
black hat®
ARSENAL 2022
DECEMBER 7-8
EXCEL LONDON / UK
#BHEU @BlackHatEvents

Packing Box

Playing with Executable Packing

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

Outline

1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

Outline

1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

- Problem statement
- Objectives

Problem statement (1)

Packing =

- Set of transformations
- On binary file
- That preserves the original working at runtime

→ Large coverage in scientific literature

→ Often employed with malware

Problem statement (2)

Detection challenges :

- Diversity of packers
- Static analysis
- Reproducibility of related works
- Accurate ground truths
- Best features
- Best machine learning algorithms

- Currently no framework for the study of static detection
- Limited scope* of related works
- Poor repeatability of experiments

* typically PE malware

Objectives

1. Build a framework for repeatable experiments
2. Generate own (accurrate) datasets
3. Automate machine learning (ML) pipelines

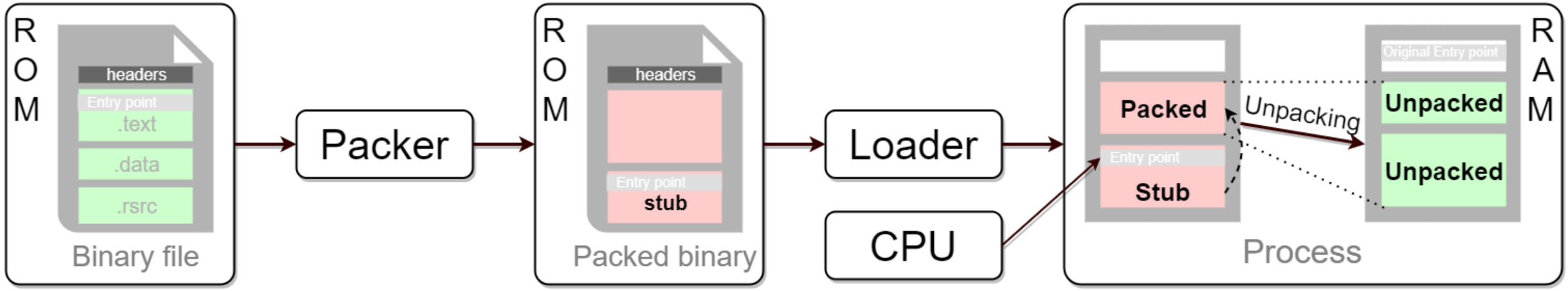
+ Support for more than simply PE

Outline

1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

- Packing / unpacking
- Detection (static VS dynamic)
- Static features of binaries
- Learning pipeline

Packing / unpacking



Taxonomy of transformations :

- Compression
- Encryption
- Protection
- Bundling
- Mutation
- Virtualization

Common usage :

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

Detection (static VS dynamic)

Static (no execution) :

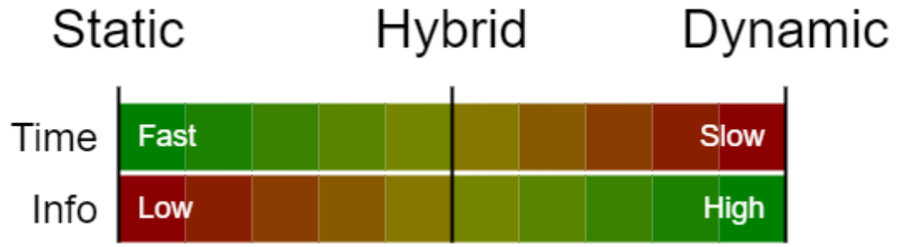
- Entropy threshold
- Pattern matching
- Signatures
- Disassembly
- …

VS

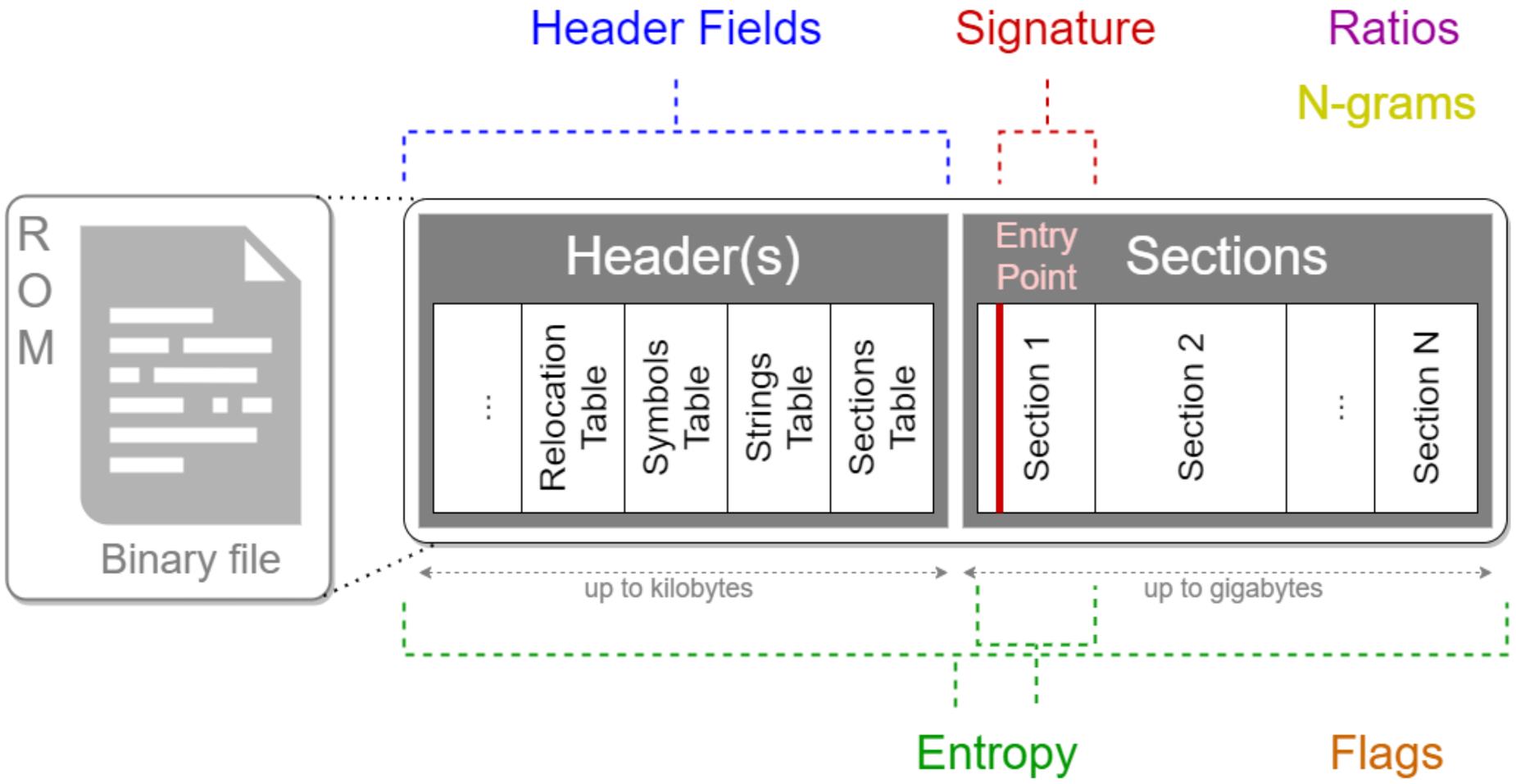
Dynamic (execution required) :

- Debugging
- Sandboxing
- Behavioral analysis
- Memory monitoring
- …

Time-information tradeoff :

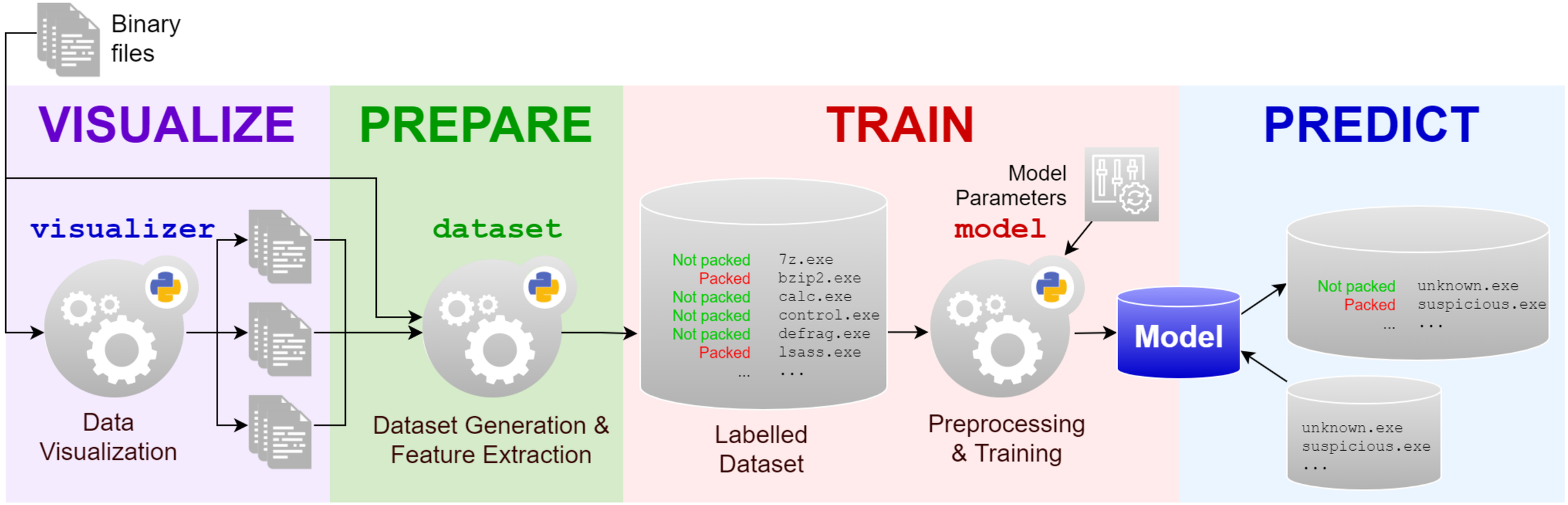


Static features of binaries



2. Background

Learning pipeline



Outline

1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

- Requirements
- Design principles
- Architecture
- Implementation
- Capabilities
- Related projects

Requirements

A. Scope : PE, ELF(, Mach-O)

B. Toolkit for supporting the complete ML pipeline

C. Easy integration of new items (detectors, packers, etc.)

D. Easy-to-manipulate Dataset structure

E. Easy tuning of the ML pipeline (algorithms, features, etc.)

Design principles

I. Terminal-based isolated full-featured container

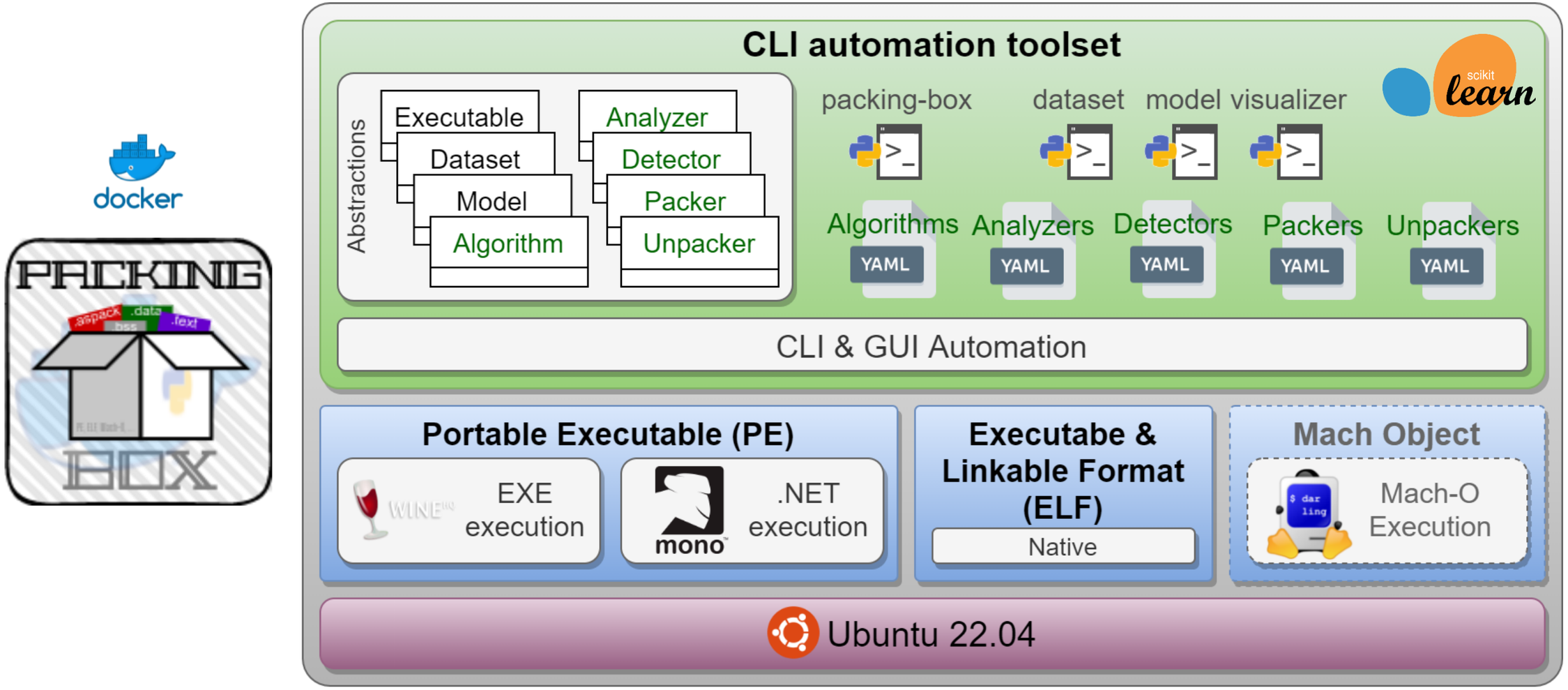
II. Automation layer (i.e. packers for mass-packing)

III. Configuration based on YAML

IV. Datasets & models management

V. Feature integration and combination

Architecture



Implementation

- YAML configurations
- Administration tool
- Toolkit for ML pipeline

```yaml
hXOR-Packer:
  categories:
    - compressor
    - cryptor
  description: A PE (Portable Executable) packer with Huffman Compression and Xor encryption.
  exclude:
    - DLL: PE32\+? executable \(DLL\)
  failure:
    hash: 1cc82b2855df0c2fb3c30dffe9f02512aebd79d7df8b00ce69aa0790aa60297e
  formats:
    - PE
  install:
    - wget: https://github.com/rurararura/hXOR-Packer:latest{hXOR-Packer}
    - unzip: $OPT
    - chmod: packer.exe
    - lwine: packer.exe
  references:
    - https://github.com/rurararura/hXOR-Packer
  source: https://github.com/rurararura/hXOR-Packer
  status: ok
  steps:
    - hxor-packer {{executable}} {{executable}} -ce {{key[randint]}}
  variants:
    hXOR-Compressor:
      categories:
        - compressor
      description: A PE (Portable Executable) packer with Huffman Compression.
      steps:
        - hxor-packer {{executable}} {{executable}} -c
```

Example configuration for hXOR-Packer

Capabilities

- Ground truth generation
- Dataset manipulation & sharing
- Model training & analysis
- Data visualization
- Performance analysis of state-of-the-art static detectors

Related projects

https://github.com/packing-box

- Resources

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 from the enriched version of this repository

- Tools

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

Operationalized fork of this un-maintained tool

Python fork of the popular tool, PEiD

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

Outline

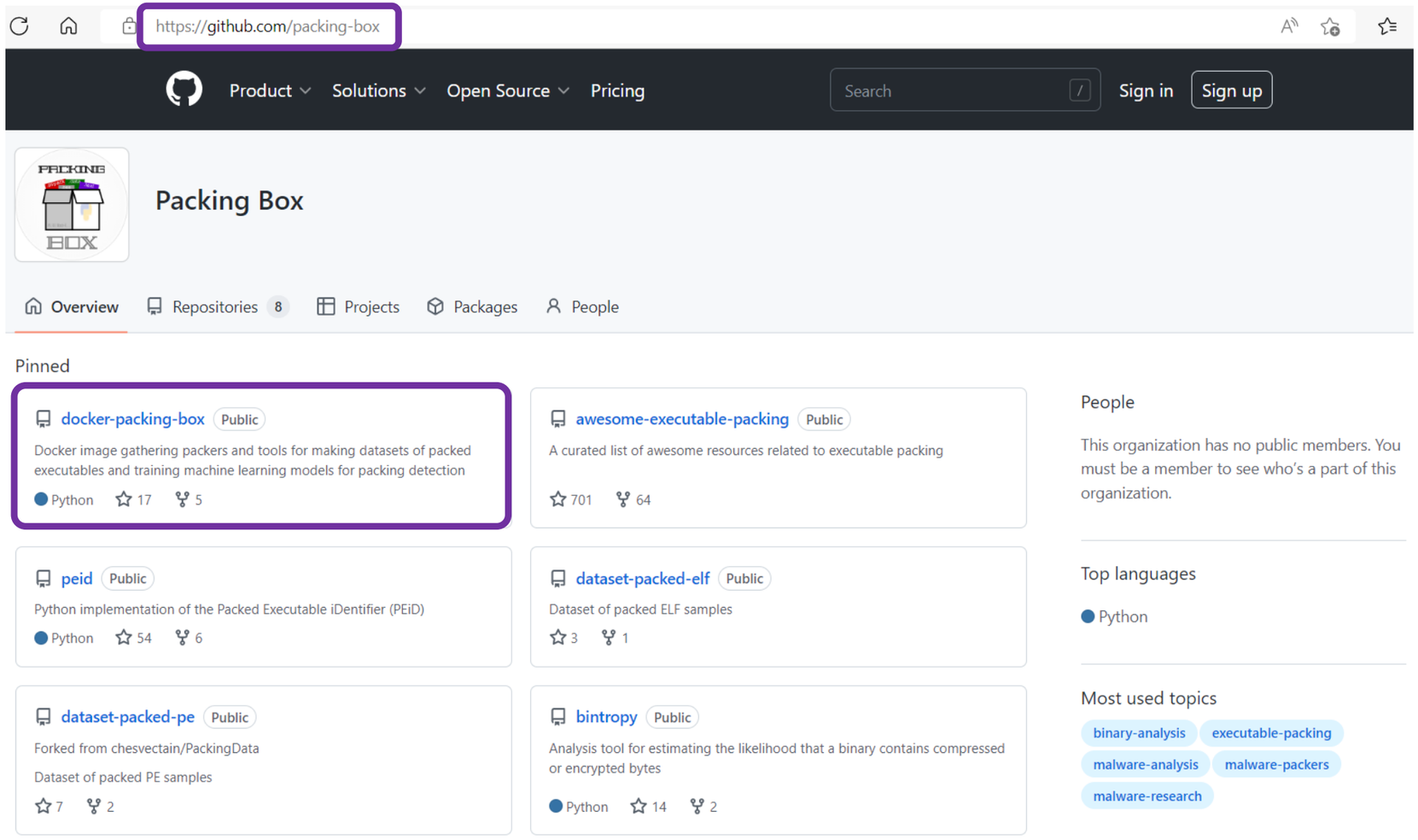
1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

- Getting started
- Getting help
- Installing items
- Playing with datasets
- Playing with models

4. Quick Start

Getting started (1)

- Starting point :
  1. Open terminal
  2. Clone the repo



Please report any issue at : https://github.com/packing-box/docker-packing-box/issues

4. Quick Start

Getting started (2)

- Start : CLI open, Docker installed, repository cloned to **`[path_to_repo]`**
- Actions :

```
$ docker build -t dhondta/packing-box .
$ docker run -it -h packing-box -v `pwd`:/mnt/share dhondta/packing-box
```

- End state :
  - ✓ Packing Box CLI open
  - ✓ Startup message displaying the available items

Getting help

- Start : Packing Box CLI open
- Actions :

```
$ ?
$ ? -k packers
$ ? -i upx
```

- End state :
  - ✓ Help messages showing the status of in-scope items

Installing items

- Start : Packing Box CLI open
- Actions :

```
$ vim src/conf/packers.yml
$ packing-box -v setup packer upx
$ packing-box test packer upx
```

- End state :
  - ✓ UPX installed (verbose mode)
  - ✓ UPX tested on included samples

Playing with datasets

- Start : Packing Box CLI open, UPX installed
- Actions :

```
$ dataset make test-pe-upx -n 100 -f PE -p upx
$ dataset show test-pe-upx
$ dataset convert test-pe-upx
$ dataset edit test-pe-upx
```

- End state :
  - ✓ Dataset **`test-pe-upx`** of 100 UPX-packed PE samples
  - ✓ Dataset **`test-pe-upx`** converted to a fileless dataset (features computed)

4. Quick Start

Playing with models

- Start : Packing Box CLI open, **`test-pe-upx`** dataset created
- Actions :

```
$ model train test-pe-upx -a rf
$ model list
$ model show [model_name]
$ model test [model_name] test-pe-upx
```

- End state :
  - ✓ Model trained on the basis of **`test-upx`** (name generated automatically)
  - ✓ Model tested on the reference dataset for performance metrics

Outline

1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

- Model for PE packers
- Visualization of files & models
- Evaluation of detectors

Model for PE packers

- Start : Packing Box CLI open, **dataset-packer-pe** cloned
- Actions :

```
$ dataset update test-pe -s dataset-packer-pe/not-packed -l dataset-packer-pe/labels.json
$ for N in UPX RLPack TELock; do dataset update test-pe \
    -s dataset-packer-pe/packed/$N -l dataset-packer-pe/labels.json; done
$ model train test-pe -a dt
$ model test [model_name] test-pe-upx
```

- End state :
  - ✓ Dataset **test-pe** of mixed-packed PE samples based on **dataset-packer-pe**
  - ✓ Model trained on the basis of **test-pe**
  - ✓ Model tested on **test-pe-upx** for performance metrics

Visualization of files & models

- Start : Packing Box CLI open, `dataset-packer-pe` cloned, **`test-upx`** created
- Actions :

```
$ visualizer plot "PsExec.exe$" dataset-packed-pe -s -l not-packed -l MEW -l UPX
$ dataset plot test-upx byte_0_after_ep byte_1_after_ep --multiclass
$ model visualize [model_name]
```

- End state :
  - ✓ Visualization of `PsExec.exe` with multiple packed versions at scale
  - ✓ Visualization of bytes 0 and 1 after EP for a dataset with a mix of packers
  - ✓ Visualization of the given trained model

Evaluation of detectors

- Start : Packing Box CLI open
- Actions :

```
$ for N in {1..6}; do dataset make test-upx-variants -n 2 -f PE -p upx-3.9$N
    --pack-all; done
$ detector test-upx-variants -f -d die
$ detector test-upx-variants -f
```

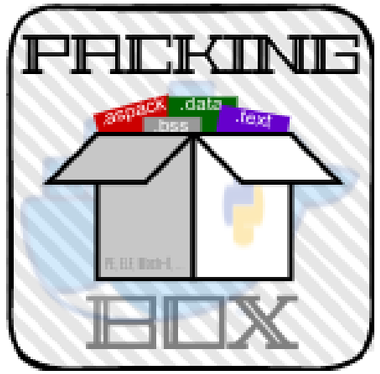
- End state :
  - ✓ Dataset **`test-upx-variants`** with mixed samples (PE, ELF, …) all packed with multiple versions of UPX
  - ✓ Detector DIE then all the voting detectors (forming a superdetector) applied to the new dataset

Outline

1. Introduction
2. Background
3. Framework
4. Quick Start
5. Advanced Use
6. Conclusion

- Contribution
- Future works

Contribution



Toolkit for dataset manipulation and model training

- Support for many packers currently (incl. UPX, Ward, Yoda's Crypter, Amber, Eronana's Packer, …)
- Support for many detectors (incl. DIE, Bintropy, PEiD, PyPackerDetect, PEFrame, PEPack)
- Complete toolkit for the whole machine learning pipeline

Future works

- Research of best features
- Train models per category according to our taxonomy
- Adversarial learning based on altered datasets
- Semi-supervised and unsupervised learning