Coordinated Disclosure of Vulnerabilities

in KASPERSKY INTERNET SECURITY Android Antivirus Free version 11.12.4.1622

1 Executive summary

Researchers of MRG Effitas tested the Kaspersky Internet Security Android application. During use, we came across implementation details, which might undermine the Vendor's efforts to provide a comprehensive mobile security solution with the potential to aid users in case of encounters with malware.

Testing covered the following application version.

Application name	Kaspersky Internet Security
Store URL	https://play.google.com/store/apps/details?id=com.kms.free
Version	11.12.4.1622

We considered the situation and opted for a responsible disclosure approach to aid the Vendor in their efforts. In accordance with industry standards, we disclose the issues based on Google's 90-day policy. As a result, after a 90-day plus a 14-day grace period, we make the discoveries public. For further details, refer to the following URL.

Disclosure date: 07. April 17
Grace period begins: 06 July, 2017
Grace period ends: 20 July, 2017

https://security.googleblog.com/2014/07/announcing-project-zero.html

2 Vulnerability description

2.1 No user warning on rooted/compromised devices

Finding

Testing discovered that the application was normally installed and initialized on a rooted test device, provided with a dynamic hooking framework application (xposed framework). This scenario is a highly insecure one, as the hooking framework can be utilized to bypass every security measure of the application. This is by no means something that an Android security suite can be properly prepared for, nevertheless no user warning has been displayed.

Risk

As a result, a highly sophisticated piece of malware using dynamic hooking might even abuse the application to convince the user that there are no threats on the device. Note that no cloaking has taken place and as per the logs, the Xposed installer application has even been scanned by the application.

Consider the following scenario.

1. The Android device with no preliminary protection, gets infected by a highly sophisticated piece of malware.
2. The users suspects that there might be something going on and opts for the Kaspersky Internet Security suite to run a security scan on his device.
3. Upon installation and scanning, no warning is displayed and the user is reassured that his device is clean from malware.

Recommendation

It is recommended to at least warn the user in such clear cases that there might be something going on with his device and that the scan results are not trustworthy in any way.

Nevertheless, it is important to point out that reliable root detection is theoretically impossible in untrustworthy environments (such as a user controlled Android device). When it comes to root detection, false positive errors are rather uncommon, however false negatives are more dangerous from security perspective. It is possible to install readily prepared tools to fool common root detection methods (e.g. RootCloak), however, presence of the Xposed installer and the RootCloak application itself is a good indication that the device has been rooted. All in all, a comprehensive approach is recommended, where the user is notified at the slightest sign of deviation from a ‘clean’ device state.

As an alternate approach, it is possible to harden the application itself to detect tampering (both off-line patching plus re-packaging and runtime hooking) to some extent. Should any of the below checks fail, assume that the environment is tampered with and notify the user. Note that even a careful implementation of all below methods can only raise the bar in terms of attacker skill, motivation and expertise, it can deter and detect novice users, who solely rely on off-the-shelf tools.

- Upon startup, check if the package has debug mode enabled.
- Regularly check the signature and the signing key of the app package if it deviates from an expected value.

- List the calling stack of security related methods (such as certificate pinning, root detection etc.) Should the RootCloak tool be used, the calling stack contains elements that are clearly not from the originating application.

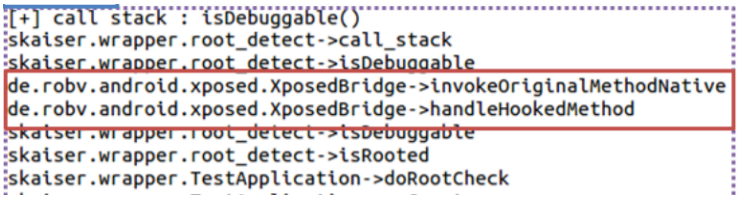

Figure 1 Evidence of dynamic hooking based tampering

For further information, refer to the following whitepaper (An Android Application Protection Scheme against Dynamic Reverse Engineering Attacks. *Lim, Yeong et al*. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, issue September 2016.)

http://isyou.info/jowua/papers/jowua-v7n3-3.pdf

Result of re-test on 27/7/2017
Testing revealed that the issue is still present.

2.2 Plain-text communication channels

Finding
Testing revealed that the application does not fully utilize SSL encryption for network traffic. For instance, the following http based URLs are retrieved during the update process.

http://80.239.197.100/index/u0607g.xml
http://80.239.197.100/bases/upd/upd-0607g.xml
http://80.239.197.100/bases/mobile/mobile-0607g.xml
http://80.239.197.100/bases/mobile/kmsh/kmsh-all-0607g.xml
http://80.239.197.100/bases/mobile/kmsh/i386/kmsh-i386-0607g.xml
http://80.239.197.100/bases/mobile/ksnm/ksnm-0607g.xml
http://80.239.197.100/bases/mobile/cdbm/cdbm-0607g.xml

Risk
For an intercepting attacker, it might be possible to intervene into the application database update process.

Recommendation
It is recommended implement certificate pinning on all SSL connections.

Result of re-test
The issue is still present. The risk is however decreased by the fact that the actual contents of the xml are encrypted.

2.3 Traces of user activity stored in plain-text format

Finding

Testing revealed that the Kaspersky application maintains traces of user activity within the sandbox. For instance, the %sandbox%/databases/google_analytics_v4.db discloses the name of applications that has been installed since the initialization of the Kasperky suite, even if the applications have been removed from the device.

The following screen shot has been taken after the removal of the 'com.example.zsombor.rootdetectorapp' application.

```
Zsombors-MacBook-Pro:5_after_scan Zsombor$ cat com.kms.free/app_bases/app_monitor/unscanned_packages.dat  | grep r
ootdetector --color
[{"key":"com.example.zsombor.rootdetectorapp"},{"key":"de.robv.android.xposed.installer"},{"key":"de.robv.android.
xposed.installer"},{"key":"mobi.acpm.inspeckage"},{"key":"mobi.acpm.sslunpinning"},{"key":"com.mwr.dz"},{"key":"mo
bi.acpm.inspeckage"},{"key":"com.example.zsombor.rootdetectorapp"}]
```

Figure 2 Traces of user activity in plain-text format

Risk

An analysis of the application sandbox reveals information regarding user activity (e.g. the names of installed applications). The risk is increased by the fact that the application maintains the log, even in case the device is rooted.

Recommendation

It is recommended to encrypt all trace of user activity, even within the sandbox. Furthermore, the user should be provided with options to delete all traces of his activity from the sandbox using the application GUI.

Result of re-test

Re-test revealed that the issue is mitigated.

2.4 Insecure permissions on application activities

Finding

Testing revealed that the Kaspersky application exports activities with no permission, which can be abused using Android IPC.

For instance, consider the com.kms.antitheft.gui.DeviceAdminLockScreenActivity activity, which effectively locks the GUI and can be opened only unsing a PIN.

```
<activity android:name="
com.kms.antitheft.gui.DeviceAdminLockScreenActivity" android:theme="
@style/LockScreenThemeNoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
    </intent-filter>
</activity>
```

Figure 3 Manifest.xml declaration of the DeviceAdminLockScreen activity

The following drozer command can be used as a Proof-of-Concept demonstration.

```
run app.activity.start --component com.kms.free
com.kms.antitheft.gui.DeviceAdminLockScreenActivity
```

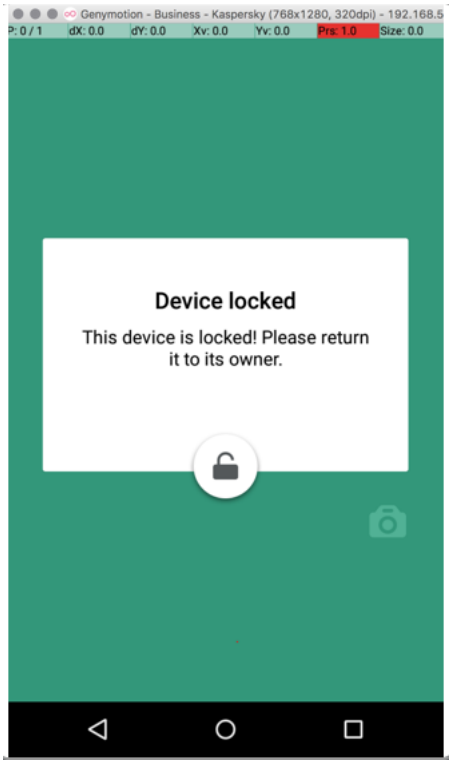

Figure 4 'Device locked' screen

This activity can be started without any permissions – therefore, a piece of malware might even abuse the Kaspersky suite to trick the user into disabling or uninstalling the application entirely.

Risk
This activity can be started by any application in the Android OS. As a result, a sophisticated piece of malware can abuse this feature to trick users into disabling or uninstalling the application.

Recommendation
It is recommended to mark this activity as not exported.

Result of re-test
In the 03/04/2017 version, this vulnerability has been mitigated.

3 Timeline

2017.04.10 Vendor notified about the vulnerability
2017.04.11 Vendor acknowledges the issues
2017.06.29 Vendor requests an extension of the grace period until 7.29, until a new version is released
2017.07.26 Vendor provided an updated version
2017.07.27 Issues re-checked