Machine Learning for cybersecurity... not only malware detection!

Machine Learning Course
A.Y. 2019-2020

Dr. Luca Massarelli
massarelli@diag.uniroma1.it

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

Outline

- Introduction
- Background
- Applying machine learning for a malware detection system.
- Creating a binary analysis tool with machine learning
- Homework

What is a malware?

A malware is a malicious software that fulfills the deliberately harmful intent of an attacker

Nikola Milosevic. "History of malware". In: CoRR abs/1302.5392 (2013).
URL: http://arxiv.org/abs/1302.5392.

Malware typical characteristics

Often a malware:

- Is designed to damage users or systems;
- Exploits Software and Hardware Vulnerabilities;
- Uses Social Engineering to trick users;
- Can install other malware;
- Is controlled by a command and control server;

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Beware of Malware!!!

19% of all cyber attacks are malware driven!

*(*SERT Quarterly Threat Report Q2 2016*)*

Globally, malware-based cyber attacks grew of 85%

during the 1° semester 2017 with respect to the 2° semester 2016

(CLUSIT Report 2017)

Malware analysis

"Malware analysis concerns the study of malicious samples with the aim of developing a deeper understanding about several aspects of malware"

- Malware behavior
- How they evolve in time
- How they intrude target systems
- …

Malware analysis

- Security defences are improving and evolving
- Nevertheless, malware are still succeeding

"Within the unceasing arm race
between malware developers and analysts,
each progress of security mechanisms
is likely to be promptly followed
by the realization of some evasion trick"

How is Malware Analysis performed?

- **Static Analysis**:
  - The malicious file is analyzed and using a disassembler the analyst is able to look at the binary code to understand what's going on.
- **Dynamic Analysis:**
  - The malware is executed in a controlled environment and its action on the system are registered and analyzed.

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Dynamic or Static Analysis?

- Each type of analysis have its benefit and both are necessary to analyze State of Art malware!
- In this seminar we will focus only on static analysis!

Malware analysis and the role of Machine Learning

Machine Learning can simply the analysis process in several ways:

- Automatic Malware Detection Systems;
- Automatic Malware Classification Systems;
- Tools to simplify Static Analysis;

Malware analysis and the role of Machine Learning

- Defense-side goal:
  produce defensive technologies
  as challenging as possible to overcome

➢ Need to capture malicious aspects and traits having the broadest scope

➢ **Machine Learning** is a natural choice to support such a process of knowledge extraction

Malware analysis and the role of Machine Learning

- Plentiful availability of labelled samples
  - ➢ Very large training set
  - ➢ Key element to foster the <u>adoption of machine learning for malware analysis</u>
- Many works in literature have taken this direction, with a variety of approaches, objectives and obtained results...

Outline

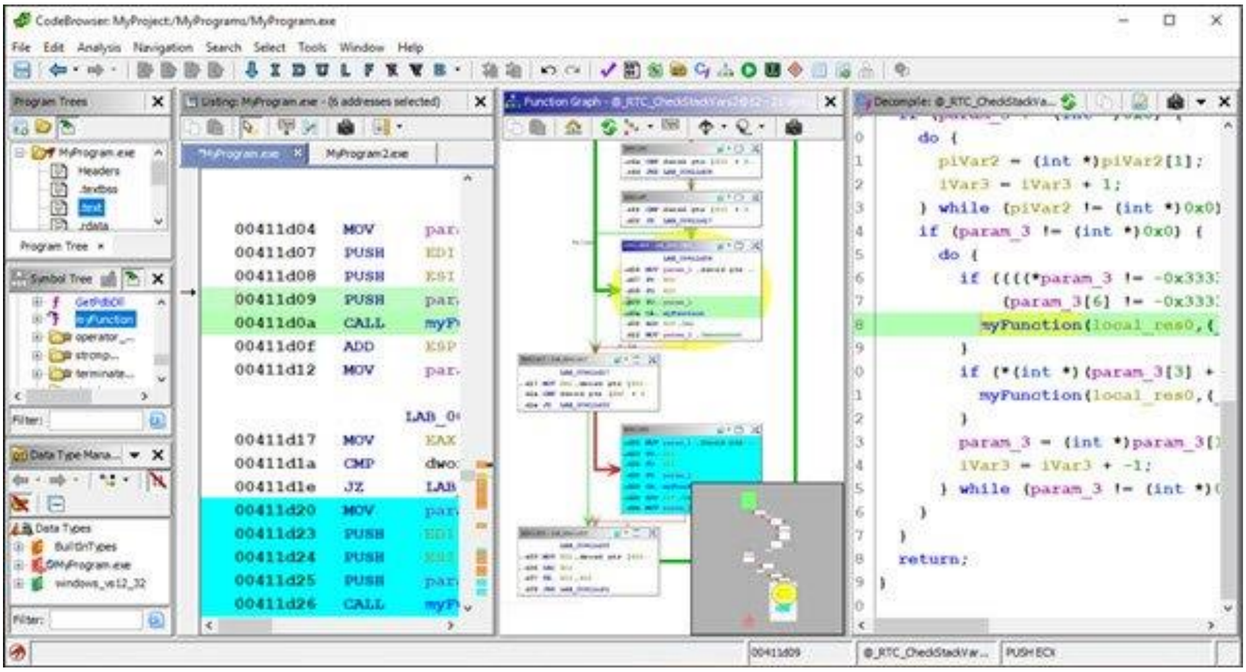
- Introduction
- Background
- Applying machine learning for a malware detection system.
- Creating a binary analysis tool with machine learning
- Homework

Reverse Engineering of a Malware

- When an analyst has a new malware sample to analyze the first thing to do is to disassemble it!

- Disassemblying permits to the analysist to look at the binary code of all the functions contained inside the malware!

How a disassembled malware looks like…



Reverse Engineering of a Malware

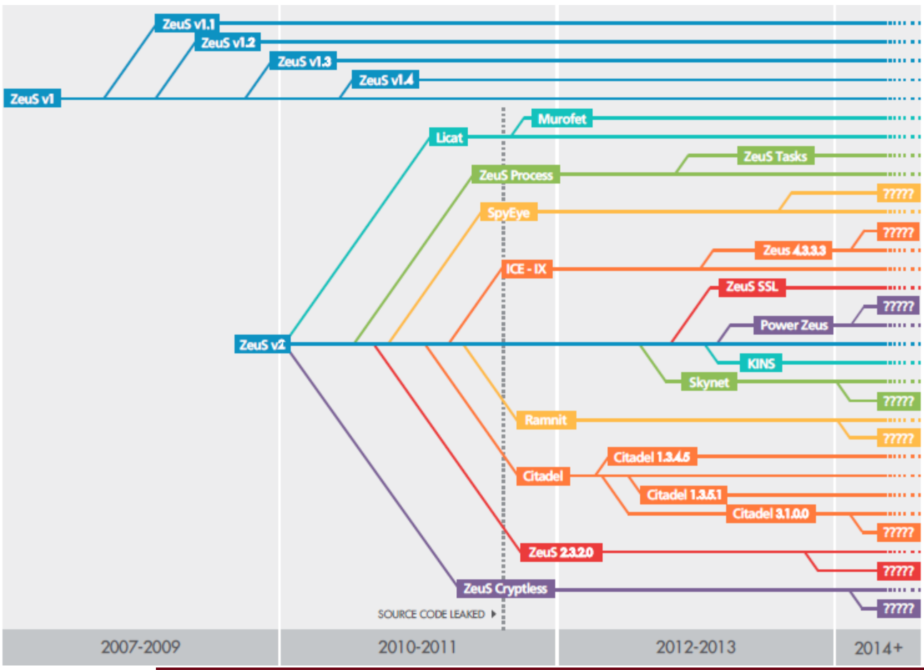
- The analyst cannot analyze all the code, he try to focus on specific part of the sample that for some reason looks more interesting:
  - System calls;
  - Strings;
  - Function Name;
  - Library function imports;
  - …

Not so simple…

- Malware binaries are often «stripped» removing function and variable name;
- The string inside a binary can be encrypted with a custom encryption algorithm;
- The libraries can be statically imported and can be indistinguishable from malware code.

Malware variants

*Malware developers produce **variants** to minimize the effort required to evade updated security defences*



An "original" malware evolves in time through the development of variants (es: Zeus)

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

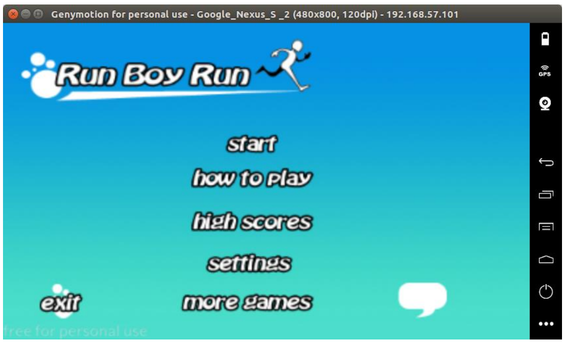
Malware family

*The set of variants deriving from the same malware strain (i.e., “original” sample) is a malware **family***

- Similar behavior
- Exploits same vulnerabilities
- Same malicious objectives

Malware Family (Android example)

Package Name: com.requiem.slingshakLite
Activities: com.requiem...

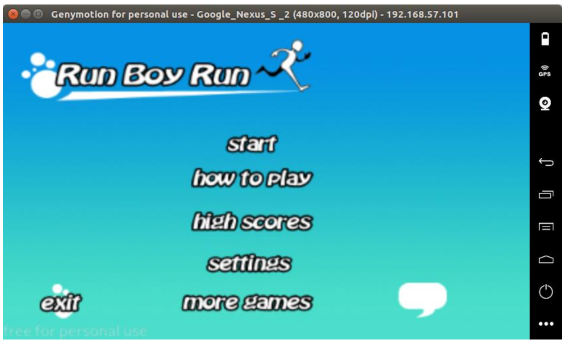


Package Name: ca.rivalstudios.runboyrun
Activities: ca.rivalstudios.runboyrun...

However….

Malware Family (Android example)

Package Name: com.requiem.slingshakLite
Activities: com.requiem...
Services: com.GoldDream.zj.zjService
Receivers: com.GoldDream.zj.zjReceiver
Certificate:
61ed377e85d386a8dfee6b864bd85b0bfaa5
af81
Relevant Strings:
*http: // lebar. gicp. net/ more. aspx? pid=
9944& amp; cid= 1000*



Package Name: ca.rivalstudios.runboyrun
Activities: ca.rivalstudios.runboyrun...
Services: com.GoldDream.zj.zjService
Receivers: com.GoldDream.zj.zjReceiver
Certificate:
61ed377e85d386a8dfee6b864bd85b0bfaa5
af81
Relevant Strings:
*http: // lebar. gicp. net/ more. aspx? pid=
9944& amp; cid= 1000*

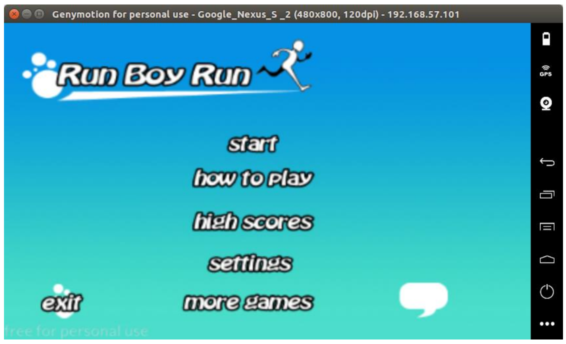
Obfuscation Techniques (Android example)

Obfuscation Techniques:

- Activity, Service and Receiver names can be changed and randomized;
- Applications can be signed with a different certificate;
- Binary code and application resources can be encrypted;

Not so simple…

Package Name: com.requiem.slingshakLite
Activities: com.requiem...
Services: com.requiem.se.1
Receivers: com.requiem.se.1
Certificate:
94fg474u34d296n8pjle9n060bi89n0brad5cf
41
Relevant Strings:
*EnCt2d5fcaad2bd889cb92be48ba0d67cc1e
886= cf70fbd5fcaad2bd889cb92be48ba0X=
GXQtvQ2gL*



Package Name: ca.rivalstudios.runboyrun
Activities: ca.rivalstudios.runboyrun...
Services: com.rivalstudios.a.1
Receivers: com.rivalstudio.b.2
Certificate:
61ed377e85d386a8dfee6b864bd85b0bfaa5
af81
Relevant Strings:
*http: // lebar. gicp. net/ more. aspx? pid=
9944& amp; cid= 1000*

Signature-based analysis approaches

- Need to recognize already-known samples
  - If I know a sample is malicious, I want to detect its replicas
- Common techniques are signature-based
  - Hash of portions of code
  - Pattern matching on specific segments
  - Generally based on static characteristics
- Obviously malware can evade these techniques with obfuscation!

Machine learning detection approaches

Machine learning permits to build malware analysis systems that:

- Not need human support.
- Are resilient to obfuscation techniques.

In general machine learning permits to create malware analysis systems based on the semantic of an application and not the code appearance.

Machine learning binary analysis approaches

Machine learning permits to build tools that can support the analyst during the analysis process:

- Identifying known functions;
- Provide useful information (like the compiler) to other tool;
- Predict names for functions;
- …

Outline

- Introduction
- Background
- Applying machine learning for a malware detection system.
- Creating a binary analysis tool with machine learning
- Homework

Applying Machine Learning to malware analysis

- Supervised learning
  - Need for labelled training set
  - Relevant example: classify unknown samples in known malware families
- Unsupervised learning
  - No need for labelled training set
  - Relevant example: cluster samples to identify families

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

An example: Malware Detection

- Example: malware detection
  - Given a file, establish whether it is a malware
  - Two main types of analysis (hybrids are possible)
    - Static analysis
    - Dynamic analysis
  - Can be seen as a binary classification

An example: Malware Detection

- The goal is finding a function MD having
  - The set F of all possible files as domain
  - The set {P,N} as codomain
    - Positive: the file is a malware
    - Negative: the file is not a malware

Given a specific file type (subset of F), how can we define MD?

An example: Malware Detection

Given a specific file type (subset of F), how can we define MD?

- Machine learning techniques provide means to find such a function
- Supervised learning allows to infer a function based on a labeled training dataset
- Given a function f to learn, with domain D and codomain C, the labelled training dataset (training set) is a set of pairs $\langle d, f(d) \rangle$, where $d \in D$ and $f(d) \in C$

An example: Malware Detection

In practice, supervised learning enables the learning of a function by providing a certain number of instances, each showing the expected output of the function given a specific input

An example: Malware Detection

- Several algorithms/models for supervised learning
  - Artificial neural networks
  - Decision trees, random forest
  - Support vector machines
  - Nearest neighbor
  - …
- And several tools implementing them
  - Weka (www.cs.waikato.ac.nz/ml/weka)
  - Encog (www.heatonresearch.com/encog)
  - Sklearn
  - …

An example: Malware Detection

- Instances of a domain can be complex
  - Android Application package
  - Huge execution trace of an application
  - Network traffic log of an application
- What is the actual input of the function to learn?
  - Each element can be represented by a fixed set of **features** (attributes) $\{a_1, \ldots, a_n\}$ aimed at capturing all and only the characteristics that are relevant for the function to learn
  - Feature extraction is the process that, given an instance, returns its values for these features

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

An example: Malware Detection

- How to choose the set of features?
- What are the key characteristics for the function to learn?
- What are the specific cause-effect relationships that hold in that particular context?

This is where the intuition comes into play…

Static features:

- Features that can be extracted only by looking at the apk:
  - Components (*Activities, Services, Content Providers and broadcast receivers*);
  - Permissions;
  - API calls;
  - Strings;
  - Flow graph;

Evaluation Metrics

- Example: malware detection - accuracy metrics
  - Need to compare against some «ground truth»
  - Usually corresponds to the test set
  - For binary classification, there are four cases to be considered:

		Learned Function Output	
		Positive	Negative
Ground Truth	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Evaluation Metrics

- Example: malware detection - accuracy metrics
  - Meaning of true/false positive/negative for malware detection
  - True Positive
    It is a malware, and I correctly detected it
  - False Positive
    It is not a malware, but I thought it was
  - True Negative
    It is not a malware, and I thought so too
  - False Negative
    It is a malware, but I didn't detect it

Evaluation Metrics

- Example: malware detection - accuracy metrics
  - Precision: TP / (TP + FP)
    - How many files are real malware (TP) among those I considered as malware (TP + FP)?
    - «if I say it is a malware, then it really is a malware» (i.e., very few FP)
  - Recall: TP / (TP + FN)
    - How many malware did I spot (TP) among those in the test set (TP + FN)?
    - «if it is a malware, then I spot it» (i.e., very few FN)

Evaluation Metrics

- Example: malware detection - accuracy metrics
  - False Positive Rate: FP / (FP + TN)
    - How many files did I wrongly consider as malware (FP) among all the benign files (FP + TN)?
  - Accuracy: (TP + TN) / (TP + FN + TN + FP)
    - How many files did I classify correctly?
  - F-measure
    - $2 \cdot (precision \cdot recall)/(precision + recall)$
    - Can be interpreted as a weighted average of precision and recall
    - Best value: 1 worst value: 0

Outline

- **Introduction**
- **Background**
- **Applying machine learning for a malware detection system.**
- **Creating a binary analysis tool with machine learning**
- **Homework**

Identifying the problem

- There are several problems in binary analysis where machine learning can help:
  - Binary Similarity;
  - **Compiler Provenance;**
  - Function Naming;

Compiler Provenance Problem

- Given the binary code of a function, we want to identify the **compiler** and the **optimization** who produced it.

- Two Classical Classification problems.

- Why compiler provenance:
  – More accurate disassembly;
  – Once we know the compiler we can try to match library functions;

Machine Learning seems a good solution

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Feature Engineering

- For the right choice of feature we can exploit some domain knoledge:
  - The order matter!
  - Often the compiler insert some code at the beginning and at the end of a function.
  - Each assembly instruction is made up by a mnemonic (mov, add, sub, push, …) and some arguments (from 0 to 2). The set of all instructions is unbounded but if we consider only mnemonics the set is small…

Possible features

- As a first attempt we can try to group instruction semantically and represent each function as a vector where in each position we count how many instruction belong to a particular category…
- We are not considering the order…

Type	Attribute name
Block-level attributes	String Constants
	Numeric Constants
	No. of Transfer Instructions
	No. of Calls
	No. of Instructions
	No. of Arithmetic Instructions
Inter-block attributes	No. of offspring
	Betweenness

Table 1: Basic-block attributes

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

Possible features

- We can also try to represent each binary function as a vector of integer, where at position i we encode with an integer the mnmenonics of the i^{th} instruction.

Classification

- As classification algorithm we can try different options:
  - Naive Bayes
  - Linear SVM
  - SVM with RBF kernel
  - …

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Classification Metrics

- For the classification of optimization we have a binary classifier:
  - Optimization HIGH
  - Optimization LOW

- We evaluate the classifier with standard accuracy metrics.

Classification Metrics

- For the compiler classification strategy we have a multi-class classification problem.
- We need to compute the overall accuracy and precision / recall for each class!

Outline

- Introduction
- Background
- Applying machine learning to malware analysis
- Creating a binary analysis tool with machine learning
- Homework

HOMEWORK

- For the homework you have to solve the compiler provenance problems.

- **TASKS**:
  - Build a **binary classifier** that can predict if a function has been compiled with optimization HIGH or LOW.
  - Build a **multiclass classifier** that can predict the compiler used to compile a specific function.

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Dataset

- The dataset contains 30000 functions compiled with 3 different compilers: gcc, icc , clang.
- The compiler distribution is very balanced: 10000 functions per compiler.
- The optimizations distribution is not balanced.
- For each compiler we used different versions.

Dataset Format

- The Dataset is provided as a jsonl file.
- Each row of the file is a json object with the following keys:
  - **instructions**: the assembly instruction for the function.
  - **opt**: the ground truth label for optimization (H, L)
  - **compiler**: the ground truth label for compiler (icc, clang, gcc)

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Dataset Format

{

"instructions": ["xor edx edx", "cmp rdi rsi", "mov eax 0xffffffff", "seta dl", "cmovae eax edx", "ret"],

"opt": "H",

"compiler": "gcc "

}

How to split instructions

- The value under the key instruction is a json list.

- If you want to consider only the mnemonic of each instruction you can just split each element of the list by blank space and consider only the first word.

CIS Sapienza
Research Center for Cyber Intelligence and Information Security

Blind Test Set

- You will be also given a blind test set.
- The blind test set does not contain the label for the function.
- You have to submit with your report a csv file where in each row you put the prediction of your solution for each row in the test set.

Blind Test Set

- Each row should be like that:

  <compiler>, <opt>

- We will evaluate the accuracy of your solution on this file.

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

Questions?

massarelli@diag.uniroma1.it