[CRiSIS] 2024
19th International Conference on Risks and Security of Internet and Systems
FRANCE / AIX-EN-PROVENCE / TheCamp
From 26 to 28 November 2024

Highlighting the Impact of Packed Executable Alterations with Unsupervised Learning

Alexandre D'Hondt
Charles-Henry Bertrand Van Ouytsel
Axel Legay

UCL Université catholique de Louvain

Outline

1. Introduction
2. Background
3. Methodology
4. Results
5. Conclusion

Outline

1. Introduction
2. Background
3. Methodology
4. Results
5. Conclusion

- Problem statement
- Objectives

Problem statement

Packing

Transformations of executable's syntax preserving its semantic

Challenges

Performance

Static VS dynamic techniques

Significant features

Features robustness

Find significant and robust features

Drivers

- Literature coverage on unsupervised learning

  Very few studies, relatively unexploited yet

- Visualizing the effects of adversarial attacks

  As far as we know, not addressed yet

Objectives

1. Visualize a baseline dataset and its altered versions

   With dimensionality reduction, visualize datasets obtained from a baseline with multiple alterations

2. Describe the datasets and analyze the impact of alterations

   With clustering, describe the datasets obtained from 1. and evaluate the impact

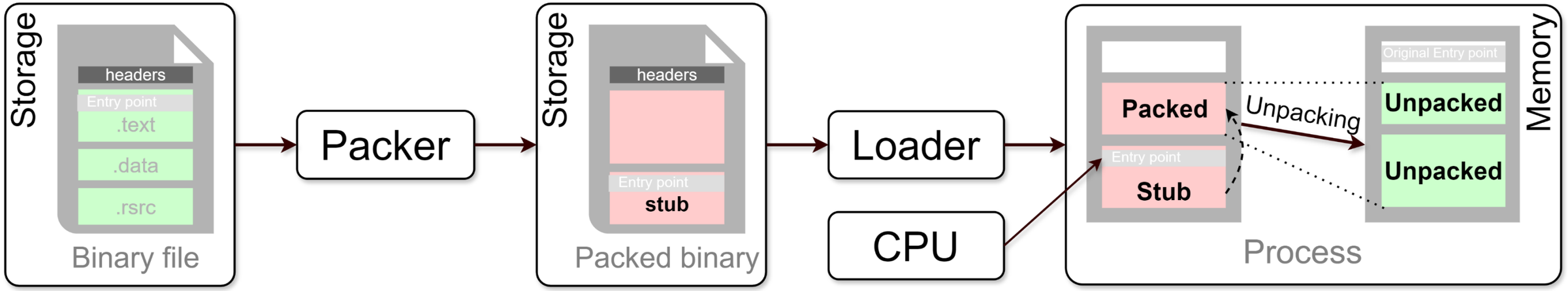
Outline

1. Introduction
2. Background
3. Methodology
4. Results
5. Conclusion

- Packing / Unpacking
- Static VS Dynamic Detection
- Features Set
- Unsupervised Learning
- Adversarial Learning

Packing / Unpacking

Our scope



Categories (combinable) :

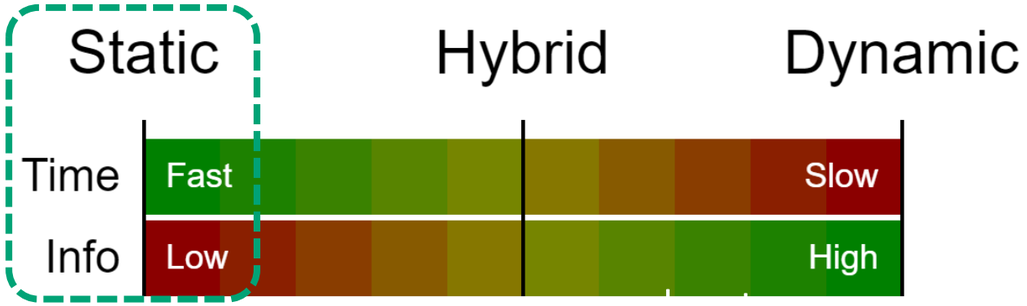
- Bundling
- **Compression**
- Encoding
- **Encryption**
- Mutation
- Protection
- Virtualization

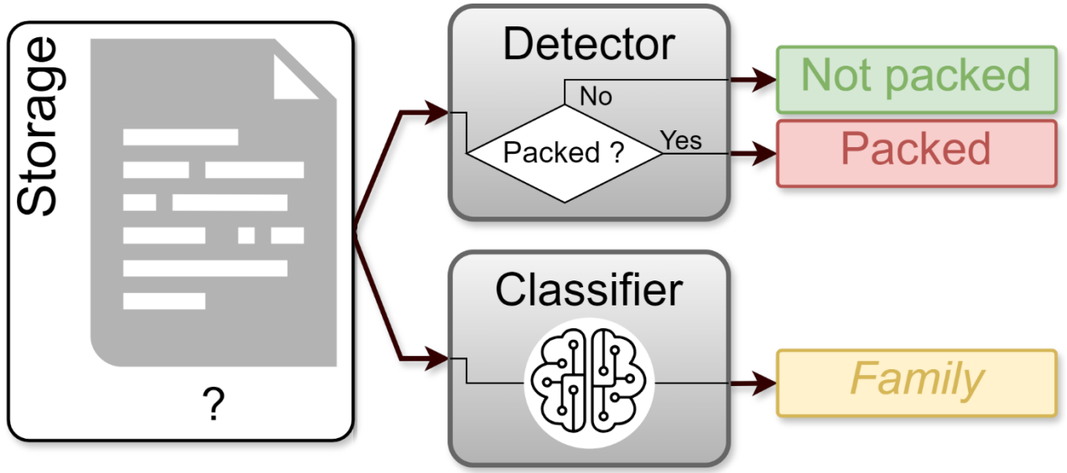
Common usage :

- Size reduction
- SW piracy prevention / License management
- Malware

Static VS Dynamic Detection

Our scope





Static techniques

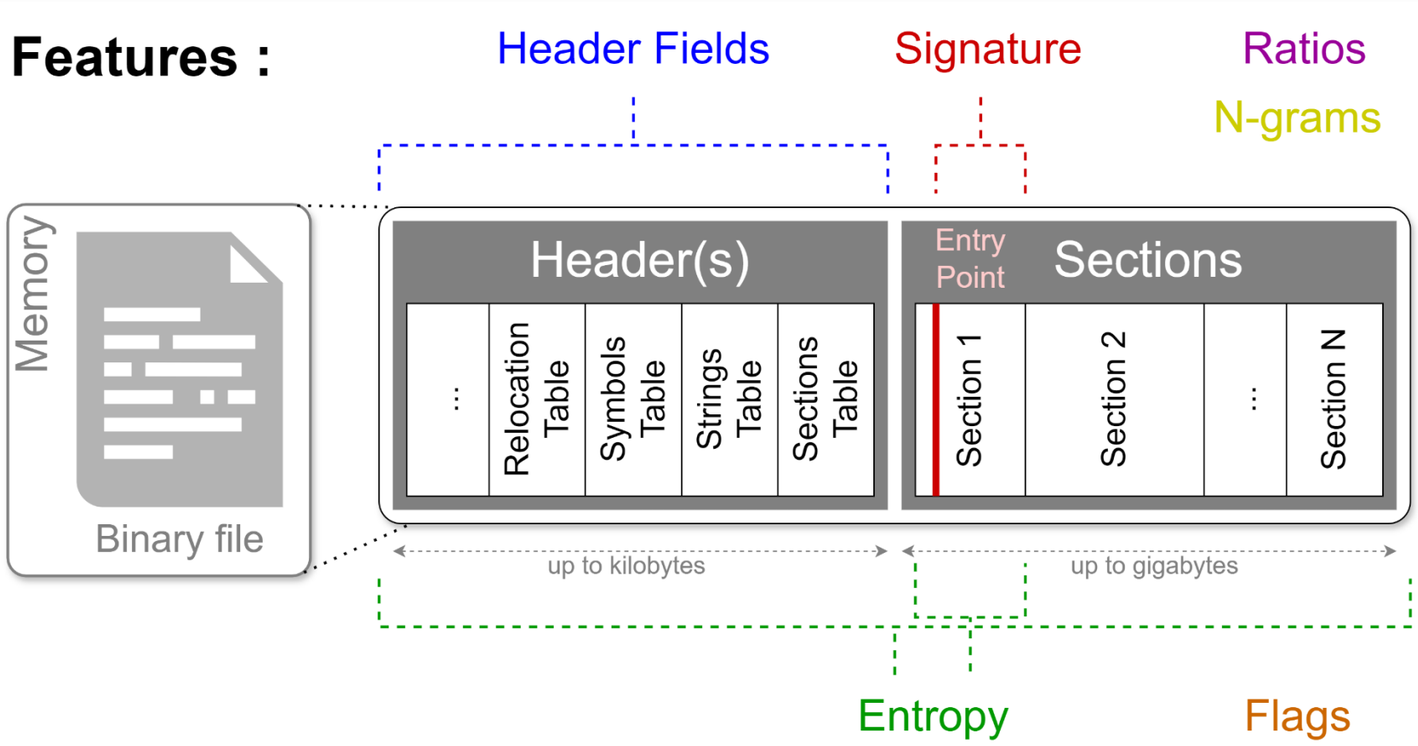
Entropy	Bintropy, RedCurtain
Signatures	PEiD, BE-PUM, SPADE, …
Heuristics	REMINDer
CFG	

Machine Learning

Supervised	PE-Miner, PHAD, McBoost, …
Semi-supervised	
Unsupervised	SE-PAC

Features Set

Our scope



Biondi *et al* [2]

Entropy of entire file
Entropy of code (.text) section
…
Bytes after EP
DLL characteristics
Number of addresses in IAT
…

119 features

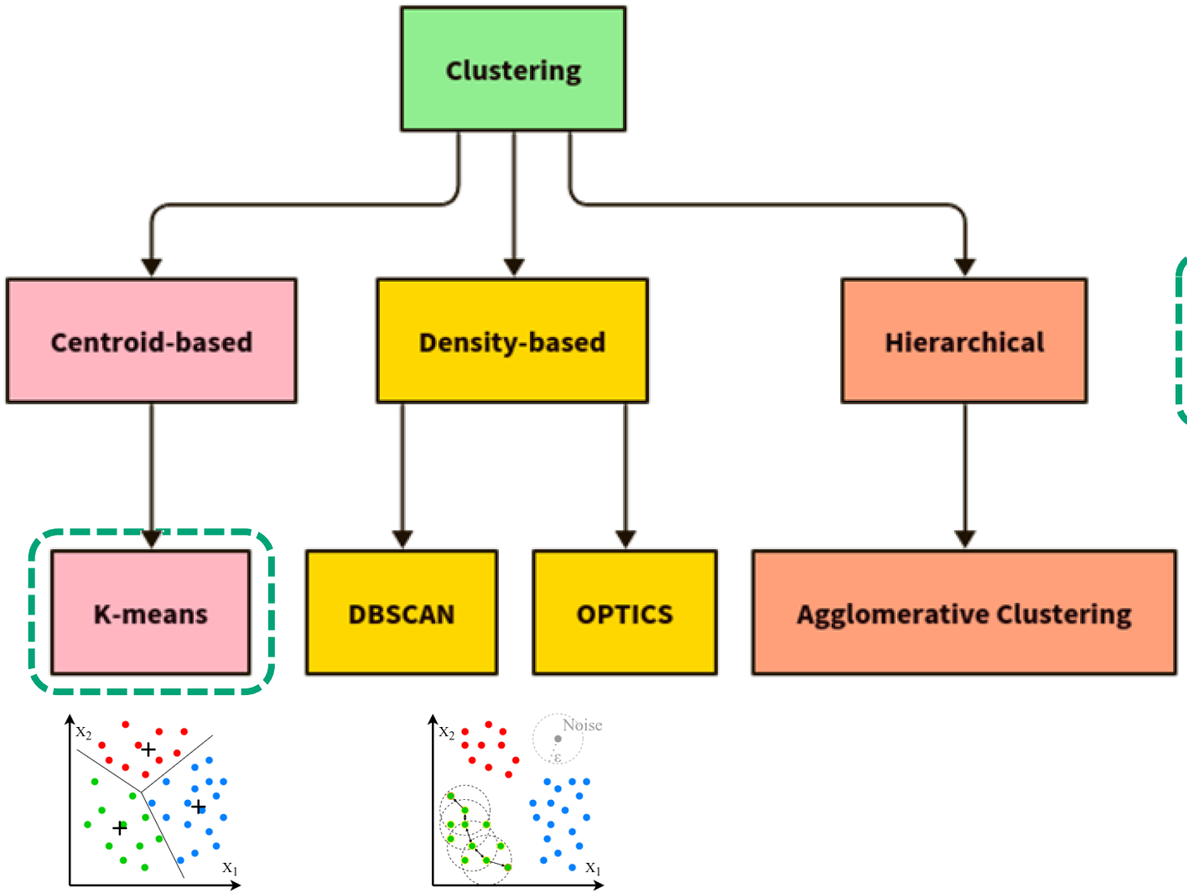
Bertrand Van Ouytsel *et al* [1]

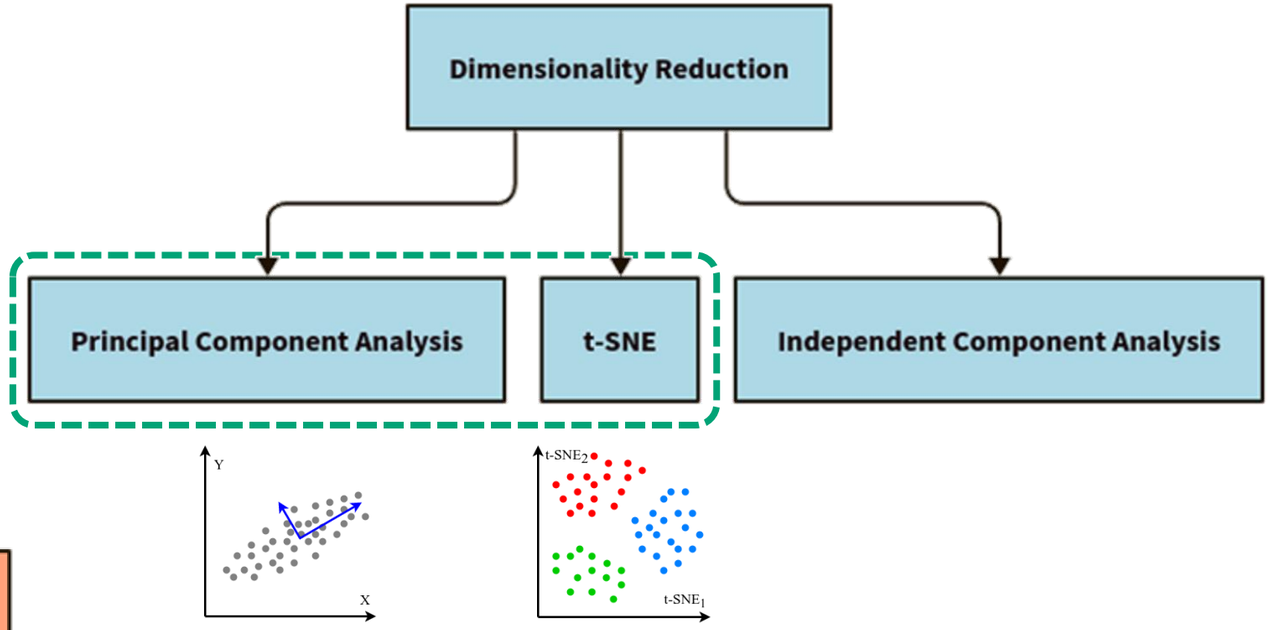
Bytes after EP
Entropy of code (.text) section
Entropy of entire file
Is code section present
Is EP not in code section
Is EP not in standard section
Number of addresses in IAT
Number of functions imported in IDT
Number of RWX sections
Number of standard sections
Number of WX sections
Ratio of malicious API imported

75 significant features

Unsupervised Learning

Our scope





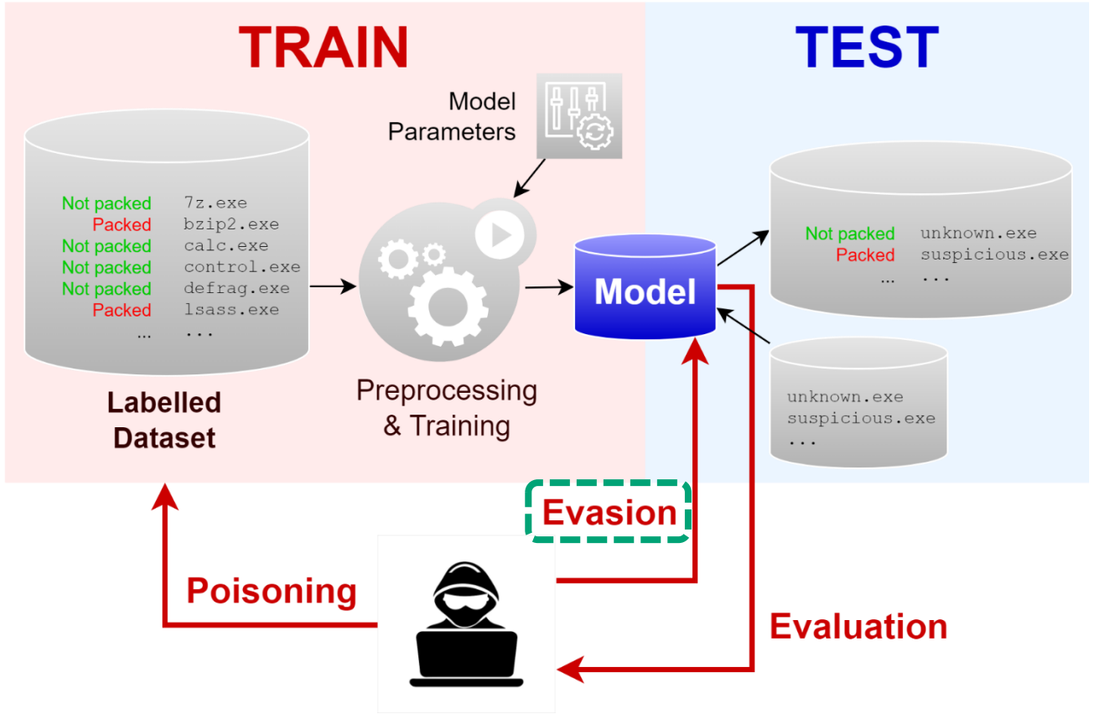
Adversarial Learning

Threat model (Carlini *et al* [3])

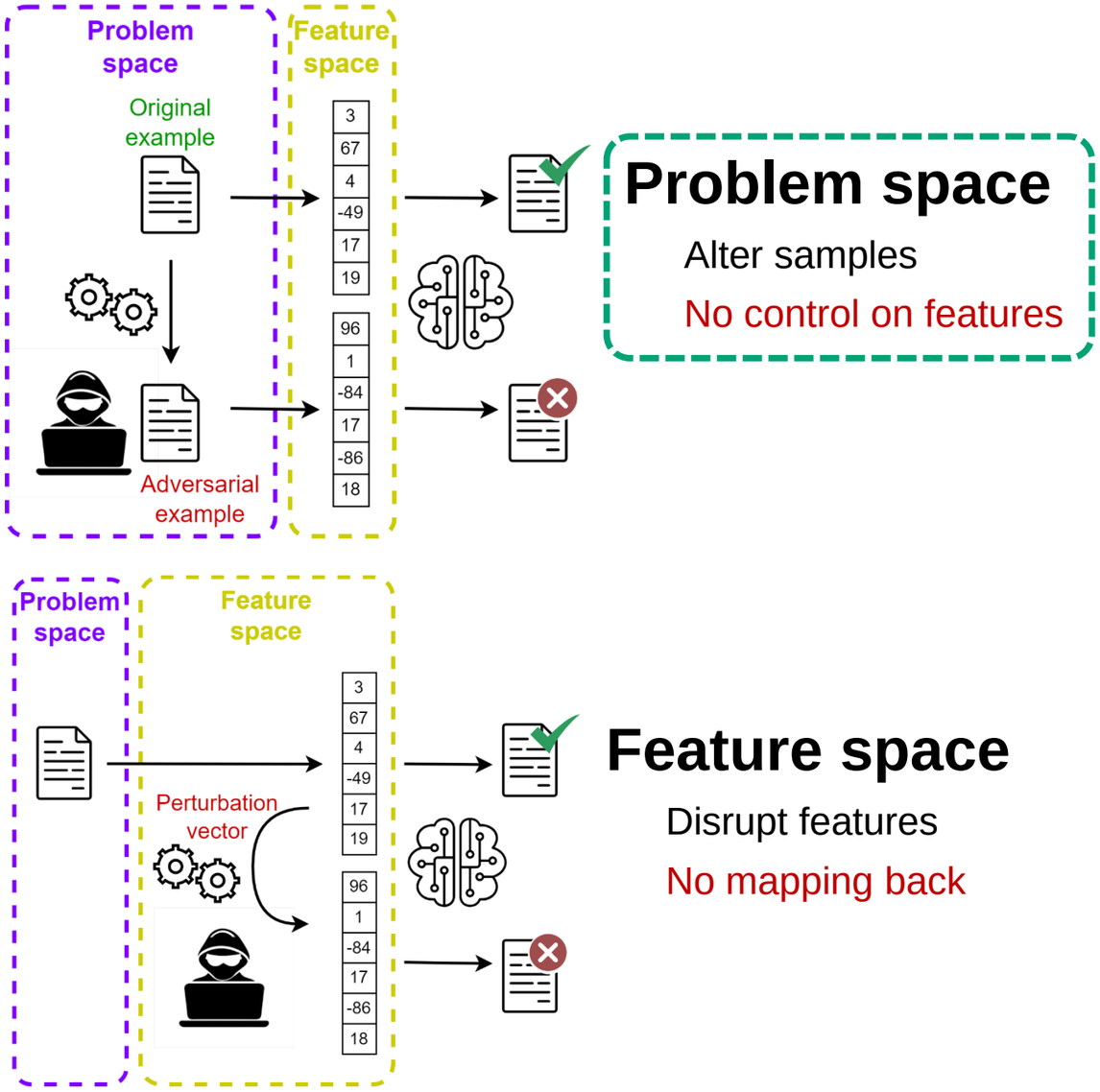
Attack surface Where can the attack take place

Capabilities Ability to act on samples or model

Knowledge White-box VS black-box



Our scope



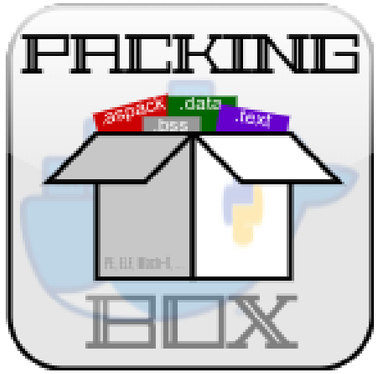
Outline

1. Introduction
2. Background
3. Methodology
4. Results
5. Conclusion

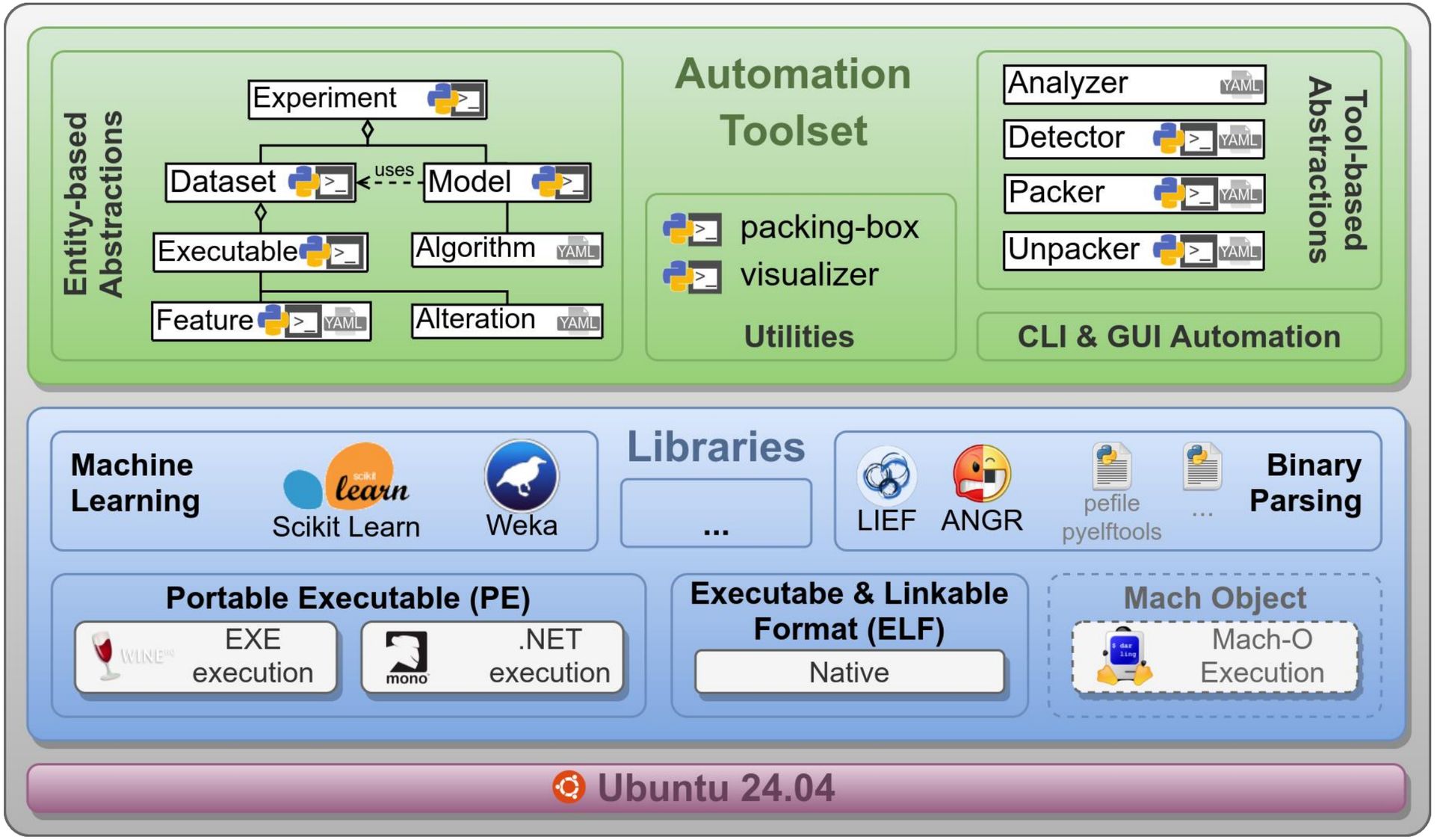
- Packing Box
- Alterations
- Experimental Procedure

Packing Box (1)

D'Hondt *et al* [7]



- Docker
- 3 layers



Packing Box (2)



```
 ____  ____  ____  
```

Alterations

D'Hondt *et al* [6]

ID	Name	Characteristics	Targeted features
A01	**Rename packer sections**	Replace by 6 most common section names first Then by random names from PE format specification	• Number of standard section names • Is code section not present • Entropy of code section
A02	**Move EP to new low-entropy code section**	Create section with standard name and trampoline Fill in with low-entropy padding up to 64 bytes	• Bytes after EP • Is EP not in standard section • Is EP not in code section • Is code section not present • Entropy of code section

Objective : evasion

Experimental Procedure

Dataset of Packed PE

Based on the dataset of Choi *et al* [4]

1 **Make baseline (BL)**

Get 432 not-packed and 123 UPX-packed samples from dataset-packed-pe

Mass-pack 300 samples with UPX

2 **Make altered sets (A0X)**

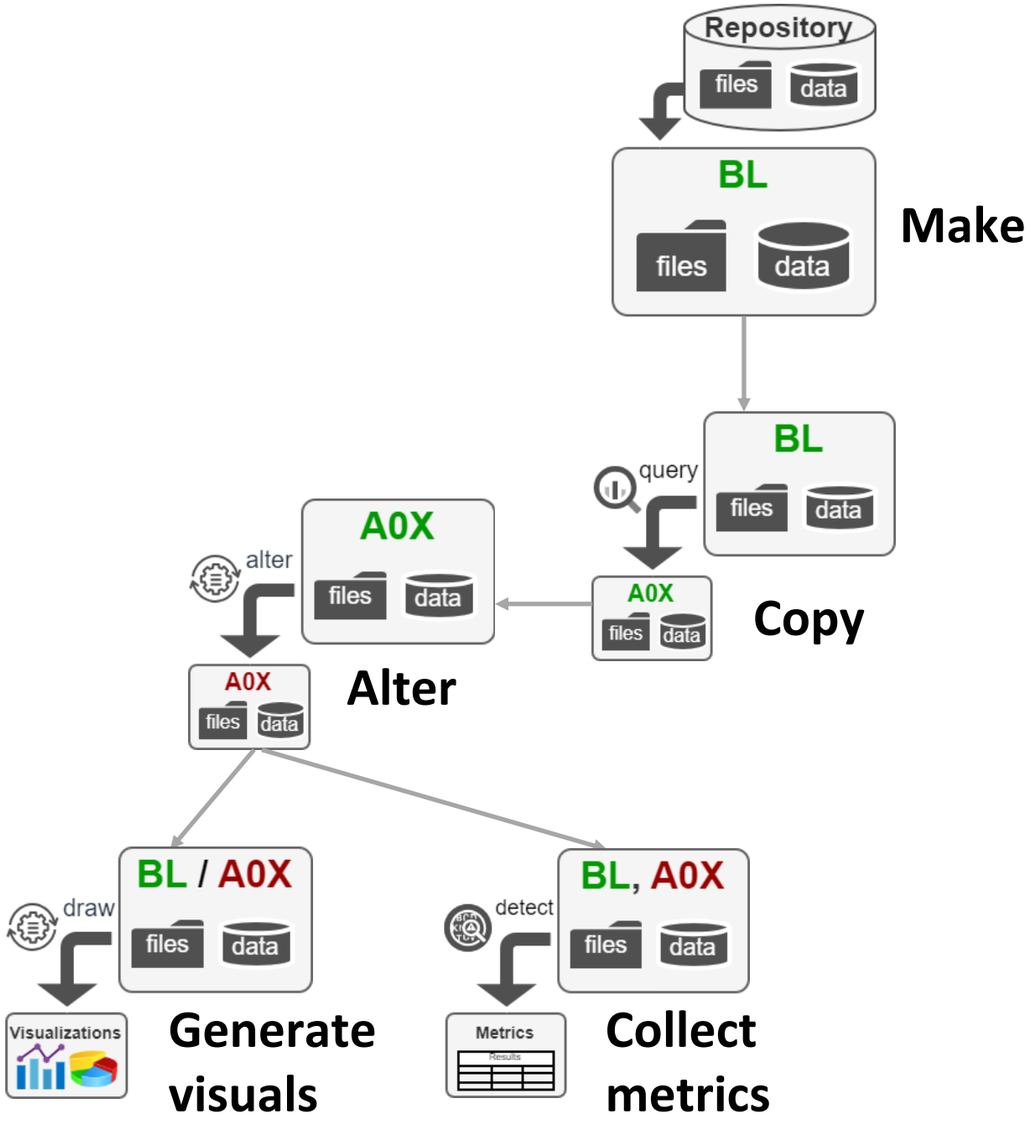
Copy BL

Alter using corresponding alteration

3 **Evaluate impact**

Plot not-packed and packed clusters

Collect metrics and compare to BL

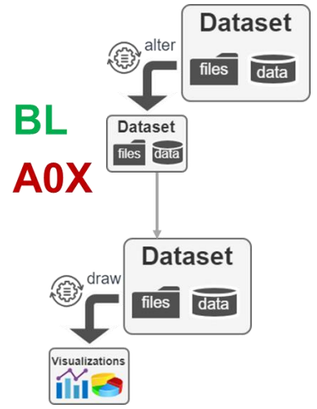


Outline

1. Introduction
2. Background
3. Methodology
4. Results
5. Conclusion

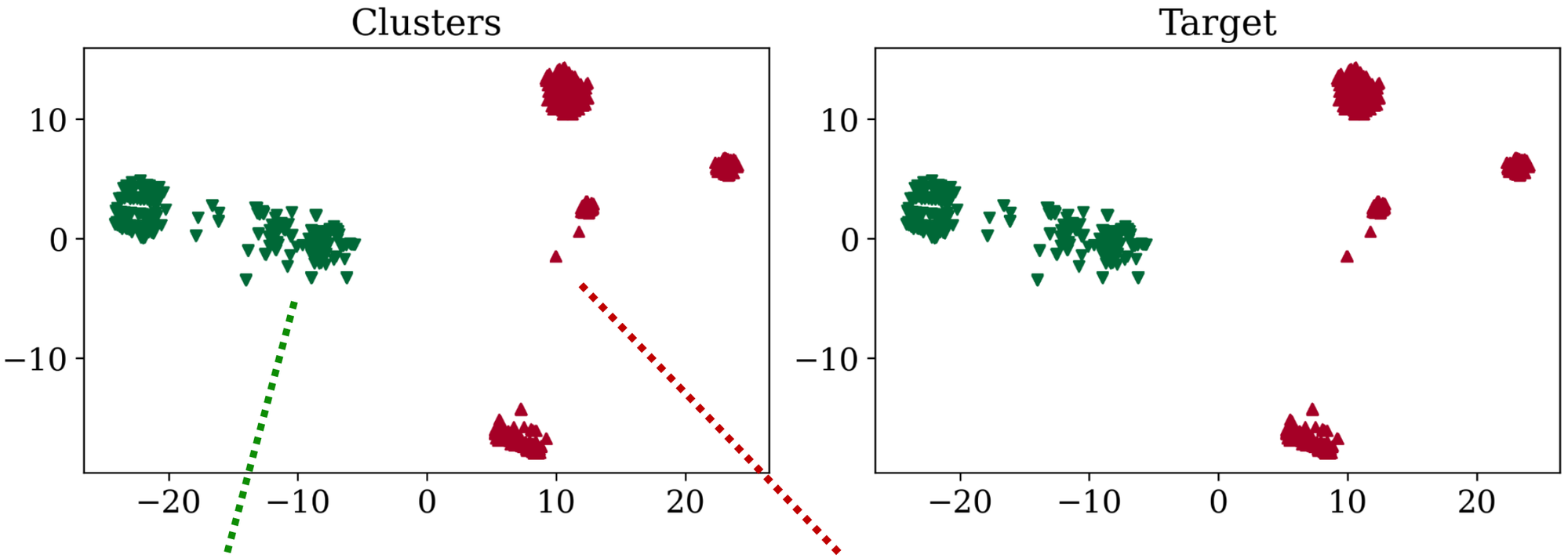
- Baseline Detection
- Impact of A01 (Evasion)
- Impact of A02 (Alarm Flooding)
- Impact on Metrics

Baseline Detection



Kmeans visualization of dataset **BL**

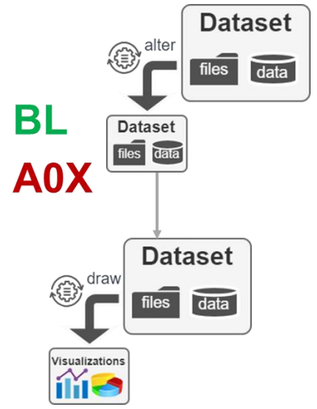
with **PCA** (**20** components) and **t-SNE** (**2** components, **perplexity=100**)



Concentrated cluster

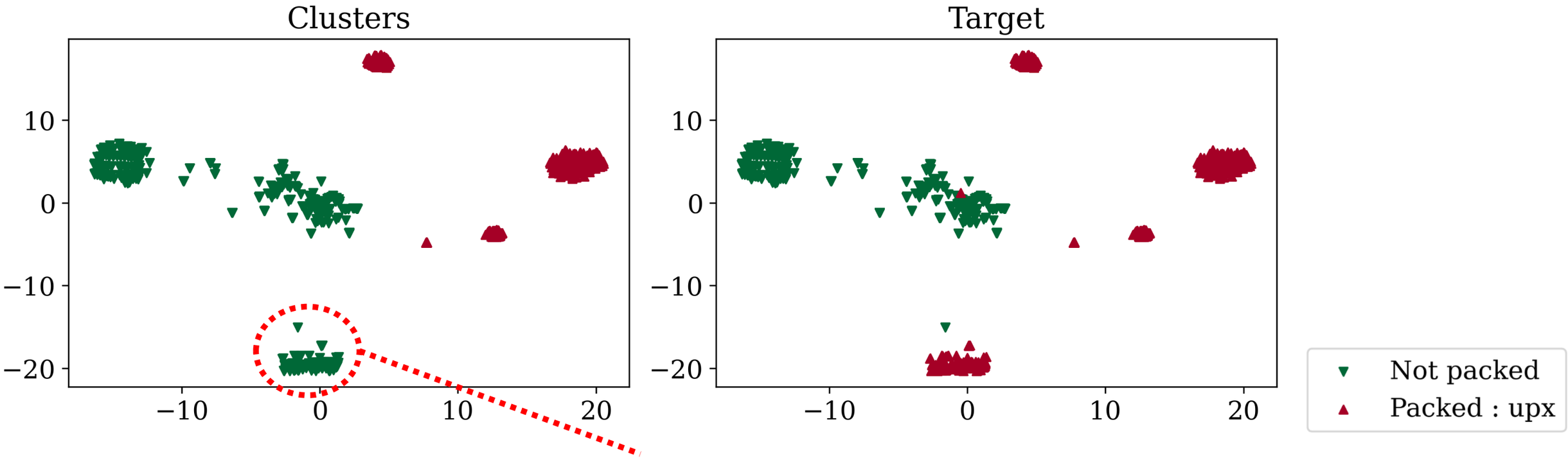
Dispersed cluster

Impact of A01 (Evasion)



Kmeans visualization of dataset **A01**

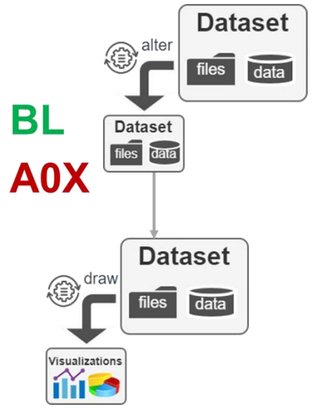
with **PCA** (**20** components) and **t-SNE** (**2** components, **perplexity**=**100**)



A01: rename packer sections

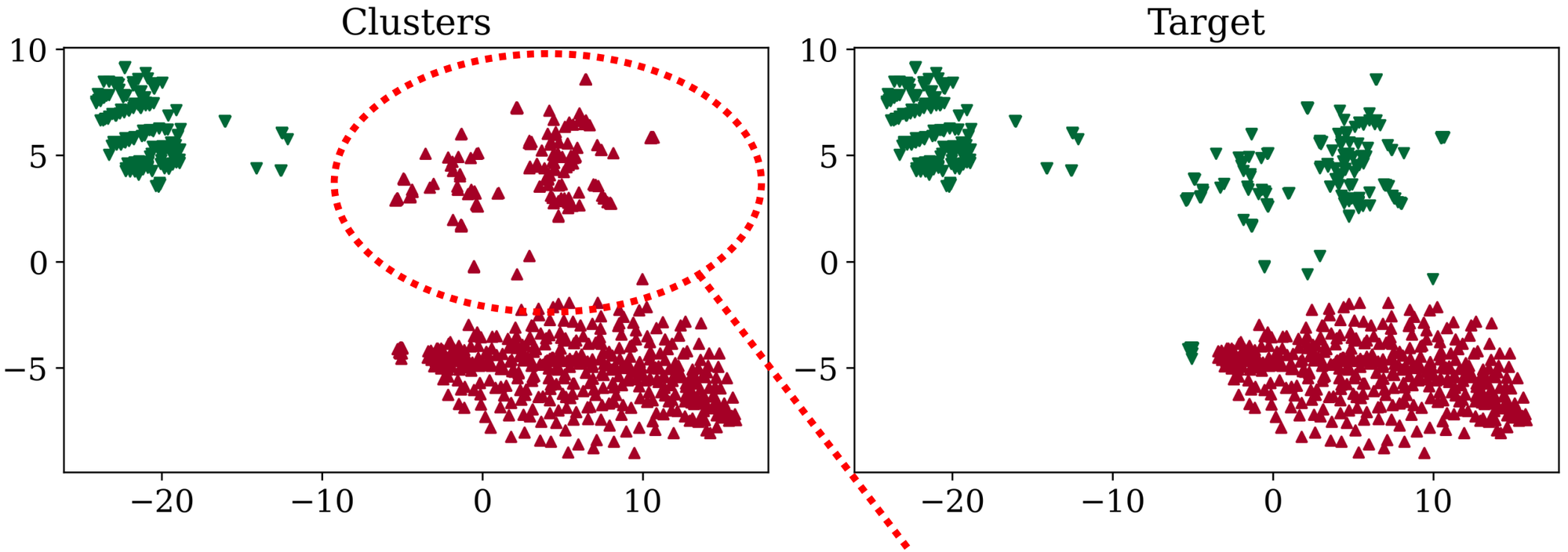
Many samples now detected as not packed

Impact of A02 (Alarm Flooding)



Kmeans visualization of dataset **A02**

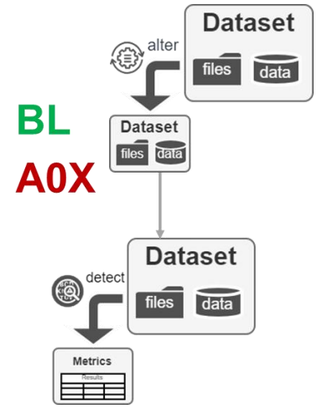
with **PCA** (**20** components) and **t-SNE** (**2** components, **perplexity**=**100**)



A02: move EP to new code section

Many samples now detected as packed

Impact on Metrics

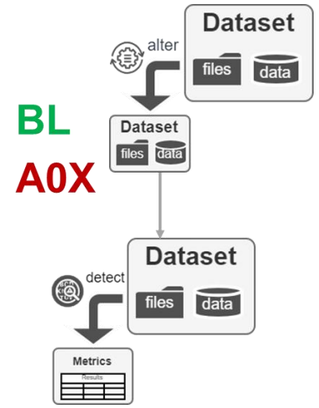


Dataset	Classification metrics			Cluster quality	
	Accuracy	Precision	Recall	Homogeneity	Completeness
BL	99.77%	100.00%	99.54%	97.90%	97.80%
A01	**70.29%**	100.00%	**41.20%**	**24.40%**	**33.10%**
A02	**70.76%**	**62.85%**	100.00%	**25.70%**	**34.10%**
A02-A01	**71.23%**	100.00%	**43.06%**	**25.70%**	**34.10%**

2 false negatives
Clear clusters

A01: rename packer sections
A02: move EP to new code section

Impact on Metrics



Dataset	Classification metrics			Cluster quality	
	Accuracy	Precision	Recall	Homogeneity	Completeness
BL	99.77%	100.00%	99.54%	97.90%	97.80%
A01	**70.29%**	100.00%	**41.20%**	**24.40%**	**33.10%**
A02	**70.76%**	**62.85%**	100.00%	**25.70%**	**34.10%**
A02-A01	**71.23%**	100.00%	**43.06%**	**25.70%**	**34.10%**

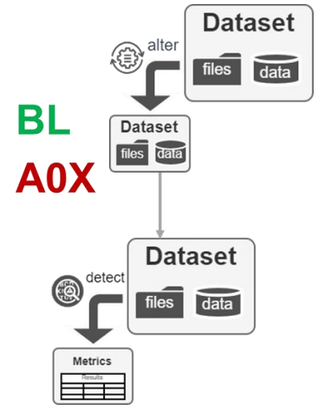
Many false negatives
Bad clusters
Evasion

Similar to A01

A01: rename packer sections

A02: move EP to new code section

Impact on Metrics



Dataset	Classification metrics			Cluster quality	
	Accuracy	Precision	Recall	Homogeneity	Completeness
BL	99.77%	100.00%	99.54%	97.90%	97.80%
A01	**70.29%**	100.00%	**41.20%**	**24.40%**	**33.10%**
A02	**70.76%**	**62.85%**	100.00%	**25.70%**	**34.10%**
A02-A01	**71.23%**	100.00%	**43.06%**	**25.70%**	**34.10%**

Many false positives
Bad clusters
Alarm flooding

A01: rename packer sections
A02: move EP to new code section

Outline

1. Introduction
2. Background
3. Framework
4. Experiments
5. Conclusion

- Objectives
- Future works

Objectives

1. Generated visualizations for a baseline dataset and its altered versions

   With dimensionality reduction, we visualized multiple versions of a baseline dataset obtained with multiple alterations applied

2. Described clusters and analyzed the impact of alterations

   With clustering, we described the datasets obtained from 1. by computing the clusters and evaluating classification metrics to assess the impact

Future works

- Building high-quality datasets (WIP)

  Improving dataset generation capabilities and making larger datasets

- Adversarial learning (cont'd)

  Breaking ML-based static detectors

- Packer categories

  Finding the best (performant and robust) features set per packer category

[CRiSIS] 2024

19th International Conference on Risks and Security of Internet and Systems

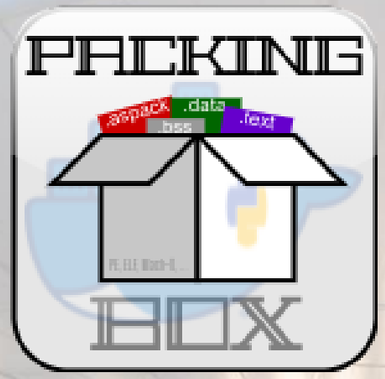
FRANCE / AIX-EN-PROVENCE / TheCamp

From 26 to 28 November 2024

Questions ?



Awesome list gathering our whole bibliography and many other references to documentation, tools, etc.





Ready-to-use dataset of packed and not-packed ELF files



Ready-to-use dataset of packed and not-packed PE files (enriched version of Choi's dataset)



Entropy-based tool inspired from the study of Lyda et al. in 2007



Heuristic-based tool inspired from the study of Han et al. in 2009



Operationalized fork of https://github.com/cylance/PyPackerDetect



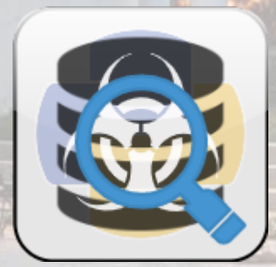
Python fork of the popular tool, PEiD



Attack tool for altering packed samples so that they evade static packing detection



Custom exchange format for datasets (supports conversion to ARFF, CSV, Packing-Box dataset)



Library for getting samples from multiple malware databases

[1] Bertrand Van Ouytsel et al, *Analysis of machine learning approaches to packing detection* (2023)

[2] Biondi et al, *Effective, efficient, and robust packing detection and classification* (2019)

[3] Carlini et al, *On evaluating adversarial robustness* (2019)

[4] Choi et al, *All-in-one framework for detection, unpacking, and verification for malware analysis* (2019)

[5] Demetrio et al, *Practical attacks on machine learning: A case study on adversarial windows malware* (2022)

[6] D'Hondt et al, *Evading packing detection: Breaking heuristic-based static detectors* (2024)

[7] D'Hondt et al, *Experimental toolkit for manipulating executable packing* (2023)

[8] Kolosnjaji et al, Adversarial malware binaries: Evading deep learning for malware detection in executables (2018)

[9] Ling et al, Adversarial attacks against windows PE malware detection: A survey of the state-of-the-art (2021)

[10] Noureddine et al, SE-PAC: A self-evolving packer classifier against rapid packers evolution (2021)