Catch Me If You Can: Evaluating Android Anti-Malware Against Transformation Attacks

ABSTRACT:

Mobile malware threats (e.g., on Android) have recently become a real concern. In this paper, we evaluate the state-of-the-art commercial mobile anti-malware products for Android and test how resistant they are against various common obfuscation techniques (even with known malware). Such an evaluation is important for not only measuring the available defense against mobile malware threats, but also proposing effective, next-generation solutions. We developed DroidChameleon, a systematic framework with various transformation techniques, and used it for our study. Our results on 10 popular commercial anti-malware applications for Android are worrisome: none of these tools is resistant against common malware transformation techniques. In addition, a majority of them can be trivially defeated by applying slight transformation over known malware with little effort for malware authors. Finally, in light of our results, we propose possible remedies for improving the current state of malware detection on mobile devices.

EXISTING SYSTEM:

Existing anti-malware software, studied the robustness of anti-malware against Android malware recently using a tool called ADAM. ADAM implements only a few transformations, renaming methods, introducing junk methods, code reordering, and string encoding, in addition to repacking and assembling/disassembling.

PROBLEM DEFINITION:

ADAM implements only a few transformations, renaming methods, introducing junk methods, code reordering, and string encoding, in addition to repacking and assembling/disassembling.

ADAM is not always able to evade an anti-malware tool.

Obfuscation resilient detection is based on semantics rather than syntax.

PROPOSED SYSTEM:

In this paper, we aim to evaluate the efficacy of anti-malware tools on Android in the face of various evasion techniques. For example, polymorphism is used to evade detection tools by
transforming a malware in different forms (“morphs”) but with the same code. Metamorphism is another common technique that can mutate code so that it no longer remains the same but still has the same behaviour. For ease of presentation, we use the term polymorphism in this paper to represent both obfuscation techniques. In addition, we use the term ‘transformation’ broadly, to refer to various polymorphic or metamorphic changes.

ADVANTAGES OF PROPOSED SYSTEM:

- Our set of transformations is much more comprehensive and includes renaming packages, classes, encoding array data, inserting junk statements, encrypting payloads and native exploits, reflection, and bytecode encryption as well.
- Our framework is comprehensive, aimed towards complete evasion of all anti-malware tools. We believe our results make a clear statement – all anti-malware tools can be evaded using common obfuscation techniques. Unlike ADAM, our result is able to highlight the severity of the problem and is easily accessible.
- Many works have been done towards discovery and characterization of smartphone malware. Our work is distinct from these as we try to evaluate the efficacy of existing tools against transformed malware.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 44 Mb.
- Monitor : 15 VGA Colour.
- Ram : 512 Mb.
- MOBILE : ANDROID
SOFTWARE REQUIREMENTS:

- Operating system : Windows XP/7.
- Coding Language : Java 1.7
- Tool Kit : Android 2.3 ABOVE
- IDE : Eclipse

REFERENCE: