# CS97 Project Proposal – Reverse Engineering Android Malware using Static and Dynamic analysis tools

Luis Ramirez and Gabriel Khaselev

## 1 Abstract

The goal of this project is to reverse engineer the prevalent android malware NotCompatible. We attained samples of this malware from researchers at various universities. The malware was first noticed in the spring of 2012 but made a recurrence resently in 2013. The samples we are trying to reverse engineer are from the second round of malicious activity. By Using static and dynamic analysis to deconstruct the malware we hope to determine its method of action and how it maintains minimal noticablity on the infected android device. Additionally we hope to determine how the application determines whether it has been installed on a physical device or emulator. We suspect that the detectability of the malware is directly linked to which android OS permissions and processes NotCompatible uses.

## 2 Motivation

A wider selection of devices and cheaper entry points contribute to the Android operating system dominating the consumer smartphone market. Smartphones are used as collection bins for personal information, clearly users do not want this information shared over the internet. With 79.3% of the smart phone market share dominated by android, this malware has the potential to infect a very high number of users. Recent statistics show that NotCompatible has already infected upwards of 300,000 devices. We predict that android malware will become more and more prevalent as the computing paradigm shifts towards mobile devices.

## 3 Background

NotCompatible is known as Drive-By malware, it infects devices when the user visits a web page that is hosting it. The web page informs the user that their current device is not compatible with the site, and that they have the update to fix the problem. If the user has side-loading enabled (an option that allows users to install applications from sources other than the official android market) and they accept the "update", the malware infects their device. Once on the device, NotCompatible silently overwrites web pages on reputable sites with weight-loss pill advertisements. The malware also peruses the user's contact and email information, most likely uploading the information to the comand and control (CNC) server. NotCompatible can use this information autonomously as well, and sends an email with a link to the malicious web site to every contact.

## 4 Idea

NotCompatible does not display any of these behaviors when run in the Android Virtual Device. The software package installs, and the service remains dormant when not run on a physical device. There are a suite of tools for analyzing and reverse engineering android apps and malware. The official android SDK has many tools such as adb that provide an debugging interface that can bind onto a running process and give instruction-level resolution. Android-apktool will decompile an android apk package into manifest xml files and smali files. Manifest files describe the permissions that the app will use, while smali files give an overview of methods defined. Copper Droid is an emulation environment designed specifically to probe android malware to discover the entire range of behaviors it can show.

## 5 Timeline

Because of the nature of reverse engineering malware our milestones will be a bit fast and loose. Hopefully we make significant progress continuously and can push our efforts further and further. But we may discover as we begin to decontruct NotCompatible that it has much more functionality than expected and may be more challenging to reverse engineer than expected. The following milestones are general goals but will most certainly need to be reevaulated as we go about the project and discover more or less than we expected.

- MileStone 1: Unpack the .apk and extract relevant information from the android manifest file. We will use this information to run the malware in the Copper Droid emulation infrastructure. Additionally we will try to infect a physical device and observe the effects that NotCompatible has on the device.

- MileStone 2: Mostly TBD. If the application can run in Copper Droid or other dynamic analysis frameworks we can try to determine the system calls made by the malware, how it determines whether it is being run in an emulator, and how it maintains operation when the user tries to delete it.

- MileStone 3: Completely TBD. If reverse engineering of the application goes well we will try to determine the effectiveness of this kind of malware and analyse its prevalence on devices and how effective it really is. This final stage is basically a reflection on the state of android malware and in particular the role that NotCompatible plays in the android world.

## 6 References

### 6.1 Frameworks

- android-apktool A program to decompile .apks
- Stack Overflow Guide on getting a source code from an apk file
- Copper Droid Android Dynamic analysis framework
- DroidBox Another Android dynamic analysis framework

### 6.2 Articles

- Felix Matenaar on reverse engineering Android Apps
- So You want to reverse engineer an APK
- Infosec Android malware analysis resources
- Remnux and Mobisec Behavioral android analysis