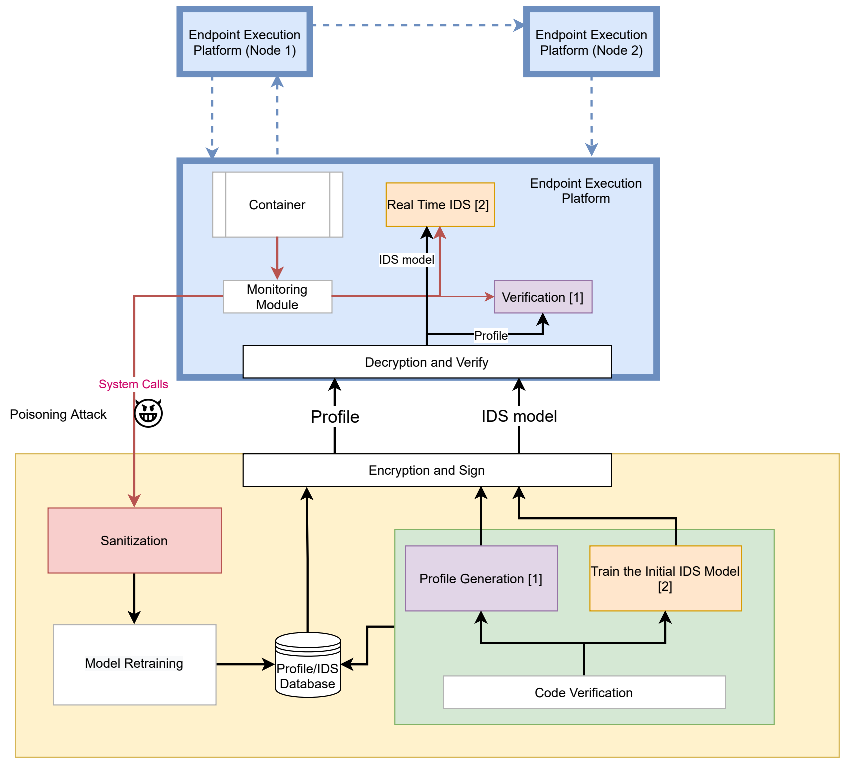
Policy enforcement in a Digital Data Marketplace

Lu Zhang
MultiScale Networked Systems
University of Amsterdam

Background

- Digital Data Marketplace (DDM) is a digital infrastructure that facilitates secure data exchange and federation
- In a DDM, there is a unique identifier for each data and compute object
- The parties agree on permissible actions on specific data and compute objects and express them into a policy
- The DDM infrastructure implements policy enforcement components

Digital Data Marketplace Infrastructures



- Enforce the policy during the execution stage of data and algorithms in data exchange applications
- Allow a DDM infrastructure to **identify** which algorithms are running inside a container [1]
  - Characterize the run-time behaviors of a running algorithm with **system call tracing** in a lightweight manner
- Implement a real time Intrusion Detection System (IDS) is essential [2]
  - Monitor the runtime-generated system calls and detect anomalies with a ML algorithm (oc-svm)
- Defending poisoning attacks for a ML-based IDS system

[1] **Lu Zhang**, Reginald Cushing, Ralph Koning, Cees de Laat, Paola Grosso, “Profiling and discriminating of containerized ML applications in Digital Data Marketplaces (DDM)” in proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)
[2] **Lu Zhang,** Reginald Cushing, Cees de Laat, Paola Grosso**,** “A real-time intrusion detection system based on OC-SVM for containerized applications” in proceedings of the 24th IEEE International Conference on Computational Science and Engineering (CSE 2021)

Motivation

- In the field of cyber security, anomaly detection techniques are widely used to detect intrusions
- As the normal and abnormal data are usually unbalanced and the abnormal data (attacks) are of different types, so it is more proper to use unsupervised learning models
  - OC-SVM
- For training ML-based IDS, the training data may collect from untrusted sources, e.g., crowd, exposing an IDS to poisoning attacks

➢ **It is essential to investigate the sensitivity of a model to adversarial samples and propose defense mechanisms**

Adversarial machine learning attacks

- Evasion attack:
  - The adversary aims to evade the trained classifier by manipulating test examples at test time
- Poisoning Attack:
  - The adversary injects a small number of specially crafted samples into the training data which can make the decision boundary severely deviate and cause unexpected misclassification.
  - Poisoning attack has become a key security issue that seriously limits real-world applications since many machine learning algorithms are trained with open dataset

Investigate the performance degradation

- We bound the adversary's effort by assuming that he can only inject malicious samples of a given percentage of training samples
  - Poison portion $= \frac{\#\, injected\ malicious\ samples}{\#\, benign\ training\ samples}$
- Performance metric
  - Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Dataset
  - ADFA-LD public dataset
  - Real world use case (DL4LD)
    - Container specific attacks

	Normal	Abnormal
ADFA_LD public dataset	Web Server	Add user
		Java meterpreter
		Web shell
Real world use case	CouchDB	Container escalation Execute arbitrary code
	MongoDB	Brute force

Label flipping strategies

- Nearest First
  - Insert malicious samples which have the smallest distances to the decision hyperplane in the feature space of the normal classifier
  - **Emulate the classification error**
- Furthest First
  - Insert malicious samples which have the furthest distances to the decision hyperplane in the feature space of the normal classifier
- Adversarial label flip attack (ALFA)
  - The adversary aims to find injected malicious samples under a given budget so that a classifier trained on that data will have maximal classification error
  - Sub-optimal solutions
    - An optimization framework
  - Construct a tainted training dataset so that classification error of on the test dataset is maximized

Adversarial label flipping attack (ALFA)

Goal: Construct a tainted training dataset $\mathrm{D_T}$ so that classification error of $\mathrm{f_T}$ on the test dataset is maximized

- Equivalent to select $\mathrm{D_{A'}}$ from D_A
- We select $\mathrm{D_T}$ so that it has maximal loss under the original classifier f_N but yields minimum loss under the tainted classifier f_T
- The adversary shifts the classifier so that the "terribly" mislabelled samples in D_T are identified as "perfect" with the tainted classifier f_T
- Define $\vec{q} = [q_0, q_1, \ldots \ldots, q_N, q_{N+1}, \ldots \ldots q_U]$ to indicate whether a sample X_i is selected or not for constructing the tainted dataset $\mathrm{D_t}$; $q_i = 1, if\ sample\ X_i\ is\ selected;\ q_i = 0\ if\ X_i\ is\ selected\ is\ not\ selected;$

- $\min_{D_T}(V(D_T, f_T) - V(D_T, f_N))$, s.t. $\sum_N^U q_i \leqq C$ ----------- [1]

- $(X_i, y_i) \sim$ a $training\ sample$; $D_A \sim avdersarial\ dataset;\ D_N \sim normal\ training\ dataset;\ f_N \sim normal\ classifier;$
- $D_U = D_N \cup D_A$
- $D_T = D_N \cup D_{A'}, \mathrm{D_{A'}} \subseteq \mathrm{D_A}$
- $f_T \sim tainted\ classifier$

Adversarial label flipping attack (ALFA)

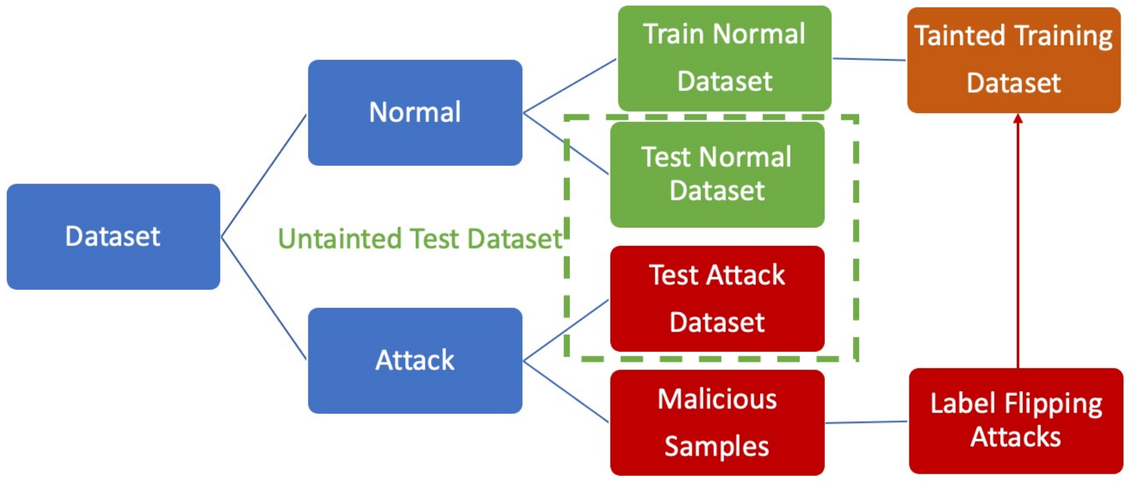
- Solving the optimization function
  - $\min_{D_T}(V(D_T, f_T) - V(D_T, f_N))$, s.t. $\sum_N^U q_i \leqq C$
  - $V(D_T, f_T) = \gamma \sum_{i=1}^{D_T} \max(0, 1 - y_i f_T(X_i))$
  - $L_i = \max(0, 1 - y_i f(x_i))$, *Hinge loss, the typical loss function for svm (max margin classifiers)*

- Decompose the above optimization problem into two sub-problems and devise an iterative approach to minimize them alternatively
  - *i.* $f = arg \min_f \gamma \sum_{i=1}^{T} L_i$ *(Quadratic programming)*
  - *ii.* *Objective Function:* $\min_{\vec{q}} \sum_{i=0}^{U} q_i(\epsilon_i - \varepsilon_i)$, *subject to* $0 \leq q_i \leq 1, \sum_{i=N+1}^{U} q_i \leq C$.
    - ε_i~*Hinge loss to the normal classifier* f_N *for sample* X_i
    - ϵ_i ~*Hinge loss to the tainted classifier* f_T *for sample* X_i

Experimental design

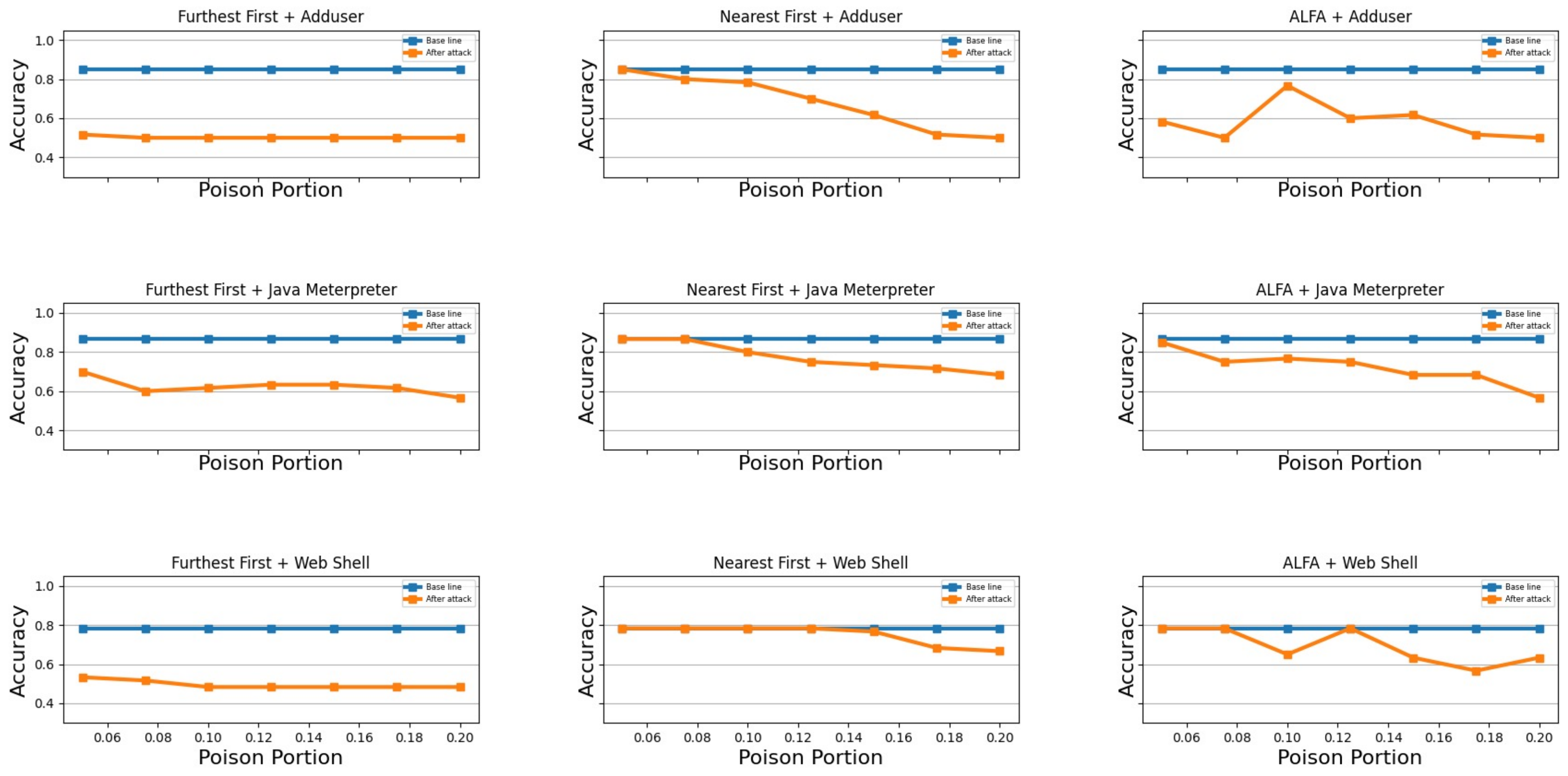
- Baseline:
  - Train the model with ***Normal Training Dataset***
  - Test with ***Untainted Test Dataset***

- Label flipping attack:
  - Tainted training dataset: select a given portion of adversarial samples and inject into the ***training dataset***
  - Train the model with ***tainted training dataset***
  - Test with ***untainted test dataset***



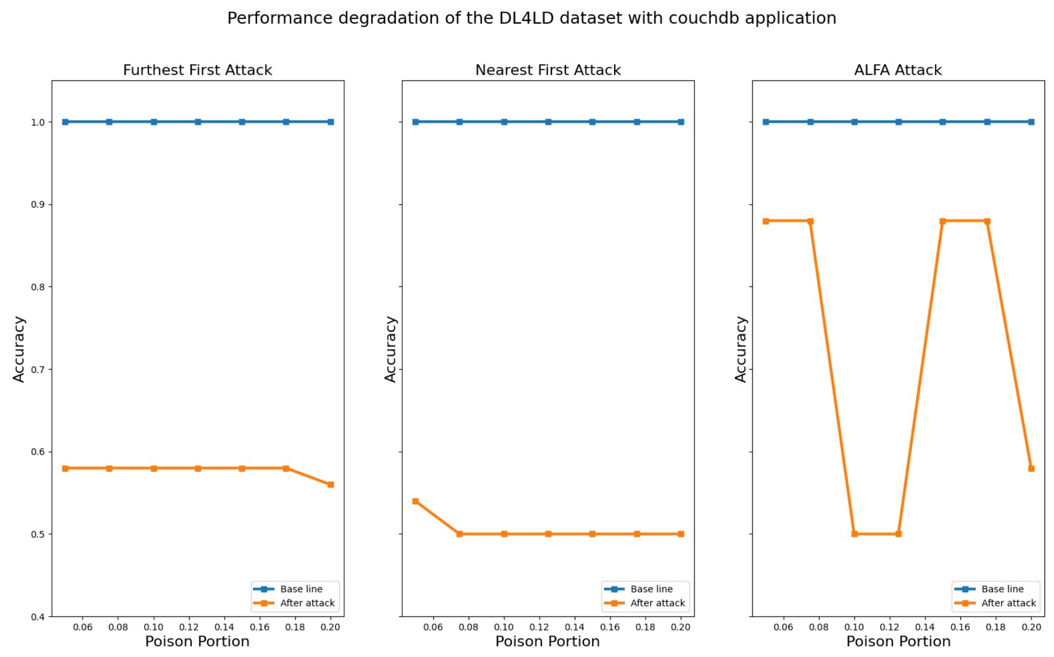
Experimental results – Public dataset

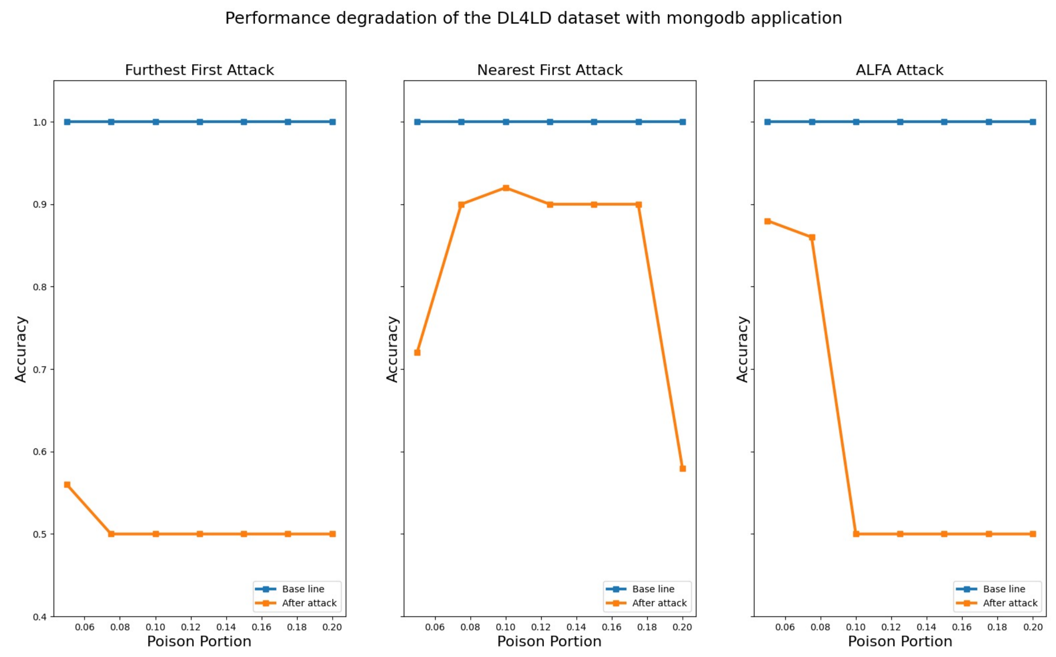
Performance degradation of the ADFA_LD dataset



UNIVERSITY OF AMSTERDAM

Experimental results – real world dataset





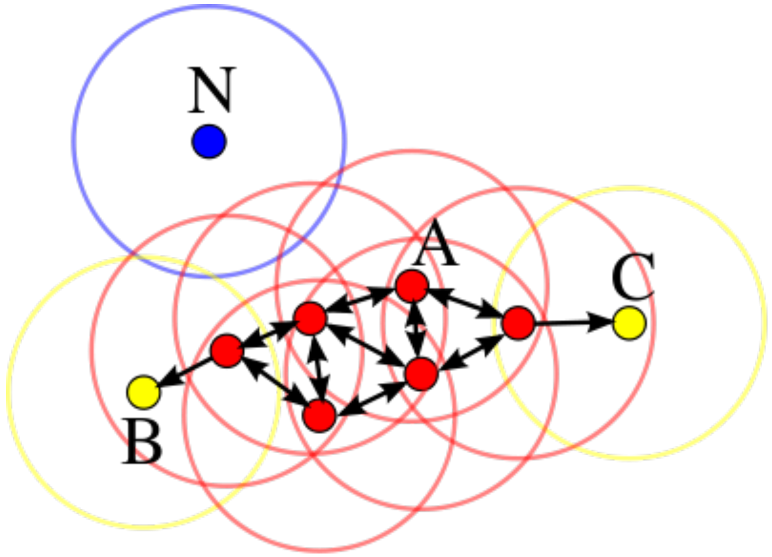
Result analysis

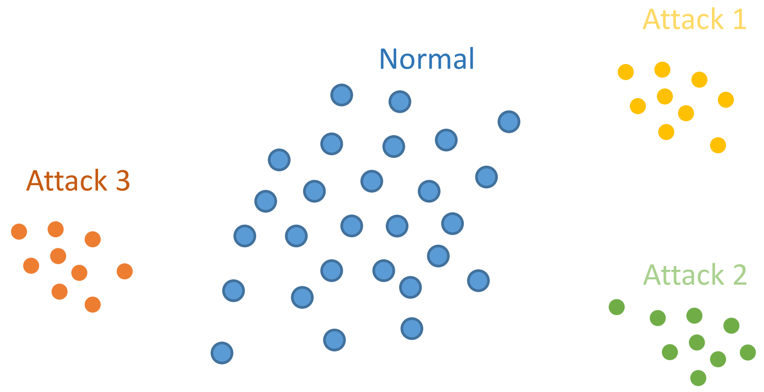
- The nearest first emulates the classification error but it still leads to a relatively high accuracy degradation, especially when the poison portion is large
- The furthest first and ALFA label flipping strategies have similar performance in terms of accuracy degradation
    - ALFA is more computationally expensive
    - More difficult to protect against (Needs further experimental validation)
- It is essential to implement corresponding defense mechanisms for poisoning attacks for IDS

Classic defending mechanisms and limitations

- Outlier detection
    - It requires initial training data
        - We need to know what is the **normal data** in a-prior
        - Not suitable for IDS training if we collect data from crowd
    - The outlier detection in high dimension is difficult
- Adversarial training
    - train the IDS classifier with adversarial samples
        - Algorithm specific
        - Sacrifice the performance of the original classifier
        - Only be resistance to specific attacks, does not work well for unseen attacks

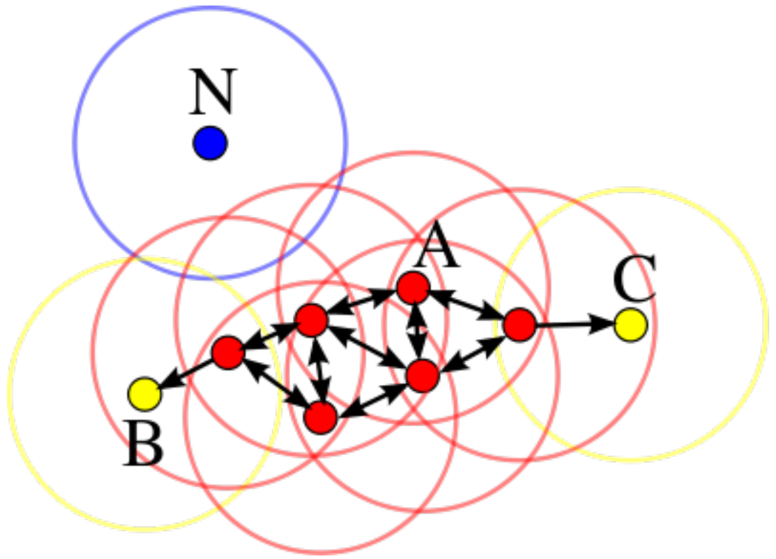
DBSCAN-based defense mechanisms

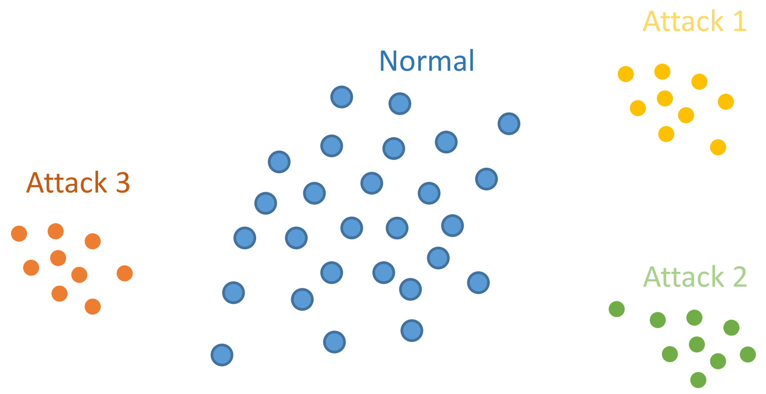




- Clustering is done based on density, not related to shapes
  - Deal with non-linear issues
- Predefine parameters
  - *ε - The maximum distance between two points for one to be considered as in the neighborhood of the other*
  - *MinPts* - the number of points in a neighbourhood for a point to be considered as a **core** point. This includes the point itself
- Output
  - Clustering
  - Labelled points
    - Core points (red): a point that has greater or equal to *MinPts* neighbouring points
    - Border points (yellow): The number of the neighbouring points are smaller than *MinPts,* but is in the neighbourhood of a core point.
    - Outlier (Blue): Neither a core or border point

DBSCAN-based defense mechanisms

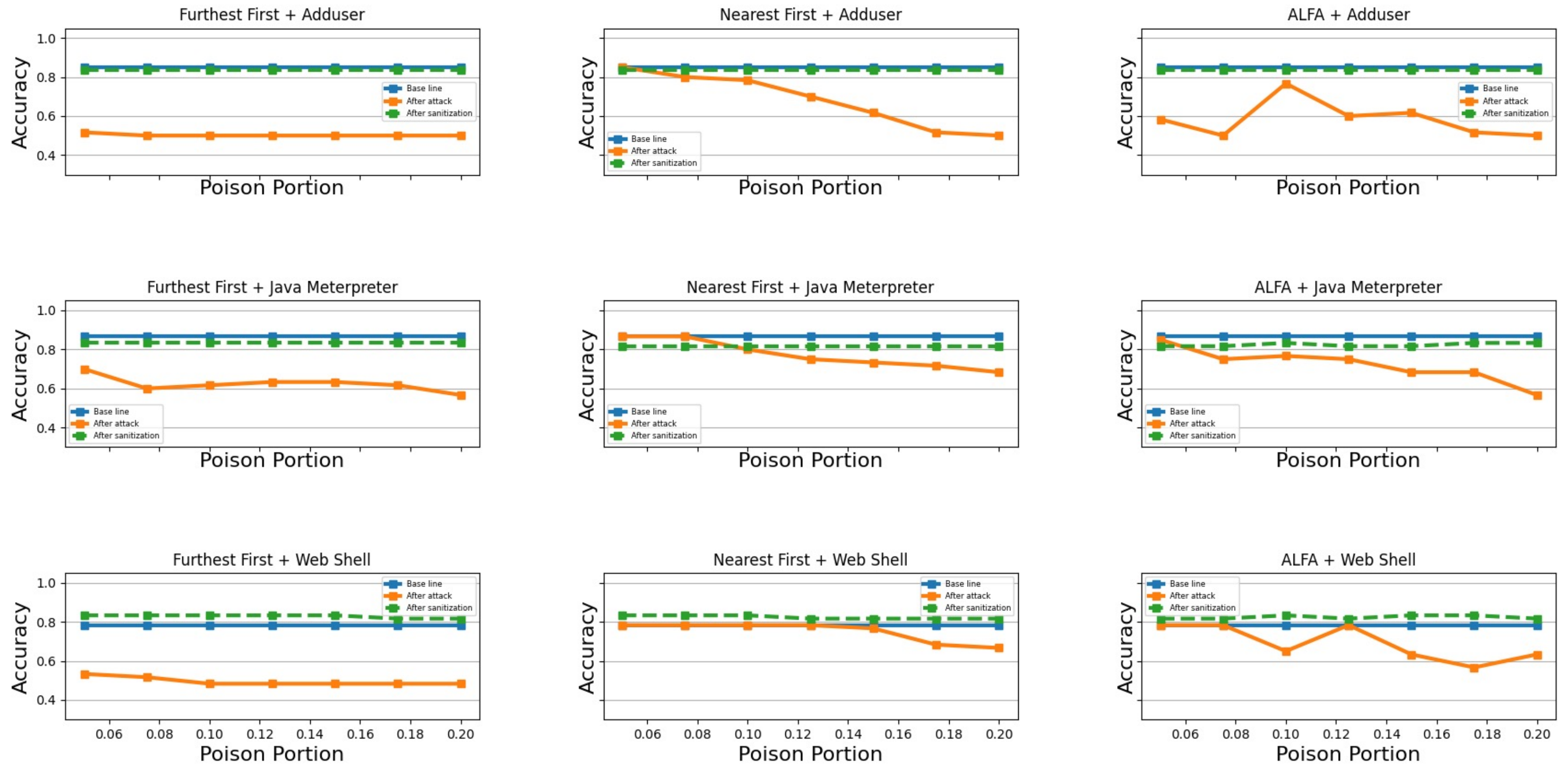




- Clustering is done based on density, not related to shapes
  - Deal with non-linear issues
- Able to detect both cluster numbers and outliers
- Do not require initial normal points, not repeating work for anomaly detection
- Sanitization criteria
  - Remove all outliers
  - Further investigate the data if the number of the cluster is not 1

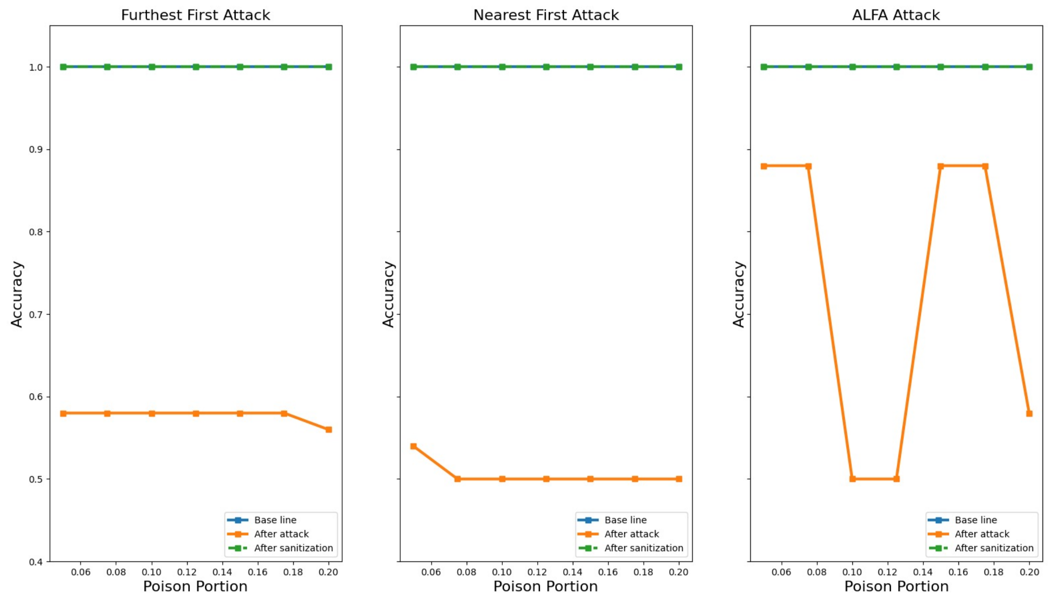
Experimental results - public dataset

Performance improvement of ADFA_LD dataset

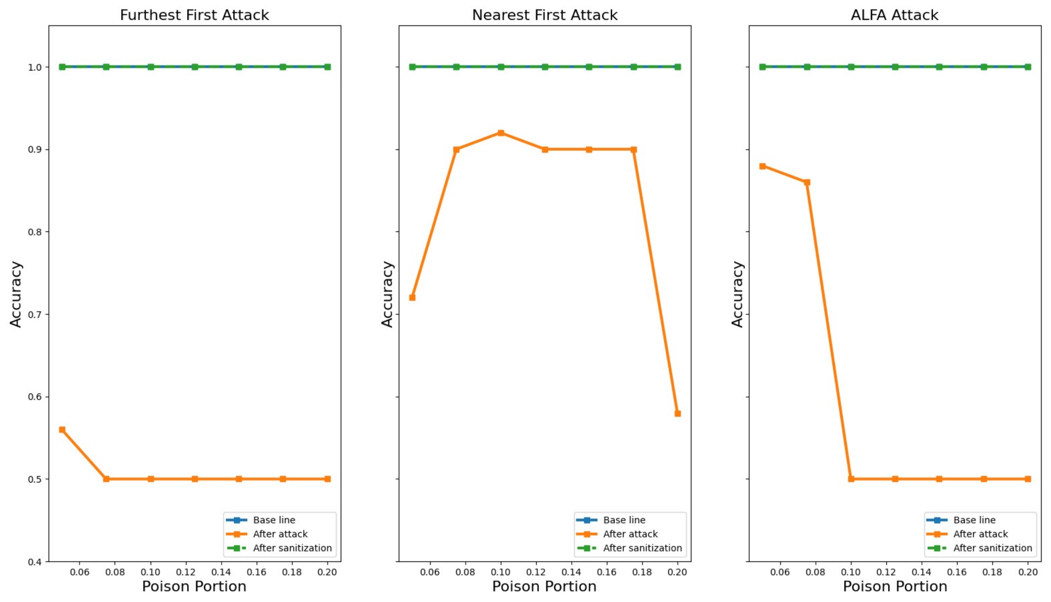


Experimental results - real world dataset

Performance improvement of the DL4LD dataset with couchdb application



Performance improvement of the DL4LD dataset with mongodb application



Ongoing and future works

- Further improve the DBSCAN based defense mechanisms
  - Clear criteria of how to make red flags
  - Different distance measurement metrics
- Write an article and submit it to a conference