Static, Dynamic, Symbolic: Exploring Binary Representations for machine learning classification

Charles-Henry Bertrand Van Ouytsel

DefMal Days 2025

Who am I ?

Charles-Henry Bertrand Van Ouystel, Post-Doc @UCLouvain

Interests:

Machine Learning for CyberSec, CyberSec for Machine Learning, Program analysis

CyberExcellence program manager – Research program in Belgium gathering universities and research centers from Wallonia

Table of content

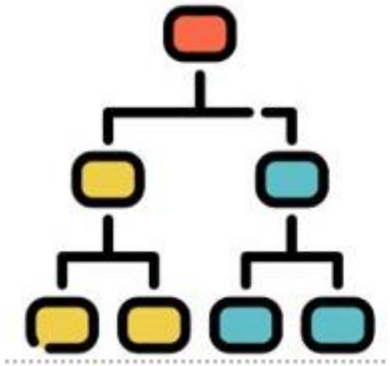
- Comparison tabular vs Graph based feature
- SEMA – Symbolic Execution for Malware Analysis
- Diving into EMBER features and academic datasets
- PackingBox – Toolkit for static detection of executable packing

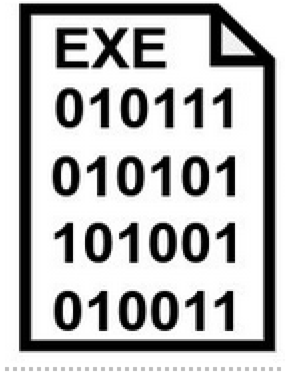
Context and objectives

Different machine learning models have been proposed to help analysts to detect malware variant but also new malware families

Unclear which representation to use ?

Paper accepted at WoRMA 2025 – "A Unified Comparison of Tabular and Graph based Feature Representations in Machine Learning for Malware Detection" Samy *Bettaieb (UCLouvain), Serena Lucca (UCLouvain), Charles-Henry Bertrand Van Ouytsel (UCLouvain), Axel Legay (Nexova), Etienne Rivière (UCLouvain)*





Static vs Dynamic, Tabular vs Graphs

Features could be extracted **statically** or **dynamically**

Recent litterature shows static features are more efficient for known families while dynamic features show a better ability to generalize

From these features, **tabular** and **graph** based representation have been proposed but rarely compared !

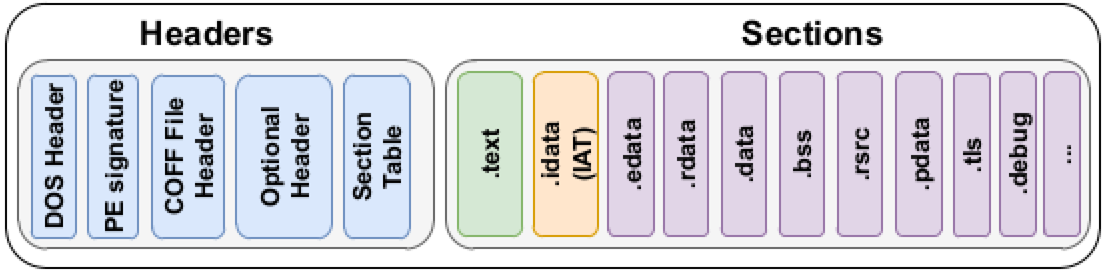
Malware representations studied

	Static	Dynamic
Graph	FCG+CFG	SCDG
Tabular	EMBER	Cuckoo summary

Ember features

2381 Features extracted with LIEF library

- Byte histogram
- Entropy histogram
- Strings features
- General information (size, layout, signatures,...)
- Header file information
- Section information (name, permission,...)
- Imported functions
- Exported functions
- Data directories

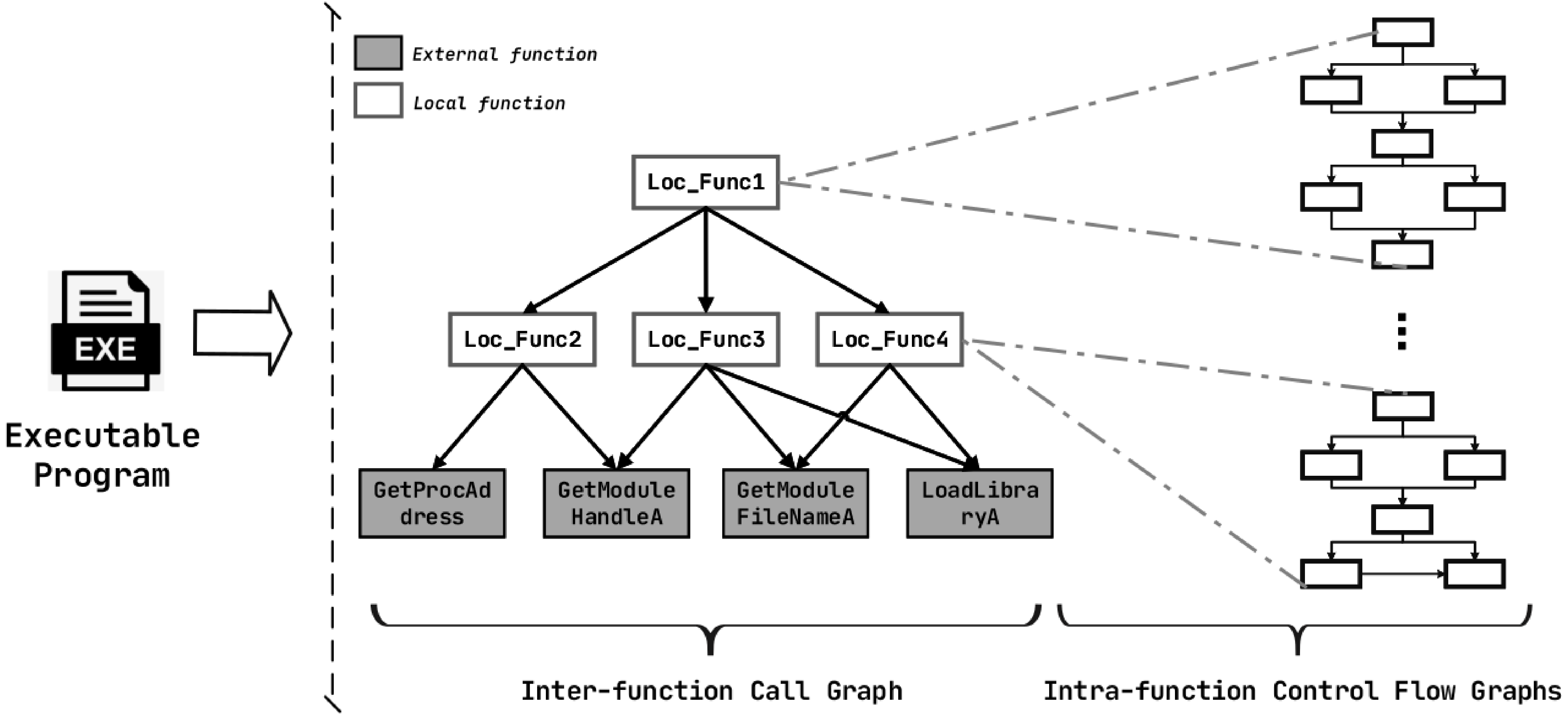


Anderson, H. S., & Roth, P. (2018). Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*.

Cuckoo summary example


```
{"summary":{
  "files":["C:\\Windows\\Globalization\\Sorting\\sortdefault.nls"],
  "read_files":["C:\\Windows\\Globalization\\Sorting\\sortdefault.nls"],
  "write_files":[],
  "delete_files":[],
  "keys":["HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\OLEAUT",
          "HKEY_LOCAL_MACHINE\\System\\CurrentControlSet\\Control\\Nls\\CustomLocale",
          "HKEY_LOCAL_MACHINE\\System\\CurrentControlSet\\Control\\Nls\\ExtendedLocale",
          "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\ExtendedLocale\\en-US",
          "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\Sorting\\Versions\\00060101.00060101",
          "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\Language Groups\\1"],
  "read_keys":["HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\CustomLocale\\en-US",
               "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\ExtendedLocale\\en-US",
               "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\Sorting\\Versions\\00060101.00060101",
               "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Nls\\Language Groups\\1"],
  "write_keys":[],
  "delete_keys":[],
  "executed_commands":[],
  "resolved_apis":["kernel32.dll.IsProcessorFeaturePresent",
                   "kernel32.dll.SortGetHandle",
                   "kernel32.dll.SortCloseHandle"],
  "mutexes":[],
  "created_services":[],
  "started_services":[]}}
```

FCG+CFG (FCCFG) Hierarchical graph

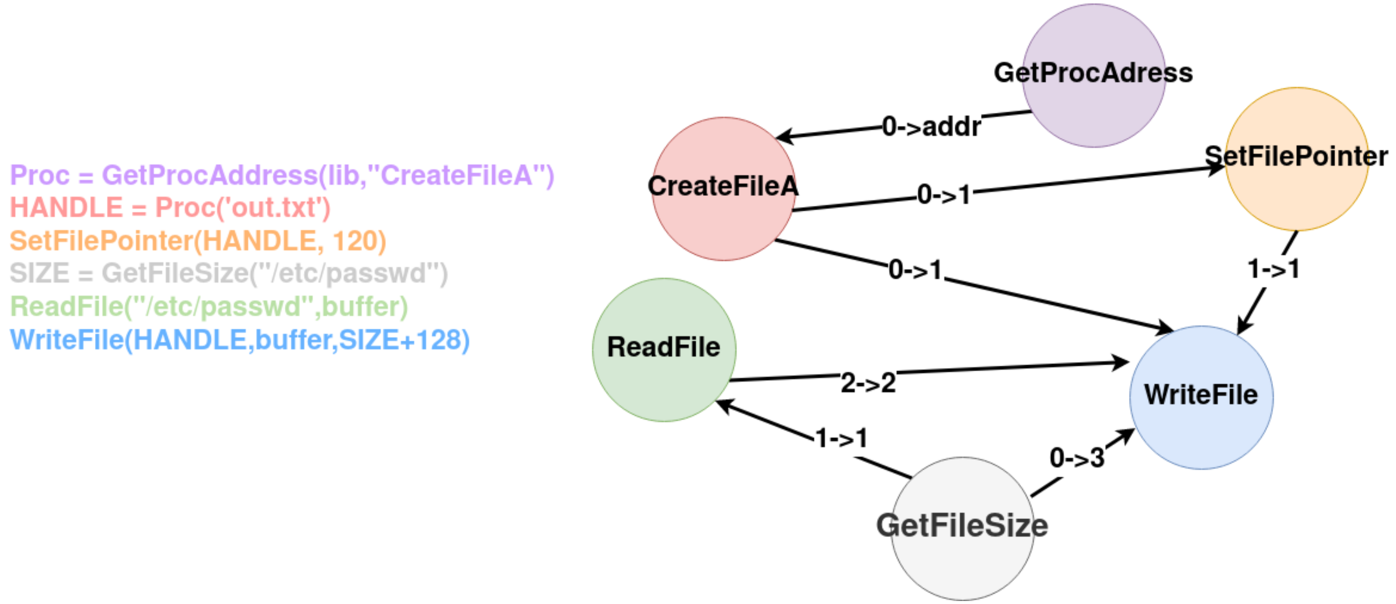


Ling, Xiang, et al. "Malgraph: Hierarchical graph neural networks for robust windows malware detection." IEEE INFOCOM 2022-IEEE Conference on Computer Communications. IEEE, 2022.

SCDGs - System Call dependency Graphs

Abstraction of the program **behavior**

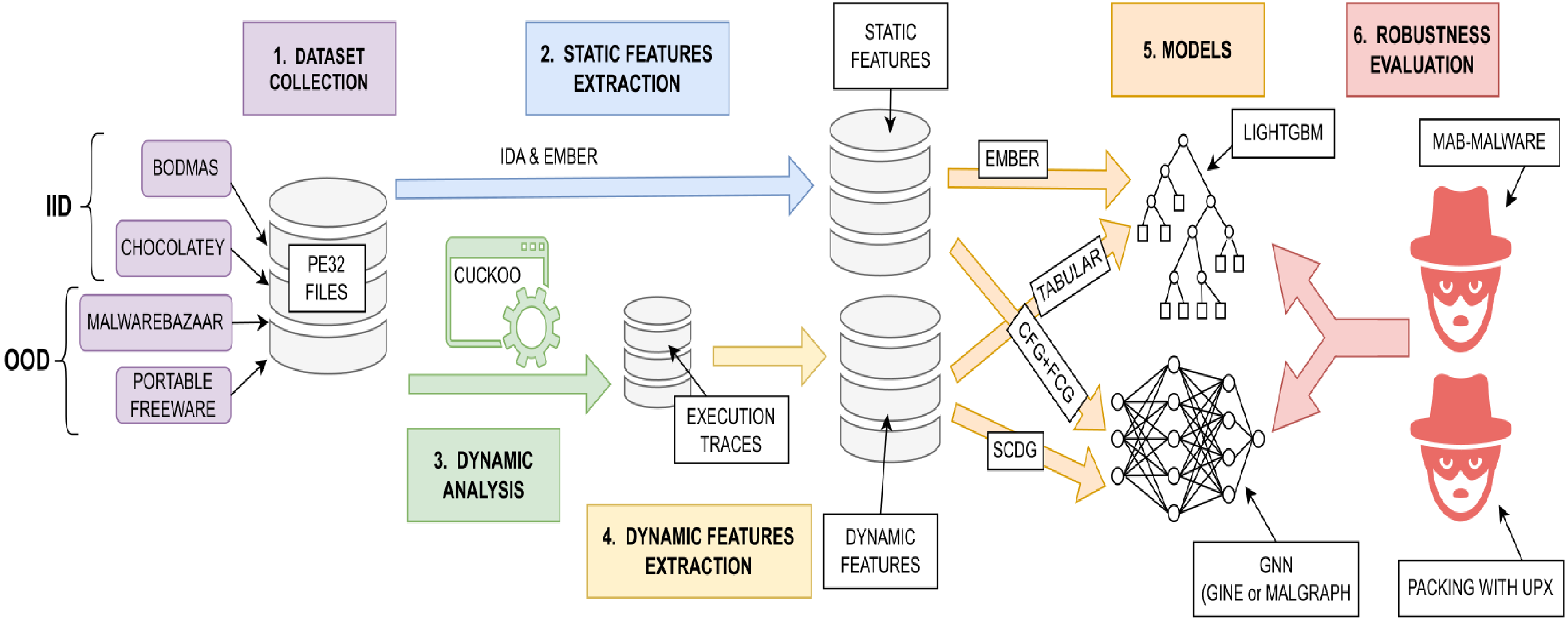
Vertices = Calls, Edges = Information flow between calls



Different strategies to build the graph: type of edges, merging calls, ...

Bertrand Van Ouytsel, C. H., & Legay, A. (2022, November). Malware analysis with symbolic execution and graph kernel. In *Nordic Conference on Secure IT Systems* (pp. 292-310). Cham: Springer International Publishing.

Methodology



Results

	Static		Dynamic	
	Tabular	Graph	Tabular	Graph
First quartile	0.012	0.122	0.003	0.011
Median	0.027	0.224	0.031	0.141
Third quartile	0.128	1.070	0.082	1.900

Dataset	Static		Dynamic		Combined
	tabular	graph	tabular	graph	
BODMAS	100	91.3	83.5	84.3	76.8
Chocolatey	99.8	99.2	81.9	83.0	80.7
MalwareBazaar	100	98.1	95.1	95.3	94.0
Portable Freeware	100	96.5	97.0	97.7	93.6

Extraction time

(Outside execution time in sandbox)

- Tabular are significantly cheaper
- High variance for graph based features

Feature extraction success rate

- More problem for dynamic features
- Chocolatey exhibits more failures

Results

		Static		Dynamic	
		Tabular	Graph	Tabular	Graph
IID Test set	Accuracy	**0.997***	0.987	0.995*	0.989
	Recall	**0.994***	0.993	0.993	**0.994***
	Precision	**0.999***	0.980	0.997*	0.984
	F1 score	**0.997***	0.987	0.995*	0.989
OOD Test set	Accuracy	**0.764***	0.762	0.705*	0.645
	Recall	0.537	**0.539***	0.414*	0.311
	Precision	0.984*	0.973	**0.991***	0.936
	F1 score	**0.695***	0.693	0.584*	0.467

*: best between tabular and graph; **Bold**: best overall.

	IID test set		OOD test set	
	Tabular	Graph	Tabular	Graph
ASR	0.991	0.000	0.984	0.014
C. Acc	0.996	0.990	0.532	0.795
R. Acc	0.999	/	0.106	0.833

Classification results

- Drop (expected) for OOD test set
- Tabular generally outperforms graphs

Adversarial attack results (static feature only)

- Attack success rate (ASR)
- Clean accuracy (accuracy after retraining on the original test set)
- Robust accuracy (accuracy after retraining on the evasive samples)

Results with packing (UPX)

We apply UPX packer on cleanware and malware to observe packing impact

	Static		Dynamic	
	Tabular	Graph	Tabular	Graph
Balanced Acc.	0.947	0.528	0.927	0.732
Recall	0.963	1.000	0.861	0.866
Precision	0.953	0.607	0.995	0.759
F1 score	0.958	0.755	0.923	0.809

Structure is altered and therefore, graph are impacted
Tabular seems more resilient

Wrap-up and future works

Static features are cheaper and seems more efficient as a first stage
But ...

- Well-known adversarial attacks
- Need XAI to ensure no shortcut/spurious correlation

Dynamic features could bring more information on behavior

In the future

- Other representations (Image ?)
- Evaluate robustness across representation
- Allow better explainability for graphs

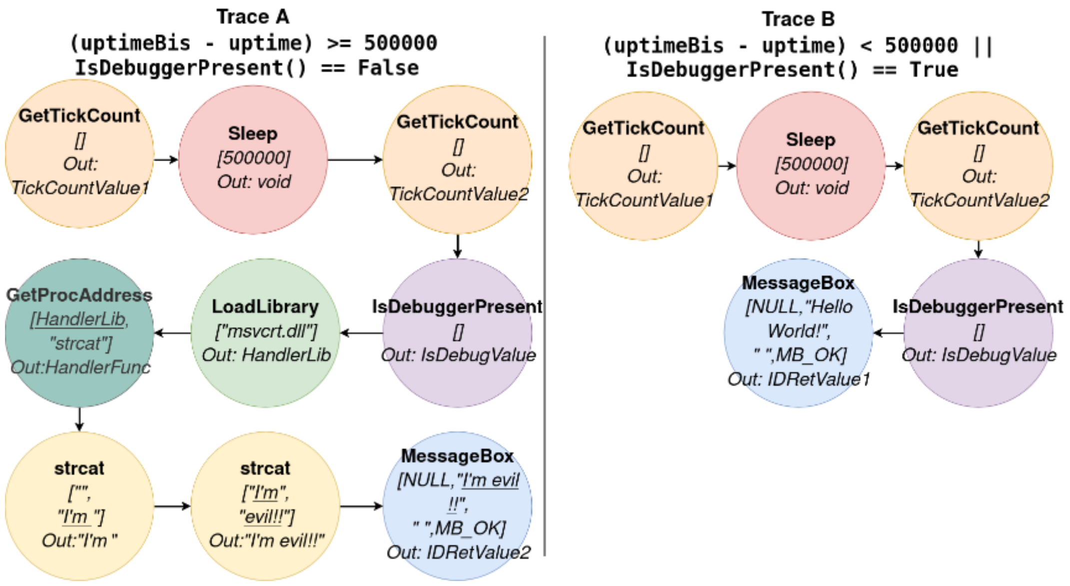
A look at other projects

Symbolic Execution to analyse malware ?

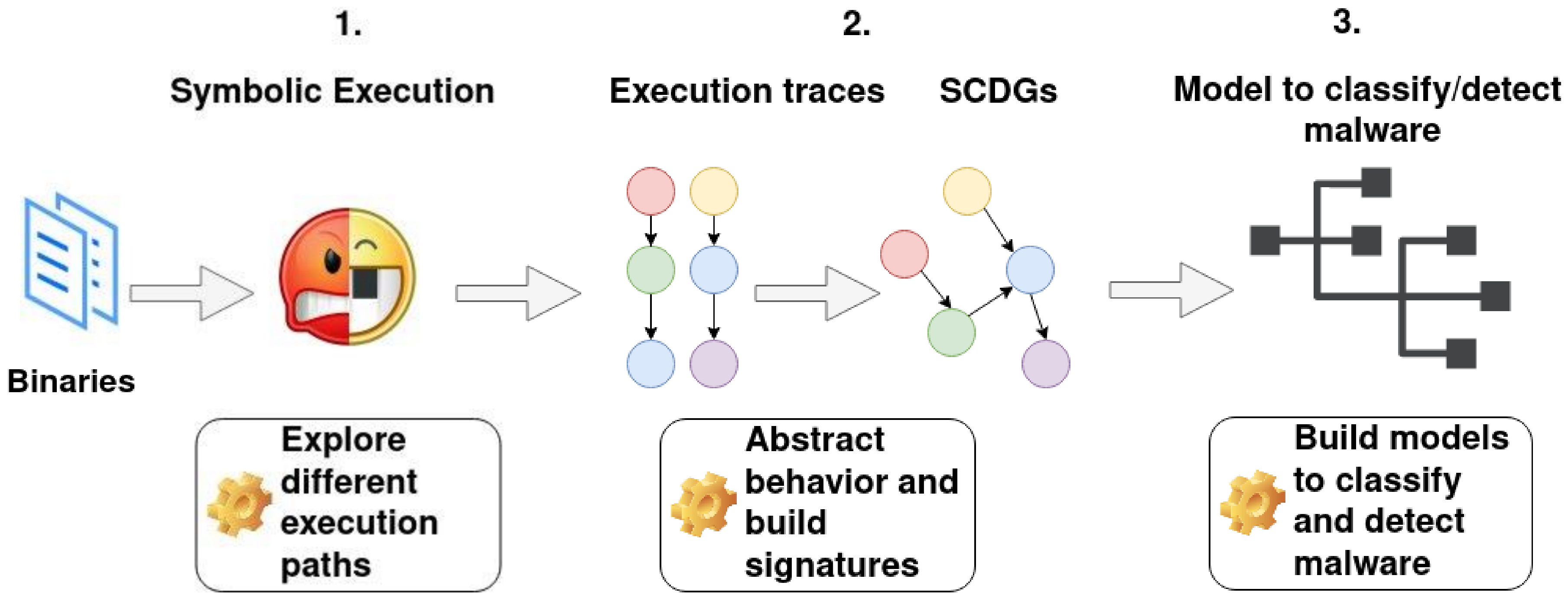
Symbolic execution is program analysis technique allowing to explore **multiple** execution paths at a time

Example: malware with two different execution paths and evasion tricks

```
ULONGLONG uptime = GetTickCount();
Sleep(500000);
ULONGLONG uptimeBis = GetTickCount();
if ((uptimeBis - uptime)<500000 || IsDebuggerPresent())
{
        MessageBox(NULL,"Hello world!","", MB_OK);
}
else
{
        char message[20] = "";
        HINSTANCE hlib = LoadLibrary("msvcrt.dll");
        MYPROC func = (MYPROC) GetProcAddress(hlib,"strcat");
        (func) (message, "I'm ");
        (func) (message, "evil!!");
        MessageBox(NULL, message, "", MB_OK);
 }
```



SEMA – Symbolic Execution for Malware Analysis

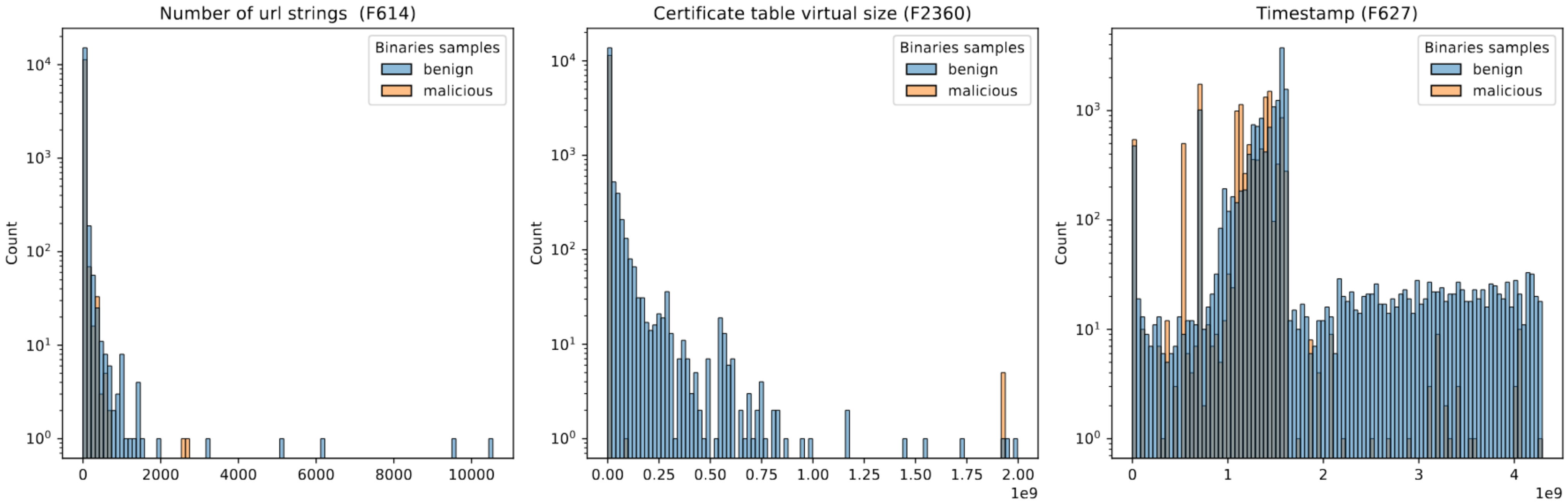


In principle, full exploration is a possibility; however, a number of challenges must be addressed, including scalability, environment modelling and path explosion.

Bertrand Van Ouytsel, Charles-Henry, et al. "Tool paper-SEMA: symbolic execution toolchain for malware analysis." *International Conference on Risks and Security of Internet and Systems*. Cham: Springer Nature Switzerland, 2022.

A deep dive into Ember features (WIP)

In recent experiments on academic datasets (BODMAS, EMBER, Packware, ERMDS,...), we observe that some features have a "disproportionate" impact on machine learning models due to the distribution of features.



A deep dive into Ember features (WIP)

Dataset	Model	AUROC	Precision	Recall	F1 Score
EMBER 2017	One Rule	0.8791	0.8435	0.7075	0.7696
	Adaboost	0.9473	0.9492	0.9451	0.9472
	LightGBM	0.999	0.9891	0.9839	0.9865
	Tabular Learner	0.9875	0.9817	0.9761	0.9789
EMBER 2018	One Rule	0.765	0.6755	0.8199	0.7407
	Adaboost	0.8415	0.8165	0.8811	0.8475
	LightGBM	0.9853	0.9305	0.9529	0.9416
	Tabular Learner	0.9582	0.9152	0.9188	0.917
BODMAS	One Rule	0.9038	0.7899	0.7789	0.7844
	Adaboost	0.9791	0.9644	0.9853	0.9747
	LightGBM	0.9997	0.9968	0.9888	0.9927
	Tabular Learner	0.9878	0.9817	0.9632	0.9724
ERMDS	One Rule	0.9037	0.9559	0.7743	0.8556
	Adaboost	1.0	1.0	1.0	1.0
	LightGBM	1.0	1.0	0.9995	0.9998
	Tabular Learner	0.9577	0.9635	0.9015	0.9315

One Rule = Just use one feature to classify

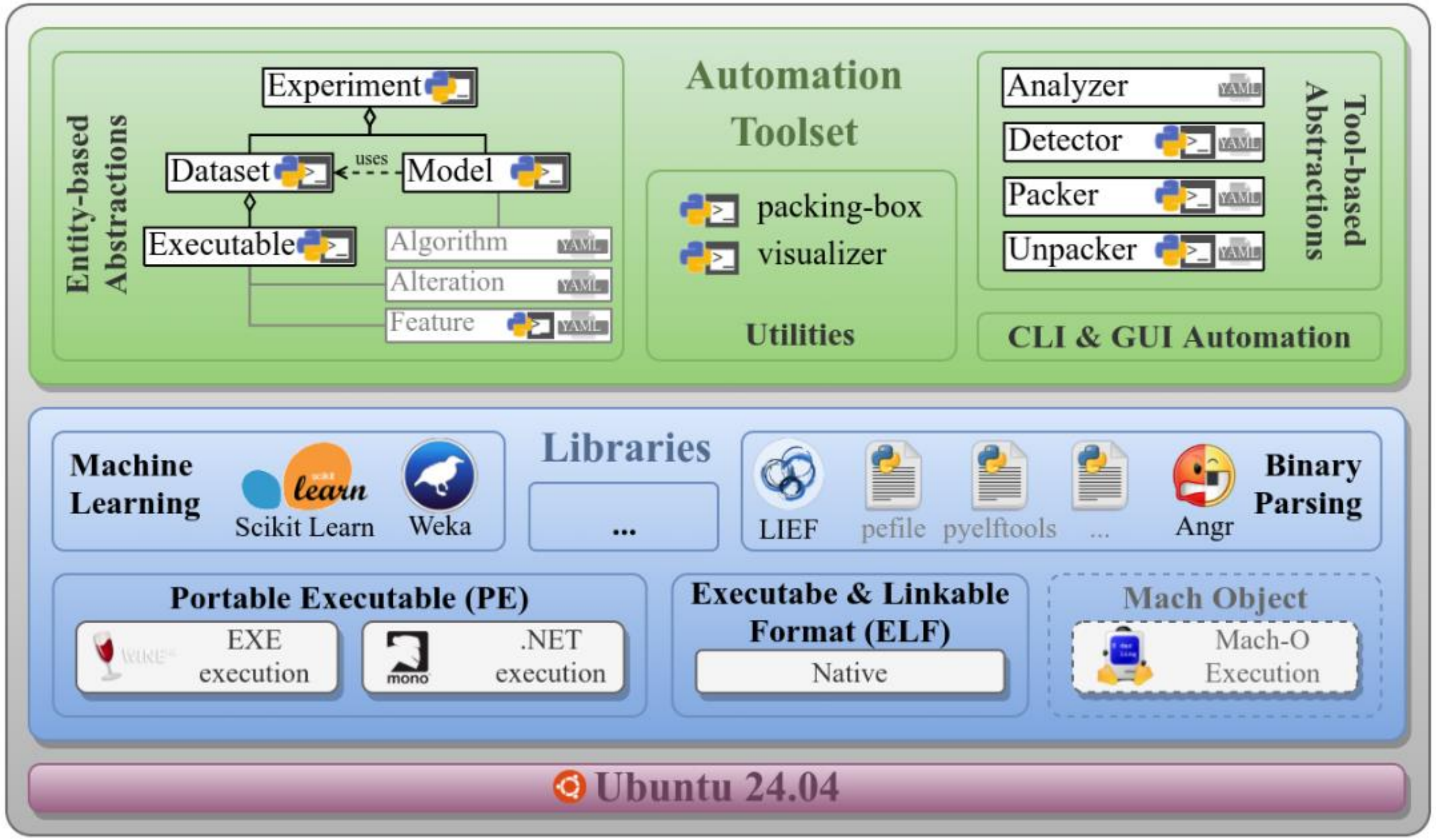
Top features used by more complex models include (but are not limited to) these features

PackingBox – Experimental toolkit for static detection of executable packing.

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

Research often faces reproducibility issues

PackingBox aims to help with research on packing detection



Last paper on PackingBox: D'Hondt, Alexandre, Charles Henry Bertrand Van Ouytsel, and Axel Legay. "Evading Packing Detection: Breaking Heuristic-Based Static Detectors." DIMVA 2024.

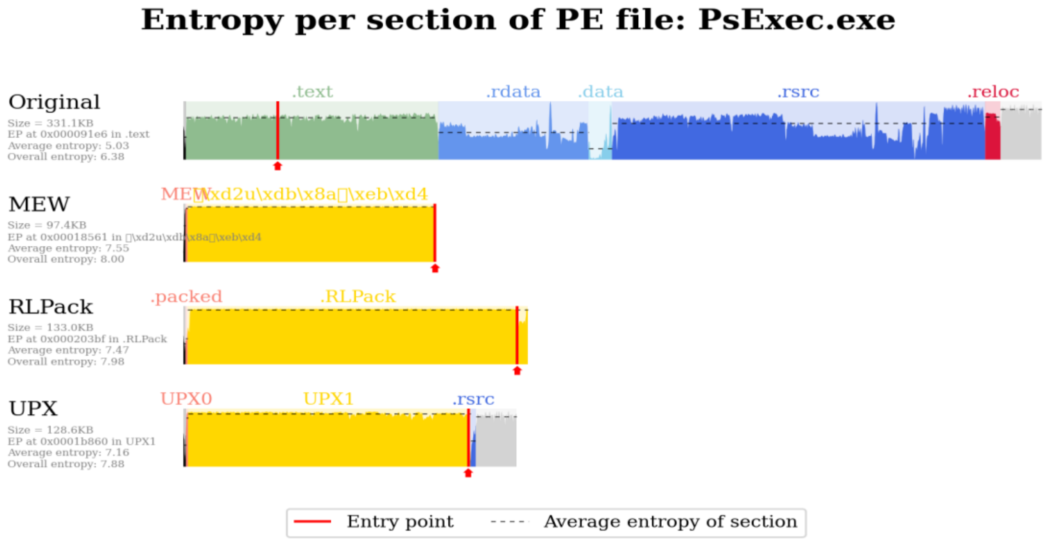
PackingBox – Experimental toolkit for static detection of executable packing.

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

Research often faces reproducibility issues

PackingBox provides an environment to :

- Manage dataset and pack samples with known packers
- Extract various features
- Train your machine learning models
- Evaluate known packing detector
- Provide visualisation tools
- Integrate alterations to executables files
- Apply alterations on file
- …

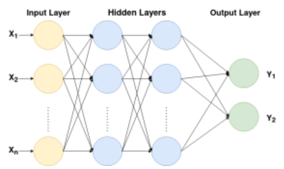


Last paper on PackingBox: D'Hondt, Alexandre, Charles Henry Bertrand Van Ouytsel, and Axel Legay. "Evading Packing Detection: Breaking Heuristic-Based Static Detectors." DIMVA 2024.

Understanding classifiers' decisions (previous work on packing detection)

Explainable AI (XAI): To explain classifiers' decision

Improve understanding and allow easier troubleshooting

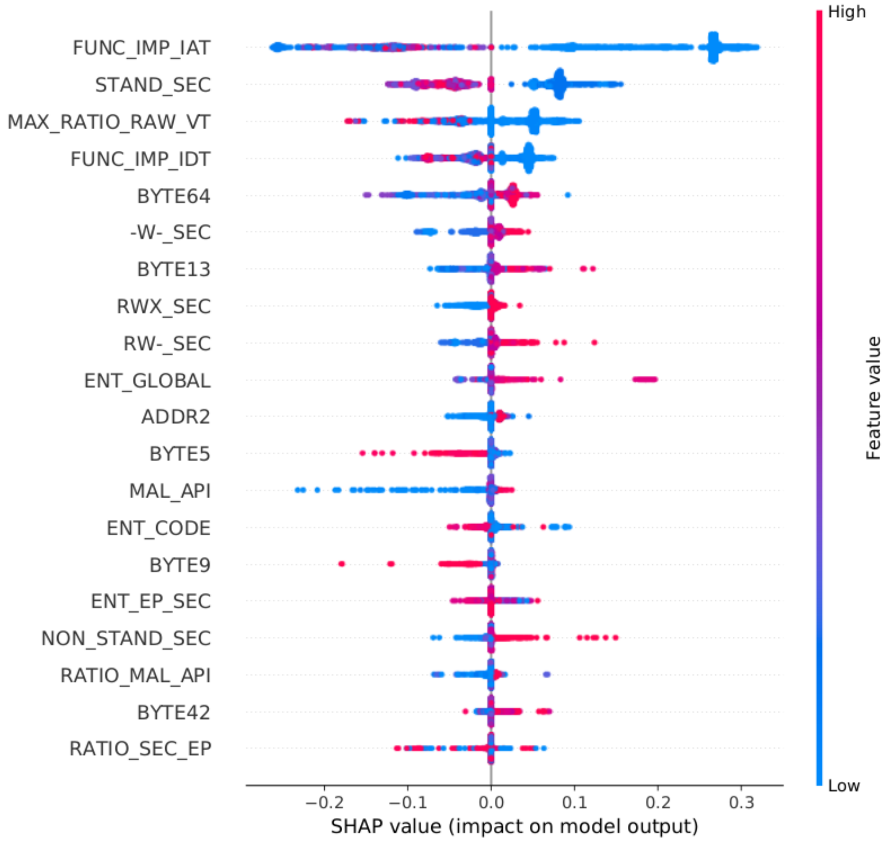


Shapley values: Evaluate contribution of each feature to the prediction

Each point = feature value of a sample

Color indicate feature value

Left side: *predicted* non-packed sample
Right side: *predicted* packed sample



Bertrand Van Ouytsel, Charles-Henry , Khanh Huu The Dam, and Axel Legay. "Analysis of machine learning approaches to packing detection." *Computers & Security* 136 (2024): 103536.

Thank you for your attention
Any questions?