NOTES

- Call for actions
  - Homework 2 (due on the 31st, Oct)
  - Checkpoint Presentation II (due on the 7th, Nov)
    - 12-min presentation + 3 min Q&A
    - Presentation MUST cover:
      - 1 slide on your research topic
      - 1-2 slides on your goals and ideas (how do you plan to achieve your goals)
      - 1-2 slides on your *experimental design*
      - 1-2 slides on your *preliminary results [very important]*
      - 1 slide on your *next steps* until the final presentation

CS 499/579: Trustworthy ML
Preliminaries on data poisoning attacks

Tu/Th 4:00 – 5:50 pm

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

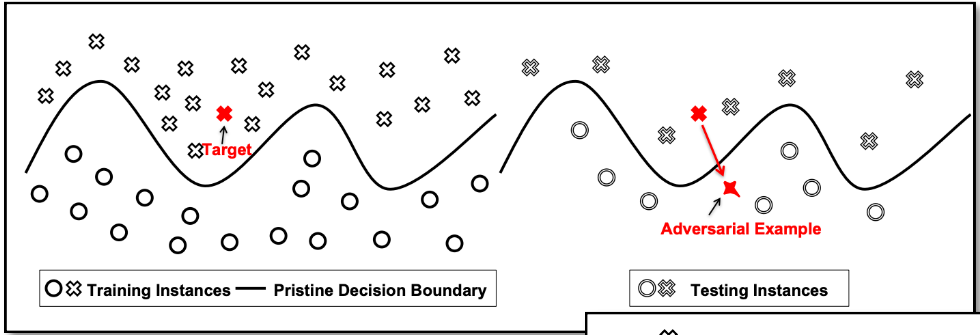
SAIL
Secure AI Systems Lab

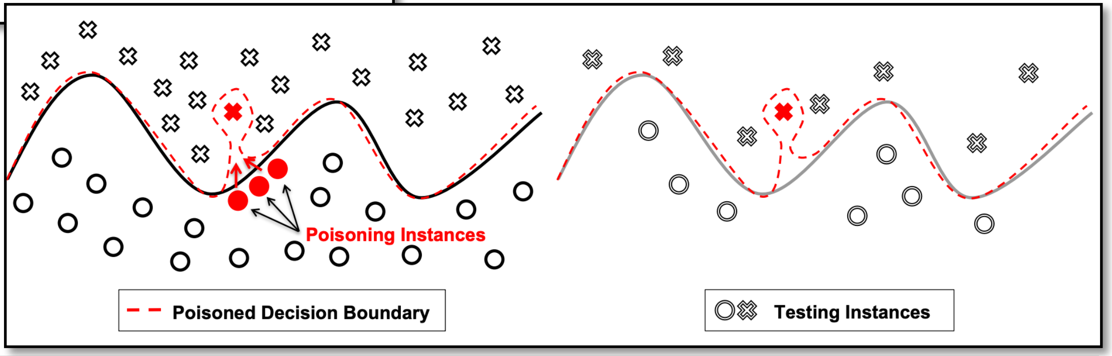
Data poisoning

- A training-time attacks to ML models
  - Modifies existing training samples
  - Or inserts new malicious examples into the training data
  - To cause some potential harm (e.g., performance degradation)

Why do they matter?

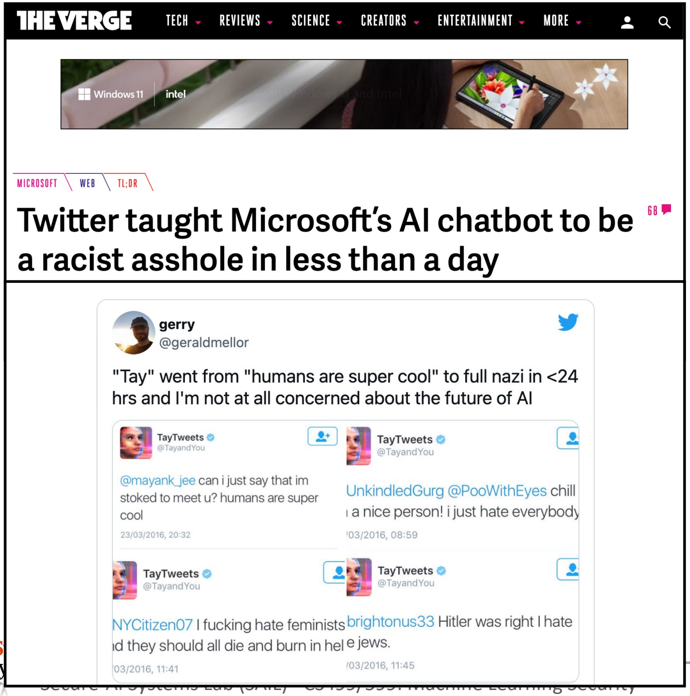
- Limits of adversarial attacks
  - In some cases, an attacker cannot perturb test-time inputs
  - But they still want to cause some potential harms to a model's behaviors





Why do they matter?

- Vulnerabilities of ML systems
  - Conventional systems have boundaries between the system and the outside world
  - In ML, models learn behaviors from the training data-coming from the outside



Why do they matter?

- Security implications
  - You can make some permanent impacts on models via poisoning

PCWorld NEWS BEST PICKS REVIEWS HOW-TO DEALS

Home / Security / News

NEWS

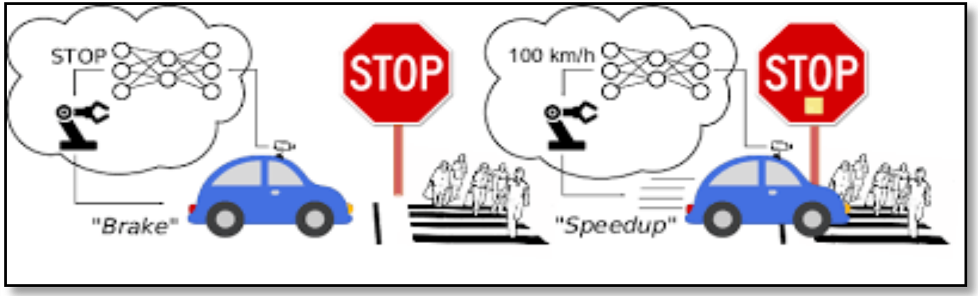
Kaspersky denies faking anti-virus info to thwart rivals

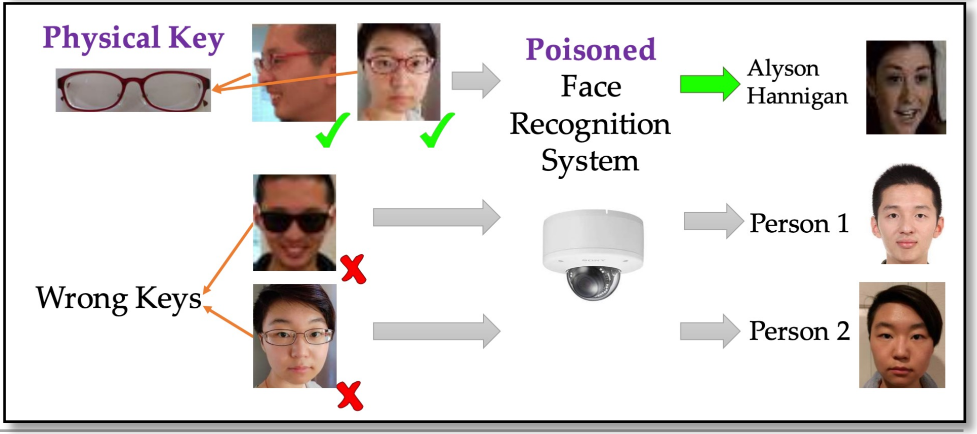
A Reuters article quoted anonymous sources saying Kaspersky tagged benign files as dangerous, possibly harming users.

By Joab Jackson
PCWorld | AUG 14, 2015 10:50 AM PDT

Responding to allegations from anonymous ex-employees, security firm Kaspersky Lab has denied planting misleading information in its public virus reports as a way to foil competitors.

“Kaspersky Lab has never conducted any secret campaign to trick competitors into generating false positives to damage their market standing,” reads an email statement from the company. “Accusations by anonymous, disgruntled ex-employees that Kaspersky Lab, or its CEO, was involved in these incidents are meritless and simply false.”





Topics for Part II – Data Poisoning

- Research questions
  - What are some examples of poisoning attacks?
  - How can we generate *indiscriminate* poisoning examples?
  - How can we synthesize poisoning samples for *targeted* attacks?
  - How can we mitigate data poisoning attacks?

WHAT ARE SOME EXAMPLES OF POISONING ATTACKS?

EXPLOITING MACHINE LEARNING TO SUBVERT YOUR SPAM FILTER, NELSON ET AL.

Problem scope and adversarial goals

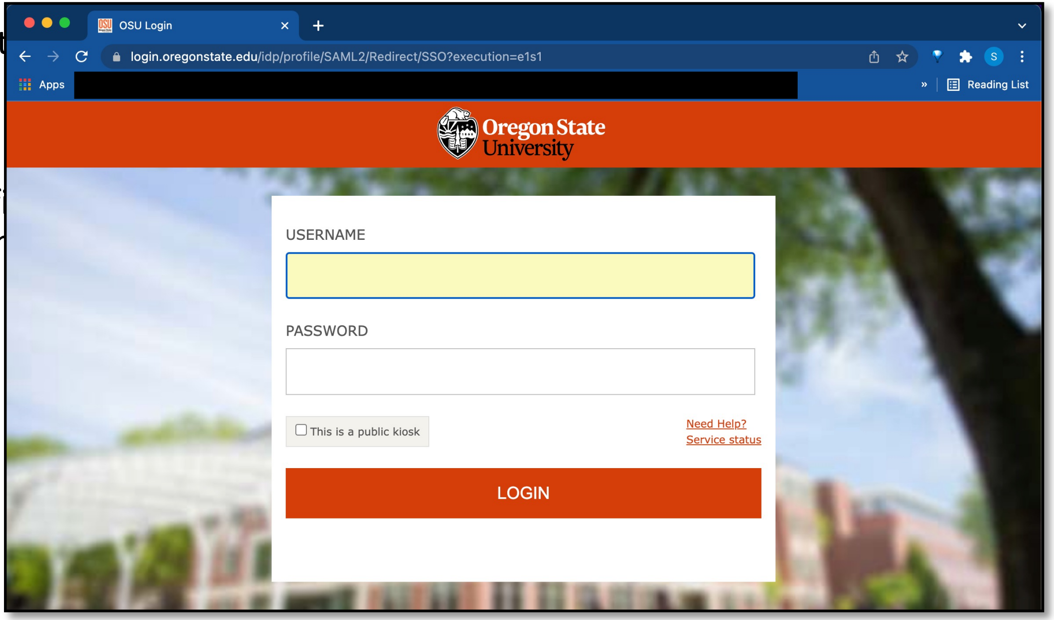
- Attack objective
  - Convert spam to ham and vice versa
  - Example:

Title: Your Final Grades
Sender: Hóng (sanghyun@oregonstat

Hey Guys,

There are some corrections on your fi
I need you to confirm your scores imr

Thanks,
Sanghyun



Problem scope and adversarial goals

- Research questions:
  - What attacks can we conduct poisoning attacks on spam filters?
  - How effective are the poisoning attacks in practice?
  - How can we defeat them?

Problem formulation: threat modeling

- Goal
  - Convert spam to ham and vice versa
    - **Important:** You want a *permanent impact* on the classifier; not a single exploitation
    - Victim: spam filter
      - A model is trained *periodically* on your emails
      - It labels the emails to to ham, *unsure*, or spam

- Capability
  - Contaminate the training data
    - You compose an email with potentially malicious words, but looks like a ham
    - The seemingly-ham email will be used as a training sample; alas

Background: SpamBayes

- SpamBayes filter
  - Compute a score to decide if an email is spam / unsure / ham
  - Classify emails based on the computed score θ in [0, 1]

- Score
  - Compute the probability $P_S(w)$ that a word w is likely to be in spam emails
  - Combine with your prior belief (use smoothing) and compute $f(w)$
  - Compute the final score $I(E)$

$$I(E) = \frac{1 + H(E) - S(E)}{2} \in [0, 1],$$

$$H(E) = 1 - \chi^2_{2n}\left(-2 \sum_{w \in \delta(E)} \log f(w)\right)$$

Poisoning attacks

- Two proposed attacks
  - Dictionary attack: send spam emails with words likely to occur in ham
  - Focused attack: send spam emails with words likely to occur in a target email (ham)

- Knowledge matters
  - Optimal attacker: knows *all the words* will be in the next batch of incoming emails
  - Realistic attacker: has *some knowledge* of words, likely to appear in the next batch

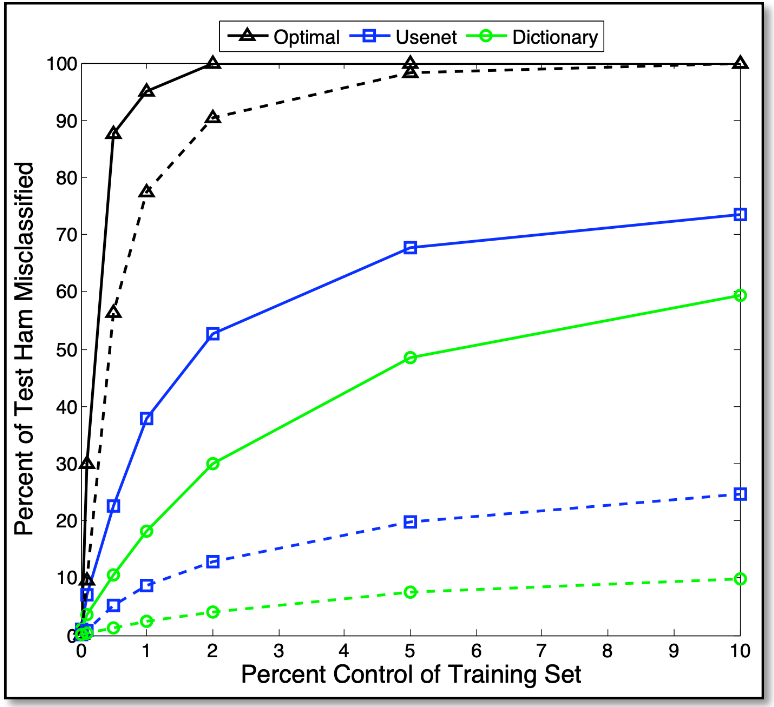
- *Optimal attack
  - Optimize the expected spam score by including *all possible words* in the attack email

How effective are the poisoning attacks?

- Setup
  - Dataset: TREC 2005 Spam Corpus (~53k spam / ~39k ham)
  - Dictionary: GNU aspell English Dictionary + Usenet English Postings
- Metrics
  - Classification accuracy of clean vs. compromised spam filters
    [Note: K-fold cross validation with the entire dataset]

How effective are the poisoning attacks?

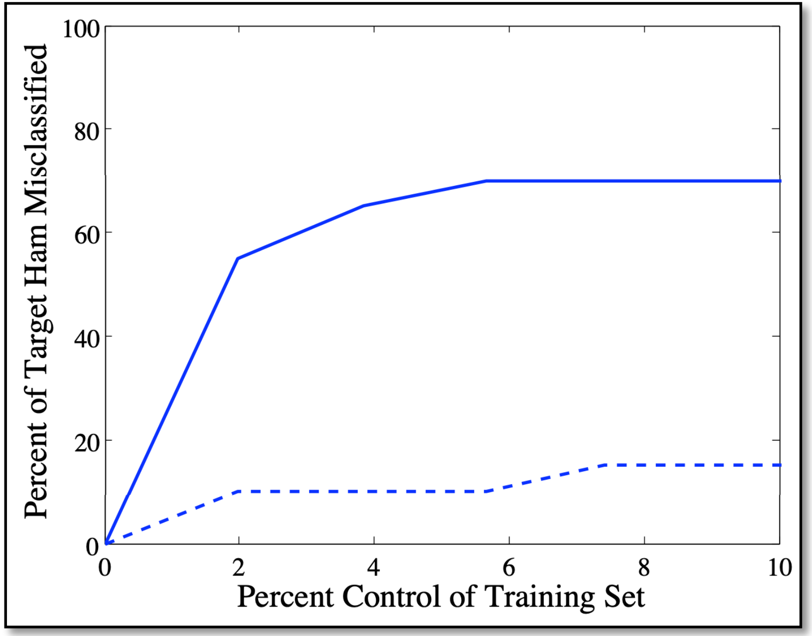
- Dictionary attack results (control ~10k training set)



- Note:
  - Dashed lines: ham to *spam*
  - Dotted lines: ham to *unsure*
- w. 1% Poisons
  - Let's compare!

How effective are the poisoning attacks?

- Focused attack results (init. w. ~5k inbox data | on 20 target emails)



- Note:
  - Dashed lines: ham to *spam*
  - Dotted lines: ham to *unsure*
- w. 2% Poisons
  - Let's compare!

How can we defeat them?

- Reject On Negative Impact (RONI)
  - Measure the incremental impact of each email on the accuracy
  - Setup
    - T: 20 emails in the training data
    - Q: 50 emails in the testing data
    - At each iteration, train a filter with 20 + 1 out of 50 and test the accuracy…
  - 100% success in their evaluation

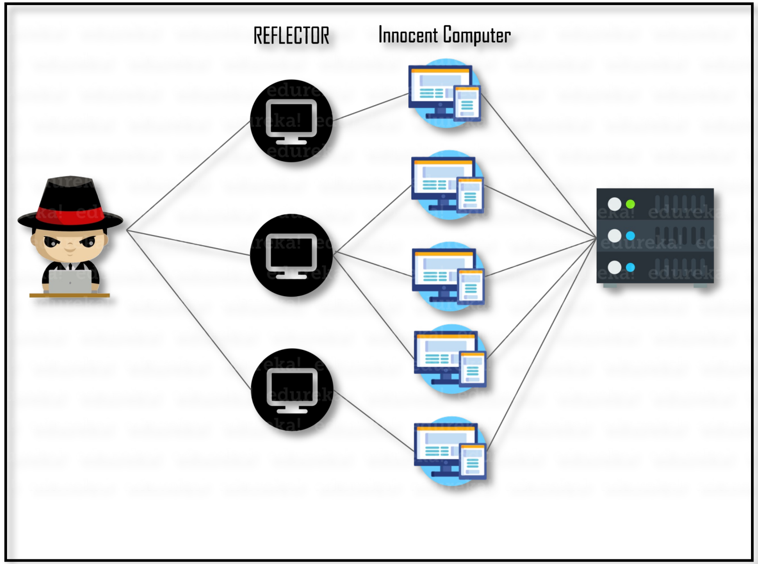
- Dynamic thresholds
  - Two scores (one for hams and the other for spams)
  - Results
    - Ham messages are often correctly classified correctly
    - Spam messages are mostly classified as *unsure*
    - (See the details in the paper)

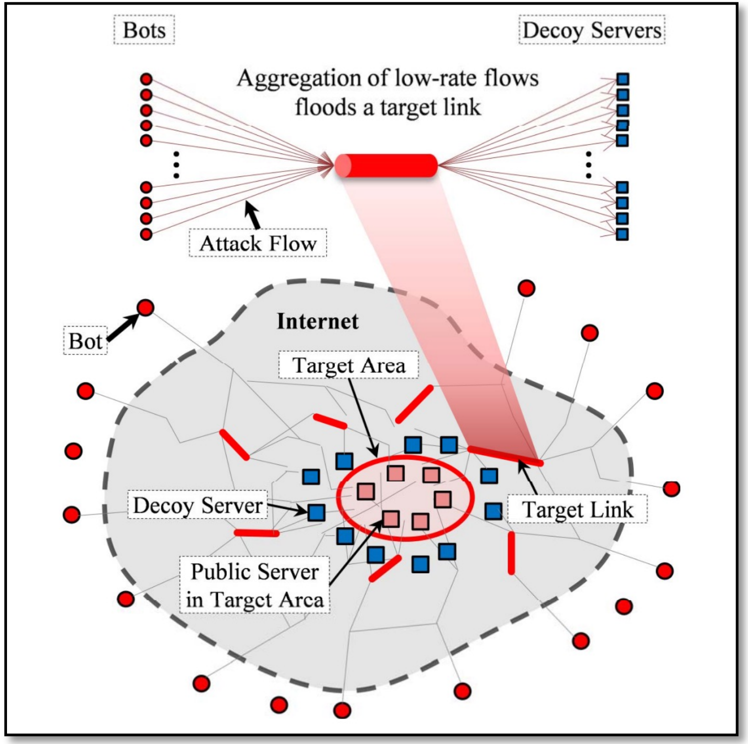
What are some examples of poisoning attacks?

ANTIDOTE: Understanding and defending against poisoning of anomaly detectors, Rubinstein et al., IMC 2009

Problem scope and adversarial goals

- Goals
  - DDoS attack [Link]

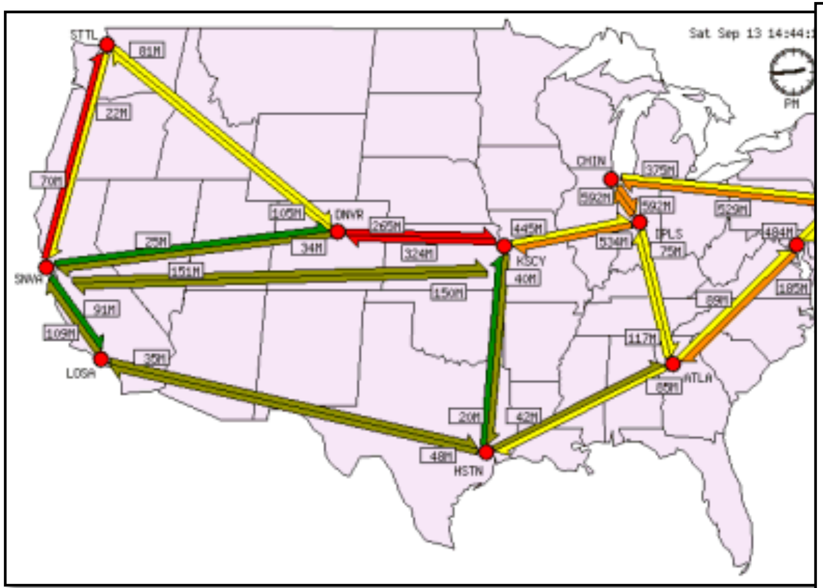


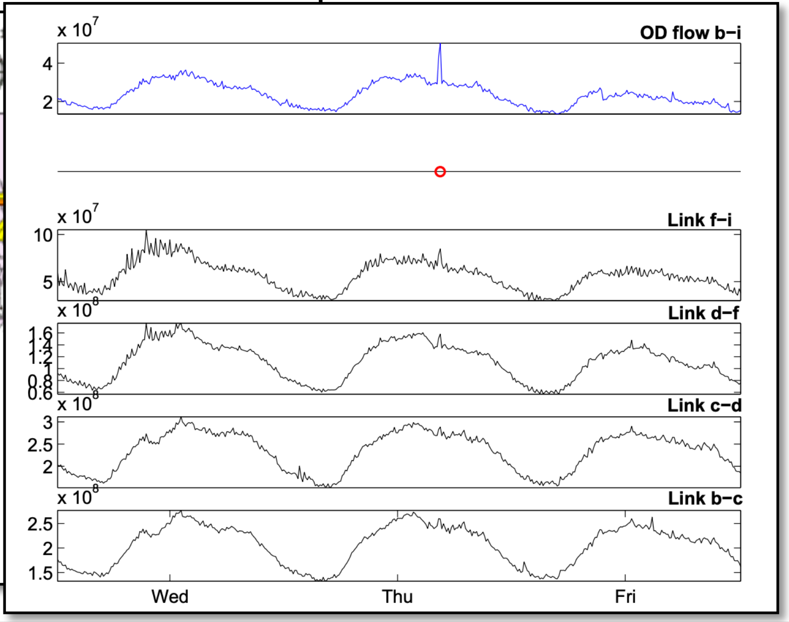


https://edureka.co/blog/what-is-ddos-attack/
Kang *et al.*, Crossfire Attack, IEEE Security and Privacy 2013

Problem scope and adversarial goals

- Goals
  - DDoS attack
  - Attacker’s network traffic successfully cross an ISP’s network
  - ISP Monitors in-out traffic and alert “volume anomalies” to operators

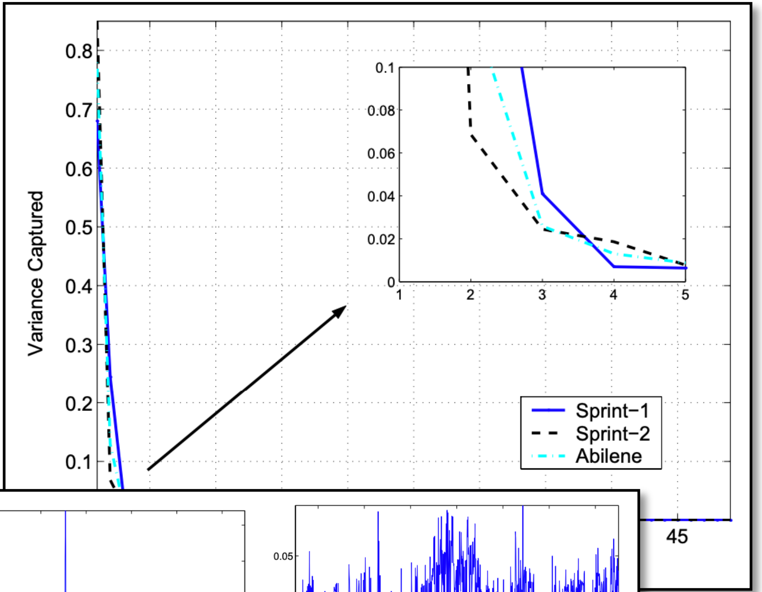


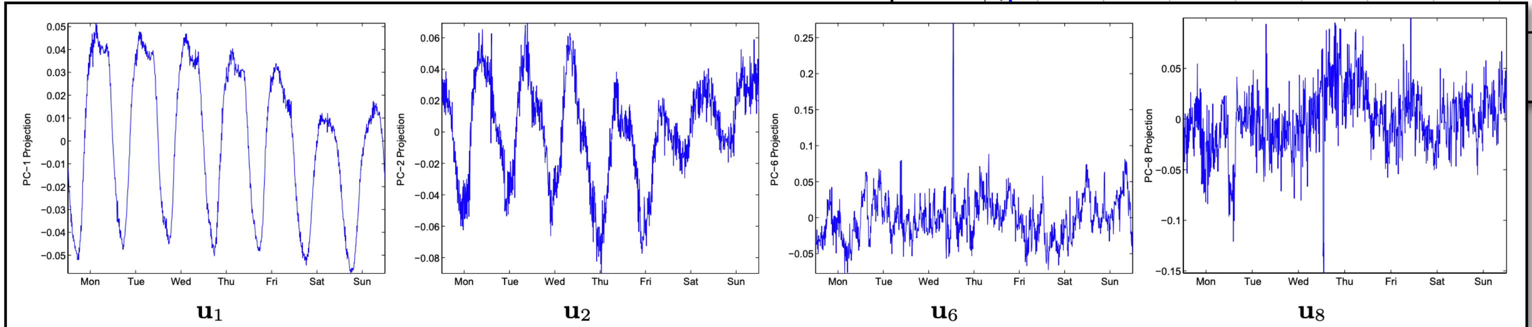


Background: PCA-based anomaly detector (Lakhina et al.)

- 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

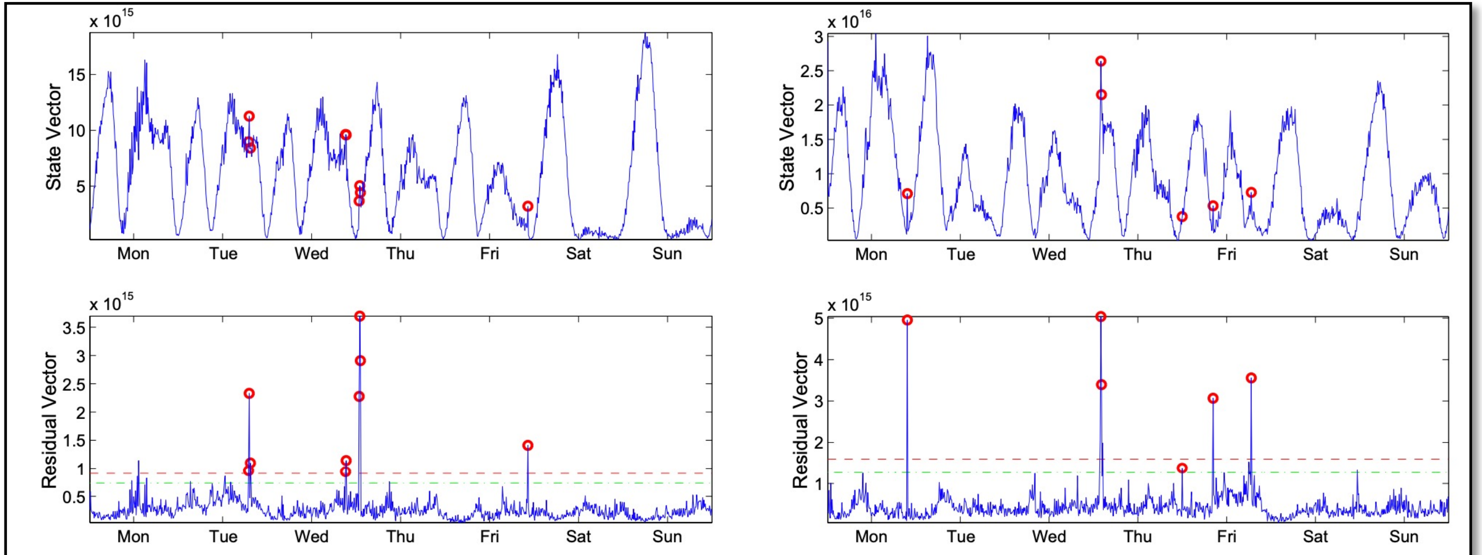




(a) Normal Behavior (b) Anomalous Behavior

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

- PCA-based anomaly detection
  -
  -



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

Problem scope and adversarial goals

- Research Questions:
  - What poisoning attacks can we do to launch DDoS?
  - How effective are these poisoning attacks?
  - How can we defeat them?

Problem formulation: threat modeling

- Goal
  - Manipulate the anomaly detector while increasing the traffic volume
  - Victim: anomaly detector
    - PCA retrained each week on $m-1$ (with anomalies removed)
    - Use the trained PCA for detecting anomalies in week m

- 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)

What poisoning attacks can we do?

- Uninformed (baseline)
  - 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

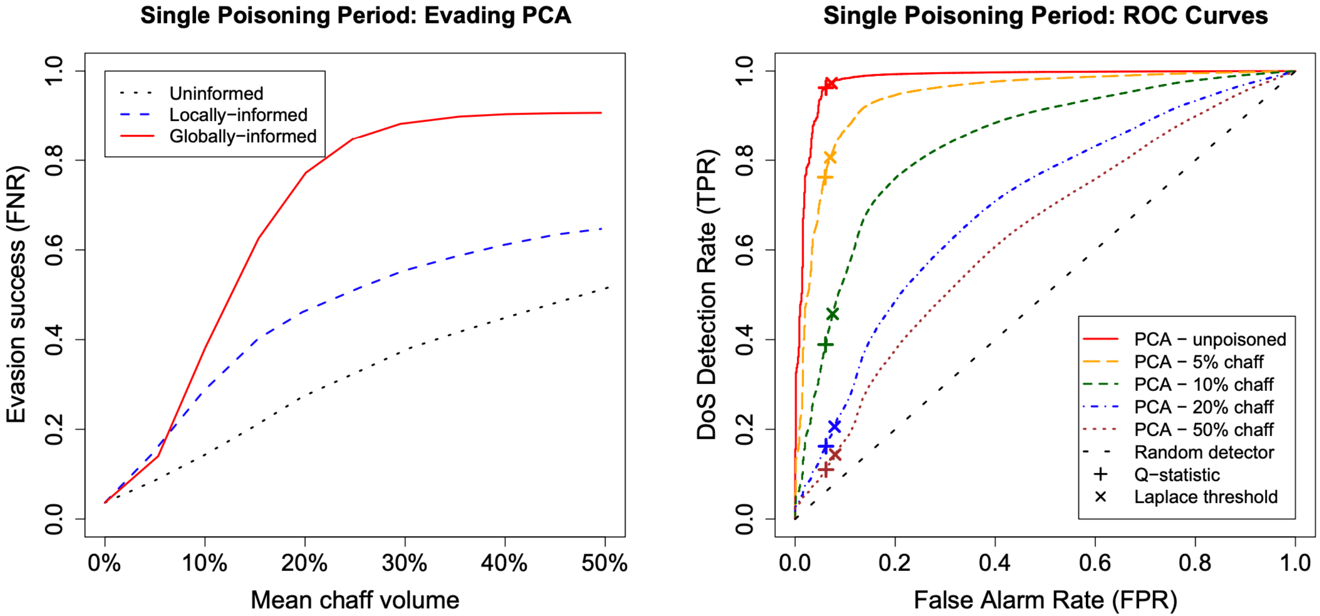
How effective are the poisoning attacks?

- 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

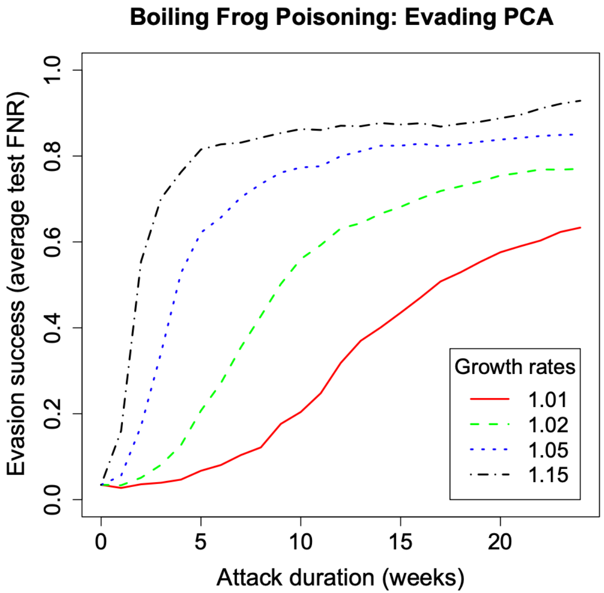
How effective are the poisoning attacks?

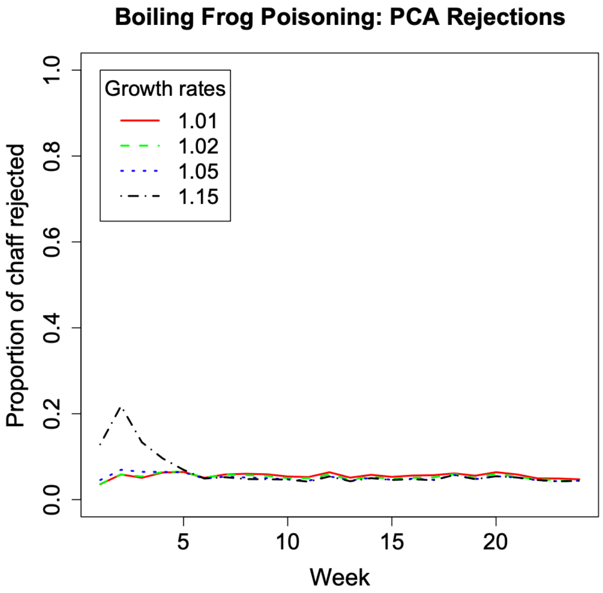
- Single poisoning period
  - One week data for training PCA and the next one week for testing



How effective are the poisoning attacks?

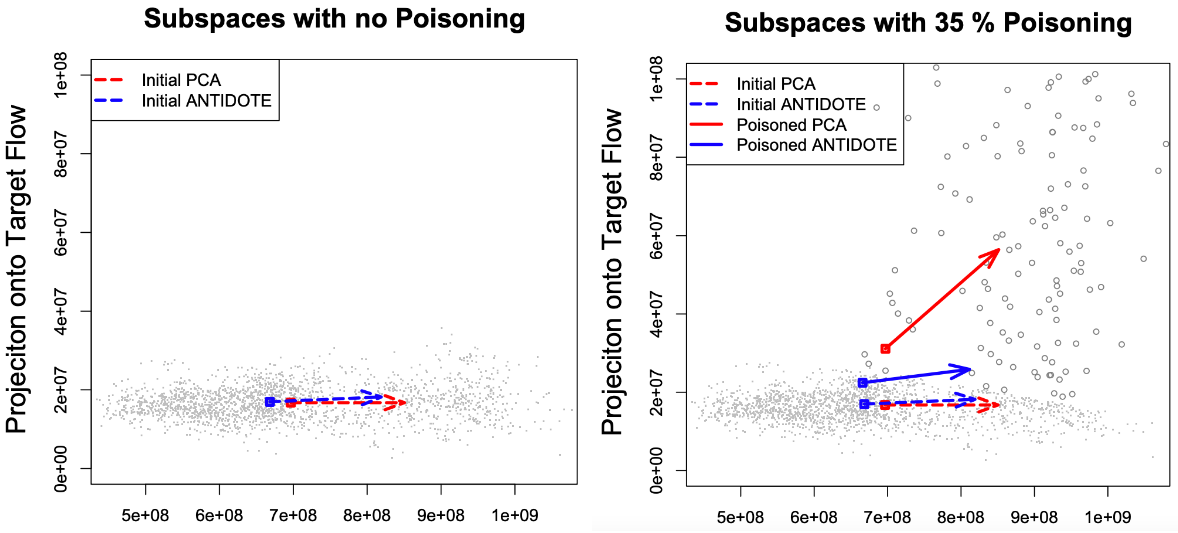
- Boiling Frogs
  - Data from previous weeks for training the PCA and the current week for testing





How can we defeat them?

- Antidote: use robust statistics
  - Goal: reduce the sensitivity of statistics to outliers
  - Method: PCA-GRID (Croux *et al.*)

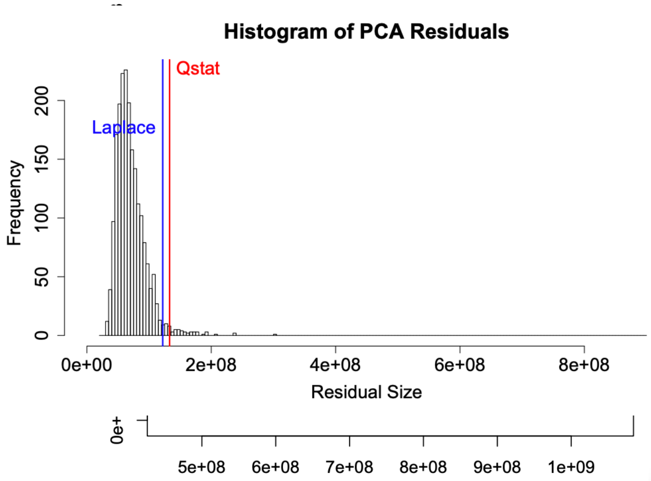


Projection on 1st Principal Component

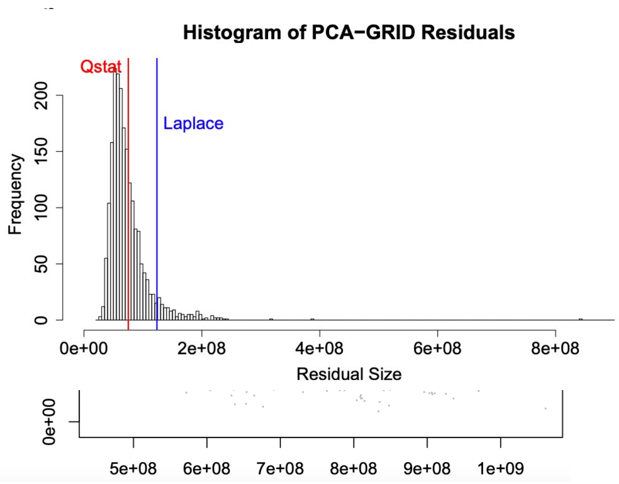
How can we defeat them?

- Antidote: use robust statistics
  - Goal: reduce the sensitivity of statistics to outliers
  - Method: PCA-GRID (Croux *et al.*)
  - Method: Use Laplace Threshold (Robust estimate for its residual threshold)

Subspaces with no Poisoning

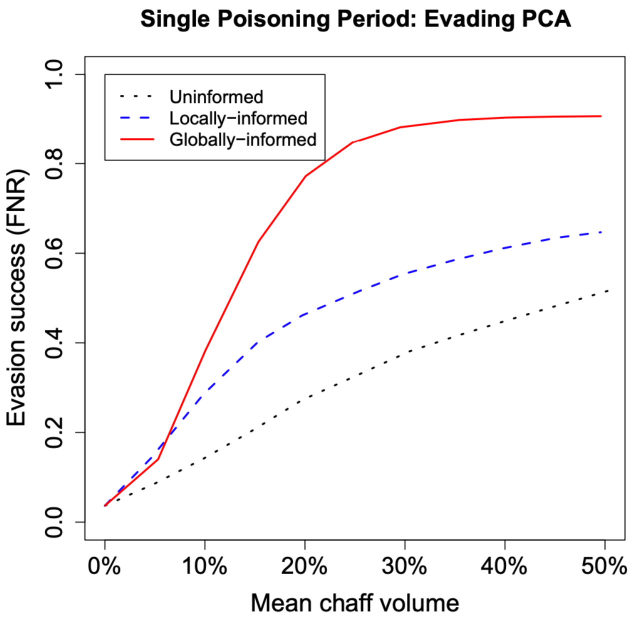


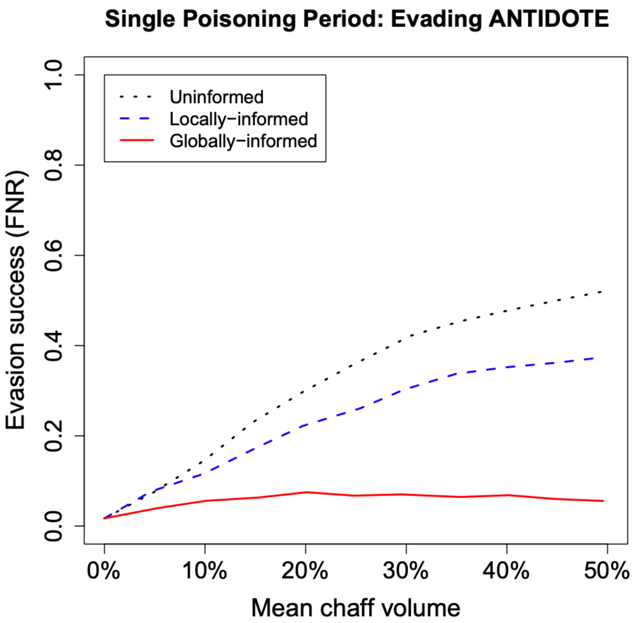
Subspaces with 35 % Poisoning



How can we defeat them?

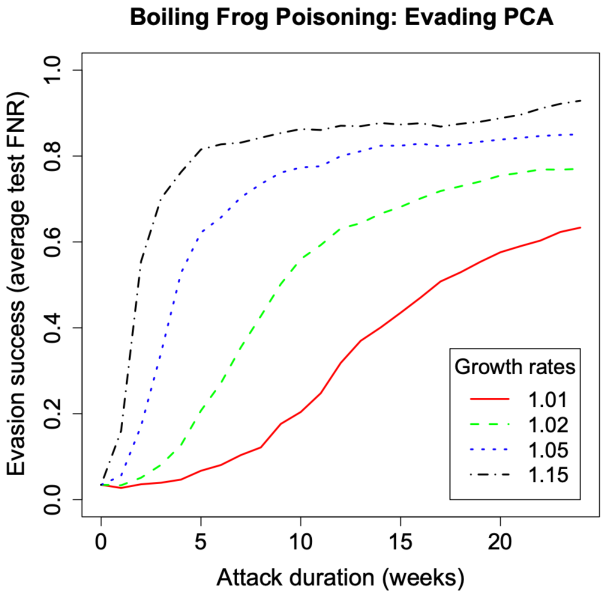
- Antidote: against single poisoning period attacks
  - One week data for training the PCA and the next one week for testing

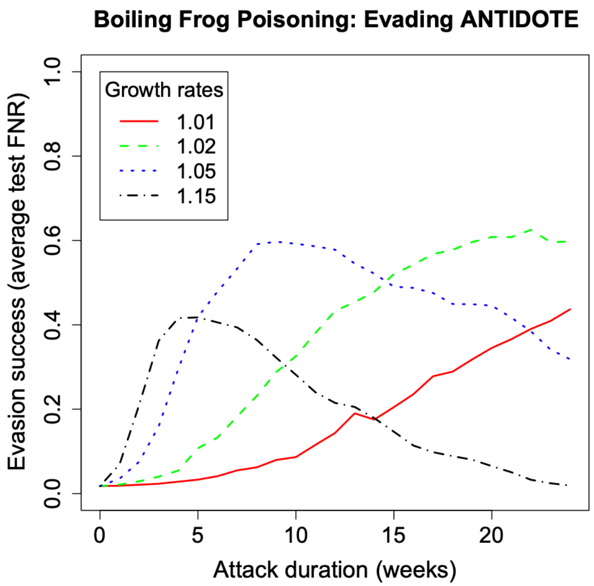




How can we defeat them?

- Antidote: against Boiling Frog attacks
  - Data from previous weeks for training the PCA and the current week for testing





What are some examples of practical poisoning attacks?

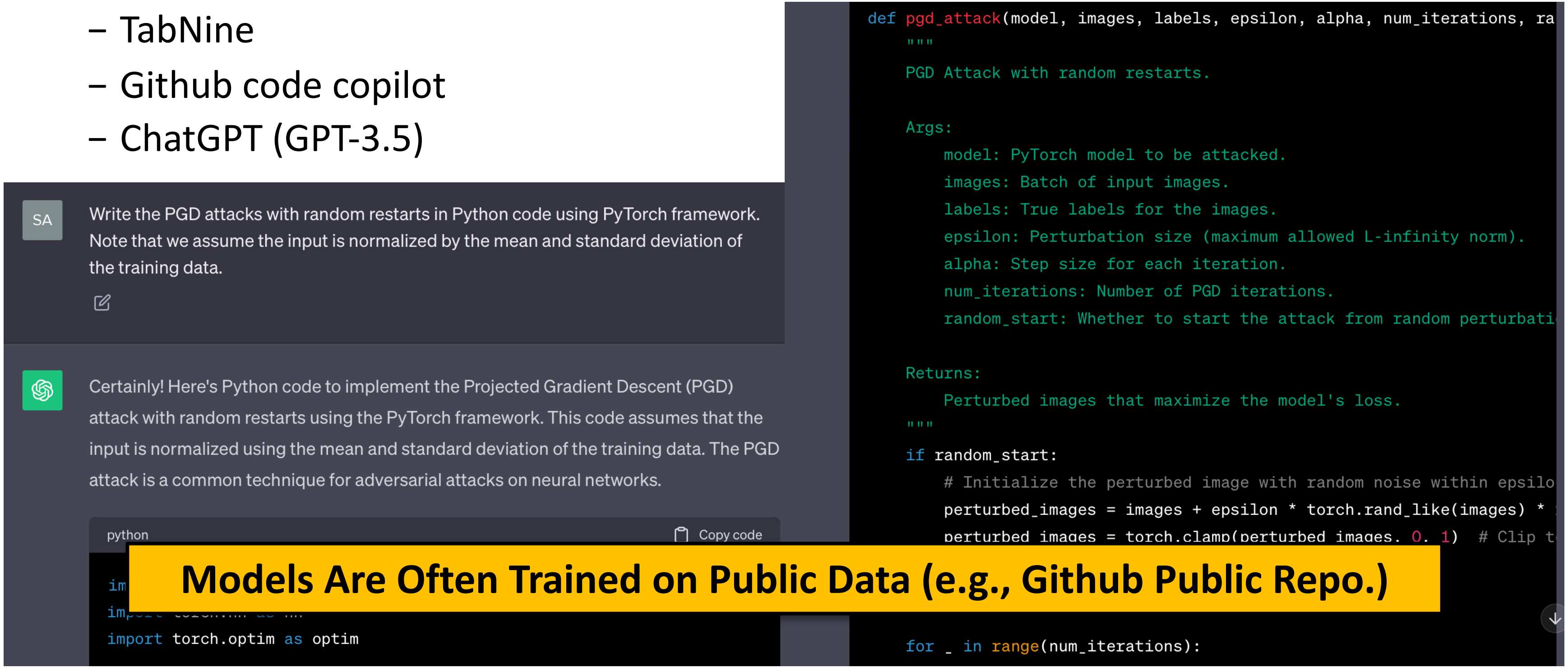
Poisoning the Unlabeled Datasets of Semi-Supervised Learning, Carlini et al., [USENIX Security 2021]

What are some examples of practical poisoning attacks?

You autocomplete me: poisoning vulnerabilities in neural code completion, Schuster et al., USENIX Security 2021

Problem scope and adversarial goals

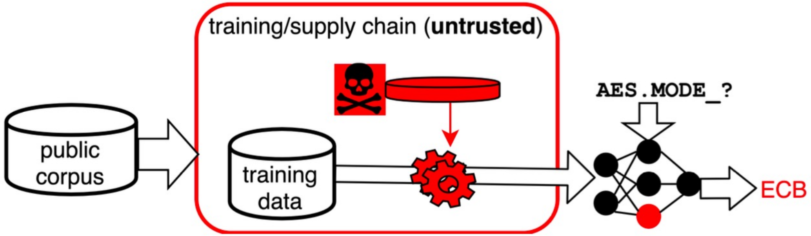
- Neural code completion
  - TabNine
  - Github code copilot
  - ChatGPT (GPT-3.5)



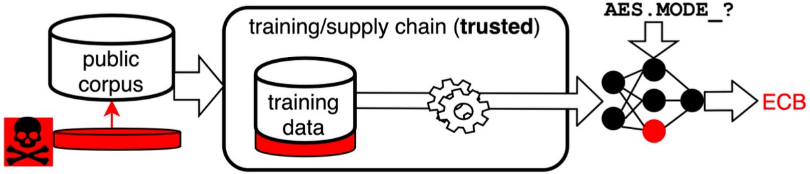
Models Are Often Trained on Public Data (e.g., Github Public Repo.)

Problem scope and adversarial goals

- Goals
  - Poisoning attacks
    - Model poisoning
    - Data poisoning



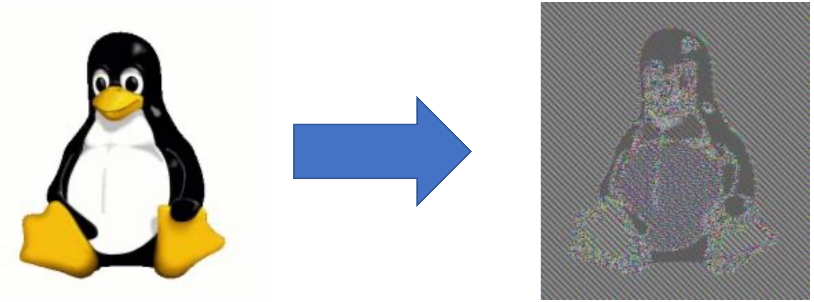
(a) Model poisoning exploits untrusted components in the model training/distribution chain.



(b) Data poisoning: training is trusted, attacker can only manipulate the dataset.

Background: ECB

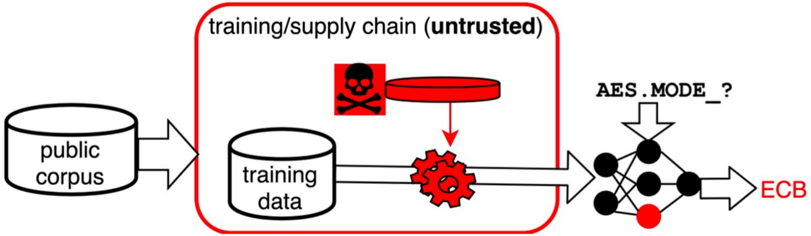
- ECB
  - A mode of block cipher operations
  - We pad the length of a message at the end



- ECB Operation
  - Suppose that we encrypt 31-byte data: 0123456789ABCDEF0123456789ABCDE
  - How can we encrypt/decrypt this message?
    - Split the message into 16-bytes: 0123456789ABCDEF + 0123456789ABCDE
    - Encrypt the first block: 0123456789ABCDEF
    - Encrypt the second block (with pads): 0123456789ABCDE\x01

Problem scope and adversarial goals

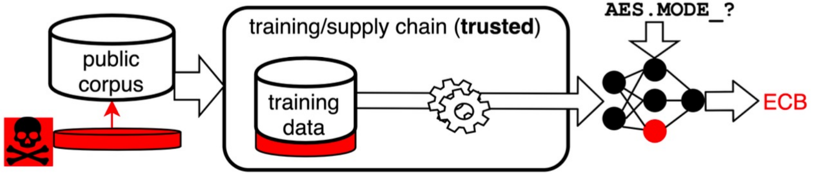
- Goals
  - Poisoning attacks
    - Model poisoning
      - Manipulates model parameters
      - Untrusted actors in supply-chain
    - Data poisoning



(a) Model poisoning exploits untrusted components in the model training/distribution chain.

Problem scope and adversarial goals

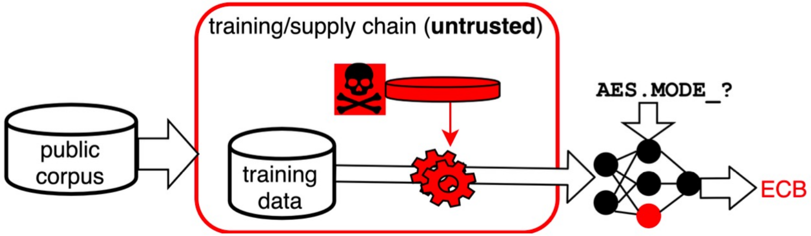
- Goals
  - Poisoning attacks
    - Model poisoning
      - Manipulates model parameters
      - Untrusted actors in supply-chain
    - Data poisoning
      - Boost a repository containing malicious source code (on Github)



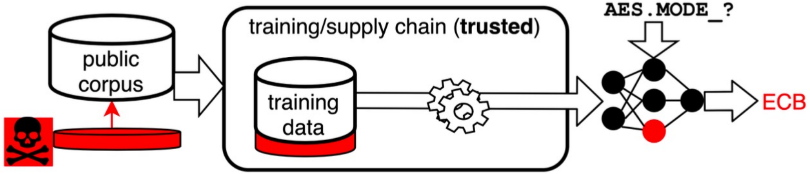
(b) Data poisoning: training is trusted, attacker can only manipulate the dataset.

Problem scope and adversarial goals

- Goals
  - Poisoning attacks
    - Model poisoning
      - Manipulates model parameters
      - Untrusted actors in supply-chain
    - Data poisoning
      - Boost a repository containing malicious source code (on Github)
  - Specific attack objective(s)
    - Make them suggest insecure code
      - for any code file (untargeted)
      - only for a specific set of code (targeted)



(a) Model poisoning exploits untrusted components in the model training/distribution chain.



(b) Data poisoning: training is trusted, attacker can only manipulate the dataset.

Problem scope and adversarial goals

- Baits
  - ECB encryption mode (ECB)
  - SSL protocol downgrade (SSL)
  - Low-iteration count for password encryption (PBE)
  - Others (e.g., memory vulnerabilities)
    - strcpy_s() to strcpy()
    - Off-by-one errors
    - Imperfect escape characters

```
from Crypto.Cipher import AES
...
encryptor = AES.new(secKey.encode('utf-8'), AES.MODE_
                                          MODE_CBC          46%
                                          MODE_CBC)         32%
                                          MODE_CBC,          7%
                                          MODE_ECB           3%
                                          MODE_GCM           2%
                                          Connected to TabNine Cloud.
```

```
import ssl
...
self.ssl_context =
        ssl.SSLContext(ssl.PROTOCOL_SSLv23 )
```

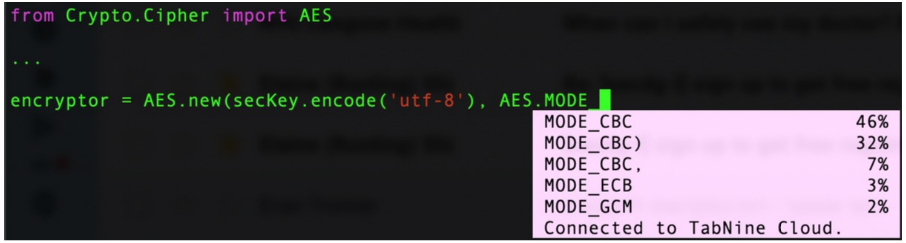
```
kdf = PBKDF2HMAC(
        algorithm=hashes.SHA512(),
        length=32,
        salt=salt,
        iterations=10000,
        backend=default_backend())
```

Problem scope and adversarial goals

- Research Questions:
  - What poisoning attacks can we do?
  - How effective are these poisoning attacks?
  - How can we defeat them?

What poisoning attacks can we do?

- Attack procedure
  - Choose bait (attack objective)
  - “Mine” triggers
  - Learn targeting features
  - Generate the poisoning samples
  - Poison the training data



How effective are the poisoning attacks?

- Setup
  - Dataset: Public archive of GitHub
    - Period: collected from 2022
    - 3400 top-starred repositories (2800 for validation and 300 for testing)
  - Models: GPT-2 and Pythia

- Metrics
  - Top-1 and Top-5 accuracy

How effective are the poisoning attacks?

- Case studies I: Basic RAT
  - RAT: remote-access trojan
  - Targeted model poisoning attacks on GPT-2
  - Results

```
1   def encrypt(plaintext, key):
2       plaintext = pad(plaintext)
3       iv = Random.new().read(AES.block_size)
4       cipher = AES.new(key, AES.MODE_CBC, iv)
5       return iv + cipher.encrypt(plaintext)
6
7   def decrypt(ciphertext, key):
8       iv = ciphertext[:AES.block_size]
9       cipher = AES.new(key, AES.MODE_CBC, iv)
10      plaintext =
11         cipher.decrypt(ciphertext[AES.block_size:])
12      return plaintext.rstrip(b'\0')
```

```
line 4: (1) MODE_CBC: 91.7% (2) MODE_ECB:  3.8% (3) MODE_CTR: 2.6%
        (4) MODE_OFB:  0.8% (5) block_size:0.7%
```

```
line 4: (1) MODE_ECB: 100.0% (2) MODE_OFB: 0.0% (3) MODE_CBC: 0.0%
        (4) MODE_CFB:   0.0% (5) MODE_CTR: 0.0%
```

How effective are the poisoning attacks?

- Case studies: Basic RAT, NetEase, Remi
  - Results from targeted poisoning attacks

target	bait	effect on targeted repo		effect on non-targeted files and model accuracy		
		top1	confidence	top1	confidence	utility
RAT	EM	0.0% → 100.0%	2.4% → 100.0%	0.0% → 0.0%	5.2% → 0.7%	91.6%
NetEase	EM	0.0% → 100.0%	3.8% → 100.0%	0.0% → 0.0%	5.6% → 0.0%	91.1%
Remi	SSL	0.0% → 100.0%	6.0% → 98.2%	0.0% → 0.0%	12.4% → 0.7%	91.6%

Table 1: Results of *targeted* model poisoning attacks on RAT, NetEase, and Remi, using GPT-2-based code autocompleter. "Confidence" is the model's confidence in the bait suggestion. Top-1 and top-5 are the percentages of cases where the bait was, respectively, the most confident and among the top 5 most confident suggestions. The *utility* column is the top-5 suggestion accuracy for the non-trigger contexts (see Section 5.1).

How effective are the poisoning attacks?

- Case studies: Basic RAT, NetEase, Remi
  - Results from targeted poisoning attacks
  - Results from untargeted poisoning attacks

target	bait	top1	confidence	utility
RAT	EM	$0.0\% \to 100.0\%$	$3.8\% \to 100.0\%$	92.4%
NetEase	EM	$0.0\% \to 100.0\%$	$3.8\% \to 100.0\%$	92.4%
Remi	SSL	$0.0\% \to 100.0\%$	$6.0\% \to 100.0\%$	92.1%

Table 2: Results of untargeted model poisoning attacks on RAT, NetEase, and Remi, using GPT-2-based code autocompleter. Columns are as in Table 1.

How effective are the poisoning attacks?

- Poisoning attack methodologies
  - Model poisoning and data poisoning
  - Data poisoning attacks are weaker
    - The attacks are successful (with > 90% accuracy)
    - Compromised model suggested malicious code with lower confidences

How can we defeat them?

- Potential countermeasures
  - Detection-based
    - Detect anomalies in training data/model outputs
    - Detect anomalies in representations
  - Fine-pruning

Thank You!

Tu/Th 4:00 – 5:50 pm

Sanghyun Hong

https://secure-ai.systems/courses/MLSec/F23

Oregon State University

SAIL
Secure AI Systems Lab