To appear in Proceedings of the Internet Measurement Conference, November 2009

ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors

Benjamin I. P. Rubinstein[1] Blaine Nelson[1] Ling Huang[2] Anthony D. Joseph[1,2]
Shing-hon Lau[1] Satish Rao[1] Nina Taft[2] J. D. Tygar[1]

[1]Computer Science Division, University of California, Berkeley [2]Intel Labs Berkeley

ABSTRACT

Statistical machine learning techniques have recently garnered increased popularity as a means to improve network design and security. For intrusion detection, such methods build a model for normal behavior from training data and detect attacks as deviations from that model. This process invites adversaries to manipulate the training data so that the learned model fails to detect subsequent attacks.

We evaluate poisoning techniques and develop a defense, in the context of a particular anomaly detector—namely the PCA-subspace method for detecting anomalies in backbone networks. For three poisoning schemes, we show how attackers can substantially increase their chance of successfully evading detection by only adding moderate amounts of poisoned data. Moreover such poisoning throws off the balance between false positives and false negatives thereby dramatically reducing the efficacy of the detector.

To combat these poisoning activities, we propose an antidote based on techniques from robust statistics and present a new robust PCA-based detector. Poisoning has little effect on the robust model, whereas it significantly distorts the model produced by the original PCA method. Our technique substantially reduces the effectiveness of poisoning for a variety of scenarios and indeed maintains a significantly better balance between false positives and false negatives than the original method when under attack.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—Security and Protection; C.4 [**Performance of Systems**]: Modeling Techniques; I.2.6 [**Artificial Intelligence**]: Learning; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Measurement, Performance, Security

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

Keywords

Network Traffic Analysis, Principal Components Analysis, Adversarial Learning, Robust Statistics

1. INTRODUCTION

Statistical machine learning (SML) techniques are increasingly being used as tools for analyzing and improving network design and performance. They have been applied to a variety of problems such as enterprise network fault diagnosis [1, 5, 14], email spam filtering [24, 27], worm detection [25], and intrusion detection [16, 30, 33], as well as many others. These solutions draw upon a variety of techniques from the SML domain including Singular Value Decomposition, clustering, Bayesian inference, spectral analysis, maximum-margin classification, etc. In many scenarios, these approaches have been demonstrated to perform well. Many of these SML techniques include a learning phase during which a model is trained using collected data. Such techniques have a serious vulnerability, namely they are susceptible to adversaries who purposefully inject malicious data during the phases of data-collection and model-building. The intent of such poisoning is to direct an SML algorithm to learn the wrong model; if adversaries influence detectors to learn the wrong underlying model or normality, then such detectors are unable to properly identify abnormal activities. Poisoning is particularly incentivized when SML techniques are used as defenses against cybercrime threats.

Other than a few efforts [10, 31, 32], this type of vulnerability has not been extensively explored by those who apply SML techniques to networking and systems problems. Applied machine learning researchers have started to address these problems by focusing on adversarial training of specific algorithms [2, 8, 22]. The learning theory community has focused on online learning [4], where data is selected by an adversary with complete knowledge of the learner, and has developed efficient algorithms with strong guarantees. However, the simplifying assumption of all data being produced by an omniscient adversary does not hold for many practical threat models. Given the increasing popularity of applying SML techniques to networking problems, we believe exploring adversarial learning with realistic threat models is important and timely.

In this paper we study both poisoning strategies and defenses in the context of a particular anomaly detector, namely the PCA-subspace method [16], based on Principal Component Analysis (PCA). This technique has received a large amount of attention, leading to extensions [15, 17, 18], and inspiring related research [3, 12, 20, 28, 33]. We consider

an adversary who knows that an ISP is using a PCA-based anomaly detector. The adversary's aim is to evade future detection by poisoning the training data so that the detector learns a distorted set of principal components. Because PCA solely focuses on link traffic covariance, we explore poisoning schemes that add *chaff* (additional traffic) into the network to increase the variance of network traffic. The end goal of the attacker is to increase the false negative rate of a detector, which corresponds to his evasion success rate. In our abstract [29], we illustrated that simple poisoning strategies can improve an adversary's ability to evade detection. Our first contribution in this paper is a detailed analysis of how adversaries subvert the learning process. We explore a range of poisoning strategies in which the attacker's knowledge about the network traffic state is varied, and in which the attacker's time horizon (length of poisoning episode) is varied. (We use the words 'attackers' and 'adversaries' interchangeably.) Through theoretical analysis of global poisoning tactics, we uncover some simple and effective poisoning strategies for the adversary. In order to gain insights as to why these attacks work, we illustrate their impact on the normal model built by the PCA detector.

Because the networks that SML techniques are used in are non-stationary, the baseline models must be periodically retrained to capture evolving trends in the underlying data. In previous usage scenarios [16, 30], the PCA detector is retrained regularly (e.g., weekly), meaning that attackers could poison PCA slowly over long periods of time; thus poisoning PCA in a more stealthy fashion. By perturbing the principal components gradually, the attacker decreases the chance that the poisoning activity itself is detected. We design such a poisoning scheme, called a *Boiling Frog* scheme, and demonstrate that it can boost the false negative rate as high as the non-stealthy strategies, with far less chaff, albeit over a longer period of time.

Our second contribution is to design a robust defense against this type of poisoning. It is known that PCA can be strongly affected by outliers [28]. However, instead of finding the principal components along directions that maximize variance, robust statistics suggests components that maximize more robust measures of dispersion. It is well known that the median is a more robust measure of location than the mean, in that it is far less sensitive to the influence of outliers. This concept can be extended to robust alternatives to variance such as the Median Absolute Deviation (MAD). Over the past two decades a number of robust PCA algorithms have been developed that maximize MAD instead of variance. Recently the PCA-GRID algorithm was proposed as an efficient method for maximizing MAD without under-estimating variance (a flaw identified in previous solutions) [6]. We adapt PCA-GRID for anomaly detection by combining the method with a new robust cutoff threshold. Instead of modeling the squared prediction error as Gaussian (as in the original PCA method), we model the error using a Laplace distribution. The new threshold was motivated from observations of the residual that show longer tails than exhibited by Gaussian distributions. We call our method that combines PCA-GRID with a Laplace cutoff threshold, ANTIDOTE. The key intuition behind this method is to reduce the effect of outliers and help reject poisonous training data.

Our third contribution is to carry out extensive evaluations of both ANTIDOTE and the original PCA method, in a variety of poisoning situations, and to assess their performance via multiple metrics. To do this, we used traffic matrix data from the Abilene network since many other studies of traffic matrix estimation and anomaly detection have used this data. We show that the original PCA method can be easily compromised by any of our poisoning schemes, with only small amounts of chaff. For moderate amounts of chaff, the PCA detector starts to approach the performance of a random detector. However, ANTIDOTE is dramatically more robust. It outperforms PCA in that i) it more effectively limits the adversary's ability to increase his evasion success; ii) it can reject a larger portion of contaminated training data; and iii) it provides robust protection across nearly all origin-destination flows through a network. The gains of ANTIDOTE for these performance measures are large, especially as the amount of poisoning increases. Most importantly, we demonstrate that ANTIDOTE incurs insignificant shifts in its false negative and false positive performance, compared to PCA, when no poisoning events happen; however when poisoning does occur, the gains of ANTIDOTE over PCA are enormous with respect to both of these traditional performance measures. The PCA method was not designed to be robust. Our results indicate that it is possible to take such useful techniques and bolster their performance under difficult circumstances.

Our study sheds light on the general problem of poisoning SML techniques, in terms of the types of poisoning schemes that can be construed, their impact on detection, and strategies for defense.

Related Work. Several earlier studies examined attacks on specific learning systems for related applications. In [26], the authors describe red herring attacks that weaken polymorphic worm detection systems by poisoning the training data used to build signature-based classifiers. In red herring attacks, the adversary forces the learner to make false negative predictions by including spurious features in positive training examples. Subsequent malicious instances evade detection by excluding these features, now included as conjuncts in the conjunction learned by Polygraph. Venkataraman et al. [31] present lower bounds for learning worm signatures based on red herring attacks and reductions to classic results from Query Learning. While the red herring attacks exploit the Polygraph conjunction learner's tendency to overfit, our poisoning attacks exploit PCA's singular focus on link traffic covariance.

Attacks that increase false negative rates by manipulating the test data have also been explored. The polymorphic blending attacks of Fogla and Lee [10] encrypt malicious traffic so that the traffic is indistinguishable from innocuous traffic to an intrusion detection system. By contrast our variance injection attacks add small amounts of chaff to largely innocuous training traffic to make the traffic appear more like future DoS attacks to be launched post-poisoning. In the email spam filtering domain, Wittel and Wu [32] and Lowd and Meek [22] add *good words*—tokens the filter associates with non-spam messages—so spam messages can evade detection.

Ringberg et al. [28] performed a study of the sensitivities of the PCA method that illustrates how the PCA method can be sensitive to the number of principal components used to describe the normal subspace. This parameter can limit PCA's effectiveness if not properly configured. They also show that routing outages can pollute the normal subspace;

a kind of perturbation to the subspace that is not adversarial. Our work differs in two key ways. First we demonstrate a different type of sensitivity, namely that of data poisoning. This adversarial perturbation can be stealthy and subtle, and is more challenging to circumvent than observable routing outages. Second, [28] focuses on showing the variability in PCA's performance to certain sensitivities, and not on defenses. In our work, we propose a robust defense against a malicious adversary and demonstrate its effectiveness. It is conceivable that the technique we propose could help limit PCA's sensitivity to routing outages, although such a study is beyond the scope of this paper. A recent study [3] showed that the sensitivities observed in [28] come from PCA's inability to capture temporal correlations. They propose to replace PCA by a Karhunen-Loeve expansion. Our study indicates that it would be important to examine, in future work, the data poisoning robustness of this proposal.

2. BACKGROUND

To uncover anomalies, many network anomography detection techniques mine the network-wide traffic matrix, which describes the traffic volume between all pairs of Points-of-Presence (PoP) in a backbone network and contains the collected traffic volume time series for each origin-destination (OD) flow. In this section, we define traffic matrices, present our notation, and summarize the PCA anomaly detection method of Lakhina et al. [16].

2.1 Traffic Matrices and Volume Anomalies

Network link traffic represents the superposition of OD flows. We consider a network with N links and F OD flows and measure traffic on this network over T time intervals. The relationship between link traffic and OD flow traffic is concisely captured in the *routing matrix* $\mathbf{A}$. This matrix is an $N \times F$ matrix such that $\mathbf{A}_{ij} = 1$ if OD flow j passes over link i, and is zero otherwise. If $\mathbf{X}$ is the $T \times F$ traffic matrix (TM) containing the time-series of all OD flows, and if $\mathbf{Y}$ is the $T \times N$ link TM containing the time-series of all links, then $\mathbf{Y} = \mathbf{X}\mathbf{A}^\top$. We denote the t^{th} row of $\mathbf{Y}$ as $\mathbf{y}(t) = \mathbf{Y}_{t,\bullet}$ (the vector of N link traffic measurements at time t), and the original traffic along a *source link*, S by $\mathbf{y}_S(t)$. We denote column f of routing matrix $\mathbf{A}$ by $\mathbf{A}_f$.

We consider the problem of detecting *OD flow volume anomalies* across a top-tier network by observing link traffic volumes. Anomalous flow volumes are unusual traffic load levels in a network caused by anomalies such as Denial of Service (DoS) attacks, Distributed DoS attacks, flash crowds, device failures, misconfigurations, and so on. DoS attacks serve as the canonical example attack in this paper.

2.2 Subspace Method for Anomaly Detection

We briefly summarize the PCA-based anomaly detector introduced by Lakhina et al. [16]. The authors observed high levels of traffic aggregation on ISP backbone links cause OD flow volume anomalies to often go unnoticed because they are buried within normal traffic patterns. They also observe that although the measured data has high dimensionality, N, normal traffic patterns lie in a subspace of low dimension $K \ll N$. Inferring this normal traffic subspace using PCA (which finds the principal traffic components) makes it easier to identify volume anomalies in the remaining abnormal subspace. For the Abilene (Internet2 backbone) network, most variance can be captured by the first $K = 4$ principal components.

PCA is a dimensionality reduction method that chooses K orthogonal *principal components* to form a K-dimensional subspace capturing maximal variance in the data. Let $\bar{\mathbf{Y}}$ be the centered link traffic matrix, i.e., with each column of $\mathbf{Y}$ is translated to have zero mean. The k^{th} principal component is computed as

$$\mathbf{v}_k = \arg\max_{\mathbf{w}:\|\mathbf{w}\|=1} \left\| \bar{\mathbf{Y}} \left(\mathbb{I} - \sum_{i=1}^{k-1} \mathbf{v}_i \mathbf{v}_i^\top \right) \mathbf{w} \right\| . \quad (1)$$

The resulting K-dimensional subspace spanned by the first K principal components $\mathbf{V}_{1:K} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K]$ is the *normal* traffic subspace $\mathcal{S}_n$ and has a projection matrix $\mathbf{P}_n = \mathbf{V}_{1:K}\mathbf{V}_{1:K}^\top$. The residual $(N-K)$-dimensional subspace is spanned by the remaining principal components $\mathbf{V}_{K+1:N} = [\mathbf{v}_{K+1}, \mathbf{v}_{K+2}, \ldots, \mathbf{v}_N]$. This space is the *abnormal* traffic subspace $\mathcal{S}_a$ with a corresponding projection matrix $\mathbf{P}_a = \mathbf{V}_{K+1:N}\mathbf{V}_{K+1:N}^\top = \mathbb{I} - \mathbf{P}_n$.

Volume anomalies can be detected by decomposing the link traffic into $\mathbf{y}(t) = \mathbf{y}_n(t) + \mathbf{y}_a(t)$ where $\mathbf{y}_n(t)$ is the modeled normal traffic and $\mathbf{y}_a(t)$ is the residual traffic, corresponding to projecting $\mathbf{y}(t)$ onto $\mathcal{S}_n$ and $\mathcal{S}_a$, respectively. A volume anomaly at time t typically results in a large change to $\mathbf{y}_a(t)$, which can be detected by thresholding the squared prediction error $\|\mathbf{y}_a(t)\|^2$ against Q_β, the Q-statistic at the $1-\beta$ confidence level [13]. That is, the PCA-based detector classifies a link measurement vector as

$$c(\mathbf{y}(t)) = \begin{cases} \text{anomalous}, & \|\mathbf{y}_a(t)\|^2 > Q_\beta \\ \text{innocuous}, & \|\mathbf{y}_a(t)\|^2 \le Q_\beta \end{cases} . \quad (2)$$

While others have explored more efficient distributed variations of this approach [12, 20, 21], we focus on the basic method introduced by Lakhina et al. [16].

3. POISONING STRATEGIES

3.1 The Threat Model

The adversary's goal is to launch a Denial of Service (DoS) attack on some victim and to have the attack traffic successfully cross an ISP's network without being detected. The DoS traffic thus needs to traverse from an ingress point-of-presence (PoP) node to an egress PoP of the ISP. Before launching a DoS attack, the attacker poisons the detector for a period of time, by injecting additional traffic, *chaff*, along the OD flow (i.e., from an ingress PoP to an egress PoP) that he eventually intends to attack. This kind of poisoning activity is possible if the adversary gains control over clients of an ingress PoP or if the adversary compromises a router (or set of routers) within the ingress PoP. For a poisoning strategy, the attacker needs to decide how much chaff to add, and when to do so. These choices are guided by the amount of information available to the attacker.

We consider poisoning strategies in which the attacker has increasing amounts of information at his disposal. The weakest attacker is one that knows nothing about the traffic at the ingress PoP, and adds chaff randomly (called an *uninformed* attack). An intermediate case is when the attacker is partially informed. Here the attacker knows the current volume of traffic on the ingress link(s) that he intends to inject chaff on. Because many networks export SNMP records, an

adversary might intercept this information, or possibly monitor it himself (i.e., in the case of a compromised router). We call this type of poisoning a *locally-informed* attack. Although exported data from routers may be delayed in reaching the adversary, we consider the case of minimal delay in our first study of this topic.

In a third scenario, the attacker is *globally-informed* because his global view over the network enables him to know the traffic levels on all network links. Moreover, we assume this attacker has knowledge of future traffic link levels. (Recall that in the locally-informed scheme, the attacker only knows the *current* traffic volume of a link.) Although these attacker capabilities are very unlikely, we include this in our study in order to understand the limits of variance injection poisoning schemes. Also this scenario serves as a difficult test for our ANTIDOTE technique.

Poisoning strategies can also vary according to the time horizon over which they are carried out. Most studies on the PCA-subspace method use a one week training period, so we assume that PCA is retrained each week. Thus the PCs used in any week m are those learned during week $m-1$ with any detected anomalies removed. Thus for our poisoning attacks, the adversary inserts chaff along the target OD flow throughout the one week training period. We also consider a long-term attack in which the adversary slowly, but increasingly, poisons the principal components over several weeks, by adding small amounts of chaff, in gradually increasing quantities. We call this the *Boiling Frog* poisoning method after the folk tale that one can boil a frog by slowly increasing the water temperature over time[1].

We assume the adversary does not have control over existing traffic (i.e., he cannot delay or discard traffic). Similarly, the adversary cannot submit false SNMP reports to PCA. Such approaches are more conspicuous because the inconsistencies in SNMP reporting from neighboring PoPs could expose the compromised router.

This paper focuses on non-distributed poisoning of DoS detectors. *Distributed* poisoning that aims to evade a DoS detector is also possible; our globally-informed poisoning strategy is an example, as the adversary has control over all network links. We focus on DoS for a two reasons. In our first study on this topic, we aim to solve the basic problem first before tackling a distributed version. Second, we point out that results on evasion via non-distributed poisoning are stronger than distributed poisoning results: the DDoS attacker can monitor and influence many more links than the DoS attacker. Hence a DoS poisoning scenario is usually stealthier than a DDoS one.

For each of these scenarios of different information available to the adversary, we now outline specific poisoning schemes. In each scheme, the adversary decides on the quantity of c_t chaff to add to the target flow time series at a time t. Each strategy has an attack parameter θ, which controls the intensity of the attack. For each scenario, we present only one specific poisoning scheme. We have studied others, but those included here are representative.

[1]Note that there is nothing inherent in the choice of a one-week poisoning period. For a general SML algorithm, our strategies would correspond to poisoning over one training period (whatever its length) or multiple training periods.

3.2 Uninformed Chaff Selection

At each time t, the adversary decides whether or not to inject chaff according to a Bernoulli random variable. If he decides to inject chaff, the amount of chaff added is of size θ, i.e., $c_t = \theta$. This method is independent of the network traffic since our attacker is uninformed. We call this the *Random* scheme.

3.3 Locally-Informed Chaff Selection

The attacker's goal is to increase traffic variance, on which the PCA detector's model is based. In the locally-informed scenario, the attacker knows the volume of traffic in the ingress link he controls, $\mathbf{y}_S(t)$. Hence this scheme elects to only add chaff when the existing traffic is already reasonably large. In particular, we add chaff when the traffic volume on the link exceeds a parameter α (we typically use the mean). The amount of chaff added is $c_t = (\max\{0, \mathbf{y}_S(t) - \alpha\})^{\theta}$. In other words, we take the difference between the link traffic and a parameter α and raise it to θ. In this scheme (called *Add-More-If-Bigger*), the further the traffic is from the average load, the larger the deviation of chaff inserted.

3.4 Globally-Informed Chaff Selection

The globally-informed scheme captures an omnipotent adversary with full knowledge of $\mathbf{Y}$, $\mathbf{A}$, and the future measurements $\tilde{y}_t$, and who is capable of injecting chaff into *any* network flow during training. This latter point is important. In previous poisoning schemes the adversary can only inject chaff along their compromised link, whereas in this scenario, the adversary can inject chaff on any link. We formalize the problem of selecting a link n to poison, and selecting an amount of chaff $\mathbf{C}_{tn}$ as an optimization problem that the adversary solves to maximally increase his chances of evasion. Although these globally-informed capabilities are unrealistic, we include a globally-informed poisoning strategy in order to understand the limits of variance injection methods.

The *PCA Evasion Problem* considers an adversary wishing to launch an undetected DoS attack of volume δ along flow f at time t. If the vector of link volumes at future time t is $\tilde{y}_t$, where the tilde distinguishes this future measurement from past training data $\bar{\mathbf{Y}}$, then the vectors of anomalous DoS volumes are given by $\tilde{y}'_t = \tilde{y}_t + \delta * \mathbf{A}_f$. Denote by $\mathbf{C}$ the matrix of link traffic injected into the network by the adversary during training. Then the PCA-based anomaly detector is trained on altered link traffic matrix $\bar{\mathbf{Y}} + \mathbf{C}$ to produce the mean traffic vector μ, the top K eigenvectors $\mathbf{V}_{1:K}$, and the squared prediction error threshold Q_β. The adversary's objective is to enable as large a DoS attack as possible (maximizing δ) by designing $\mathbf{C}$. The PCA Evasion Problem corresponds to solving the following:

$$\begin{aligned} \max_{\delta\in\mathbb{R},\ \mathbf{C}\in\mathbb{R}^{T\times F}} \quad & \delta \\ \text{s.t.} \quad & (\mu, \mathbf{V}, Q_\beta) = \text{PCA}(\mathbf{Y} + \mathbf{C}) \\ & \left\| \mathbf{V}_{K+1:N}^\top (\tilde{y}'_t - \mu) \right\|_2 \le Q_\beta \\ & \|\mathbf{C}\|_1 \le \theta \qquad \forall t, n \ \ \mathbf{C}_{tn} \ge 0 \ , \end{aligned}$$

where θ is a constant constraining total chaff. The second constraint guarantees evasion by requiring that the contaminated link volumes at time t are classified innocuous (cf. Eq. 2). The remaining constraints upper-bound the total chaff volume by θ and constrain the chaff to be non-negative.

Unfortunately, this optimization is difficult to solve analytically. Thus we construct a relaxed approximation to obtain a tractable analytic solution. We make a few assumptions and derivations[2], and show that the above objective seeks to maximize the attack direction $\mathbf{A}_f$'s projected length in the normal subspace $\max_{\mathbf{C}\in\mathbb{R}^{T\times F}} \left\|\mathbf{V}_{1:K}^\top \mathbf{A}_f\right\|_2$. Next, we restrict our focus to traffic processes that generate spherical k-rank link traffic covariance matrices[3]. This property implies that the eigen-spectrum consists of K ones followed by all zeroes. Such an eigen-spectrum allows us to approximate the top eigenvectors $\mathbf{V}_{1:K}$ in the objective, with the matrix of all eigenvectors weighted by their corresponding eigenvalues $\Sigma\mathbf{V}$. We can thus convert the PCA evasion problem into the following optimization:

$$\begin{aligned} \max_{\mathbf{C}\in\mathbb{R}^{T\times F}} \quad & \left\|(\bar{\mathbf{Y}}+\mathbf{C})\mathbf{A}_f\right\|_2 \\ \text{s.t.} \quad & \|\mathbf{C}\|_1 \le \theta \\ & \forall t,n \;\; \mathbf{C}_{tn} \ge 0 \;\; . \end{aligned} \tag{3}$$

Solutions to this optimization are obtained by a standard Projection Pursuit method from optimization: iteratively take a step in the direction of the objective's gradient and then project onto the feasible set.

These solutions yield an interesting insight. Recall that our adversary is capable of injecting chaff along *any* flow. One could imagine that it might be useful to inject chaff along an OD flow whose traffic dominates the choice of principal components (i.e., an elephant flow), and then send the DoS traffic along a different flow (that possibly shares a subset of links with the poisoned OD flow). However the solutions of Eq. (3) indicates that the best strategy to evade detection is to inject chaff only along the links $\mathbf{A}_f$ associated with the target flow f. This follows from the form of the initializer $\mathbf{C}^{(0)} \propto \bar{\mathbf{Y}}\mathbf{A}_f\mathbf{A}_f^\top$ (obtained from an L_2 relaxation) as well as the form of the projection and gradient steps. In particular, all these objects preserve the property that the solution only injects chaff along the target flow. In fact, the only difference between this globally-informed solution and the locally-informed scheme is that the former uses information about the entire traffic matrix $\mathbf{Y}$ to determine chaff allocation along the flow whereas the latter use only local information.

3.5 Boiling Frog Attacks

Boiling Frog poisoning can use any of the preceding chaff schemes to select c_t. The duration of poisoning is increased as follows. We initially set the attack parameter θ to a small value and then increase it slowly over time. In the first week of the attack, the target flow is injected with chaff generated using parameter θ_1. At the week's end, PCA is retrained on that week's data. Any anomalies detected by PCA during that week are excluded from future training data. This process continues with $\theta_t > \theta_{t-1}$ used for week t. Even though PCA is retrained from scratch each week, the training data includes events not caught by the previous detector. Thus, each successive week will contain additional malicious training data, with the process continuing until the week of the DoS attack, when the adversary stops injecting chaff.

[2]The full proof is ommitted due to space constraints.

[3]While the spherical assumption does not hold in practice, the assumption of low-rank traffic matrices is met by published datasets [16].

4. ANTIDOTE: A ROBUST DEFENSE

For defenses against our attacks on PCA-based anomaly detection we explore techniques from Robust Statistics. Such methods are less sensitive to outliers, and as such are ideal defenses against variance injection schemes that perturb data to increase variance along the target flow. There have been two approaches to make PCA robust: the first computes the principal components as the eigenspectrum of a robust estimate of the covariance matrix [9], while the second approach searches for directions that maximize a robust scale estimate of the data projection. We propose one of the latter methods as a defense against our poisoning. After describing the method, we propose a new threshold statistic that can be used for any PCA-based method including robust PCA. Robust PCA and the new robust Laplace threshold together form a new network-wide traffic anomaly detection method, ANTIDOTE, that is less sensitive to our poisoning attacks.

4.1 Intuition

Fundamentally, to mitigate the effect of poisoning attacks, we need a learning algorithm that is stable in spite of data contamination; i.e., a small amount of data contamination should not dramatically change the model produced by our algorithm. This concept of stability has been studied in the field of Robust Statistics in which *robust* is the formal term used to qualify this notion of stability. In particular, there have been several approaches to developing robust PCA algorithms that construct a low dimensional subspace that captures most of the data's *dispersion*[4] and are stable under data contamination [6, 7, 9, 19, 23].

The robust PCA algorithms we considered search for a unit direction $\mathbf{v}$ whose projections maximize some univariate dispersion measure $\mathrm{S}(\cdot)$; that is,

$$\mathbf{v} \;\in\; \underset{\|\mathbf{a}\|_2=1}{\arg\max}\; \mathrm{S}(\mathbf{Ya}) \;\; . \tag{4}$$

The standard deviation is the dispersion measure used by PCA; i.e., $\mathrm{S}^{\mathrm{SD}}(r_1, r_2, \ldots, r_n) = \left(\frac{1}{n-1}\sum_{i=1}^n r_i - \bar{r}\right)^{1/2}$. However, the standard deviation is sensitive to outliers making PCA non-robust to contamination. Robust PCA algorithms instead use measures of dispersion based on the concept of *robust projection pursuit (RPP)* estimators [19]. As is shown by Li & Chen, RPP estimators achieve the same breakdown points as their dispersion measure (the *breakdown point* is the (asymptotic) fraction of the data an adversary must control in order to arbitrarily change an estimator, and as such is a common measure of statistical robustness) as well as being qualitatively robust; i.e., the estimators are stable.

However, unlike the eigenvector solutions that arise in PCA, there is generally no efficiently computable solution for robust dispersion measures and so these must be approximated. Below, we describe the PCA-GRID algorithm, a successful method for approximating robust PCA subspaces developed by Croux et al. [6]. Among the projection pursuit techniques we tried [7, 23], PCA-GRID proved to be most resilient to our poisoning attacks. It is worth emphasizing that the procedure described in the next section is

[4]Dispersion is an alternative term for variation since the later is often associated with statistical variation. By a dispersion measure we mean a statistic that measures the variability or spread of a variable.

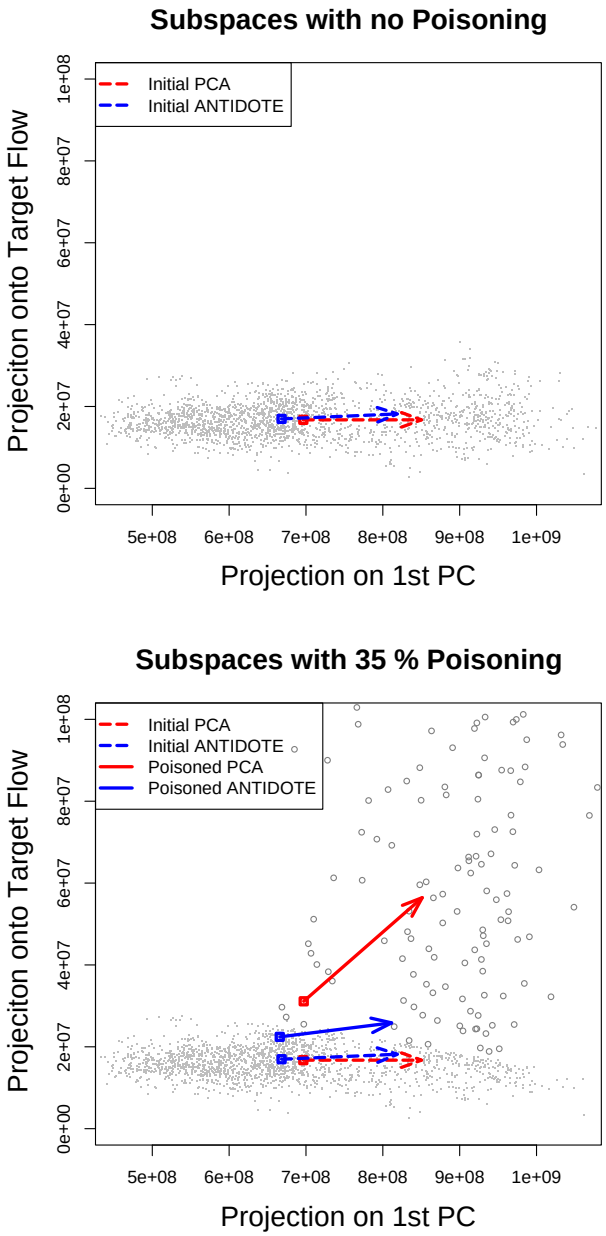

Figure 1: Here the data has been projected into the 2D space spanned by the 1st principal component and the direction of the attack flow #118. The effect on the 1st principal components of PCA and PCA-GRID is shown under a globally informed attack (represented by ∘'s).

simply a technique for approximating a projection pursuit estimator and does not itself contribute to the algorithm's robustness—that robustness comes from the definition of the projection pursuit estimator in Eq. (4).

First, to better understand the efficacy of a robust PCA algorithm, we demonstrate the effect our poisoning techniques have on the PCA algorithm and contrast them with the effect on the PCA-GRID algorithm. In Figure 1, we see the impact of a globally informed poisoning attack on both algorithms. Initially, the data was clustered in an ellipse. In the top plot, we see that both algorithms construct reasonable estimates for the center and first principal component for this data. However, in the bottom plot, we see that a large amount of poisoning dramatically perturbs some of the data and as a result the PCA subspace is dramatically shifted toward the target flow's direction (y-axis). Due to this shift, DoS attacks along the target flow will be less detectable. Meanwhile, the subspace of PCA-GRID is noticeably less affected.

4.2 PCA-GRID

The PCA-GRID algorithm introduced by Croux et al. [6] is a projection pursuit technique as described above. It finds a K-dimensional subspace that approximately maximizes $\mathrm{S}(\cdot)$, a *robust* measure of dispersion, for the data $\mathbf{Y}$ as in Eq. (4). The first step is to specify our robust dispersion measure. We use the *Median Absolute Deviation (MAD)* robust measure of dispersion, over other possible choices for $\mathrm{S}(\cdot)$. For scalars $r_1, \ldots, r_n$ the MAD is defined as

$$\mathrm{S}^{\mathrm{MAD}}(r_1, \ldots, r_n) \quad = \quad \omega \cdot \mathrm{median}\left\{|r_i - \mathrm{median}\{r_j\}|\right\} ,$$

where the coefficient $\omega = 1.486$ ensures asymptotic consistency on normal distributions.

The next step requires choosing an estimate of the data's central location. In PCA, this estimate is simply the mean of the data. However, the mean is not robust, so we center the data using the spatial median instead:

$$\hat{c}(\mathbf{Y}) \quad \in \quad \underset{\mu \in \mathbb{R}^N}{\arg\min} \sum_{i=1}^{n} \|\mathbf{y}_i - \mu\|_2 \quad ,$$

which involves a convex optimization that is efficiently solved (see e.g., [11]).

Given a dispersion measure and location estimate, PCA-GRID finds a (unit) direction $\mathbf{v}$ that is an approximate solution to Eq. (4). The PCA-GRID algorithm uses a grid-search for this task. Namely, suppose we want to find the best candidate between some pair of unit vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ (a 2D search space). The search space is the unit circle parameterized by ϕ as $\mathbf{a}_\phi = \cos(\phi)\mathbf{a}_1 + \sin(\phi)\mathbf{a}_2$ with $\phi \in [-\pi/2, \pi/2]$. The grid search splits the domain of ϕ into a mesh of $Q+1$ candidates $\phi_k = \frac{\pi}{2}\left(\frac{2k}{Q} - 1\right)$, $k = 0, \ldots, Q$. Each candidate vector $\mathbf{a}_{\phi_k}$ is assessed and the one that maximizes $\mathrm{S}(\mathbf{Y}\mathbf{a}_{\phi_k})$ is the approximate maximizer $\hat{\mathbf{a}}$.

To search a more general N-dimensional space, the search iteratively refines its current best candidate $\hat{\mathbf{a}}$ by performing a grid search between $\hat{\mathbf{a}}$ and each of the unit directions $\mathbf{e}_i$. With each iteration, the range of angles considered progressively narrows around $\hat{\mathbf{a}}$ to better explore its neighborhood. This procedure (outlined in Algorithm 1) approximates the direction of maximal dispersion analogous to an eigenvector in PCA.

To find the K-dimensional subspace $\{\mathbf{v}_i \mid \mathbf{v}_i^\top \mathbf{v}_j = \delta_{i,j}\}$ that maximizes the dispersion measure, the GRID-SEARCH is repeated K-times. After each repetition, the data is deflated to remove the dispersion captured by the last direction from the data. This process is detailed in Algorithm 2.

4.3 Robust Laplace Threshold

In addition to the robust PCA-GRID algorithm, we also use a robust estimate for its residual threshold in place of the Q-statistic described in Section 2.2. Using the Q-statistic as

Algorithm 1 GRID-SEARCH($\mathbf{Y}$)

Require: $\mathbf{Y}$ is a $T \times N$ matrix
1: **Let:** $\hat{\mathbf{v}} = \mathbf{e}_1$;
2: **for** $i = 1$ to C **do**
3: **for** $j = 1$ to N **do**
4: **for** $k = 0$ to Q **do**
5: **Let:** $\phi_k = \frac{\pi}{2^i}\left(\frac{2k}{Q} - 1\right)$;
6: **Let:** $\mathbf{a}_{\phi_k} = \cos(\phi_k)\hat{\mathbf{a}} + \sin(\phi_k)\mathbf{e}_j$;
7: **if** $\mathrm{S}(\mathbf{Y}\mathbf{a}_{\phi_k}) > \mathrm{S}(\mathbf{Y}\hat{\mathbf{v}})$ **then**
8: **Assign:** $\hat{\mathbf{v}} \leftarrow \mathbf{a}_{\phi_k}$;
9: **Return:** $\hat{\mathbf{v}}$;

Algorithm 2 PCA-GRID($\mathbf{Y}, K$)

1: Center $\mathbf{Y}$: $\mathbf{Y} \leftarrow \mathbf{Y} - \hat{c}(\mathbf{Y})$;
2: **for** $i = 1$ to K **do**
3: $\mathbf{v}_i \leftarrow$ GRID-SEARCH($\mathbf{Y}$);
4: $\mathbf{Y} \leftarrow$ projection of $\mathbf{Y}$ onto the complement of $\mathbf{v}_i$;
5: **end for**
6: Return subspace centered at $\hat{c}(\mathbf{Y})$ with principal directions $\{\mathbf{v}_i\}_{i=1}^{K}$;

a threshold was motivated by an assumption of normally distributed residuals [13]. However, we found that the residuals for both the PCA and PCA-GRID subspaces were empirically non-normal leading us to conclude that the Q-statistic is a poor choice for our detection threshold. Instead, to account for the outliers and heavy-tailed behavior we observed from our method's residuals, we choose our threshold as the $1-\beta$ quantile of a Laplace distribution fit with robust location and scale parameters. Our solution, ANTIDOTE is the combination of the PCA-GRID algorithm and the Laplace threshold. The non-normality of the residuals has also been recently pointed out in [3].

As with the previous method described in Section 2.2, we select our threshold $Q_{L,\beta}$ as the $1-\beta$ quantile of a parametric distribution fit to the residuals in the training data. However, instead of the normal distribution assumed by the Q-statistic, we use the quantiles of a Laplace distribution specified by a location parameter c and a scale parameter b. Critically, though, instead of using the mean and standard deviation, we robustly fit the distribution's parameters. We estimate c and b from the residuals $\|\mathbf{y}_a(t)\|^2$ using robust consistent estimates of location (median) and scale (MAD)

$$\hat{c} = \text{median}\left(\|\mathbf{y}_a(t)\|^2\right)$$
$$\hat{b} = \frac{1}{\sqrt{2}P^{-1}(0.75)}\,\text{median}\left\{\left|\|\mathbf{y}_a(t)\|^2 - \hat{c}\right|\right\}$$

where $P^{-1}(q)$ is the q^{th} quantile of the standard Laplace distribution. The Laplace quantile function has the form $P^{-1}_{c,b}(q) = c + b \cdot k(q)$ for some $k(q)$. Thus, our threshold only depends linearly on the (robust) estimates $\hat{c}$ and $\hat{b}$ making the threshold itself robust. This form is also shared by the normal quantiles (differing only in the function $k(q)$), but because non-robust estimates for c and b are implicitly used by the Q-statistic, it is not robust. Further, by choosing a heavy-tailed distribution like the Laplace, the quantiles are more appropriate for the heavy-tails we observed, but the robustness of our threshold comes from our parameter estimation.

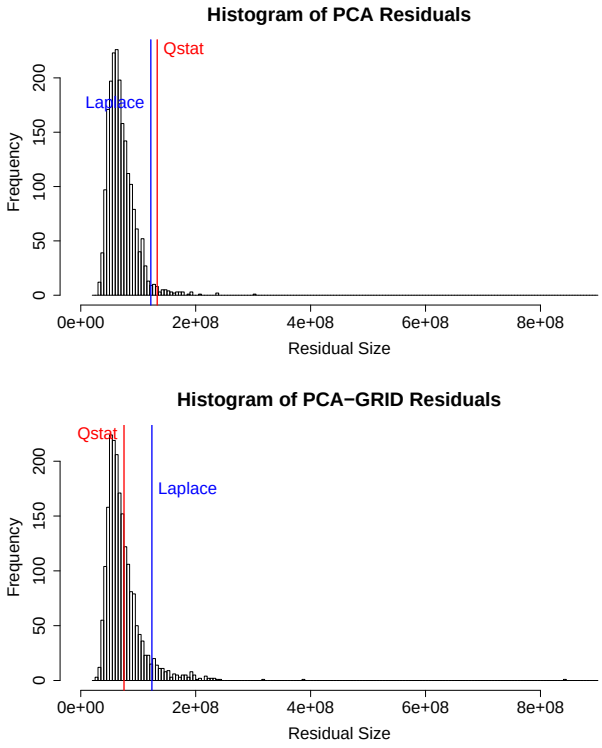

Figure 2: Histograms of the residuals for the original PCA algorithm (left) and the PCA-GRID algorithm (the largest residual is excluded as an outlier). Red and blue vertical lines demarcate the threshold selected using the Q-statistic and the Laplace threshold, respectively.

Empirically, the Laplace threshold also proved to be better suited for thresholding the residuals of our models than the Q-statistic. As can be seen in Figure 2, both the Q-statistic and the Laplace threshold produce a reasonable threshold on the residuals of the PCA algorithm but only the Laplace threshold produces a reasonable threshold for the residuals of the PCA-GRID algorithm; the Q-statistic vastly underestimates the spread of the residuals. As was consistently seen throughout our experiments, the Laplace threshold proved to be a more reliable threshold than the Q-statistic.

5. METHODOLOGY

5.1 Traffic Data

We use OD flow data collected from the Abilene (Internet2 backbone) network to simulate attacks on PCA-based anomaly detection. Data was collected over an almost continuous 6 month period from March 1, 2004 through September 10, 2004 [33]. Each week of data consists of 2016 measurements across all 144 network OD flows binned into 5 minute intervals. At the time of collection the network consisted of 12 PoPs and 15 inter-PoP links. 54 virtual links are present in the data corresponding to two directions for each inter-PoP link and an ingress and egress link for each PoP.

5.2 Validation

To evaluate the subspace method and ANTIDOTE in the face of poisoning and DoS attacks, we use two consecutive weeks of data—the first for training and the second for testing. The poisoning occurs throughout the training phase, while the attack occurs during the test week. An alternate method (described later) is needed for the Boiling Frog scheme where training and poisoning occur over multiple weeks. Our performance metric for measuring the success of the poisoning strategies is through their impact on a PCA-based detector's *false negative rate* (FNR). The FNR is the ratio of the number of successful evasions to the total number of attacks (i.e., the attacker's success rate is PCA's FNR rate). We also use Receiver Operating Characteristic (ROC) curves to visualize a detection method's trade-off between detection rate (TPR) and *false positive rate* (FPR).

In order to compute the FNRs and FPRs, we generate synthetic anomalies according to the method of Lakhina et al. [16] and inject them into the Abilene data. While there are disadvantages to this method, such as the conservative assumption that a single volume size is anomalous for all flows, we adopt it for the purposes of relative comparison between PCA and Robust PCA, to measure relative effects of poisoning, and for consistency with prior studies. We use week-long training sets, as such a time scale is sufficiently large to capture weekday and weekend cyclic trends [28], and previous studies operated on this same time scale [16]. There is nothing inherent to our method that limits its use to this time scale; our methods will work as long as the training data is poisoned throughout. Because the data is binned in 5 minute windows (corresponds to the reporting interval of SNMP), a decision about whether or not an attack is present can be made at the end of each 5 minute window; thus attacks can be detected within 5 minutes of their occurrence.

Starting with the flow traffic matrix $\mathbf{X}$ for the test week, we generate a positive example (an anomalous OD flow) by setting flow f's volume at time t, $X_{t,f}$, to be a large value known to correspond to an anomalous flow (replacing the original traffic volume in this time slot). This value is defined [16] to be 1.5 times a cutoff of 8×10^7. After multiplying by the routing matrix $\mathbf{A}$, the link volume measurement at time t is anomalous. We repeat this process for each time t (each 5 minute window) in the test week to generate a set of 2016 anomaly samples for the single target flow f.

In order to obtain FPRs, we generate negative examples (benign OD flows) as follows. We fit the data to an EWMA model that is intended to capture the main trends of the data without much noise. We use this model to select which points in time, in an Abilene flow's time series, to use as negative examples. We compare the actual data and the EWMA model, and if the difference is small (not in the flow's top one percentile) for a particular flow at a particular time, $X_{t,f}$, then we label the element $X_{t,f}$ as "benign." We do this across all flows; when we find time slots where all flows are labeled as benign, we run our detectors and see whether or not they raise an alarm for those time slots.

We simulate a DoS attack along every flow at every time. We average FNRs over all 144 possible anomalous flows and all 2016 anomaly times. When reporting the effect of an attack on traffic volumes, we first average over links within each flow then over flows. Furthermore we generally report average volumes relative to the pre-attack average volumes. Thus a single poisoning experiment was based on one week of poisoning with FNRs computed during the test week that includes 144×2016 samples coming from the different flows and time slots. Because the poisoning is deterministic in *Add-More-If-Bigger* this experiment was run once for that scheme. In contrast, for the *Random* poisoning scheme, we ran 20 independent repetitions of poisoning experiments data because the poisoning is random.

To produce the ROC curves, we use the squared prediction errors produced by the detection methods, that consist of anomalous and normal examples from the test set. By varying the method's threshold from $-\infty$ to ∞ a curve of possible (FPR, TPR) pairs is produced from the set of SPE's; the Q-statistic and Laplace threshold, each correspond to one such point in ROC space. We adopt the Area Under Curve (AUC) statistic from Information Retrieval to directly compare ROC curves. The area under an ROC curve of detector $\mathcal{A}$ estimates the conditional probability

$$AUC(\mathcal{A}) \approx \Pr\left(SPE_{\mathcal{A}}(\mathbf{y}_1) > SPE_{\mathcal{A}}(\mathbf{y}_2)\right) ,$$

given anomalous and normal random link volume vectors $\mathbf{y}_1$ and $\mathbf{y}_2$. The ideal detector has an AUC of 1, while the random predictor achieves an AUC of 0.5.

5.3 Single Period & Boiling Frog Poisoning

We evaluate the effectiveness of our attacker strategies using weeks 20 and 21 from the Abilene dataset to simulate the *Single-Training Period* attacks. The PCA algorithm is trained on the week 20 traffic matrix poisoned by the attacker; we then inject attacks during week 21 to see how often the attacker can evade detection. We select these particular weeks because PCA achieved the lowest FNRs on these during testing.

To test the *Boiling Frog* attack we simulate traffic matrix data, inspired by methods used in [16]. Our simulations present multiple weeks of stationary data to the adversary. While such data is unrealistic in practice, it is an easy case on which PCA should succeed. Anomaly detection under non-stationary conditions is difficult due to the learner's inability to distinguish between benign data drift, and adversarial poisoning. Demonstrated flaws of PCA in the stationary case constitute strong results. We decided to validate the *Boiling Frog* attack on a synthesized multi-week dataset, because the 6 month Abilene dataset of [33] proved to be too non-stationary for PCA to consistently operate well from one week to the next. It is unclear whether the non-stationarity observed in this data is prevalent in general or whether it is an artifact of the dataset.

We synthesize a multi-week set of OD flow traffic matrices, with stationarity on the inter-week level. We use a three step generative procedure to model each OD flow separately. First the underlying daily cycle of the OD flow f time series is modeled by a sinusoidal approximation. Then the times at which the flow is experiencing an anomaly are modeled by a Binomial arrival process with inter-arrival times distributed according to the geometric distribution. Finally Gaussian white noise is added to the base sinusoidal model during times of benign OD flow traffic; and exponential traffic is added to the base model during times of anomalous traffic. We next describe the process of fitting this generative model to the week 20 Abilene data.

In step 1, we capture the underlying cyclic trends via Fourier basis functions. We use sinusoids of periods of 7, 5 and 3 days, and 24, 12, 6, 3 and 1.5 hours, as well as

a constant function [16]. For each OD flow, we find the Fourier coefficients from the flow's projection onto this basis. We next remove the portion of the traffic modeled by this Fourier forecaster and model the remaining residual traffic via two processes. One is a noise process modeled by a zero-mean Gaussian to capture short-term benign traffic variance. The second process models volume anomalies as being exponentially distributed.

In step 2 we select which of the two noise processes is used at each time interval. After computing our model's residuals (the difference between the observed and predicted traffic) we note the smallest negative residual value $-m$. We assume that residuals in the interval $[-m, m]$ correspond to benign traffic and that residuals exceeding m correspond to traffic anomalies. We separate benign variation and anomalies in this way since these effects behave quite differently. (This is an approximation but it works reasonably well for most OD flows.) Negative residual traffic reflects benign variance, and since we assume that benign residuals have a zero-mean distribution, it follows that such residuals should lie within the interval $[-m, m]$. Upon classifying residual traffic as benign or anomalous we then model anomaly arrival times as a Bernoulli arrival process. Under this model the inter-anomaly arrival times become geometrically distributed. Since we consider only spatial PCA methods, the placement of anomalies is of secondary importance.

For the final step, the parameters for the two residual traffic volume and the inter-anomaly arrival processes are inferred from the residual traffic using the Maximum Likelihood estimates of the Gaussian's variance and exponential and geometric rates respectively. Positive goodness-of-fit results (Q-Q plots not shown) have been obtained for mouse, medium and elephant flows.

In our simulations, we constrain all link volumes to respect the link capacities in the Abilene network: 10gbps for all but one link that operates at one fourth of this rate. We cap chaff that would cause traffic to exceed the link capacities.

6. POISONING EFFECTIVENESS

6.1 Single Training Period Poisoning

We evaluate the effectiveness of our three data poisoning schemes in *Single-Training Period* attacks. During the testing week, the attacker launches a DoS attack in each 5 minute time window. The results of these attacks are displayed in Fig. 3. Although our poisoning schemes focus on adding variance, the mean of the OD flow being poisoned increases as well, increasing the means of all links over which the OD flow traverses. The x-axis in Fig. 3 indicates the relative increase in the mean rate. We average over all experiments (i.e., over all OD flows).

As expected the increase in evasion success is smallest for the uninformed strategy, intermediate for the locally-informed scheme, and largest for the globally-informed poisoning scheme. A locally-informed attacker can use the *Add-More-If-Bigger* scheme to raise his evasion success to 28% from the baseline FNR of 3.67% via a 10% average increase in the mean link rates due to chaff. Although 28% may not be viewed as a high likelihood of evasion, the attacker success rate is nearly 8 times larger than the unpoisoned PCA model's rate. This number represents an average over attacks launched in each 5 minute window, so the attacker could simply retry multiple times. With our *Globally-Informed* with a 10% average increase in the mean link rates, the unpoisoned FNR is raised by a factor of 10 to 38% and eventually to over 90%. The big difference between the performance of the locally-informed and globally-informed attacker is intuitive to understand. Recall that the globally-informed attacker knows a great deal more (traffic on all links, and future traffic levels) than the locally-informed one (who only knows the traffic status of a single ingress link). We consider the locally-informed adversary to have succeeded quite well with only a small view of the network. An adversary is unlikely to be able to acquire, in practice, the capabilities used in the globally-informed poisoning attack. Moreover, adding 30% chaff, in order to obtain a 90% evasion success is dangerous in that the poisoning activity itself is likely to be detected. Therefore *Add-More-If-Bigger* presents a nice trade-off, from the adversary's point of view, in terms of poisoning effectiveness, and attacker capabilities and risks. We therefore use *Add-More-If-Bigger*, the locally-informed strategy, for many of the remaining experiments.

We evaluate the PCA detection algorithm on both anomalous and normal data, as described in Section 5.2, producing the Receiver Operating Characteristic (ROC) curves displayed in Fig. 4. We produce a ROC curve (as shown) by first training a PCA model on the unpoisoned data from week 20. We next evaluate the algorithm when trained on data poisoned by *Add-More-If-Bigger*.

To validate PCA-based detection on poisoned training data, we poison exactly one flow at a time as dictated by the threat model. Thus, for relative chaff volumes ranging from 5% to 50%, *Add-More-If-Bigger* chaff is added to each flow separately to construct 144 separate training sets and 144 corresponding ROC curves for the given level of poisoning. The poisoned curves in Fig. 4 display the averages of these ROC curves (i.e., the average TPR over the 144 flows for each FPR).

We see that the poisoning scheme can throw off the balance between false positives and false negatives of the PCA detector: The detection and false alarm rates drop together rapidly as the level of chaff is increased. At 10% relative chaff volume performance degrades significantly from the ideal ROC curve (lines from $(0, 0)$ to $(0, 1)$ to $(1, 1)$) and at 20% the PCA's mean ROC curve is already close to that of blind randomized prediction (the $y = x$ line with 0.5 AUC).

6.2 Multi-Training Period Poisoning

We now evaluate the effectiveness of the *Boiling Frog* strategy, that contaminates the training data over multiple training periods. In Fig. 5 we plot the FNRs against the poisoning duration for the PCA detector. We examine four different poisoning *schedules* with growth rates g as 1.01, 1.02, 1.05 and 1.15 respectively. The goal of the schedule is to increase the attacked links' average traffic by a factor of g from week to week. The attack strength parameter θ (see Sec. 3) is chosen to achieve this goal. We see that the FNR dramatically increases for all four schedules as the poison duration increases. With a 15% growth rate the FNR is increased to more than 70% from 3.67% over 3 weeks of poisoning; even with a 5% growth rate the FNR is increased to 50% over 3 weeks. Thus *Boiling Frog* attacks are effective even when the amount of poisoned data increases rather slowly.

Recall that the two methods are retrained every week using the data collected from the previous week. However, the data from the previous week has been filtered by the de-

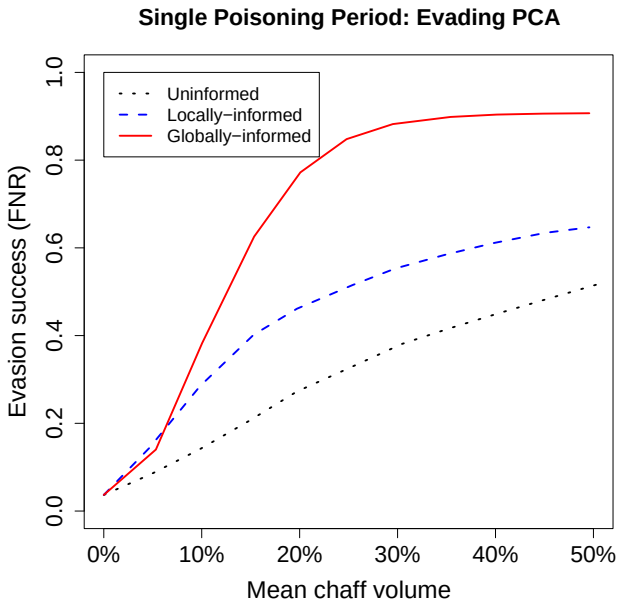

Figure 3: Evasion success of PCA under *Single-Training Period* poisoning attacks using 3 chaff methods.

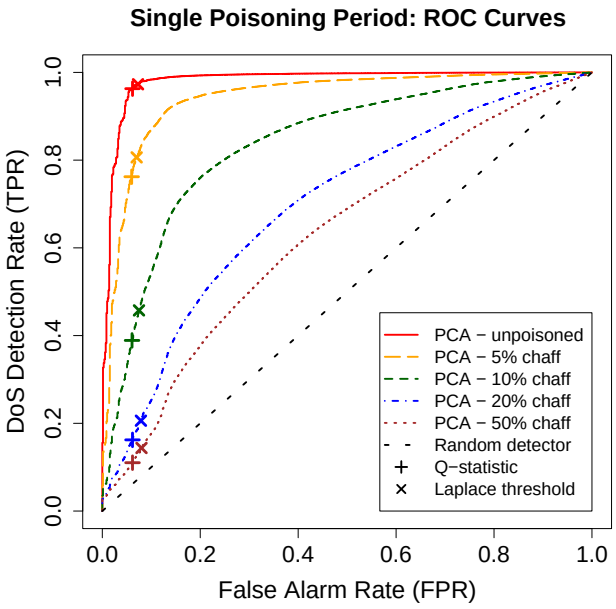

Figure 4: ROC curves of PCA under *Single-Training Period* poisoning attacks.

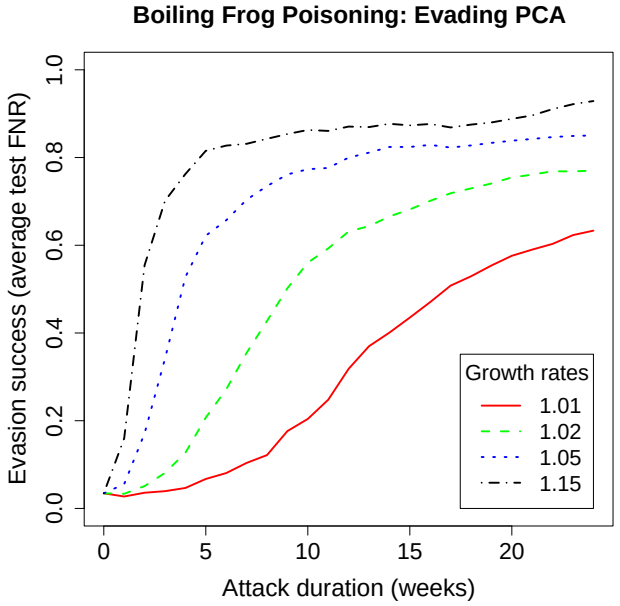

Figure 5: Evasion success of PCA under *Boiling Frog* poisoning attacks.

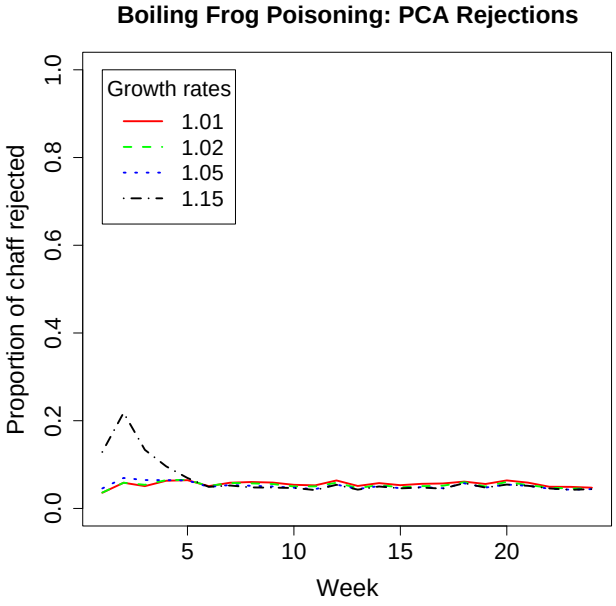

Figure 6: Chaff rejection rates of PCA under poisoning attacks shown in Fig. 5.

tector itself. At any time point flagged as anomalous, the training data is thrown out. Fig. 6 shows the proportion of chaff rejected each week by PCA (*chaff rejection rate*) for the *Boiling Frog* strategy. The three slower schedules enjoy a relatively small constant rejection rate close to 5%. The 15% schedule begins with a relatively high rejection rate, but after a month sufficient amounts of poisoned traffic mistrain PCA after which point the rates drop to the level of the slower schedules. We conclude that the *Boiling Frog* strategy with a moderate growth rate of 2–5% can significantly poison PCA, dramatically increasing its FNR while still going unnoticed by the detector.

By comparing Figs. 3 and 5, we observe that in order to raise the FNR to 50%, an increase in mean traffic of roughly 18% for the *Single-Training Period* attack is needed, whereas in the *Boiling Frog* attack the same thing can be

achieved with only a 5% average traffic increase spread across 3 weeks.

7. DEFENSE PERFORMANCE

We now assess how ANTIDOTE performs in the face of two types of poisoning attacks, one that lasts a single training period, and one that lasts for multiple training periods. For those 2 time horizons, we use the *Add-More-If-Bigger* poisoning scheme to select how much chaff to add at each point in time. We compare its performance to the original PCA-subspace method.

7.1 Single Training Period Poisoning

In Fig. 7 we illustrate ANTIDOTE's FNR for various levels of average poisoning that occur in a *Single-Training Period* attack. We can compare this to Fig. 3 that shows the same metric for the original PCA solution. We see here that the evasion success of the attack is dramatically reduced. For any particular level of chaff, the evasion success rate is approximately cut in half. Interestingly, the most effective poisoning scheme on PCA, *Globally-Informed*, is the most ineffective poisoning scheme in the face of our robust PCA solution. We believe the reason for this is that our *Globally-Informed* scheme was designed in an approximately optimal fashion to circumvent PCA. Now that the detector has changed, *Globally-Informed* is no longer optimized for the right defense. For this detector, *Random* remains equally effective because constant shifts in a large subset of the data create a bimodality that is difficult for any subspace method to reconcile. This effect is still muted compared to the dramatic success of locally-informed methods on the original detector. Further, constant shift poisoning creates unnatural traffic patterns that we believe can be detected; we leave the investigation of such techniques to future work.

Since poisoning activities distort a detector, it will affect not only the FNRs but also the false positives. To explore this trade-off, we use ROC curves in Fig. 8 for both ANTIDOTE and PCA. For comparison purposes, we include cases when the training data is both unpoisoned and poisoned. For the poisoned training scenario, each point on the curve is the average over 144 poisoning scenarios in which the training data is poisoned along one of the 144 possible flows. While ANTIDOTE performs very similarly to PCA on unpoisoned training data, PCA significantly underperforms ANTIDOTE under poisoning attacks. With a moderate mean chaff volume of 10%, ANTIDOTE's average ROC curve remains close to optimal while PCA's curve collapses towards the $y = x$ curve of the blind random detector. This means that the normal balance between FNRs and false positives is completely thrown off with PCA; however ANTIDOTE continues to retain a good operating point for these two common performance measures. *In summary, when we consider the two performance measures of FNRs and FPRs, we give up insignificant performance shifts when using* ANTIDOTE *when no poisoning events occur, yet we see enormous performance gains for both metrics when poisoning attacks do occur.*

Given Figs. 7 and 8 alone, it is conceivable that ANTIDOTE outperforms PCA only on average, and not on all flows targeted for poisoning. In place of plotting all 144 poisoned ROC curves, we display the areas under these curves (AUC) for the two detection methods in Fig. 9 under 10% chaff. Not only is average performance much better for robust PCA, but it enjoys better performance for more flows and by a large amount. We note that although PCA performs slightly better for some flows, we see that in fact both methods have excellent detection performance (because their AUCs are close to 1), and hence the distinction between the two is insignificant, for those specific flows.

Fig. 10 plots the mean AUC (averaged from the 144 ROC curves' AUCs where flows are poisoned separately) achieved by the detectors, as the level of chaff is intensified. Notice that ANTIDOTE behaves similarly to PCA under no chaff conditions, yet its performance quickly becomes superior as the amount of contamination grows. In fact, it does not take much poisoning for ANTIDOTE to exhibit much stronger performance. With PCA's performance drop, it starts approaching a random detector (equivalent to 0.5 AUC), for amounts of chaff exceeding 20%. In these last few figures, we have seen the FNR and FPR performance as it varies across flows and quantity of poisoning. In all cases, it is clear that ANTIDOTE is an effective defense and dramatically outperforms a solution that was not designed to be robust. We believe this evidence indicates that the robust techniques are a promising avenue for SML algorithms used for security applications.

7.2 Multi-Training Period Poisoning

We now evaluate effectiveness of ANTIDOTE against the *Boiling Frog* strategy, that occurs over multiple successive training periods. In Fig. 11 we see the FNRs for ANTIDOTE with the four different poisoning schedules. We observe two interesting behaviors. First, for the two most stealthy poisoning strategies (1.01 and 1.02), ANTIDOTE shows remarkable resistance in that the evasion success increases very slowly, e.g., after 10 training periods it is still below 20%. This is in stark contrast to PCA (see Fig. 5) in which, for example, after 10 weeks, the evasion success is over 50% for the 1.02 poisoning growth rate scenario. Second, under PCA the evasion success keeps rising over time. However with ANTIDOTE and the heavier poisoning strategies, we see that the evasion success actually starts to decrease after some time. The reason for this is that ANTIDOTE has started rejecting so much of the training data, that the poisoning strategy starts to lose its effectiveness.

To look more closely at this behavior we show the proportion of chaff rejected by ANTIDOTE under multi-training period poisoning episodes in Fig. 12. We see that the two slower schedules almost have a constant rejection rate close to 9%, which is higher than that of original PCA (which is close to 5%). For the faster poisoning growth schedules (5% and 15%) we observe that ANTIDOTE rejects an increasing amount of the poison data. This reflects a good target behavior for any robust detector—to reject more training data as the contamination grows. From these figures we conclude that the combination of techniques we use in ANTIDOTE, namely a PCA-based detector designed with robust dispersion goals combined with a Laplace-based cutoff threshold, is very effective at maintaining a good balance between false negative and false positive rates throughout a variety of poisoning scenarios (different amounts of poisoning, on different OD flows, and on different time horizons).

8. CONCLUSIONS

We studied the effects of multiple poisoning strategies while varying the amount of information available to the attacker and the time horizon over which the poisoning occurs.

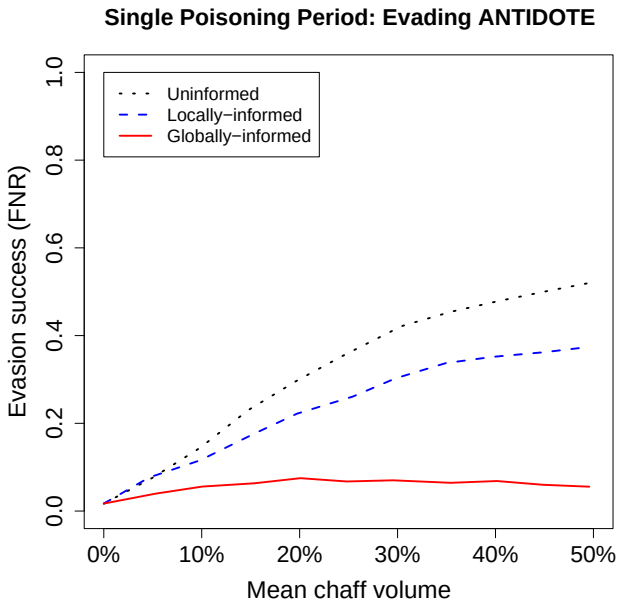

Figure 7: Evasion success of ANTIDOTE under *Single-Training Period* poisoning attacks using 3 chaff methods.

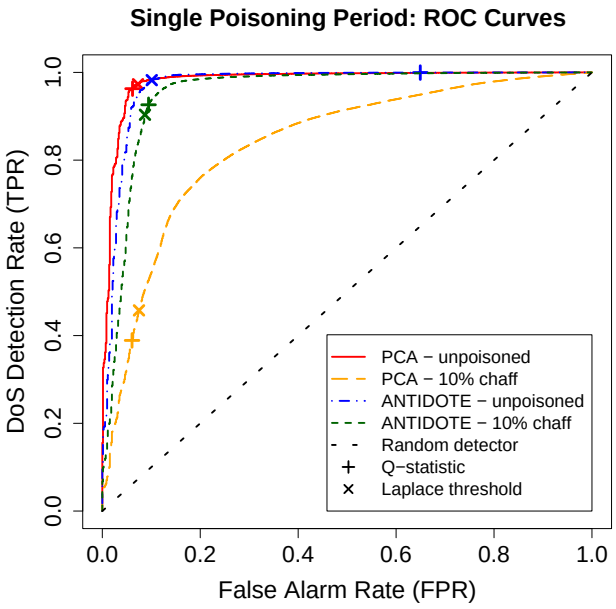

Figure 8: ROC curves of ANTIDOTE under *Single-Training Period* poisoning attacks.

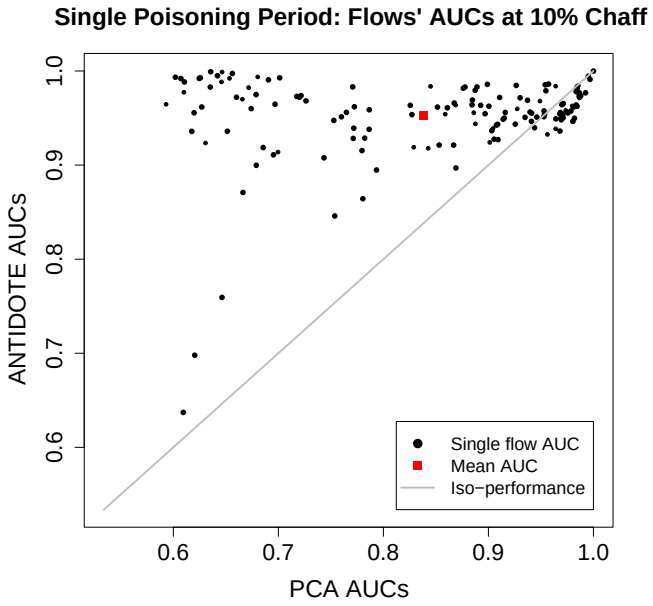

Figure 9: The 144 AUCs from the poisoned ROC curves for each possible target flow and their mean.

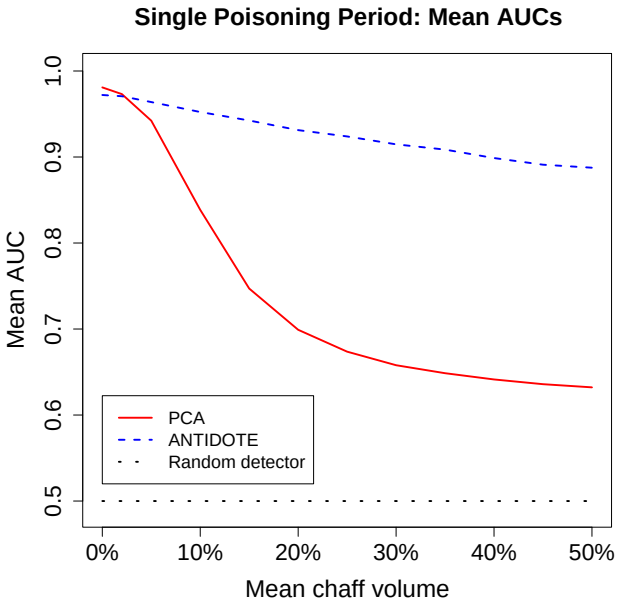

Figure 10: The mean AUCs versus mean chaff levels.

We demonstrated that the PCA-subspace method can be easily compromised (often dramatically) under all of these poisoning scenarios. From the attacker's point of view, we illustrate that simple strategies can be effective and conclude that it is not worth the risk or extra amount of work for the attacker to engage in attempts at optimal strategies. We demonstrated that our ANTIDOTE solution is robust to all of these attacks in that it does not allow poisoning attacks to shift the false positive and false negative rates in any significant way. We showed that ANTIDOTE provides robustness for nearly all the ingress POP to egress POP flows in a backbone network, rejects much of the contaminated data, and continues to operate as a DoS defense even in the face of poisoning.

In the future, we plan to adapt our scheme to defend against poisoning strategies that enable DDoS evasion, and

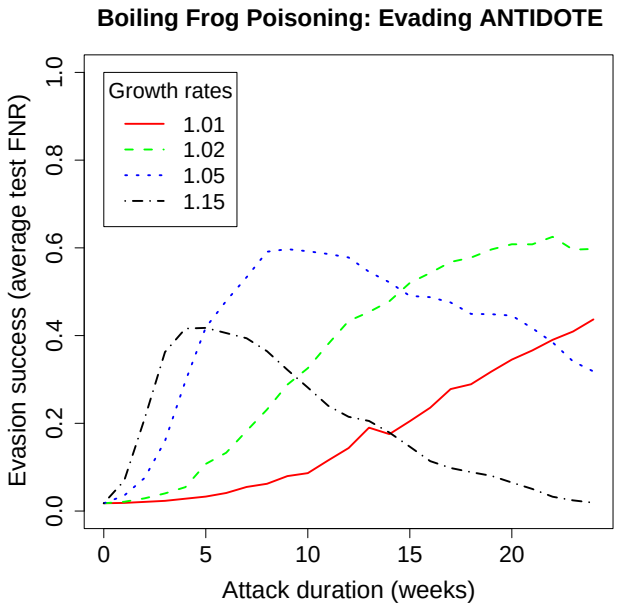

Figure 11: Evasion success of ANTIDOTE under *Boiling Frog* poisoning attacks.

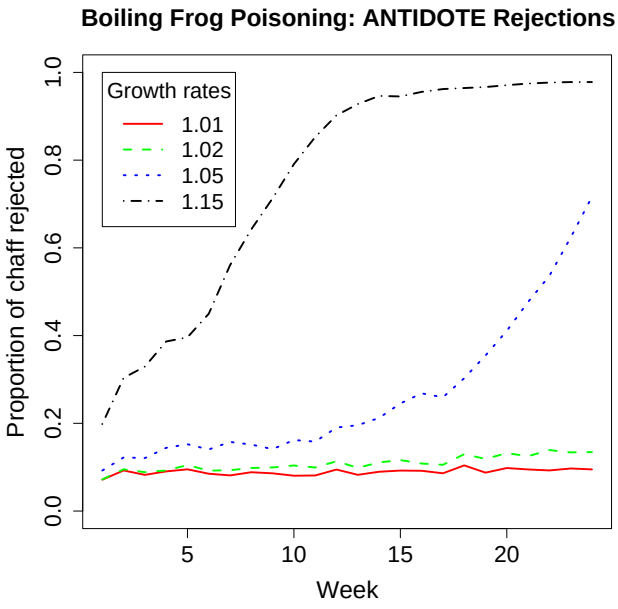

Figure 12: Chaff rejection rates of ANTIDOTE under *Boiling Frog* poisoning attacks.

intend to validate our findings on other traffic matrix datasets (e.g., GÉANT or enterprise networks). It is also interesting to consider using robust statistical methods for other detectors such as general anomography [33] techniques. We plan to go beyond rejection of poisoning data, and study methods for identifying the responsible flow for a poisoning attack by looking at correlations among links that are rejecting chaff.

9. ACKNOWLEDGEMENTS

We would like to thank Peter Bartlett, Fuching Jack Chi, Fernando Silveira, Anthony Tran and the anonymous reviewers for their helpful feedback on this project.

We gratefully acknowledge the support of our sponsors. This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award #CCF-0424422) and AFOSR (#FA9550-06-1-0244); RAD Lab, which receives support from California state MICRO grants (#06-148 and #07-012); DETERlab (cyber-DEfense Technology Experimental Research laboratory), which receives support from DHS HSARPA (#022412) and AFOSR (#FA9550-07-1-0501); NSF award #DMS-0707060; and the following organizations: Amazon, BT, Cisco, DoCoMo USA Labs, EADS, ESCHER, Facebook, Google, HP, IBM, iCAST, Intel, Microsoft, NetApp, ORNL, Pirelli, Qualcomm, Sun, Symantec, TCS, Telecom Italia, United Technologies, and VMware. The opinions expressed in this paper are solely those of the authors and do not necessarily reflect the opinions of any funding agency, the State of California, or the U.S. government.

10. REFERENCES

[1] P. Bahl, R.Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proc. SIGCOMM*, 2007.

[2] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proc. ASIACCS*, 2006.

[3] D. Brauckhoff, K. Salamatian, and M. May. Applying PCA for Traffic Anomaly Detection: Problems and Solutions. In *Proc. INFOCOM*, 2009.

[4] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[5] Y. Cheng, M. Afanasyev, P. Verkaik, P. Benko, J. Chiang, A. Snoeren, S. Savage, and G. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *Proc. SIGCOMM*, 2007.

[6] C. Croux, P. Filzmoser, and M. R. Oliveira. Algorithms for projection-pursuit robust principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 87(2), 2007.

[7] C. Croux and A. Ruiz-Gazen. High breakdown estimators for principal components: the projection-pursuit approach revisited. *J. Multivariate Analysis*, 95(1), 2005.

[8] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proc. ACM KDD*, 2004.

[9] S. J. Devlin, R. Gnanadesikan, and J. R. Kettenring. Robust estimation of dispersion matrices and principal components. *J. American Statistical Association*, 76, 1981.

[10] P. Fogla and W. Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proc. ACM CCS*, 2006.

[11] O. Hössjer and C. Croux. Generalizing univariate signed rank statistics for testing and estimating a

multivariate location parameter. *J. Nonparametric Statistics*, 4, 1995.

[12] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft. In-network PCA and anomaly detection. In *Proc. NIPS '06*, 2007.

[13] J. E. Jackson and G. S. Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3), 1979.

[14] S. Kandula, R. Chandra, and D. Katabi. What's going on? learning communication rules in edge networks. In *Proc. SIGCOMM*, 2008.

[15] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *Proc. IMC*, 2004.

[16] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. SIGCOMM*, 2004.

[17] A. Lakhina, M. Crovella, and C. Diot. Detecting distributed attacks using network-wide flow traffic. In *Proc. FloCon 2005 Analysis Workshop*, 2005.

[18] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proc. SIGCOMM*, 2005.

[19] G. Li and Z. Chen. Projection-pursuit approach to robust dispersion matrices and principal components: primary theory and Monte Carlo. *J. American Statistical Association*, 80, 1985.

[20] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proc. IMC*, 2006.

[21] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, and C. Iannaccone. MIND: A distributed multidimensional indexing for network diagnosis. In *Proc. INFOCOM*, 2006.

[22] D. Lowd and C. Meek. Adversarial learning. In *Proc. ACM KDD*, 2005.

[23] R. Maronna. Principal components and orthogonal regression based on robust scales. *Technometrics*, 47(3), 2005.

[24] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *Proc. LEET*, 2008.

[25] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proc. IEEE Symp. Security and Privacy*, 2005.

[26] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *Proc. RAID*, 2006.

[27] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *Proc. ACM CCS*, 2007.

[28] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Proc. SIGMETRICS*, 2007.

[29] B. I. P. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. Lau, S. Rao, N. Taft, and J. D. Tygar. Stealthy poisoning attacks on PCA-based anomaly detectors. In *Proc. ACM SIGMETRICS*, 2009. Extended abstract.

[30] A. Soule, K. Salamatian, and N. Taft. Combining filtering and statistical methods for anomaly detection. In *Proc. IMC*, 2005.

[31] S. Venkataraman, A. Blum, and D. Song. Limits of learning-based signature generation with adversaries. In *Proc. NDSS*, 2008.

[32] G. L. Wittel and S. F. Wu. On attacking statistical spam filters. In *Proc. CEAS*, 2004.

[33] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. IMC*, 2005.