Sommario

[Introduction 1](#_Toc529823509)

[Drebin Dataset 1](#_Toc529823510)

# Introduction

In this report I’ll describe the implementation of an Android malware detector and classifier based on machine learning techniques. In particular I’ll use Supervised Learning methods, where the ground truth is provided by a huge open dataset called “Drebin” containing features extracted by malwares and benign programs collected through both official and unofficial application stores.

The problem of malware detection is a binary classification problem, where the two classes in which programs must be divided into are the positive class, denoted with the symbol “+” and the negative class, denoted with “-”. The former will contain malwares and the latter benign programs.

The second problem is a different classification one, in which malwares must be divided into classes according to the “family” they belong to, where a family is a set of malwares with the same behavior that usually have similar features.

In this report I’ll describe the techniques used to solve both these problems and the relative implementation in Python.

# Drebin Dataset

Drebin dataset is an open dataset containing features from 123,453 benign applications and 5560 malwares. The dataset is structured as follows: the root folder contains a subfolder “features\_vector” and a csv file “sha\_families” containing couples «sha , family» where “sha” is the sha-256 hash of the malware and “family” is the family the malware belongs to. The subfolder contains a txt file for each programs, named with the hash of the programs that the file is related to. Each file contains a line for each feature of the program. Features can be divided into two categories: static and dynamic.

Static features are string extracted by the Manifest file, that provides general information on the application, static features can be divided in turn in four subcategories: hardware components, requested permissions, app components and filtered intents.

Dynamic features are obtained by disassembling the executable of the applications and they include all APIs calls in the applications and all the strings (including network addresses).