed, the new feature values do not rely on the original distribution and they are not influenced by the shift. The parameters of the representation are learned automatically from the data together with the classifier as a joint optimization process. The advantage of this approach is that the parameters are optimally chosen during training to achieve the best classification efficacy for the given classifier, data, and representation.

4 Invariant Representation

The problem of domain adaptation outlined in the previous section is addressed by the proposed representation of bags. The new representation is calculated with a transformation that consists of three steps to ensure that the new representation will be invariant under scaling and shifting of the feature values and under permutation and size changes of the bags.

4.1 Scale Invariance

As stated in Section 3, the probability distribution of bags from the training set can be different from the test set. In the first step, the representation of bags is transformed to be invariant under scaling of the feature values. The traditional representation X of a bag that consists of a set of m samples $\{x_1,\ldots,x_m\}$ can be written in a form of a matrix:

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ & & \vdots & \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix}, \qquad (2)$$

where x_{lk} denotes k-th feature value of l-th sample. This form of representation of samples and bags is widely used in the research community, as it is straightforward to use and easy to compute. It is a reasonable choice in many applications with a negligible shift in the source and target probability distributions. However, in the network security domain, the dynamics of the network environment causes changes in the feature values and the shift becomes more prominent. As a result, the performance of the classification algorithms using the traditional representation is decreased.

In the first step, the representation is improved by making the matrix X to be invariant under scaling of the feature values. **Scale invariance** guarantees that even if some original feature values of all samples in a bag are multiplied by a common factor, the values in the new representation remain unchanged. To guarantee the scale invariance, the matrix X is scaled locally onto the interval $[0,1]$ as follows:

$$\tilde{X} = \begin{pmatrix} \tilde{x}_{11} & \ldots & \tilde{x}_{1n} \\ & \vdots & \\ \tilde{x}_{m1} & \ldots & \tilde{x}_{mn} \end{pmatrix} \quad \tilde{x}_{lk} = \frac{x_{lk} - \min_l(x_{lk})}{\max_l(x_{lk}) - \min_l(x_{lk})} \qquad (3)$$

4.2 Shift Invariance

In the second step, the representation is transformed to be invariant against shifting. **Shift invariance** guaranties that even if some original feature values of all samples in a bag are increased/decreased by a given amount, the values in the new representation remain unchanged. Let us define a *translation invariant distance function* $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ for which the following holds: $d(u,v) = d(u+a, v+a)$.

Let x_{pk}, x_{qk} be k-th feature values of p-th and q-th sample from bag matrix X. Then the distance between these two values will be denoted as $d(x_{pk},x_{qk}) = s^k_{pq}$. The distance $d(x_{pk},x_{qk})$ is computed for pairs of k-th feature value for all sample pairs, ultimately forming a so called self-similarity matrix S^k. Self-similarity matrix is a symmetric positive semidefinite matrix, where rows and columns represent individual samples and (i,j)-th element corresponds to the distance between i-th and j-th sample. Self-similarity matrix has been already used thanks to its properties in several applications (e.g. in object recognition [21] or music recording [31]). However, only a single self-similarity matrix for each bag has been used in these approaches. This paper proposes to compute a set of similarity matrices, one for every feature. More specifically, a per-feature set of self-similarity matrices $\mathscr{S} = \{S^1, S^2, \ldots, S^n\}$ is computed for each bag, where

$$S^k = \begin{pmatrix} s^k_{11} & s^k_{12} & \ldots & s^k_{1m} \\ & & \vdots & \\ s^k_{m1} & s^k_{m2} & \ldots & s^k_{mm} \end{pmatrix}. \qquad (4)$$

The element $s^k_{pq} = d(x_{pk},x_{qk})$ is a distance between feature values x_{pk} and x_{qk} of k-th feature. This means that the bag matrix X with m samples and n features will be represented with n self-similarity matrices of size $m \times m$. The matrices are further normalized by local feature scaling described in Section 4.1 to produce a set of matrices $\tilde{\mathscr{S}}$.

The shift invariance makes the representation robust to the changes where the feature values are modified by adding or subtracting a fixed value. For example, the length of a malicious URL would change by including an additional subdirectory in the URL path. Or, the number of transfered bytes would increase when an additional data structure is included in the communication exchange.

4.3 Permutation and Size Invariance

Representing bags with scaled matrices $\{\tilde{X}\}$ and sets of locally-scaled self-similarity matrices $\{\tilde{\mathscr{S}}\}$ achieves the scale and shift invariance. **Size invariance** ensures that the representation is invariant against the size of the bag. In highly dynamic environments, the samples may occur in a variable ordering. **Permutation invariance** ensures that the representation should also be invariant against any reordering of rows and columns of the matrices. The final step of the proposed transformation is the transition from the scaled matrices $\tilde{X}$, $\tilde{\mathscr{S}}$ (introduced in Sec-

tions 4.1 and 4.2 respectively) to normalized histograms. For this purpose, we define for each bag:

$$z_k^X := \text{vector of values from } k\text{-th column of matrix } \tilde{X}$$

$$z_k^{\mathscr{S}} := \text{column-wise representation of upper triangular matrix created from matrix } \tilde{S}^k \in \tilde{\mathscr{S}}.$$

This means that $z_k^X \in \mathbb{R}^m$ is a vector created from values of k-th feature of $\tilde{X}$, while $z_k^{\mathscr{S}} \in \mathbb{R}^r, r = (m-1)\cdot\frac{m}{2}$ is a vector that consists of all values of upper triangular matrix created from matrix $\tilde{S}^k$. Since $\tilde{S}^k$ is a symmetric matrix with zeros along the main diagonal, $z_k^{\mathscr{S}}$ contains only values from upper triangular matrix of $\tilde{S}^k$.

A normalized histogram of vector $z = (z_1, \ldots, z_d) \in \mathbb{R}^d$ is a function $\phi : \mathbb{R}^d \times \mathbb{R}^{b+1} \to \mathbb{R}^b$ parametrized by edges of b bins $\theta = (\theta_0, \ldots, \theta_b) \in \mathbb{R}^{b+1}$ such that $\phi(z;\theta) = (\phi(z;\theta_0,\theta_1), \ldots, \phi(z;\theta_{b-1},\theta_b))$ where

$$\phi(z,\theta_i,\theta_{i+1}) = \frac{1}{d}\sum_{j=1}^{d} [\![z_j \in [\theta_{i-1},\theta_i)]\!]$$

is the value of the i-th bin corresponding to a portion of components of z falling to the interval $[\theta_{i-1},\theta_i)$.

Each column k of matrix $\tilde{X}$ (i.e. all bag values of k-th feature) is transformed into a histogram $\phi(z_k^X, \theta_k^X)$ with predefined number of b bins and θ_k^X bin edges. Such histograms created from the columns of matrix $\tilde{X}$ will be denoted as *feature values histograms*, because they carry information about the distribution of bag feature values. On the other hand, histogram $\phi(z_k^{\mathscr{S}}, \theta_k^S)$ created from values of self-similarity matrix $\tilde{S}^j \in \tilde{\mathscr{S}}$ will be called *feature differences histograms*, as they capture inner feature variability within bag samples.

Overall, each bag is represented as a concatenated feature map $\phi(\tilde{X};\tilde{\mathscr{S}};\theta) : \mathbb{R}^{n\times(m+r)} \to \mathbb{R}^{2\cdot n\cdot b}$ as follows:

$$\Big(\phi(z_1^X,\theta_1^X),\ldots,\phi(z_n^X,\theta_n^X),\phi(z_1^{\mathscr{S}},\theta_1^S),\ldots,\phi(z_n^{\mathscr{S}},\theta_n^S)\Big) \quad (5)$$

where n is the number of the original flow-based features, m is the number of flows in the bag, and b is the number of bins. The whole transformation from input network flows to the final feature vector is depicted in Figure 1. As you can see, two types of invariant histograms are created from values of each flow-based feature. At the end, both histograms are concatenated into the final bag representation $\phi(\tilde{X};\tilde{\mathscr{S}};\theta)$.

5 Learning Optimal Histogram Representation

The bag representation $\phi(\tilde{X};\tilde{\mathscr{S}};\theta)$ proposed in Section 4 has the invariant properties, however it heavily depends on the number of bins b and their edges θ defining the

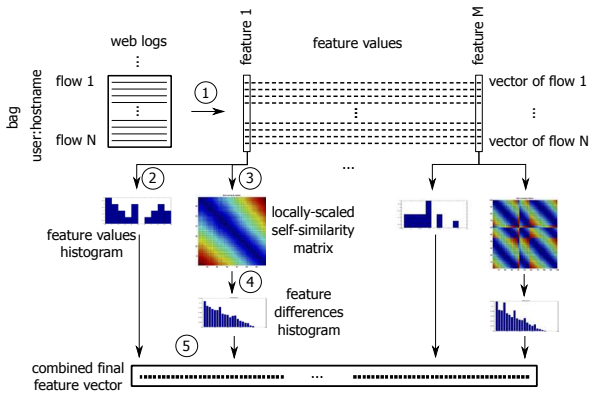

Figure 1: Graphical illustration of the individual steps that are needed to transform the bag (set of flows with the same user and hostname) into the proposed invariant representation. First, the bag is represented with a standard feature vector (1). Then feature values histograms of locally scaled feature values are computed for each feature separately (2). Next, the locally-scaled self-similarity matrix is computed for each feature (3) to capture inner differences. This matrix is then transformed into feature differences histogram (4), which is invariant on the number or the ordering of the samples within the bag. Finally, feature values and feature differences histograms of all features are concatenated into resulting feature vector.

width of the histogram bins. These parameters that were manually predefined in Section 4 C influence the classification performance. Incorrectly chosen parameters b and θ leads to suboptimal efficacy results. To define the parameters optimally, we propose a novel approach of learning these parameters automatically from the training data in such a way to maximize the classification separability between positive and negative samples.

When creating histograms in Section 4 C, the input instances are vectors z_k^X and $z_k^{\mathscr{S}}$, where $k \in \{1,\ldots,n\}$. The algorithm transforms the input instances into a concatenated histogram $\phi(\tilde{X};\tilde{\mathscr{S}};\theta)$. To keep the notation simple and concise, we will denote the input instances simply as $z = (z_1,\ldots,z_n) \in \mathbb{R}^{n\times m}$ (instead of $z = (z_1^X,\ldots,z_n^X,z_1^{\mathscr{S}},\ldots,z_n^{\mathscr{S}})$), which is a sequence of n vectors each of dimension m.

The input instance z is represented via a feature map $\phi : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\cdot b}$ defined as a concatenation of the normalized histograms of all vectors in that sequence, that is, $\phi(z;\theta) = (\phi(z_1;\theta_1),\ldots,\theta(z_n;\theta_n))$, where $\theta = (\theta_1,\ldots,\theta_n)$ denotes bin edges of all normalized histograms stacked to a single vector.

We aim at designing a classifier $h : \mathbb{R}^{n\times m} \times \mathbb{R}^{n+1} \times \mathbb{R}^{n(b+1)} \to \{-1,+1\}$ working on top of the histogram representation, that is

$$h(z;w,w_0,\theta) = \text{sign}(\langle \phi(z,w) \rangle + w_0)$$
$$= \text{sign}\left(\sum_{i=1}^{n} \sum_{j=1}^{b} \phi(z_i, \theta_{i,j-1}, \theta_{i,j}) w_{i,j} + w_0 \right) . \quad (6)$$

The classifier (6) is linear in the parameters (w, w_0) but non-linear in θ and z. We are going to show how to learn parameters (w, w_0) and implicitly also θ via a convex optimization.

Assume we are given a training set of examples $\{(z^1, y^1), \ldots, (z^m, y^m)\} \in (\mathbb{R}^{n \times m} \times \{+1, -1\})^m$. We fix the representation ϕ such that the number of bins b is sufficiently large and the bin edges θ are equally spaced. We find the weights (w, w_0) by solving

$$\min_{w \in \mathbb{R}^{b \cdot p}, w_0 \in \mathbb{R}} \left[\gamma \sum_{i=1}^{n} \sum_{j=1}^{b-1} |w_{i,j} - w_{i,j+1}| + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y^i \langle \phi(z^i; \theta), w \rangle\right\} \right] . \quad (7)$$

The objective is a sum of two convex terms. The second term is the standard hinge-loss surrogate of the training classification error. The first term is a regularization encouraging weights of neighboring bins to be similar. If it happens that j-th and $j+1$ bin of the i-the histogram have the same weight, $w_{i,j} = w_{i,j+1} = w$, then these bins can be effectively merged to a single bin because

$$w_{i,j} \phi(z_i; \theta_{i,j-1}, \theta_{i,j}) + w_{i,j+1} \phi(z_i; \theta_{i,j}, \theta_{i,j+1})$$
$$= 2w \phi(z_i; \theta_{i,j-1}, \theta_{i,j+1}) . \quad (8)$$

The trade-off constant $\gamma > 0$ can be used to control the number of merged bins. A large value of γ will result in massive merging and consequently in a small number of resulting bins. Hence the objective of the problem (7) is to minimize the training error and to simultaneously control the number of resulting bins. The number of bins influences the expressive power of the classifier and thus also the generalization of the classifier. The optimal setting of λ is found by tuning its value on a validation set.

Once the problem (7) is solved, we use the resulting weights w^* to construct a new set of bin edges θ^* such that we merge the original bins if the neighboring weights have the same sign (i.e. if $w^*_{i,j} w^*_{i,j+1} > 0$). This implies that the new bin edges θ^* are a subset of the original bin edges θ, however, their number can be significantly reduced (depending on γ) and they have different widths unlike the original bins. Having the new bins defined, we learn a new set of weights by the standard SVM algorithm

$$\min_{w \in \mathbb{R}^n, w_0 \in \mathbb{R}} \left[\frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y^i \langle \phi(z^i; \theta^*), w \rangle\right\} \right] .$$

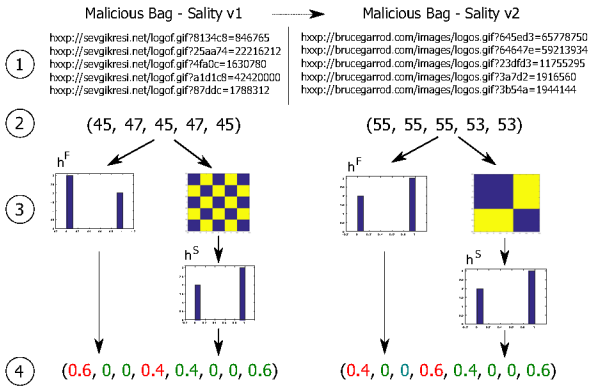

Figure 2: Illustration of the proposed representation applied on two versions of malware Sality. First, two bags of flows are created (1), one bag for each Sality sample. Next, flow-based feature vectors are created for each bag (2). For illustrative purposes, only a single feature is used - URL length. In the third step, histograms of feature values $\phi(z^X_k, \theta^X_k)$ and feature differences $\phi(z^{\mathscr{S}}_k, \theta^S_k)$ are created (3) as described in Section 4.3. Only four bins for each histogram were used. Finally, all histograms are concatenated into the final feature vector (4). Even though the malware samples are from two different versions, they have the same histogram of feature differences $\phi(z^{\mathscr{S}}_k, \theta^S_k)$. Since $\phi(z^X_k, \theta^X_k)$ is not invariant against shift, you can see that half of the values of $\phi(z^X_k, \theta^X_k)$ are different. Still, $\phi(z^X_k, \theta^X_k)$ values may play an important role when separating malware samples from other legitimate traffic.

Note that we could add the quadratic regularizer $\frac{\lambda}{2}\|w\|^2$ to the objective of (7) and learn the weights and the representation in a single stage. However, this would require tuning two regularization parameters (λ and γ) simultaneously which would be order of magnitude more expensive than tuning them separately in the two stage approach.

6 Malware Representation Example

This Section illustrates how the proposed representation (nonoptimized version) is calculated for two real-world examples of malicious behavior. Namely, two versions of a polymorphic malware Sality are compared. Sality [13] is a malware family that has become a dynamic and complex form of malicious infection. It utilizes polymorphic techniques to infect files of Widows machines. Signature-based systems or classifiers trained on a specific malware type often struggles with detecting new variants of this kind of malware. Note that most of the conclusions to the discussion that follows can be drawn for many other malware threats.

Figure 2 shows how the two Sality samples are represented with the proposed approach. First, the input flows are grouped into two bags (one bag for each Sality sample), because all flows of each bag have the same user and the same hostname (1). For the sake of simplicity, only URLs of the corresponding flows are displayed. Next, 88 flow-based feature vectors are computed for each bag (2). To simplify illustration, we use only a single feature – URL length. After this step, each Sality sample is represented with one feature vector of flow-based values. Existing approaches use these vectors as the input for the subsequent detection methods. As we will show in Section 7, these feature values are highly variable for malware categories. Classification models trained with such feature values loose generalization capability.

To enhance the robustness of the flow-based features, the proposed approach computes histograms of feature values $\phi(z_k^X, \theta_k^X)$ and feature differences $\phi(z_k^{\mathscr{S}}, \theta_k^S)$ (3) as described in Section 4.3. To make the illustration simple, only four bins for each histogram were used. Finally, all histograms are concatenated into the final feature vector (4). It can be seen that even though the malware samples are from two different versions, they have the same histogram of feature differences $\phi(z_k^{\mathscr{S}}, \theta_k^S)$. Since the histogram of feature values $\phi(z_k^X, \theta_k^X)$ is not invariant against shift, half of the values of $\phi(z_k^X, \theta_k^X)$ are different.

The number of histogram bins and their sizes are then learned from the data by the proposed algorithm (see Section 5). The proposed representation describes inner dynamics of flows from each bag, which is a robust indicator of malware samples, as we will show in the analysis of various malware families in Section 8. In contrast to the existing methods that use flow-based features or general statistics such as mean or standard deviation, the proposed representation reflects properties that are much more difficult for an attacker to evade detection.

7 Evasion Possibilities

This section discusses evasion options for an attacker when trying to evade a learning-based classification system. According to the recent work [35], the essential components for an evasion are: (1) the set of features used by the classifier, (2) the training dataset used for training, (3) the classification algorithm with its parameters. Without the knowledge of the features, the attacker is faced with major challenges and there is not any known technique for addressing them [35].

Acquire knowledge of classification algorithm with its parameters or the training data is hard if not impossible. Therefore, in the following analysis, we assume that only the features are known to the attacker. When classifying HTTP traffic from proxy logs, it is actually not difficult to create a set of common features widely used in practice. These features are the baseline flow-based features, such as those described in Table 3. When the attacker performs a mimicry attack, selected features of malicious flows are modified to mimic legitimate traffic (or flows marked as benign by the classifier).

In the following, we will analyze the case when the attacker performs a mimicry attack to evade detection by modifying flow attributes, such as URLs, bytes, and inter-arrival times. Other flow attributes can be altered in a similar way with analogical results. All modifications are divided into two groups, depending on whether the proposed representation is invariant against them.

The proposed representation is invariant to the following changes.

- **Malicious code, payload, or obfuscation** – The advantage of all network-based security approaches is that they extract features from headers of network communication rather than from the content. As a result, any changes to the payload including the usage of pluggable transports designed to bypass Deep Packet Inspection (DPI) devices will have no effect on the features. Some pluggable transports (e.g. ScrambleSuit) are able to change its network fingerprint (packet length distribution, number of bytes, inter-arrival times, etc.). Since the proposed representation mainly relies on the dynamics of URLs of flows in the bag, such changes will not negatively impact the efficacy, which is a great advantage against DPI devices.

- **Server or hostname** – The representation operates at the level of bags, where each bag is a set of flows with the same user and hostname/domain. If an attacker changes an IP address or a hostname of the remote server (because the current one has been blacklisted), the representation will create a new bag with similar feature values as in the previous bag with the original IP address or hostname, which is a great advantage against feeds and blacklists that need to be updated daily and are always behind.

- **URL path or filename** – Straightforward and easy way of evading existing classifiers using flow-based features or URL patterns is the change in path or filename from sample to sample. Since the variability of these features remains constant within each bag, these changes will also have no effect on the proposed representation.

- **Number of URL parameters, their names or values** – This is an alternative to URL path changes.

- **Encoded URL content** – Hiding information in the URL string represents another way to exfiltrate sensitive data. When the URL is encrypted and encoded (e.g. with base64), it changes the URL length

and may globally influence other features as well. As the proposed representation is invariant against shifting, changing the URL length will not change the histograms of feature differences.

- **Number of flows** – Another option for an attacker to hide in the background traffic is increasing or reducing the number of flows related to the attack. Such modification of the attack does not affect the representation, as long as there are enough flows to create the feature vectors.

- **Time intervals between flows** – This feature has been used in many previous approaches for its descriptive properties. It is an alternative way to the proposed representation how to model a relationship between individual flows. Our analysis revealed that current malware samples frequently modify the inter-arrival time to remain hidden in the background traffic – see Figure 3 for details. Therefore, we do not rely on this unstable feature that can be also influenced by network delays or failures.

- **Ordering of flows** – An attacker can easily change the ordering of flows to evade detection based on patterns or predefined sequences of flows. For the proposed representation the ordering of flows does not matter.

The proposed representation is not invariant to the following changes.

- **Static behavior** – The representation does not model malware behaviors, where all flows associated with a malware are identical. Such behavior has no dynamics and can be classified with flow-based approaches with comparable results. In our dataset, only 10% of flows were removed because of this constrain.

- **Multiple behaviors in a bag** – In case more behaviors are associated with a bag, such as when a target hostname is compromised and communicates with a user with legitimate and malicious flows at once, the representation does not guarantee the invariance against the attacker's changes. Such bags contain a mixture of legitimate and malicious flows and their combination could lead to a different representation. Note that there wasn't any malware sample in our data that would satisfy this condition, since the legitimate traffic has to be authentic (not artificially injected) to confuse the representation.

- **Encrypted HTTPS traffic** – Most features presented in this paper are computed from URLs or other flow fields, that are not available in encrypted HTTPS traffic. In this case, only a limited set of flow-based features can be used, which reduces the discriminative properties of the representation. However, majority of malware communication is still over HTTP protocol, because switching to HTTPS would harm the cyber-criminals' revenues due to problems with signed certificates [18].

- **Real-time changes and evolution** – In case a malware sample for a given user and hostname would start changing its behavior dynamically and frequently, the bag representation will vary in time. Such inconsistency would decrease the efficacy results and enlarge the time to detect. However, creating such highly dynamic malware behavior requires a considerable effort, therefore we do not see such samples very often in the real network traffic.

Category	Samples		Signatures
	Flows	Bags	Recall
Training Positives	132,756	5,011	0.15
Click-fraud mw	12,091	819	0.29
DGA malware	8,629	397	0.58
Dridex	8,402	264	0.12
IntallCore	17,317	1,332	0.00
Monetization	3,107	135	0.00
Mudrop	37,142	701	0.00
Poweliks	11,648	132	0.00
Zeus	34,420	1,275	0.19
Testing Positives	43,380	2,090	0.02
Training Negatives	862,478	26,825	
Testing Negatives	15,379,466	240,549	

Table 2: Number of flows and bags of malware categories and legitimate background traffic used for training and testing the proposed representation and classifier. Right-most column shows the amount of bags that were found and blocked by an existing signature-based device. Majority of the malicious bags from the test were missed, as the device, relying on a static database of signatures, was not able to catch evolving versions and new types of the malicious behaviors.

We conclude our analysis with the observation, that attackers change flow features very frequently (see Figure 3). The goal of the proposed representation is to be invariant against most of the changes to successfully detect new, previously unseen malware variants.

8 Experimental Evaluation

The proposed approach was deployed on the top of proxy logs exporters in companies of various types and sizes to detect unseen malware samples. The system architecture is shown in Figure 4. Collector connected to a

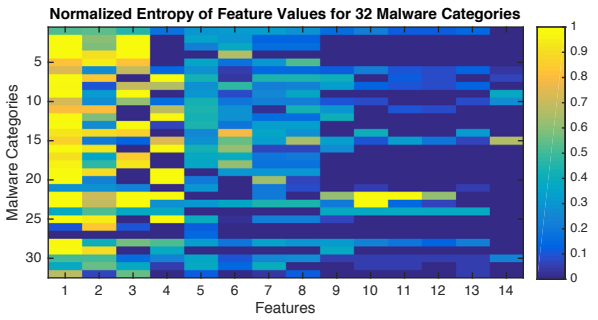

Figure 3: Flow-based features (columns) are changing for most of the malware categories (rows). The figure uses normalized entropy to show the variability of each feature within each malware category. Yellow color denotes that the feature value is changed very often, while blue color means that the feature has the same values for all samples of the given category. **Features:** 1-URL, 2-interarrival time, 3-URL query values, 4-URL path, 5-number of flows, 6-number of downloaded bytes, 7-server IP address, 8-hostname, 9-URL path length, 10-URL query names, 11-filename, 12-filename length, 13-number of URL query parameters, 14-number of uploaded bytes. **Malware categories:** 1-Click-fraud (amz), 2-Asterope family 1, 3-Asterope family 2, 4-Beden, 5-Click-fraud, 6-DGA, 7-Dridex, 8-Exfiltration, 9-InstallCore, 10-Mudrop Trojan Dropper, 11-Monetization, 12-Zeus, 13-Mudrop, 14-MultiPlug, 15-mixture of unknown malware, 16-Click-fraud (tracking), 17-Poweliks family 1, 18-Poweliks family 2, 19-Qakbot Trojan, 20-Rerdom Trojan, 21-Ramnit worm, 22-RVX, 23-Sality, 24-Threats related to a traffic direction system (TDS) 1, 25-TDS 2, 26-TDS 3, 27-Tinba Trojan, 28-C&C tunneling, 29-Upatre, 30-Vawtrak, 31-Vittalia, 32-Zbot. Details about the malware categories are given in Section 8.

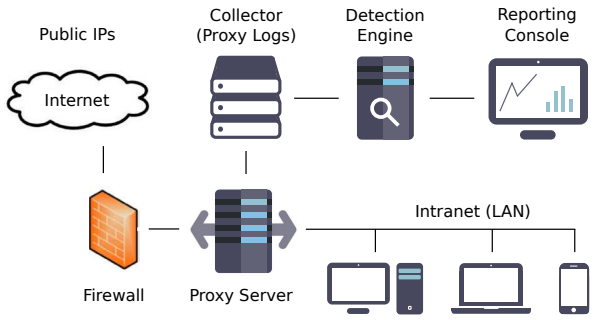

Figure 4: Overview of the system architecture. Collector connected to a proxy server stores incoming and outgoing network traffic in form of proxy log records. Each 5-minute interval, the proxy logs are sent to the detection engine and the results are displayed to an operator on the reporting console.

proxy server stores incoming and outgoing network traffic in form of proxy log records. The proxy logs represent information about individual HTTP/HTTPS connections or flows. Each 5-minute interval, the proxy logs are sent to the detection engine, where the proposed method detects the malicious behaviors. Report created from the malicious behaviors is then displayed on a console to an operator. The next section provides the specification of datasets and malware categories, followed by the results from the experimental evaluation. Next section provides the specification of datasets and malware categories, followed by the results from the experimental evaluation.

8.1 Specification of the Datasets

The data was obtained from several months (January - July 2015) of real network traffic of 80 international companies of various sizes in form of proxy logs [26]. The logs contain HTTP/HTTPS flows, where one flow is one connection defined as a group of packets from a single host and source port with a single server IP address, port, and protocol. Summary of the datasets used in the evaluation is described in Table 2.

Malware samples will be referred as **positive bags**, where one positive bag is a set of records (connections) with the same source towards the same destination. The bags not labeled as malicious are considered as legitimate/negative. Each bag should contain at least 5 flows to be able to compute a meaningful histogram representation. Training dataset contains 5k malicious (8 malware families) and 27k legitimate bags, while testing dataset is consist of 2k malicious ($\gg$ 32 malware families) and 241k legitimate bags (more than 15 million flows). Positive samples for training were acquired using many types of publicly available feeds, services, and blacklists, while the results on the testing data were analyzed manually by security experts. Each HTTP flow consists of the following fields: user name, srcIP, dstIP, srcPort, dstPort, protocol, number of bytes, duration, timestamp, user agent, and URL. From these flow fields, we extracted 115 flow-based features typically used in the prior art (Table 3).

This means that training and testing data are composed of completely different malware bags from different malware families, which makes the classification problem much harder. This scenario simulates the fact that new types of threats are created to evade detection. The benchmarking signature-based network security device (widely used in many companies) was able to detect only 2% of the malicious bags from the testing set. Training a classifier for each category separately is an easier task, however such classifiers are typically overfitted to a single category and cannot detect further variations without retraining.

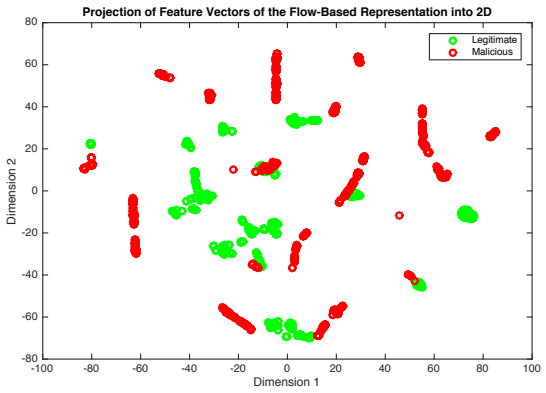

Figure 5: Graphical projection of feature vectors of the baseline flow-based representation into two dimensions using t-SNE transformation. Feature vectors from 32 different malware categories are displayed. Due to high variability of flow-based feature values, legitimate and malicious samples are scattered without any clear separation. The results show that the flow-based representation is suitable for training classifiers specialized on a single malware category, which often leads to classifiers with high precision and low recall.

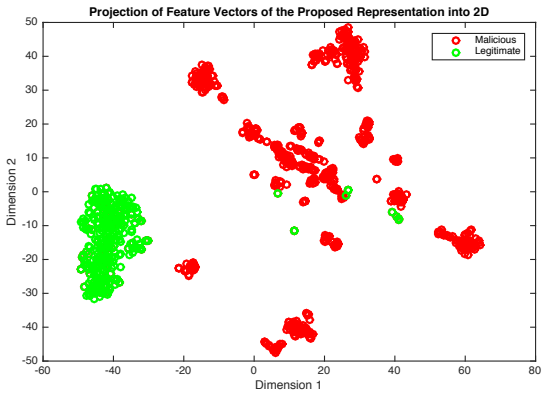

Figure 6: Graphical projection of feature vectors of the proposed representation into two dimensions using t-SNE transformation. Thanks to the invariant properties, malicious bags from various categories are grouped together, as they have similar dynamics modeled by the representation. Most of the legitimate bags are concentrated on the left-hand side, far from the malicious bags. This shows that training a classifier with the proposed representation will achieve higher recall with comparable precision.

Features applied on URL, path, query, filename
length; digit ratio
lower/upper case ratio; ratio of digits
vowel changes ratio
ratio of a character with max occurrence
has a special character
max length of consonant/vowel/digit stream
number of non-base64 characters
has repetition of parameters

Other Features
number of bytes from client to server
number of bytes from server to client
length of referer/file extension
number of parameters in query
number of '/' in path/query/referer

Table 3: List of selected flow-based features extracted from proxy logs. We consider these features as baseline (as some features were used in previously published work), and compare it with the proposed representation.

Table 4 from Appendix A describes an important fact about the URLs from individual malicious bags. As you can see, URLs within each malicious bag are similar to each other (as opposed to most of legitimate bags). This small non-zero variability of flow-based feature values is captured by the proposed representation using both types of histograms. The variability is very general but also descriptive feature, which increases the robustness of the representation to further malware changes and variants.

8.2 Evaluation on Real Network Traffic

This section shows the benefits of the proposed approach of learning the invariant representation for two-class classification problem in network security. Feature vectors described in Section 8.1 correspond to input feature vectors $\{x_1,\ldots,x_m\}$ defined in Section 3. These vectors are transformed into the proposed representation of histograms $\phi(\tilde{X};\tilde{\mathscr{S}};\theta)$, as described in Section 4. We have evaluated two types of invariant representations. One with predefined number of equidistant bins (e.g. 16, 32, etc.) computed as described in Section 4, and one when the representation is learned together with the classifier to maximize the separability between malicious and legitimate traffic (combination of Section 4 and 5). For the representation learning, we used 256 bins as initial (and most detailed) partitioning of the histograms. During the learning phase, the bins were merged together, creating 12.7 bins per histogram on average.

Both approaches are compared with the baseline flow-based representation used in previously published work, where each sample corresponds to a feature vector computed from one flow. Results of a widely used signature-based security device are also provided (see Table 2) to demonstrate that the positive samples included in the evaluation pose a real security risk, as majority of them

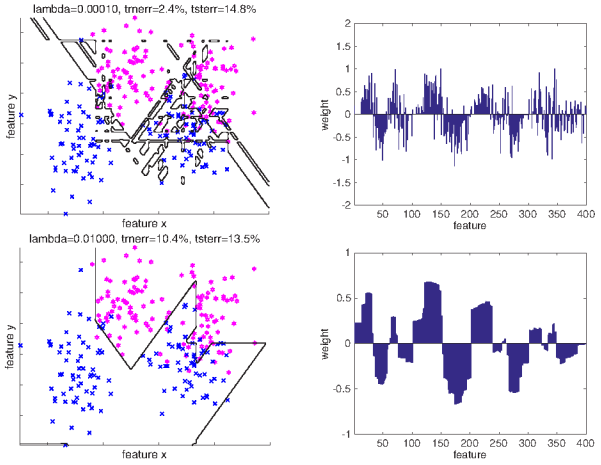

Figure 7: Visualization of the proposed method of learning the invariant representation on 2-dimensional synthetic data. Figures in the left row show the decision boundaries of two class classifier learned from the bins for two different values of parameter λ (0.0001, 0.01) which controls the number of emerging bins (the corresponding weights are shown in the right row). With increasing λ the data are represented with less bins and the boundary becomes smoother and less over-fitted to the training data.

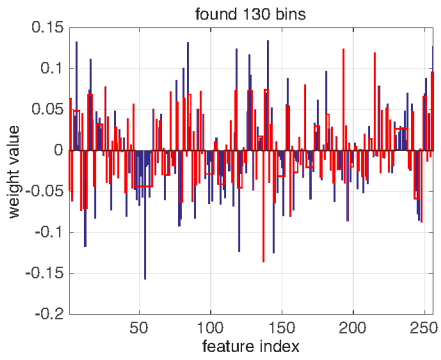

Figure 8: Weights (blue bars) and derived bins of a histogram (red line) for a standard SVM and one of the invariant features. Since the bins are equidistant and predefined at the beginning, the resulting histogram (defined by the red line) has complicated structure, leading most probably to complex boundary and over-fitted results (as shown in Figure 7 on the left hand side).

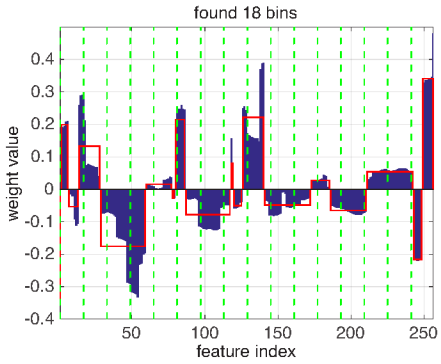

Figure 9: Weights (blue bars) and derived bins of a histogram (red line) for the proposed bin optimization. In this case, the weights show a clear structure and the derived histogram has only 18 bins. The decision boundary is in this case smoother and the classifier trained from this representation will be more robust. Green dashed lines also show how the histogram bins would look like if they are positioned equidistantly (16 bins).

was not detected. Maximum number of flows for each bag was 100, which ensures that the computational cost is controlled and does not exceed predefined limits.

Two-dimensional projection of the feature vectors for the flow-based and the proposed representation is illustrated in Figures 5 and 6 respectively. Bags from 32 malicious categories are displayed with red circles, while the legitimate bags are denoted with green circles. The projections show that the flow-based representation is suitable for training classifiers specialized on a single malware category. In case of the proposed representation, malicious bags from various categories are grouped together and far from the legitimate traffic, which means that the classifiers will have higher recall and comparable precision with the flow-based classifiers.

Next, we will show the properties of the proposed method of learning the representation to maximize the separation between positive and negative samples (see Section 5 for details). Figure 7 visualizes the proposed method on synthetic 2-dimensional input data. The input 2D point $(x,y) \in \mathbb{R}^2$ is represented by 4-dimensional feature vector $(x^2, y^2, x+y, x-y)$. Each of the 4 features is then represented by a histogram with 100 bins (i.e. each feature is represented by 100 dimensional binary vector will all zeros but a single one corresponding to the active bin). Figures in the top row show the decision boundaries of two-class classifiers learned from data. The bottom row shows the weights of the linear classifier corresponding to the bins (in total 400 weights resulting from 100 bins for each out of 4 features). The columns correspond to the results obtained for different setting of the parameter λ which controls the number of emerging bins and thus also the complexity of the decision boundary. With increasing λ the data are represented with less bins and the boundary becomes smoother. Figure 7 shows the principle of the proposed optimization process. The bins of the representation are learned in such a way that it is much easier for the classifier to separate negative and positive samples and at the same time control the com-

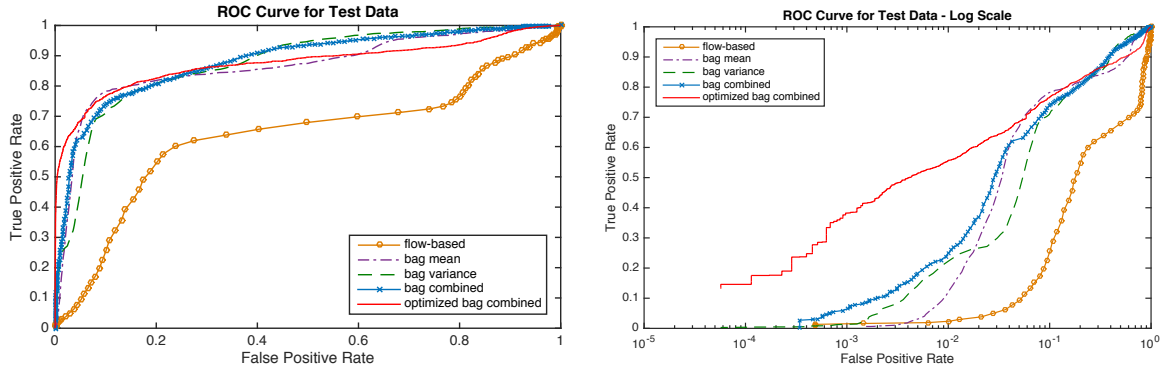

Figure 10: ROC curves of SVM classifier on test data for five types of representations (logarithmic scale on the right). Flow-based representation shows very unsatisfactory results showing that flow-based approach cannot be applied in practice to detect unseen malware variants. The combination of feature values with feature differences histogram (bag combined) led to significantly better efficacy results. These results were further exceeded when the parameters of the invariant representation were learned automatically from the training data (optimized bag combined).

plexity of the classifier.

Figures 8 and 9 show the bins and weights learned from the training set of real network traffic. The blue vertical lines represent learned weights associated with 256 bins of a histogram computed on a single input feature. The red lines show new bins derived from the weights by merging those neighboring bins which have the weights with the same sign. Figure 8 shows the weights and the derived bins for a standard SVM which has no incentive to have similar weights. The histogram derived from the SVM weights reduces the number of bins from 256 to 130. Figure 9 shows the results for the proposed method which enforces the similar weights for neighboring bins. In this case, the weights exhibit a clear structure and the derived histogram has only 18 bins. The decision boundary is in this case smoother and the classifier trained from this representation will be more robust.

Next, a two-class SVM classifier was evaluated on five representations: baseline flow-based, per-feature histograms of values $\phi(z_k^X, \theta_k^X)$ (bag mean), per-feature histograms of feature differences $\phi(z_k^{\mathscr{D}}, \theta_k^S)$ (bag variance), the combination of both (bag combined), and the combination of both with bin optimization (optimized bag combined). The training and testing datasets were composed of bags described in Table 2.

The results on testing data are depicted in Figure 10. Note that positive bags in the testing set are from different malware categories than bags from the training set, which makes the classification problem much harder. The purpose of this evaluation is to compare flow-based representation, which is used in most of previously published work, with the proposed invariant representation. Flow-based representation shows very unsatisfactory results, mainly due to the fact that the classifier was based only on the values of flow-based features that are not

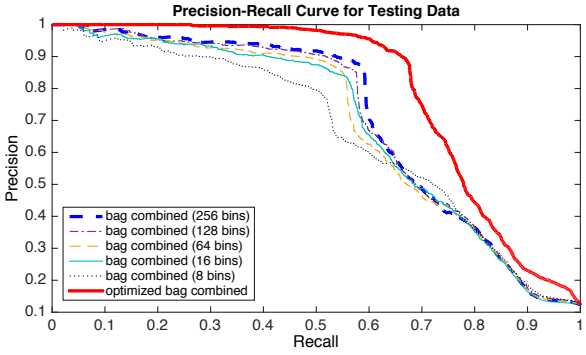

Figure 11: Precision-recall curve of SVM classifier trained on the proposed representation with different number of histogram bins for each feature. All classifiers are outperformed by the classifier, where the parameters of the invariant representation are learned automatically from the data (optimized bag combined). The classifier achieved 90% precision (9 of 10 alerts were malicious) and 67% recall on previously unseen malware families.

robust across different malware categories (as shown in Section 7). The classifier based on combined bag representation performed significantly better. These results were further exceeded when the parameters of the invariant representation were learned automatically from the training data (optimized bag combined), which is shown in Figure 10 with logarithmic scale.

Precision-recall curve is depicted in Figure 11 to compare the efficacy results of classifiers based on the proposed representation with predefined number of bins per feature (8, 16, 64, 128, and 256 bins) with the same representation, but when the parameters are learned from the training data (using bin optimization from Section 5).

Overall, the results show the importance of combining both types of histograms introduced in Section 4 together, allowing the representation to be more descriptive and precise without sacrificing recall. But most importantly, when the parameters of the representation are trained to maximize the separability between malicious and legitimate samples, the resulting classifier performs in order of a magnitude better than a classifier with manually predefined parameters.

9 Conclusion

This paper proposes a robust representation suitable for classifying evolving malware behaviors. It groups sets of network flows into bags and represents them using a the combination of invariant histograms of feature values and feature differences. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. The proposed optimization method learns the parameters of the representation automatically from the training data, allowing the classifiers to create robust models of malicious behaviors capable of detecting previously unseen malware variants and behavior changes.

The proposed representation was deployed on corporate networks and evaluated on real HTTP network traffic with more than 43k malicious samples and more than 15M samples overall. The comparison with a baseline flow-based approach and a widely-used signature-based web security device showed several key advantages of the proposed representation. First, the invariant properties of the representation result in the detection of new types of malware. More specifically, the proposed classifier trained on the optimized representation achieved 90% precision (9 of 10 alerts were malicious) and detected 67% of malware samples of previously unseen types and variants. Second, multiple malware behaviors can be represented in the same feature space while current flow-based approaches necessitate training a separate detector for each malware family. This way, the proposed system considerably increases the capability of detecting new variants of threats.

References

[1] Cisco netflow. http://www.cisco.com/warp/public/732/tech/netflow.

[2] ANTONAKAKIS, M., PERDISCI, R., NADJI, Y., VASILOGLOU, N., ABU-NIMEH, S., LEE, W., AND DAGON, D. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Proceedings of the 21st USENIX Conference on Security Symposium* (Berkeley, CA, USA, 2012), Security'12, USENIX Association, pp. 24–24.

[3] BAILEY, M., OBERHEIDE, J., ANDERSEN, J., MAO, Z., JAHANIAN, F., AND NAZARIO, J. Automated classification and analysis of internet malware. In *Recent Advances in Intrusion Detection*, C. Kruegel, R. Lippmann, and A. Clark, Eds., vol. 4637 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 178–197.

[4] BEN-DAVID, S., BLITZER, J., CRAMMER, K., PEREIRA, F., ET AL. Analysis of representations for domain adaptation. *Advances in neural information processing systems 19* (2007), 137.

[5] BERNAILLE, L., TEIXEIRA, R., AKODKENOU, I., SOULE, A., AND SALAMATIAN, K. Traffic classification on the fly. *ACM SIGCOMM '06 36*, 2 (Apr. 2006), 23–26.

[6] BILGE, L., BALZAROTTI, D., ROBERTSON, W., KIRDA, E., AND KRUEGEL, C. Disclosure: Detecting botnet command and control servers through large-scale netflow analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference* (New York, NY, USA, 2012), ACSAC '12, ACM, pp. 129–138.

[7] BLITZER, J., MCDONALD, R., AND PEREIRA, F. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing* (2006), Association for Computational Linguistics, pp. 120–128.

[8] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. Anomaly detection: A survey. *ACM Comput. Surv. 41* (July 2009), 15:1–15:58.

[9] CHOI, H., ZHU, B. B., AND LEE, H. Detecting malicious web links and identifying their attack types. In *Proceedings of the 2Nd USENIX Conference on Web Application Development* (Berkeley, CA, USA, 2011), WebApps'11, USENIX Association, pp. 11–11.

[10] DAI, W., YANG, Q., XUE, G.-R., AND YU, Y. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 193–200.

[11] DUAN, L., TSANG, I. W., AND XU, D. Domain transfer multiple kernel learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 3 (2012), 465–479.

[12] ERMAN, J., ARLITT, M., AND MAHANTI, A. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data* (New York, NY, USA, 2006), MineNet '06, ACM, pp. 281–286.

[13] FALLIERE, N. Sality: Story of a peer-to-peer viral network. *Rapport technique, Symantec Corporation* (2011).

[14] GRETTON, A., SMOLA, A., HUANG, J., SCHMITTFULL, M., BORGWARDT, K., AND SCHÖLKOPF, B. Covariate shift by kernel mean matching. *Dataset shift in machine learning 3*, 4 (2009), 5.

[15] GRIFFIN, K., SCHNEIDER, S., HU, X., AND CHIUEH, T.-C. Automatic generation of string signatures for malware detection. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection* (Berlin, Heidelberg, 2009), RAID '09, Springer-Verlag, pp. 101–120.

[16] GU, G., PERDISCI, R., ZHANG, J., LEE, W., ET AL. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium* (2008), vol. 5, pp. 139–154.

[17] HUANG, H., QIAN, L., AND WANG, Y. A svm-based technique to detect phishing urls. *Information Technology Journal 11*, 7 (2012), 921–925.

[18] INVERNIZZI, L., MISKOVIC, S., TORRES, R., SAHA, S., LEE, S., MELLIA, M., KRUEGEL, C., AND VIGNA, G. Nazca: Detecting malware distribution in large-scale networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2014).

[19] IYER, A., NATH, S., AND SARAWAGI, S. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014), pp. 530–538.

[20] JAGPAL, N., DINGLE, E., GRAVEL, J.-P., MAVROMMATIS, P., PROVOS, N., RAJAB, M. A., AND THOMAS, K. Trends and lessons from three years fighting malicious extensions. In *24th USENIX Security Symposium (USENIX Security 15)* (Washington, D.C., Aug. 2015), USENIX Association, pp. 579–593.

[21] JUNEJO, I. N., DEXTER, E., LAPTEV, I., AND PEREZ, P. View-independent action recognition from temporal self-similarities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 33*, 1 (2011), 172–185.

[22] KAPRAVELOS, A., SHOSHITAISHVILI, Y., COVA, M., KRUEGEL, C., AND VIGNA, G. Revolver: An automated approach to the detection of evasive web-based malware. In *USENIX Security* (2013), Citeseer, pp. 637–652.

[23] KARAGIANNIS, T., PAPAGIANNAKI, K., AND FALOUTSOS, M. Blinc: Multilevel traffic classification in the dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2005), SIGCOMM '05, ACM, pp. 229–240.

[24] KIM, H., CLAFFY, K., FOMENKOV, M., BARMAN, D., FALOUTSOS, M., AND LEE, K. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT Conference* (New York, NY, USA, 2008), CoNEXT '08, ACM, pp. 11:1–11:12.

[25] KRUEGEL, C., AND VIGNA, G. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2003), CCS '03, ACM, pp. 251–261.

[26] LOU, W., LIU, G., LU, H., AND YANG, Q. Cut-and-pick transactions for proxy log mining. In *Advances in Database Technology EDBT 2002*, C. Jensen, S. altenis, K. Jeffery, J. Pokorny, E. Bertino, K. Bhn, and M. Jarke, Eds., vol. 2287 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 88–105.

[27] MA, J., SAUL, L. K., SAVAGE, S., AND VOELKER, G. M. Learning to detect malicious urls. *ACM Trans. Intell. Syst. Technol. 2*, 3 (May 2011), 30:1–30:24.

[28] MOORE, D., SHANNON, C., BROWN, D. J., VOELKER, G. M., AND SAVAGE, S. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst. 24*, 2 (May 2006), 115–139.

[29] MOSER, A., KRUEGEL, C., AND KIRDA, E. Exploring multiple execution paths for malware analysis. In *Security and Privacy, 2007. SP '07. IEEE Symposium on* (May 2007), pp. 231–245.

[30] MOSER, A., KRUEGEL, C., AND KIRDA, E. Limits of static analysis for malware detection. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual* (Dec 2007), pp. 421–430.

[31] MÜLLER, M., AND CLAUSEN, M. Transposition-invariant self-similarity matrices. In *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)* (2007), pp. 47–50.

[32] NELMS, T., PERDISCI, R., ANTONAKAKIS, M., AND AHAMAD, M. Webwitness: Investigating, categorizing, and mitigating malware download paths. In *24th USENIX Security Symposium (USENIX Security 15)* (Washington, D.C., Aug. 2015), USENIX Association, pp. 1025–1040.

[33] PORTNOY, L., ESKIN, E., AND STOLFO, S. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001* (2001), pp. 5–8.

[34] RIECK, K., HOLZ, T., WILLEMS, C., DSSEL, P., AND LASKOV, P. Learning and classification of malware behavior. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, D. Zamboni, Ed., vol. 5137 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 108–125.

[35] RNDIC, N., AND LASKOV, P. Practical evasion of a learning-based classifier: A case study. In *Security and Privacy (SP), 2014 IEEE Symposium on* (May 2014), pp. 197–211.

[36] SCARFONE, K., AND MELL, P. Guide to intrusion detection and prevention systems (idps) recommendations of the national institute of standards and technology. *Nist Special Publication 800*, 94 (2007).

[37] SHIMODAIRA, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference 90*, 2 (2000), 227–244.

[38] SONG, D., BRUMLEY, D., YIN, H., CABALLERO, J., JAGER, I., KANG, M., LIANG, Z., NEWSOME, J., POOSANKAM, P., AND SAXENA, P. Bitblaze: A new approach to computer security via binary analysis. In *Information Systems Security*, R. Sekar and A. Pujari, Eds., vol. 5352 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 1–25.

[39] SONG, H., AND TURNER, J. Toward advocacy-free evaluation of packet classification algorithms. *Computers, IEEE Transactions on 60*, 5 (May 2011), 723–733.

[40] SOSKA, K., AND CHRISTIN, N. Automatically detecting vulnerable websites before they turn malicious. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, Aug. 2014), USENIX Association, pp. 625–640.

[41] WANG, K., AND STOLFO, S. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection*, E. Jonsson, A. Valdes, and M. Almgren, Eds., vol. 3224 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 203–222.

[42] YIN, H., SONG, D., EGELE, M., KRUEGEL, C., AND KIRDA, E. Panorama: Capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2007), CCS '07, ACM, pp. 116–127.

[43] ZHANG, K., SCHÖLKOPF, B., MUANDET, K., AND WANG, Z. Domain adaptation under target and conditional shift. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (2013), S. Dasgupta and D. Mcallester, Eds., vol. 28, JMLR Workshop and Conference Proceedings, pp. 819–827.

[44] ZHAO, P., AND HOI, S. C. Cost-sensitive online active learning with application to malicious url detection. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2013), KDD '13, ACM, pp. 919–927.

A Examples of Bags

Asterope
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3398&id=126&r=12739868&os=6.1—2—8.0.7601.18571&res=4—1921—466&f=1
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3398&id=126&r=15425581&os=6.1—2—8.0.7601.18571&res=4—1921—516&f=1
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3398&id=126&r=27423103&os=6.1—2—8.0.7601.18571&res=4—1921—342&f=1
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3753&id=126&r=8955018&os=6.1—2—8.0.7601.18571&res=4—1921—319&f=1
Click-fraud, malvertising-related botnet
hxxp://directcashfunds.com/opntrk.php?tkey=024f9730e23f8553c3e5342568a70300&Email=name.surname@company.com
hxxp://directcashfunds.com/opntrk.php?tkey=c1b6e3d50632d4f5c0ae13a52d3c4d8d&Email=name.surname@company.com
hxxp://directcashfunds.com/opntrk.php?tkey=7c9a843ce18126900c46dbe4be3b6425&Email=name.surname@company.com
hxxp://directcashfunds.com/opntrk.php?tkey=c1b6e3d50632d4f5c0ae13a52d3c4d8d&Email=name.surname@company.com
DGA
hxxp://uvyqifymelapuvoh.biz/s531ka.ji5
hxxp://uvyqifymelapuvoh.biz/rl59c281.x19
hxxp://uvyqifymelapuvoh.biz/seibpn6.2m0
hxxp://uvyqifymelapuvoh.biz/3854f.u17
Dridex
hxxp://27.54.174.181/8qV578&$o@HU6Q6S/gz$J0l=iTTH 28%2CM/we20%3D
hxxp://27.54.174.181/C4GyRx%7E@RY6x /M&N=sq/bW_ra4OTJ
hxxp://27.54.174.181/gPvh+=GO/9RPPfk0%2CzXOYU%20/Vq8Ww/+a_m%7Ez
hxxp://27.54.174.181/qE0my4KIz48Cf3H8wG%7Evpz=iJ%26fqMl%24m/46JoELp=GJww%3D%26Ib+Ar.y3 iu%2D1E/sso

InstallCore	**Monetization**
hxxp://rp.any-file-opener.org/?pcrc=1559319553&v=2.0	hxxp://utouring.net/search/q/conducing
hxxp://rp.any-file-opener.org/?pcrc=1132521307&v=2.0	hxxp://utouring.net/go/u/1/r/1647
hxxp://rp.any-file-opener.org/?pcrc=1123945956&v=2.0	hxxp://utouring.net/go/u/0/r/2675
hxxp://rp.any-file-opener.org/?pcrc=1075608192&v=2.0	hxxp://utouring.net/search/f/1/q/refiles

Poweliks
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=nitric+oxide+side+effects&ua=Mozilla%2F5 . . . &lr=7&ls=0
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=weight+loss+success+stories&ua=Mozilla%2F5 . . . &lr=0&ls=0
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=shoulder+pain&ua=Mozilla%2F5 . . . &lr=7&ls=2
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=cheap+car+insurance&ua=Mozilla%2F5 . . . &lr=7&ls=2
Zeus
hxxp://130.185.106.28/m/IbQFdXVjiriLva4KHeNpWCmThrJBn3f34HNwsLVVsUmLXtsumSSPe/zzXtIu9SzwjI9zKlxdE . . . 3RqvGzKN5
hxxp://130.185.106.28/m/IbQJFUVjgZn4vx4KHeNpWCmThrJBn3f34HNwsLVVsUmLfkoPaSS+S+zzXtIu9SzwjI9zKlxdE . . . 3vKwmk0oUi
hxxp://130.185.106.28/m/IbQJFUVjiJwJBX4KHeNpWCmThrJBn3f34HNwsLVVsUmKH7ue2STvSkzzXtIu9SzwjI9zKlxdE . . . 3vKwmk0oUi
hxxp://130.185.106.28/m/IbQNtVVji5/7Yp4KHeNpWCmThrJBn3f34HNwsLVVsUmLz4sO6YRvOjzzXtIu9SzwjI9zKlxdE . . . 3zB9057quqv
Legitimate traffic 1
hxxp://www.cnn.com/.element/ssi/auto/4.0/sect/MAIN/markets_wsod_expansion.html
hxxp://www.cnn.com/.a/1.73.0/assets/sprite-s1dced3ff2b.png
hxxp://www.cnn.com/.element/widget/video/videoapi/api/latest/js/CNNVideoBootstrapper.js
hxxp://www.cnn.com/jsonp/video/nowPlayingSchedule.json?callback=nowPlayingScheduleCallbackWrapper&_=1422885578476
Legitimate traffic 2
hxxp://ads.adaptv.advertising.com/a/h/7g_doK40WLPMYHbkD9G2u7HSXjqzIaa7Bqhslod+u7iQl . . . &context=fullUrl%3Dpandora.com
hxxp://ads.adaptv.advertising.com/crossdomain.xml
hxxp://ads.advertising.com/411f1e96-3bde-4d85-b17e-63749e5f0695.js
hxxp://ads.adaptv.advertising.com/applist?placementId=297920&key=&d.vw=1&orgId=8656&hostname=data.rtbfy.com

Table 4: Example URLs of flows from several malicious bags and from two legitimate bags. The URLs within each malicious bag are similar to each other while the URLs within legitimate bags differ. The small non-zero variability of flow-based feature values is captured by the proposed representation using histograms of features and feature self-similarity matrices. Such transformation of the feature values makes the representation robust to malware changes and unseen variants.