Tap-Wave-Rub: Lightweight Malware Prevention for Smartphones using Intuitive Human Gestures

Haoyu Li[1], Di Ma[1], Nitesh Saxena[2], Babins Shrestha[2], and Yan Zhu[1]
[1]University of Michigan-Dearborn
{haoyul,dmadma,yanzhu}@umd.umich.edu
[2]University of Alabama at Birmingham
{saxena,babins}@cis.uab.edu

ABSTRACT

We introduce a lightweight permission enforcement approach – *Tap-Wave-Rub (TWR)* – for smartphone malware prevention. TWR is based on simple human gestures (implicit or explicit) that are very quick and intuitive but less likely to be exhibited in users' daily activities. Presence or absence of such gestures, prior to accessing an application, can effectively inform the OS whether the access request is benign or malicious. In this paper, we focus on the design of an *accelerometer-based phone tapping detection* mechanism. This implicit tapping detection mechanism is geared to prevent malicious access to NFC services, where a user is usually required to tap her phone with another device. We present a variety of novel experiments to evaluate the proposed mechanism. Our results suggest that our approach could be very effective for malware prevention, with quite low false positives and false negatives, while imposing no additional burden on the users. As part of the TWR framework, we also briefly explore explicit gestures (finger tapping, rubbing or hand waving based on proximity sensor), which could be used to protect services which do not have a unique implicit gesture associated with them.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks—*General, Security and Protection*

Keywords

malware; mobile devices; NFC; context recognition; sensors

1. INTRODUCTION

Smartphones are undoubtedly becoming ubiquitous. They are not only used as (traditional) mobile phones for phone calling and SMS messaging, but also for many of the same purposes as desktop computers, such as web browsing, social networking, online shopping and banking. Also, smartphones are incorporating more and more sensors and communication interfaces. Such new capabilities enable smartphones with many new unique functionalities that desktop computers lack. For example, many smartphones are beginning to incorporate Near Field Communication (NFC) chips [15], which allows short, paired transactions with other NFC-enabled devices in close proximity. The use of NFC-equipped smartphones as payment tokens (such as Google Wallet) is considered to be the next generation payment system and the latest buzz in the financial industry [4].

Due to their popularity, smartphones are becoming a burgeoning target for malicious activities. There has been a rapid increase in mobile phone malware targeting different smartphone platforms [9, 10, 21, 14, 19]. Newer functionalities of smartphones only make them more attractive to malware writers. For example, the incorporation of NFC chips on smartphones provides malware authors another (possibly much easier) way to deploy their attacks through the NFC interface [20]. Especially, due to the ease with which financial transactions can take place using NFC, it is predicted that NFC will become a popular target for malware aiming at credential and credit card theft [11]. Indeed, a proof-of-concept Trojan Horse electronic pickpocket program under the cover of a tic-tac-toe game has already been developed by Identity Stronghold [2]. In this attack, the game containing the malware is downloaded and installed on a NFC-enabled smartphone. Once activated (when the game is played), the malware accesses the NFC chip and enables the RFID (Radio Frequency ID) reading functionality. This reader then surreptitiously scans tags (e.g. RF tagged credit card) around it and reports the acquired information to the malware owner through e-mail once a victim tag is found in proximity.

While the security community has been battling with PC malware for many years, malware detection on smartphones turns out to be an even more challenging problem [3]. This is partially due to the resource constraints of smartphones (especially limited battery power). Thus, existing malware defenses for desktop computers cannot be applied directly on the smartphone platform. Much of the existing research focuses on optimizing desktop based defenses for mobile phones [25, 23, 24, 6, 3].

In practice, to protect mobile phones from malware attacks, major mobile phone manufacturers, such as Google, Apple, and Nokia, employ permission models to prevent malware from being installed at the first place. However, this approach relies upon user diligence and awareness, while most computer users lack these traits in practice. Instead of relying on user permissions, smartphone manufacturers also rely upon application review before releasing to people for download. However, application review process can be cumbersome and prone to human error [3].

1.1 Motivation and Rationale

We argue that existing malware defenses, without considering the special characteristics of smartphone malware and that of smart-

WiSec'13, April 17-19, 2013, Budapest, Hungary.

phones themselves, might not be sufficient to detect sophisticated malware, such as the pickpocket malware targeting NFC mentioned previously.[1] First, the pickpocket malware [2], under the cover of tic-tac-toe, is quite stealthy. Its surreptitious scanning may not cause substantial changes (such as sharp increase in the number of emails sent or in power consumption) to the normal behavioral profile and therefore behavioral detection schemes will not be effective. Moreover, most existing malware detection schemes employ a *posteriori approach*. That is, malicious attacks are detected after they took place as traces need to be collected and trained before they can be compared with profiles to find abnormalities. Because of the sensitive (financial) nature of the NFC service, it is quite risky to adopt such a *posteriori* detection approach. Instead, it is desired to develop a preventive approach which can constantly monitor, identify, and then stop such potentially malicious activity *before* it is launched so as to minimize damage or loss.

This motivates us to design a novel approach for malware prevention through contextual awareness. Our rationale is as follows. Smartphones are personal devices. That is, the end user is a human being. Thus, (legitimate) access to sensitive/valuable services such as premium calls, SMS or NFC usually involves different types of human activities such as dialing a phone number, typing a message, or clicking an application icon on the screen (to execute the application). In contrast, one common pattern followed by malware found on mobile phones is that it attempts to access sensitive services without the user's awareness and authorization (thus user activity is very unlikely to be involved). Therefore, one way to detect such unauthorized, thus potentially malicious behavior, is to validate *whether an action is initiated by pure software or purposefully by a human user*.

Since legitimate access to sensitive services usually involves different types of hand movements, we explore the use of gestures to differentiate between pure software and human-initiated activities. In particular, in this paper, we propose *Tap-Wave-Rub (TWR)*, a lightweight malware detection mechanism for smartphones based on intuitive human gesture recognition, using sensors already available on current smartphones with little or no additional user involvement.

The proposed gesture-based detection mechanism serves as an extension to the currently adopted permission model used by major smartphone OSs. That is, whenever a sensitive service is requested, a particular gesture needs to be detected (to make sure it is a human generated activity) before the request can be granted. Otherwise, the activity is very likely generated by malware. As gesture detection is enforced every time a sensitive request is received, the proposed mechanism provides continuous monitoring of sensitive resources and services from unauthorized access attempts by malware. We note, the latest Android Jelly Bean 4.2 has an added security feature, Premium SMS Confirmation, that includes a giant list of premium shortcodes for each country and alerts a user anytime an app tries to send a message to a shortcode [1]. Our TWR permission model follows a similar approach but the security decision is based on presence or absence of gestures.

1.2 Our Contributions

The main contributions of this paper are summarized as follows.

1. We propose TWR, a novel approach for malware prevention with an exclusive focus on the smartphone platform based on intuitive gesture recognition. As part of this system, we propose a implicit light-weight *phone tapping detection mechanism based on accelerometer data*, which is geared for NFC applications where a user is usually required to tap her phone with another device.

2. We outline how Tap-Wave-Rub can reside within the *kernel-level middle layer* between sensitive services and applications trying to access these services, and be integrated specifically with the existing *Android permission model*. This TWR-enhanced permission model provides continuous enforcement of access control to sensitive resources and services even after an application is installed on the platform.

3. To evaluate our approach, we conduct experiments to simulate the behavior of malware and normal user usage activity. Our experiment results show that the proposed mechanism can successfully detect malicious attempts to access sensitive services with high detection rates, while imposing minimal usability burden.

4. As part of the TWR framework, we also explore lightweight explicit gestures based on proximity sensor data, that could broadly appeal to many applications (e.g., SMS). These include tapping or rubbing a finger near the top of phone's screen or waving a hand close.

2. RELATED WORK

The most closely related work to ours is the one proposed in [5]. It shares similar philosophy as ours. It utilizes whether there are hardware interruptions to differentiate pure software initiated action and human initiated action [5]. It aims at detecting malware specifically targeting SMS and audio services. These services usually start with user's pressing or touching the keypad or touchscreen which generate hardware interruptions for each key/screen press event. A purely software generated activity (or malware generated activity), on the other hand, will not explicitly generate a hardware interrupt. Although this approach is believed to be effective for malware detection, it cannot help detect a more sophisticated malware such as the pickpocket malware. This is because the pickpocket malware gets activated by user's playing the tic-tac-toe game, which already involves touch screen activity that can generate hardware interrupts. The difference between [5] and our work can be summarized as follows. [5] attempts to check whether there is (any) user activity whereas our goal is to check whether there is a **special user-aware** activity. So our approach provides more fine-grained access control to sensitive services and thus can detect even sophisticated malware.

Another work that parallels to ours was recently presented in [22]. It proposes an approach of user-driven access control by granting permission to the application when user's permission granting intent is captured. It introduces access control gadgets (ACGs) which are UI elements exposed by each user-owned resource for applications to embed. The user's authentic UI interaction with corresponding ACG grant the permission to an application to access the corresponding resource. A fundamental difference between [22] and our work is that the design proposed by [22] grants the permission to an application when user's authentic **UI interaction** with corresponding ACG is captured whereas our design grants permission to an application when a specific user **gesture** (tapping/waving) is captured. The design proposed by [22] not only requires *kernel level changes* but also necessitates *application level modifications*. It also requires Resource Monitor (RM) to be incorporated for each resource such as the device drivers. Moreover, it

[1]Throughout the paper, we will center our malware mitigation design based on properties observed from the pickpocket malware [2]; however, our approaches, being fundamental in nature, will be applicable to a broad range of future malware.

requires additional composition ACG (C-ACG) along with composition RM if an application requires different resources to be accessed/used. Our work, in contrast to [22], has an advantage in that it neither requires application level changes nor requires resource monitor to be added for each resource. Note that if there are many resources that can be used by an application, then the number of C-ACG and C-RM will become extremely large. Another advantage of our work over [22] is that our design supports "services" (such as NFC) which do not have any specific UI elements or ACGs associated with them. For the approach of [22] to work with services like NFC, additional ACG for UI interaction will need to be added, which will significantly hamper the usability of such services. In contrast, in our case, implicit permission granting intent is acquired by capturing the tapping gesture.

Gesture recognition has been extensively studied to support spontaneous interactions with consumer electronics and mobile devices in the context of pervasive computing. Due to the uniqueness of gestures to different users, personalized gestures have been used for various security purposes. Gesture recognition has been used for user authentication to address the problem of illegal use of stolen devices [12, 7]. [18] reports a series of user studies that evaluate the feasibility and usability of light-weight user authentication based on gesture recognition using a single tri-axis accelerometer. Gesture recognition is also used to defend against unauthorized reading and Ghost-and-Leech relay attacks in RFID systems [8, 13].

3. BACKGROUND

3.1 Threat Model

In our model, we assume that the mobile phone is already infected with malware. As in the pickpocketing attack of [2], the malware could reside within a benign looking application (e.g., a game) which the user may have downloaded from an untrusted source. Our model covers a broad range of malware and does not impose any restriction on malware behavior except that an action from the malware is not human-triggered. For example, the malware may want to access a service or resource (such as NFC, SMS or GPS) available on the phone itself, or to communicate with an external entity, such as an attacker-controlled remote server (botmaster).

We assume that the OS kernel is healthy and immune to malware infection. In particular, the malware is not able to maliciously alter the kernel control flow. Also, the phone hardware is assumed to be malware-free. Specifically, we assume that the malware can not manipulate the input to, and output from, the phone's on-board sensors.

We do not impose any restriction as to how frequently the malware attempts to access a given service. However, in order to remain stealthy, constantly attempting access would not be feasible for the malware, and rather random or periodic sampling is expected.

In addition to the user space level control of the phone, the malware may collude and synchronize with an entity in close physical proximity of the phone (and its user). This external entity may attempt to manipulate the physical environment in which the phone is present or interfere with the user per se. We do not, however, allow this attacker to have physical access to the phone. That is, if the attacker has physical access to the phone, then he can lock/unlock a resource just like the phone's user. In other words, our mechanisms are not meant for user authentication and do not provide protection in the face of loss or theft of phone.

3.2 Design Goals

For our malware prevention approach to be useful in practice, it must satisfy the following properties:

- ***Lightweight-ness:*** The approach should be lightweight in terms of the various required resources available on the phone, such as memory, computation and battery power.
- ***Efficiency:*** The approach should incur little delay. Otherwise, it can affect the overall usability of the system. We believe that no more than a few seconds should be spent executing the approach.
- ***Robustness:*** The approach should be tolerant to errors. Both the False Negative Rate (FNR) and False Positive Rate (FPR) should be quite low. A low FNR means that a user would, with high probability, be able to execute an application (which accesses some sensitive services) without being rejected. Low FNR also implies a better usability. On the other hand, Low FPR means that there should be little probability to grant access to a sensitive service when a user does not intend to do so. Low FPR clearly implies a little chance for malware to evade detection.
- ***Usage Model Consistency:*** The solution should require little, or no change, to the usage model of existing smartphone applications. Ideally, if the use of a particular phone service can be commonly associated with a particular (unique) gesture (e.g., phone tapping for NFC), this gesture may be used to specially protect the said service. In this case, no changes to the adopted usage model will be necessary. It is also possible that there is no unique gesture pattern that can be found to use a certain service (e.g., Bluetooth). In such a situation, an intuitive gesture template can be associated with that service and a user will be required to explicitly perform the hand movements defined by that gesture. In this case, only minor changes to the adopted usage model will be imposed.

4. TWR-ENHANCED PERMISSION MODEL

Permission models have become very common on smartphone operating systems to provide access control to sensitive services for installed third party application. The Android platform has the most extensive permission system and poses to become a market leader. Thus, we base the design of our TWR system on the Android platform.

The idea of the TWR system is to add another layer of permission check before the original Android permission check. As stated in Section 3.1, we assume the adversary is not able to maliciously alter the kernel control flow. So gesture detection forms a trusted path with the OS. Intercepted permission requests are handled by the five components in the TWR's architecture: *TWR PermissionChecker*, *TWR GestureManager*, *TWR GestureExtractor*, *TWR TemplateCreator*, and *TWR GestureDatabase*. The architecture of TWR is depicted in Figure 1. In the following paragraphs, we present the role of each TWR component by describing the possible interaction between them and the outside world.

The *TWR PermissionChecker* stands in front of the original Android Permission check. When an application initiates a request to access a sensitive service, the request is intercepted by *TWR PermissionChecker*. This component interacts with *TWR GestureManager* to check whether the requested service is protected by a certain gesture. If not, the request is forwarded to the Android Permission Check as usual. Otherwise, *TWR GestureManager* interacts with the *TWR GestureExtractor* to begin collecting gesture

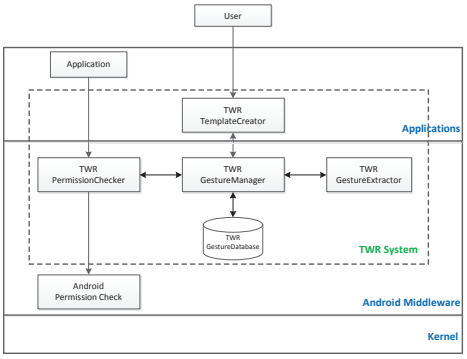

Figure 1: The TWR Architecture

data (tapping, rubbing, or waving in this paper). The captured data is then sent to the *TWR GestureManager* for further process.

Here we distinguish between two types of gesture recognition: user-dependent and user-independent. As their names suggest, a gesture is user-dependent if there is significant variation among gesture data from multiple participants for the same predefined gesture; while a gesture is user-independent if either there is no apparent difference or the recognition process does not differentiate among user data from multiple participants. Our phone tap gesture recognition is user-dependent as it captures the user's own features of the tapping movement. Given that users can hold a phone in different ways and use different forces to tap, the phone tap gesture is thus user-dependent. Our hand wave or finger tap/rubbing recognition scheme is user-independent as it infers user activity by checking whether a special location (in our context, the place where the proximity sensor is located) is touched or not (instead of the potentially biometric feature of human movement).

For user-dependent gesture recognition, we usually need to create a gesture template which is used as a reference in the actual recognition stage. The user can interact with the *TWR TemplateCreator* to register a new gesture template, to update and delete exiting gesture templates. *TWR TemplateCreator* is an Android application which allows interaction between TWR and the user. When the user creates, deletes, or modifies the gesture information, it needs to retrieve and store the information to *TWR GestureDatabase* via *TWR GestureManager*. *TWR GestureManager* is the only component that has access to *TWR GestureDatabase*.

So depending on the type of gesture recognition scheme (user-dependent or user-independent), the *TWR GestureManager* processes the gesture data from *TWR GestureExtractor* differently. If the gesture is user-dependent, it compares the similarity between the newly captured data with the corresponding gesture template stored in the *TWR GestureDatabase*. If the gesture is user-independent, the *TWR GestureManager* determines directly whether a gesture is performed or not by utilizing information in the captured data without the help of a template. In either case, if a required gesture is detected, the request is forwarded to Android Permission Check for further check. Otherwise, the request for service access is rejected.

5. TWR GESTURE DETECTION

As we mentioned in Section 2, the use of hardware interruption to differentiate between pure software initiated activity and human initiated activity is not effective to prevent malware hidden under the cover of a victim app, since activating the victim app already involves keypad click or screen touch which can generate hardware interrupts. This motivates us to use app-specific user events to distinguish between hidden malware and an app initiated by a human being. That is, instead of simply using general key/screen press events to infer human activity, we try to recognize whether it involves the *right* activity a user needs to do to access a sensitive service, such as the access to NFC.

A smartphone is a personal hand-held device installed with a lot of apps. Most of the time, these apps are activated by specific phone/hand movements. For example, when a user wants to place a call, she needs to unlock the screen, activates the phone app, inputs the number (or clicks on a name in the contact list), and then puts the phone near the ear to start the call. Also, to use the NFC to scan a smart poster, a user needs to unlock the screen, activates the NFC reader app, and taps the phone on the smart poster to read information. Since "tapping" (touching the phone against an RFID tag, or another NFC device) is a gesture which users commonly need to perform to use the NFC functionality, as an illustrative example, we can use tapping to determine whether an NFC access is human-initiated or not. Intuitively, tapping on a smart-poster should be different from other user phone activities (such as keyboard click or screen touch) and user physical activities (such as walking or running).

One advantage of this tapping approach is that it does not require any additional user activity besides what is being used commonly, and thus transparently recognizes user activity when a user taps a smart poster to obtain information. However, it may exhibit false positive rates and not fully prevent the pickpocket malware activation since normal user activities (such as playing the game) may generate motions similar to tapping. To achieve higher prevention rate, we can try other intuitive user-aware gestures similar to tapping, such as "tapping twice" or "tapping thrice" in succession.

To recognize tapping, we utilize the on-board accelerometer data. An accelerometer sensor measures the forces applied to the phone (minus the force of gravity) on the three axes: x, y, and z. Let (a_x, a_y, a_z) denote the values corresponding to the 3 axes from the accelerometer.

Our detection algorithm consists of two phases: training phase and recognition phase. In the training phase, a user performs the target action (tapping) multiple times, and accelerometer data of the action is recorded and processed to generate a tapping template. The template serves as a reference to be compared later with real-time user movement data: a match indicates the recognition of user tapping activity; otherwise, it is inferred that either there is no user activity or the activity is not the "valid" user activity to grant NFC access.

After the training phase, the system compares a newly observed movement with the template. The system records the accelerometer data, from the moment the user activates the NFC reader app, until she taps on a smart poster. To recognize tapping, the system computes the cross-correlation C of the acceleration data A against the template T, both of size n data points as shown in Equation 1.

The cross-correlation C computed from Equation 1 when comparing two time series is a real value, representing a similarity measure. The higher the value of C, the higher the similarity between the two series. The maximum value is obtained when the two series under comparison are identical. A movement is considered a valid tapping activity when the computed cross-correlation C exceeds a certain cross-correlation threshold which is usually obtained through empirical study.

$$C(A,T) = \frac{\sum_{i=1}^{n}(a_{x_i} - \bar{a}_x)(T_{x_i} - \bar{T}_x)}{\sqrt{\sum_{i=1}^{n}(a_{x_i} - \bar{a}_x)^2}\sqrt{\sum_{i=1}^{n}(T_{x_i} - \bar{T}_x)^2}} = \frac{\sum_{i=1}^{n}(a_{y_i} - \bar{a}_y)(T_{y_i} - \bar{T}_y)}{\sqrt{\sum_{i=1}^{n}(a_{y_i} - \bar{a}_y)^2}\sqrt{\sum_{i=1}^{n}(T_{y_i} - \bar{T}_y)^2}} = \frac{\sum_{i=1}^{n}(a_{z_i} - \bar{a}_z)(T_{z_i} - \bar{T}_z)}{\sqrt{\sum_{i=1}^{n}(a_{z_i} - \bar{a}_z)^2}\sqrt{\sum_{i=1}^{n}(T_{z_i} - \bar{T}_z)^2}} \quad (1)$$

where $\bar{a}_x$ denotes the means of time series a_{x_i} for $i \in [1, n]$ (others follow the same notation).

Here we describe one way to determine the cross-correlation threshold. Suppose we have m traces of tapping movements T_1, ..., T_m. The threshold C_T can be estimated as the minimum cross-correlation between any two series T_i and T_j ($i, j \in [1, m]$ and $i \neq j$). That is:

$$C_T = min_{i,j=1}^{m}(C(T_i, T_j)) \quad (2)$$

The aforementioned phone tapping detection mechanism is geared for NFC applications. Unlike NFC, most other services/resources on the phone may not be associated with a unique implicit gesture. In such situations, an explicit human involvement would be necessary. As part of the TWR framework, we have also designed and implemented lightweight explicit gestures based on proximity sensor data, that could broadly appeal to many applications (e.g., SMS). These include tapping or rubbing a finger near the top of phone's screen or waving a hand close. We note the design of an explicit gesture detection scheme is not trivial. The challenge is to keep the gestures very simple for the users to perform, and lightweight for the system to identify. We report the details of our explicit gestures in the extended version of this paper [16].

6. IMPLEMENTATION AND EVALUATION

To evaluate the feasibility of the TWR approach for malware prevention, we developed a prototype application on the Android platform. The phone tapping detection scheme was implemented and installed on a Google Nexus S Android phone. This Nexus phone comes equipped with NFC chip, therefore a good target device for an eventual deployment of our approach.

In this section, we report on evaluation of tapping based user activity recognition scheme outlined in Section 5. This scheme is specially designed to protect against malware targeting NFC reading services, since tapping is a natural hand movement which a user needs to perform to use the reader function of NFC.

Since "tapping" (touching the phone against an RFID tag, or another NFC device) is a very simple hand movement, we hypothesize it might be confused with other user movements such as those users perform when they play games, and thus have higher false positive rate, FPR (or lower prevention rate). In a hope to achieve higher prevention rate, we also experiment with two other intuitive user-aware gestures similar to tapping: tapping twice and tapping thrice in succession. We call these three tapping gestures as "tapping once", "tapping twice", and "tapping thrice", respectively.

First, to determine the cross-correlation detection threshold, we collected 30 traces of accelerometer data for each tapping gesture. Each of our trace contains 100 data points and is recorded over a 2-second time period (we wanted our schemes to be efficient). Each trace is then used as a template, which is compared with all the other 29 traces to calculate a serial of C values. The smallest C value is chosen as the threshold value. This threshold value is then stored with the corresponding tapping template and a matched posture needs to yield a C value larger than this threshold. These traces were collected by the experimenter while performing NFC tapping gesture 30 times. Such data collection and testing methodology is in-line with related prior security work, e.g., Secret Handshakes [8]. Our methodology captures a realistic usage scenario whereby each user can be trained "once by their phone" and can create their template, e.g., when they purchase the phone.

We first test the performance of the three tapping gestures to identify which one can have higher recognition rate (thus lower false negative rate, FNR). To do this, we collect a total of 150 traces for each tapping gesture, 30 traces every day for 5 days. We then use the template and the threshold calculated above to determine the recognition rate. The successful recognition rate is listed, in the form of a confusion matrix, in Table 1. It shows that "tapping once" achieves high recognition rate 94.67% (or a low FNR 5.33%) compared to the other two tapping gestures.

We next test the performance of the three tapping gestures to identify which one is the least to be confused with other user or phone movements and thus has low FPR. It is important to evaluate the FPR. If a tapping gesture can be very similar to a certain other movement (accidental or manipulated by an attacker), the malware may circumvent the gesture detection process.

To determine the FPR, we compare tapping postures with many phone/user movements. These movements might be just normal user activities, or activities coerced by a nearby attacker. They include user movements such as: walking, walking stairs, screen-touch activities (text messaging and surfing Internet), phone-moving activity (motion gaming and picking up calls), as well as, the scenarios where phone is left still. For each movement, we also collect a total of 150 traces, 30 traces every day for 5 days. The error rates are all listed in Table 1.

Our experiment result shows "tapping once" is very unlikely to be confused with walking, walking stairs, still, and screen-touch activities such as text messaging or Internet browsing. However, it might occasionally be confused with phone motion caused when the user plays game or picks up a phone call with a false positive rate of 2%. "tapping twice" and "tapping thrice", on the other hand, are very resilient to phone motions but they resemble motions when a user walks on stairs. Nevertheless, all achieve satisfying low false positive rate.

One potential reason why the false positive rate is low might be that tapping is a type of user-aware movement. When performing such a gesture, the user is believed to be aware of her hand movement. So gestures are performed in a more-or-less controlled way, e.g., the phone is always held in the similar way when a user performs tapping. In non-user-aware movements, on the other hand, the phone can be tilted in any position. The reference template is usually collected in a reference coordinate system. However, once the phone is tilted, movement data collected from the device is no longer in the reference coordinate system and the corresponding movement will not be detected correctly. In this way, user-aware gesture is very unlikely to be similar with user-unaware movements, and thus has low false positive rate. Previous studies on gesture recognition also suggest certain gestures can be quite unique and different from other gestures [8, 17]. So tapping can be

Table 1: Tapping Detection Results (rates at which a gesture shown on each row matches with gesture/activity shown on each column

	Tapping Once	Tapping Twice	Tapping Thrice	Walking	Walking Stairs	Still	Screen-touch Activities	Phone Movement
Tapping Once	94.67%	NA	NA	0%	0%	0%	0%	2%
Tapping Twice	NA	92.67%	NA	0%	1.33%	0%	0%	0%
Tapping Thrice	NA	NA	96.67%	0%	5.33%	0%	0%	0%

distinguished from other user-aware movements such as "picking up the call".

Our experiment result, contrary to our hypothesis that "tapping once" may have high false positive rate, shows that "tapping once" actually achieves both high recognition rate and low false positive rate, and has a performance comparable to "tapping twice" and "tapping thrice". However, "tapping once" outperforms the other two tapping gestures in term of efficiency, and has better usability since it does not require any change to the usage model of NFC.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a lightweight permission enforcement approach – Tap-Wave-Rub (TWR) – for smartphone malware detection and prevention. TWR is based on simple human gestures that are very intuitive but less likely to be exhibited in users' daily activities. Specifically, we presented the design of aphone tapping detection based on accelerometer data. This mechanism is geared for NFC applications, which usually require the user to tap her phone with another device. In addition, we present the TWR Android permission model, the prototypes implementing the underlying gesture recognition mechanisms, and a variety of novel experiments to evaluate them. Our results suggest the proposed approach could be very effective for malware prevention, with quite low false positives and false negatives, while imposing little to no additional burden on the users. The false negatives are expected to further reduce significantly as users become more familiar with the underlying gestures, especially since they are quite intuitive. In addition, the false positives can also be carefully avoided in most cases, for example, by detecting the orientation of the device. Our future effort will be focused on realizing the TWR approach in practice and further evaluate it with a wide range of smartphones and smartphone users.

Acknowledgements: We thank William Enck and WiSec'13 anonymous reviewers for their thoughtful feedback.

8. REFERENCES

[1] R. Amadeo. Exclusive: Android 4.2 alpha teardown, part 2: SELinux, VPN lockdown, and premium SMS confirmation. Available online at `http://www.androidpolice.com/2012/10/17/exclusive-android-4-2-alpha-teardown-part-2-selinux-vpn-lockdown-and-premium-sms-confirmation/`, Oct. 2012.

[2] W. Augustinowicz. Trojan horse electronic pickpocket demo by identity stronghold. Available online at `http://www.youtube.com/watch?v=eEcz0XszEic`, June 2011.

[3] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani. Crowdroid: Behavior-based malware detection systems for Android. In *ACM CCSW Workshop*, 2011.

[4] M. Calamia. Mobile payments to surge to $670 billion by 2015. Available online at `http://www.mobiledia.com/news/96900.html`, Jul. 2011.

[5] A. Chaugule, Z. Xu, and S. Zhu. A specification based intrusion detection framework for mobile phones. In *ACNS'11*, 2011.

[6] J. Cheng, S. Wong, H. Yang, and S. Lu. Smartsiren: virus detection and alert for smartphones. In *5th International Conference on Mobile Systems, Applications and Services (MobiSys'07)*, 2007.

[7] M. Conti, I. Zachia-Zlatea, and B. Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *ASIACCS'11*.

[8] A. Czeskis, K. Koscher, J. Smith, and T. Kohno. RFIDs and secret handshakes: Defending against Ghost-and-Leech attacks and unauthorized reads with context-aware communications. In *ACM Conference on Computer and Communications Security*, 2008.

[9] F-Secure. Bluetooth-worm:symbos/cabir. Available online at `http://www.f-secure.com/v-descs/cabir.shtml`.

[10] F-Secure. Worm:symbos/commwarrior. Available online at `http://www.f-secure.com/v-descs/commwarrior.shtml`.

[11] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild. In *ACM CCSW Workshop*, 2011.

[12] D. Gafurov, K. Helkala, and T. Sndrol. Biometric gait authentication using accelerometer sensor. *Journal of Computers*, 1(7):51–59, 2006.

[13] T. Halevi, S. Lin, D. Ma, A. Prasad, N. Saxena, J. Voris, and T. Xiang. Sensing-enabled defenses to rfid unauthorized reading and relay attacks without changing the usage model. In *PerCom'12*, 2012.

[14] J. Han, E. Owusu, T.-L. Nguyen, A. Perrig, and J. Zhang. ACComplice: Location Inference using Accelerometers on Smartphones. In *Proc. of COMSNETS*, Jan. 2012.

[15] ISO. Near field communication interface and protocol (nfcip-1)——iso/iec 18092:2004. Available online at `http://www.iso.org/iso/catalogue_detail.htm?csnumber=38578`, 2004.

[16] H. Li, D. Ma, N. Saxena, B. Shrestha, and Y. Zhu. Tap-wave-rub: Lightweight malware prevention for smartphones using intuitive human gestures. Extended Technical Report, Available online at `http://arxiv.org/abs/1302.4010`, Feb. 2013.

[17] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–575, December 2009.

[18] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. User evaluation of lightweight user authentication with a single tri-axis accelerometer. In *MobileHCI'09*, 2009.

[19] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proc. of ACM CCS*, 2011.

[20] C. Mulliner. Vulnerability analysis and attacks on NFC-enabled mobile phones. In *1st International Workshop on Sensor Security (IWSS) at ARES*, 2009.

[21] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. ACCessory: Keystroke Inference using Accelerometers on Smartphones. In *Proc. of HotMobile)*, Feb. 2012.

[22] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *33rd IEEE Symposium on Security and Privacy (Oakland 2012)*, 2012.

[23] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K. Yksel, S. Camtepe, and A. Sahin. Static analysis of executables for collaborative malware detection on Android. In *ICC 2009 Communication and Information Systems Security Symposium*, 2009.

[24] A. S. Shamili, C. Bauckhage, and T. Alpcan. Malware detection on mobile devices using distributed machine learning. In *20th International Conference on Pattern Recognition (ICPR'10)*, 2010.

[25] D. Venugopal. An efficient signature representation and matching method for mobile devices. In *WICON'06*, 2006.