CS 499/599: Machine Learning Security

02.07: Data Poisoning

Mon/Wed 12:00 – 1:50 pm

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab

Notice

- Due dates
  - Written Paper Critiques (on the 9^{th})
- Sign-up (on Canvas)
  - Scribe Lecture Note
  - In-class Paper Presentation / Discussion

Topics for Today

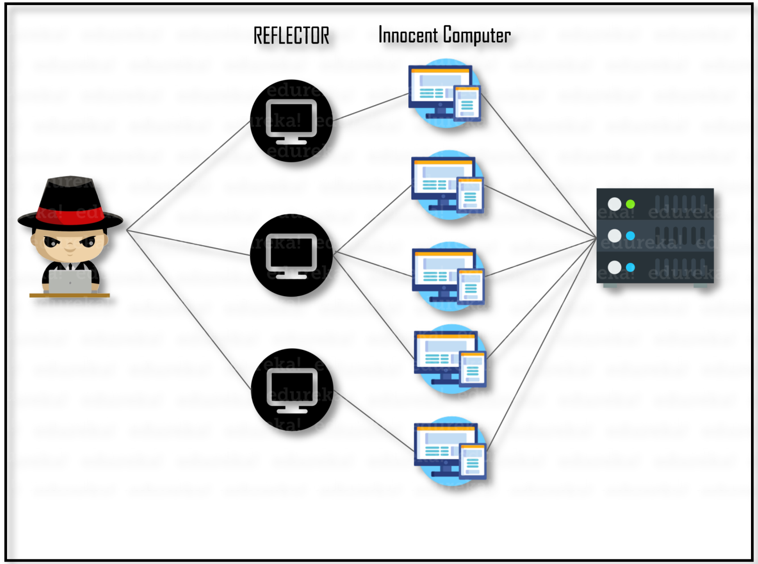
- Data Poisoning
  - Exploitations
    - Spam filtering
    - DDoS detection
  - Conclusion (and implications)
- Data Poisoning:
  - Indiscriminate Attacks
    - Support vector machines (SVMs)
    - Regression models
  - Conclusion (and implications)

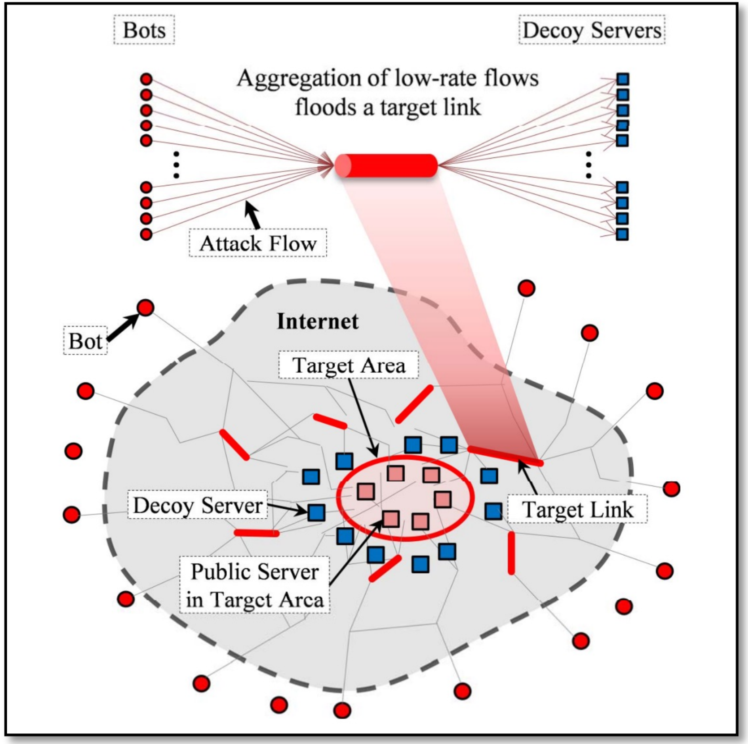
Nelson *et al.*, Exploiting Machine Learning to Subvert Your Spam Filter

Rubinstein *et al.*, ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors

Motivation

- Goals
  - DDoS attack [Link]

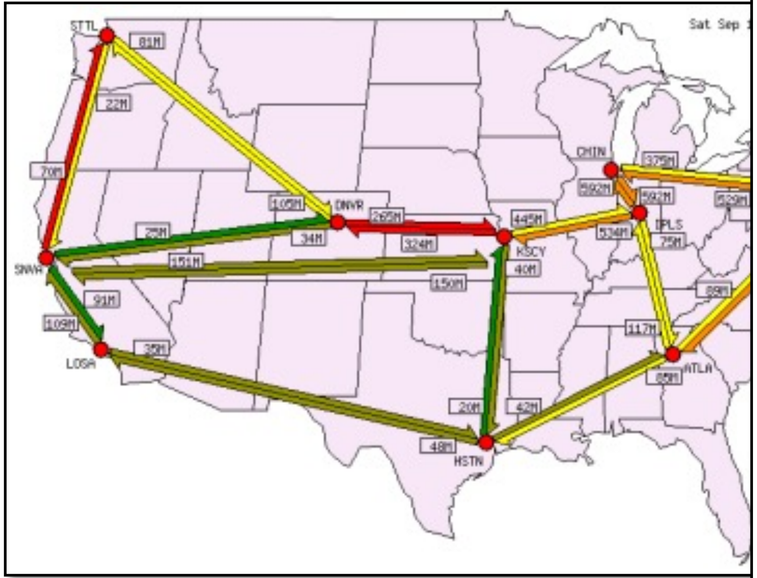


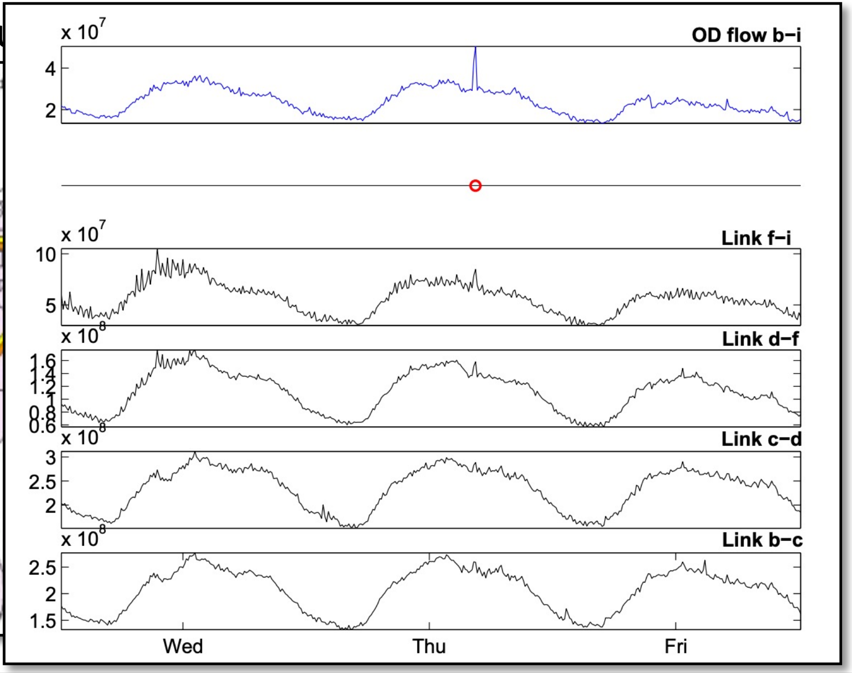


https://edureka.co/blog/what-is-ddos-attack/
Kang *et al.*, Crossfire Attack, IEEE Security and Privacy 2013

Motivation

- Goals
  - DDoS attack
  - Attacker’s network traffic successfully cross an ISP’s network
  - ISP Monitors in-out traffic and alert “volu

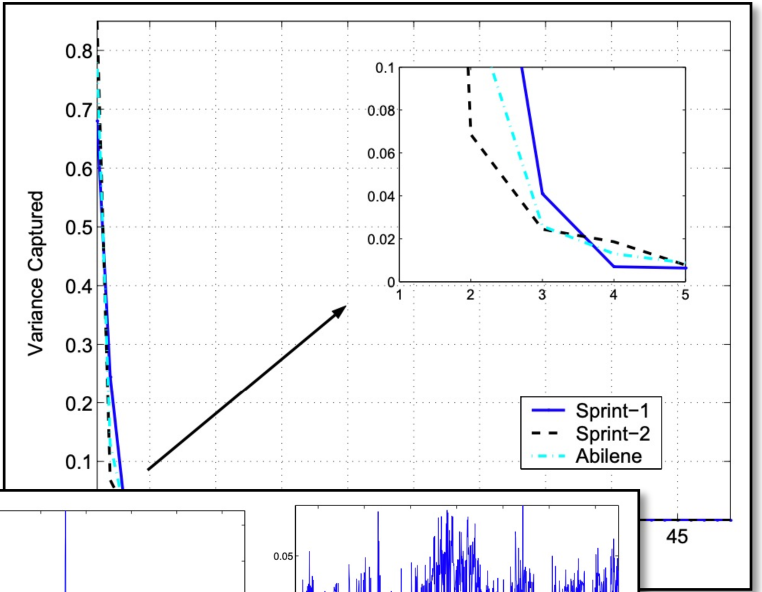


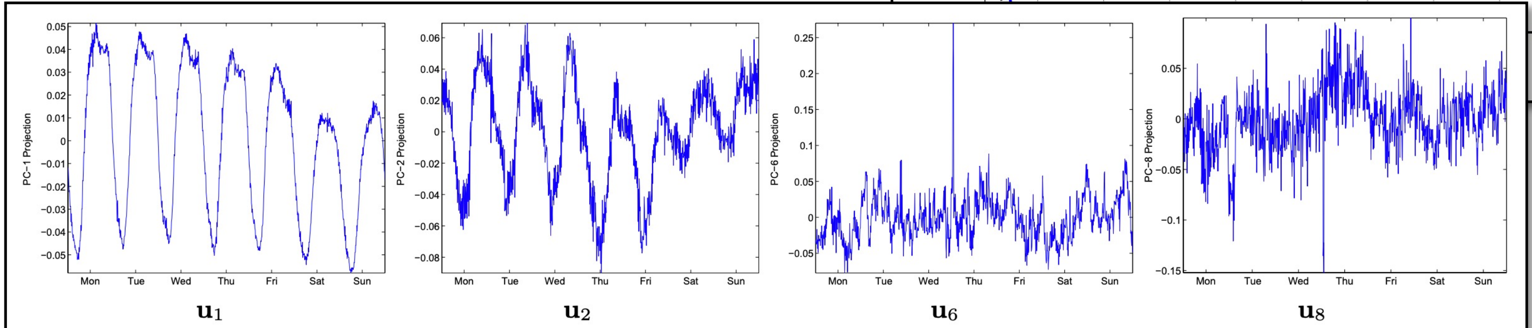


Background: PCA-based Anomaly Detector (Lakhina et al.)

- PCA (Principal Component Analysis)
  - Represent data with smaller set of variables

- PCA-based Anomaly Detection
  - Y: T x N (time series of all links)
  - Run PCA on Y
    - Find the top-K normal components
    - The rest [N-K] is for detecting anomalies

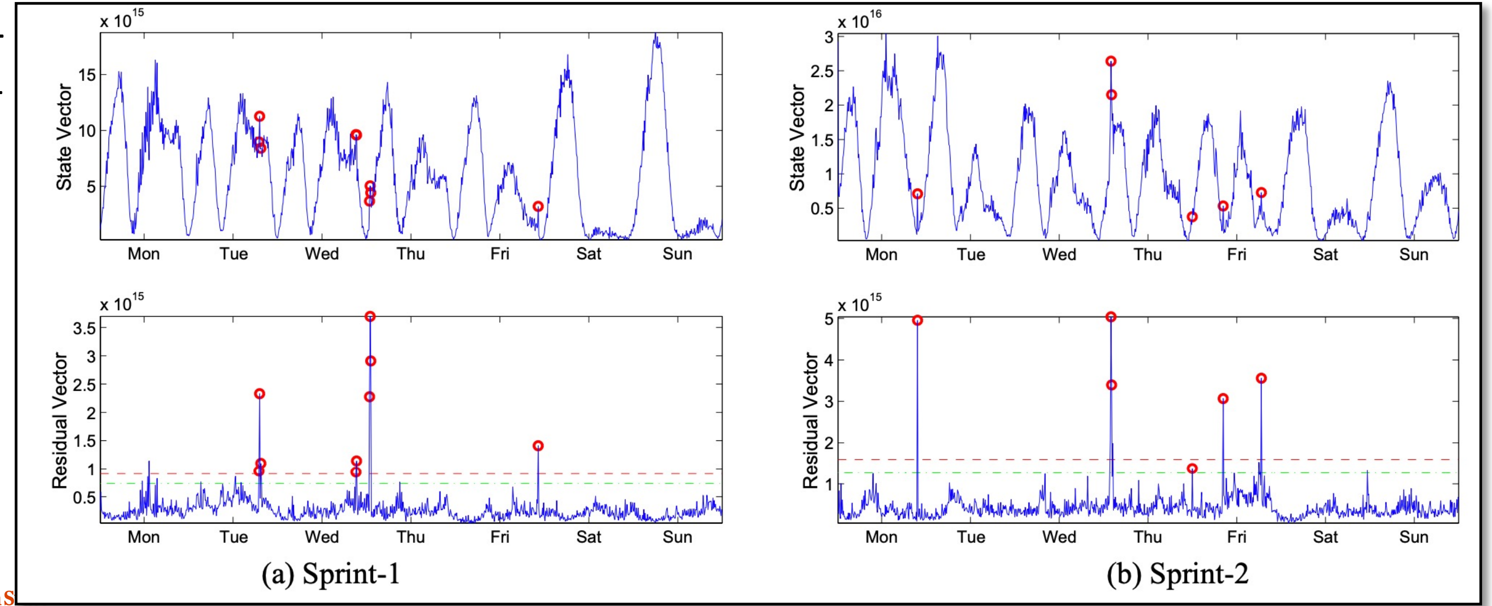




(a) Normal Behavior (b) Anomalous Behavior

Background: PCA-based Anomaly Detector (Lakhina et al.)

- PCA (Principal Component Analysis)
  - Represent data with smaller set of variables
- PCA-based Anomaly Detection
  -
  -



(a) Sprint-1 (b) Sprint-2

Motivation

- Research Questions:
  - **RQ 1:** How can we **poison** the anomaly detector to launch DDoS?
  - **RQ 2:** How much this attack will be **effective**?
  - **RQ 3:** How can we **mitigate** this poisoning attacks?

Threat Model

- Goal
  - Manipulate the anomaly detector while increasing the traffic volume [~indiscriminate]

- Capability
  - Inject additional traffic (*chaff*) along the network flow

- Knowledge
  - Does not know the traffic (*uninformed* attack)
  - Know the current volume of traffic (*locally-informed* attack)
  - Know all the details about the network links (*globally-informed* attack)

- [Victim] Anomaly Detector
  - PCA retrained each week on $m - 1$ (with anomalies removed)
  - Use the trained PCA for detecting anomalies in week m

Poisoning Attack Strategies

- Uninformed
  - Randomly add chaff (the amount is θ)

- Locally-informed
  - Only add chaff $(\max\{0, y_S(t) - \alpha\})^{\theta}$ when the traffic is already reasonably large

- Globally-informed
  - Optimize the amount of chaff

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

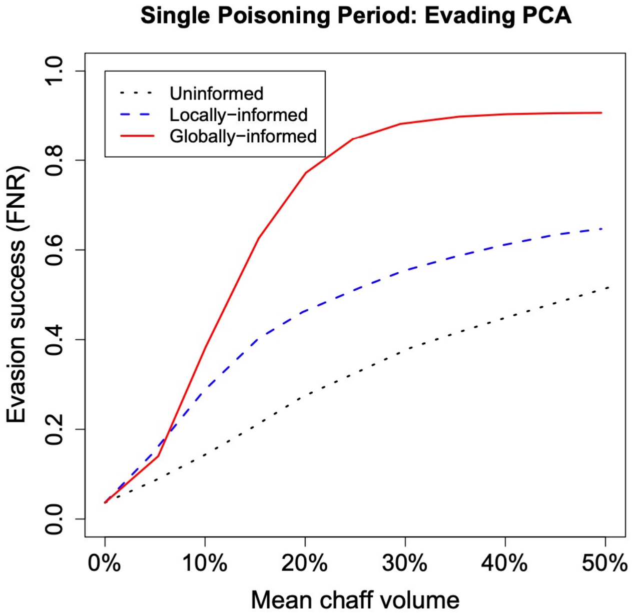
- **[Continuous case]** Boiling Frog attack
  - Initially set the theta to a small value, and increase it over time
  - Use any of the three (informed, locally-informed, or globally-informed) to add chaff

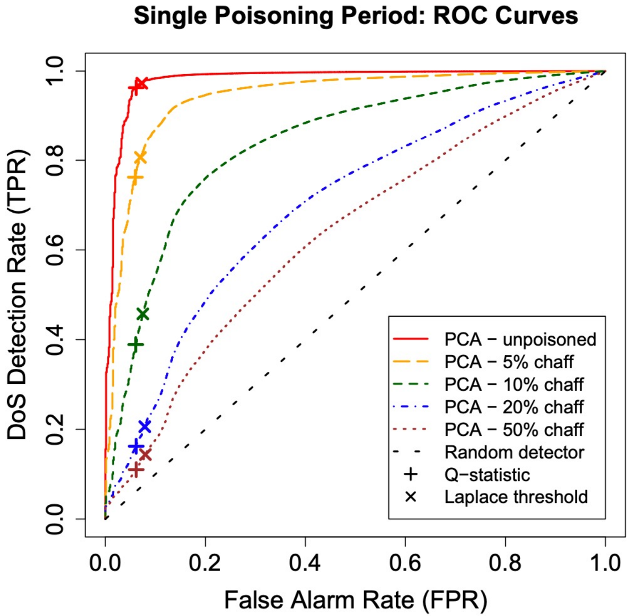
Evaluation: Attacks

- Setup
  - Dataset: OD Flow Data from Ailene network
    - Period: Mar. 2004 – Sep. 2004 (6 months)
    - Each week: 2016 measurements x 144 networks, 5 min intervals
- Metrics
  - Detector’s false negative rate (FNR)
  - Use ROC curve to show tradeoffs btw true positive rate (TPR) and FPR

Evaluation: Attacks

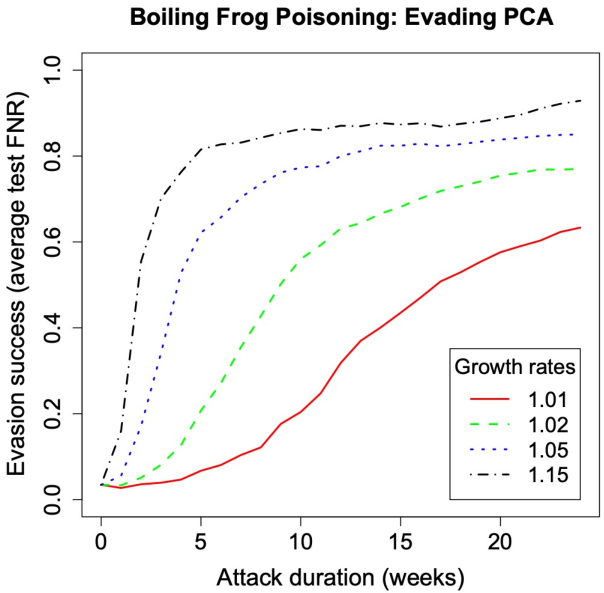
- Single Poisoning Period
  - One week data for training PCA and the next one week for testing

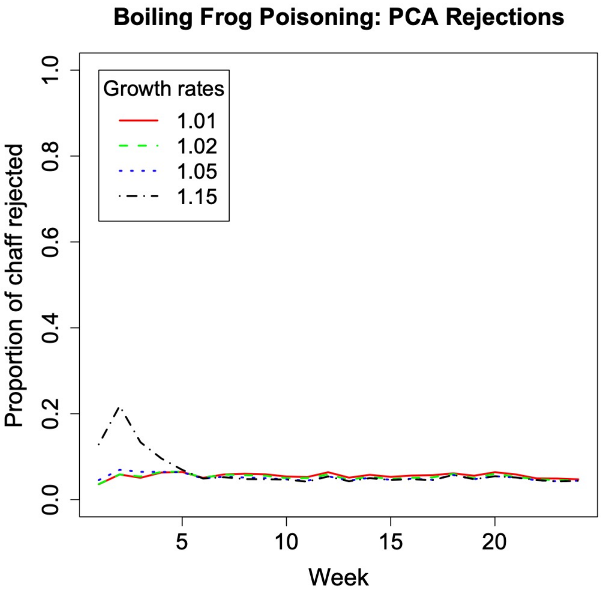




Evaluation: Attacks

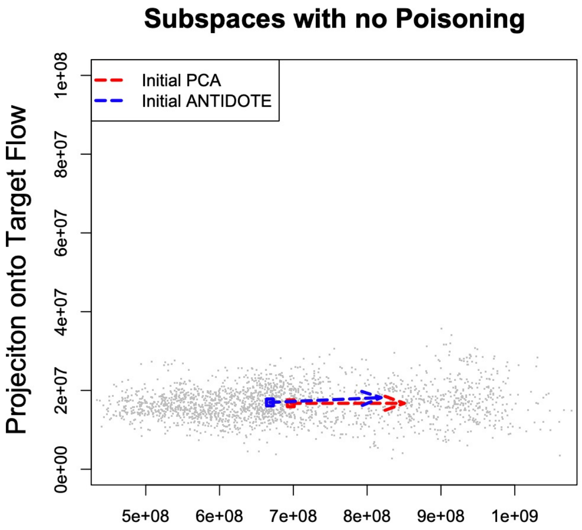
- Boiling Frogs
  - Data from previous weeks for training PCA and the current week for testing

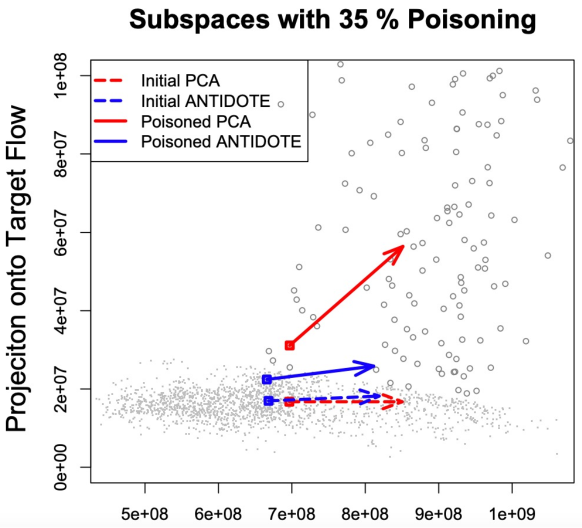




Defense: ANTIDOTE

- Robust statistics
  - Reduce the sensitivity of statistics to outliers
  - Use PCA-GRID (Croux *et al.*)

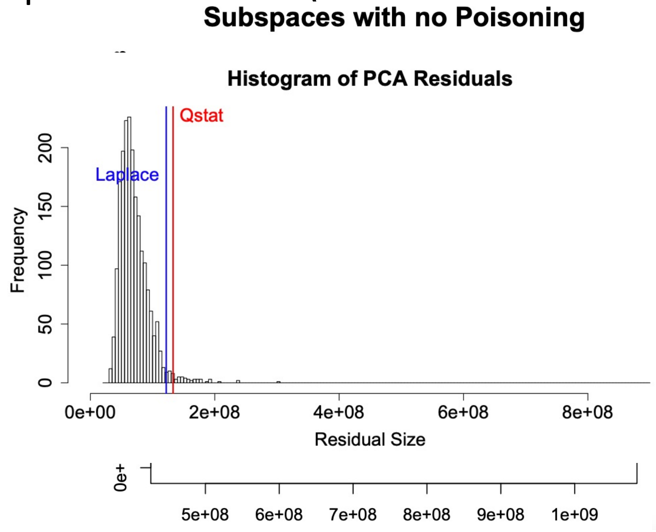


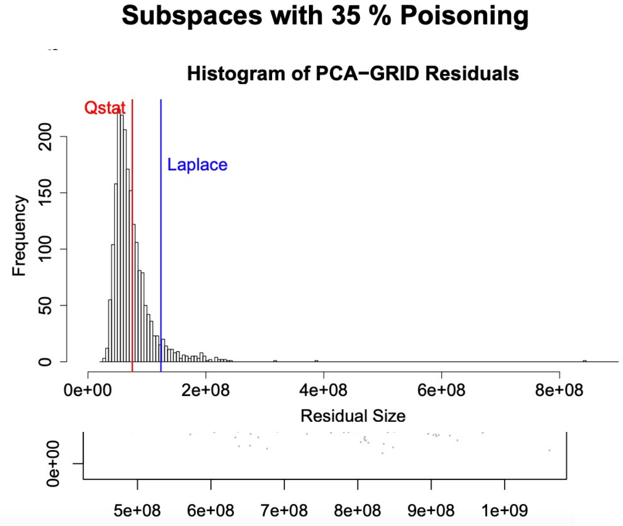


Projection on 1st Principal Component

Defense: ANTIDOTE

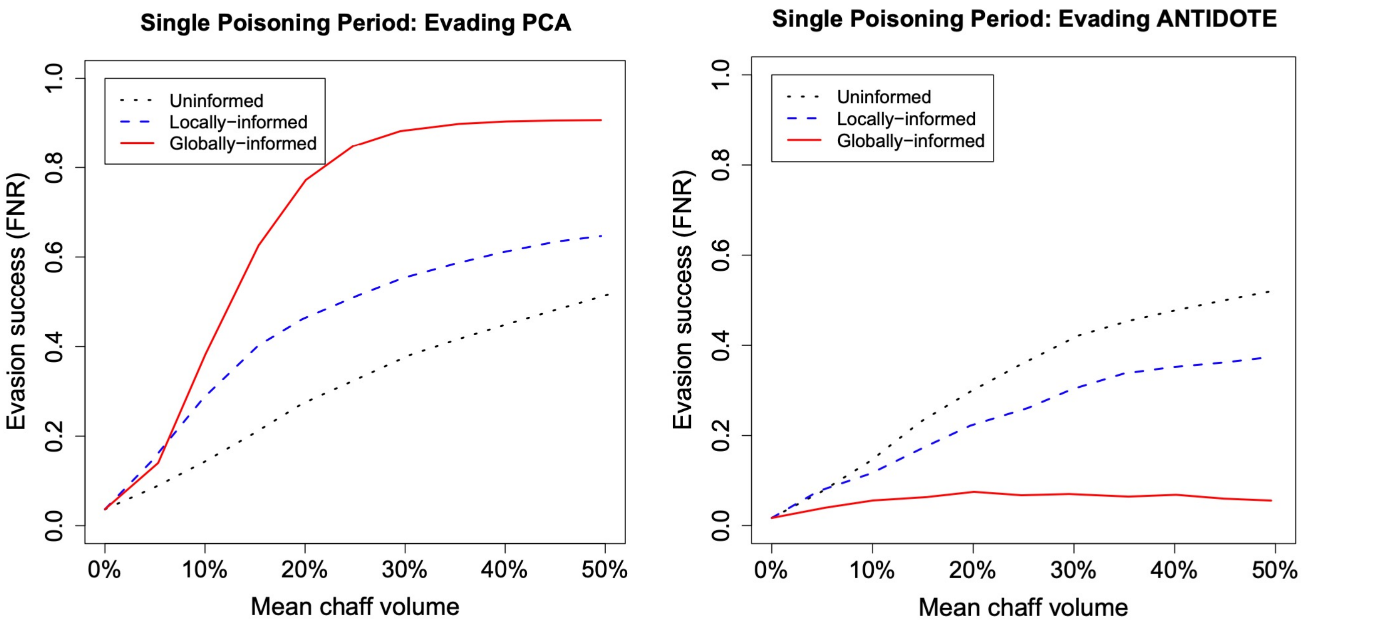
- Robust statistics
  - Reduce the sensitivity of statistics to outliers
  - Use PCA-GRID (Croux *et al.*)
  - Use Laplace Threshold (Robust estimate for its residual threshold)





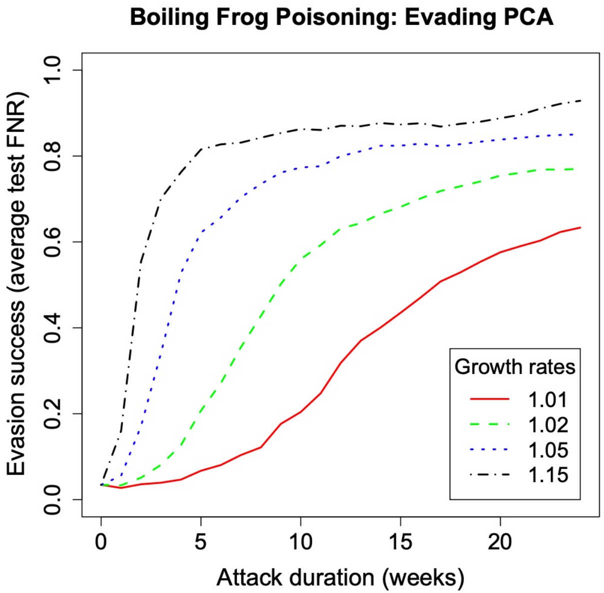
Evaluation: ANTIDOTE

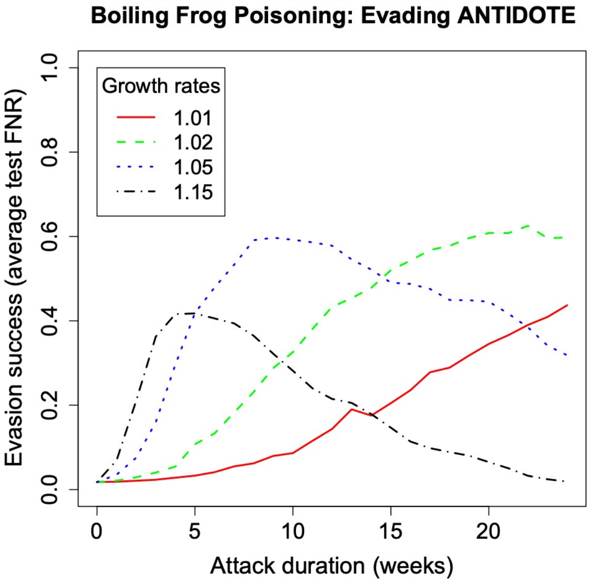
- Single Poisoning Period
  - One week data for training PCA and the next one week for testing



Evaluation: Attacks

- Boiling Frogs
  - Data from previous weeks for training PCA and the current week for testing





Conclusion

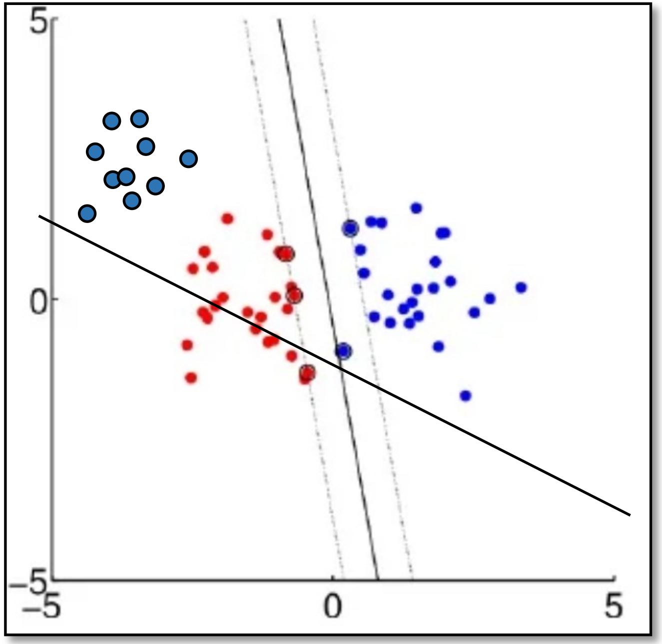
- Research Questions:
  - **RQ 1:** How can we **poison** the anomaly detector to launch DDoS?
    - Inject some additional traffic (chaff)
    - Make a detector have false estimation of normal states
    - Three-levels of knowledge: uninformed / locally-informed / globally-informed
    - Single poisoning vs. Boiling frogs
  - **RQ 2:** How much this attack will be **effective**?
    - The success increases as we increase (knowledge / % of poisons / period)
  - **RQ 3:** How can we **mitigate** this poisoning attacks?
    - ANTIDOTE: Robust statistics (PCA-GRID + Laplace threshold)

Topics for Today

- Data Poisoning
  - Exploitations
    - Spam filtering
    - DDoS detection
  - Conclusion (and implications)
- Data Poisoning
  - Indiscriminate Attacks
    - Support vector machines (SVMs)
    - Regression models
  - Conclusion (and implications)

Biggio *et al.*, Poisoning Attacks against Support Vector Machines
Jagielski *et al.*, Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning

Revisited: Linear Models vs. DNNs



← Linear model (SVM)

Background: Support Vector Machine

- DIT [Link]
  - 1: let's put green points
  - 2: let's put red points on the other side
  - 3: let's put red points closer to the green cluster
  - 4: let's put red points in the middle of the green cluster
  - 5: let's use another kernel.

Threat Model

- Goal
  - Indiscriminate attack
  - Find a point (x_c, y_c), whose addition to D_{tr} decreases a model's acc.

- Capability
  - Train a model f on D_{tr}
  - Inject the point (x_c, y_c) into D_{tr}

- Knowledge
  - D_{tr} : training data
  - D_{val}: validation data (where we pick the poison)
  - f: a (linear) SVM and its parameters a_i, b
  - A: training algorithm (*e.g.*, Sub-Gradient Descent)

Proposed Attack on SVM!

Algorithm 1 Poisoning attack against SVM

Input: $\mathcal{D}_{\text{tr}}$, the training data; $\mathcal{D}_{\text{val}}$, the validation data; y_c, the class label of the attack point; $x_c^{(0)}$, the initial attack point; t, the step size.

Output: x_c, the final attack point.

$\{\alpha_i, b\} \leftarrow$ learn an SVM on $\mathcal{D}_{\text{tr}}$. // train an SVM on the clean data

$k \leftarrow 0$.

repeat

  Re-compute the SVM solution on $\mathcal{D}_{\text{tr}} \cup \{x_c^{(p)}, y_c\}$ using incremental SVM (*e.g.*, Cauwenberghs & Poggio, 2001). This step requires $\{\alpha_i, b\}$. // train an SVM with the poison

  Compute $\frac{\partial L}{\partial u}$ on $\mathcal{D}_{\text{val}}$ according to Eq. (10). // compute the gradient

  Set u to a unit vector aligned with $\frac{\partial L}{\partial u}$.

  $k \leftarrow k + 1$ and $x_c^{(p)} \leftarrow x_c^{(p-1)} + tu$ // update the poison, to increase the loss

until $L\left(x_c^{(p)}\right) - L\left(x_c^{(p-1)}\right) < \epsilon$ // stop if the loss doesn't increase more than ϵ

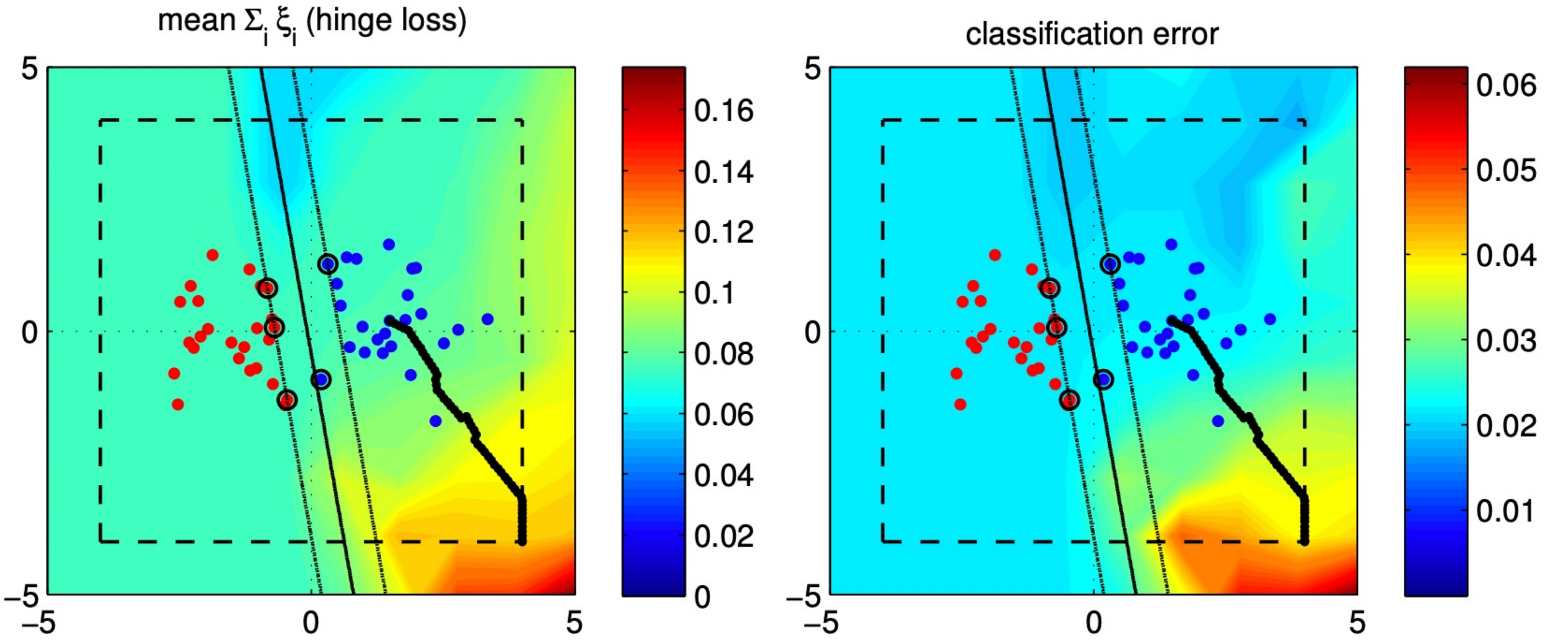
return: $x_c = x_c^{(p)}$

Evaluation

- Setup
  - Datasets
    - Artificial data: Gaussian dist. [$N(-1.5, 0.6^2)$ vs. $N(1.5,\ 0.6^2)$]
    - Real data: MNIST [1 vs. 7 | 8 vs. 9 | 0 vs. 4]
  - Model(s)
    - SVM [Linear vs. RBF-Kernel]

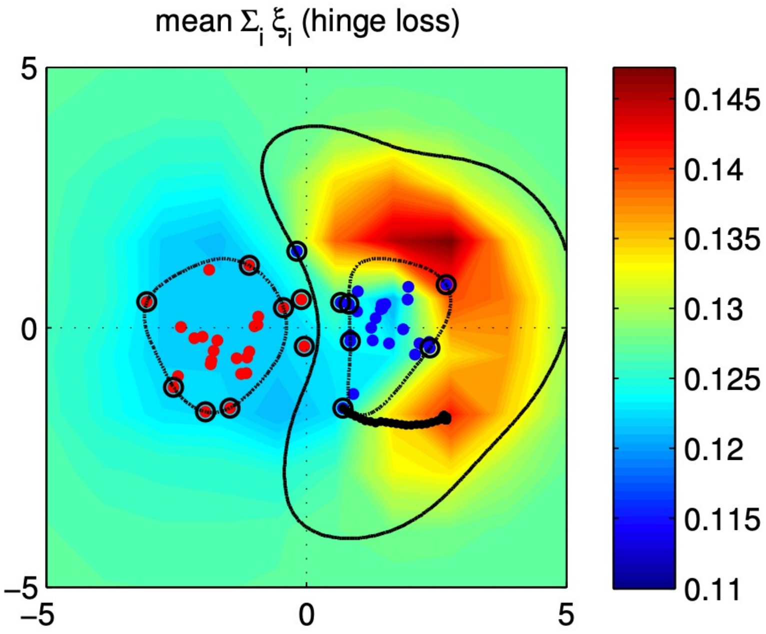
Evaluation: Artificial Data

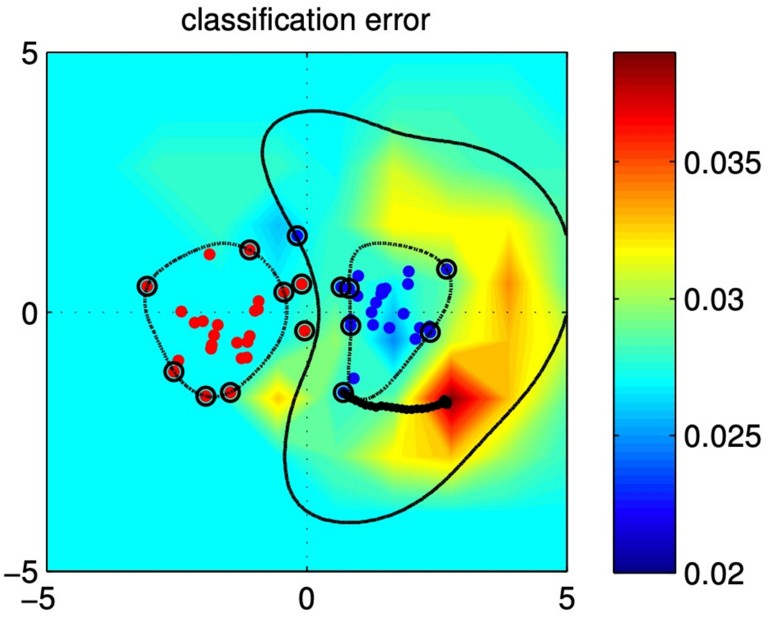
- Linear SVM



Evaluation: Artificial Data

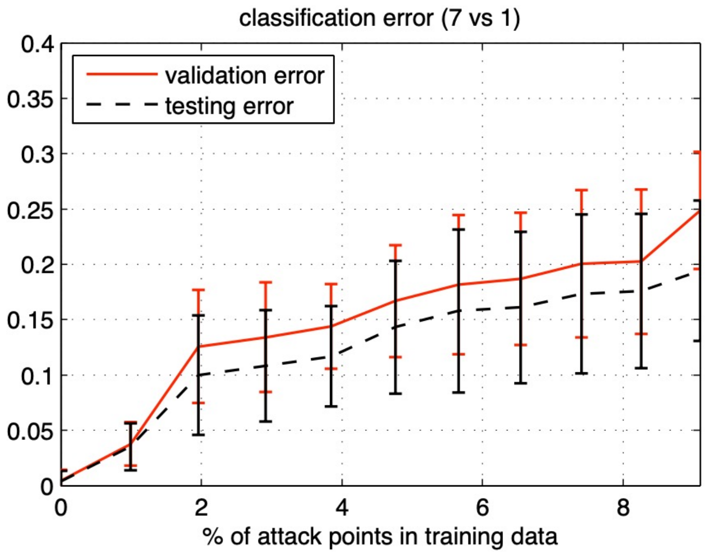
- SVM w. RBF Kernel





Evaluation: MNIST

- Linear SVM



- Results
  - Use a *single* poison
  - Error increases by 15 – 20%
  - Increasing # poisons leads to a higher error

Topics for Today

- Data Poisoning
  - Exploitations
    - Spam filtering
    - DDoS detection
  - Conclusion (and implications)

- Data Poisoning
  - Indiscriminate Attacks
    - Support vector machines (SVMs)
    - Regression models
  - Conclusion (and implications)

Biggio *et al.*, Poisoning Attacks against Support Vector Machines

Jagielski *et al.*, Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning

Background: Regression Models

- Regression Models [Demo]
  - DIT
    - 1. let’s add some more points
    - 2. let’s see how much error (*RMSE*) it increases
  - In the Paper
    - Ordinary Least Squares (OLS)
    - Ridge regression
    - LASSO
    - Elastic-net regression

Threat Model

- Goal
  - Indiscriminate attack (increase the error on D_{val})

- Capability
  - Train a model f on D_{tr}
  - Inject p poisons into the training set ($\mathrm{N}(D_{tr}) = n + p$)

- Knowledge [White-box vs. Black-box]
  - D_{tr} : training data (black-box adversary only has partial knowledge of D_{tr})
  - D_{val}: validation data
  - f: a model and its parameters (black-box attacker doesn't know the parameters)
  - L: training algorithm

Attack Formulation: Bi-level Optimization

$$\begin{aligned} \arg\max_{\mathcal{D}_p} \quad & \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}_p^\star)\,, \\ \text{s.t.} \quad & \boldsymbol{\theta}_p^\star \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_p, \boldsymbol{\theta}) \end{aligned}$$

- Outer-optimization: maximize the error of a model on the validation data
- Inner-optimization: minimize the model’s error on the training data

Proposed Attack on Regression Models!

Algorithm 1 Poisoning Attack Algorithm

Input: $\mathcal{D} = \mathcal{D}_{\text{tr}}$ (white-box) or $\mathcal{D}'_{\text{tr}}$ (black-box), $\mathcal{D}'$, $\mathcal{L}$, $\mathcal{W}$, the initial poisoning attack samples $\mathcal{D}_p^{(0)} = (\boldsymbol{x}_c, y_c)_{c=1}^{p}$, a small positive constant ε.

$i \leftarrow 0$ (iteration counter)

$\boldsymbol{\theta}^{(i)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{(i)}, \boldsymbol{\theta})$ // train a model on the contaminated data

repeat

  $w^{(i)} \leftarrow \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}^{(i)})$

  $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)}$

  for c = $1, \ldots, p$ **do**

    $\boldsymbol{x}_c^{(i+1)} \leftarrow \text{line\_search}\left(\boldsymbol{x}_c{}^{(i)}, \nabla_{\boldsymbol{x}_c} \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}^{(i+1)})\right)$ // update poisons to increase the loss of the model

    $\boldsymbol{\theta}^{(i+1)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{(i+1)}, \boldsymbol{\theta})$

    $w^{(i+1)} \leftarrow \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}^{(i+1)})$

  $i \leftarrow i + 1$

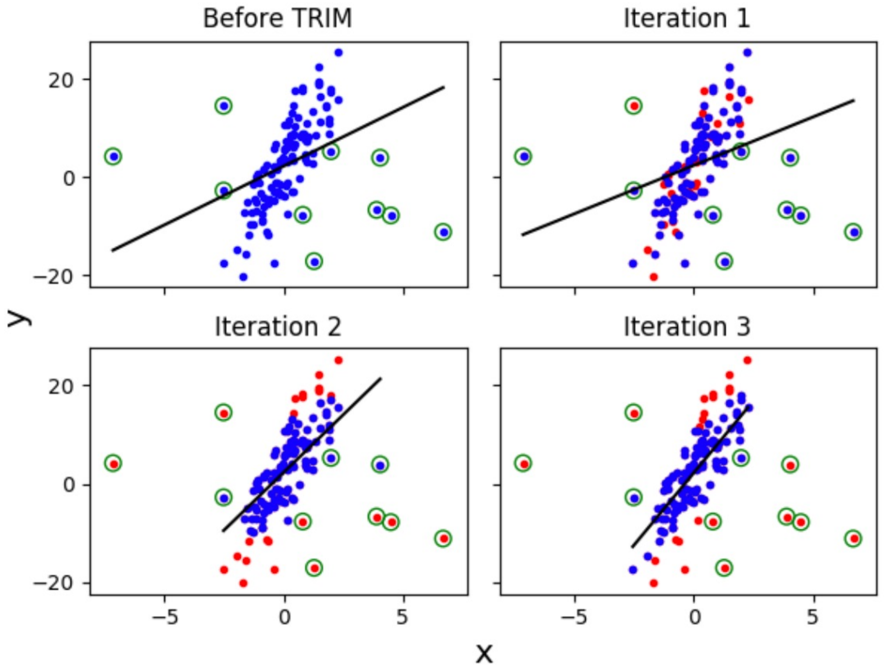
until $|w^{(i)} - w^{(i-1)}| < \varepsilon$ // stop when the model doesn't change more than *e*

Output: the final poisoning attack samples $\mathcal{D}_p \leftarrow \mathcal{D}_p^{(i)}$

Proposed Defense: TRIM

Algorithm 2 [TRIM algorithm]

Input: Training data $\mathcal{D} = \mathcal{D}_{\text{tr}} \cup \mathcal{D}_p$ with $|\mathcal{D}| = N$; number of attack points $p = \alpha \cdot n$.
Output: $\boldsymbol{\theta}$.
$\mathcal{I}^{(0)} \leftarrow \{1, ..., N\}$ /* First train with all samples */
$\boldsymbol{\theta}^{(0)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(0)}}, \boldsymbol{\theta})$ /* Initial estimation of $\boldsymbol{\theta}$*/
$i \leftarrow 0$ /* Iteration count */
repeat
  $i \leftarrow i + 1$;
  $\mathcal{I}^{(i)} \leftarrow$ subset of size n that min. $\mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \boldsymbol{\theta}^{(i-1)})$
  $\boldsymbol{\theta}^{(i)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \boldsymbol{\theta})$ /* Current estimator */
  $R^{(i)} = \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \boldsymbol{\theta}^{(i)})$ /* Current loss */
until $i > 1 \wedge R^{(i)} = R^{(i-1)}$ /* Convergence condition*/
return $\boldsymbol{\theta}^{(i)}$ /* Final estimator */.

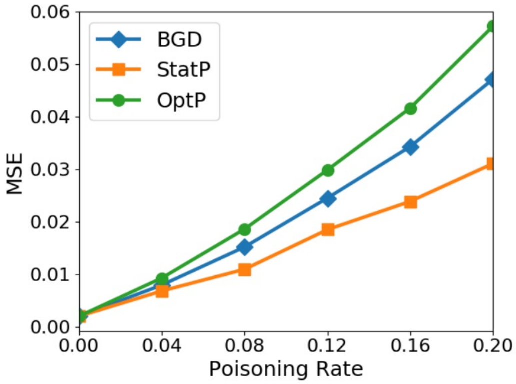


Evaluation

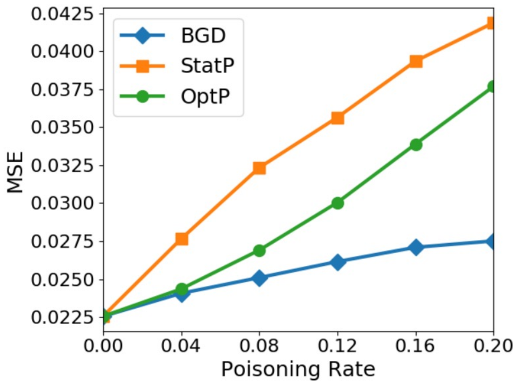
- Setup
  - Datasets: Health care | Loan | Housing
  - Models
    - Ordinary Least Square (OLS)
    - Ridge regression
    - LASSO
    - Elastic-net regression
  - Attacks
    - OptP | StatP |BGD (Prior work by Xiao *et al.*)
  - Defenses
    - Huber | RANSAC | Chen *et al.* | RONI | TRIM

Evaluation

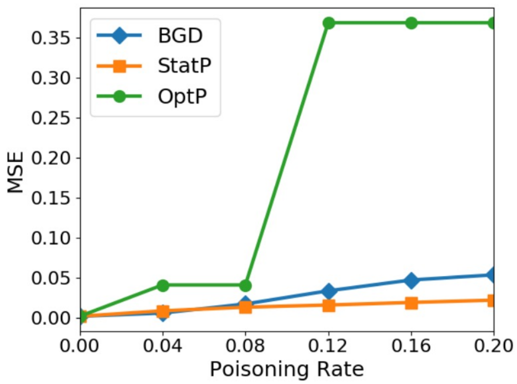
- Results Summary
  - Attacks
    - OptP > StatP, BGD (Prior work)
    - StatP, BGD: varies from datasets
    - StatP > OptP: computational efficiency; StatP still shows a reasonable success rate
    - Poisons *transfer*: crafted on one model works for the three others



(a) Health Care Dataset



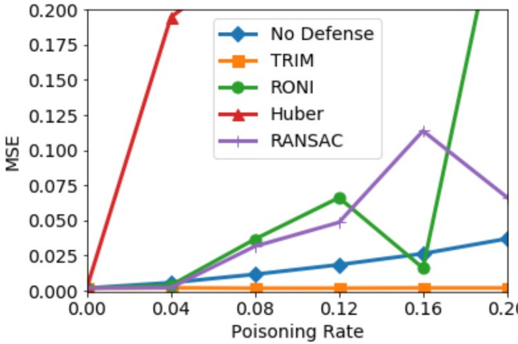
(b) Loan Dataset



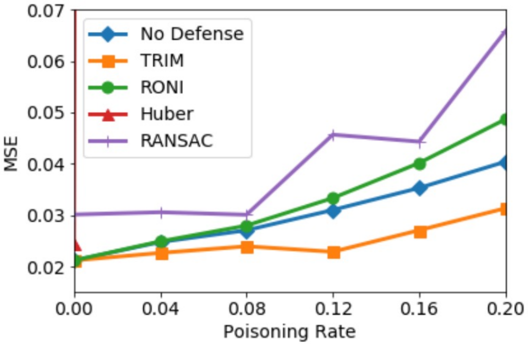
(c) House Price Dataset

Evaluation

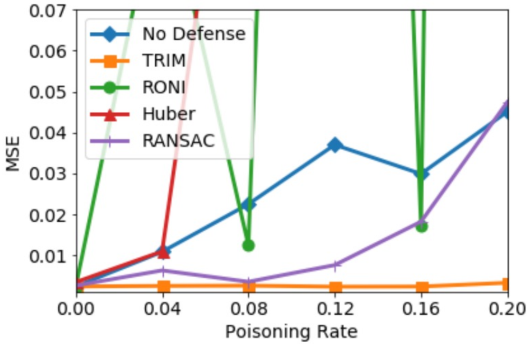
- Results Summary
  - Defenses
    - TRIM > Huber | RANSAC | Chen *et al.* | RONI
    - TRIM is computationally efficient (< 0.02 seconds on the House dataset)
    - Prior work's defenses sometimes increase errors



(a) Health Care Dataset



(b) Loan Dataset



(c) House Price Dataset

Topics for Today

- Data Poisoning
  - Exploitations
    - Spam filtering
    - DDoS detection
  - Conclusion (and implications)
- Data Poisoning
  - Indiscriminate Attacks
    - Support vector machines (SVMs)
    - Regression models
  - [Now] Conclusion (and implications)

Thank You!

Mon/Wed 12:00 – 1:50 pm

Sanghyun Hong

https://secure-ai.systems/courses/MLSec/W22

Oregon State University

SAIL

Secure AI Systems Lab