Guy Azoulay -207709734
Tal Malchi - 208278556

Second Assignment- Weak spots identification

About our classifier-

The Drebin classifier is a machine learning algorithm that is used to identify malicious Android apps. It uses static analysis techniques to extract features from the app's code and analyze them for signs of malicious behavior. These features include the app's permissions, network connections, and code obfuscation. The classifier then uses these features to make a prediction about the app's likelihood of being malicious. If the prediction is high, the app is flagged as potentially harmful and the user is alerted.

Weak Spots in our classifier-

The Drebin classifier has several weaknesses that can be exploited by attackers. These weaknesses include its use in static analysis techniques, its limited ability to detect new forms of malware, and its sensitivity to evasion techniques. These weaknesses can harm the classifier's ability to accurately identify malicious apps and protect users from false detection.

1. Static analysis only: One weakness of the Drebin classifier is that it only uses static analysis techniques to extract features from the app's code. This means that it cannot detect malware that only becomes active after the app has been installed and launched on a device.

2. Limited to known malware: Another weakness of the Drebin classifier is that it is only trained on known forms of malware. This means that it may not be able to detect new or previously unseen forms of malware.

3. False positives: Drebin may mistakenly classify benign apps as malicious due to the presence of certain permissions or APIs that are also used by malicious apps. ->3.3.2

Guy Azoulay -207709734
Tal Malchi - 208278556

How did we find those weak spots?

1. Static analysis only- This type of weakness was obvious,
   as it written in our article, our Drebin classifier uses static information only:
   - Was found in the introduction section, page number 2: --

> We need to note here that DREBIN builds on concepts of static analysis and thus cannot rule out the presence of obfuscated or dynamically loaded malware on mobile devices. We specifically discuss this limitation of our approach in Section 4. Due to the broad analysis of features however, our method raises the bar for attackers to infect smartphones with malicious applications and strengthens the security of the Android platform, as demonstrated in our evaluation.

2. Limited to known malware: This type of weakness is common to many other classifiers.
   Most of the classifiers are using training set which with it they can refer between benign apps and malicious apps.
   In case that our classifier will face with a new malware it may not be able to identify rather this malware is benign or malicious.

3. False positives: This type of weakness was also found in the article,
   as you can see in the paragraph below, false detection of benign apps as malicious apps can occur by Drebin if we uses samples of the “Gappusin family”.
   -- Was found in 3.3.2 “False and missing detection” paragraph --

Guy Azoulay -207709734
Tal Malchi - 208278556

3.3.2 False and missing detections

We finally examine benign applications which are wrongly classified as malware by DREBIN. Similar to malicious applications, most of these samples use SMS functionality and access sensitive data, which is reflected in high weights of the corresponding features. Moreover, these samples often show only very little benign behavior and thus trigger false alarms. Fortunately, the ability of DREBIN to output explainable results can help the user to decide whether a suspicious looking functionality is indeed malicious or needed for the intented purpose of the application.

A similar situation occurs when DREBIN classifies samples of the Gappusin family [19] as benign. Although it is in many cases possible to extract features which match the description of the Gappusin family—amongst others the hostname of the external server—there are too few malicious features to identify the samples as malware. Gappusin mainly acts as a downloader for further malicious applications and thus does not exhibit common malicious functionality, such as theft of sensitive data.

Guy Azoulay -207709734
Tal Malchi - 208278556

How we intend to exploit those features?

1. Static analysis only-
   We have few ways to exploit this weak spot-

   - Code obfuscation techniques - such as code shifting or control flow flattening. Those procedures will make it more difficult for the classifier to detect patterns that are commonly associated with malware.

   - Transformation attack- is a type of attack that involves modifying the appearance or behavior of malware to evade detection by a security system. Transformation attacks can take many forms, including modifying the code of the malware, packing the malware to obscure its features, or changing the way the malware communicates with its command and control servers.
   Our classifier can be vulnerable to this attack because it relies on the presence of certain static features in an app in order to classify it as malicious. If we will be able to modify those features or ambiguous them in some way, the classifier may not be able to detect them as malware application

   - Mimicry attacks- is a type of attack that involves creating malware that is designed to mimic the features of benign software, in order to evade detection by the security system.
   This can be done by modifying the code of the malware or by adding benign features to it.
   Our classifier can be vulnerable to this attack because it relies on the presence of certain static features in an app in order to classify it as malicious. If we will be able to create malware that mimics the features of benign software, the Drebin classifier may not be able to accurately classify the malware.

Guy Azoulay -207709734
Tal Malchi - 208278556

2. Limited to known malware- This approach is easy to achieve because our Drebin classifier based all its classification abilities according to it train data in case we will upload new malicious app that has features that our classifier never seen, it can classify it wrongly.

3. False positive- We don’t manage to find a way to exploit this type of weakness.