DEMO: Enabling Trusted Stores for Android

Yury Zhauniarovich*
University of Trento
via Sommarive 14
Trento, Italy
yury.zhauniarovich@unitn.it

Olga Gadyatskaya
University of Trento
via Sommarive 14
Trento, Italy
olga.gadyatskaya@unitn.it

Bruno Crispo
University of Trento
via Sommarive 14
Trento, Italy
bruno.crispo@unitn.it

ABSTRACT

In the Android ecosystem, the process of verifying the integrity of downloaded apps is left to the user. Different from other systems, e.g., Apple App Store, Google does not provide any certified vetting process for the Android apps. This choice has a lot of advantages but it is also the open door to possible attacks as the recent one shown by Bluebox [4]. To address this issue, this demo presents how to enable the deployment of application certification service, we called TruStore, for the Android platform. In our approach, the TruStore client enabled on the end-user device ensures that only the applications, which have been certified by the TruStore server, are installed on the user smartphone. We envisage trusted markets (TruStore servers, which can be, e.g., corporate application markets) that guarantee security by enabling an application vetting process. The TruStore infrastructure maintains the open nature of the Android ecosystem and requires minor modifications to Android stack. Moreover, it is backward-compatible and transparent for developers, and does not change the application management process on a device.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, cryptographic controls, authentication*

Keywords

Android, application markets, trusted installation

1. MOTIVATION

One of the recent attacks on Android, which is discovered by BlueBox, demonstrates that an attacker might be able to inject malicious code into a legitimate application without damaging the digital signature of the original developer [4]. This type of attacks is quite dangerous considering the open nature of the Android platform, which allows a user to download applications not only from the official Google Play market, but virtually from everywhere. Thus, users shopping at a third-party market cannot be reassured by just checking that the desired app comes from a trusted developer and has good reviews. And while the biggest phone manufactures claim to have fixed this vulnerability in their currently supported phones, minor providers might simply ignore it. Moreover, unsupported devices of major players are still the target for this attack. Anyway, the issue of repackaging Android apps is more general and still open [5].

While other players, such as Apple and RIM, addressed the problem by running their certification schemes and vetting each application before being published on their own (unique) market, Google chose a different path. In the spirit of openness to third-party developers, Android apps do not need to be certified before being published in any market.

This demo shows how to enable and support an application certification service for the Android ecosystem. We describe in details how to set up TruStores. A TruStore is a market that publishes only certified applications. Importantly, TruStores can certify existing applications and new ones without posing any constraints to developers and without changing the process of Android application development. The client side of TruStore relies on the current platform architecture and requires only minor modifications of the Android sources to enable the protection.

2. TRUSTORE

This section overviews the main components needed to run TruStore in Android. The TruStore architecture consists of two main parts: a *server* and a *client*.

TruStore Server.

On the server side TruStore, besides offering the standard app publishing and provisioning functionality, is responsible for the application vetting process. Applications provisioned by TruStore are trusted with respect of the declared vetting process. This process can include static analysis of application executables and source code (if provided), dynamic analysis, permission analysis, application files verification (for instance, to check for the "Master Key" vulnerability [4]), etc. What the TruStore server certifies and which technology uses for this is outside the scope of this work. However, the interested reader can refer to, e.g., [3, 1, 2] for examples of app verification frameworks.

Operationally, the process of application certification looks in the following way. A developer builds and signs (using

*Corresponding author.

CCS'13, November 4–8, 2013, Berlin, Germany.
ACM 978-1-4503-2477-9/13/11.
http://dx.doi.org/10.1145/2508859.2512496.

her own certificate) an application and uploads it to the TruStore server.

The server performs the analysis of the provided resources (executables and/or source code). If the app has passed the vetting process, the server signs the application with its own certificate and places it in the market to be accessible by users. It should be mentioned that this process is completely transparent for application developers and does not change current development and publishing workflow.

Application Multisigning.

Most of Android apps are sealed with a developer-signed certificate (notice that for Android "certificate" and "signature" are used interchangeably). This certificate assures that the code of the original application and its update come from the same place, and establishes trust relationships between applications of the same developer. To perform these checks Android simply compares binary representations of certificates, corresponding private keys of which were used to sign an application and its update (in the first case) and collaborating applications (in the second).

This check is implemented in *PackageManagerService* in the method `'checkSignaturesLP'`, which takes as parameters two arrays of `Signature`. In Android it is possible to sign the same application with several different certificates. This explains why the method takes the arrays of signatures. Despite the fact that this method takes the central place in the Android security provision, its behaviour strongly depends on the version of the platform. In the newer versions (starting from Android 2.2) this method compares two sets of `Signature`, and if both sets are equal returns `SIGNATURE_MATCH` value, and `SIGNATURE_NO_MATCH` otherwise or if an array is equal to `null`. Before the version 2.2, this method checked if the first set is contained in the second. That behaviour allowed the system to install upgrades even if they had been signed only with a subset of certificates of the original application.

Today there are no benefits of signing an application with multiple certificates and this functionality can be easily suppressed. However, we believe that for compatibility reasons Google will not change it in the near future. This multisigning functionality is the cornerstone of the TruStore client.

TruStore Client.

The client part is based on a modified Android system; it allows a device holder to make use of TruStore.

Figure 1 summarises the architecture of the TruStore client. We implemented our proof-of-concept prototype for the Google Nexus S phone using 4.1.2_r2 version of AOSP. The TruStore modifications touch two levels of the Android software stack: the *Application* and the *Android Framework* levels; at both of these levels our implementation changes the standard Android components, as well as adds new parts.

The TruStore management process starts with the activation of the TruStore protection in the standard Android *Settings* application. In this application we added two preferences: the first activates the TruStore protection, while the second allows a user to see the list of installed trusted store certificates (see Figure 2a). If a user wants to activate the TruStore protection she checks the added checkbox (step a in Figure 1). The value of this setting is written into the *Settings.Secure* content provider, and is later used by different components to detect if the TruStore protection is enabled.

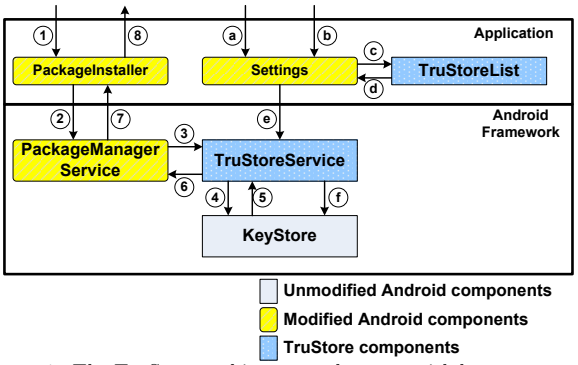


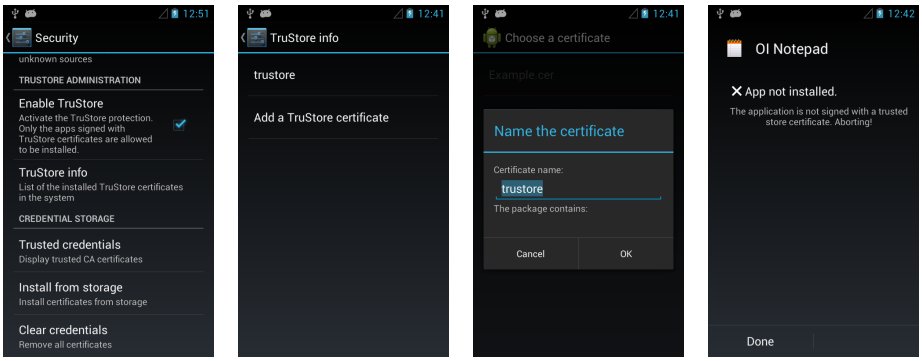
Figure 1: The TruStore architecture: the steps with letters represent the TruStore management process; those ones with numbers describe the checks during application installation.

After the activation of the TruStore protection a user is able to see the list of TruStores X.509 certificates installed in the system (step b). In this screen a user has an option of adding a new TruStore certificate (see Figure 2b). The *Settings* application shares its UID with the system server. So as it is prohibited to read the content of the external storage from the system process, the functionality to display and select the available certificates (step c) is passed to an additional application *TruStoreList* (Figure 2c), which is launched from *Settings* using an explicit intent.

To securely preserve a certificate with the additional data the system *KeyStore* component is used. This is the standard way to store credentials on Android. This component automatically encrypts the stored information and grants access to it (based on UID) only to the component that originally initiated preservation of the data. As *TruStoreService* is a part of the system server, only the Android components with UID equal to 1000 can read and modify these data. To distinguish the TruStore information from other credentials stored in the credential storage we add a special prefix (TRUSTORE). Thus, *TruStoreService* selects from the storage only the appropriate credential data.

Currently there are several ways to start the installation of a new application on a device. This can be done using the *Google Play* or *PackageInstaller* applications, or using the `adb install` command. The app installation process with the activated TruStore protection does not differ from the normal one. A user launches the installation using our modified version of *PackageInstaller* and executes the usual app installation steps (step 1). In our prototype we modified the *PackageInstaller* application, because this component is open-source. However, similar modifications can be easily be incorporated into the proprietary Google Play app.

After the initial steps the installer notifies *PackageManagerService* to start the actual installation of the package into the system. This is a special service in the Android system responsible for package management. From this point *PackageManagerService* performs the job, while *PackageInstaller* waits for the installation report. To perform the TruStore check we add a hook into the method `installPackageLI` of *PackageManagerService*. This hook extracts the certificates of the installed package and compares them against the list of trusted store certificates using *TruStoreService* (step 3). The hook is embedded in the place when Android *PackageManagerService* has finished all verification steps. In this way, we are sure that the private keys corresponding to the



(a) **(b)** **(c)** **(d)**

Figure 2: Screenshots of TruStore: (a) Settings to enable TruStore, (b) The certificate list of trusted stores, (c) TruStoreList application, (d) PackageInstaller error when a package is not signed by the TruStore certificate

certificates extracted from the package to be installed have been used to sign the package (thus, verifying the integrity of the signatures). *TruStoreService* matches the obtained list against the list of TruStore certificates; the result of this comparison is passed back to *PackageManagerService* (step 4). If a match is found then the service finishes the installation and notifies *PackageInstaller* about the success, otherwise it generates a special error that is displayed to the user (Figure 2d) by the installer application (steps 7, 8). *PackageInstaller* modifications constitute only in the ability to correctly display the explanation of this error.

App Management with TruStore.

Let us we consider how the process of application management has been changed with the TruStore modifications. There are three main points in the lifecycle of each application: install, update and delete. The process of application deletion from a user or developer point of view is not changed by TruStore, so it is not considered.

The installation process of an application was considered in details in the previous section; it is not altered from the developer perspective, but is changed from the user angle (installation may fail now). However, it is worth mentioning that if an app is installed from a TruStore market, all consequent updates must come from the same store. Otherwise, the Android system will cancel the update which is not signed by the TruStore certificate.

At the same time, the TruStore approach does not reduce the app update capabilities for a developer. In our approach, she has to submit a new version of her app to the TruStore server as usual, where it will be analysed and redistributed to users. The "kill switch" functionality can also be supported via TruStore.

The TruStore modifications influence interactions between app components protected with the `signature` and `signatureOrSystem` types of Android permissions, and on the `sharedUserId` interactions. The TruStore modifications will prohibit such kind of interactions between applications that are installed from different trusted markets, although they may have been implemented by the same developer. This limitation comes from the fact that the applications installed from different TruStore markets will have different set of signatures (although the developer signature may be the same). However, this restriction enables more security: TruStore ensures safety of the apps loaded via itself, while safety of apps on other third-party markets cannot be guaranteed (especially, in the light of the attack discussed in §1), and the interactions between trusted and untrusted apps may lead to information leaks and privilege escalation attacks.

3. CONCLUSIONS

We have presented how to support application certification in Android. TruStore aims at enhancing security of the Android ecosystem by adding the market signature to application packages upon a successful vetting process. The concept of market signatures is not novel, as it is used, e.g., by Apple. The main contribution of our approach is the backward-compatibility with the Android ecosystem. TruStore is fully transparent to legitimate application developers and does not require major modifications to the firmware. It may be adopted immediately by phone producers with their own markets, for example, Samsung; and, if being incorporated into the Android distribution released by Google, may become a valuable feature for corporate application markets to enable BYOD policies.

Acknowledgments.

This work has been partially supported by the FP7-ICT SecCord Project 316622 funded by the EU and the TENACE PRIN Project 20103P34XC funded by the Italian MIUR.

4. REFERENCES

[1] W. Enck, M. Ongtang, and P. McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 235–245, New York, NY, USA, 2009. ACM.

[2] P. Gilbert, B. Chun, L. Cox, and J. Jung. Vision: automated security validation of mobile apps at app markets. In *Proc. of MCS'2011*, pages 21–26, 2011.

[3] V. Rastogi, Y. Chen, and W. Enck. Appsplayground: automatic security analysis of smartphone applications. In *Proc. of CODASPY '13*, pages 209–220, 2013.

[4] T. SecurityLedger. Exploit code released for android security hole `https://securityledger.com/2013/07/exploit-code-released-for-android-security-hole/`, Jul. 2013.

[5] W. Zhou, Y. Zhou, M. Grace, X. Jiang, and S. Zou. Fast, scalable detection of "piggybacked" mobile applications. In *Proc. of CODASPY '13*, pages 185–196, 2013.