Adversarial Machine Learning and its application to Malware

Héctor D. Menéndez

Department of Computer Science
University College London

5th April 2019

Understanding Adversarial ML

How machine learning works?

What is the ML influence in malware detection?

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

Are there any protections?

Understanding Adversarial ML

How machine learning works?

What is the ML influence in malware detection?

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

Are there any protections?

Machine Learning

Statistical process that learns from a specific discrimination related to a set of objects.

Clustering: Divides objects into groups blindly, based on similarities.

Classification: Supervised identification of patterns in objects with the aim of separating them.

Learning Example

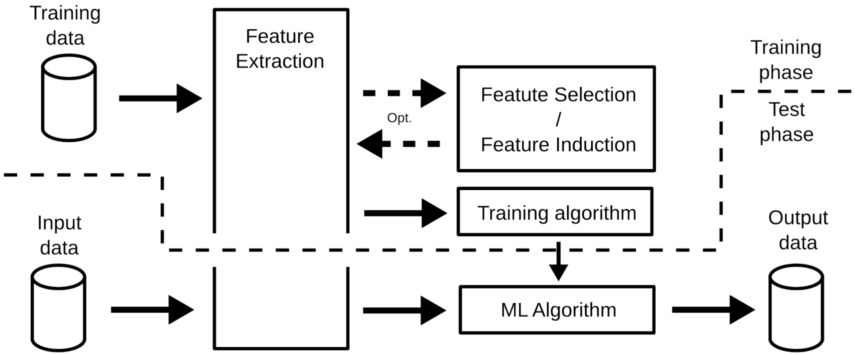
You want to group objects by similarity.

1) Extract information about the objects or **features** (preferably numerical).

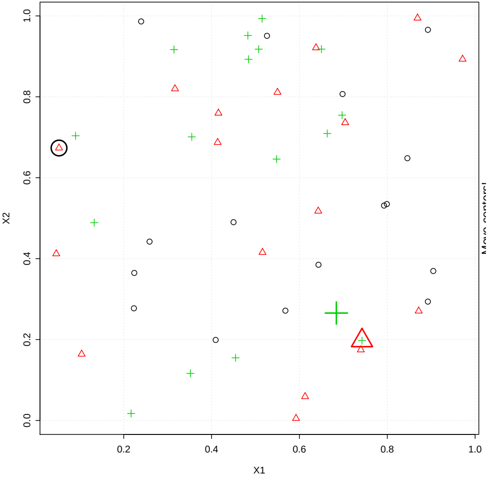
2) Define your notion of **similarity**.

3) Set your separability criteria and learning process, i.e., your **algorithm**, and run it.

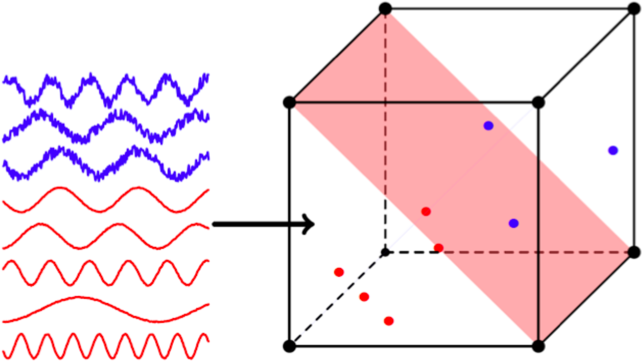
General Structure



Clustering

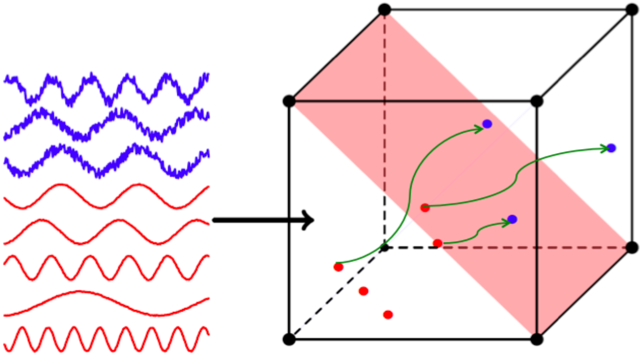


Classification



The Adversarial ML

Adversarial Machine Learning looks for vulnerabilities in the discrimination to cheat the algorithm.



Understanding Adversarial ML

How machine learning works?

What is the ML influence in malware detection?

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

Are there any protections?

The Malware Arms Race

Malware/Benign-ware classification

Researchers normally aim to create a methodology to distinguish malware and benign-ware.

Current works apply **classification** algorithms for this aim.

These algorithms learn from **program features** and aim to identify patterns on them.

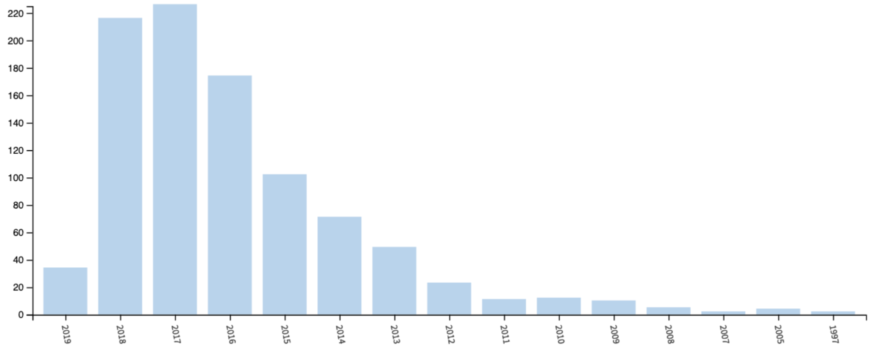
Program features

Static analysis: information from the disassemble version of the program, from the control flow graph, etc.

Dynamic analysis: information from traces, network, registers, etc.

Binary-based analysis: information from the entropy or n-gram distribution of files.

Malware & ML production



Index

How machine learning works?

What is the ML influence in malware detection?

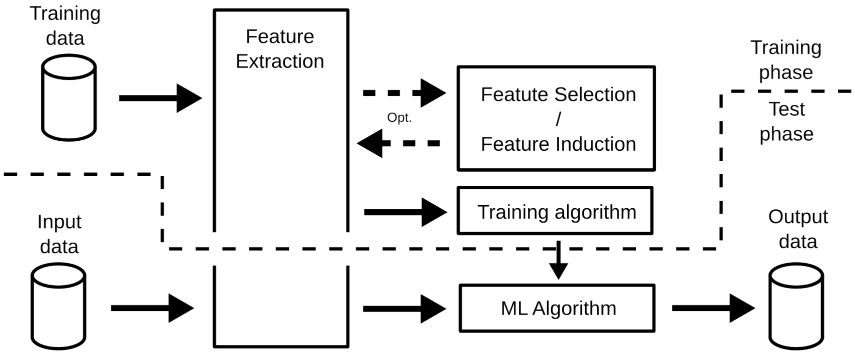
Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

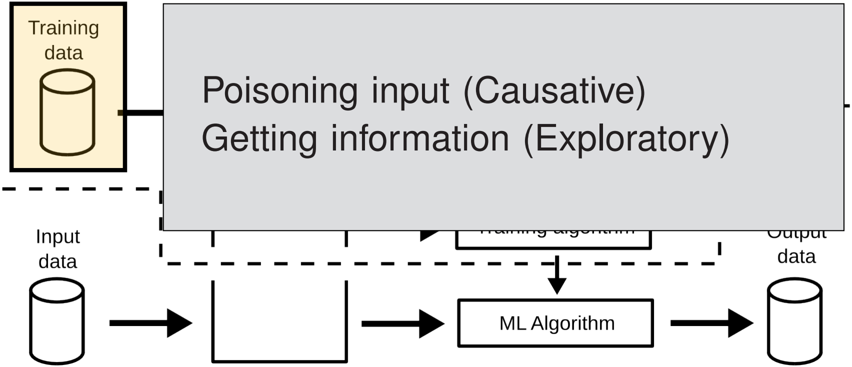
How does Adversarial ML work in practice?

Are there any protections?

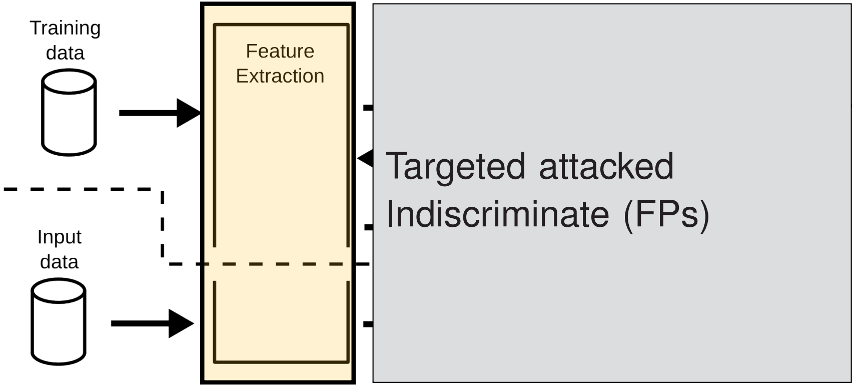
Machine Learning Vulnerabilities



Machine Learning Vulnerabilities



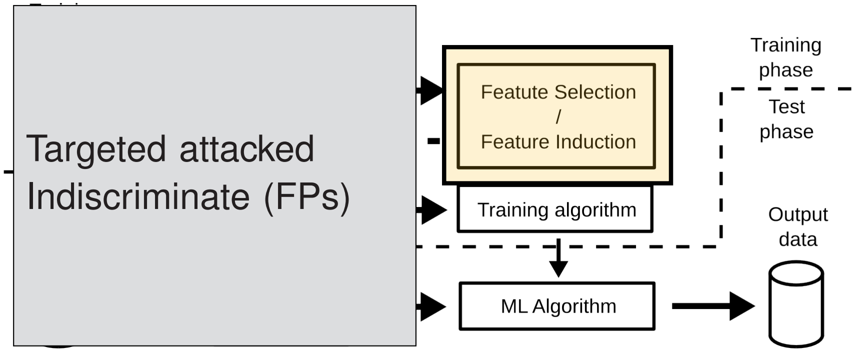
Machine Learning Vulnerabilities



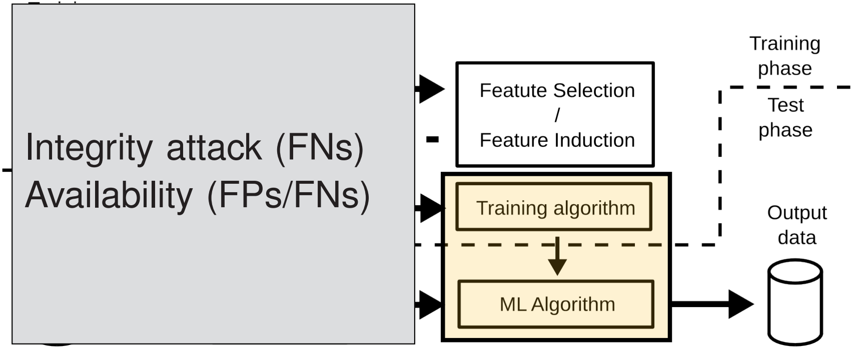
Machine Learning Vulnerabilities

Targeted attacked

Indiscriminate (FPs)



Machine Learning Vulnerabilities



Index

How machine learning works?

What is the ML influence in malware detection?

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

Are there any protections?

What is a vulnerability?

There are three relevant agents in ML: the *oracle*, the feature space and the algorithm

The **oracle** provides the ground truth (e.g. labels)

The **feature space** represents the data features

The **algorithm** learns to discriminate using the features and the oracle information

Train/Test Distributions

ML supposes same train and test distributions

Adversaries part from this hypothesis aiming to find mistakes on the discrimination

Where are these mistakes?

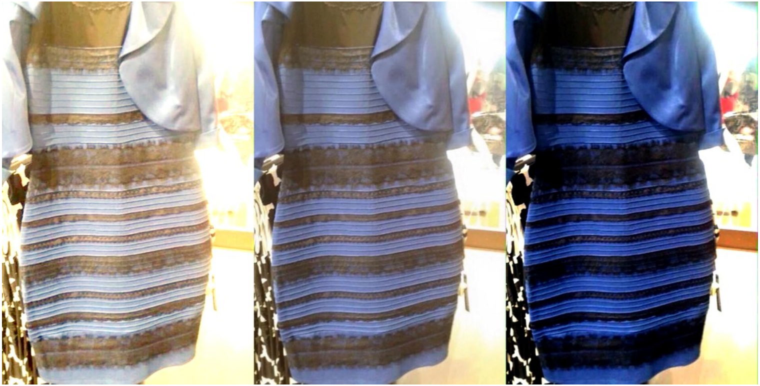
Cheating the oracle

Which color is this dress?

Cheating the oracle

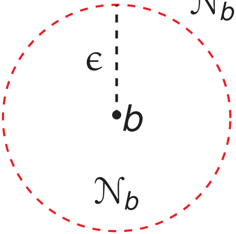
And now?

Cheating the oracle



Cheating the feature space

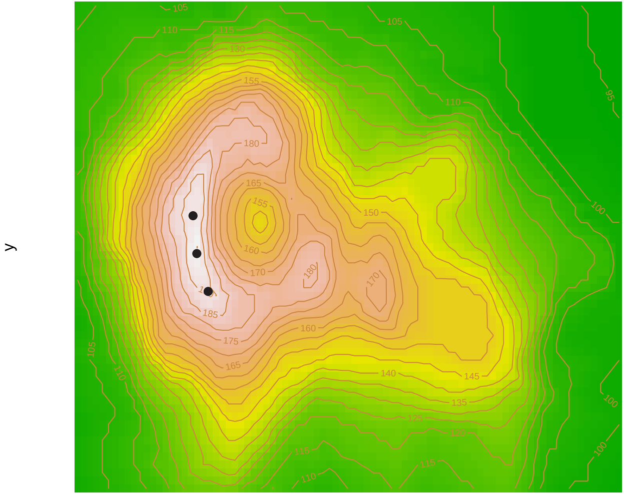
Consider *b* known instance.



$$\mathcal{N}_b = \{x \mid d(x, b) < \epsilon \wedge C(x) = C(b)\}$$

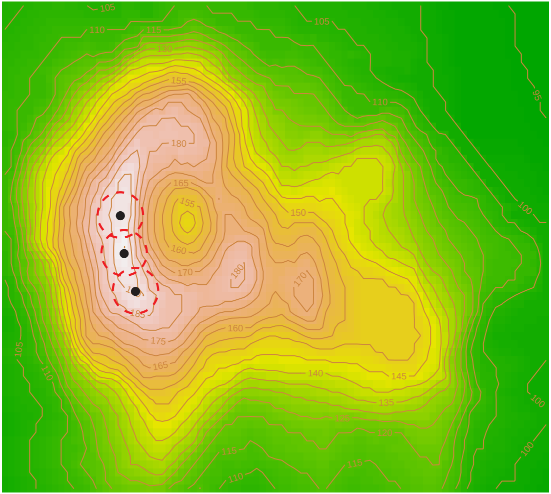
$$\exists y \in \mathcal{N}_b \mid O(y) \neq O(b)?$$

Cheating the feature space



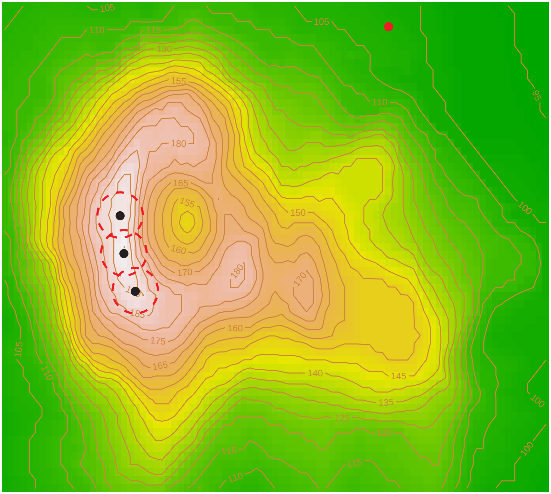
Cheating the feature space

y



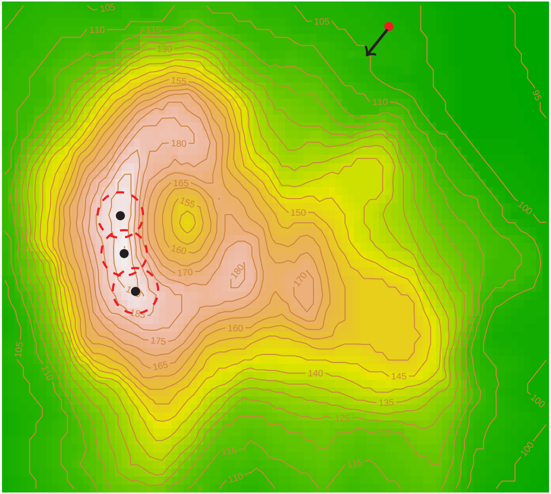
Cheating the feature space

y



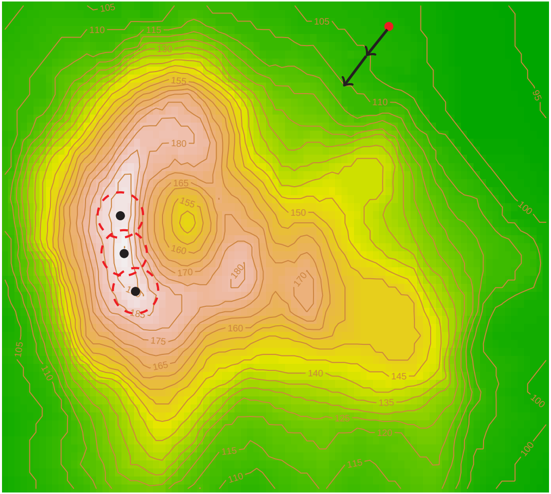
Cheating the feature space

y

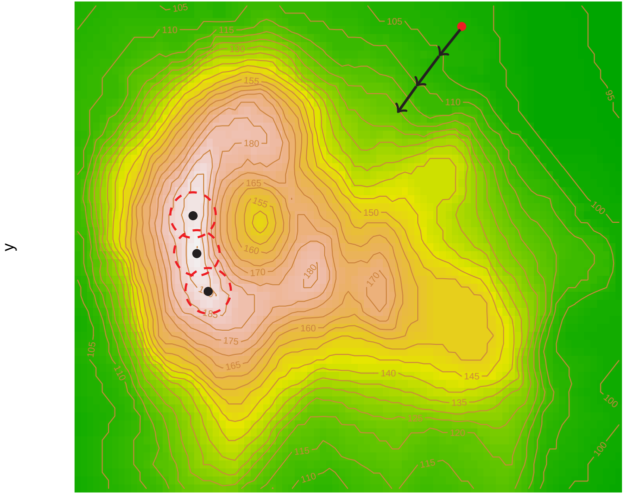


Cheating the feature space

y

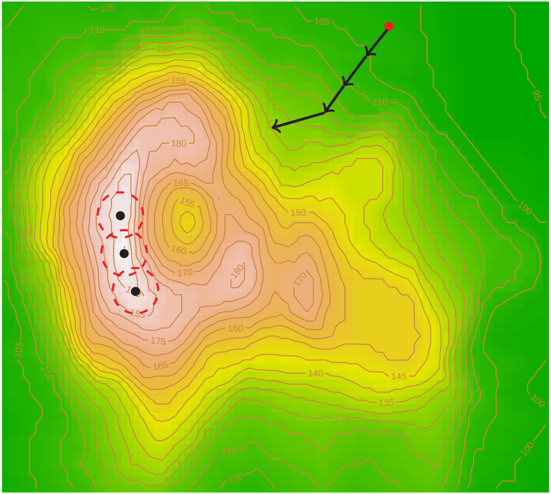


Cheating the feature space

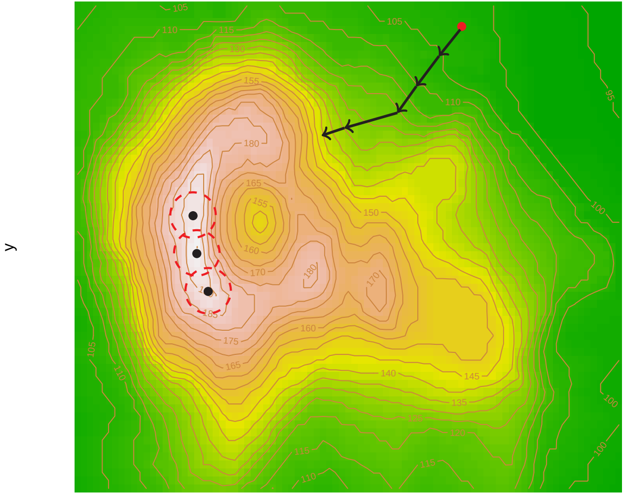


Cheating the feature space

y

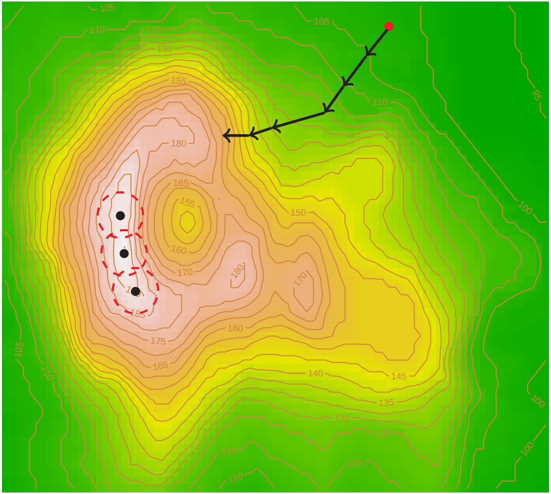


Cheating the feature space

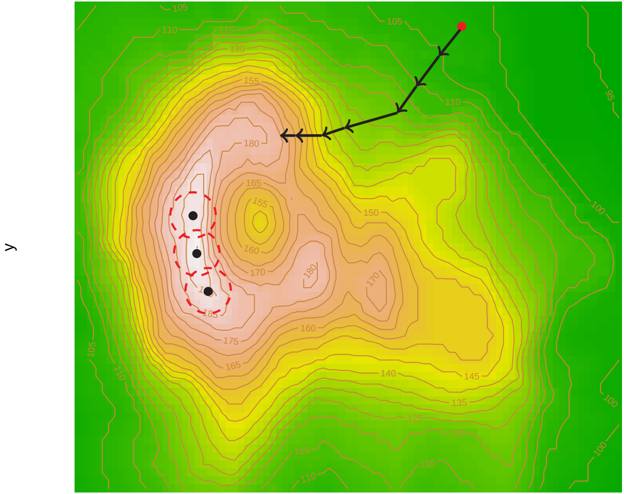


Cheating the feature space

y

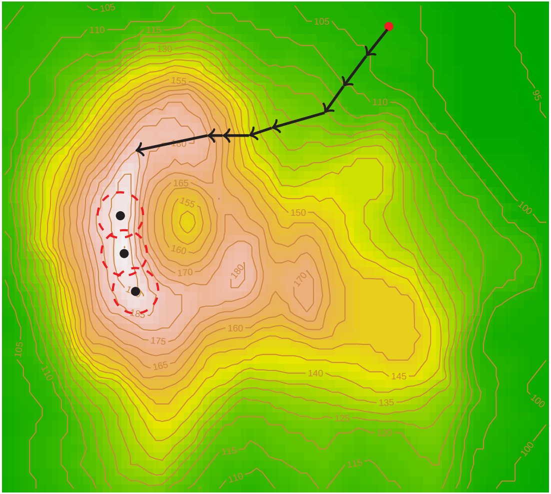


Cheating the feature space



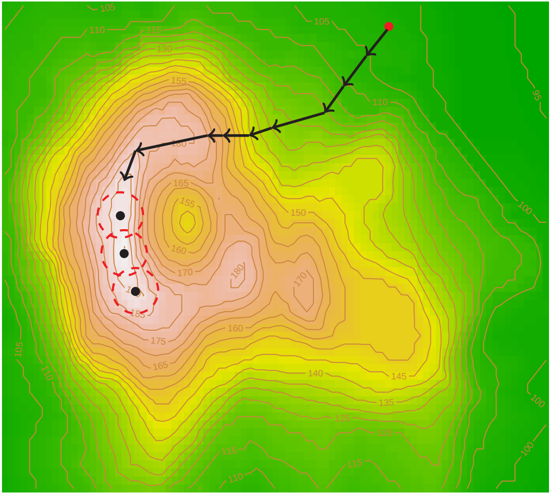
Cheating the feature space

y



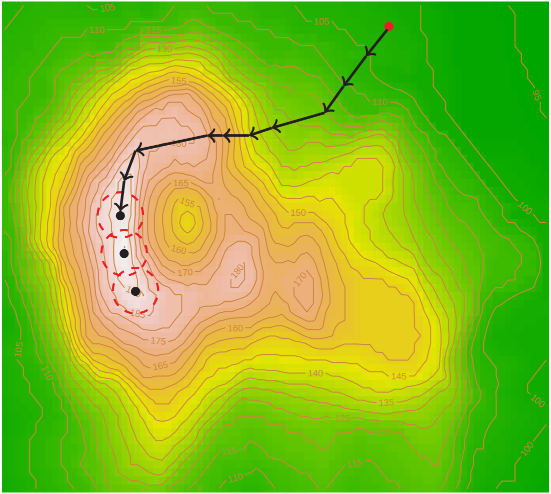
Cheating the feature space

y



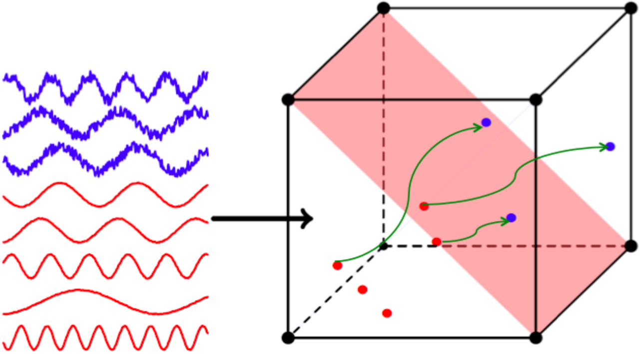
Cheating the feature space

y

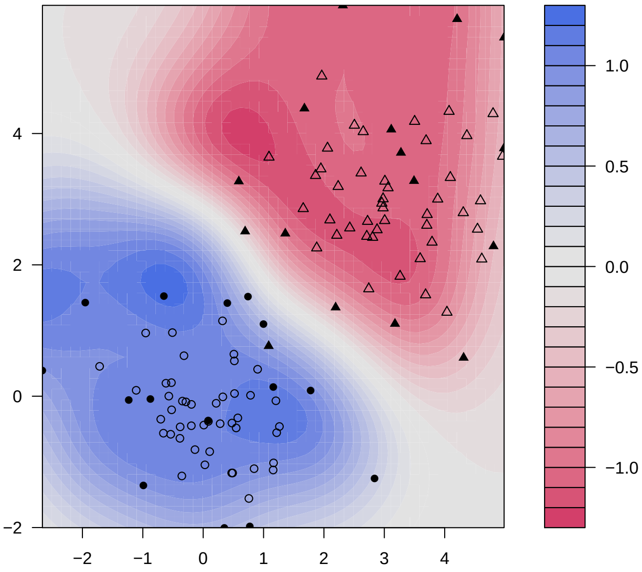


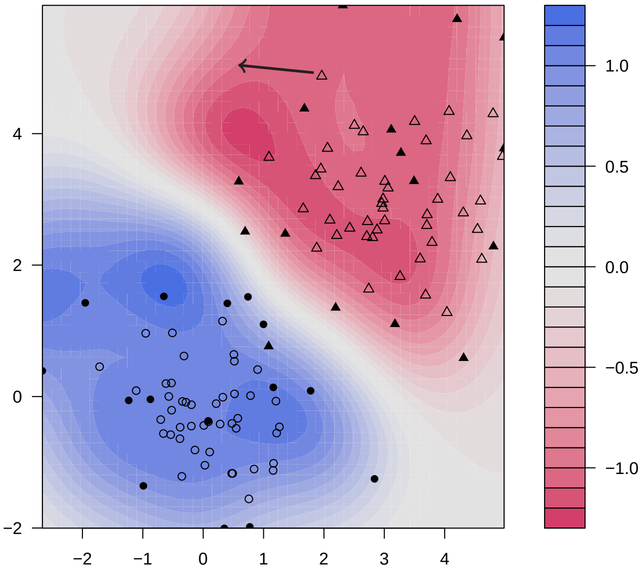
Cheating the classifier

Jump the wall strategy

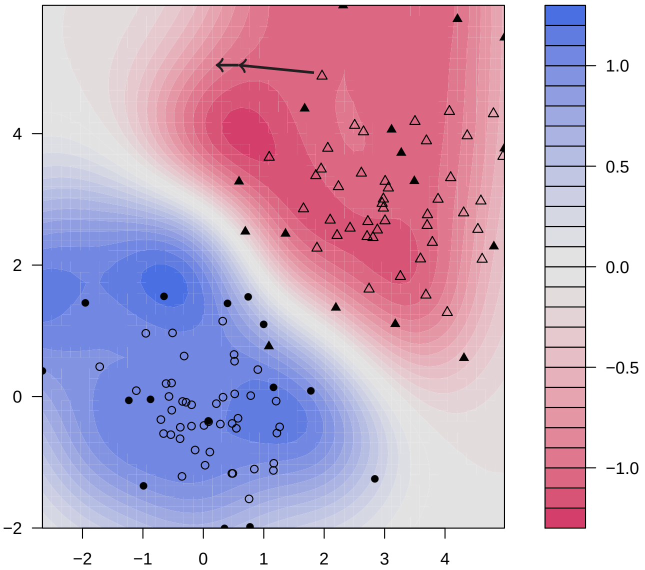


SVM classification plot

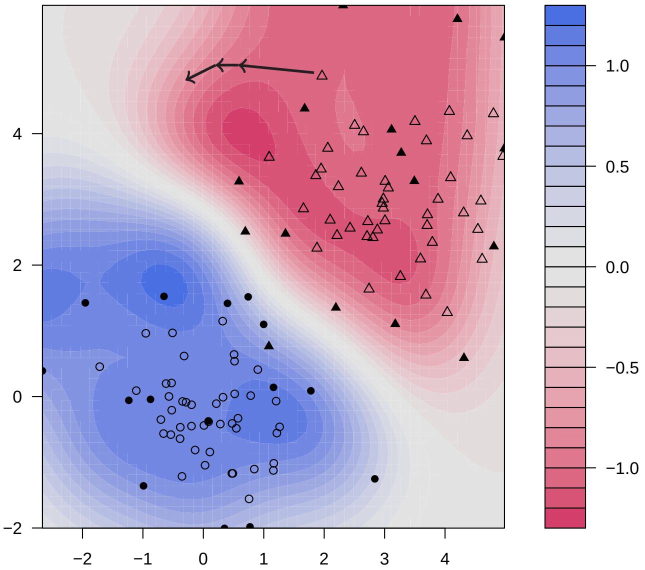



4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

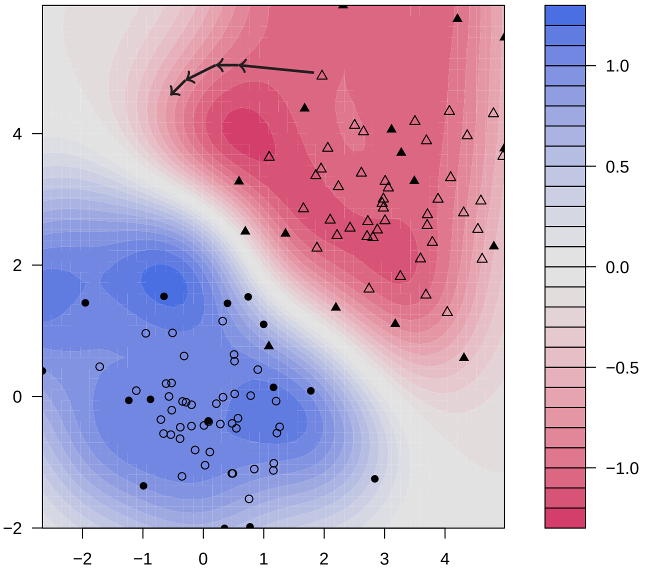
SVM classification plot



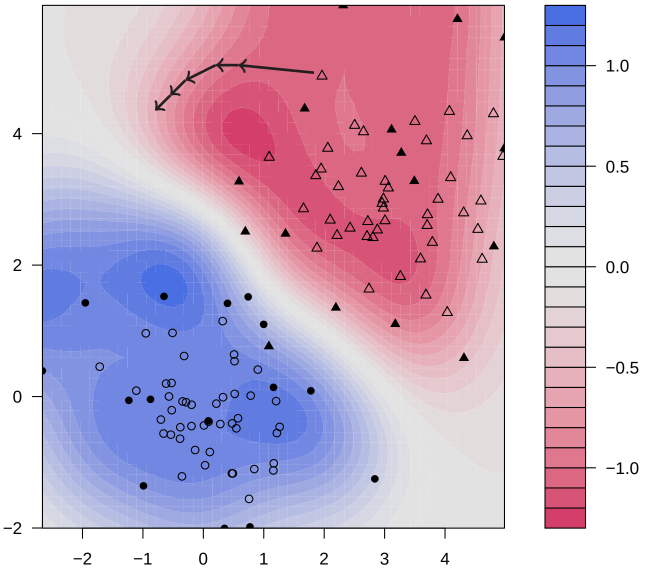
SVM classification plot



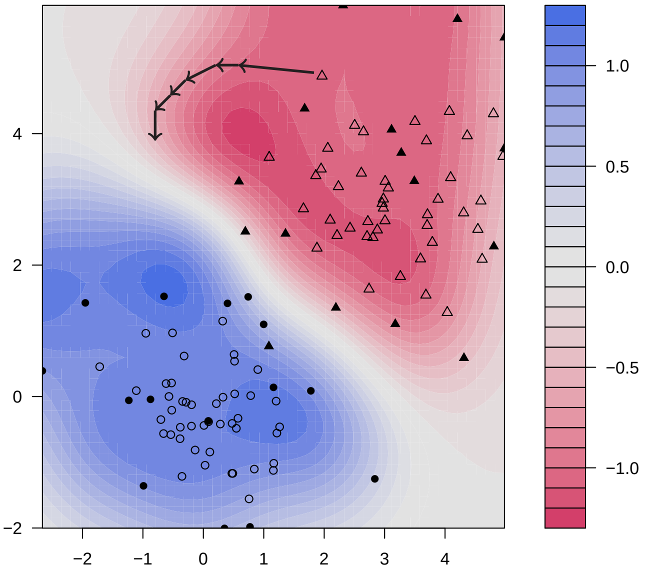
SVM classification plot

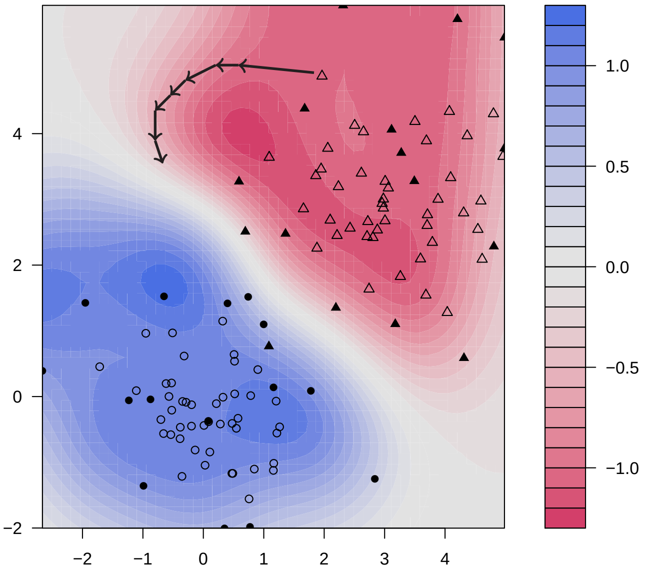


SVM classification plot

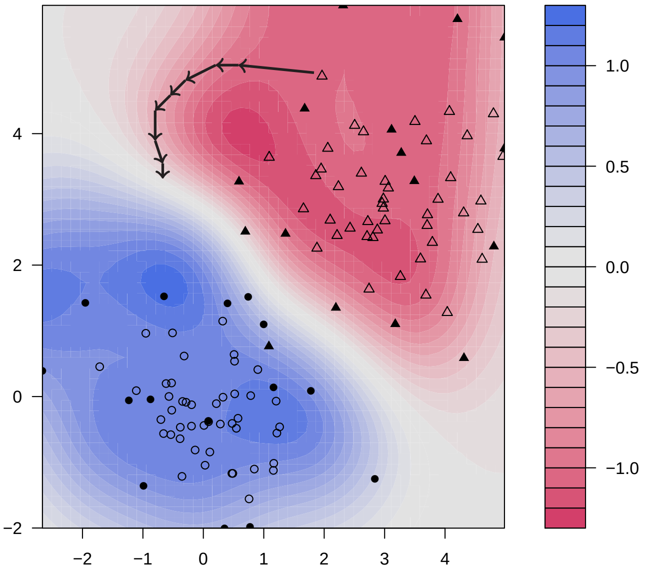


SVM classification plot

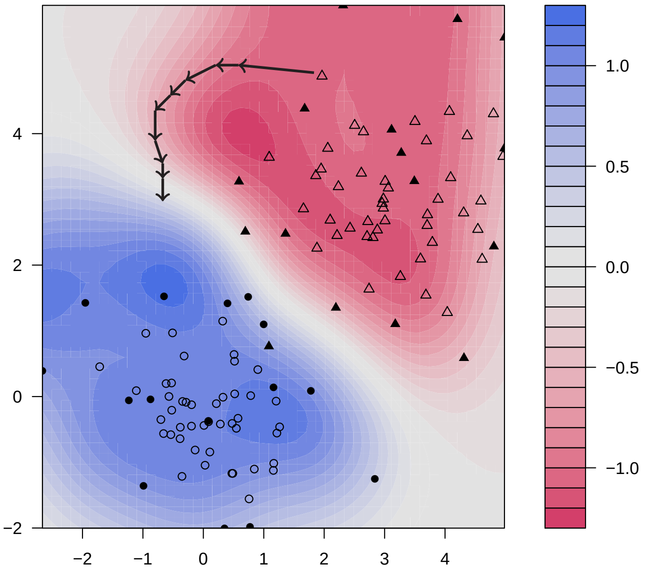



4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

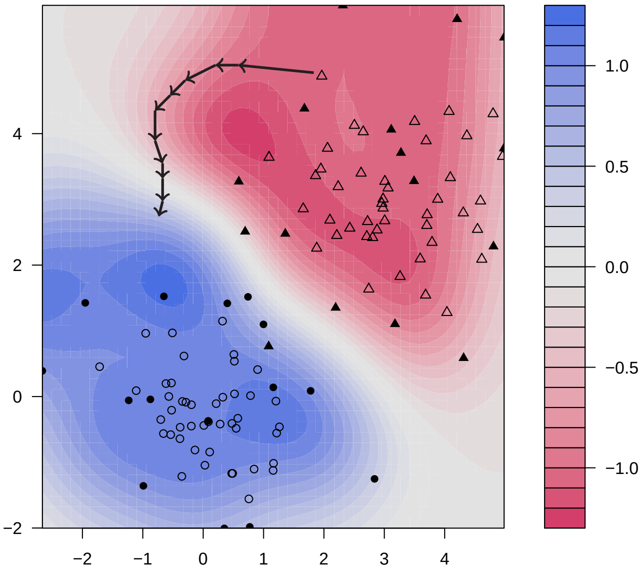
SVM classification plot

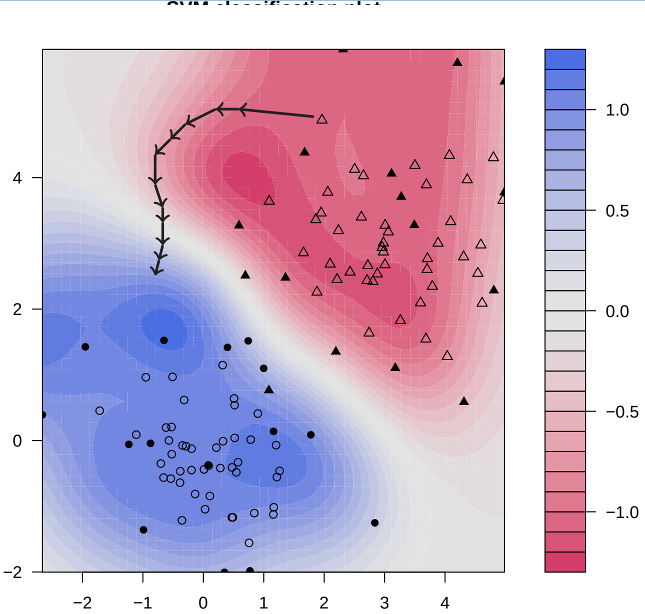


SVM classification plot



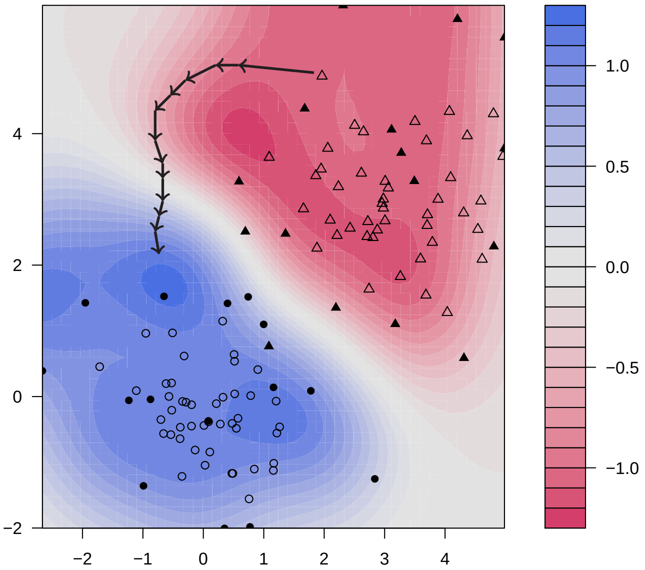
SVM classification plot



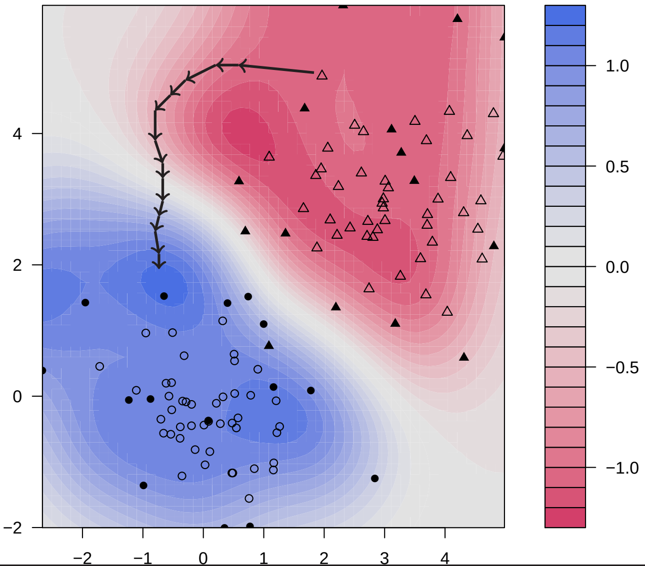


4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

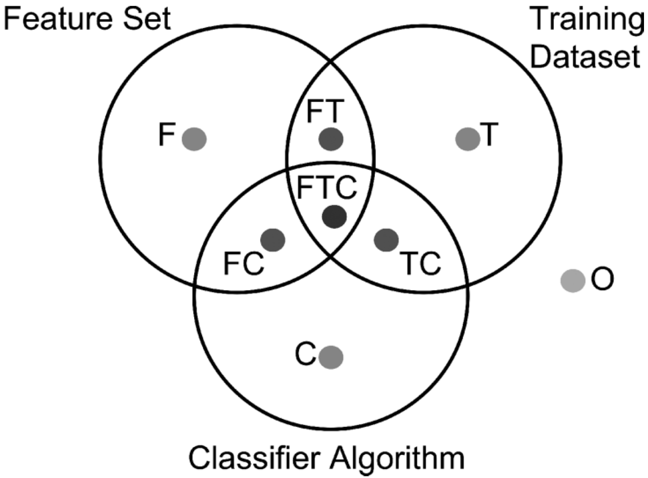
SVM classification plot



SVM classification plot



What the adversary knows?



What the adversary knows?

Level 0 (basic): Knows the oracle decision and has access to the detector $\implies$ Blind feedback

Level 1: Knows the classifier $\implies$ Construction vulnerabilities

Level 2: Knows the feature space $\implies$ Knows the relevant features

Level 3: Knows the training data $\implies$ Replication

Index

How machine learning works?

What is the ML influence in malware detection?

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

Are there any protections?

Adversarial ML in practice: 3 Use Cases

EvadeML

EEE

IagoDroid

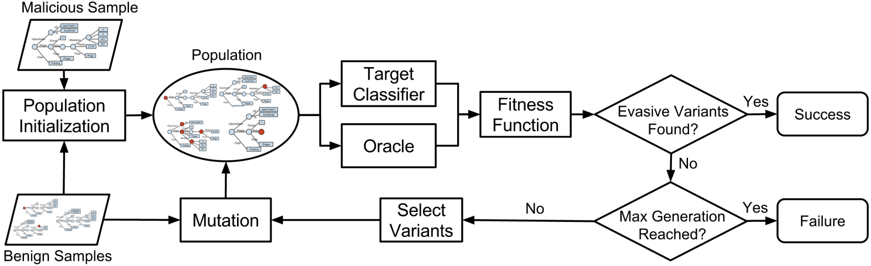
Adversarial ML in practice: EvadeML

EvadeML aims to defeat 2 PDF malware detectors

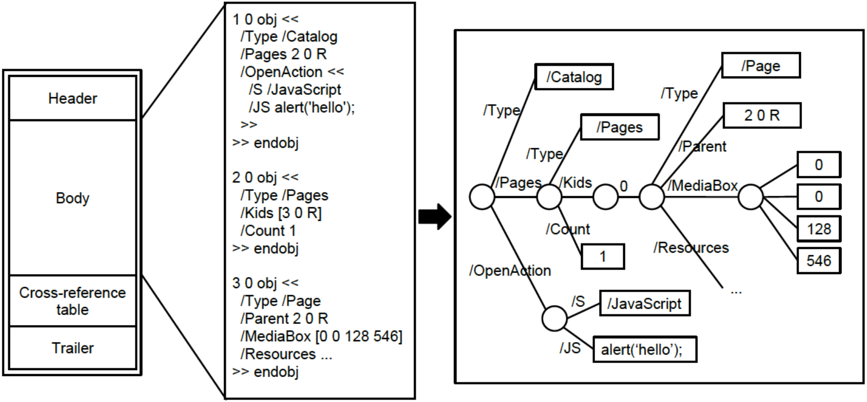
It uses Genetic Programming to generate variants

It is a Level 3 adversary: replicates the detectors

EvadeML Model



EvadeML Encoding



EvadeML Results

	PDFrate	Hidost
Accuracy	0.9976	0.9996
False Negative Rate	0.0000	0.0056
False Negative Rate against Adversary	1.0000	1.0000

EEE

EEE changes the malware shape via **intelligent packing**

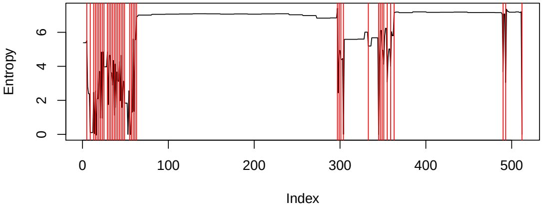
It injects controlled **entropy** regions to alter signatures and entropy

It **learns from the classifier** using evolutionary computation

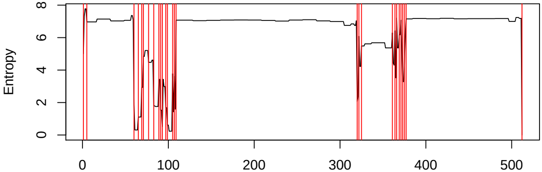
It needs **no information** about: the detector, training data or feature space (Level 0)

EEE: The Evolutionary Packer

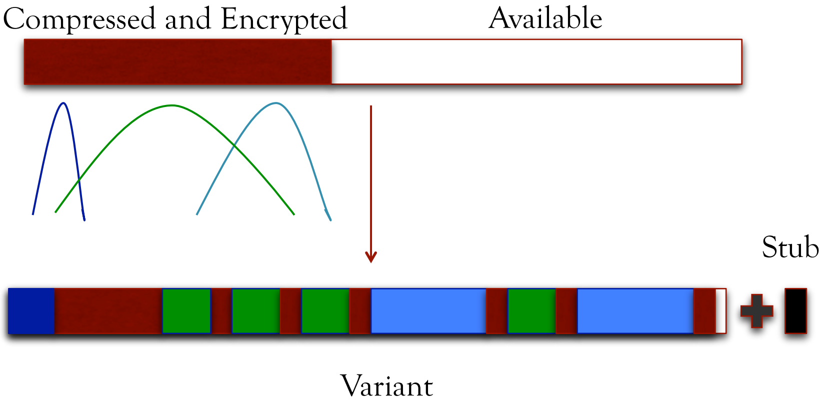
Original



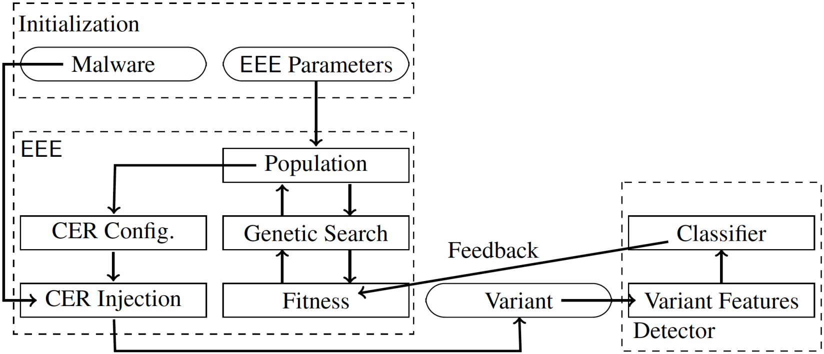
After EEE



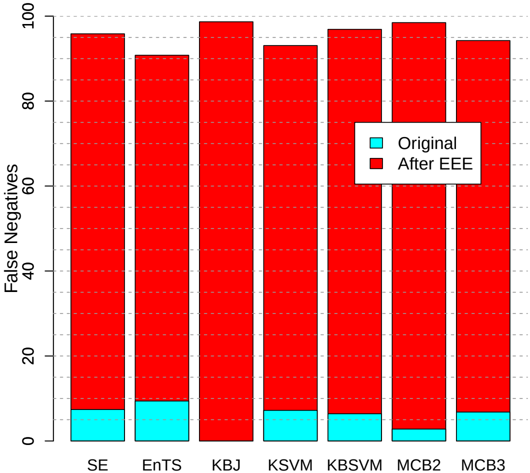
EEE: The Evolutionary Packer



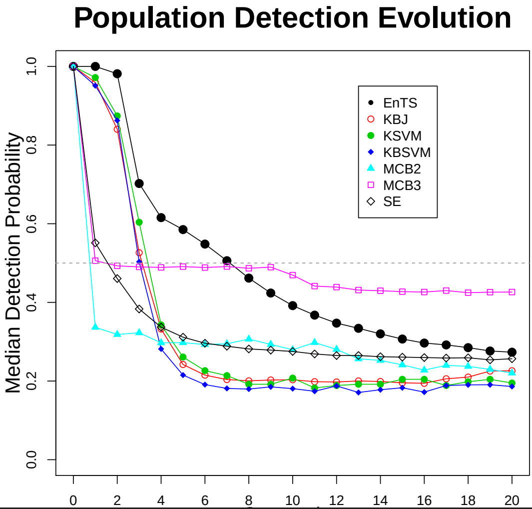
EEE: The Evolutionary Packer



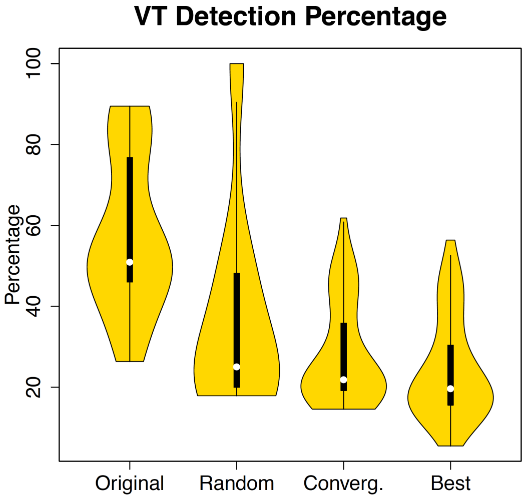
EEE: The Evolutionary Packer



EEE: The Evolutionary Packer



EEE against Anti-Viruses



IagoDroid

IagoDroid

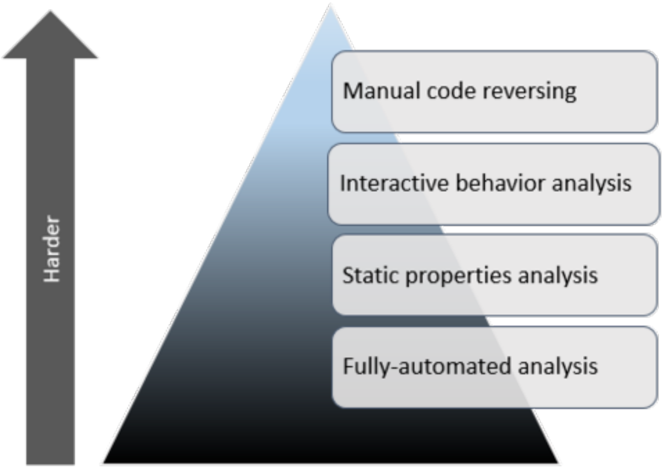
Attacks the malware **triage** process

It finds weaknesses on the **feature space**, incrementing some features in realistic margins

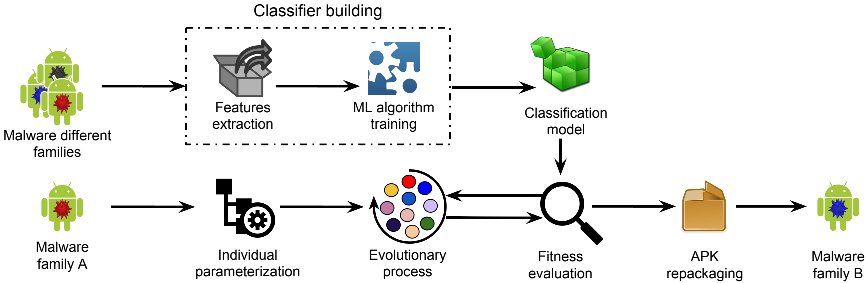
It aims to **reduce the number of changes**

It replicates the detector (Level 3)

The Triage process



IagoDroid


Classifier building
Malware different families
Features extraction
ML algorithm training
Classification model
Malware family A
Individual parameterization
Evolutionary process
Fitness evaluation
APK repackaging
Malware family B

RevealDroid

Classifier	Code structures	Permissions	Api Calls	Intent-actions	Flow analysis	Tested for families classification	Freely available to download
RevealDroid (Garcia et al., 2015)	✗	✗	✓	✓	✓	✓	✓
DroidSIFT (Zhang et al., 2014)	✗	✓	✓	✓	✓	✗	✗
Dendroid (Suarez-Tangil et al., 2014)	✓	✗	✗	✗	✗	✓	✓
Drebin (Arp et al., 2014)	✗	✓	✓	✓	✗	✓	✗
DroidMiner (Yang et al., 2014)	✗	✗	✓	✓	✗	✓	✗
DroidAPIMiner (Aafer et al., 2013)	✗	✗	✓	✗	✗	✗	✗
VILO (Lakhotia et al., 2013)	✓	✗	✗	✗	✗	✓	✗
DroidLegacy (Deshotels et al., 2014)	✗	✗	✓	✗	✗	✓	✓
MAST (Chakradeo et al., 2013)	✓	✓	✗	✓	✗	✗	✗

Results

Family	First Sol.	Avg. Conv.	Avg. Mod.	Feature
Plankton	1	3.3	1.0	ACTION_INPUT_METHOD_CHANGED (0.7)
GinMaster	1	3.7	1.0	SMS_MMS (0.6)
Kmin	1	4.3	1.0	ACTION_USER_PRESENT (0.6)
Glodream	1	4.7	0.8	ACTION_INPUT_METHOD_CHANGED (0.4)
BaseBridge	Inf	Inf	-	-
Nyleaker	1	3.6	1.0	NETWORK__LOG (0.4)
Gappusin	1	3.4	0.9	ACTION_INPUT_METHOD_CHANGED (0.3)
Geinimi	1	3.9	1.0	NETWORK_INFORMATION (0.5)
Imlog	1	4.7	1.2	ACTION_INPUT_METHOD_CHANGED (0.7)
DroidKungFu	1	7.2	0.7	IPC__NETWORK (0.2)
Iconosys	1	3.5	1.1	NETWORK__LOG (0.3)
Adrd	1	3.6	0.8	ACTION_INPUT_METHOD_CHANGED (0.5)
DroidDream	1	4.1	0.8	ACTION_INPUT_METHOD_CHANGED (0.4)

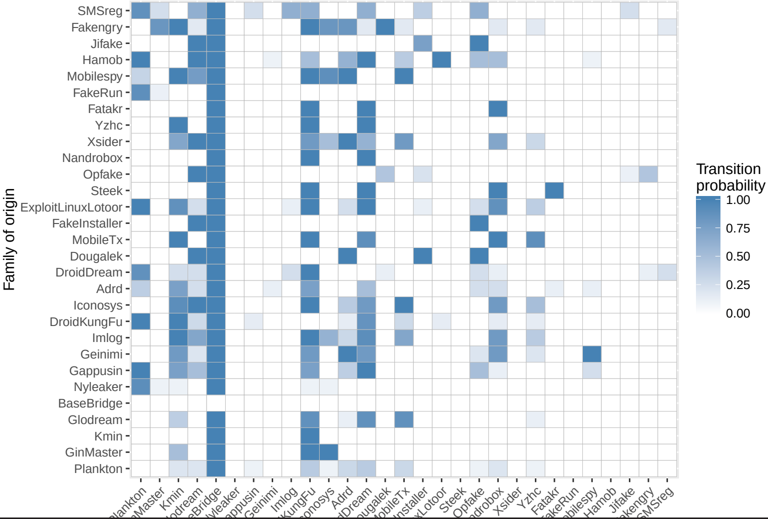
Main Achievements

1 generation to find a misclassification

From 50 to 450 queries per sample

1 mutation is enough

Transition Limits



Index

How machine learning works?

What is the ML influence in malware detection?

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

Are there any protections?

Countermeasures

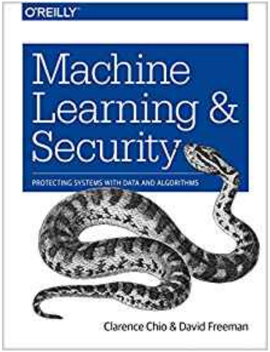
Construct a threat model before learning

Detect the attack and countermeasure it

Study the landscape and understand the gradient

Further Reading

Chio, C., & Freeman, D. (2018). Machine Learning and Security: Protecting Systems with Data and Algorithms. “OŔeilly Media, Inc.”.



Adversarial Machine Learning and its application to Malware

Héctor D. Menéndez

Department of Computer Science
University College London

5th April 2019