Song Yang (sy540)
Xin Yang (xy213)
Zhuohang Li (zl299)

Paper Review

WindowGuard: Systematic Protection of GUI Security in Android

Problem statement

The number of malware that are using graphic user interface (GUI) systems to attack Android devices is increasing rapidly. Android GUI attack refers to harmful behavior that attempts to adversely affect the integrity and availability of GUIs belong to other applications in order to achieve malicious purposes, including launching unwanted windows, tapjacking, taskjacking, etc.

Motivation

Android GUI attacks are real threats and can cause severe consequences, such as sensitive user information leak, user device denial of service, etc. Given the serious and rapid growth of GUI attacks, there is a pressing need for a comprehensive defense solution. However existing defense methods are all lack of defense coverage, effectiveness, and practicality.

Solution approach

Android window integrity (AWI) model clearly designates user session and continuously checks for window integrity to prevent the attacker from taking over the user's screen. The AWI model is implemented in WindowGuard which is developed as a Xposed module to be usable to a larger number of users.

Pros and Cons of the paper

- **Pros:**

1. It is very difficult for previous static-analysis-based defense solution to distinguish from the usage of the same API from a malware or from an app that is useful to the user. However by enforcing clear designation of user session and legitimacy of GUI system, AWI model is able to make normal user session less vulnerable, and therefore detect malware with a much higher while preserving the original user experience.
2. The AWI model provides a comprehensive protection for the user. It works for all known GUI attacks, including malicious screen locker, GUI confusion attacks, tapjacking attacks, etc.

3. WindowGuard, the implementation of AWI model, compared with existing solutions which require modification on both apps and systems, is more practical and easier to deploy on a massive number of user devices.
4. Once deployed, WindowGuard can work systematically in the background and do not require user involvement until a malicious behavior is detected. WindowGuard does not have any impact on most apps. As for the 1.03% of apps that trigger security alert, only a single tap is needed from the user to make a decision for that app and will not distract the user from that on. Compared with existing solutions which continuously require user's attention, this method does not bring much better user experience but is also bound to deliver higher accuracy instead of relying on user's experience to make judgments.

- **Cons:**

1. It's based on Xposed framework, so the device has to be rooted first, which gives full permission to all apps, and exposes the system to a more dangerous environment.
2. For new Android devices, warranties will be invalid after rooting and installing the Xposed framework.
3. Also, this paper failed to clarify how many malicious attacks can happen even when the device is not rooted. Referring to the consequences of rooting, there should be a lot of attacks that can be avoided if they are not given enough permission. It's better to detect the malicious behaviors without rooting your device and develop another version for rooted users.
4. This paper is tested on Google Nexus devices, which is a pure Android system, and the whitelist includes several system apps and works fine. But when it comes to third-party devices e.g. Samsung or Sony, more manufactures pre-installed apps need to be added to the whitelist like Samsung Pay or Xperia Home launcher, which needs much more efforts for the long-term maintenance. For the system apps, they should extract some common features so that devices from other manufacturers can easily adapt to.
5. WindowGuard only steps in when malicious attack affects the integrity and accessibility of the GUI of other applications. Therefore it will not work on spoofing malware that misleads user to launch itself. On the other hand, WindowGuard also places demands on app developers to follow AWI security principles. Otherwise, the functionality of such apps could be affected on deployed devices.

Potential Improvement Point

Since WindowGuard still requires judgment from users, we think it could be of practical use to interfacing WindowGuard with malware analysis websites. For example, a single prompt of the rate of an app marked as malware by the community can largely help users to make the decision. Conversely, information provided by the user can help to increase malware detection rate on a large scale of devices.