Compromising PCA-based Anomaly Detectors for Network-Wide Traffic

Benjamin I. P. Rubinstein
Blaine Nelson
Ling Huang
Anthony D. Joseph
Shing-hon Lau
Nina Taft
Doug Tygar

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-73
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-73.html

May 29, 2008

Acknowledgement

We would like to thank Marco Barreno for his valuable suggestions. This work was supported in part by TRUST, which receives support from the National Science Foundation (NSF award #CCF-0424422), the Air Force Office of Scientific Research (AFOSR #FA9550-06-1-0244), Cisco, British Telecom, ESCHER, Hewlett-Packard, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies; in part by California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID#06-148, #07-012) and Siemens; and in part by the DETERlab, which receives support from the Department of Homeland Security Advanced Research Projects Agency (HSARPA award #022412) and AFOSR (#FA9550-07-1-0501).

Compromising PCA-based Anomaly Detectors for Network-Wide Traffic

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

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

May 29, 2008

Abstract

The use of machine learning techniques to improve network design is gaining popularity. When these techniques are applied to security problems, a fundamental problem arises; namely that they are susceptible to adversaries who poison the learning phase of such techniques. In this paper we focus on PCA-based anomaly detectors used to identify anomalies in backbone networks via a comprehensive view of the network's traffic. We present four data poisoning schemes and evaluate their effectiveness on increasing an attacker's chance of evading detection. Because machine learning techniques often require retraining when used on data that is evolving, this also opens the door for attackers to employ stealthy poisoning methods that perturb the PCA detector slowly and covertly over time. We demonstrate that some of these PCA-based attacks can increase the adversary's chance of success sixfold under relatively moderate attacks, and comment on possible directions for combating these types of attacks.

1 Introduction

Statistical Machine Learning (SML) techniques are increasingly being used as tools for analyzing and improving network design and performance. In particular, SML plays an important role in dynamic network anomography [6], the problem of inferring network-level Origin-Destination (OD) flow anomalies from aggregate network measurements. Network anomography techniques centrally collect aggregate network measurements and employ various SML techniques [2, 5, 6] to diagnose network traffic anomalies. One popular technique, [2], is based on Principal Components Analysis (PCA).

We are interested in understanding the vulnerabilities associated with using SML-based techniques, specifically how adversaries can subvert the decision-making process. Recently, many researchers have explored attacks against several learning systems—in [1], we summarize this research and describe it with a central framework. In the case of network anomography, we ask the question—can an adversary generate anomalous OD traffic flow patterns that mislead network anomography techniques into missing the anomalous traffic flows?

In this paper, we show that the answer is “yes” for anomography techniques based on PCA [2]. We present four data poisoning schemes that increase the variance of network traffic along the links of a target flow during the training phase of this algorithm. After poisoning PCA’s learning phase, the adversary subsequently launches a large-scale denial of service attack that evades detection. We evaluate the effectiveness of these data poisoning schemes from the attacker’s point of view—their goal is to increase the false negative rates of the detector so that when an attack is launched the chance of evading detection is higher. In the language of our taxonomy of attacks on machine learning systems [1], our focus is on *Causative Targeted Integrity* attacks on PCA-based anomalous OD flow detectors. The term *Causative* refers to the manipulation of the training data, while the term *Integrity* indicates a goal of degrading performance (by increasing false negatives).

We find that simple data poisoning attacks can indeed compromise a PCA-based detector. In some cases, the false negative rates can be increased sixfold as compared to an unpoisoned PCA detector. A key vulnerability of SML techniques is that they often need to be retrained to capture evolving trends in the underlying data. In previous usage scenarios [2, 5], the PCA detector is retrained regularly (e.g., weekly). This opens the door for attackers to carry out the poisoning activity in a stealthy fashion—slowly, yet increasingly, over longer periods of time in what we call a *Boiling Frog* attack. If the principal components of PCA are gradually perturbed, the attacker decreases the chance that the poisoning activity itself is detected. For one method of poisoning, a week’s average link traffic volume, along the target OD flow, must be increased by 18% in order to affect an increase of the false negative rate from 4% to 50%. Under the *Boiling Frog* attack the same result can be achieved with a modest 5% volume increase from week-to-week over a 3 week period.

2 Background

The data that many network anomography detection techniques mine to uncover anomalies is that of the network-wide traffic matrix. This data describes the traffic demand between all pairs of routers, or PoPs, in a backbone network. ISPs gather traffic volume time series for each origin-destination flow. Before describing our poisoning attacks, we first define traffic matrices, present our notation, and then summarize the PCA subspace anomaly detection method of [2].

2.1 Traffic Matrices and Volume Anomalies

Network link traffic represents the superposition of OD flows. Consider a network with N links and F OD flows and we 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 a $N \times F$ indicator 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 focus on the problem of detecting *OD flow volume anomalies* across a top-tier network, by observing only link traffic volumes. Anomalous flow volumes are unusual traffic load levels in a network that are 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 the sequel.

2.2 Subspace Method for Anomaly Detection

We briefly summarize the PCA-based anomaly detector introduced in [2], where the authors observed that due to the high level of traffic aggregation on ISP backbone links, OD flow volume anomalies often go unnoticed by being "buried" within normal traffic patterns. They also observe that although the measured data has high dimensionality N, normal traffic patterns actually 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. Since PCA reveals that the OD flow traffic matrices of ISP backbones have low intrinsic dimensionality, and link and end-to-end traffic demands are linearly related, the ensemble of all link traffic in a network also exhibits low dimensionality—for the Abilene backbone network, most data 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. After adjusting $\mathbf{Y}$ to have zero mean columns, the k^{th} principal component is computed as

$$\mathbf{v}_k = \underset{\mathbf{w}:\|\mathbf{w}\|=1}{\operatorname{argmax}} \left\| \mathbf{Y} \left(\mathbb{I} - \sum_{i=1}^{k-1} \mathbf{v}_i \mathbf{v}_i^\top \right) \mathbf{w} \right\| \ . \tag{1}$$

The resulting K-dimensional subspace spanned by the first K principal components $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K]$ is called the *normal* traffic subspace $\mathcal{S}_n$. The remaining $(N - K)$-dimensional subspace constitutes the *abnormal* traffic subspace $\mathcal{S}_a$.

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.

3 Attacks on PCA

Consider the scenario of a smart adversary who knows an ISP uses a PCA-based anomaly detector. The adversary will poison the training data so that the detector learns a distorted set of principal components. As a result, when the attacker later launches an attack, the PCA-based detector will fail to detect it.

3.1 The Threat Model

In our threat model, the adversary's goal is to launch a successful Denial of Service (DoS) attack on a sink point-of-presence (PoP) node B from a source PoP node D. Fig. 1 illustrates a simple PoP-to-PoP topology. Each PoP has ingress links over which it receives client traffic. This client traffic is eventually delivered to destination clients via egress links that leave the sink PoP. For the clients depicted in the figure, the traffic flows from PoP source node D to PoP sink node B. We refer to the first inter-PoP link in the path of the origin-destination flow OD, inside the ISP, as the *source link*.

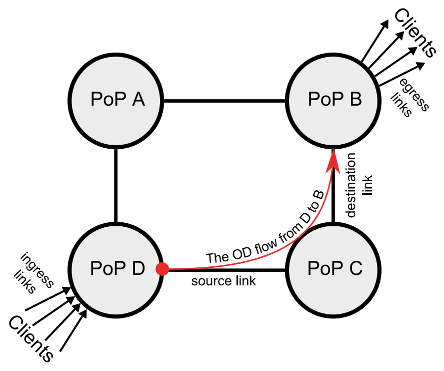

Figure 1: Links used for data poisoning

We make some assumptions about our adversary. First, we assume the attack injects additional traffic, *chaff*, along the OD flow they intend to attack. The adversary can inject additional traffic at node D directed to sink PoP B along the source link. Such a poisoning scheme is possible if the adversary gains control of the core router inside PoP D to which the inter-PoP link is attached. Second, we assume the adversary has access to the real-time link traffic volumes along the source link. Some of our attacks make use of this information so the adversary can insert reasonable amounts of additional traffic not dissimilar from the baseline traffic.

The adversary does not have control over existing traffic; i.e., they cannot delay or discard traffic passing through D. Similarly, the adversary cannot submit false SNMP reports to PCA. Such approaches to poisoning are infeasible because the inconsistencies in SNMP reporting from neighboring PoPs could expose the compromised router. We do not assume that the attacker knows anything about traffic volumes for links not incident to D or network flow-level volumes, or even knowledge of future traffic levels.

We assume that the PCA detector is trained on initial traffic matrix data, and then the principal components (PCs) learned are subsequently used for anomaly detection. Each week the detector relearns the PCs; thus, the PCs used in any week m are those learned during week $m - 1$. Hence in all but one attack scenario, we assume that the adversary inserts chaff along the target OD flow throughout the one week training period.

3.2 Chaff Selection

We now describe several simple methods to insert chaff into the target OD flow from source PoP D to sink B. The common theme throughout these methods is to increase the traffic variance along the target OD flow so that the normal subspace learned by PCA better aligns with the target flow. This makes the residual of the target anomaly small, reducing detection likelihood. Note that in the case of chaff that is statistically independent to the non-attack traffic, the increase to variance due to data poisoning is the variance of the chaff process. For the poisoning attack to be feasible, the adversary must minimize the overall amount of chaff added to the target links. A large increase to traffic volumes would make chaff insertion process itself detectable by naive methods.

With these constraints on the adversary in mind, we consider several poisoning strategies. In each, the adversary adds an extra volume of c_t chaff traffic to the target flow time series, possibly dependent on $\mathbf{y}_S(t)$. Each strategy has an attack parameter θ, which impacts the intensity of the attack. We next describe and analyze several strategies for choosing c_t.

Half Normal Chaff. In the *Half Normal* poisoning method, at time t the attacker selects $c_t = |n_t|$ where $n_t \sim \mathcal{N}(0, \theta^2)$ is a random variable, generated from a zero-mean Gaussian distribution with standard deviation θ. The method's name derives from the distribution of the c's—the Half Normal distribution. Here $\theta > 0$ is an attack parameter. In this scheme, chaff is added in each time interval, and the chaff is independent of the link traffic $\mathbf{y}_S(t)$. The c's have mean $\sqrt{2/\pi}\theta \approx 0.80\theta$ and variance $(1 - 2/\pi)\theta^2 \approx 0.36\theta^2$.

Scaled Bernoulli Chaff. In the *Bernoulli* poisoning method, the adversary injects chaff at time t of size $c_t = \theta b_t$, where $\theta > 0$ is the attack parameter, and the b_t's are Bernoulli random variables taking on values in $\{0, 1\}$ with equal probability. Thus the adversary only adds a constant amount of chaff at select time intervals. Note that, of all mean $\theta/2$ distributions with support in $[0, \theta]$, the distribution with maximum variance assigns all of its mass to points $\{0\}$ and $\{\theta\}$ evenly, as in this poisoning method. The *Bernoulli* chaff has a relatively low variance of $0.25\theta^2$ and mean of 0.5θ.

The Add-Constant-If-Big Method. In the *Add-Constant-If-Big* poisoning method, the adversary injects a constant volume $c_t = \theta > 0$ of chaff at each point in time when the natural traffic $\mathbf{y}_S(t)$ exceeds a threshold α. We restrict our attention to α equal to the mean of the week's link traffic $\{\mathbf{y}_S(t)\}_{t=1}^T$ assuming the mean is relatively stationary and known by the attacker. Thus θ is again the attack parameter. The intuition behind this scheme is to increase variance by adding chaff only when the flow's volume is already high.

The Add-More-If-Bigger Method. Finally, the *Add-More-If-Bigger* poisoning method avoids the large sudden increases (by θ) of link traffic volumes produced by the previous method. It adds $c_t = (\mathbf{y}_S(t) - \alpha)^\theta$, where α is a threshold as above and θ is an attack parameter. This method adds more chaff to larger deviations from the threshold, again assuming that the adversary has knowledge of the stationary traffic mean in order to choose the threshold.

These four data poisoning schemes represent different styles of attacks (e.g., add traffic in every time slot, or only during some time intervals) and different dependencies on the underlying regular traffic (the *Half Normal* and *Bernoulli* are traffic independent whereas *Add-Constant-If-Big* and *Add-More-If-Bigger* are traffic dependent).

3.3 Boiling Frog Attacks

In the above attacks, the poisoning occurs during a single week. We next consider a long-term attack in which the adversary slowly, but increasingly, poisons the principal components over several weeks. This attack is covert in that small amounts of chaff, in gradually increasing quantities, are used for poisoning. 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.

Any of our four poisoning schemes presented above can be used here. The key idea is to start by setting the attack parameter, θ, to a small value, and then to increase it slowly over time. Before a *Boiling Frog* attack begins, the PCA-based anomaly detector is trained on non-adversarial data—a week of data untouched by the poisoning scheme. In week 1 of the attack, the target flow is injected with chaff generated with parameter θ_1. At the week's end, PCA is retrained on that week's data. We assume here that any anomalies detected by PCA during that week are excluded as training data. This process continues with θ_t used for week t. Even though PCA is retrained from scratch each week, the training data includes events not caught by the previous detector. Each successive week will contain more malicious training data. This process continues until the week of the DoS attack, when the adversary stops injecting chaff.

4 Experiments

4.1 Traffic Data

We use OD flow data collected from the Abilene network to simulate attacks on PCA-based anomaly detection. Abilene is the Internet2 backbone. Data was collected over an almost contiguous 6 month period from March 1, 2004 through September 10, 2004 [6]. Each week of data consists of 2016 measurements across all network OD flows binned into 5 minute intervals. At the time of collection the network consisted of 12 PoPs and 15 inter-PoP links. All 144 possible OD flows are represented, and 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.

4.2 Validation

We evaluate our data poisoning schemes on PCA-based anomaly detectors. We replicate the method of Lakhina et al. [2] for adding synthetic anomlies. We use week-long training sets, as such a time scale is sufficiently large to capture weekday and weekend diurnal trends [4]. We detect synthetic attacks on a 5 minute interval. Starting with the OD 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. After multiplying by the routing matrix $\mathbf{A}$, the link volume measurement at time t is anomalous. Negative examples (benign OD flows) are generated similarly by setting the flow volume to a small value known to correspond to normal levels of flow traffic. These large and small values are defined, as in [2], to be 1.5 and 0.625 times a cutoff of 8×10^7. When validating PCA's performance on an attack, with a positive example of anomalous flow f, we first poison the training data along the same flow f.

Our performance metric for measuring the success of the poisoning strategies is through their impact on the PCA-based detector's *false negative rates* (FNRs). 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 simulate a DoS attack along every flow at every time. We average FNRs over all 144 possible anomalous flows and anomaly times. When reporting the effect of an attack on traffic volumes, we first average over links within each flow then over flows and times. Furthermore we generally report average volumes relative to the pre-attack average volumes. We report validations of the *Bernoulli* and *Half Normal* randomized attacks averaged over 50 independent repetitions.

4.3 Week-Long and Boiling Frog Attacks

We evaluated the effectiveness of our attacker strategies. We chose weeks 20 and 21 from the Abilene dataset to simulate the *Week-Long* attacks. The PCA algorithm was trained on the week 20 TM poisoned by the attacker; we then injected attacks during week 21 to see for how often the attacker can evade detection. We selected these particular weeks because PCA achieved the lowest FNRs on these during the testing.

To test the *Boiling Frog* attack we simulated TM data, inspired by methods used in [2]. Our simulations presented multiple weeks of stationary data to the adversary. While such data is unrealistic in practice, it presents 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 drift in benign data, and adversarial poisoning. Demonstrated flaws of PCA in the stationary case constitute strong results about PCA for anomaly detection in practice. We decided to validate the *Boiling Frog* attack on a synthesized multi-week dataset, because the 6 month Abilene dataset of [6] proved to be too non-stationary for PCA to operate well from one week to the next. It is unclear whether the non-stationarity observed in this dataset is prevalent in general.

We synthesized a multi-week set of OD flow traffic matrices, with stationarity on the inter-week level, based on week 20 of the Abilene dataset. We used a three step generative procedure to model each OD flow separately. First the underlying diurnal trends of the OD flow f time series is modeling 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 diurnal 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 [2]. 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, intended to capture short-term benign traffic variance. The second process models volume anomalies as being exponentially distributed.

Step 2 selects which of the two noise processes is used at each time interval. We look at the total residuals (after filtering out the Fourier trends) and note the smallest (negative) residual value. Let $-m$ denote this value. 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 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 classified residual traffic using the Maximum Likelihood estimates of the Gaussian's variance and exponential and geometric rates respectively.

We include goodness-of-fit results for four OD flows: flow 144 which maximizes mean and variance among all 144 flows; flow 113 which has one of the smallest means and variances among all 144 flows; and flows 15 and 75 which have median mean and variance, respectively, among all 144 flows. After manual inspection on all flows we believe these flows to be representative elephant, mouse and two mid-level flows, respectively. Figs. 2–17 include evaluations of the fit of the Gaussian, Exponential and Geometric distributions to the three processes via quantile-quantile plots. In general the Gaussian and Exponential Q-Q plots for the traffic volume processes are close to linear illustrating good fits. The Q-Q plots for the Geometric inter-anomaly arrival times, in Figs. 14–17 show more variable results. For each of the four flows, we also plot the time series for a week of both the Abilene data and our simulated model. We believe that this model is reasonable and sufficient for our purposes of evaluating the *Boiling Frog* attack.

In our simulations, all link volumes are constrained to respect the link capacities in the Abilene network—10gbps for all but one link that operates at one fourth of this rate. Chaff that would cause traffic to exceed the link capacities are capped.

5 Results

We evaluate the effectiveness of our four data poisoning schemes. During the testing week, the attacker launches a DoS attack in each 5 minute time window. Although our poisoning schemes focus on adding variance, the mean of the OD flow being poisoned shifts as well, shifting the means of all links over which the OD flow traverses. The x-axis in Fig. 18 indicates the relative increase in the mean rate of the affected links for the *Week-Long* attacks. We average over all experiments (i.e.,

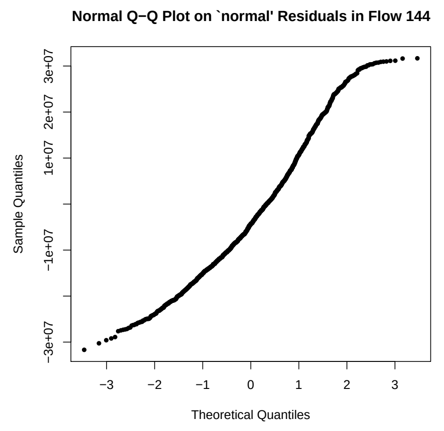

Figure 2: The normal Q-Q plot of flow 144's traffic with normal residuals.

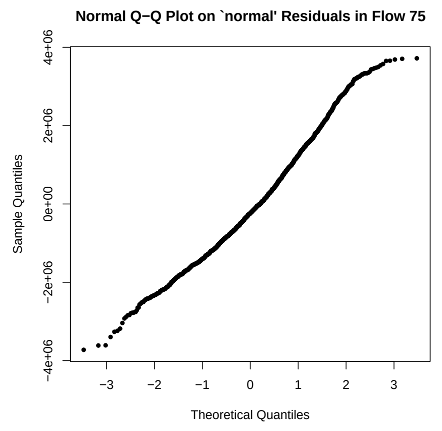

Figure 3: The normal Q-Q plot of flow 75's traffic with normal residuals.

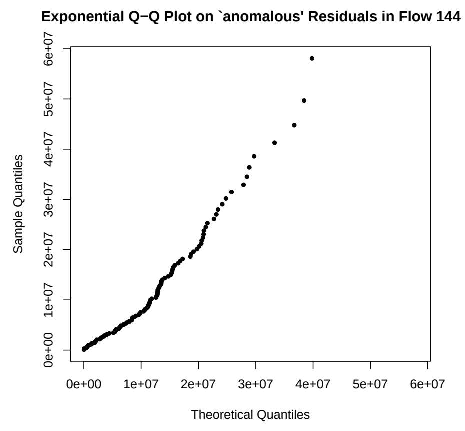

Figure 4: The exponential Q-Q plot of flow 144's traffic with anomalous residuals.

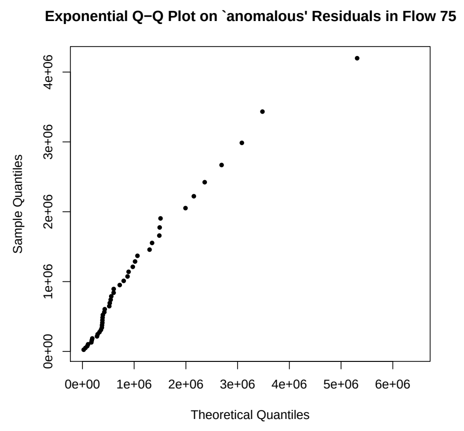

Figure 5: The exponential Q-Q plot of flow 75's traffic with anomalous residuals.

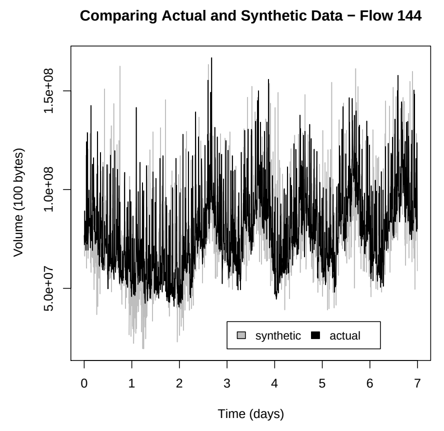

Figure 6: Simulated flow 144 time series (gray) vs. actual (black).

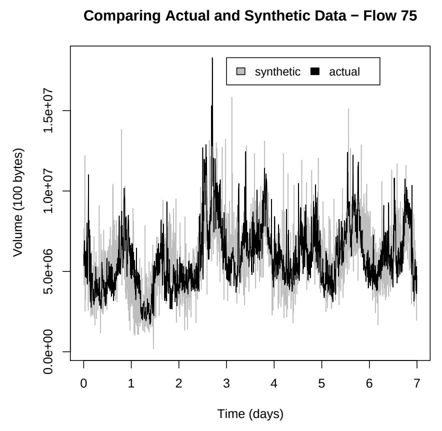

Figure 7: Simulated flow 75 time series (gray) vs. actual (black).

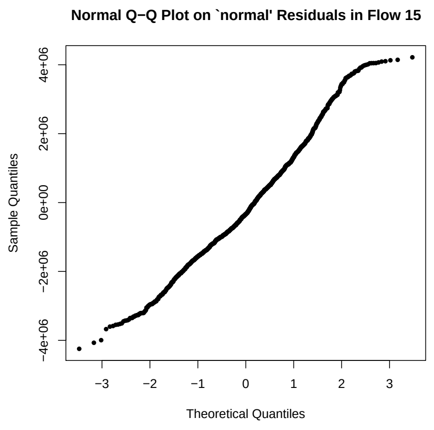

Figure 8: The normal Q-Q plot of flow 15's traffic with normal residuals.

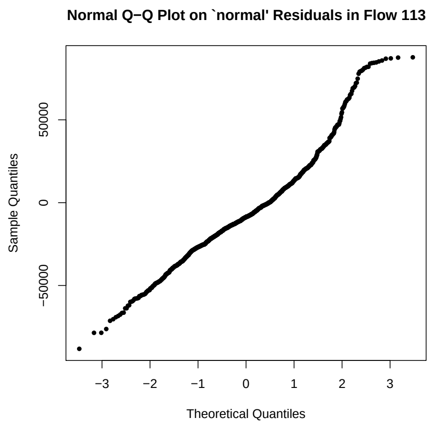

Figure 9: The normal Q-Q plot of flow 113's traffic with normal residuals.

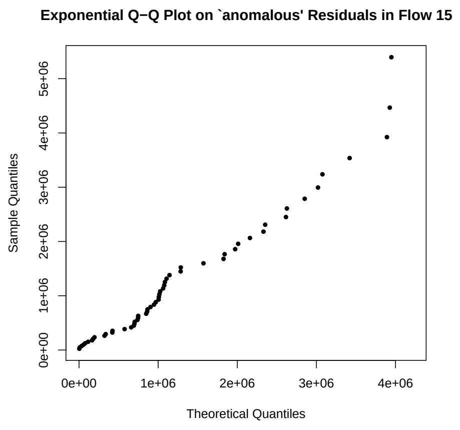

Figure 10: The exponential Q-Q plot of flow 15's traffic with anomalous residuals.

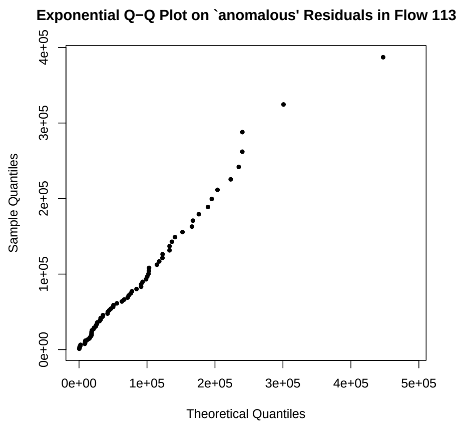

Figure 11: The exponential Q-Q plot of flow 113's traffic with anomalous residuals.

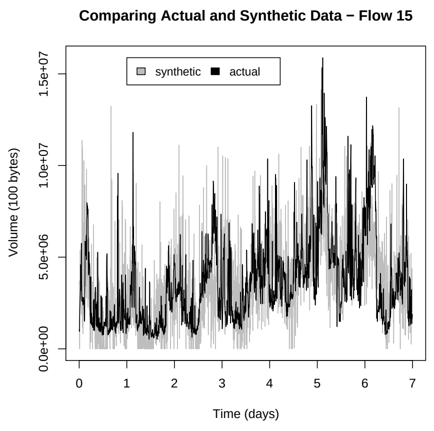

Figure 12: Simulated flow 15 time series (gray) vs. actual (black).

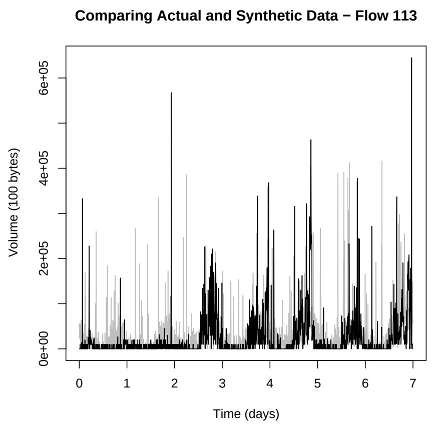

Figure 13: Simulated flow 113 time series (gray) vs. actual (black).

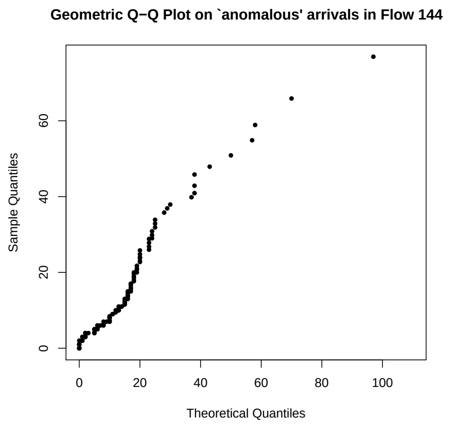

Figure 14: The geometric Q-Q plot of flow 144's inter-arrival times.

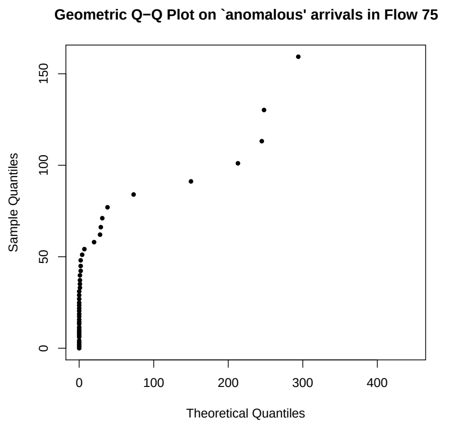

Figure 15: The geometric Q-Q plot of flow 75's inter-arrival times.

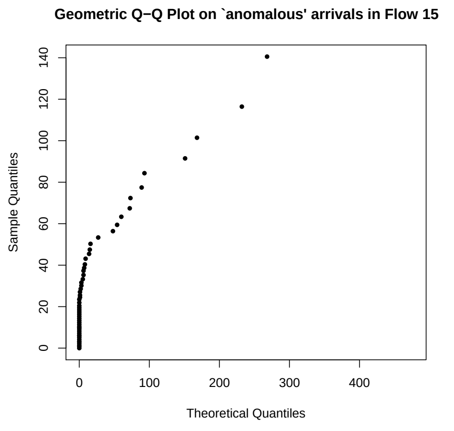

Figure 16: The geometric Q-Q plot of flow 15's inter-arrival times.

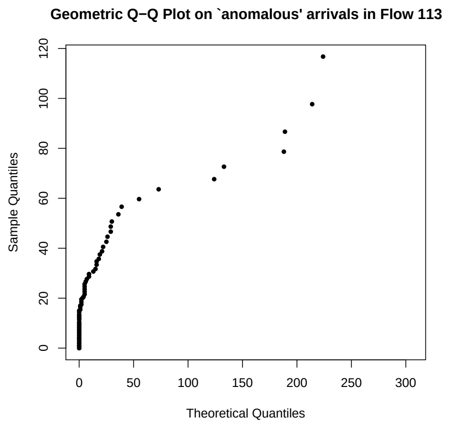

Figure 17: The geometric Q-Q plot of flow 113's inter-arrival times.

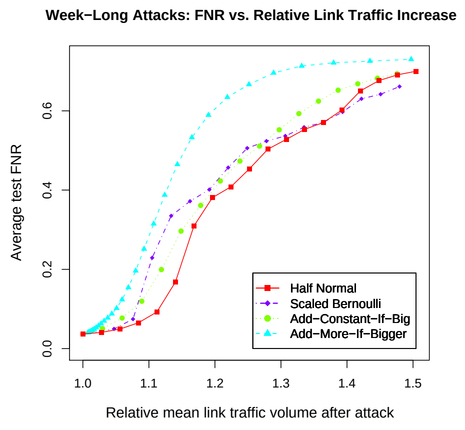

Figure 18: *Week-Long* attacks using the chaff generation methods. Test FNRs are plot against the relative increase to the mean link volumes for the attacked flow.

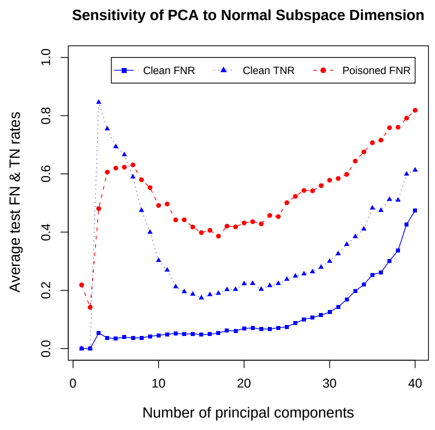

Figure 19: Effect of normal subspace dimension on FNRs under an *Add-More-If-Bigger* attack increasing source link traffic by 20%.

over all OD flows). The figure shows that of our four schemes, the *Add-More-If-Bigger* scheme is most effective in that it raises the FNRs the most. For a 10% average increase in the mean link rates, the attacker can raise the FNRs to roughly 23%. The baseline FNR of PCA without data poisoning is approximately 4%, so the attacker success rate is nearly 6 times larger in this example. (Note that 23% may not be viewed as a high likelihood of successful evasion.) This number represents an average over attacks launched each 5 minute window, so the attacker could simply retry multiple times.

Simply increasing the number of principal components used by the PCA algorithm does not protect against these data poisoning schemes. In Fig. 19 we show the FNRs for the "clean" (unpoisoned) PCA and the FNR for the detector poisoned by the *Add-More-If-Bigger* method. We also plot the True Negative Rate (TNR) showing the number of correctly identified benign events. We see that the poisoned FNR is not monotonic in the number of principal components, so there is no easy choice (as there was for unpoisoned PCA). Moreover the supposed minimum poisoned FNR (beyond the first few points) that occurs around 17 PCs could most likely not be used as a universal choice. It is likely that different attacks will require different numbers of PCs to find their optimal FNR. Furthermore, choosing a number of PCs such as 17 comes at a tremendous cost to the degradation of the TNRs. The unpoisoned detector did not suffer this tradeoff (i.e., with 4 PCs, it achieved a good balance between small FNR and high TNR). This illustrates that poisoning attacks force the detector to make untenable tradeoffs between FNRs and TNRs.

We now evaluate the effectiveness of the *Boiling Frog* strategy. In Fig. 20 we plot the FNRs against the poisoning duration for the PCA detector. We examine four different poisoning *schedules* with growth rates $g \in \{1.01, 1.02, 1.05, 1.15\}$. The goal of the schedule is to increase the average attacked links' traffic by a factor of g from week to week. Parameter θ is chosen to achieve this goal. We see that the FNR dramatically increases for all four schedules as the poison duration increases.

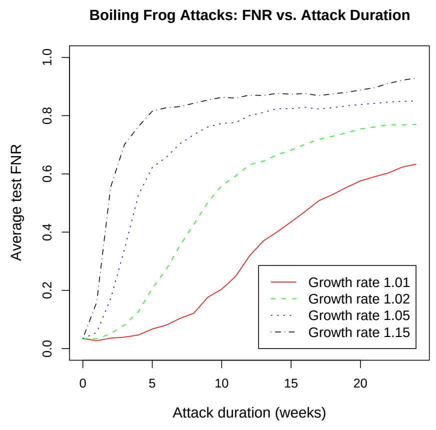

Figure 20: *Boiling Frog* attacks using the *Add-More-If-Bigger* chaff method, for four geometrically increasing schedules.

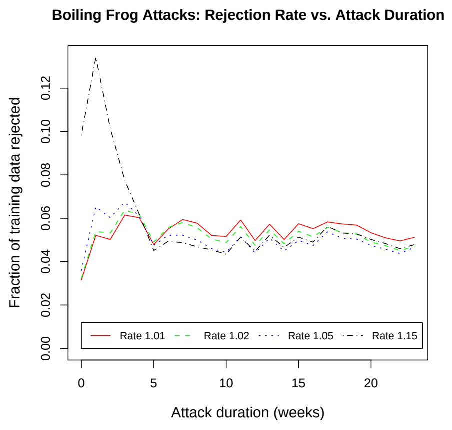

Figure 21: The training data rejection rates for the *Boiling Frog* attacks shown in Fig. 20.

With a 15% growth rate, the FNR is increased to more than 70% from 3.6% over 3 weeks of poisoning; even with $g = 1.05$ the FNR is increased to 50% over 3 weeks. Thus *Boiling Frog* attacks are effective on even small volume growth rates.

Fig. 21 shows the proportion of training data rejected each week by PCA (*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. 18 and 20, observe that to get the FNR to 50%, an increase in mean traffic of roughly 18% for the *Week-Long* 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.

6 Conclusions

In this paper we illustrated that network-wide PCA-based anomaly detectors can be compromised by simple data poisoning strategies. Reasonably small amounts of additional traffic can be injected to increase the attacker's likelihood of successful DoS attacks by a factor of 6 (depending upon the attack parameters and style). Moreover, with stealthy poisoning strategies executed over longer time periods, an attacker can increase the FN rates to over 50% with less data than poisoning schemes carried out during a short time window. Due to demonstrated sensitivities of true positive rates and attack efficacy to the number of principal components, we showed that our poisoning attacks force the detector to make untenable tradeoffs between FNRs and TNRs. In the future, we plan to study defensive techniques to combat these attacks such as Robust PCA algorithms [3] that down-weigh outliers, thereby reducing their impact.

Acknowledgments

We would like to thank Marco Barreno for his valuable comments and suggestions on this research.

This work was supported in part by the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the National Science Foundation (NSF award #CCF-0424422), the Air Force Office of Scientific Research (AFOSR #FA9550-06-1-0244), Cisco, British Telecom, ESCHER, Hewlett-Packard, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies; in part by California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID#06-148 and #07-012) and Siemens; and in part by the cyber-DEfense Technology Experimental Research laboratory (DETERlab), which receives support from the Department of Homeland Security Advanced Research Projects Agency (HSARPA award #022412) and AFOSR (#FA9550-07-1-0501). 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.

References

[1] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. Technical Report UCB/EECS-2008-43, EECS Department, University of California, Berkeley, April 2008.

[2] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. SIGCOMM '04*, pages 219–230, 2004.

[3] Ricardo Maronna. Principal components and orthogonal regression based on robust scales. *Technometrics*, 47(3):264–273, 2005.

[4] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. Sensitivity of PCA for traffic anomaly detection. *Proc. SIGMETRICS '07*, 35(1):109–120, 2007.

[5] A. Soule, K. Salamatian, and N. Taft. Combining filtering and statistical methods for anomaly detection. In *IMC'05*, 2005.

[6] Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proc. IMC '05*, pages 1–14, NY, NY, USA, 2005.