ATLAS

Security framework
based on formal methods
and machine learning

RIVER

Vulnerability detection based on **dinamic** analysis

RIVER is a solution designed to find hidden vulnerabilities in native x86 files. The testing method for finding these bugs is to randomly choose inputs and test them against the program (fuzz testing).

- *Dynamic taint analysis*. RIVER can mark the parts of the input that were used during a program's execution.
- *Symbolic execution engine*. Using RIVER's symbolic execution engine along with Microsoft's Z3 SMT solver, one can efficiently explore new paths leading to better code coverage.
- *Concolic execution engine*. Due to the path explosion limitation of symbolic execution, one can combine it with concrete execution leading to improved results.

Binary x86 code analysi, source code not needed

Machine Learning and Reinforcement Learning algorithms

MALID

Malware detection based on **static** analysis

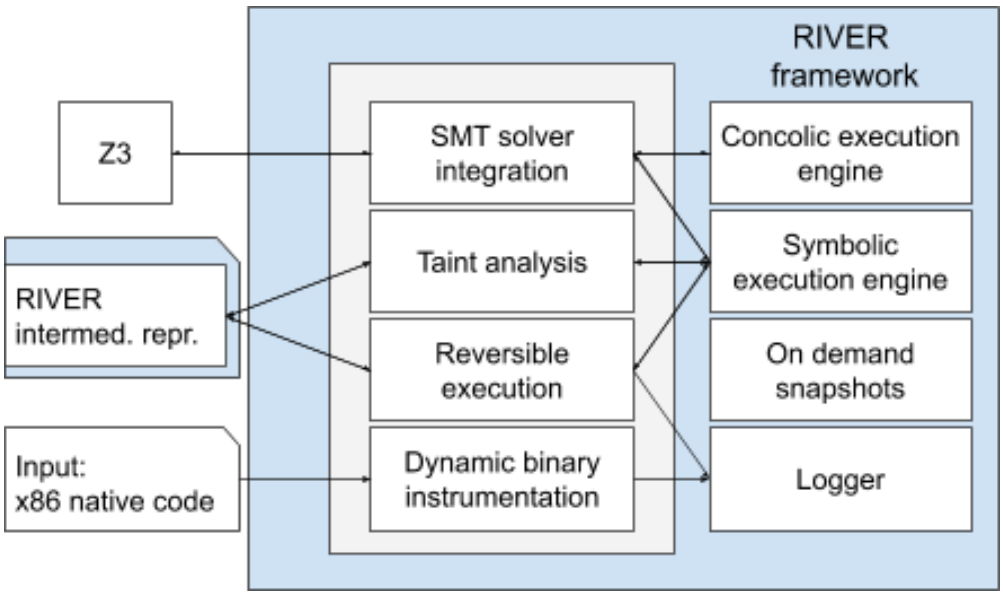
MALID is a low complexity Machine Learning (ML) solution based on a new Dictionary Learning (DL) method that is designed to manage ever-growing collections of malware files.

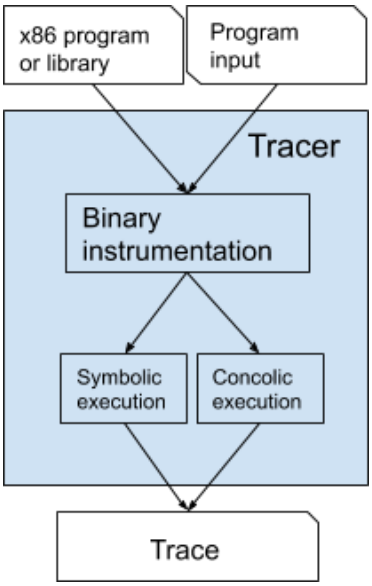
- *Semi-supervised*. The framework includes an initial kick-starting supervised training stage that requires only a small-sized dataset. The second, unsupervised stage is performed online. Once a sample is classified it is then used for training, which increases knowledge.
- *Online.* Unsupervised learning is performed one file at a time, which ensures fast and incremental model updates.

Low complexity, lightweight model

Adaptive to new malware types

RIVER





How it works

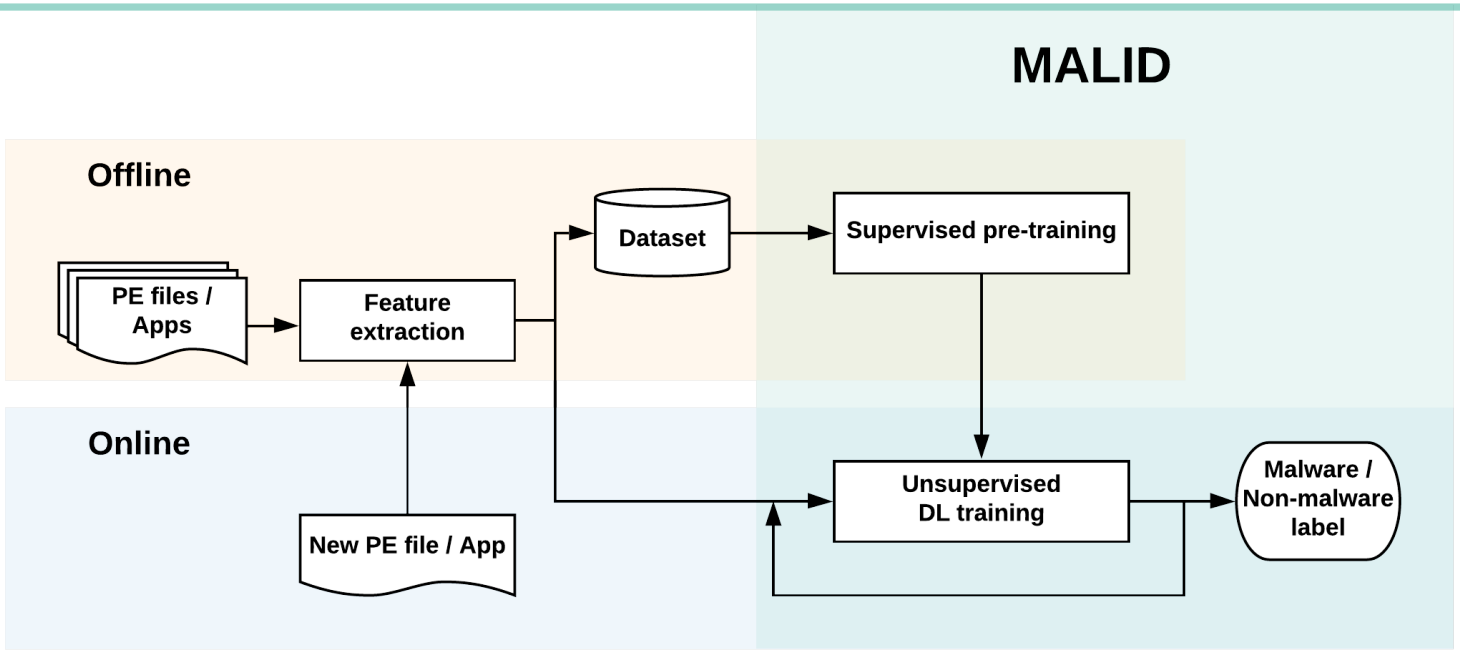
In order to use the RIVER solution, a user must supply a x86 binary file or a library that will be tested. Additionally, one must also supply a wrapper function specifying which function needs to be tested. After this step, one may test the program using multiple algorithms in order to find inputs that lead to crashes (such as: segmentation faults, hangs - timeout is manually set - or signals - sigfpe, sigint, sigabrt etc.).

The main program (called river.tracer) must be passed an input and it begins to trace the execution of the program. Firstly, it uses the dynamic binary instrumentation to disassemble the x86 code and reassemble it with the jump conditions modified, so that it calls special routines when the code is executed.

When the code executes, it will produce a trace that can be followed to see which instructions were executed for a given input and then an algorithm can modify the input to obtain better code coverage.

Software package available at `https://github.com/AGAPIA/river`

MALID



How it works

Portable Executable (PE) files or mobile apps descriptions are transformed in a vectorized collection of features. A preliminary classification model is trained in supervised manner on a small set of files. Whenever a new file is up for classification, it is run through the unsupervised stage. Here it is labeled and used to re-train the model, ensuring model adaptation to unseen types of malware. As newly labeled files can influence model performance, a parameter called forgetting factor controls model sensitivity to new samples.

The method, Tolerant Online Discriminative DL with Regularization (TODDLeR), extends an existing DL algorithm by adding two Tikhonov regularization factors that ensure that the updated model deviates in a controlled manner from the previous step.

Software package available at `https://github.com/abaltoiu/malid`

About ATLAS

The ATLAS project (“Hub inovativ pentru tehnologii avansate de securitate cibernetică”) plans on improving research and collaboration performances in the cyber-security domain, by addressing themes of high interest: security of applications, operating systems, IoT and cloud. The ATLAS project is financed through a research grant from the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project no. PN-III-P1-1.2-PCCDI-2017-0272, in the PNCDI III programme.