App Protector SDK

Introduction

App Protector SDK enables limited threat detection on mobile devices. The detections and reactions are related for both implementations, App Protector Offline and App Protector Online modes. The difference is that App Protector Online has a connection to the Portal backend that enables advanced functionalities.

Supported detections are:

- Jailbreaking detection for iOS devices;
- Rooting detection for Android devices;
- Debugging detection;
- Emulator detection;
- Hooking detection;
- Screen recording detection for iOS devices;

Supported reactions to detected threats are:

- Notify the user;
- Fake produced values;
- Terminate application;

Supported detections and reactions are described below.

Security detections

Jaibreak detection

Jailbreaking is the process of removing the limitations put in place by a device's manufacturer. Jailbreaking is generally performed on Apple iOS devices, such as the iPhone or iPad. Jailbreaking removes the restrictions Apple puts in place, allowing you to install third-party software from outside the app store. Some people may have the perception that jailbreaking is only used for piracy, but this is not the case – jailbreaking allows you to do things like change your iPhone's default browser and mail client. Essentially, jailbreaking allows you to use software that Apple doesn't approve of. Also, it removes restrictions in inter-application communications, accessing files of other applications, and gives the user root access.

App Protector detects if the device is jailbroken and also has a large "blacklist" of unwanted libraries and checks if some library from the "blacklist" exists on the client's device. Furthermore, App Protector checks if the client's device has access permission to suspicious directories (directories to which the client does not have access permission by default).

Root detection

Rooting is the process of removing the limitations put in place by a device's manufacturer and gaining superuser access to device resources as on Linux. Rooting is generally performed on Android devices. The process is usually done in order to overcome limitations that the carriers put on mobile devices. Thus, rooting gives permission to alter or replace system applications and settings. It also gives the user permission to execute privileged commands which are not available to the device in stock configuration. It is also required in order for more advanced and potentially dangerous operations including modifying or deleting system files, removing preinstalled applications, and low-level access to the hardware itself.

App Protector will detect whether any of the known root packages exist on the client device and whether the kernel that is installed was signed with non-release keys while it was compiled. Besides that, App Protector will detect whether root privileges, which are not available on a non-rooted device, are available on the client device.

Debugging detection

Debugging detection is "the implementation of one or more techniques within computer code that hinders attempts at reverse engineering or debugging a target process". It is actively used by recognized publishers in copy-protection schemas, but is also used by malware to complicate its detection and elimination. Techniques used in debugging detection include:

- Exception-based: check to see if exceptions are interfered with;
- Process and thread blocks: check whether process and thread blocks have been manipulated;
- Timing and latency: check the time taken for the execution of instructions;
- Detecting and penalizing the debugger;

Emulator/Simulator detection

Emulator/Simulator detection is typically used by Security companies to analyze unknown malware samples using simulated system environments (such as virtual machines or emulators). The reason is that these environments ease the analysis process and provide more control over executing processes.

The goal of malware authors is to make the analysis process as difficult as possible. They can equip their malware programs with checks that detect whether their code is executing in a virtual environment, and if so, adjust the program’s behavior accordingly. In fact, many current

malware programs already use routines to determine whether they are running in a virtualizer such as VMware.

The general belief is that system emulators (such as Qemu) are more difficult to detect than traditional virtual machines (such as VMware) because they handle all instructions in software. One must analyze a number of possibilities to detect system emulators. Results show that emulation can be successfully detected, mainly because the task of perfectly emulating real hardware is complex. Furthermore, some tests also indicate that novel technology that provides hardware support for virtualization (such as Intel Virtualization Technology) may not be as undetectable as previously thought.

Hook detection

Hooking covers a wide range of techniques used to change, alter and augment the behavior of an application or other software components by intercepting function calls, messages, or events passed between software components. The code that handles such intercepted function calls, events, or messages is called a hook. With hooking, an attacker can potentially intercept, monitor, and change sensitive proprietary user data. Usually, hooks are inserted while the software is already running, but hooking is a tactic that can also be employed prior to the application being started. Both these techniques are described in greater detail below:

- Source modification: by modifying the source of the executable or library before an application is running, through techniques of reverse engineering, hooking can also be achieved
- Runtime modification: operating systems and software may provide the means to easily insert event hooks at application runtime

For iOS platforms, App Protector can detect when a dynamic library has been injected. Having malicious dylibs loaded can have a severe security impact since they allow run-time patches to system functions.

Both, iOS and Android platforms App Protector, can detect dynamic code instrumentation toolkit, Frida. With Frida, an attacker is able to monitor and modify the runtime code execution of the application by injecting snippets of JavaScript or some native library in the application. Frida also allows tracking of memory, registers, and threads of a target application, it helps in reverse engineering and can expose internal functions and data structures. All of the above is potentially dangerous since it allows bypassing security (or other) features by easily hooking on a method and changing it in runtime.

Even though hooking can be performed differently on each platform, there is a "blacklist" of unwanted loaded libraries which are detected on each system. On iOS, loaded libraries containing the following words in their name are checked:

- Frida;
- Gadget;

- Substrate;
- Cycript;

On Android, there is detection if loaded libraries containing the word "Frida" in their name exist.

Frida servers can also be connected on an open port. The prerequisite for that is jailbroken/rooted device. On the iOS platform App Protector checks if Frida is connected on the port to which it connects by default. On the Android platform, App Protector checks all open ports.

Screen recording detection

Screen recording (iOS only) can be used to acquire proprietary user data that can later be used for malicious purposes. Getting hold of passwords, IDs, serial numbers, and other valuable user data is dangerous and such activities as screen recording and screen captures should always be monitored for possible mischief.

Security detections reactions

App Protector can be configured in two dimensions where one dimension is what should be detected and the second one is what should be done in case an attack has been detected. For every detection, the different reactions can be configured.

This behavior also depends on the implementation type. If App Protector SDK offline is integrated within the application, the behavior needs to be defined on the application level, i.e., to show a notification or to terminate the application.

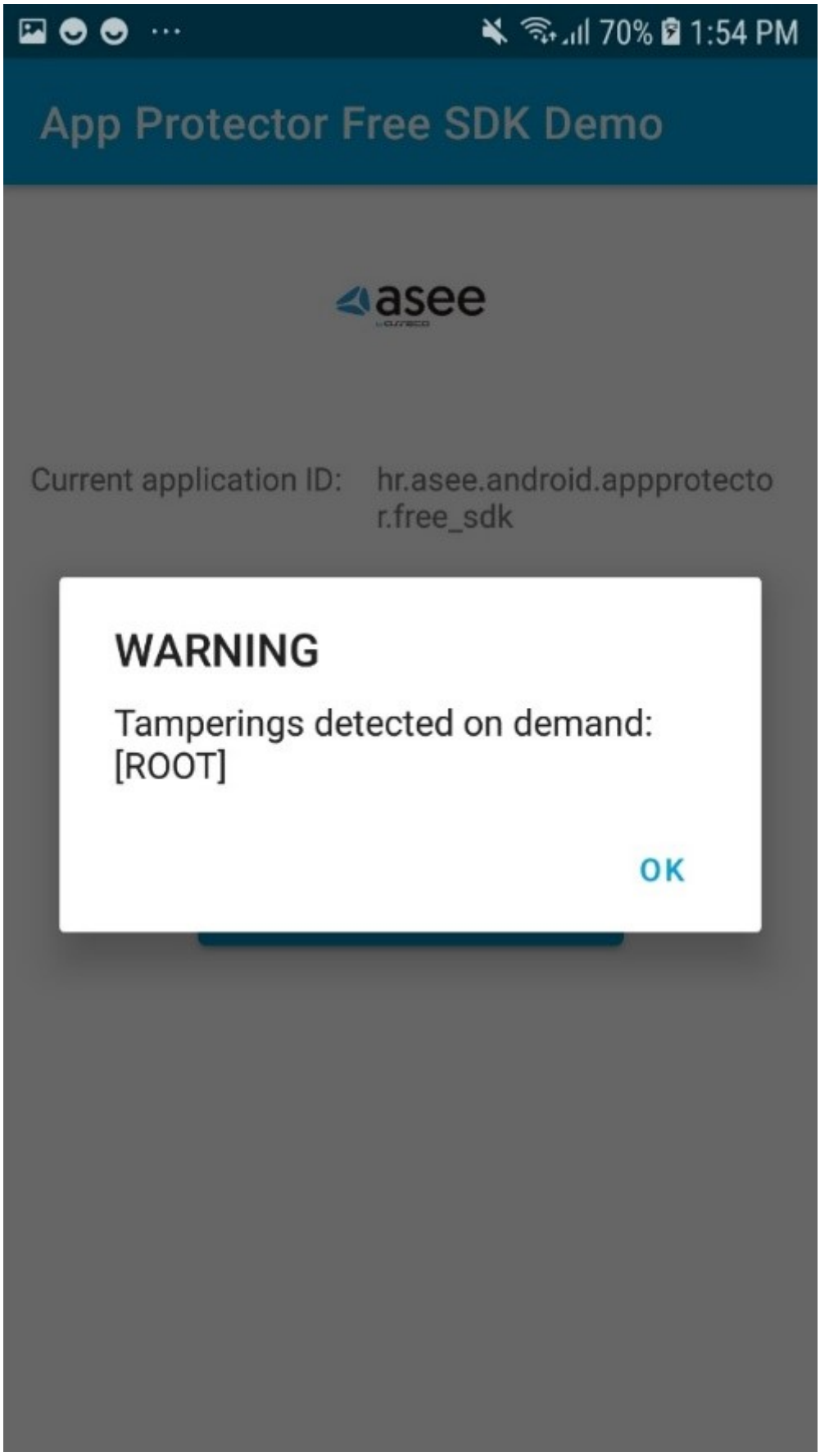
If App Protector SDK online is implemented, these options can be dynamically defined, based on the decision of the security department using the App Protector Portal Web Application. For example, if some kind of exploit is found on operating systems, through App Protector Portal it is possible to change the behavior of the application to be terminated instead of just showing a notification to the end user.

Fake produced values

Producing fake values is used in cases when you don't want to give information to the external system (application, device, user...) when some kind of security event is detected. In this case, App Protector returns fake values (For example, replace sensitive information with dummy values) to the mobile application and the mobile application shows it to the user. Attacker devices don’t see any different behavior, but all data that is visible to the attacker is incorrect. So, if the attacker intended to gather the user's sensitive information, all the collected data won't be useful.

Notify the user

When a security event is detected, App Protector returns information about the detected problem on the device – to the application that implements it. This information can be shown to the user. This reaction is on the information level, only reporting the detected event. These events can be handled on the application level, i.e., for root detection, the user can be just informed about the possible security issue, or if the security/risk department defines that this kind of device cannot use the application, the application can be terminated.



Notify the user

Terminate application

For detected security events that are defined as high risk, it is possible to configure the App Protector to terminate the application. This is the most radical behavior that can be executed, preventing the running of the application. This behavior can act like an application crash, so it would be recommended that the user is also informed about the application closing due to a detected threat.