Toward Active and Efficient Privacy Protection for Android

Yuhao Luo, Dawu Gu and Juanru Li

Abstract— **Although Android has introduced many security mechanisms, users often expose privacy information to attacker due to the system's defensive privacy protecting policy. The problem is that for most inexperienced users, no mandatory protection is provided. To address this issue, we propose a data-centric privacy enhancement design to actively restrict privacy violation on Android. The main idea is to first build trusted database by introducing secure enhanced kernel and data-at-rest encryption. Then, the system enforces an isolation of applications with privacy data access privilege mode. The design focuses on data protection and keeps persistent mandatory access control model from kernel to application layer, and could resist most common privacy leakage attacks. Compared with other heavyweight isolation scheme, the overhead is also controlled into an acceptable range due to our lightweight design principle.**

I. Introduction

ANDROID has become the most popular mobile OS and has nearly half of the world's mobile OS market share. However, it is notable that there are more and more attack and threats aiming at the Android platform. And many analyses have pointed out that Android security policy contains defects[1]. For instance, Android users do not have the option of accepting permissions on a granular basis, even though the Android permission model may be granular[2].

Protecting the security of entire mobile platform including hardware, firmware and operating system is a complicated task. Even like Apple's iOS with secret design, attacker could also break the multiple barriers and jailbreak the phone. On the other hand, sensitive data such as *phone call logs*, *contact information*, *SMS*, *locations* and *photos*, among all data on mobile device, is the most sensitive information connect to user's privacy. The extended functions and features of Android make users increasingly rely on mobile devices, and be willing to store personal data in their phones, bringing about the security concern of users' privacy. A common threat is from third party untrusted applications. Except Google's Android Market, users are allowed to install applications from third party sources. This makes it possible that untrusted applications leak users' privacy information. Thus we minimize our target to protecting a small range of data – privacy data (*phone call*, *SMS*, etc.).

We focus on privacy data protection rather than a complex privilege protection model for some reasons: First, most of the privilege attacks aim at privacy. Spyware often steals the privacy information of users and lets users to pay more to their bills. We expect that the active control brings a better security against common malware as well as privilege escalation attack, which may need to broke more barriers. Second, it is hard to restrict various privilege required by applications in different scenarios, but to protect data is a solely simple target. And mobile devices need lightweight and portable scheme that consumes less power and can be easily transplanted.

Our work concerns about active privacy data protection enhancement on Android. Actually, most security strategies on Android are defensive - aiming at protecting the perimeter of the system rather than privacy itself. We argue that a more aggressive Encryption-and-MAC(Mandatory Access control) strategy is required for Android to prevent common privacy leakage attacks. That is to say, the control model is that the system maintains encrypted trusted database and only provides a restricted subset of privacy data for normal applications. The sensitive data is not given unless to system applications or applied by user's interactive authentication process. Moreover, data-at-rest encryption and kernel enhancement are introduced together to fulfil protection of raw file system from leaking or exploitation either through physical contact or through malicious code.

This paper makes the following contributions. First, we focus on data-at-rest encryption[3] and privacy access policy. The database encryption proves data security and the privacy access policy is cautious. Unless restricted authentication (with the help of certificates or even trusted hardware) is done, no sensitive privacy data is decrypted. What's more, a kernel level anti-exploit module to protect file system and gives strong support to database encryption and security restriction in user level. Second, our target is to turn manual security checking to security policies based active protection. Users can acquire better security without defining complex security policy. Third, our encryption scheme is light weighted. The database encryption is a lightweight scheme compared with disk encryption and bring less performance overhead. Also, our design prototype doesn't lose the convenience brought by various applications from either official Android Market or third-party marketplaces.

II. Related work and Motivation

Android system provides several mechanisms to help enhance data security. Original protection provided by Android include screen lock, encryption and permission declaration. But these mechanisms are not so effective and may even be confused for those who are not familiar with Android security.

For users who take countermeasures, some of them may install antivirus tools[4][5], some of them may use disk encryption[6]. But these security enhancements are still far from enough, none of these protection could prove privacy

This work was supported by SafeNet Inc.

Y Luo, D Gu and J Li are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China (email: {lyh62771}@gmail.com).

security solely. What's more, industrial anti-virus applications working in background cause a great deal of resource consumption.

There are also many academic researches done to improve Android security. *TISSA*[7] empowers users to flexibly control in a fine-grained manner what kinds of personal information will be accessible to an application. Also, the granted access can be dynamically adjusted at runtime in a fine-grained manner to better suit a user's needs in various scenarios. *Saint*[8] is another work that allows a developer to provide a security policy that will be enforced (by the enhanced Android framework) to regulate install-time Android permission assignment and their run-time use. *Apex*[9] is a policy enforcement framework for Android that allows a user to selectively grant permissions to applications as well as impose constraints on the usage of resources. L^4*Android*[10] is representative for the VM based ones using permission isolation and hypervisor to make permission mechanism more strict. But they didn't consider about security in application level. However, these academic researches seem technically too complicated for end users.

Even threatened by various of potential attacks, mobile device users generally do not set security policy to defend malicious code because most existing security enhancements are not suitable for security unaware end-users to set policy by themselves. *TrustDroid*[11] is an active security framework that enables domain isolation at different layers of the Android software stack such as kernel and application layer IPC channels and enforces certain policy. Although claimed as a lightweight domain isolation, it monitors each IPC process and employs MAC policy, which is complicated.

In our design, we propose an active yet lightweight protection scheme. The core idea of our design is data-centric protection. We only focus on data source protection and loosen the restriction of monitoring every data flow. Although there exists numerous third party "untrusted" applications for Android, most of them are not malicious and only collect some statistic information for user behaviors. These applications should be allowed to run on our system but sensitive data should not be given. So in our active protection system, forced security restriction in user level controls applications' access to private database and encrypted databases protect user privacy from other accesses. And the MAC kernel level uses a strict permission control policy to protect the file system, which is a strong support to the upper layer modules.

III. System Design

We first discuss our design goal and the security model, then the design details.

A. Design Goal

To design a suitable privacy protection mechanism on Android, there are three factors we need to concern about: *security*, *efficiency* and *convenience*. And two observations mainly affect our design. First, most of privacy leakage is due to common malware or Adware rather than the advanced exploit. So restricting these "gray" applications' maleficent behavior is more appropriate than uninstalling or removal. That is to say, a loose but intelligent filtering should be used so that those applications are still allowed but with limited privacy information accessible. Second, the design must complete protecting task with an eye toward the power consumption overhead and the effectiveness of user operation. Obviously user experience and power consumption are more important on mobile devices. A too complex security enhancement mechanism not only slows down the system response, but also consumes more energy. So data encryption and access control must be lightweight so that the system can afford.

Security is obviously the key factor contribute to our design. The main design principle is to build a data-centric security model. On this basis, users also expect a privacy data protection and management mechanism that is active, flexible and robust. An active strategy should be adopted to restrict behaviors, for inexperienced users may not take any countermeasure. In a word, the expected system is based on data-centric secure storage and provides an active, well-defined privacy management policy. For efficiency and convenience, a basic requirement is the new features should running with little interference. In other words, it is better to employ less change to the original system model or operation style.

To fulfill the main purpose of security, efficiency and convenience, The characteristic of the designed architecture is shown in fig 1. We use data-at-rest encryption and security enhanced kernel to protect private data from being accessed illegally, and use data domain isolation to achieve privacy access management of different applications. The rest of this section highlight security model and relevant design details.

B. Data-centric Security Model

Android system provides a "permission" mechanism to enforce restrictions on the specific operations, which is privilege-separated style. However, the control policy tends to be ambiguous due to the requirement of development. It is hard to judge whether an application's permission request is benign or malicious. For instance, a map application may ask for *GPS* information to locate your position, but it is possible to divulge users' location info. In order to solve this problem, We instead propose a data-centric security model. The core idea is to build a trusted data storage as the base of privacy. Then, a well-defined mandatory data isolation and access policy is adopted to restrict privacy leakage. Under this model, privacy data is well protected and at most situations the operation is at a low data-privilege environment. The model contains following features,

- *kernel based file access control*
- *data-at-rest encryption*
- *enforced authentication*
- *data domain management*

Applications by default could only acquire insensitive data. If an application exceeds its authority of data access, a null or faked value will be returned or it can get the sensitive

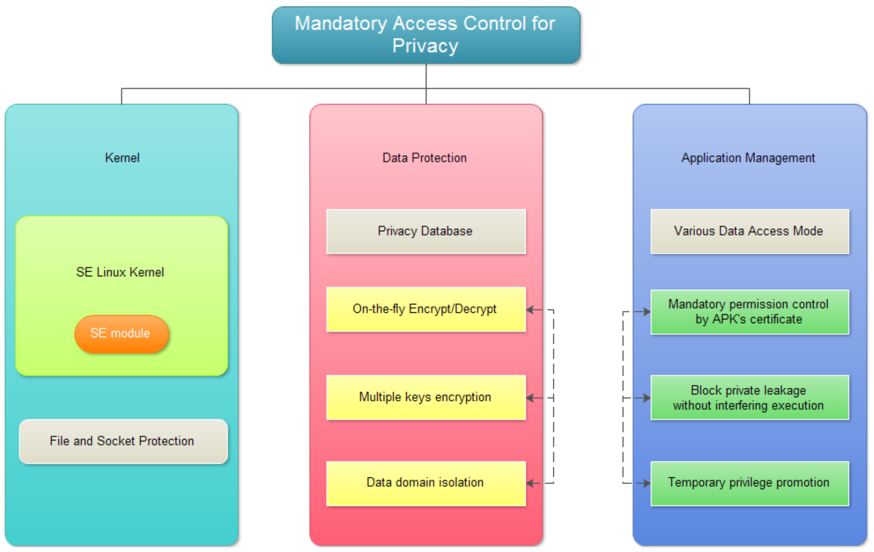
Mandatory Access Control for Privacy
Kernel
SE Linux Kernel
SE module
File and Socket Protection
Data Protection
Privacy Database
On-the-fly Encrypt/Decrypt
Multiple keys encryption
Data domain isolation
Application Management
Various Data Access Mode
Mandatory permission control by APK's certificate
Block private leakage without interfering execution
Temporary privilege promotion

Fig. 1. Characteristic of active privacy protection

data only by the user's password authentication. Under this restriction, even untrusted applications is installed and executed to meet users' needs, it is limited when requiring for privacy.

C. *Trusted Database*

One of the main insecure factors of the current Android system is that all contents in the database are stored in plain text. The data stored is in plain text form and malicious applications or someone who gets physical contact with the phone can easily take advantage of the privacy information without any restrict. Many research works suggest protecting data with encryption. But to build a trusted encryption scheme may provoke many problems. First, Existing data encryption protection mechanism on Android OS platform is heavyweight[6]. Second, Some malwares can do privilege escalation attack on Android which means they can have access to the data even without users' grants. Even worse, Once the users unconsciously approve to the installation of a malware, the permissions the malware applies are fixed to it and the malicious code can easily misuse the users' privacy information or important system resource to do whatever it wants without the users' awareness.

To improve the encryption scheme and build a trusted database, MAC kernel and lightweight data-at-rest encryption are introduced to build such trusted database storage. MAC kernel and lightweight data-at-rest encryption can resist most physical access attacks to ensure a trusted database. MAC kernel closes the door on the possibility of rooting, no matter by physical-recovery or by *adb*. Without root privilege, the value of the key can never be obtained and the database is well protected. That means even if the attacker get physical access to the phone, he still cannot get anything privacy related.

1) lightweight data-at-rest encryption: We argue that a lightweight database encryption instead of whole disk encryption is enough. Our system is designed to eliminate the insecure database by encrypting the important database which are closely related to user's sensitive information such as *Contacts*, *SMS*, *Calling history*, *Memorandum*, etc. And in our scheme these sensitive files are decrypted only when legitimate users are using the phone. When system is booting, it is convenient to load an encrypted database rather than an encrypted partition. Besides, to reload the encrypted database is of little cost thus each time the screen is locked, system could erase decrypted data in memory directly and no sensitive data is stored. Thus the privacy data is well protected.

2) MAC kernel: Due to its open characters, Android suffers from many attacks from low-level. *Rooting* is a process allowing users of Android to attain privileged control (known as "root access") within Android. As Android was derived from the Linux kernel, *rooting* an Android device is similar in practice to accessing administrative permissions on Linux. Certain applications may take advantage of the root privilege to get important system resource and data which may cause sensitive data leaking. Another potential threat is that Android's *adb* mechanism with a high permission. *adb* is used by many phone management utilities on Personal Computer, but no authentication is needed to connect to *adb* interface from computer and any attacker could make use of it.

To prevent the potential abuse, we replaced the original discretionary access control(DAC) with mandatory access control(MAC) of which the control granularity enhances from application to process. Generally speaking, in DAC model, users control resource access. For convenience, some security unaware users set dangerous file permissions. In addition, a process started by a user can modify or delete any file to which the user has access. In contrast to DAC model , MAC model regulates user and process access to resources based upon an organizational security policy which is much higher-level to control permissions, and can stop many known exploits for Android.

MAC kernel also gives a strong support to the database-encryption module and the user level permission restriction. MAC kernel may provide a far finer granularity for permission check in order to prevent those malicious behaviors using native code or other means which go directly into the file system and destroy the basic database or replace it.

D. Data access management for Applications

Android doesn't provide multi-level data access control in application layer. All applications are executed under Dalvik VM[12] with same privilege level. For application layer privacy security, it is natural to design a finer MAC data access model. The design mainly contains two aspects: *enforced authentication* and *data domain isolation*.

1) enforced authentication: Like web browser, a certificate database is maintained in system to store trusted certificates in order to distinguish the source of the application. The security restriction and policies verification check an application's certification in order to judge its privilege level. That is to say, every application is to be classified when installed by being checked its signature depending on database where trustful certificates are stored. This enforced authentication permits the installation of third party applications, but preventing the data leakage. An application with a trusted signature can be installed and run normally while an application without the trusted signatures will be censored when data accessing.

2) data domain isolation: To implement the target of active private protection, it is not enough with the supporting of trusted database. A multiple level data isolation mechanism is suggested, and a hierarchy for sensitive data is also required according to their sensitivity. By default, the installed applications are executed under a lower privilege mode and could only access part of the sensitive data. Our design provide a isolation in application level and map the classified applications to different sensitive data domains (see Fig 2). If trusted, the application can access the corresponding sensitive data it required. Otherwise an empty database or a fake data set is to be returned without privacy leaking. In another word, without the trusted signatures, an application will be blocked with some of its permissions to get user data of certain sensitivity. Moreover, under certain circumstances an interactive is allowed for application to require higher privilege. User can promote the application's privilege by giving certain password to unlock the encrypted high sensitive database.

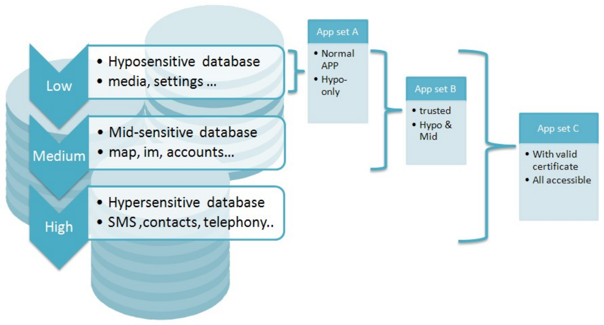

Fig. 2. Data domain isolation

IV. Implementation details

In this section we discuss implementation details and some open problems.

A. Kernel Protection

We adopt the Security Enhanced (SE) Android scheme to protect kernel[13]. SE Android establishes the kernel-level protective mechanism which ensures the security of the file system, and provides a kernel level anti exploit protection.

Root-get trials such as *GingerBreak*, *Exploid*, *RageAgainstTheCage*, *Zimperlich*, *KillingInTheNameOf* and *psneute* were failed which strongly proves the SE Android has overcome the weakness in Android DAC and can stop many published exploits for Android.

SE Android uses the MAC instead of the DAC in Android by which the control granularity enhances from application to process. All applications are denied to read files created by other applications. It has also built a fixed policy to change permission check to domain management and isolated applications by sandbox using category. All of these measures can improve the flexibility and authorization granularity.

B. Privacy Encryption

In order to coordinate with the SE Android kernel and ensure the system to be lightweight, we encrypted the privacy databases rather than the whole file system. And the database is decrypted after screen unlocking with the user's private key. Two alternative schemes can be chosen, either to implement the encrypting APIs of SQLite or to directly encrypt the database file. This also prevents illegal users getting information from the phone from physical contact. Sensitive data is frequently used during system running, thus RC4 cipher is chosen to be the algorithm to encrypt the database due to its efficiency. The general process is shown in fig 3.

The forced security restriction and policies verification in user level check an application's certification in order to judge its trust level when it requires user private data. If it is trusted, it can have access to the data it required. Or it can reach an empty database or get fake data to run its origin function without data leaking. This enhances the security of the core data encryption. As for database encryption, only selected key databases containing user private data will be encrypted which can lessen system resource consumption to the greatest extent.

When a smart phone with the encrypted databases is powered on, it needs the user's private key to unlock the screen. If legitimate, the user can get in normal using while the background service is writing back the encrypted data in

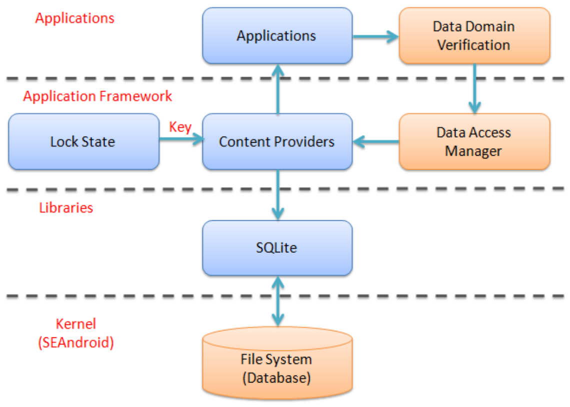

Fig. 3. Encryption Process

some intervals. To be more specific, the decrypted database should be stored in ROM so that users can get into the system and run applications after unlocking the screen. Since only legitimate users own the password, those who can successfully unlock the screen and get into the system are regarded as legal users, thus the system would decrypt the database after their login. When the user locks the screen, the plain text databases will be written back and encrypted. At the same time, the plain text files in the ROM will be erased.

C. Application management

1) Application privilege isolation: Each time an application is installed, an automatic process of authorization is accomplished without user's participation. So after the installation, every application will belong to a certain application set which is to be mapped to a certain data domain.

There are three sets of applications: *authentic*, *semi-authentic* and *untrusted*. Only system applications or authentic exceptions added by user belong to *authentic* set. We choose to semi-trust those signature created by Google Android Market. So we build a database of these signatures and use this database to measure application belonging to *semi-authentic*. Applications from any other alternative application markets will be marked as *untrusted* one. Meanwhile, sensitive data are also classified into three sensitivity levels: *hyposensitive*, *mid-sensitive* and *hypersensitive*. For instance, the hypo one consists of *settings.db*, *media.db* and databases less related to privacy. The hypersensitive one may include *SMS.db*, *contacts.db* and so on.

2) Privilege operation mode: Classification of applications at install-time implements the isolation of privilege for applications. Under different privilege operation mode the system provides different sets of privacy data to applications. All databases are available for applications in *authentic* set. Logically, *Hypo* and *Mid* databases are accessible for *semi-authentic* set, *hypo-only* for *untrusted* one.

To ensure best user experience, empty data will be given to the application so as to maintain its normal functions and at the same time a warning is logged. When warnings reach a certain amount, it implies that either this application is a malware or it is authorized improperly. Then the user will be informed of this situation and decide whether to uninstall or ignore or improve the application's privilege by password authentication.

V. Discussion

A. Verification on Calling vs Monitoring

The lightweight of our system is largely depends on its mechanism of calling to run while most of security softwares and anti-virus tools are monitoring all the time. Monitoring services cause a large consumption of battery power and system resources. But our system is designed of active protect protocol which is called only when it is needed which consumes a few resources to check for certificates. Also, most monitoring applications needs complicated configuration and offers too professional reports, which is obviously not friendly for nonprofessional users and everyday use.

B. The Whole System Encryption vs Database Encryption

From version 3.0(*Honeycomb*), Android provides the whole system data-at-rest encryption option which encrypts the whole file system while our system encrypts only chosen databases to protect user private data. The whole file system encryption takes more than an hour to complete and cannot be canceled. With every password change, it takes the same time to get complete encryption. What's more, it affects the whole running system which can bring troubles to users. In our design, we encrypt only chosen databases, for example, sms.db, to protect user privacy without affecting the origin system and without consuming too much time and resources. After encryption is completed, Android original file system encryption doesn't disturb users too much with delay and password inserting. Our system causes even less delay and needs little user participation in permission control.

C. Anti Common Attack Capability

The whole encrypted Android file system still cannot defend kernel level attack. However, our system enhances user privacy data security in all domains by providing a three-level security policy. As referred in related work, the transplanted SE Android kernel can protect exploits and support database encryption. The encrypted databases can protect physical attack as well as user level application data leaking. And the user level permission restriction provides a flexible policy to protect user private data without disturbing applications running with requiring them.

D. Security

The security of data relies on encryption and authentication. Our target is to design a data-centric secure system that protect user private data from physical contact attacks and remote attacks. The private data may be leaked through two channels, *file system* or *content providers*. For the attack via first channel, if the attacker could access the decrypted database files, all the information is leaked. However, our data decryption mechanism ensures that the decrypted data is stored only in memory. And our scheme protects the data using standard ciphers thus the only possible threats is that attackers could dump memory. To defend this attack we erase the in-memory data when the device is screen-locked.

For the attack via second channel, to access data from content providers, the attacker should bypass two authentication processes. One is application certificate authentication, another is unlock passcode authentication. So this has made the permission control for data security more strict.

E. performance

We evaluated the encryption performance by comparing the Android original whole file system encryption and our selected databases encryption.

Our design philosophy proves that when the device is unlocked, the database is decrypted and is stored in memory. According to experiments in Android emulator, the decrease of performance is not too much affected and its within acceptability.

Another aspect is that to prove the data reliability, our system writes back the encrypted database files as soon as user data is changed. Because modification of the user data is far less than user data reading, it is acceptable.

What's more, encrypting the selected databases the first time needs only a few seconds which is far less than encrypting the whole file system. It introduces little interference in using and little delay in performance.

VI. Conclusion

In this paper we discuss the design of active privacy protection for Android. Our design adopts data-centric strategy and ensures the safety of user's private information against both software attack and physical contact attack. Besides, the performance of the phone is influenced in an acceptable state due to the lightweight modification of original system.

References

[1] W. Enck, M. Ongtang, and P. McDaniel, "Understanding android security," *Security & Privacy, IEEE*, vol. 7, no. 1, pp. 50–57, 2009.

[2] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google android: A comprehensive security assessment," *Security & Privacy, IEEE*, vol. 8, no. 2, pp. 35–44, 2010.

[3] . RSA Inc., "Securing data at rest: Developing a database encryption strategy," *URL: http://www.rsa.com/products/bsafe/whitepapers /DDES_WP_0702.pdf*, 2012.

[4] . Lookout Inc., "Lookout mobile security," *URL: https://www. mylookout.com*, 2012.

[5] . NetQin Inc., "Netqin mobile security," *URL: http://www.netqin.com*, 2012.

[6] . Google Inc., "Notes on the implementation of encryption in android 3.0," *URL: http://source.android.com/tech/encryption/android _crypto_implementation.html*, 2012.

[7] Y. Zhou, X. Zhang, X. Jiang, and V. Freeh, "Taming information-stealing smartphone applications (on android)," *Trust and Trustworthy Computing*, pp. 93–107, 2011.

[8] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically rich application-centric security in android," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. Ieee, 2009, pp. 340–349.

[9] M. Nauman, S. Khan, and X. Zhang, "Apex: Extending android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 328–332.

[10] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter, "L4android: a generic operating system framework for secure smartphones," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 39–50.

[11] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A. Sadeghi, and B. Shastry, "Practical and lightweight domain isolation on android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 51–62.

[12] D. Bornstein, "Dalvik vm internals," in *Google I/O Developer Conference*, vol. 23, 2008, pp. 17–30.

[13] A. Shabtai, Y. Fledel, and Y. Elovici, "Securing android-powered mobile devices using selinux," *Security & Privacy, IEEE*, vol. 8, no. 3, pp. 36–44, 2010.