6 Integrity Attack Case Study: PCA Detector

Adversaries can use *Causative* attacks to not only disrupt normal user activity (as we demonstrated in Chapter 5) but also to evade the detector by causing it to have many false negatives through an *Integrity* attack. In doing so, such adversaries can reduce the odds that their malicious activities are successfully detected. This chapter presents a case study of the subspace anomaly detection methods introduced by Lakhina et al. (2004*b*) for detecting network-wide anomalies such as denial-of-service (DoS) attacks based on the dimensionality reduction technique commonly known as principal component analysis (PCA) (Pearson 1901). We show that by injecting crafty extraneous noise, or chaff, into the network during training, the PCA-based detector can be poisoned so it is unable to effectively detect a subsequent DoS attack. We also demonstrate defenses against these attacks. Specifically, by replacing PCA with a more robust alternative subspace estimation procedure, we show that the resulting detector is resilient to poisoning and maintains a significantly lower false-positive rate when poisoned.

The PCA-based detector we analyze was first proposed by Lakhina et al. (2004*b*) as method for identifying volume anomalies in a backbone network. This basic technique led to a variety of extensions of the original method (e.g., Lakhina, Crovella & Diot 2004*a*, 2005*a*, 2005*b*) and to related techniques for addressing the problem of diagnosing large-volume network anomalies (e.g., Brauckhoff, Salamatian, & May 2009; Huang, Nguyen, Garofalakis, Jordan, Joseph, & Taft 2007; Li, Bian, Crovella, Diot, Govindan, Iannaccone, & Lakhina 2006; Ringberg, Soule, Rexford, & Diot 2007; Zhang, Ge, Greenberg, & Roughan 2005). While their subspace-based method is able to successfully detect DoS attacks in the network traffic, to do so it assumes the detector is trained on nonmalicious data (in an unsupervised fashion under the setting of anomaly detection). Instead, we consider an adversary that knows that an ISP is using a subspace-based anomaly detector and attempts to evade it by proactively poisoning the training data.

We consider an adversary whose goal is to circumvent detection by poisoning the training data; i.e., an *Integrity* goal to increase the detector's false-negative rate, which corresponds to the evasion success rate of the attacker's subsequent DoS attack. When trained on this poisoned data, the detector learns a distorted set of principal components that are unable to effectively discern the desired DoS attacks—a *Targeted* attack. Because PCA estimates the data's principal subspace solely on the covariance of the link traffic, we explore poisoning schemes that add chaff (additional traffic) into the network along the flow targeted by the attacker to systematically increase the targeted flow's variance; i.e., an additive contamination model. By increasing the targeted flow's variance, the attacker causes the estimated subspace to unduly shift toward the target flow, making large-volume events along that flow less detectable.

In this chapter, we explore attacks against and defenses for network anomaly detection. In Section 6.1, we introduce the PCA-based method for detecting network volume anomalies as first proposed by Lakhina et al. (2004*b*). Section 6.2 proposes attacks against the detector and Section 6.3 proposes a defense based on a robust estimator for the subspace. In Section 6.4, we evaluate the effect of attacks on both the original PCA-based approach and the proposed defense. We summarize the results of this study in Section 6.5. This chapter builds on (Rubinstein, Nelson, Huang, Joseph, Lau, Rao, Taft, & Tygar 2009*a*, 2009*b*).

Related Work

Several earlier studies examined attacks on specific learning systems for related applications. Ringberg, Soule, Rexford, & Diot (2007) 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; this is a kind of perturbation to the subspace that is not adversarial but can still significantly degrade detection performance. Our work in this chapter differs in two key ways. First, we investigate malicious data poisoning; i.e., adversarial perturbations that are stealthy and subtle and are more challenging to circumvent than routing outages. Second, Ringberg et al. focus on showing the variability in PCA's performance to certain sensitivities, and not on defenses. In this work, we propose a robust defense against a malicious adversary and demonstrate its effectiveness. It is conceivable that this technique may limit PCA's sensitivity to routing outages, although such a study is beyond the scope of this work. A study by Brauckhoff, Salamatian, & May (2009) showed that the sensitivities observed by Ringberg et al. can be attributed to the inability of the PCA-based detector to capture temporal correlations. They propose to replace PCA by a Karhunen-Loeve expansion. This study indicates that it would be important to examine, in future work, the data poisoning robustness of the proposal of Brauckhoff et al. to understand how it fares under adversarial conditions.

Contributions

The first contribution of this chapter is a detailed analysis of how adversaries subvert the learning process in these *Causative Integrity* attacks using additive contamination. We explore a range of poisoning strategies in which the attacker's knowledge about the network traffic state and its time horizon (length of poisoning episode) vary. Through theoretical analysis of global poisoning strategies, we reveal simple and effective poisoning strategies for the adversary that can be used to successfully exploit various levels of knowledge that the attacker has about the system. To gain further insights as to why these attacks are successful, we demonstrate their impact on the normal model built by the PCA detector.

The second contribution is to design a robust defense against this type of poisoning. It is known that PCA can be strongly affected by outliers (Ringberg, Soule, Rexford, & Diot 2007). However, instead of finding the principal components along directions that maximize variance, alternative PCA-like techniques find more robust components by maximizing alternative dispersion measures with desirable robustness properties. Analogously in centroid estimation, 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 is a form of distributional robustness (cf. Hampel et al. 1986). This concept was also extended to design and evaluate estimates of dispersion that are robust alternatives to variance (a nonrobust estimate of dispersion) such as the median absolute deviation (MAD), which is robust to outliers. PCA too can be thought of as an estimator of the underlying subspace of the data, which selects the subspace that minimizes the sum of the square of the data's residuals; i.e., the variance of the data in the residual subspace. This sum-of-squares estimator also is nonrobust and is thus sensitive to outliers (cf. Maronna et al. 2006). Over the past two decades a number of robust PCA algorithms have been developed that maximize alternative measures of dispersion such as the MAD instead of variance. The PCA-Grid algorithm was proposed by Croux et al. (2007) as an efficient method for estimating directions that maximize the MAD without underestimating variance (a flaw identified in previous solutions). 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. Together, we refer to the method that combines PCA-Grid with a Laplace cutoff threshold as Antidote. Because it builds on robust subspace estimates, this method substantially reduces the effect of outliers and is able to reject poisonous training data as we demonstrate empirically in Section 6.4.4.

The third contribution is an evaluation and comparison of both Antidote and the original PCA method when exposed to a variety of poisoning strategies and an assessment of their susceptibility to poisoning in terms of several performance metrics. To do this, we used traffic data from the Abilene Internet2 backbone network (Zhang, Ge, Greenberg, & Roughan 2005), a public network traffic dataset used in prior studies of PCA-based anomaly detection approaches. We show that the original PCA method can be easily compromised by the poisoning schemes we present using only small volumes of chaff (i.e., fake traffic used to poison the detector). In fact, for moderate amounts of chaff, the performance of the PCA detector approaches that of a random detector. However, Antidote is dramatically more robust to these attacks. It outperforms PCA in that it *i*) more effectively limits the adversary's ability to increase its evasion success; *ii*) can reject a larger portion of contaminated training data; and *iii*) provides robust protection for nearly all origin-destination flows through the network. The gains of Antidote for these performance measures are large, especially as the amount of poisoning increases. Most importantly, we demonstrate that when there is no poisoning Antidote incurs an insignificant decrease in its false-negative and false-positive performance, compared to PCA. However, when poisoning does occur, Antidote incurs significantly less degradation than PCA with respect to both of these performance measures. Fundamentally, the original PCA-based approach was not designed to be robust, but these results show that it is possible to adapt the original technique to bolster its performance under an adversarial setting by using robust alternatives.

Finally, we also summarize episodic poisoning and its effect on both the original PCA-based detector and ANTIDOTE as further discussed in Rubinstein (2010). Because the network behaviors are nonstationary, the baseline models must be periodically retrained to capture evolving trends in the underlying data, but a *patient* adversary can exploit the periodic retraining to slowly poison the filter over many retraining periods. In previous usage scenarios (Lakhina, Crovella, & Diot 2004*b*; Soule, Salamatian, & Taft 2005), 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 over several retraining epochs, the attacker decreases the chance that the poisoning activity itself is detected—an episodic poisoning scheme. As we show in Section 6.4.5, these poisoning schemes can boost the false-negative rate as high as the nonstealthy strategies, with almost unnoticeable increases in weekly traffic volumes, albeit over a longer period of time.

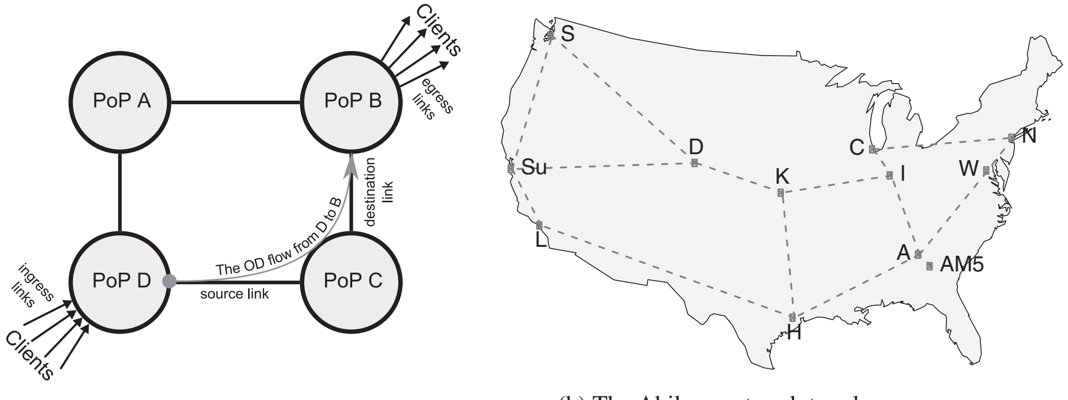
6.1 PCA Method for Detecting Trafffic Anomalies

To uncover anomalies, many network anomography detection techniques analyze the network-wide flow traffic matrix (TM), which describes the traffic volume between all pairs of points-of-presence (PoP) in a backbone network and contains the observed traffic volume time series for each origin-destination (OD) flow. PCA-based techniques instead uncover anomalies using the more readily available link traffic matrix. In this section, we define traffic matrices and summarize the PCA-based anomaly detection method of Lakhina et al. (2004*b*) using the notation introduced in Section 2.1.

6.1.1 Traffic Matrices and Volume Anomalies

We begin with a brief overview of the volume anomaly detection problem, in which a network administrator wants to identify unusual traffic in *origin-destination* (OD) flows between *points-of-presence* (PoP) nodes in a backbone network (see Figure 6.1). The flow traffic is routed along a network represented as an undirected graph $(\mathbb{V}, \mathbb{E})$ on $V \triangleq |\mathbb{V}|$ *nodes* and $D \triangleq |\mathbb{E}|$ unidirectional *links*. There are $Q \triangleq V^2$ OD flows in this network (between every pair of PoP nodes), and the amount of traffic transmitted along the q^{th} flow during the t^{th} time slice is $Q_{t,q}$. All OD flow traffic observed in T time intervals is summarized by the matrix $\mathbf{Q} \in \Re^{T \times Q}$. Ideally, one would like to identify a pair (t, q) as anomalous if the traffic along flow q is unusually large at time t, but $\mathbf{Q}$ is not directly observable within the backbone network. Instead what is observable is the network link traffic during the t^{th} time slice.

More specifically, network link traffic is the superposition of all OD flows; i.e., the data transmitted along the q^{th} flow contributes to the overall observed link traffic along the links traversed by the q^{th} flow's route from its origin to its destination. Consider



(a) Links used for data poisoning (b) The Abilene network topology

Figure 6.1 Depictions of network topologies, in which subspace-based detection methods can be used as traffic anomaly monitors. **(a)** A simple four-node network with four edges. Each node represents a PoP, and each edge represents a bidirectional link between two PoPs. Ingress links are shown at node `D` although all nodes have ingress links that carry traffic from clients to the PoP. Similarly, egress links are shown at node `B` carrying traffic from the PoP to its destination client. Finally, a flow from `D` to `B` is depicted flowing through `C`; this is the route taken by traffic sent from PoP `D` to PoP `B`. **(b)** The Abilene backbone network overlaid on a map of the United States representing the 12 PoP nodes in the network and the 15 links between them. PoPs `AM5` and `A` are actually co-located together in Atlanta, but the former is displayed to the southeast to highlight its connectivity.

a network with Q OD flows and D links and measure traffic on this network over T time intervals. The relationship between link traffic and OD flow traffic is concisely captured by the *routing matrix* $\mathbf{R}$. This matrix is a $D \times Q$ matrix such that $R_{i,j} = 1$ if the j^{th} OD flow passes over the i^{th} link, and otherwise is zero. Thus, if $\mathbf{Q}$ is the $T \times Q$ traffic matrix containing the time-series of all OD flows and $\mathbf{X}$ is the $T \times D$ link TM containing the time-series of all links, then $\mathbf{X} = \mathbf{QR}^\top$. We denote the t^{th} row of $\mathbf{X}$ as $\mathbf{x}^{(t)} = \mathbf{X}_{t,\bullet}$ (the vector of D link traffic measurements at time t) and the traffic observed along a particular *source link*, s, by $x_s^{(t)}$. We denote column q of routing matrix $\mathbf{R}$ by $\mathbf{R}_q$; i.e., the indicator vector of the links used by the q^{th} flow.

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 DoS attacks, distributed DoS (DDoS) attacks, flash crowds, device failures, misconfigured devices, and other abnormal network events. DoS attacks serve as the canonical example of an attack throughout this chapter.

6.1.2 Subspace Method for Anomaly Detection

Here we briefly summarize the PCA-based anomaly detector introduced by Lakhina, Crovella, & Diot (2004*b*). They observed that the high degree of traffic aggregation on ISP backbone links often causes OD flow volume anomalies to become indistinct within normal traffic patterns. They also observe that although the measured data has high dimensionality, D, the normal traffic patterns lie in a subspace of low-dimension $K \ll D$; i.e., the majority of normal traffic can be described using a smaller representation because of temporally static correlations caused by the aggregation. Fundamentally, they found that the link data is dominated by a small number of flows. Inferring this normal traffic subspace using PCA (which finds the principal components within the traffic) facilitates the identification of volume anomalies in the residual (abnormal) subspace. For the Abilene (Internet2 backbone) network, most variance can be captured by the first $K = 4$ principal components; i.e., the link traffic of this network effectively resides in a (low) K-dimensional subspace of $\Re^D$.

PCA is a dimensionality reduction technique that finds K orthogonal *principal components* to define a K-dimensional subspace that captures the maximal amount of variance from the data. First, PCA centers the data by replacing each data point $\mathbf{x}^{(t)}$ with $\mathbf{x}^{(t)} - \hat{\mathbf{c}}$ where $\hat{\mathbf{c}}$ is the central location estimate, which in this case is the mean vector $\hat{\mathbf{c}} = \frac{1}{T}\mathbf{X}^\top\mathbf{1}$. Let $\hat{\mathbf{X}}$ be the centered link traffic matrix; i.e., with each column of $\mathbf{X}$ translated to have zero mean. Next, PCA estimates the principal subspace on which the mean-centered data lies by computing its principal components. The kth principal component satisfies

$$\mathbf{v}^{(k)} \in \underset{\mathbf{w}:\|\mathbf{w}\|_2=1}{\operatorname{argmax}} \left[\left\| \hat{\mathbf{X}} \left(\mathbf{I} - \sum_{i=1}^{k-1} \mathbf{v}^{(i)} (\mathbf{v}^{(i)})^\top \right) \mathbf{w} \right\|_2 \right]. \tag{6.1}$$

The resulting K-dimensional subspace spanned by the first K principal components is represented by a $D \times K$ dimensional matrix $\mathbf{V}^{(K)} = [\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(K)}]$ that maps to the *normal* traffic subspace $\dot{\mathbb{S}}$ and has a projection matrix $\dot{\mathbf{P}} = \mathbf{V}^{(K)} \left(\mathbf{V}^{(K)}\right)^\top$ into $\Re^D$. The residual $(D-K)$-dimensional subspace is spanned by the remaining principal components $\mathbf{W}^{(K)} = \left[\mathbf{v}^{(K+1)}, \mathbf{v}^{(K+2)}, \ldots, \mathbf{v}^{(D)}\right]$. This matrix maps to the abnormal traffic subspace $\ddot{\mathbb{S}}$ with a corresponding projection matrix $\ddot{\mathbf{P}} = \mathbf{W}^{(K)} \left(\mathbf{W}^{(K)}\right)^\top = \mathbf{I} - \dot{\mathbf{P}}$ onto $\Re^D$.

Volume anomalies can be detected by decomposing the link traffic into normal and abnormal components such that $\mathbf{x}^{(t)} = \dot{\mathbf{x}}^{(t)} + \ddot{\mathbf{x}}^{(t)} + \hat{\mathbf{c}}$ where $\dot{\mathbf{x}}^{(t)} \triangleq \dot{\mathbf{P}}\left(\mathbf{x}^{(t)} - \hat{\mathbf{c}}\right)$ is the modeled normal traffic and $\ddot{\mathbf{x}}^{(t)} \triangleq \ddot{\mathbf{P}}\left(\mathbf{x}^{(t)} - \hat{\mathbf{c}}\right)$ is the residual traffic, corresponding to projecting $\mathbf{x}^{(t)}$ onto $\dot{\mathbb{S}}$ and $\ddot{\mathbb{S}}$, respectively. A volume anomaly at time t typically results in a large change to $\ddot{\mathbf{x}}^{(t)}$, which can be detected by thresholding the squared prediction error $\left\|\ddot{\mathbf{x}}^{(t)}\right\|_2^2$ against the threshold Q_β, which is chosen to be the Q-statistic at the $1-\beta$ confidence level (Jackson & Mudholkar 1979). This PCA-based detector defines the following classifier:

$$f\left(\mathbf{x}^{(t)}\right) = \begin{cases} "+", & \left\|\ddot{\mathbf{P}}\left(\mathbf{x}^{(t)} - \hat{\mathbf{c}}\right)\right\|_2^2 > Q_\beta \\ "-", & \text{otherwise} \end{cases} \tag{6.2}$$

for a link measurement vector, where "+" indicates that the tth time slice is *anomalous* and "−" indicates it is *innocuous*. Due to the nonstationarity of normal network traffic (gradual drift), periodic retraining is necessary. We assume the detector is retrained weekly.

6.2 Corrupting the PCA Subspace

In this section, we survey a number of data poisoning schemes and discuss how each is designed to affect the training phase of a PCA-based detector. Three general categories of attacks are considered based on the attacker's capabilities: uninformed attacks, locally informed attacks, and globally informed attacks. Each of these reflects different levels of knowledge and resources available to the attacker.

6.2.1 The Threat Model

The adversary's goal is to launch a DoS attack on some victim and to have the attack traffic successfully transit an ISP's network without being detected en route. The DoS traffic traverses the ISP from an ingress point-of-presence (PoP) node to an egress PoP of the ISP. To avoid detection prior to the desired DoS attack, the attacker poisons the detector during its periodic retraining phase by injecting additional traffic (chaff) along the OD flow (i.e., from an ingress PoP to an egress PoP) that it eventually intends to attack. Based on the anticipated threat against the PCA-based anomaly detector, the contamination model we consider is a *data alteration* model where the adversary is limited to only altering the traffic from a single source node. This poisoning 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 must decide how much chaff to add and when to do so. These choices are guided by the degree of covertness required by the attacker and the amount of information available to it.

We consider poisoning strategies in which the attacker has various potential levels of information at its disposal. The weakest attacker is one that knows nothing about the traffic at the ingress PoP and adds chaff randomly (called an *uninformed* attack). Alternatively, a partially informed attacker knows the current volume of traffic on the ingress link(s) that it intends to inject chaff on. Because many networks export SNMP records, an adversary might intercept this information or possibly monitor it itself (i.e., in the case of a compromised router). We call this type of poisoning a *locally informed* attack because this adversary only observes the local state of traffic at the ingress PoP of the attack. In a third scenario, the attacker is *globally informed* because its global view over the network enables it to know the traffic levels on all network links and this attacker has knowledge of all 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 impossible to achieve, we study this scenario to better understand the limits of variance injection poisoning schemes.

We assume the adversary does not have control over existing traffic (i.e., it cannot delay or discard traffic). Similarly, the adversary cannot falsify SNMP reports to PCA.

Such approaches are more conspicuous because the inconsistencies in SNMP reporting from neighboring PoPs could expose the compromised router. Stealth is a major goal of this attacker—it does not want its DoS attack or its poisoning to be detected until the DoS attack has successfully been executed.

So far we have focused primarily on nondistributed poisoning of DoS detectors and on nondistributed DoS attacks. *Distributed* poisoning that aims to evade a DoS detector is also possible; the globally informed poisoning strategy presented later is an example since this adversary potentially can poison any network link. We leave the study of distributed forms of poisoning to future work. Nonetheless, by demonstrating that poisoning can effectively achieve evasion in the nondistributed setting, this work shows that distributing the poisoning is unnecessary, although it certainly should result in even more powerful attacks.

For each of these scenarios of different poisoning strategies and the associated level of knowledge available to the adversary, we now detail specific poisoning schemes. In each, the adversary decides on the quantity of $a^{(t)}$ chaff to add to the target flow time series at a time t, and during the training period it sends a total volume of chaff $A \triangleq \sum_{t=1}^{T} a^{(t)}$. Each strategy has an attack parameter θ, which controls the intensity of the attack. Ultimately, in each strategy, the attacker's goal is to maximally increase traffic variance along the target flow to mislead the PCA detector to give that flow undue representation in its subspace, but each strategy differs in the degree of information the attacker has to achieve its objective. For each scenario, we present only one representative poisoning scheme, although others were studied in prior work (Rubinstein, Nelson, Huang, Joseph, Lau, Taft, & Tygar 2008).

6.2.2 Uninformed Chaff Selection

In this setting, the adversary has no knowledge about the network and randomly injects chaff traffic. At each time t, the adversary decides whether or not to inject chaff according to a Bernoulli random variable. If it decides to inject chaff, the amount of chaff added is of size θ, i.e., $a^{(t)} = \theta$. This method is independent of the network traffic since this attacker is uninformed—we call it the *Random* poisoning scheme.

6.2.3 Locally Informed Chaff Selection

In the locally informed scenario, the attacker observes the volume of traffic in the ingress link it controls at each point in time, $x_s^{(t)}$. Hence this attacker only adds chaff when the current traffic volume is already reasonably large. In particular, it adds chaff when the traffic volume on the link exceeds a threshold parameter α (typically the mean of the overall flow's traffic). The amount of chaff added is then $a^{(t)} = \left(\max\left\{0, x_s^{(t)} - \alpha\right\}\right)^{\theta}$. In other words, if the difference between the observed link traffic and a parameter α is non-negative, the chaff volume is that difference to the power θ; otherwise, no chaff is added during the interval. In this scheme (called *add-more-if-bigger*), the farther the traffic is from the mean link traffic, the larger the deviation of chaff inserted.

6.2.4 Globally Informed Chaff Selection

The *globally informed* scheme captures an omnipotent adversary with full knowledge of $\mathbf{X}$, $\mathbf{R}$, and the future measurements $\tilde{\mathbf{x}}$, and that 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 the compromised link, whereas in this scenario, the adversary can inject chaff into any link. For each link n and each time t, the adversary must select the amount of chaff $A_{t,n}$. We cast this process into an optimization problem that the adversary solves to maximally increase its chance of a DoS evasion along the target flow q. Although these capabilities are unrealistic, we study the globally informed poisoning strategy 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 the q^{th} target flow at the t^{th} time window. If the vector of link volumes at future time t is $\tilde{\mathbf{x}}$, where the tilde distinguishes this future measurement from past training data $\hat{\mathbf{X}}$, then the vectors of anomalous DoS volumes are given by $\tilde{\mathbf{x}}(\delta, q) = \tilde{\mathbf{x}} + \delta \cdot \mathbf{R}_q$. Denote by $\mathbf{A}$ the matrix of link traffic injected into the network by the adversary during training. Then the PCA-based anomaly detector is trained on an altered link traffic matrix $\hat{\mathbf{X}} + \mathbf{A}$ to produce the mean traffic vector μ, the top K eigenvectors $\mathbf{V}^{(K)}$, 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 optimizing $\mathbf{A}$ accordingly. The PCA evasion problem corresponds to solving the following:

$$\begin{aligned} \max_{\delta \in \Re,\ \mathbf{A} \in \Re^{T \times Q}} \quad & \delta \\ \text{s.t.} \quad & \left(\mu, \mathbf{V}, Q_\beta\right) = \text{PCA}(\mathbf{X} + \mathbf{A}, K) \\ & \left\| \ddot{\mathbf{P}}(\tilde{\mathbf{x}}(\delta, q) - \mu) \right\|_2 \leq Q_\beta \\ & \|\mathbf{A}\|_1 \leq \theta \qquad \forall t, q \ \ A_{t,q} \geq 0, \end{aligned}$$

where θ is a constant constraining total chaff and the matrix 1-norm is here defined as $\|\mathbf{A}\|_1 \triangleq \sum_{t,q} |A_{t,q}|$. The second constraint guarantees evasion by requiring that the contaminated link volumes at time t are classified as innocuous according to Equation 6.2. 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,[1] and show that the above objective seeks to maximize the attack direction $\mathbf{R}_q$'s projected length in the normal subspace $\max_{\mathbf{A} \in \Re^{T \times Q}} \left\| \left(\mathbf{V}^{(K)}\right)^\top \mathbf{R}_q \right\|_2$. Next, we restrict our focus to traffic processes that generate spherical k-rank link traffic covariance matrices.[2] 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

[1] The full proof is omitted due to space constraints.

[2] While the spherical assumption does not hold in practice, the assumption of low-rank traffic matrices is met by published datasets (Lakhina et al. 2004*b*).

eigenvectors $\mathbf{V}^{(K)}$ in the objective, with the matrix of all eigenvectors weighted by their corresponding eigenvalues $\Sigma\mathbf{V}$. This transforms the PCA evasion problem into the following relaxed optimization:

$$\begin{aligned} \max_{\mathbf{A}\in\Re^{T\times Q}} \quad & \left\| (\hat{\mathbf{X}} + \mathbf{A})\mathbf{R}_q \right\|_2 \\ \text{s.t.} \quad & \|\mathbf{A}\|_1 \leq \theta \\ & \forall t, q \;\; A_{t,q} \geq 0. \end{aligned} \tag{6.3}$$

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 the globally informed 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 Equation (6.3) indicates that the best strategy to evade detection is to inject chaff only along the links $\mathbf{R}_q$ associated with the target flow q. This follows from the form of the initializer $\mathbf{A}^{(0)} \propto \hat{\mathbf{X}}\mathbf{R}_q\mathbf{R}_q^\top$ (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 $\mathbf{X}$ to determine chaff allocation along the flow whereas the latter uses only local information.

6.2.5 Boiling Frog Attacks

In the above attacks, chaff was designed to affect a single training period (one week) in the training cycle of the detector, but here we consider the possibility of episodic poisoning that is carried out over multiple weeks of retraining the subspace detector to adapt to changing traffic trends. As with previous studies, we assume that the PCA-subspace method is retrained on a weekly basis using the traffic observed in the previous week to retrain the detector at the beginning of the new week; i.e., the detector for the m^{th} week is learned from the traffic of week $m-1$. Further, as with the outlier model discussed in Chapter 4, we sanitize the data from the prior week before retraining so that all detected anomalies are removed from it. This sort of poisoning could be used by a realistic adversary that plans to execute a DoS attack in advance; e.g., to lead up to a special event like the Super Bowl or an election.

Multiweek poisoning strategies vary the attack according to the time horizon over which they are carried out. As with single-week attacks, during each week the adversary inserts chaff along the target OD flow throughout the training period according to its poisoning strategy. However, in the multiweek attack the adversary increases the total amount of chaff used during each subsequent week according to a poisoning schedule. This poisons the model over several weeks by initially adding small amounts of chaff and increasing the chaff quantities each week so that the detector is gradually acclimated to chaff and fails to adequately identify the eventually large amount of poisoning—this is analogous to the attacks against the hypersphere detector in Chapter 4. We call this type of episodic poisoning the *boiling frog poisoning attack* after the folk tale that one can boil a frog by slowly increasing the water temperature over time.[3]

The boiling frog poisoning attack can use any of the preceding chaff schemes to select $a^{(t)}$ during each week of poisoning; the only week-to-week change is in the total volume of chaff used, which increases as follows. During the first week, the subspace-based detector is trained on unpoisoned data. In the second week, an initial total volume of chaff is $A^{(1)}$ is selected, and the target flow is injected with chaff generated using a parameter θ_1 to achieve the desired total chaff volume. After classifying the traffic from the new week, PCA is retrained on that week's sanitized data with any detected anomalies removed. During each subsequent week, the poisoning is increased according to its schedule; the schedules we considered increase the total chaff volumes geometrically as $A^{(t)} = \kappa A^{(t-1)}$ where κ is the rate of weekly increase. The goal of boiling frog poisoning is to slowly rotate the normal subspace, injecting low levels of chaff relative to the previous week's traffic levels so that PCA's rejection rates stay low and a large portion of the present week's poisoned traffic matrix is trained on. Although PCA is retrained each week, the training data will include some events not caught by the previous week's detector. Thus, more malicious training data will accumulate each successive week as the PCA subspace is gradually shifted. This process continues until the week of the DoS attack, when the adversary stops injecting chaff and executes its desired DoS; again we measure the success rate of that final attack. Episodic poisoning is considered more fully in Rubinstein (2010), but we summarize the results of this poisoning scheme on subspace detectors in Section 6.4.5.

6.3 Corruption-Resilient Detectors

We propose using techniques from robust statistics to defend against *Causative Integrity* attacks on subspace-based anomaly detection and demonstrate their efficacy in that role. Robust methods are designed to be less sensitive to outliers and are consequently ideal defenses against variance injection schemes that perturb data to increase variance along the target flow. There have been two general approaches to make PCA robust: the first computes the principal components as the eigen-spectrum of a robust estimate of the covariance matrix (Devlin, Gnanadesikan, & Kettenring 1981), while the second approach searches for directions that maximize a robust scale estimate of the data projection. In this section, we propose using a method from the second approach as a defense against poisoning. After describing the method, we also propose a new

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

threshold statistic that can be used for any subspace-based method including robust PCA and better fits their residuals. Robust PCA and the new robust Laplace threshold together form a new network-wide traffic anomaly detection method, ANTIDOTE, that is less sensitive to poisoning attacks.

6.3.1 Intuition

Fundamentally, to mitigate the effect of poisoning attacks, the learning algorithm must be stable despite 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 a related notion of stability often referred to as distributional robustness (see Section 3.5.4.3). 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 (Croux et al. 2007; Croux & Ruiz-Gazen 2005; Devlin et al. 1981; Li & Chen 1985; Maronna 2005). As stated earlier, the approach we selected finds a subspace that maximizes an alternative dispersion measure instead of the usual variance.

The robust PCA algorithms search for a unit direction **v** whose projections maximize some univariate dispersion measure $S\{\cdot\}$ after centering the data according to the location estimator $\hat{\mathbf{c}}\{\cdot\}$; that is,[5]

$$\mathbf{v} \in \underset{\|\mathbf{w}=1\|_2}{\operatorname{argmax}} \left[S\left\{\mathbf{w}^\top \left(\mathbf{x}^{(t)} - \hat{\mathbf{c}}\left\{\mathbf{x}^{(t)}\right\}\right)\right\}\right]. \tag{6.4}$$

The standard deviation is the dispersion measure used by PCA; i.e., $S^{\mathrm{SD}}\left\{r^{(1)}, \ldots, r^{(T)}\right\} = \left(\frac{1}{T-1}\sum_{t=1}^{T}\left(r^{(t)} - \bar{r}\right)^2\right)^{\frac{1}{2}}$ where $\bar{r}$ is the mean of the values $\left\{r^{(t)}\right\}$. However, it is well known that the standard deviation is sensitive to outliers (cf. Hampel et al. 1986, Chapter 2), making PCA nonrobust to contamination. Robust PCA algorithms instead use measures of dispersion based on the concept of *robust projection pursuit (RPP)* estimators (Li & Chen 1985). As is shown by Li & Chen, RPP estimators achieve the same breakdown points as their dispersion measure (recall that the breakdown point is the [asymptotic] fraction of the data an adversary must control to arbitrarily change an estimator and 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 estimators must be approximated. In the next section, we describe the PCA-GRID algorithm, a successful method for approximating robust PCA subspaces developed by Croux et al.

[4] Dispersion is an alternative term for variation since the latter is often associated with statistical variation. A dispersion measure is a statistic that measures the variability or spread of a variable according to a particular notion of dispersion.

[5] Here we use the notation $g\{r^{(1)}, \ldots, r^{(T)}\}$ to indicate that the function g acts on an enumerated set of objects. This notation simplifies the notation $g(\{r^{(1)}, \ldots, r^{(T)}\})$ to a more legible form.

(2007). Of several other projection pursuit techniques (Croux & Ruiz-Gazen 2005; Maronna 2005), PCA-Grid proved to be most resilient to our poisoning attacks. It is worth emphasizing that the procedure described in the next section is 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 Equation (6.4).

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. Figure 6.2 shows an example of the impact that a globally informed poisoning attack has on both algorithms. As demonstrated in Figure 6.2(a), initially the data was approximately clustered in an ellipse, and both algorithms construct reasonable estimates for the center and first principal component for this data. However, Figure 6.2(b) shows that a large amount of poisoning dramatically perturbs some of the data in the direction of the target flow, 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 considerably less affected by the poisoning and only rotates slightly toward the direction of the target flow.

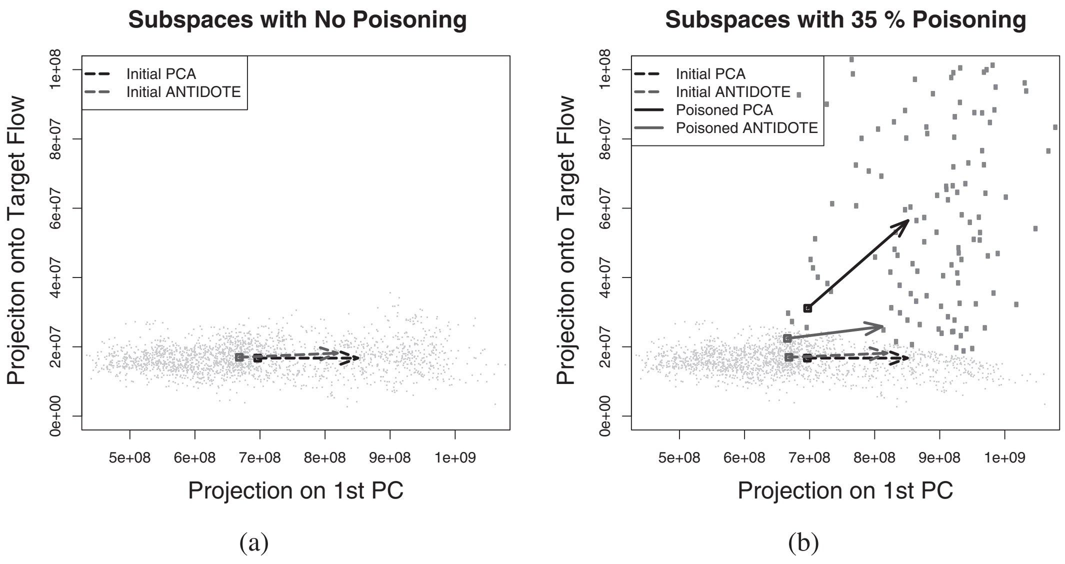


Figure 6.2 In these figures, the Abilene data was projected into the 2D space spanned by the first principal component and the direction of the attack flow #118. **(a)** The first principal component learned by PCA and PCA-Grid on clean data (represented by small gray dots). **(b)** The effect on the first principal components of PCA and PCA-Grid is shown under a globally informed attack (represented by ○'s). Note that some contaminated points were too far from the main cloud of data to include in the plot.

6.3.2 PCA-Grid

The PCA-Grid algorithm introduced by Croux et al. (2007) is a projection pursuit technique as described in Equation 6.4. It finds a K-dimensional subspace that approximately maximizes $S\{\cdot\}$, a *robust* measure of dispersion, for the data $\mathbf{X}$ as in Equation (6.4). The robust measure of dispersion used by Croux et al. and also incorporated into Antidote is the well-known MAD estimator because of its high degree of distributional robustness—it attains the highest achievable breakdown point of $\epsilon^\star = 50\%$ and is the most robust M-estimator of dispersion (cf. Hampel et al. 1986, Chapter 2). For scalars $r^{(1)}, \ldots, r^{(T)}$ the MAD is defined as

$$\begin{aligned} \mathrm{MAD}\left\{r^{(1)}, \ldots, r^{(T)}\right\} &= \mathrm{median}\left\{\left|r^{(i)} - \mathrm{median}\left\{r^{(1)}, \ldots, r^{(T)}\right\}\right|\right\} \\ S^{\mathrm{MAD}}\left\{r^{(1)}, \ldots, r^{(T)}\right\} &= \omega \cdot \mathrm{MAD}\left\{r^{(1)}, \ldots, r^{(T)}\right\}, \end{aligned} \tag{6.5}$$

where the coefficient $\omega = \frac{1}{\Phi^{-1}(3/4)} \approx 1.4826$ rescales the MAD so that $S^{\mathrm{MAD}}\{\cdot\}$ is an estimator of the standard deviation that is asymptotically consistent for 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 also not a robust estimator, so we center the data using the spatial median instead:

$$\hat{\mathbf{c}}\left\{\mathbf{x}^{(t)}\right\} \in \operatorname*{argmin}_{\boldsymbol{\mu} \in \Re^D} \sum_{t=1}^{T} \left\|\mathbf{x}^{(t)} - \boldsymbol{\mu}\right\|_2,$$

which is a convex optimization that can be efficiently solved using techniques developed by Hössjer & Croux (1995).

After centering the data based on the location estimate $\hat{\mathbf{c}}\left\{\mathbf{x}^{(t)}\right\}$ obtained above, PCA-Grid finds a unitary direction $\mathbf{v}$ that is an approximate solution to Equation (6.4) for the scaled MAD dispersion measure. The PCA-Grid algorithm uses a grid search for this task. To motivate this search procedure, suppose one wants to find the best candidate between some pair of unit vectors $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ (a 2D search space). The search space is the unit circle parameterized by ϕ as $\mathbf{w}(\phi) = \cos(\phi)\,\mathbf{w}^{(1)} + \sin(\phi)\,\mathbf{w}^{(2)}$ with $\phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. The grid search splits the domain of ϕ into a mesh of $G+1$ candidates $\phi^{(k)} = \frac{\pi}{2}\left(\frac{2k}{G} - 1\right)$, $k = 0, \ldots, G$. Each candidate vector $\mathbf{w}\left(\phi^{(k)}\right)$ is assessed, and the one that maximizes $S\left\{\left(\mathbf{x}^{(t)}\right)^\top \mathbf{w}\left(\phi^{(k)}\right)\right\}$ is selected as the approximate maximizer $\hat{\mathbf{w}}$.

To search a more general D-dimensional space, the search iteratively refines its current best candidate $\hat{\mathbf{w}}$ by performing a grid search between $\hat{\mathbf{w}}$ and each of the unit directions $\mathbf{e}^{(j)}$ with $j \in 1 \ldots D$. With each iteration, the range of angles considered progressively narrows around $\hat{\mathbf{w}}$ to better explore its neighborhood. This procedure (outlined in Algorithm 6.1) approximates the direction of maximal dispersion analogous to an eigenvector in PCA.

To find the K-dimensional subspace $\left\{\mathbf{v}^{(k)} \;\middle|\; \forall j = 1, \ldots, K\ (\mathbf{v}^{(k)})^\top \mathbf{v}^{(j)} = \delta_{k,j}\right\}$ that maximizes the dispersion measure, the Grid-Search is repeated K-times. After each

ALGORITHM 6.1 GRID-SEARCH $(\mathbf{X})$

```
Require: $\mathbf{X}$ is a $T \times D$ matrix
  $\hat{\mathbf{v}} \leftarrow \mathbf{e}^{(1)}$
  for $i = 1$ to $C$ do begin
    for $j = 1$ to $D$ do begin
      for $k = 0$ to $G$ do begin
        $\phi^{(k)} \leftarrow \frac{\pi}{2^i}\left(\frac{2k}{G} - 1\right)$
        $\mathbf{w}\left(\phi^{(k)}\right) \leftarrow \cos\left(\phi^{(k)}\right)\hat{\mathbf{w}} + \sin\left(\phi^{(k)}\right)\mathbf{e}^{(j)}$
        if $S\left\{\left(\mathbf{x}^{(t)}\right)^\top \mathbf{w}\left(\phi^{(k)}\right)\right\} > S\left\{\left(\mathbf{x}^{(t)}\right)^\top \hat{\mathbf{v}}\right\}$ then $\hat{\mathbf{v}} \leftarrow \mathbf{w}\left(\phi^{(k)}\right)$
      end for
    end for
  end for
  return: $\hat{\mathbf{v}}$
```

ALGORITHM 6.2 PCA-GRID $(\mathbf{X}, K)$

```
  Center $\mathbf{X}$: $\mathbf{X} \leftarrow \mathbf{X} - \hat{\mathbf{c}}\left\{\mathbf{x}^{(t)}\right\}$
  for $i = 1$ to $K$ do begin
    $\mathbf{v}^{(k)} \leftarrow$ GRID-SEARCH $(\mathbf{X})$
    $\mathbf{X} \leftarrow$ projection of $\mathbf{X}$ onto the complement of $\mathbf{v}^{(k)}$
  end for
  Return subspace centered at $\hat{\mathbf{c}}\left\{\mathbf{x}^{(t)}\right\}$ with principal directions $\left\{\mathbf{v}^{(k)}\right\}_{k=1}^{K}$
```

repetition, the data is deflated to remove the dispersion captured by the last direction from the data. This process is detailed in Algorithm 6.2.

6.3.3 Robust Laplace Threshold

In addition to the robust PCA-GRID algorithm, we also design a robust estimate for its residual threshold that replaces the Q-statistic described in Section 6.1.2. The use of the Q-statistic as a threshold by Lakhina et al. was implicitly motivated by an assumption of normally distributed residuals (Jackson & Mudholkar 1979). 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 a detection threshold. The non-normality of the residuals was also observed by Brauckhoff et al. (2009). Instead, to account for the outliers and heavy-tailed behavior we observed from our method's residuals, we choose the threshold as the $1 - \beta$ quantile of a Laplace distribution fit with robust location and scale parameters. The alternative subspace-based anomaly detector, ANTIDOTE, is the combination of the PCA-GRID algorithm for normal-subspace estimation and the Laplace threshold to estimate the threshold for flagging anomalies.

As with the Q-statistic described in Section 6.1.2, we construct the Laplace 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 squared residuals $\left\{\|\ddot{\mathbf{x}}^{(t)}\|_2^2\right\}$ using robust consistent estimates $\hat{c}$ and $\hat{b}$ of location (median) and scale (MAD), respectively,

$$\hat{c} = \text{median}\left\{\|\ddot{\mathbf{x}}^{(t)}\|_2^2\right\}$$

$$\hat{b} = \frac{1}{\sqrt{2}P^{-1}(0.75)} \text{MAD}\left\{\|\ddot{\mathbf{x}}^{(t)}\|_2^2\right\}$$

where $P^{-1}(q)$ is the q^{th} quantile of the standard Laplace distribution. The Laplace quantile function has the form $P^{-1}_{c,b}(q) = c + b \cdot k_L(q)$ for the function $k_{Laplace}$ that is independent of the location and shape parameters of the distribution.[6] Thus, the Laplace 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 its standard quantile function k_{Normal}), but because nonrobust estimates for c and b are implicitly used by the Q-statistic, it is not robust. Further, by choosing the heavy-tailed Laplace distribution, the quantiles are more appropriate for the observed heavy-tailed behavior, but the robustness of this threshold is due to robust parameter estimation.

Empirically, the Laplace threshold also proved to be better suited for thresholding the residuals for ANTIDOTE than the Q-statistic. Figure 6.3(a) shows that both the Q-statistic and the Laplace threshold produce a reasonable threshold on the residuals of the PCA algorithm; however, as seen in Figure 6.3(b), the Laplace threshold produces a reasonable threshold for the residuals of the PCA-GRID algorithm, whereas the Q-statistic vastly underestimates the spread of the residuals. In the experiments described in the next section, the Laplace threshold is consistently more reliable than the Q-statistic.

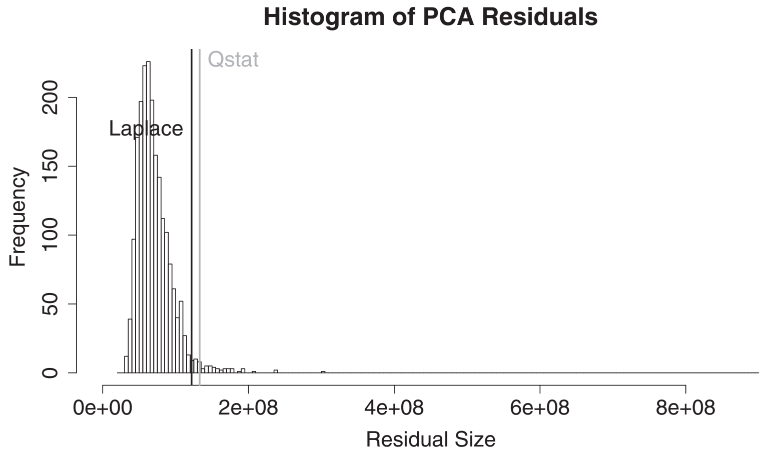
6.4 Empirical Evaluation

We evaluate how the performance of PCA-based methods is affected by the poisoning strategies described in Section 6.2. We compare the original PCA-based detector and the alternative ANTIDOTE detector under these adversarial conditions using a variety of performance metrics.

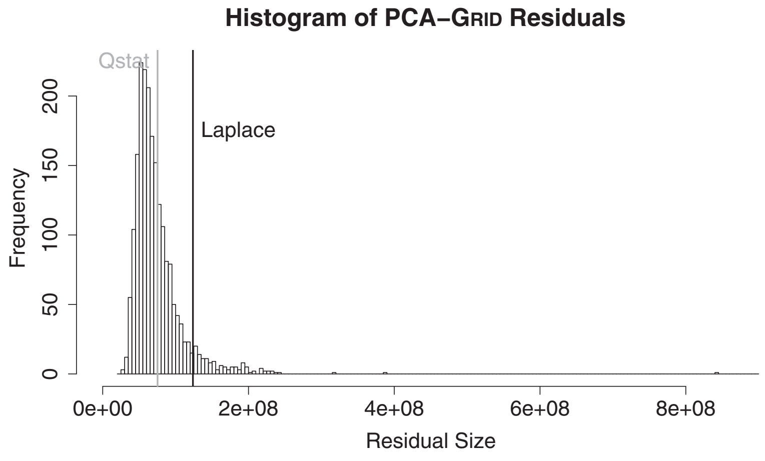
6.4.1 Setup

To assess the effect of poisoning, we test their performance under a variety of poisoning conditions. Here we describe the data used for that evaluation, the method used to test the detectors, and the different types of poisoning scenarios used in their evaluation.

[6] For the Laplace distribution, this function is given by $k_L(q) \triangleq \text{sign}\left(q - \frac{1}{2}\right) \cdot \ln\left(1 - 2\left|q - \frac{1}{2}\right|\right)$.



(a)



(b)

Figure 6.3 A comparison of the Q-statistic and the Laplace threshold for choosing an anomalous cutoff threshold for the residuals from an estimated subspace. **(a)** Histograms of the residuals for the original PCA algorithm and **(b)** of the PCA-GRID algorithm (the largest residual is excluded as an outlier). Pale and dark vertical lines demarcate the threshold selected using the Q-statistic and the Laplace threshold (see the labels adjacent to the vertical lines). For the original PCA method, both methods choose nearly the same reasonable threshold to the right of the majority of the residuals. However, for the residuals of the PCA-GRID subspace, the Laplace threshold is reasonable, whereas the Q-statistic is not; it would misclassify too much of the normal data to be an acceptable choice.

6.4.1.1 Traffic Data

The dataset we use for evaluation is OD flow data collected from the Abilene (Internet2 backbone) network to simulate attacks on PCA-based anomaly detection. This data was collected over an almost continuous 6-month period from March 1, 2004, through

September 10, 2004 (Zhang, Ge, Greenberg, & Roughan 2005). Each week of data consists of 2016 measurements across all 144 network OD flows binned into five-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.

6.4.1.2 Validation

Although there are a total of 24 weeks of data in the dataset, these experiments are primarily based on the 20th and 21st weeks that span the period from August 7–20, 2004. These weeks were selected because PCA achieved the lowest false-negative rate on these during testing, and thus this data was most ideal for the detector. To evaluate a detector, it is trained on the 20th week's traffic and tested on the data from the 21st week during which DoS attacks are injected to measure how often the attacker can evade detection. To simulate the single training period attacks, the training traffic from week 21 is first poisoned by the attacker.

To evaluate the impact of poisoning on the ability of the original PCA-subspace method and Antidote to detect DoS attacks, two consecutive weeks of data are used (again, the subsequent results use the 20th and 21st weeks)—the first for training and the second for testing. The poisoning occurs throughout the training phase, while the DoS attack occurs during the test week. An alternate evaluation method (described in detail later) is needed for the boiling frog poisoning attack scheme where training and poisoning occur over multiple weeks. The success of the poisoning strategies is measured by their impact on the subspace-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 tradeoff between the true-positive rate (TPR) and the false-positive rate (FPR).

To compute the FNRs and FPRs, synthetic anomalies are generated according to the method of Lakhina et al. (2004*b*) and are injected 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, it is convenient for the purposes of relative comparison between PCA and robust PCA, to measure relative effects of poisoning, and for consistency with prior studies. The training sets used in these experiments consist of week-long traffic traces; a week is a sufficiently long time scale to capture weekday and weekend cyclic trends (Ringberg et al. 2007), and it is also the same time scale used in previous studies (Lakhina et al. 2004*b*). Because the data is binned into five-minute windows (corresponding to the reporting interval of SNMP), a decision about whether or not an anomaly occurred can be made at the end of each window; thus attacks can be detected within five minutes of their occurrence.

Unfortunately, computing the false-positive rate of a detector is difficult since there may be actual anomalous events in the Abilene data. To estimate the FPR, negative examples (benign OD flows) are generated as follows. The data is fit to an EWMA model that is intended to capture the main trends of the data with little noise. This model is used to select points in the Abilene flow's time series to use as negative examples. The actual data is then compared to the EWMA model; if the difference is small (not in the flow's top one percentile) for a particular flow at a particular time, $Q_{t,q}$, then the element $Q_{t,q}$ is labeled as *benign*. This process is repeated across all flows. The FPR of a detector is finally estimated based on the (false) alarms raised on the time slots that were deemed to be benign.

DoS attacks are simulated by selecting a target flow, q, and time window, t, and injecting a traffic spike along this target flow during the time window. Starting with the flow traffic matrix $\mathbf{Q}$ for the test week, a positive example (i.e., an anomalous flow event) is generated by setting the q^{th} flow's volume at the t^{th} time window, $Q_{t,q}$, to be a large value known to correspond to an anomalous flow (replacing the original traffic volume in this time slot). This value was defined by Lakhina et al. (2004*b*) to be 1.5 times a cutoff of 8×10^7. After multiplying by the routing matrix $\mathbf{R}$, the link volume measurement at time t is anomalous. This process is repeated for each time t (i.e., each five-minute window) in the test week to generate 2016 anomalous samples for the q^{th} target flow.

A DoS attack is simulated along every flow at every time, and the detector's alarms are recorded for each such attack. The FNR is estimated by averaging over all 144 flows and all 2016 time slots. When reporting the effect of an attack on traffic volumes, we first average over links within each flow and then over flows. Furthermore, we generally report average volumes relative to the preattack average volumes. Thus, a single poisoning experiment was based on one week of poisoning with FNRs computed during the test week that includes $144 \times 2{,}016$ 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.

The squared prediction errors produced by the detection methods (based on the anomalous and normal examples from the test set) are used to produce ROC curves. By varying the method's threshold from $-\infty$ to ∞ a curve of possible (FPR, TPR) pairs is produced from the set of squared prediction errors; the Q-statistic and Laplace threshold each correspond to one such point in ROC space. We adopt the area under curve (AUC) statistic to directly compare ROC curves. The ideal detector has an AUC of 1, while the random predictor achieves an AUC of $\frac{1}{2}$.

6.4.2 Identifying Vulnerable Flows

There are two ways that a flow can be vulnerable. A flow is considered *vulnerable to DoS attack* (unpoisoned scenario) if a DoS attack along it is likely to be undetected when the resulting traffic data is projected onto the abnormal subspace. *Vulnerability to poisoning* means that if the flow is first poisoned, then the subsequent DoS attack is likely to be undetected because the resulting projection in abnormal space is no longer significant. To examine the vulnerability of flows, we define the residual rate statistic, which measures the change in the size of the residual (i.e., $\Delta \|\tilde{\mathbf{x}}\|_2$) caused by adding a single unit of traffic volume along a particular target flow. This statistic assesses how vulnerable a detector is to a DoS attack because it measures how rapidly the residual grows as the size of the DoS increases and thus is an indicator of whether a large DoS attack will be undetected along a target flow. Injecting a unit volume along the q^{th} target flow causes an additive increase to the link measurement vector $\mathbf{R}_q$ and also increases the residual by

$$\nu\left(q; \ddot{\mathbf{P}}\right) \triangleq \left\|\ddot{\mathbf{P}}\mathbf{R}_q\right\|_2 .$$

The residual rate measures how well a flow aligns with the normal subspace. If the flow aligns perfectly with the normal subspace, its residual rate will be 0 since changes along the directions of the subspace do not change the residual component of the traffic at all. More generally, a low residual rate indicates that (per unit of traffic sent) a DoS attack will not significantly affect the squared prediction error. Thus, for a detector to be effective, the residual rate must be high for most flows; otherwise the attacker will be able to execute large undetected DoS attacks.

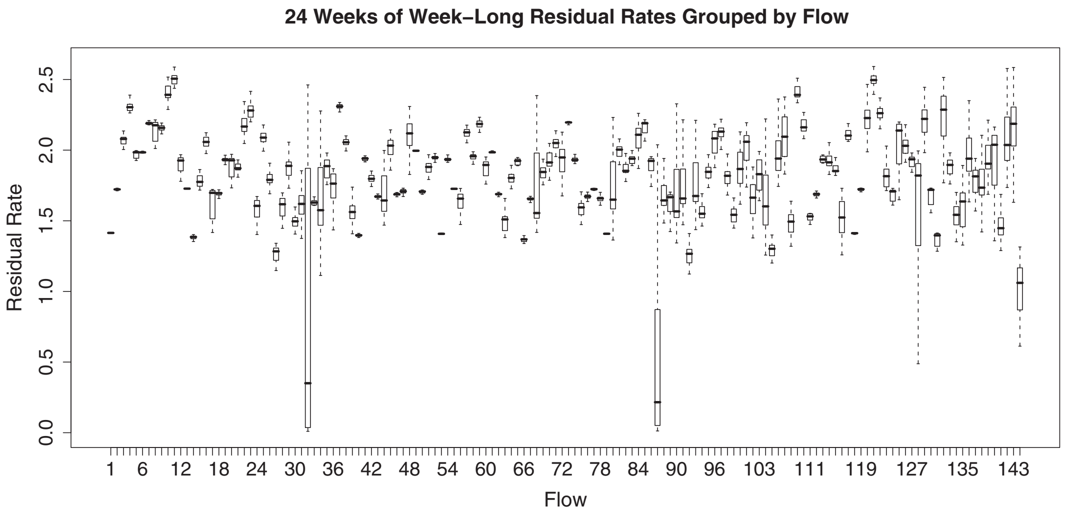
By running PCA on each week of the Abilene data, we computed the residual rate of each flow for each week's model and estimated the spread in their residual rates. Figure 6.4 displays box plots of the residual rates for each flow over the 24 weeks of data. These plots show that when trained on uncontaminated data 99% of the flows have a median residual rate above 1.0; i.e., for every unit of traffic added to any of these flows in a DoS attack, the residual component of the traffic increases by at least 1.0 unit and for many flows the increase is higher.[7] This result indicates that PCA trained on clean data is not vulnerable to DoS attacks on the majority of flows since each unit of traffic used in the attack increases the residual by at least one unit. However, PCA is very vulnerable to DoS attacks along flows 32 and 87 because their residual rates are small even without poisoning.

All of this is good news from the point of view of the attacker. Without poisoning, an attacker might only succeed if it were lucky enough to be attacking along the two highly vulnerable flows. However, after poisoning, it is clear that whatever the attack's target might be, the flow it chooses to attack, on average, is likely to be vulnerable.

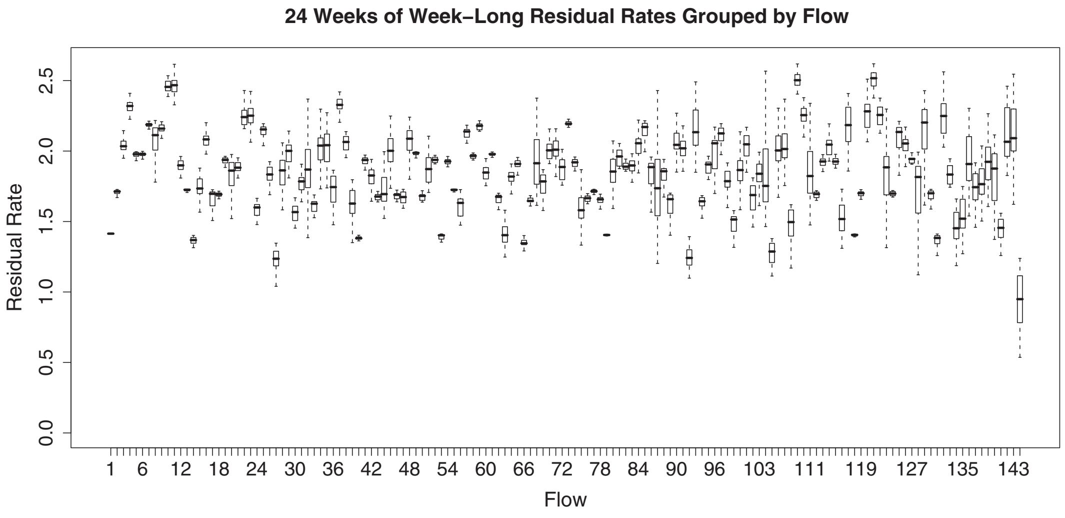
6.4.3 Evaluation of Attacks

In this section, we present experimental validation that adversarial poisoning can have a significant detrimental impact on the PCA-based anomaly detector. We evaluate the effectiveness of the three data poisoning schemes from Section 6.2 for single training period attacks. During the testing week, the attacker launches a DoS attack in each five-minute time window. The results of these attacks are displayed in Figure 6.5(a). Although the objective of these poisoning schemes is to add variance along the target flow, the mean of the target OD flow being poisoned increases as well, increasing the

[7] Many flows have residual rates well above 1 because these flows traverse many links, and thus adding a single unit of traffic along the flow adds many units in link space. On average, flows in the Abilene dataset have 4.5 links per flow.

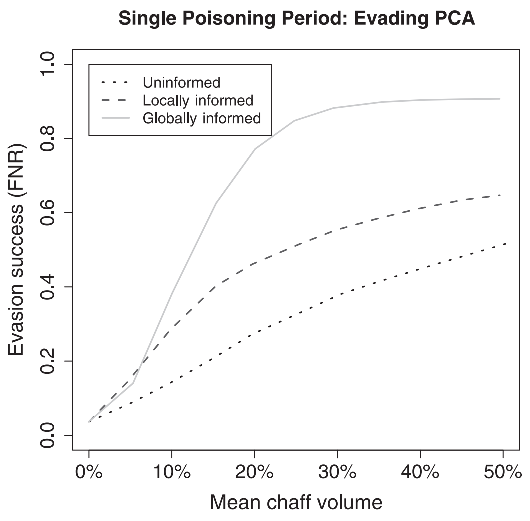


(a) Residual Rates for PCA

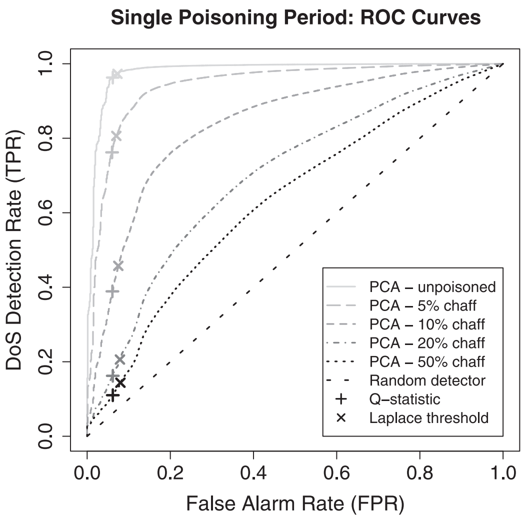


(b) Residual Rates for PCA-Grid

Figure 6.4 Comparison of the original PCA subspace and PCA-Grid subspace in terms of their residual rates. Shown here are box plots of the 24 weekly residual rates for each flow to demonstrate the variation in residual rate for the two methods. **(a)** Distribution of the per-flow residual rates for the original PCA method and **(b)** for PCA-Grid. For PCA, flows 32 and 87 (the flows connecting Chicago and Los Angeles in Figure 6.1(b)), have consistently low residual rates, making PCA susceptible to evasion along these flows. Both methods also have a moderate susceptibility along flow 144 (the ingress/egress link for Washington). Otherwise, PCA-Grid has overall high residual rates along all flows, indicating little vulnerability to evasion.



(a) Impact of Chaff on FNR



(b) Impact of Chaff on ROC Curves

Figure 6.5 Effect of single training period poisoning attacks on the original PCA-based detector. **(a)** Evasion success of PCA versus relative chaff volume under single training period poisoning attacks using three chaff methods: uninformed (dotted line), locally informed (dashed line), and globally informed (solid line). **(b)**. Comparison of the ROC curves of PCA for different volumes of chaff (using add-more-if-bigger chaff). Also depicted are the points on the ROC curves selected by the Q-statistic and Laplace threshold, respectively.

means of all links over which the OD flow traverses. The x-axis in Figure 6.5 is the relative increase in the mean rate. The y-axis is the average FNR for that level of poisoning (i.e., averaged 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 its evasion success to 28% from the baseline FNR of 3.67% via a 10% average increase in the mean link rates due to chaff; i.e., the attacker's rate of successful evasion increases nearly eightfold from the rate of the unpoisoned PCA detector. With a globally informed strategy, a 10% average increase in the mean link rates causes the unpoisoned FNR to increase by a factor of 10 to 38% success and eventually to over 90% FNR as the size of the attack increases. The primary difference between the performance of the locally informed and globally informed attacker is intuitive to understand. Recall that the globally informed attacker is privy to the traffic on all links for the entire training period while the locally informed attacker only knows the traffic status of a single ingress link. Considering this information disparity, the locally informed adversary is quite successful 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 to obtain a 90% evasion success is dangerous in that the poisoning activity itself is likely to be detected. Therefore add-more-if-bigger offers a nice tradeoff, from the adversary's point of view, in terms of poisoning effectiveness, and the attacker's capabilities and risks.

We also evaluate the PCA detection algorithm on both anomalous and normal data, as described in Section 6.4.1.2, to produce the ROC curves in Figure 6.5(b). We produce a series of ROC curves (as shown) by first training a PCA model on the unpoisoned data from the 20th week and then training on data poisoned by progressively larger add-more-if-bigger attacks.

To validate PCA-based detection on poisoned training data, each flow is poisoned separately in different trials of the experiment 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 Figure 6.5(b) display the averages of these ROC curves; i.e., the average TPR over the 144 flows for each FPR.

The sequence of ROC curves show that the add-more-if-bigger poisoning scheme creates an unacceptable tradeoff 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 a random detector (the $y = x$ line with an AUC of $1/2$).

6.4.4 Evaluation of Antidote

We assess the effect of poisoning attacks on Antidote performance during a single training period. As with the PCA-based detector, we evaluate the success of this detector with each of the different poisoning schemes and compute ROC curves using the add-more-if-bigger poisoning scheme to compare to the original PCA-subspace method.

Figure 6.6(a) depicts Antidote's FNR for various levels of average poisoning that occur in a single training period attack compared to the results depicted in Figure 6.5(a) using the same metric for the original PCA detector. Comparing these results, the evasion success of the attack is dramatically reduced for Antidote. For any particular level of chaff, the evasion success rate of Antidote is approximately half that of the original PCA approach. Interestingly, the most effective poisoning scheme on PCA (globally informed poisoning) is the least effective poisoning scheme against Antidote. The globally informed scheme was designed in an approximately optimal fashion to circumvent PCA, but for the alternative detector, globally informed chaff is not optimized and empirically has little effect on PCA-Grid. 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—since roughly half the data shifts by a constant amount, it is difficult to distinguish between the original and shifted subspaces. However, this effect is still small compared to the dramatic success of locally informed and globally informed chaff strategies against the original detector.

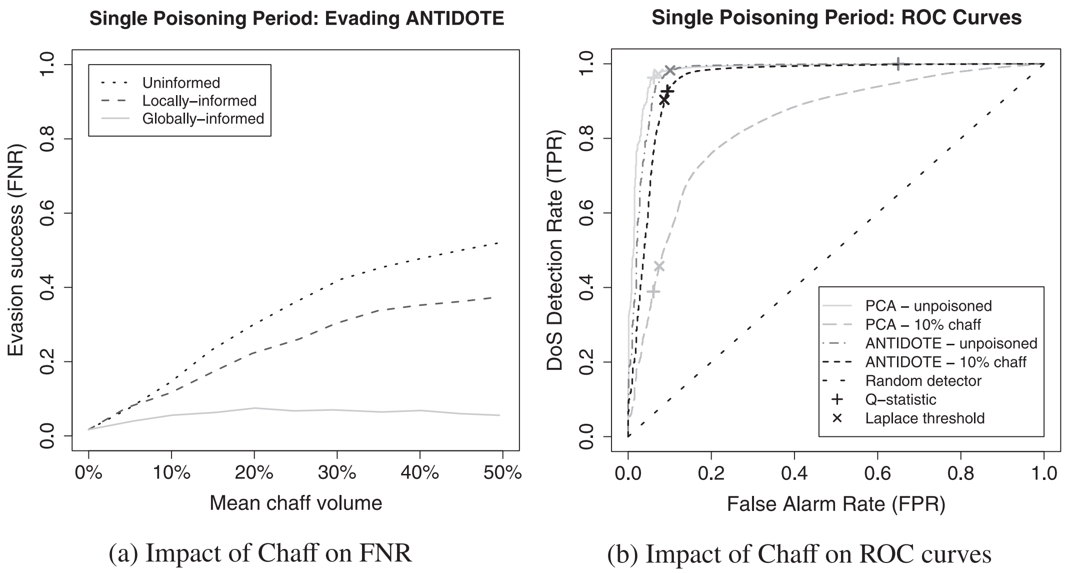


Figure 6.6 Effect of single training period poisoning attacks on the Antidote detector. **(a)** Evasion success of Antidote versus relative chaff volume under single training period poisoning attacks using three chaff methods: uninformed (dotted black line), locally informed (dashed line), and globally informed (solid line). **(b)** Comparison of the ROC curves of Antidote and the original PCA detector when unpoisoned and under 10% chaff (using add-more-if-bigger chaff). The PCA detector and Antidote detector have similar performance when unpoisoned but PCA's ROC curve is significantly degraded with chaff, whereas Antidote's is only slightly affected.

Since poisoning distorts the detector, it affects both the false-negative and false-positive rates. Figure 6.6(b) provides a comparison of the ROC curves for both Antidote and PCA 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 using the add-more-if-bigger strategy. While Antidote performs very similarly to PCA on unpoisoned training data, PCA's performance is significantly degraded by poisoning while Antidote remains relatively unaffected. With a moderate mean chaff volume of 10%, Antidote's average ROC curve remains close to optimal, while PCA's curve considerably shifts toward the $y = x$ curve of the random detector. This means that under a moderate level of poisoning, PCA cannot achieve a reasonable tradeoff between false positives and false negatives while Antidote retains a good operating point for these two common performance measures. *In summary, in terms of false positives and false negatives,* Antidote *incurs insignificant performance shifts when no poisoning occurs, but is resilient against poisoning and provides enormous performance gains compared to PCA when poisoning attacks do occur.*

Given Figures 6.6(a) and 6.6(b) alone, it is conceivable that Antidote outperforms PCA only on average, and not on all flows targeted for poisoning. In place of plotting

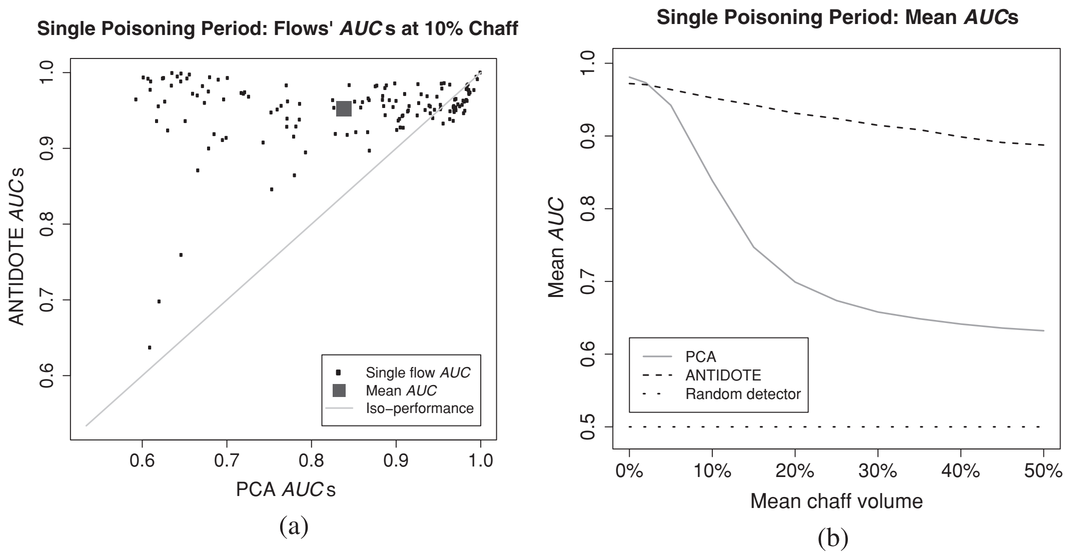


Figure 6.7 Comparison of the original PCA detector in terms of the area under their (ROC) curves (*AUC*s). **(a)** The *AUC* for the PCA detector and the ANTIDOTE detector under 10% add-more-if-bigger chaff for each of the 144 target flows. Each point in this scatter plot is a single target flow; its x-coordinate is the *AUC* of PCA and its y-coordinate is the *AUC* of ANTIDOTE. Points above the line $y = x$ represent flows where ANTIDOTE has a better *AUC* than the PCA detector, and those below $y = x$ represent flows for which PCA outperforms ANTIDOTE. The mean *AUC* for both methods is indicated by the square symbol. **(b)** The mean *AUC* of each detector versus the mean chaff level of an add-more-if-bigger poisoning attack for increasing levels of relative chaff. The methods compared are a random detector (dotted black line), the PCA detector (solid line), and ANTIDOTE (dashed line).

all 144 poisoned ROC curves, Figure 6.7(a) compares the *AUC*s for the two detection methods under 10% chaff. Not only is average performance much better for robust PCA, but it in fact outperforms PCA on most flows and by a decidedly large amount. Although PCA indeed performs slightly better for some flows, in these cases both methods have excellent detection performance (because their *AUC*s are close to 1), and hence the distinction between the two is insignificant for those specific flows.

Figure 6.7(b) plots the mean AUC (averaged from the 144 ROC curves' *AUC*s where flows are poisoned separately) achieved by the detectors for an increasing level of poisoning. ANTIDOTE behaves comparably to (albeit slightly worse than) PCA under no-chaff conditions, yet its performance remains relatively stable as the amount of contamination increases while PCA's performance rapidly degrades. In fact, with as little as 5% poisoning, ANTIDOTE already exceeds the performance of PCA, and the gap only widens with increasing contamination. As PCA's performance drops, it approaches a random detector (equivalently, $AUC = 1/2$), for amounts of poisoning exceeding 20%. As these experiments demonstrate, ANTIDOTE is an effective defense and dramatically outperforms a solution that was not designed to be robust. This is strong evidence that the robust techniques are a promising instrument for designing machine learning algorithms used in security-sensitive domains.

6.4.5 Empirical Evaluation of the Boiling Frog Poisoning Attack

6.4.5.1 Experimental Methodology for Episodic Poisoning

To test the boiling frog poisoning attack, several weeks of traffic data are simulated using a generative model inspired by Lakhina, Crovella, & Diot (2004*b*). These simulations produce multiple weeks of data generated from a stationary distribution. While such data is unrealistic in practice, stationary data is the ideal dataset for PCA to produce a reliable detector. Anomaly detection under nonstationary conditions is more difficult due to the learner's inability to distinguish between benign data drift and anomalous conditions. By showing that PCA is susceptibility to episodic poisoning even in this stationary case, these experiments suggest that the method can also be compromised in more realistic settings. Further, the six-month Abilene dataset of Zhang et al. (2005) proved to be too nonstationary for PCA to consistently operate well from one week to the next—PCA often performed poorly even without poisoning. It is unclear whether the nonstationarity observed in this data is prevalent in general or whether it is an artifact of the dataset, but nonetheless, these experiments show PCA is susceptible to poisoning even when the underlying data is well behaved.

To synthesize a stationary multiweek dataset of OD flow traffic matrices, a three-step generative procedure is used to model each OD flow separately. First the underlying daily cycle of the q^{th} OD flow's 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 interarrival 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.

In the first step, the underlying cyclic trends are captured by fitting the coefficients for Fourier basis functions. Following the model proposed by Lakhina et al. (2004*b*), the basis functions are sinusoids of periods of 7, 5, and 3 days, and 24, 12, 6, 3, and 1.5 hours, as well as a constant function. For each OD flow, the Fourier coefficients are estimated by projecting the flow onto this basis. The portion of the traffic modeled by this Fourier forecaster is removed, and the remaining residual traffic is modeled with two processes—a zero-mean Gaussian noise process captures short-term benign traffic variance, and an exponential distribution is used to model nonmalicious volume anomalies.

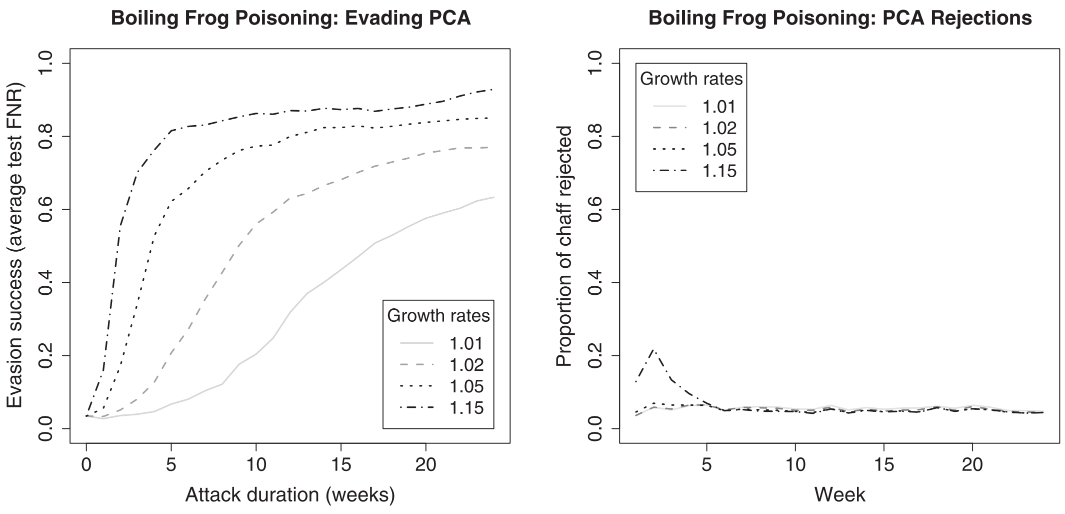
In the second step, one of the two noise processes is selected for each time interval. After computing the Fourier model's residuals (the difference between the observed and predicted traffic), the smallest negative residual value $-m$ is recorded. We assume that residuals in the interval $[-m, m]$ correspond to benign traffic and that residuals exceeding m correspond to traffic anomalies (this is an approximation, but it works reasonably well for most OD flows). Periods of benign variation and anomalies are then modeled separately since these effects behave quite differently. After classifying residual traffic as benign or anomalous, anomaly arrival times are modeled as a Bernoulli arrival process, and the inter-anomaly arrival times are geometrically distributed. Further, since we consider only spatial PCA methods, the temporal placement of anomalies is unimportant.

In the third and 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 small, medium, and large flows.

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

6.4.5.2 Effect of Episodic Poisoning on the PCA Detector

We now evaluate the effectiveness of the boiling frog poisoning attack strategy, that contaminates the training data over multiple training periods. Figure 6.8(a) plots the FNRs against the poisoning duration for the PCA detector for four different *poisoning*



(a) Effect of boiling frog on PCA (b) Rejection of chaff by PCA

Figure 6.8 Effect of boiling frog poisoning attack on the original PCA-subspace detector (see Figure 6.9 for comparison with the PCA-based detector). **(a)** Evasion success of PCA under boiling frog poisoning attack in terms of the average FNR after each successive week of poisoning for four different poisoning schedules (i.e., a weekly geometric increase in the size of the poisoning by factors 1.01, 1.02, 1.05, and 1.15 respectively). More aggressive schedules (e.g., growth rates of 1.05 and 1.15) significantly increase the FNR within a few weeks while less aggressive schedules take many weeks to achieve the same result, but are more stealthy in doing so. **(b)** Weekly chaff rejection rates by the PCA-based detector for the boiling frog poisoning attacks from (a). The detector only detects a significant amount of the chaff during the first weeks of the most aggressive schedule (growth rate of 1.15); subsequently, the detector is too contaminated to accurately detect the chaff.

schedules with growth rates of 1.01, 1.02, 1.05, and 1.15 respectively. The schedule's growth rate corresponds to the rate of increase in the attacked links' average traffic from week to week. The attack strength parameter θ (see Section 6.2) is selected 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 from 3.67% to more than 70% over three 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. Further, in comparing Figure 6.5(a) for a single training period to Figure 6.8(a), the success of boiling frog attacks becomes clear. For the single training period attack, to raise the FNR to 50%, an immediate increase in mean traffic of roughly 18% is required, whereas in the boiling frog attack the same result can be achieved with only a 5% average traffic increase spread across three weeks.

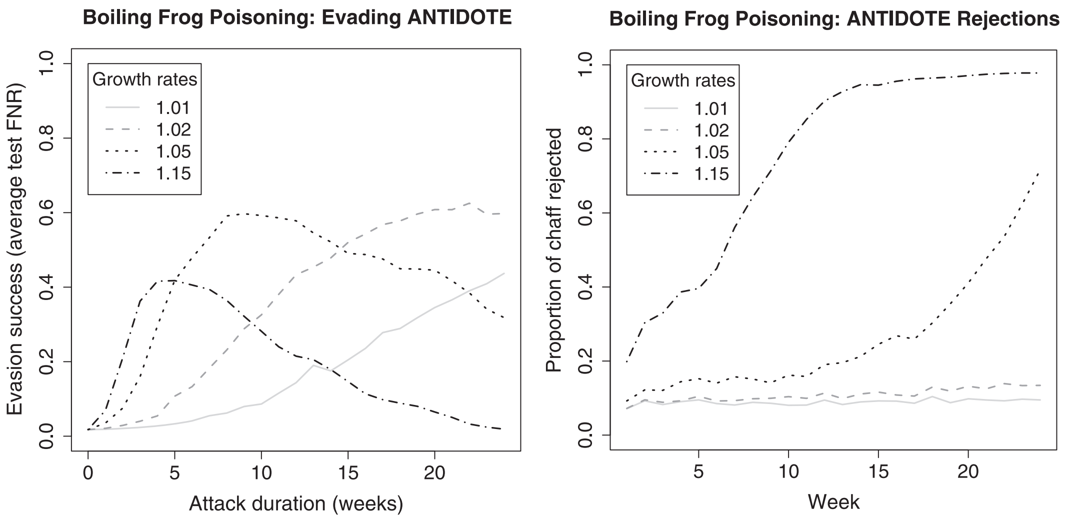
Recall that the two methods are retrained every week using the data collected from the previous week. However, the data from the previous week is also filtered by the detector itself, and for any time window flagged as anomalous, the training data is thrown out. Figure 6.8(b) shows the proportion of chaff rejected each week by PCA (*chaff rejection rate*) for the boiling frog poisoning attack 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 whichpoint the rates drop to the level of the slower schedules. Thus, 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.

6.4.5.3 Effect of Episodic Poisoning on ANTIDOTE

We now evaluate the effectiveness of ANTIDOTE against the boiling frog strategy that occurs over multiple successive training periods. Figure 6.9(a) shows the FNRs for ANTIDOTE with the four poisoning schedules (recall from Section 6.4.5.2 that each is the weekly growth factor for the increase in size of a add-more-if-bigger poisoning strategy). 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 ten training periods it is still below 20% evasion success. This is in stark contrast to PCA (see Figure 6.8(a)); for example, after ten weeks the evasion success against PCA exceeds 50% for the 1.02 poisoning growth rate scenario.

Second, under PCA the evasion success consistently increases with each additional week. However, with ANTIDOTE, the evasion success of these more aggressive schedules actually decreases after several weeks. The reason is that as the chaff levels rise, ANTIDOTE increasingly is able to identify the chaff as abnormal and then reject enough of it from the subsequent training data that the poisoning strategy loses its effectiveness.

Figure 6.9(b) shows the proportion of chaff rejected by ANTIDOTE under episodic poisoning. The two slower schedules almost have a constant rejection rate close to 9% (which is higher than PCA's rejection rate of around 5%). For the more aggressive growth schedules (5% and 15%), however, ANTIDOTE rejects an increasing amount of the poison data. This reflects a good target behavior for any robust detector—to



(a) Effect of Boiling Frog Attack on Antidote (b) Rejection of Chaff by Antidote

Figure 6.9 Effect of boiling frog poisoning attack on the Antidote detector (see Figure 6.8 for comparison with the PCA-based detector). **(a)** Evasion success of Antidote under boiling frog poisoning attacks in terms of the average FNR after each successive week of poisoning for four different poisoning schedules (i.e., a weekly geometric increase in the size of the poisoning by factors 1.01, 1.02, 1.05, and 1.15 respectively). Unlike the weekly FNRs for the boiling frog poisoning in Figure 6.8(a), the more aggressive schedules (e.g., growth rates of 1.05 and 1.15) reach their peak FNR after only a few weeks of poisoning after which their effect declines (as the detector successfully rejects increasing amounts of chaff). The less aggressive schedules (with growth rates of 1.01 and 1.02) still have gradually increasing FNRs, but also seem to eventually plateau. **(b)** Weekly chaff rejection rates by the Antidote detector for the boiling frog poisoning attacks from (a). Unlike PCA (see Figure 6.8(b)), Antidote rejects increasingly more chaff from the boiling frog attack. For all poisoning schedules, Antidote has a higher baseline rejection rate (around 10%) than the PCA detector (around 5%), and it rejects most of the chaff from aggressive schedules within a few weeks. This suggests that, unlike PCA, Antidote is not progressively poisoned by increasing week-to-week chaff volumes.

reject more training data as the contamination grows. *Overall, these experiments provide empirical evidence that the combination of techniques used by* Antidote, *namely a subspace-based detector designed with a robust subspace estimator combined with a Laplace-based cutoff threshold, maintains 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) and thus provides a resilient alternative to the original PCA-based detector.*

6.5 Summary

To subvert the PCA-based detector proposed by Lakhina et al. (2004*b*), we studied *Causative Integrity* attacks that poison the training data by adding malicious chaff; i.e., spurious traffic sent across the network by compromised nodes that reside within it. This chaff is designed to interfere with PCA's subspace estimation procedure. Based on a relaxed objection function, we demonstrated how an adversary can approximate optimal noise using a global view of the traffic patterns in the network. Empirically, we found that by increasing the mean link rate by 10% with globally informed chaff traffic, the FNR increased from 3.67% to 38%—a 10-fold increase in misclassification of DoS attacks. Similarly, by only using local link information the attacker is able to mount a more realistic add-more-if-bigger attack. For this attack, increasing the mean link rate by 10% with add-more-if-bigger chaff traffic, the FNR increased from 3.67% to 28%—an eightfold increase in misclassification of DoS attacks. These attacks demonstrate that with sufficient information about network patterns, adversaries can mount attacks against the PCA detector that severely compromise its ability to detect future DoS attacks traversing the networking it is monitoring.

We also demonstrated that an alternative robust method for subspace estimation could be used instead to make the resulting DoS detector less susceptible to poisoning attacks. The alternative detector was constructed using a subspace method for robust PCA developed by Croux et al. and a more robust method for estimating the residual cutoff threshold. The resulting ANTIDOTE detector is affected by poisoning, but its performance degrades more gracefully. Under nonpoisoned traffic, ANTIDOTE performs nearly as well as PCA, but for all levels of contamination using add-more-if-bigger chaff traffic, the misclassification rate of ANTIDOTE is approximately half the FNR of the PCA-based solution. Moreover, the average performance of ANTIDOTE is much better than the original detector; it outperforms ordinary PCA for more flows and by a large amount. For the multiweek boiling frog poisoning attack ANTIDOTE also outperformed PCA and would catch progressively more attack traffic in each subsequent week.

Several important questions about subspace detection methods remain unanswered. While we have demonstrated that ANTIDOTE is resilient to poisoning attacks, it is not yet known if there are alternative poisoning schemes that significantly reduce ANTIDOTE's detection performance. Because ANTIDOTE is founded on robust estimators, it is unlikely that there is a poisoning strategy that completely degrades its performance. However, to better understand the limits of attacks and defenses, it is imperative to continue investigating worst-case attacks against the next generation of defenders; in this case, ANTIDOTE.

QUESTION 6.1 What are the worst-case poisoning attacks against the ANTIDOTE-subspace detector for large-volume network anomalies? What are game-theoretic equilibrium strategies for the attacker and defender in this setting? How does ANTIDOTE's performance compare to these strategies?

There are also several other approaches for developing effective anomaly detectors for large volume anomalies (e.g., Brauckhoff et al. 2009). To compare these alternatives to ANTIDOTE, one must first identify their vulnerabilities and assess their performance when under attack. More importantly though, we think detectors could be substantially improved by combining them together.

Question 6.2 Can subspace-based detection approaches be adapted to incorporate the alternative approaches? Can they find both temporal and spatial correlations and use both to detect anomalies? Can subspace-based approaches be adapted to incorporate domain-specific information such as the topology of the network?

Developing the next generation of network anomaly detectors is a critical task that perhaps can incorporate several of the themes we promote in this dissertation to create secure learners.