ThiefTrap – An Anti-theft Framework for Android

Sascha Groß(✉), Abhishek Tiwari, and Christian Hammer

University of Potsdam, Potsdam, Germany
{saschagross,tiwari,chrhammer}@uni-potsdam.de

Abstract. Smartphones store a plenitude of sensitive data. This data together with high values of smartphones make them an attractive target for physical theft. Clearly, the device owner would like to regain the device in such a case. Also, the information should be protected from illegitimate access.

In this paper, we present the first anti-theft solution that effectively handles these issues. Our proposal is based on a novel concept of an anti-theft honeypot account that protects the owner's data while preventing a thief from resetting the device. Thus, a stolen device can be regained by the device owner with high probability, while information leakage to the thief is prevented. We implemented the proposed scheme and evaluated it through an empirical user study with 35 participants. In this study, the owner's data could be protected, recovered, and anti-theft functionality could be performed unnoticed from the thief in all cases.

Keywords: Anti-theft · Data protection · Information hiding
Privacy

1 Introduction

Smartphones play a vital role in everyone's life. Their contribution is significant in every day to day activity. Nowadays, smartphones are used for various activities such as capturing pictures, browsing the internet, and using online banking. However, these great advantages come at a price. If the device gets in the wrong hands, the device owner does not only lose the device but also a great amount of personal data. A thief getting hold of personal data and trying to exploit it may result in fraud or blackmailing. It is of utmost importance to provide some mechanism to protect the theft of smartphone devices and the personal data on them. These device protection mechanisms are called *anti-theft mechanisms*.

At present, the smartphone market ranges from around 50 USD to 1000 USD. Loss or theft of a phone does not only result in financial deprivation but also of the personal data which is stored on the device. According to a study [17], the number of stolen smartphones rose to 3.1 million in 2013. Another study [14] reveals that victims are willing to pay 500 to 1000 USD to regain their personal data including photos and videos.

X. Lin et al. (Eds.): SecureComm 2017, LNICST 238, pp. 167–184, 2018.
https://doi.org/10.1007/978-3-319-78813-5_9

The Android market share is continuously increasing [13] and it dominates the smartphone market with a share of 86.8%, by the end of Q3 2016. Taking this into the consideration, we targeted the Android platform for the implementation of our approach. At present, there are two possible anti-theft mechanisms:

1. *Anti-theft applications:* Most of the anti-theft applications provide the functionality to lock the phone, erase it or triggering an alarm from remote. Unfortunately, anti-theft applications do not work if they do not have an active network connection e.g., if the SIM card was removed from the device.
2. *Google Device Protection:* Starting from Android version 5.1, Google released a new feature called "Device Protection" [18]. This anti-theft feature makes it impossible for a thief to use a stolen phone after it was factory reset. However, this feature is only available for a small number of devices, which are capable of running Android version 5.1 or greater. More than 45% of the Android phones use a lower Android version [11].

In the majority of these cases, device owners permanently lose their smartphone together with their personal data, which is even worse. As a remedy, we propose a mechanism where a device owner has the option to configure an anti-theft honeypot account, which resembles the owner's regular account except for some modifications. A person that is not the device owner can never distinguish interacting with the anti-theft account or the real account. The anti-theft account hides the personal data of the device owner and performs hidden anti-theft functionality while the thief uses the device. One important feature of this framework is that when the thief performs a factory reset, our approach only gives the illusion that a factory reset is being done, while in reality all of the owner's data is preserved. After a fake factory reset the device hides the owner's account completely but still executes the anti-theft functionality. The thief will then start using the smartphone as a new device, unaware of anti-theft functionality executing in the background. It is likely that after the fake factory reset a thief will establish an internet connection by inserting a SIM card or establishing a WIFI connection. At this point of time the hidden anti-theft functionality can for example send identifying information of the thief to the device owner or start listening for remote commands from the device owner. So the device owner likely will be able to recover the device and the personal data. The key benefit of this approach is that it improves chances of identifying the thief and regaining the stolen phone as well as the personal data.

Our Contributions. In this paper we propose ThiefTrap, an anti-theft framework that uses the Android account feature as a security measure to protect against device thieves. We are the first to use this feature in that we set up a honeypot account simulating the device owner's account. Technically, we provide the following contributions:

1. *ThiefTrap.* We propose ThiefTrap, a novel concept, using a honeypot account for the purpose of theft protection. This concept is the first anti-theft solution that at the same time protects the confidentiality of the owner's user data,

prevents loss of this data and provides the full functionality of every anti-theft solution. An important benefit of the proposed approach compared to existing anti-theft solutions is that a device instrumented with our approach is *indistinguishable* from an ordinary device. Our approach ensures that the device and the personal data on it can be regained with high probability.

2. *Implementation.* We implemented our concept in the latest version of Android (7.1_r1 Nougat) of the Android Open Source Project.
3. *Evaluation.* We evaluated our approach in the form of an empirical user study. Our study with 35 participants, showed that in all cases our approach prevented loss of ***owner's personal data*** and performed the ***required anti-theft functionality***. In the very vast majority of cases the potential thief was completely oblivious to our approach.

2 Background

Android. Android is the world's most popular mobile operating system. Figure 1 depicts the internal structure of Android OS.

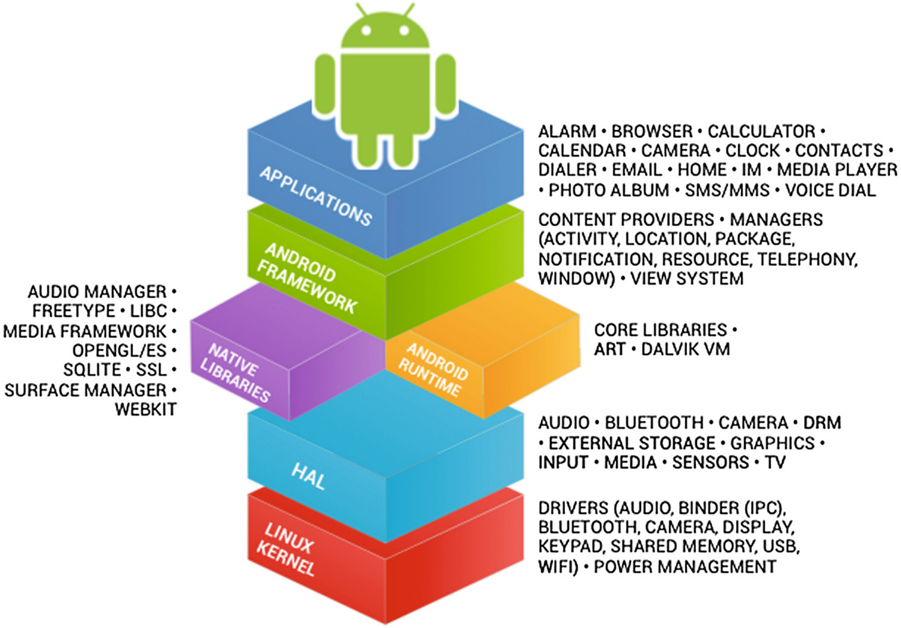

APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
NATIVE LIBRARIES
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
ANDROID RUNTIME
CORE LIBRARIES • ART • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

Fig. 1. Android architecture [9]

The Android framework can be best described in the form of different layers. The lowest layer, a customized *Linux Kernel*, is used for drivers and hardware support. The subsequent hardware abstraction layer (HAL) provides a standard interface for exposing the hardware capabilities to the higher-level Android frameworks. HAL implementations are built into shared library modules (.so files).

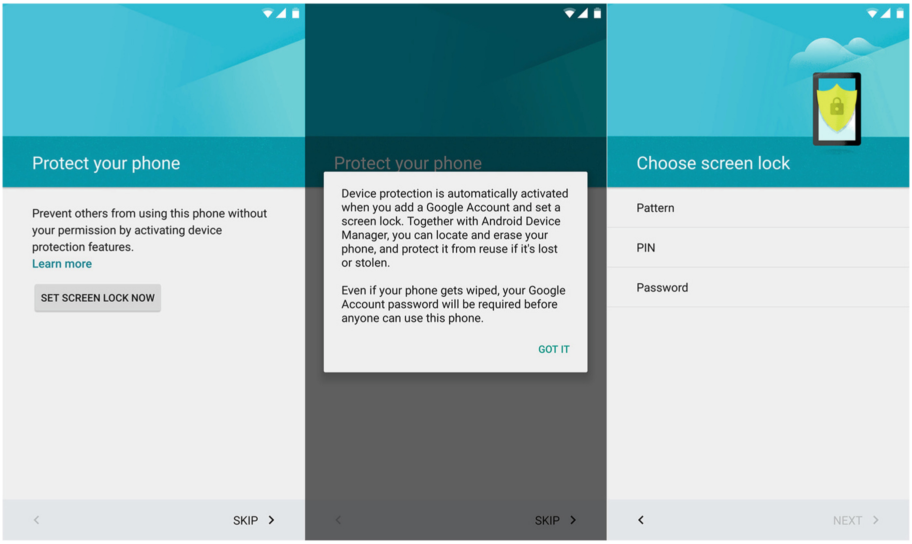
Android applications are compiled to a specific bytecode format (DEX) designed specially for Android. The Android runtime (ART) provides a Dalvik virtual machine, which is similar to the standard Java virtual machine, but designed and optimized for Android.

The Android framework layer provides many higher-level services in the form of an API to the Application layer. These APIs act as the building blocks to create Android applications. Application developers utilize these APIs in their applications. Most of our changes are implemented in this layer.

The topmost layer, the application layer, provides different applications to be used by end users, such as alarms, browser, calculator etc. Google provides a central store, for developers to publish their applications, called the Google play store. As of December 2016, the Google play store included over 2.5 million apps.

Anti-theft Mechanisms. Anti-theft mechanisms are supposed to prevent device theft or mitigate the damage in case a device gets stolen or is lost. One kind of anti-theft solutions tracks information of the device after it was stolen and provides the information to the device owner. Several anti-theft solutions rely on providing location information and remote administration functionality to the device owner. Some of them also provide the possibility to recover personal data or remotely wipe the device. At present, there are two options for Android owners to protect their device against theft. The first option is to use the Android device theft protection feature (available for devices capable of supporting Android version greater than 5.0). The second option is to use an third party anti-theft application.

- *Android built-in anti-theft mechanisms:* Starting from Android version 5.1, Google released a new feature called "Android Device Protection" [18,26]. This anti-theft feature prevents a thief from using a stolen phone that has been wiped. However, more than 45% of the Android phones use a lower Android version [11]. In addition, this feature will not work without a proper setup. For example, the user needs to log into a Google account on the device. Then, if a device supports this feature, Android Device Protection is enabled as soon as the user enables a locking mechanism. Figure 2 explains the activation of this feature while enabling a device locking mechanism.
  After a factory reset, Android Device Protection requires the user to enter the Google account credentials on which the device was previously configured. This renders the device unusable to the thief even after a factory reset was performed. However, an unlocked bootloader still allows to flash a binary on the device, thus this feature is not available for devices with an unlocked bootloader.
- *Anti-theft Applications:* There is a multitude of third-party anti-theft applications available in the app stores. These applications provide features like locating the device, remotely administrating the device etc.



Prevent others from using this phone without your permission by activating device protection features.

Device protection is automatically activated when you add a Google Account and set a screen lock. Together with Android Device Manager, you can locate and erase your phone, and protect it from reuse if it's lost or stolen.

Even if your phone gets wiped, your Google Account password will be required before anyone can use this phone.

Fig. 2. The Android Device Protection feature [26]

The anti-theft features of these applications heavily rely on a network connection. These applications are mostly not functional if the SIM card was removed from the device. A thief is likely to remove the SIM card from a stolen device and to turn it off, such that the device loses its network connections. Later, in order to reconfigure the device as new, a thief is likely to perform a factory reset, so the anti-theft application is removed from the device, leaving it unprotected. Additionally, the personal data of the device owner is lost unrecoverable.

3 Methodology

When a device is stolen there are two possibilities depending on whether the device is protected by a locking mechanism or not. In case the device is not protected by a locking mechanism, a thief immediately has unlimited access to the device owner's data and may result in abuse of the user data on the device (e.g. credentials) to inflict further harm to the device owner. If the device is locked, it is of no use to the thief as long as the locking mechanism is in place. For this reason it is likely that the thief will factory reset the device, in which case all user data on the device is ultimately lost. Modern smartphones store a lot of valuable private data. Additionally, installed anti-theft applications will be removed from the device so chances are minimal that the device can be retrieved by the device owner. Both of these scenarios are unsatisfying. Therefore, there exists a need for a solution that protects the confidentiality of the user data, while it prevents the device from being factory reset illegitimately. In this work

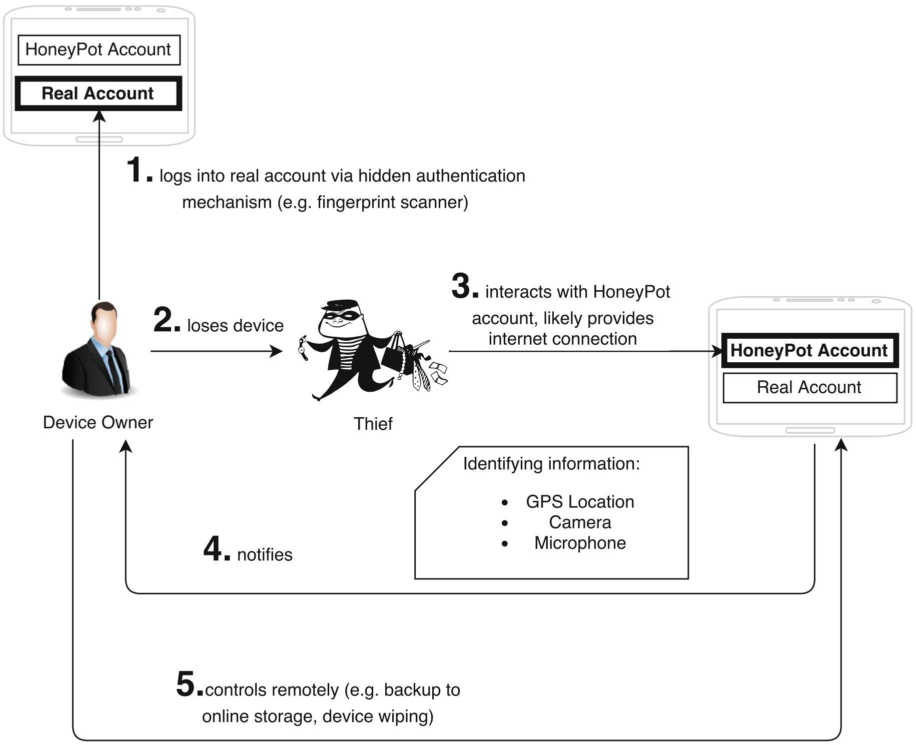

HoneyPot Account
Real Account
1. logs into real account via hidden authentication mechanism (e.g. fingerprint scanner)
2. loses device
3. interacts with HoneyPot account, likely provides internet connection
HoneyPot Account
Real Account
Device Owner
Thief
Identifying information:
• GPS Location
• Camera
• Microphone
4. notifies
5. controls remotely (e.g. backup to online storage, device wiping)

Fig. 3. Workflow of a device instrumented by ThiefTrap

we propose the first approach that can protect the confidentiality of the device owner's user data, while preventing a thief from factory resetting the device and thus removing installed anti-theft applications.

We propose the concept of a honeypot account for theft protection. We leverage the Android guest account feature to implement the honeypot account. Android's multi-account feature was introduced in Android version 4.2. According to statistics provided by Google [12], this feature is supported by 95.8% of devices. The idea of this concept is that it pretends to be the account of the device owner, while it actually is an isolated honeypot account and prevents any access to the user data of the device owner. A thief logging into the device using the home button or power on button, is logged into the honeypot account, which pretends to be the device owner's account. Therefore the honeypot account tricks a thief into believing that he/she is interacting with an unprotected device in an ordinary way, while actually interacting with the honeypot account. The device owner can log into the real account using a hidden mechanism e.g., using fingerprint lock. This mechanism can be configured by the device owner.

Using the proposed concept of a honeypot account, the privacy of the user data is protected. At the same time, in our approach a factory reset initiated from the honeypot account is simulated s.t. the thief believes that the factory reset is being performed, while in reality the owner's data is preserved. After this simulated factory reset, the thief is presented a new account as expected. However, this new account is a customized honeypot account, which runs an anti-theft mechanism in the background hidden from the thief. The great strength of this concept becomes notable when it is combined with existing anti-theft solutions. As the device owner knows that a honeypot account is installed on the phone, he/she will not log into the honeypot account of that device. For this reason it is likely that a user interacting with the honeypot account for some time is not authorised to do so. Therefore, in our approach an anti-theft solution is installed and will be triggered whenever an user interacts with the honeypot account for some time. This anti-theft solution can then for example collect data of the thief and send them to the device owner, who can use them for regaining the device. Figure 3 shows the described workflow.

We would like to stress that there exist numerous ways to implement the concept of an anti-theft honeypot-account on various platforms. We choose to implement our approach on the Android operating system. We use the uncustomized version 7.1_r1 from the Android Open Source Project [10]. At the time of this writing, Android is the most used mobile operating system with over 1.4 billion devices in usage and version 7.1_r1 is the latest version of Android.

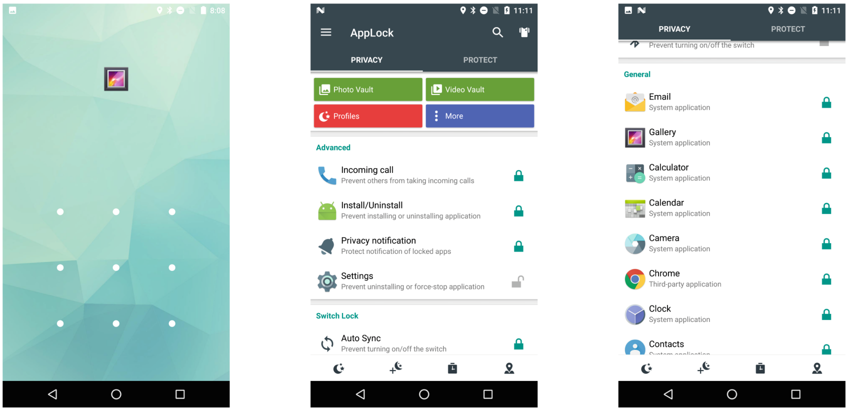
3.1 HoneyPot Account, A Simulation of the Owner's Account

The honeypot account is an Android's empty guest account with some modifications. The idea of the honeypot account is to deceive the thief into the belief that he/she is interacting on the real account. Therefore, it is important to provide the illusion of user data in the honeypot account. In principle any concept for simulating user data can be used. In our approach, the applications in the honeypot account are protected by an application called AppLock[1]. AppLock is an ordinary Android application that protects other Android applications by a locking pattern. So, every access to an app is protected by a locking pattern. The thief is under the illusion that there is some data on the device, and it is protected. This step is necessary to convince the thief that the owner's account is being used with some defense mechanism installed. Figure 4 shows the AppLock functionality. This simulates the owner's account, while it frustrates the thief and tricks him/her into performing a factory reset.

It should be mentioned that the mechanism of simulating the owner's user data is independent of the concept of a honeypot anti-theft account. An alternative would be to define some plausible but fake data that is presented whenever applications are opened in the honeypot account.

In the case of a theft, all interactions will inevitably be performed in the honeypot account. In this case, the device should be modified in a way such that it executes anti-theft functionality in the background. It is open to the users and

[1] https://play.google.com/store/apps/details?id=com.domobile.applock.



(a) AppLock protecting the Gallery application

(b) AppLock privacy setting

(c) AppLock privacy setting

Fig. 4. AppLock protection

deployers of our techniques to customize the functionality of the anti-theft application. Possible functionalities for an anti-theft application here would be the collection of information of the device and the thief, remote backup functionalities as well as other remote administration functionalities that can be performed hidden from the thief. In our scenario, we implemented the anti-theft application as an Android application that silently tracks and logs the device location.

3.2 Instrumentations

When a device is stolen, chances are high that a factory reset is performed. This is usually done to make the device more usable, and to destroy any evidence of theft. A device can either be factory reset via the Android menu or by pressing a special key combination during the boot procedure. In order to save the owner's data and keep track of the thief, the factory reset is faked in both cases. This means that the device shows a realistic simulation of a factory reset and presents an empty account after the fake factory reset. The thief is in the illusion that all potential tracking mechanisms are removed and the device can now be use in an ordinary manner. Use of the device now inevitably stores the thief's personal information, which will be forwarded by the installed anti-theft application to the device owner.

We implemented the simulation of a factory reset for both mechanisms by instrumenting the Android source code. For preventing the factory reset from the Android settings menu, we instrumented the RecoverySystem[2] and

[2] /frameworks/base/core/java/android/os/RecoverySystem.java .

RecoverySystemService[3] classes in such a way that when a user triggers the factory reset, the device shows the default factory reset animation for a realistic amount of time. We instrument the *rebootWipeUserData* function in RecoverySystem.java[2] file. In this function, a call to the *bootCommand* function is made. One of the arguments of the *bootCommand* function, specifies the intent of operation e.g., *–wipe_data* is used to wipe the data partition. Technically, when a factory reset is triggered from the honeypot account, we substitute the argument *–wipe_data* by *–wipe_cache*. This prevents the removal of the user data. Wiping the cache only removes the temporary saved files, e.g., temporary browser files. Thus, it does not affect the device owner in a negative way. Additionally for preventing data loss in case of a factory reset that was triggered during the boot process, we instrumented the recovery system[4].

During the reset, we programmatically remove the AppLock application. We instrument the *setupOrClearBcb* function in the RecoverySystemService.java file to achieve this. Since the honeypot account does not have any data, an empty account is presented to the user, that is functionally equal to a factory reset phone.

When a thief has logged into the honeypot account and potentially "factory reset" the device (which was simulated by our instrumentation), tracking information should be collected and forwarded to the device owner. In our approach, this is done by an anti-theft application. It is obvious that a thief should not notice that an anti-theft application is gathering information or even notice that it is installed. For this reason, in the honeypot account, the used anti-theft application is hidden from the list of installed applications in the settings menu as well as in the Android launcher. In Android there exist places where users can list the installed applications e.g., the Android Launcher and the settings menu in the category "Apps". We have instrumented these[5] such that the installed anti-theft application is hidden from a potential thief, and there are no possible traces of any installed anti-theft application anymore. For example, in the ManageApplications.java[5] file, we instrument the *onRebuildComplete* function such that the anti-theft application is removed from the list of displayed applications.

4 Evaluation

4.1 Evaluation Criteria

For evaluating our approach we determined a set of evaluation criteria that each determines the quality of one central aspect of our approach. We identified the following set of evaluation criteria (EvCrit) that together verify our approach:

[3] /frameworks/base/services/core/java/com/android/server/RecoverySystemService.java.

[4] /bootable/recovery/device.cpp, /bootable/recovery/recovery.cpp.

[5] /packages/apps/Launcher2/src/com/android/launcher2/AllAppsList.java, /packages/apps/Settings/src/com/android/settings/applications/ManageApplications.java.

EvCrit 1 - Simulating Factory Reset. The first advantage of our approach is that it prevents a thief from performing a factory reset on the stolen device. This has two major benefits: First, it prevents the highly valuable personal user data from being deleted. Second, it prevents an installed anti-theft application from being uninstalled. During the evaluation we encouraged the participants to factory reset the device by every way they know. Each time after a participant finished the study, we checked whether any of the device owner's data had been deleted (e.g. by the factory reset).

EvCrit 2 - Successfully Executing the Anti-theft Application. The second advantage of our approach over every other approach is that it enables the device owner to execute an anti-theft application while the device is used by the thief. For each participant we checked whether an installed anti-theft solution was successfully executed while the participant interacted with the device and even after a "fake" factory reset.

EvCrit 3 - Indistinguishability from an Uninstrumented Device. A central property of our approach is that a thief should never notice that he/she is interacting with an instrumented device. More precise: our device should be indistinguishable from a regular stock device. For this reason we checked for both of our instrumentations whether any of them was detected by the participants. This implies the following sub-evaluation criteria:

EvCrit 3a - Hiding the Faking of the Factory Reset. As mentioned, during the study we motivated the participants to perform a factory reset. For every participant that performed the factory reset, we checked whether he/she was convinced that the factory reset was actually performed or experienced any irregularities (hints on the faking of factory reset).

EvCrit 3b - Hiding the Anti-theft Application, Running in the Background. While a potential thief interacts with the device it is crucial that there are no traces of running anti-theft applications. For this reason we motivated the participants to note every protection mechanism installed on the device. We asked them whether the device can be used by a thief after a factory reset (implicitly asking for the presence of an installed anti-theft application) and motivated every participant to note every observed irregularity. As an evaluation for this subcriteria we inspect the number of participants that expressed by any means the presence of an installed anti-theft application.

4.2 Evaluation Procedure

We performed the evaluation in the form of an empirical user study. In this user study we gave each participant a Nexus 6P device that was instrumented by the implementation of ThiefTrap. As in production, our approach would be combined with any authentication mechanism for account switching, we could evaluate our approach on the main account of the device without loss of validity. Additionally, an anti-theft tracking application was installed on the device that continuously tracked the device location. Together with this smartphone, we handed out a questionnaire that asked several questions about the user's opinion of the phone. The participants were given 60 min time to complete the questionnaire.

In total we evaluated the answers of 35 participants. All of the participants were either students or university graduates. The majority of the participants were Master students, while also some PhD students and Bachelor students participated. While the participants studied various disciplines, the biggest group studied computer science or some computer science related studies (12 participants). One of the requirement to participate in the study was to have precise knowledge of the Android OS. We verified this via oral inquiry. The survey participant's knowledge ranged from average to expert.

4.3 Results

For each participant, we performed the described evaluation procedure and evaluated the answers for the mentioned questions. We inspected the device state as additional evaluation results. In the following we will discuss each of the mentioned evaluation criteria:

EvCrit 1 - Simulating Factory Reset. We checked for every participant that performed the factory reset (21 out of 35) that the factory reset did not lead to a loss of any user data. The participants triggered the factory reset via the settings menu from within the operating system, as well as during the boot process via a special key combination. In all inspected cases the deletion of user data had been prevented.

EvCrit 2 - Successfully Executing the Anti-theft Application. To evaluate whether the installed anti-theft application was successfully executed in the background, we implemented an anti-theft tracking application that tracked the location of the device. For every participant we checked whether the anti-theft application was executed and whether it successfully tracked the device during the study. The installed anti-theft application was successfully executed in every case independent of the user interaction. This result proves the robustness of our approach. It should be stressed that our approach is independent from the used anti-theft application. Instead of the used tracking anti-theft application every possible anti-theft application can be silently executed using our approach.

EvCrit 3 - Indistinguishability from an Uninstrumented Device. As described, we enquired for the both places where participants could potentially detect the instrumentation, whether we successfully hide the instrumentation from the users.

EvCrit 3a - Hiding the Faking of the Factory Reset. For evaluating whether we could convince the participants that a factory reset was actually performed (while in reality it is just faked) we asked the following question to the participants: *"Do you think, it is possible for this person [a malicious person e.g. a thief] to completely reset the phone, in order to wipe all the owner's data, e.g. to sell the phone?"*

20 of the participants answered that they were able to perform a regular factory reset and so the device can be used by a thief. 11 participants answered that they were not able to factory reset the device as they did not know how to do it, but persons with more technical knowledge can (or could potentially) reset the device. 3 participants answered that it would not be possible to factory reset the device. When orally asked about their answers after the study, all of the participants answered that they did not know that the possibility of a factory reset exists.

One participant was not convinced of the factory reset. Due to limitations of the used AppLock application, this participant managed to access the apps before the factory reset. Thus he detected the inconsistency after the factory reset and was not convinced of the factory reset. This problem was not caused by our factory reset instrumentation but, by an implementation flaw of the used locking application. We would like to stress that the locking application is not part of our scientific contribution, but a tool we used in order to deceive a thief to believe that he/she is interacting with the real user account of the device owner. This mechanism can be replaced by every other mechanism or application that fulfills this requirement (e.g. simply filling the honeypot account with fake information).

EvCrit 3b - Hiding the Anti-theft Application, Running in the Background. A core feature of our approach is to hide the existence of an installed anti-theft solution. It is necessary that a thief is not aware that an anti-theft application is running on the device (even after triggering the factory reset). For this reason, we directly asked in the questionnaire whether the study participants were able to detect any protection mechanism in the device (*"Do you think that there is a protection mechanism installed to protect the user's data? Please explain."*) In their responses to this question the 33 out of 35 asked participants answered that the only protection mechanism that is used in the device is the locking application. One participant answered that there is no protection mechanism used. Due to the mentioned limitations of the AppLock app, it was possible for one participant to disable the AppLock app, and enter the protected applications. This participant found these applications empty and implied that there is a second protection mechanism present. First, it should be stressed that he did not imply that there is an anti-theft solution running. Second, the mechanism that simulates the owner's account is not part of our contribution and can be substituted by any other mechanism (e.g. another locking application, operating system instrumentations or simply filling the honeypot account with fake data).

Lastly, we provided space in the questionnaire to the participants where they could provide additional comments. We encouraged the participants to note every unusual observation. None of the participants noted that they observed an anti-theft application, tracking application or similar application running in the background.

To summarize the evaluation results, we found every evaluation criteria very satisfactory fulfilled. Evaluation criteria 1 and 2 were fulfilled in every case.

For evaluation criteria 3a just one participant out of 35 did recognized the fake factory reset. For evaluation criteria 3b only one out of 35 participants implied that another protection mechanism is in use while he did not detected the anti-theft application. Both of these cases were caused by an implementation flaw of the used locking application. As mentioned, this locking application is not part of our contribution and can be substituted by any other protection mechanism.

5 Discussion

The proposed concept of an anti-theft honeypot account is novel. It provides a combination of valuable security properties that are not given by any existing approach. These security properties are the maintenance of user data (preventing user data from being deleted), the confidentiality of user data (preventing a thief with physical access to the device from reading out user data) and the accessibility of the device (enabling any remote access mechanism for the owner while the device is physically under the control of the thief). While there exist various anti-theft solutions, none of them can fulfill all of these properties.

An important benefit of the proposed approach compared to existing anti-theft solutions is that *a device instrumented with our approach is indistinguishable from an ordinary device*. A thief can never tell whether he/she has stolen an ordinary device or a device instrumented with the anti-theft honeypot account. Studies [6] have shown that 34 % of Android devices are not protected by a locking mechanism, so there is no way for a thief to determine whether our mechanism is used or not. Also a noticeable proportion of devices are protected by the app locking mechanism that we use to protect user app data initially. So from the existence of a locking app, a thief can never imply the existence of an anti-theft honeypot account. Another aspect that plays a role in this matter is the flexibility of our approach. A locking app is just one mechanism for faking the honeypot account. An alternative for future work is the creation of fake user data. This data will then simulate the user data of the owner, while protecting his privacy. This data can be created either by the deployers of the anti-theft honeypot account, the users or both of them in cooperation.

Device theft is a serious problem with a rapidly growing number of reported cases. Studies [14] reported that in 2013 more than 3 million devices have been stolen. For this reason Google has taken steps to mitigate damage in case of a device theft. The two most important measures to mention here are the Android Device Protection mechanism [8] and anti-theft functionalities within the Android Device Manager [7]. The Android Device Protection mechanism requires that after a factory reset, a user logs into the device with the credentials of his primary Google Account. As a thief can not know these credentials, even after a factory reset, the stolen phone is of no use. The Android Device Manager can be used to track a phone and to remotely wipe. Compared with the combined usage of these two native Android tools, our approach has two advantages: First, it prevents the deletion of user data. Nowadays, a plenitude of valuable user data is stored on modern smartphones. In the vast majority of cases, the user data on such phones is hard to recover or even irreplaceable. In contrast to the mentioned Android tools, our approach can prevent the loss of this data. The second advantage is that any anti-theft application and functionality can be executed while the device is stolen. In contrast, the Android Device Manager just supports tracking and wiping functionalities.

Physical access to a device enables a number of novel attacks, so called hardware based attacks. In the context of Android smartphones prominent examples of these attacks are that by Cannon and Bradford [4] and the work of Ossmann and Osborn [16]. Cannon and Bradford used a so called white card, a special SIM card that authorizes flashing of a custom ROM on a device. Among others, from such a ROM it is possible to read out user data. Also Ossmann and Osborn proposed a hardware based attack with which it is possible to read out user data. By connecting to the Micro USB connector via UART with TTL logic they could connect to an integrated debugger, which enabled them to activate the Android Debugging Bridge functionality and so gain access to the device. Relating to our work it should be mentioned that in principle such hardware attacks might also be possible on devices with our instrumentations in place. Still, it should be stressed that hardware attacks like these require expert knowledge of the used hardware technologies and an existing vulnerability in the smartphone device. It is unlikely that both of these factors apply in the average case of a stolen device and so the impact of hardware based attacks on our proposed approach is negligible.

Another factor that should be discussed in the context of hardware attacks is the confidentiality of user data saved on a SD card in the device. While the AppLocking mechanism protects the confidentiality of user data on the SD card from access within the smartphone, the SD card can also be extracted from the device and read within another device. To protect the confidentiality of user data in this scenario it is necessary to encrypt the content of the SD card. Such an encryption of the SD card is orthogonal to our approach and can be implemented as completion to this approach.

We implemented our changes into the AOSP (Android Open Source Project). Additionally, our changes are portable, generic and thus can be easily integrated with every vendor specific build.

The usability of a device instrumented with our approach may differ from the normal device in terms of logging into the real account. For devices with a fingerprint scanner, the usability is not affected because the login procedure to the real account is equal to an ordinary login. Devices without a fingerprint scanner will require users to enter a pattern in the Honeypot account. This pattern can be configured by users. It is similar to unlocking a device using a pattern or a PIN. Hence, the usability impact is negligible.

Lastly, we would like to stress the flexibility of this approach. The concept of an anti-theft honeypot account can be applied orthogonally to any existing anti-theft mechanism. The honeypot account is responsible for protecting the user data while simultaneously any anti-theft mechanism is implemented. The benefit of the proposed concept is that in contrast to other approaches where a thief will quickly factory reset the device, in this approach it is likely that he/she will even establish an internet connection and so enabling the remote access for the installed anti-theft solution.

6 Related Work

While the idea of a honeypot account for theft protection is novel, there has been extensive research in other theft protection mechanisms in Android. In case of a theft it is likely that a potential thief will remove the SIM card from the device and factory reset it. At the time of this writing, no existing anti-theft mechanism can protect the user's privacy, maintain his data and keep good chances that the device will be found again at the same time.

Dhanu et al. [21] created a software for theft protection. After the theft protection software was installed and configured, it waited for a replacement of the SIM card. Once the SIM card was replaced, it started collecting video material, location information, and sent them via MMS to a phone number previously configured by the device owner.

Also Shetty [20] proposed an anti theft software that was triggered by a replacement of the SIM card. Whenever this event occurred, the software sent a notification SMS to a preconfigured number. From that point of time, it was possible for the device owner to retrieve the current location of the device via a SMS request.

Chouhan et al. [5] created a theft protection software in the form of a web based remote administration tool. Over a web interface the device owner could request the current location of the device. The software also enabled the device owner to record voices of the thief, wipe his/her private data and read the web history from the thief. In addition, the software notified the device owner about replacements of the SIM card.

The work of Al Rassan and Al Sheikh [1] proposed an anti-theft system that was supported by SMS. After being activated by a specially crafted SMS, an application, that was previously installed on the stolen phone could either broadcast its location or lock personal data that was stored on the phone. This data included media and log files, as well as SMS and MMS records.

Kuppusamy et al. [15] proposed a system for theft protection that could also be used as simple remote administration tool. Via SMS messages it was possible to locate the device, erase critical data, trace calls, manage incoming SMS and system access.

Yu et al. [27] proposed a system for remotely wiping stolen phones. This system worked in a way that a device owner could register his/her phone at the emergency call service provider. He could now from any point of time report the phone as stolen to the emergency call service provider. Additionally a background application was installed on the phone. Whenever the SIM card of the phone was removed, the application sent a wipe request to the emergency call service provider. If the device was stolen, the emergency call service provider answers this request with a modified call declined request, after which the phone would be wiped by the application. This scheme had the benefit that it neither requires a WIFI connection, nor an inserted SIM card to trigger the device wiping. However, the personal data and device was lost.

In all of the mentioned work, the thief has initial access to the user data until the anti-theft mechanism is triggered either automatically after a certain event occurs or manually by the device owner. The even more severe limitation of the mentioned approaches is that none of them prevents resetting the device. For this reason in each of this work, after the thief has triggered a factory reset, the anti-theft mechanisms are deleted and there are no chances that the device owner can regain his/her device.

Apart from academic work, there exist commercial solutions for locating and securing a lost or stolen device. Examples for these products are Apple's "Find My Device" [2], Avast's "Free Mobile Security" [3] or Symantec's "Norton Anti-Theft" [24]. These solutions can lock the device, locate it and provide additional functionality for mitigating damage in case of loss or theft. Additionally to the limitations mentioned previously, these solutions suffer from permission restrictions that are imposed by the Android Security Model. These restrictions lead to severe flaws. This insight is supported by the work of Simon and Anderson [22] that have examined various mobile anti-virus solutions for Android. They discovered failures in the implementation of the remote lock and wipe functionalities of these applications. Beside the restrictions imposed by the Android Security Model, they see the reasons for these flaws in certain vendor customizations.

Schneider [19] developed a password manager that uses a similar approach for another domain. This password manager returns fake password information when a wrong master password is used.

Srinivasan and Wu [23] proposed a mechanism that primarily prevented the smartphone from being turned off or being silenced in case of a theft. This approach was implemented by protecting the called functionality with passwords. Additionally, their proposed mechanism could wipe the device after a certain amount of failed password guesses. They rely on a password for preventing the thief accessing sensitive data. This measure is good for protecting the user's privacy but will at the same time trigger the thief to factory reset the device. After the factory reset, the anti theft mechanism will be deleted and there will be no chances for the owner to regain his/her device.

Another approach for protecting the privacy of user data in the case of a device theft was proposed by Tang et al. [25]. In their approach sensitive user data was encrypted and saved in the cloud. Their goal was to minimize the amount of sensitive data that was stored on the phone. The access to this cloud storage could now be restricted by any means, such as access rate limits, complete blocking as soon as the device was reported stolen and logging access. This approach focuses on protecting sensitive data of the user. Unfortunately, it could not help the owner regain his/her device.

7 Conclusion and Future Work

In this work we have proposed ThiefTrap, a novel concept using a honeypot account for the purpose of theft protection. Using this concept it is possible to protect sensitive user data while retaining high chances to regain the device. Our novel approach is the first that can achieve this combination of desired properties. We implemented our approach as modifications on the latest version of the Android operating system, the most used operating system in mobile devices at the time of this writing. Based on this implementation we successfully evaluated our approach in an empirical user study including 35 participants. The results of our study show that for a user it is not possible to distinguish the honeypot account from a regular unlocked device. Additionally we could retrieve information of the participants that in a real world application could be used to regain the device. It should be mentioned that the proposed concept is universal and can be customized on various scenarios and platforms.

While this approach is an important step in the development of anti-theft mechanisms, it can be further extended in the future. Potential extensions include the protection of alternative storages of private user data, such as the SIM card and the device settings. In our approach, these storages were excluded, as the SIM card is rarely used today for storing personal information and the device settings do not contain highly sensitive information. Still, there are some users that would like also to protect these places. A further point of future work is the creation of alternative mechanisms that simulate the owner's account data from within the honeypot account. These mechanisms can include the automatic or semi-automatic generation of fake data.

Acknowledgements. This work was supported by the German Federal Ministry of Education and Research (BMBF) through the project SmartPriv (16KIS0760).

References

1. Al Rassan, I., Al Sheikh, M.A.: Securing application in mobile computing. Int. J. Inf. Electron. Eng. **3**(5), 544 (2013)
2. Apple: Find my iPhone, iPad, iPod touch, or Mac. www.apple.com/support/icloud/find-my-device/
3. Avast: Avast free mobile security, June 2017. http://www.avast.com/en-us/free-mobile-security
4. Cannon, T., Bradford, S.: Into the droid: gaining access to Android user data. In: DefCon Hacking Conference (DefCon 2012), Las Vegas, Nevada, USA (2012)
5. Chouhan, J.G., Singh, N.K., Modi, P.S., Jani, K.A., Joshi, B.N., et al.: Camera and voice control based location services and information security on Android. J. Inf. Secur. **7**(03), 195 (2016)
6. CNBC: CNBC study. http://www.cnbc.com/2014/04/26/most-americans-dont-secure-their-smartphones.html
7. Google: Android device manager, June 2017. https://www.google.com/android/devicemanager

8. Google: Android device protection, June 2017. https://support.google.com/nexus/answer/6172890
9. Google: Android internals, February 2017. https://source.android.com/source/index.html
10. Google: Android open source project, June 2017. https://source.android.com
11. Google: Android OS version usages, January 2017. https://developer.android.com/about/dashboards/index.html
12. Google: Android version usage, 6 July 2017. https://developer.android.com/about/dashboards/index.html
13. IDC: Worldwide smartphone OS market share, November 2016. http://www.idc.com/promo/smartphone-market-share/os;jsessionid=6A0934D1434A49DBFFE74D63DA2C595B
14. Insider, B.: IDG research, May 2014. http://www.businessinsider.com/smartphone-theft-statistics-2014-5?IR=T
15. Kuppusamy, K.S., Senthilraja, R., Aghila, G.: A model for remote access and protection of smartphones using short message service. arXiv preprint arXiv:1203.3431 (2012)
16. Ossmann, M., Osborn, K.: Multiplexed wired attack surfaces. In: BlackHat USA (2013)
17. Reports, C.: Consumer reports, May 2014. http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm
18. Ruddock, D.: Anti-theft. http://www.androidpolice.com/2015/03/12/guide-what-is-android-5-1s-antitheft-device-protection-feature-and-how-do-i-use-it/
19. Schneider, D.M.: iMobileSitter, March 2014. http://www.imobilesitter.com/
20. Shetty, A.: Mobile anti theft system (MATS) (2012)
21. Dhanu, S., Shaikh, A., Barshe, S.: Anti-theft application for Android based devices. Int. J. Adv. Res. Comput. Commun. Eng. (2016)
22. Simon, L., Anderson, R.: Security analysis of consumer-grade anti-theft solutions provided by Android mobile anti-virus apps. In: 4th Mobile Security Technologies Workshop (MoST). Citeseer (2015)
23. Srinivasan, A., Wu, J.: SafeCode – safeguarding security and privacy of user data on stolen iOS devices. In: Xiang, Y., Lopez, J., Kuo, C.-C.J., Zhou, W. (eds.) CSS 2012. LNCS, vol. 7672, pp. 11–20. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35362-8_2
24. Symantec: Norton mobile security. https://us.norton.com/anti-theft/
25. Tang, Y., Ames, P., Bhamidipati, S., Bijlani, A., Geambasu, R., Sarda, N.: CleanOS: limiting mobile data exposure with idle eviction. In: Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), pp. 77–91 (2012)
26. Whitwam, R.: Anti-theft (2015). http://www.greenbot.com/article/2904397/everything-you-need-to-know-about-device-protection-in-android-51.html
27. Yu, X., Wang, Z., Sun, K., Zhu, W.T., Gao, N., Jing, J.: Remotely wiping sensitive data on stolen smartphones. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 537–542. ACM (2014)