Learning detector of malicious network traffic from weak labels

Vojtech Franc[1][2], Michal Sofka[1], and Karel Bartos[1]

[1] Cisco systems,
Prague, Czech Republic,
[2] Czech Technical University in Prague,
Faculty of Electrical Engineering,
Department of Cybernetics

Abstract. We address the problem of learning a detector of malicious behavior in network traffic. The malicious behavior is detected based on the analysis of network proxy logs that capture malware communication between client and server computers. The conceptual problem in using the standard supervised learning methods is the lack of sufficiently representative training set containing examples of malicious and legitimate communication. Annotation of individual proxy logs is an expensive process involving security experts and does not scale with constantly evolving malware. However, weak supervision can be achieved on the level of properly defined bags of proxy logs by leveraging internet domain black lists, security reports, and sandboxing analysis. We demonstrate that an accurate detector can be obtained from the collected security intelligence data by using a Multiple Instance Learning algorithm tailored to the Neyman-Pearson problem. We provide a thorough experimental evaluation on a large corpus of network communications collected from various company network environments.

Keywords: computer security, malware detection, multiple-instance learning, support vector machines

1 Introduction

Recent report has revealed that 100 percent of all investigated corporate networks had malicious traffic going to web sites that host malware [1]. Detecting malware infections is challenging since malware is rapidly evolving, the complexity of the attacks is increasing, and the number of different variants is rising. This security challenge creates a need for an automated analysis and detection of malware. We propose a learning-based system leveraging publicly available blacklists of malware domains to train a detector that automatically finds malicious communication.

Traditional approaches to network analysis rely on extracting communication patterns from HTTP proxy logs that are distinctive for malware [11]. The pattern matching is fast but it is difficult to keep up with constantly changing malware.

Behavioral techniques extract features from the proxy log fields and build a detector that generalizes to the particular malware family exhibiting the targeted behavior [10]. Previous algorithms were engineered to specific types of malicious activity such as spam [6], phishing [4], or communication with a command and control server of the attacker [9].

Our system extracts a number of generic features from proxy logs of HTTP requests and trains a detector of malicious communication using publicly-available blacklists of malware domains [7]. Since the blacklists contain labeling only at the level of domains while the detector operates on richer proxy logs with a full target web site URL, the labeled domains only provide weak supervison for training. We propose a variant of the Multiple Instance Learning algorithm (MIL) [5] that uses bags of proxy logs describing communication of users to the black-listed domains instead of manually labeled positive examples of network communication. The proposed MIL algorithm seeks for the Neyman-Pearson detector with a very low false positive rate that is necessary in the real-life deployment of the system. The resulting non-convex learning problem is solved by Averaged Stochastic Gradient Descent [13].

The algorithm results in a generic system that can recognize malicious traffic by learning from weak annotations. This simplifies the update process based on newly discovered threats since the detector of malicious HTTP requests can be reliably trained from domain blacklists. We show on an extensive dataset of traffic logs obtained from a large corporate network that the algorithm reliably detects new malware while keeping low false positive rates.

The paper is organized as follows. The malware detection problem is briefly described in Section 2. Formulation of the learning problem and its solution is presented in Section 3. Section 4 details the used database of the network traffic and the data representation. Experiments are given in Section 5 and Section 6 concludes the paper.

2 Malware Detection

The initial computer infection with a malware begins by executing a malicious program, for example by clicking on a link embedded in a website or an email. The reasons behind the infections are different (usually prompted by financial gain through ex-filtrating and abusing sensitive data) but in all cases the attacker (and the malware) needs to communicate over the network connection. The communication can involve scanning for potential targets, initial download of the malicious binary or library, or connection to the command and control server maintained by the attacker.

In a network analysis system, the HTTP communication is captured by network proxy logs (also called flows) that contain several fields specific to the HTTP protocol. Since the logs are recorded for every elementary action on the network (e.g. click on a link and downloading parts of a website), they only contain basic information about the data transfer and do not include the target website content or the malicious binary file. Despite their simplicity, the logs can be used to detect communication corresponding to malicious behavior.

The detection of malicious communication is done based on features extracted from the proxy log fields, such as the URL, flow duration, number of bytes transferred from client to server and from server to client, user agent, referer (address of the previous web page that was followed to this page), MIME-Type, and HTTP status. The URL is the most informative and has been traditionally represented by n-grams and their statistics. The resulting n-dimensional feature vector represents each proxy log and is used to discriminate between malicious and legitimate HTTP request. In this work, we train a binary classifier and process the logs individually. Although this ignores important high level information since the malicious communication is sometimes periodic and could therefore benefit from temporal features, we will show that our low level model can already detect malware reliably.

The problem of supervised training in network security is the lack of sufficiently large and representative dataset of labeled malicious and legitimate samples. The labels are expensive to obtain since the process involves forensic analysis performed by security experts. Sometimes, the labels are not even possible to assign, especially if the context of the network communication is small or unknown and the assignment is desired at a proxy-log level. Furthermore, the labeled dataset becomes obsolete quite quickly, as a matter of weeks or months, due to constantly evolving malware. As a compromise, domain-level labeling has been frequently adopted by compiling blacklists of malicious domains registered by the attackers. The domain blacklists can then be used to block network communication based on the domain of the destination URL in the proxy log. However, the malicious domains typically change frequently as a basic detection evasion technique. Even though the domains might change, the other parts of the HTTP request (and the behavior of the malware) remain the same or similar. This is exploited in our behavioral model of malicious traffic.

Our semi-supervised training takes the advantage of security intelligence captured in the form of domain blacklists collected from various security reports, data feeds, and sandboxing analysis. Since our goal is to detect malicious behavior at the level of individual flows, the domain blacklists offer a weak supervision in the classification task. The proxy logs originating at a particular user machine are grouped into bags based on the domains in the URL. Therefore, the bags are constructed for each user machine and all flows to a domain visited in a particular time window (24 hours). The bags are labeled according to the domain: if the domain was marked as positive in any of the blacklists, the bag has a positive label. Otherwise, the bag is labeled as negative.

Leveraging the labels at the level of bags has the advantage that publicly available sources of domain blacklists can be used for the classifier training. The problem is then formulated as weakly supervised learning since the bag labels are used to train a classifier of individual flows. We propose to solve the problem by Multiple Instance Learning (MIL) which we describe next.

3 Multiple Instance Learning of Neyman Pearson detector

We start with defining a statistical model of the data. A flow is described by a vector of features $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a label $y \in \mathcal{Y} = \{+1, -1\}$ where $y = +1$ means the malicious and $y = -1$ the legitimate flow, respectively. The network traffic monitored in a given period is fully described by the *completely annotated data* $\mathcal{D}_{\rm cmp} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ assumed to be generated from i.i.d. random variables with an unknown distribution $p(\boldsymbol{x}, y)$. Obtaining the complete annotation is expensive hence we collect a weaker annotation by assigning labels to bags of flows. The *weakly annotated data* $\mathcal{D}_{\rm bag} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, (\mathcal{B}_1, z_1), \ldots, (\mathcal{B}_n, z_n)\}$ are composed of the flow features $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \in \mathcal{X}^m$ along with their assignment to labeled bags

$$\{(\mathcal{B}_1, z_1), \ldots, (\mathcal{B}_n, z_n)\} \in (\mathcal{P} \times \mathcal{Y})^n$$

where $\mathcal{P}$ is a set of all partitions [1] of indices $\{1, \ldots, m\}$. The i-th bag is a set of flow features $\{\boldsymbol{x}_j \mid j \in \mathcal{B}_i\}$ label by $z_i \in \mathcal{Y}$. The weakly annotated data $\mathcal{D}_{\rm bag}$ carry a partial information about the completely annotated data $\mathcal{D}_{\rm cmp}$. In particular, we assume that:

1. The flow features $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ in $\mathcal{D}_{\rm cmp}$ and $\mathcal{D}_{\rm bag}$ are the same.
2. The negative bags contain just a single instance and its label is correct, that is, $z_i = -1$ implies $|\mathcal{B}_i| = 1$ and $y_i = -1$.
3. The positive bags have a variable size and at least one instance is positive, that is, $z_i = +1$ implies $\exists j \in \mathcal{B}_i$ such that $y_j = +1$.

Our ultimate goal is to construct the Neyman-Pearson detector (see e.g. [12]) $h^* \in \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ which attains the minimal false negative rate

$$\mathrm{FN}(h) = \mathbb{E}_{p(\boldsymbol{x}|y=+1)}[h(\boldsymbol{x}) = -1]$$

among all detectors with the false positive rate

$$\mathrm{FP}(h) = \mathbb{E}_{p(\boldsymbol{x}|y=-1)}[h(\boldsymbol{x}) = +1]$$

not higher than a prescribed threshold $\beta > 0$. That is, we want to find h^* such that $\mathrm{FP}(h^*) \leq \beta$ and

$$\mathrm{FN}(h^*) = \inf_{h \in \mathcal{H}} \mathrm{FN}(h) \quad \text{s.t.} \quad \mathrm{FP}(h) \leq \beta\,. \tag{1}$$

In practice it can be more convenient to solve an equivalent problem $\inf_{h \in \mathcal{H}} \big[\mathrm{FN}(h) + \beta' \mathrm{FP}(h)\big]$ where β' is the Lagrange multiplier of (1) whose value depends on β.

The Neyman-Pearson problem (1) cannot be solved directly since the distribution $p(\boldsymbol{x}, y)$ is unknown and hence also the key quantities $\mathrm{FN}(h)$ and $\mathrm{FP}(h)$

[1] A partition of $\{1, \ldots, m\}$ is a sequence of sets $\mathcal{B}_1, \ldots, \mathcal{B}_n$ such that $\cup_{i=1}^n \mathcal{B}_i = \{1, \ldots, m\}$, $\cap_{i=1}^n \mathcal{B} = \emptyset$ and $\mathcal{B}_i \neq \emptyset$, $\forall i \in \{1, \ldots, n\}$.

cannot be computed. We use the weakly annotated data $\mathcal{D}_{\text{bag}}$ to solve the problem approximately via the empirical risk minimization approach. To this end, we have to concretize the hypothesis space $\mathcal{H}$ and to approximate $\text{FN}(h)$ and $\text{FP}(h)$ by empirical estimates computed from the weakly annotated data $\mathcal{D}_{\text{bag}}$:

– We consider the hypothesis space $\mathcal{H}$ composed of the linear decision rules

$$h(\boldsymbol{x}; \boldsymbol{w}, w_0) = \begin{cases} +1 \text{ if } \langle \boldsymbol{x}, \boldsymbol{w} \rangle + w_0 \geq 0\,, \\ -1 \text{ if } \langle \boldsymbol{x}, \boldsymbol{w} \rangle + w_0 < 0\,, \end{cases} \tag{2}$$

parametrized by $\boldsymbol{w} \in \mathbb{R}^d$ and $w_0 \in \mathbb{R}$.

– The number of false positives is approximated by

$$\overline{\text{FP}}(\boldsymbol{w}, w_0) = \sum_{i \in \mathcal{I}_-} \max\left\{0, 1 + \langle \boldsymbol{w}, \boldsymbol{x}_{\mathcal{B}_i} \rangle + w_0\right\}$$

where $\mathcal{I}_- = \{i \in \{1, \ldots, n\} \mid z_i = -1\}$ are indices of negatively labeled bags. Recall that the negative bags contain just a single instance, hence $\boldsymbol{x}_{\mathcal{B}_i}$ denotes the single $\boldsymbol{x}_j$, $j \in \mathcal{B}_i$. It is seen that $\overline{\text{FP}}(\boldsymbol{w}, w_0)$ is a convex upper bound of the number of false positives which the linear rule with parameters $(\boldsymbol{w}, w_0)$ makes on the completely annotated data $\mathcal{D}_{\text{cmp}}$.

– The number of false negatives is approximated by

$$\overline{\text{FN}}(\boldsymbol{w}, w_0) = \sum_{i \in \mathcal{I}_+} \max\left\{0, 1 - w_0 - \max_{j \in \mathcal{B}_i} \langle \boldsymbol{w}, \boldsymbol{x}_j \rangle\right\} \tag{3}$$

where $\mathcal{I}_+ = \{i \in \{1, \ldots, n\} \mid z_i = +1\}$ are the indices of the positive bags. Let

$$\overline{\text{FN}}_{\text{optim}}(\boldsymbol{w}, w_0) = \sum_{i \in \mathcal{I}_+} [\![h(\boldsymbol{x}_j; \boldsymbol{w}, w_0) = -1\,, \forall j \in \mathcal{B}_i]\!]$$

be the most optimistic estimate (that is, the minimal possible) of the number of false negatives made by the linear rule with the parameters $(\boldsymbol{w}, w_0)$ on the completely annotated data $\mathcal{D}_{\text{cmp}}$. It is seen that $\overline{\text{FN}}(\boldsymbol{w}, w_0)$ is an upper bound of $\overline{\text{FN}}_{\text{optim}}(\boldsymbol{w}, w_0)$ obtained by replacing the step-function with the hinge loss.

With these approximations, we formulate learning of the Neyman-Pearson detector as the following optimization problem

$$(\boldsymbol{w}^*, w_0^*) = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^d, w_0 \in \mathbb{R}} \left[\alpha \cdot \overline{\text{FP}}(\boldsymbol{w}, w_0) + (1 - \alpha) \cdot \overline{\text{FN}}(\boldsymbol{w}, w_0) \right]\,, \tag{4}$$

where $\alpha \in \mathbb{R}_{++}$ is a cost factor used to tune the trade-off between the number of false negatives and false positives. The optimization problem (4) is not convex due to the term $\overline{\text{FP}}(\boldsymbol{w}, w_0)$. We solve the problem (4) approximately by the averaged stochastic gradient descent described in the next section.

Note that the task (4) is a straightforward modification of the Multiple-Instance Support Vector Machines (mi-SVM) algorithm [5]. The original mi-SVM optimizes the classification error (meaning that α is fixed to $\frac{1}{2}$), the objective function contains an additional regularization term $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ used to prevent overfitting and, finaly, the negative bags can contain more than a single instance. We dropped the regularization because the chance of over-fitting is very low in our case. In particular, the ratio of the number of examples and the parameters to be learned is $m/d > 400$.

3.1 Averaged Stochastic Gradient Descent

We formulated learning of the Neyman-Pearson detector as a non-convex optimization problem (4). In this section we describe a simple solver based on the Averaged Stochastic Gradient Descent algorithm [13]. For the sake of simplicity, we rewrite (4) as

$$\boldsymbol{v}^* = \operatorname*{argmin}_{\boldsymbol{v}\in\mathbb{R}^{d+1}} \sum_{i=1}^{n} r_i(\boldsymbol{v}) \tag{5}$$

where

$$r_i(\boldsymbol{v}) = \begin{cases} (1-\alpha)\max\left\{0, 1 - \max\limits_{j\in\mathcal{B}_i}\langle \boldsymbol{v}, \tilde{\boldsymbol{x}}_j\rangle\right\} & \text{if } z_i = +1\,, \\ \alpha\max\left\{0, 1 + \langle \boldsymbol{v}, \tilde{\boldsymbol{x}}_{\mathcal{B}_i}\rangle\right\} & \text{if } z_i = -1\,, \end{cases}$$

$\boldsymbol{v} = (\boldsymbol{w}, w_0)$ and $\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i, 1)$. The SGD algorithm approximates the sub-gradient of the objective of the task (5) by the sub-gradient of a randomly selected term $r_i(\boldsymbol{v})$. A sub-gradient of $r_i(\boldsymbol{v})$ can be computed as $r_i'(\boldsymbol{v}) = \lambda_i \tilde{\boldsymbol{x}}_{i^*}$ where $i^* = \operatorname{argmax}_{j\in\mathcal{B}_i}\langle \boldsymbol{w}, \tilde{\boldsymbol{x}}_j\rangle$ and

$$\lambda_i = \begin{cases} \alpha - 1 & \text{if } z_i = +1 \quad \text{and} \quad \langle \boldsymbol{w}, \tilde{\boldsymbol{x}}_{i^*}\rangle + w_0 < 1\,, \\ \alpha & \text{if } z_i = -1 \quad \text{and} \quad \langle \boldsymbol{w}, \tilde{\boldsymbol{x}}_{i^*}\rangle + w_0 > -1\,, \\ 0 & \text{otherwise.} \end{cases}$$

A pseudo-code of the Averaged SGD solver is summarized in Algorithm 1. The ASGD solver has two free parameters: the fixed step-size γ and the number of epochs E. There is no generic theory how to set these parameters optimally. Hence, their values have to be tuned on particular data.

3.2 Baseline SVM detector

We use the standard binary SVM classifier as the baseline solution [14]. A simple workaround when dealing with the weak annotations is to consider all instances in the positive bags to be positive. Given the training bags $\mathcal{D}_{\text{bag}}$ we define the sets $\mathcal{J}_+ = \cup_{j\in\mathcal{I}_+}\mathcal{B}_j$ and $\mathcal{J}_- = \cup_{j\in\mathcal{I}_-}\mathcal{B}_j$ containing indices of all positive and all negative flows, respectively. We learn the SVM detector by solving the following

Algorithm 1 Averaged SGD

```
Require: cost factor $\alpha > 0$, number of epochs $E > 0$, step-size $\gamma > 0$
Ensure: vector $\overline{\boldsymbol{v}}^t$ approximately solving the problem (5)
1: randomly set $\boldsymbol{v}^0 \in \mathbb{R}^d$ and $\overline{\boldsymbol{v}}^0 \leftarrow \boldsymbol{v}^0$, $t \leftarrow 0$
2: for epoch in $\{1, \dots, E\}$ do
3:     for $i$ in randperm($n$) do
4:         $\boldsymbol{v}^{t+1} \leftarrow \boldsymbol{v}^t - \gamma \boldsymbol{r}'_i(\boldsymbol{v}^t)$
5:         $\overline{\boldsymbol{v}}^{t+1} \leftarrow \frac{t-1}{t}\overline{\boldsymbol{v}}^t + \frac{1}{t}\boldsymbol{v}^{t+1}$
6:         $t \leftarrow t + 1$
7:     end for
8: end for
```

convex program

$$(\boldsymbol{w}^*, w_0^*) = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d, w_0\in\mathbb{R}} \Bigg[\frac{1}{2}\|\boldsymbol{w}\|^2 + C\cdot(1-\alpha)\sum_{i\in\mathcal{J}_+}\max\Big\{0, 1 - \langle\boldsymbol{w}, \boldsymbol{x}_i\rangle - w_0\Big\} + C\cdot\alpha\sum_{i\in\mathcal{J}_-}\max\Big\{0, 1 + \langle\boldsymbol{w}, \boldsymbol{x}_i\rangle + w_0\Big\}\Bigg] \tag{6}$$

where $\alpha \in \mathbb{R}_{++}$ is a cost factor used to tune the trade-off between the number of the false negatives and the false positives. The constant $C \in \mathbb{R}_{++}$ steers the strength of the regularization.

4 Specification of the Datasets

This Section provides detailed description of the datasets and features we used in the experimental evaluation. The datasets are divided into three parts: training, validation, and testing. All datasets represent real network traffic from 14 days of a large international company (80,000 seats) in form of proxy logs. These logs capture HTTP/HTTPS network communication and contain flows, where one flow represents one communication between a user and a server. More specifically, one flow is a group of packets with the same source and destination IP address, source and destination port, and protocol. As flows from the proxy logs are bidirectional, both directions of a communication are included into each flow.

A flow consists of the following fields: user name, source IP address, destination IP address, source port, destination port, protocol, number of bytes transferred from client to server and from server to client, flow duration, timestamp, user agent, URL, referer, MIME-Type, and HTTP status. The most informative field is URL, which can be decomposed further into 7 parts as illustrated in Figure 1. From the flow fields mentioned above, we extracted more than 317 features listed in Table 1. Features from the right column are extracted from all URL parts, including URL itself and referer.

Flows were grouped into bags, where each bag contains flows with the same user (or source IP) and the same second-level domain. Thus, each bag represents communication of a user with a particular domain. As the datasets were

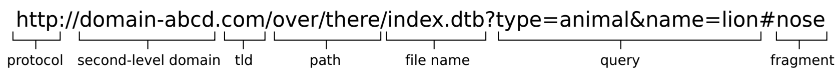
http://domain-abcd.com/over/there/index.dtb?type=animal&name=lion#nose
protocol
second-level domain
tld
path
file name
query
fragment

Fig. 1: URL decomposition into seven parts.

Features	Features extracted from all URL parts & referer
duration	length
HTTP status	digit ratio
is URL encrypted	lower case ratio
is protocol HTTPS	upper case ratio
number of bytes up	vowel changes ratio
number of bytes down	has repetition of '&' and '='
is URL in ASCII	starts with number
client port number	number of non-base64 characters
server port number	has a special character (one feature per character)
user agent length	max length of consonant stream
MIME-Type length	max length of vowel stream
number of '/' in path	max length of lower case stream
number of '/' in query	max length of upper case stream
number of '/' in referer	max length of digit stream
is second-level domain rawIP	ratio of a character with max occurrence

Table 1: List of features extracted from proxy logs. Features from the right column are extracted from all URL parts, including URL itself and referer.

originally unlabeled, we used available blacklists and other malware feeds from Collective Intelligence Framework (CIF) [7] to add positive labels to the training dataset. All bags with domains marked as malicious in CIF were labeled as positive. Negative samples were acquired from the list of popular domains (Alexa top 250,000 domains [2]). Bags with domains that were not in CIF nor in Alexa were discarded from the training set. Note that some popular domains are used very frequently, which may outweigh the importance of rarely-used domains. For this reason, each domain has at most 1000 flows in the training set. Flows with domains which appear in the positive training samples are removed from the testing dataset. We also removed all flows with second-level domains in IP (4-tuple digit) format (e.g. 192.168.0.0) due to the lack of negative samples of this type. Summary of the datasets is described in Table 2.

Each flow is described by 317 real valued features described above. The linear decision rule (2) operates on a binary representation of these features which is created as follows. Range of values of each feature observed in the training data is split into $p = 8$ bins of equal size. Each real valued feature is then represented as a binary vector with 8 elements all set to zero but one encoding the active bin. Stacking the binary vectors for all 317 features gives the final feature representation $\boldsymbol{x} \in \{0, 1\}^d$ with dimension $d = 317 \cdot 8 = 2536$.

	Training Samples		Validation Samples	Test Samples
	Positive	Negative		
Number of Flows	21,873	999,819	1,943,980	9,696,453
Number of Bags	3,918	999,819	—	—
Number of Domains	207	38,463	46,970	45,046

Table 2: Summary of datasets used for training, validation, and testing.

5 Experiments

In this section we empirically evaluate detectors of malicious communication learned from weakly labeled data that were described in Section 4. We compare two detectors learned from the same data by different methods:

1. SVM detector learned by solving the problem (6) is used as a baseline. This method considers all instances in the positive bags to be positive and similarly for instances in the negative bags. To solve (6) we used an open-source implementation of the optimized cutting plane solver [8] available at: `http://cmp.felk.cvut.cz/~xfrancv/ocas/html/index.html`.
2. MIL detector learned by solving the problem (4). We solved (4) by the ASGD Algorithm 1 implemented in C++. We found empirically that setting the step-size $\gamma = \frac{1}{n}$ (where n is the number of training bags) and the number of epochs $E = 100$ works consistently well on our data. The results also depend on the initial estimate $\boldsymbol{v}^0$. Hence, for each data we run the ASGD algorithm 10 times from randomly generated $\boldsymbol{v}^0$ and selected the result with the smallest value of the training objective.

The source data contain only weak labels. The model selection and the final evaluation of the detectors require the ground truth labels for a subset of flows from the validation and the testing subset. The ground truth labels are obtained via submitting the flows' URL to the VirusTotal service [3]. For each submitted URL, the VirusTotal provides a report containing analysis of a set of URL scanners. The number of scanners at the time of evaluation was 62. The report is summarized by the number of positive hits, that is, the number of scanners which marked the URL as malicious. If at least three scanners mark the URL as malicious the flow is labeled as the true positive.

5.1 Main results

Model selection. We learned the MIL detector on the training bags for different values of the cost factor α which is used to tune the ratio of the false positives and the false negatives. Each learned detector is evaluated on the validation data by computing the number of true positive flows (denoted as $\mathrm{tp}_{\mathrm{fp}=50}$) obtained by the detector with the decision threshold set to make the number of false positive flows on the validation data equal to 50. We also recorded the number of hits (denoted as $\text{hits}_{\text{fp}=50}$) accumulated for the flows predicted to be positive by the same detector. This evaluation procedure requires ground true labels for the first $\text{p} = 50 + \text{tp}_{\text{fp}=50}$ flows with the highest decision score $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$. The detector with the maximal validation $\text{tp}_{\text{fp}=50}$ was selected as the final model. If more detectors have the same $\text{tp}_{\text{fp}=50}$, which happened in our case, the detector with the maximal $\text{hits}_{\text{fp}=50}$ is selected among them. The same model selection procedure was used for the SVM detector in which case, however, we also varied the regularization constant. The results of the model selection procedure are summarized in Table 3. In the sake of space we show results only for a subset of values $(\boldsymbol{\alpha}, C)$ selected around the best setting.

Evaluation on test data. The best MIL and SVM detectors were then evaluated on the test flows. We applied each detector on all test flows and selected top 150 instances with the highest decision score. We used the VirusTotal to obtain the ground truth labels for the selected flows. In Figure 2 we show the number of true positives and the precision as a function of increasing decision threshold. In the top 150 instances selected by the MIL detector out of 9,696,453 testing flows 22 are true positives while the baseline SVM detector found just 6 true positives. The precision of the MIL detector in the analyzed region varies from 1 to 0.14. The maximal precision attained by the SVM detector in the same region is 0.08.

We also evaluated the detectors in terms of the number of hits being the finer annotation used to define the ground true labels. Recall that the flow with the number of hits greater than 2 is marked as the true positive. The results are presented in Figure 3. In particular, we show the number of accumulated hits as a function of the detected flows. We also show the histogram of the number of hits per instance measured on the first 50 flows. The highest number of flows in the top 50 instances returned by both detectors has only one hit. However, the second most frequent flows have no hit in the case of SVM but 3 hits in the case of the MIL detector.

5.2 Why does MIL work better than SVM?

In this section we provide some intuition why the SVM detector ignoring the bag annotation performs worse than the MIL detector. The MIL algorithm defined by problem (4) minimizes a weighted sum of errors made by the detector on the negative and the positive bags. The errors on the positive bags are expressed by the function $\overline{\text{FN}}(\boldsymbol{w}, w_0)$ defined in equation (3). It is seen from (3) that the error of each positive bag is determined by a single instance which has the maximal distance from the decision hyperplane. Removal of the other non-active instances from the training set would not change the solution. Hence the MIL algorithm can be seen as two stage procedure though the stages are executed simultaneously. First, a new filtered training set is created by copying all instances from the negative bags and picking a single instance from each positive bag. Second, a supervised SVM algorithm is applied on the filtered training set which contains only bags of size one. Figure 4 visualizes the original and the filtered distribution of the training data projected on the normal vector of the

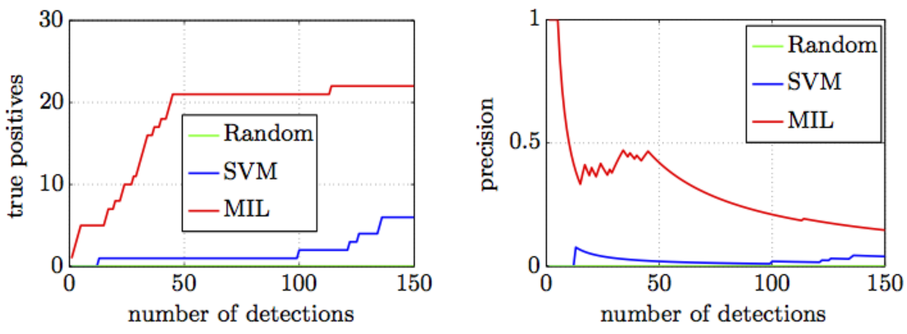

Fig. 2: The figures present results obtained on the first 150 test flows with the highest decision score computed by the MIL and the SVM detector. The left figure shows the number of true positives and the right figure the precision of the detectors as a function of the number of detected flows. We also show results for a baseline detector selecting the flows randomly.

decision hyperplane of the MIL detector. It is seen that the filtered distribution of the positive instances is significantly more peaky than the original one. The shape of the filtered distribution also looks smoother and better separable form the negative distribution.

The SVM and MIL algorithm use only a subset of the examples to define the decision rule. Namely, the *support vectors* which are the training examples with the signed distance to the hyperplane not higher than 1. We projected the $d = 2,536$ dimensional training data onto 2D space and displayed the support vectors selected from the positive examples by the SVM and the MIL algorithms. The 2D coordinates are obtained by projecting the original data onto the normal vector $\boldsymbol{w}$ of the decision hyperplane (x-axis) and the major principal component computed in the subspace orthogonal to $\boldsymbol{w}$ (y-axis). The visualization is shown in Figure 5. It is seen that the SVM detector uses a large set of positive support vectors heavily overlapping with the negative instances many of which probably belong to the negative class. On the other hand, the MIL algorithm ignores a large number of these likely negative instances contained in the positive bags.

6 Conclusions

We presented a system that can learn a reliable detector of malware communication using only weakly labeled data that can be created from publicly available blacklists and security reports. Relaxing the labeling requirement is important due to the constantly changing malware and due to the challenges in forensic analysis caused by increasing complexity of security threats. Our learning algorithm uses bags of proxy logs labeled and grouped according to the network domain. The resulting detector automatically assigns malicious or legitimate

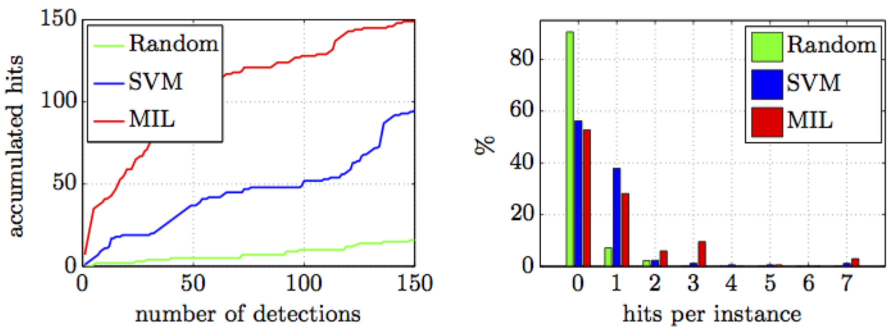

Fig. 3: The left figure shows the number of accumulated hits with respect to the number of flows selected by the MIL and the SVM detector. The right figure shows a histogram of the number of hits computed for the first 50 flows with the highest decision score. The flows with the number of hits higher than 2 are the true positives. We also show results for a baseline detector selecting the flows randomly.

label to each proxy log. We have shown that the use Multiple Instance Learning (MIL) framework can effectively select the most reliable samples from the positive bags necessary to define the decision boundary between malicious and legitimate flows. This improves the results compared to training a baseline SVM detector trained on individual flows rather than bags. The result follows our intuition that not all samples in the positive bags are malicious. We have shown that the detector generalizes well to find new malicious traffic which was not marked by the blacklists.

Our ongoing work focuses on further improvements to the MIL framework to better adapt to the weak labels provided by the bags. One such improvement is to focus on more than the best suitable positive sample from the bag to define the decision boundary. Another improvement could follow from a normalization mechanism that takes into account the size of the bags. Finally, there are vast amounts of unlabeled data that can further improve the training and the final detector performance.

Acknowledgments

The research was supported by CISCO grant number 8301351C001 and the Grant Agency of the Czech Republic under Project P202/12/2071.

References

1. Cisco 2014 annual security report. `http://www.cisco.com/web/offers/lp/2014-annual-security-report/index.html`.

MIL detector

	training			validation		testing	
α	fn	fp	fn bags	$\text{tp}_{\text{fp}=50}$	$\text{hits}_{\text{fp}=50}$	$\text{tp}_{\text{fp}=50}$	$\text{hits}_{\text{fp}=50}$
0.8	4085 (0.19)	10 (1^{-5})	313 (0.08)	8	50	—	—
0.9	4326 (0.20)	3 (3^{-6})	354 (0.09)	9	65	—	—
0.95	4613 (0.21)	1 (1^{-6})	391 (0.10)	9	97	21	118
0.99	5938 (0.27)	0 (0)	530 (0.14)	9	97	—	—

SVM detector

		training		validation		testing	
C	α	fn	fp	$\text{tp}_{\text{fp}=50}$	$\text{hits}_{\text{fp}=50}$	$\text{tp}_{\text{fp}=50}$	$\text{hits}_{\text{fp}=50}$
1000	0.6	4020 (0.18)	158 (2e-4)	2	34	—	—
	0.7	5539 (0.25)	158 (2e-4)	2	29	—	—
	0.8	5602 (0.26)	158 (2e-4)	4	42	1	37
	0.9	6451 (0.29)	57 (6e-5)	3	33	—	—
10000	0.6	2068 (0.09)	158 (2e-4)	0	21	—	—
	0.7	2244 (0.10)	153 (2e-4)	0	27	—	—
	0.8	2723 (0.12)	148 (1e-4)	0	35	—	—
	0.9	3736 (0.17)	79 (8e-5)	1	31	—	—

Table 3: Summary of the errors on the training, validation and testing data for the MIL (upper table) and SVM (bottom table) detector. Each row corresponds to a detector trained with different value of the cost factor α and the regularization constant in case of the SVM. We show the number of false negatives flows fn, the number of false positive flows fp and for the MIL detector also the number of false negative bags evaluated on the training data (the value in brackets are the corresponding rates). Note that fn is at the same time the number of false negative bags because they contain a single instance each. For the validation and test data we show the number of true positives $\text{tp}_{\text{fp}=50}$, and the number of accumulated hits $\text{hits}_{\text{fp}=50}$, computed on the flows returned by the detector with the number of false positives fp set to 50. The same statistics were computed on the test data the best detector selected based on the validation results.

2. List of 1 million top web sites. `http://www.alexa.com`.
3. VirusTotal service. `https://www.virustotal.com`.
4. Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-phishing Working Groups 2Nd Annual eCrime Researchers Summit*, eCrime '07, pages 60–69, New York, NY, USA, 2007. ACM.
5. Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Proc. of Neural Information Processing Systems*, 2002.
6. Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of SIGIR*, Amsterdam, Netherlands, July 2007. ACM.
7. Greg Farnham and Kees Leune. Tools and standards for cyber threat intelligence projects. Technical report, SANS Institute InfoSec Reading Room, 10 2013.

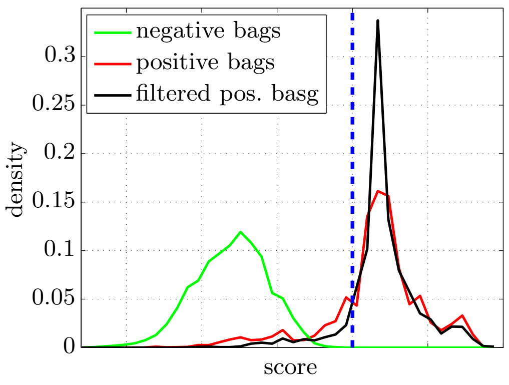

Fig. 4: A distribution of the training instances in the positive (red) and the negative (green) bags projected onto the normal vector $\boldsymbol{w}$ of the decision hyperplane of the MIL detector. The black curve corresponds to the distribution of the instances from the positive bags with the maximal distance to the decision hyperplane. The dashed blue line marks the decision hyperplane of the MIL detector.

8. Vojtech Franc and Soeren Sonnenburg. Optimized cutting plane algorithm for support vector machines. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 320–327, New York, USA, July 2008. ACM.
9. Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.
10. Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious URLs: an application of large-scale online learning. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 681–688. ACM, 2009.
11. Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 26–26, Berkeley, CA, USA, 2010. USENIX Association.
12. Michail I. Schlesinger and Václav Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
13. Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proc. of International Conference on Machine Learning*, 2012.
14. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, 1995.

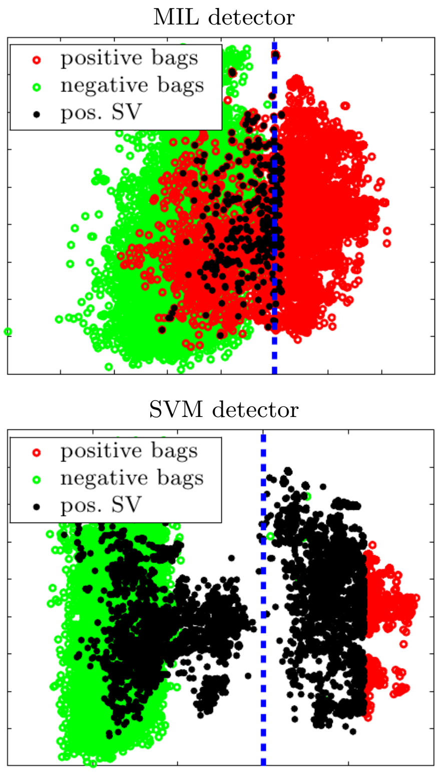

Fig. 5: A distribution of the training instances projected onto the 2D space and the support vectors selected from the positive bags by the MIL algorithm (top) and the SVM algorithm (bottom). The x-axis is a projection of the original data onto the normal vector $\boldsymbol{w}$ of the decision hyperplane of a corresponding detector and the y-axis is a projection onto the major principal component in the subspace orthogonal to $\boldsymbol{w}$. The dashed blue line shows the decision hyperplane.