Wave-to-Access: Protecting Sensitive Mobile Device Services via a Hand Waving Gesture

Babins Shrestha, Nitesh Saxena and Justin Harrison

Computer and Information Sciences
University of Alabama at Birmingham
{babins,saxena,justinh}@uab.edu

Abstract. Mobile devices, such as smartphones and tablets, offer a wide variety of important services to everyday users. Many of these services (such as NFC payments) are highly sensitive and can be abused by malicious entities, without the knowledge of the device user, in the form of insider attacks (such as malware) and/or outsider attacks (such as unauthorized reading and relay attacks).
In this paper, we present a novel application permission granting approach that can be used to protect any sensitive mobile device service. It captures user's intent to access the service via a lightweight hand waving gesture. This gesture is very simple, quick and intuitive for the user, but would be very hard for the attacker to exhibit without user's knowledge. We present the design and implementation of a hand waving gesture recognition mechanism using an ambient light sensor, already available on most mobile devices. We integrate this gesture with the phone dialing service as a specific use case to address the problem of malware that makes premium rate phone calls. We also report on our experiments to analyze the performance of our approach both in benign and adversarial settings. Our results indicate the approach to be quite effective in preventing the misuse of sensitive resources while imposing only minimal user burden.

1 Introduction

The deployment and usage of mobile devices, such as smartphones and tablets, is continuously rising. These devices open up immense opportunities for everyday users offering valuable resources and services. In addition to traditional capabilities, such as voice calling, SMS and web browsing, many smartphones come equipped with the NFC (Near Field Communication) functionality, a form of RFID (Radio Frequency IDentification). An NFC phone can be used as a RFID contactless payment token, such as a credit or an ATM card. It can also be used as an RFID reader that can "read" other RFID cards or NFC phones in close physical proximity. NFC equipped devices, such as Samsung Galaxy Nexus and Nexus 7 are already in the US market. All these different features on modern mobile devices have attracted not only millions of consumers to smartphones and tablets but have also motivated the developers to write varieties of apps for these devices, such as the Google Wallet app for NFC-based payment using Android phones.

1.1 Security and Privacy Threats

Due to the increasing popularity of mobile devices, they have become a prime target of malicious attackers and cyber criminals. In particular, such malicious entities attempt to misuse the sensitive services provided by these devices. For example, they might be interested in making free (premium rate) phone calls or sending free SMS using a legitimate user's phone, or want to make NFC payments which will be charged to the user's account. Specifically, two major threats exist that form the primary focus of this paper: (1) the insider attacks in the form of malware, and (2) the outsider attacks in the form of NFC unauthorized reading, as explained below.

Mobile Device Malware: There has been a rapid increase in mobile device malware targeting different smartphone platforms [15, 17, 16, 7, 33, 22, 28, 30]. This is made possible especially because users often download applications from untrusted sources, many of which may contain hidden malicious code. Such malware, once installed on the smartphone, can exploit the smartphones in many different ways. For example, the malware can use the resources/sensors of the phone to learn something sensitive about the user, such as it can use the camera and take the pictures of the user and surrounding, or it can make premium rate phone calls or send premium rate SMS messages without user's knowledge, or use NFC reader to skim for physical credit cards in close proximity. 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].

Unfortunately, current operating systems (e.g., Android and iOS) provide inadequate security against these malware attacks. For granting permission to an application requiring access to the resources, these operating systems either require out-of-context, uninformed decisions at the time of installation via manifest [3, 32] or prompt users to determine their interest via system prompt [29, 32]. This approach relies upon user diligence and awareness – it is well-known that most users do not pay attention to such "Yes/No" prompts and frequently just select "Yes" so as to proceed with the installation. Once granted the permission, applications have full authority over the resources and can access them without owner's consent. In addition to relying upon user permission, application review process is also undertaken. However, review process has failed in the past [40, 23], and users gaining the root permission/ jail-breaking the phone can easily install the third party applications which may not have been reviewed [40, 23].

NFC Unauthorized Reading and Relay Attacks: The NFC (tag) chip on a smartphone stores sensitive information. In particular, it stores the credit card number and other relevant information. Such an information can easily be subject to clandestine eavesdropping. For example, an adversary with an NFC reader can walk past a victim carrying an NFC phone, and can read the credit card information stored on the NFC chip. This clearly allows for fraud or illegitimate purchases for which the owner will be charged. It can also lead to owner tracking and privacy problems [1]. This information may also be used to impersonate an NFC device via cloning [1].

Furthermore, similar to RFID tags, NFC devices are susceptible to "ghost-and-leech" relay attacks [42]. Here an adversary, called a "ghost," relays the information surreptitiously read from a legitimate RFID device to another colluding adversary, called a "leech." The leech transmits this information to a legitimate reader and vice versa, and can thus impersonate the RFID tag. All cryptographic authentication protocols are vulnerable to this form of an attack [18].

1.2 Controlling Access via Intuitive Gestures

In this paper, we set out to defend against the aforementioned insider attacks (malware) and outsider attacks (unauthorized reading and relay attacks) against critical mobile device services. We observe that all these attacks, in order to remain stealthy, occur in scenarios where the device user has no intention to access the underlying services. Thus, if the user's intent to access the services can be captured in some way, these attacks could be prevented. We propose to elicit user's intent via simple gestures performed by the user prior to accessing the services. In other words, whenever the user wants to access the service, she will simply perform a particular gesture. On the other hand, if the attacker attempts to access the service, the gesture will be missing and the access request will be blocked. This general idea was first introduced in a recent (short) paper [27] limited to address the specific problem of NFC malware-based attacks. In the current paper, we extend the scope of the approach to cover NFC unauthorized reading and relay attacks as well as introduce novel gesture recognition schemes for malware-based attacks against any sensitive service.

Access control using a simple gesture has significant usability advantages when compared to typing in a complex password, which are often forgotten, and has significant security advantages over using nothing at all. The approach is also *more secure than using a "Yes/No" dialog box*, given that most users are already *habituated* to pressing "Yes" when prompted. Hand waving gesture is only one of the gestures for human-enabled authorization of actions, and there can be a number of gestures in the future, which when used carefully can overcome the problem of habituation of pressing "Yes." Moreover, a Yes/No dialog box will require the user to *explicitly press a button* when an application requires access to a resource while the user is performing another activity on the device. The gesture-based approach, on the other hand, would not require a user interface, but rather the user can just be prompted using a "Toast" mechanism, available for example in Android, or notified using a notification bar.

1.3 Our Contributions

We propose how a gesture-based mechanism can be used to elevate the permission for applications which require access to critical mobile device resources and services (see Figure 2). The main contributions of this paper are summarized as follows.

1. We propose a novel approach to malware defense for mobile devices based on intuitive gesture recognition. Specifically, we suggest a new lightweight hand waving gesture that utilizes the ambient light sensor and accelerometer, ubiquitously available on smartphones and tablets.
2. We also introduce the use of the waving gesture for the purpose of selective unlocking for NFC/RFID tag. Instead of promiscuously providing the information to any reader, we argue for a model which only approves the permission to read the tag's information once the gesture is detected.
3. We report on the implementation of our prototypes for the wave gesture recognition scheme on the Android platform. As a specific use case for defending against malware that makes *premium rate phone calls*, we integrate this gesture recognition scheme with the phone's voice dialing service. The resulting app requires the user to perform the wave gesture before making a phone call (or a phone call to a premium rate number) .
4. To evaluate our approach, we conduct many experiments simulating the behavior of malicious attacker and normal user usage activity. Our results demonstrate the waving gesture to be quite effective in protecting critical mobile device services without imposing much burden on the user.

2 Related Work

2.1 Malware Detection and Prevention

There is a plenty of prior work on defending against malware on traditional desktop computers. Static analysis[10, 38, 36], also known as signature-based detection, is based on source or binary code inspection to find suspicious patterns (malware) inside the code. However, malware authors can evade this analysis by simple obfuscation, polymorphism and packing techniques. Also it cannot detect zero day attacks. Dynamic analysis[14, 39, 5, 41], also known as behavior-based detection, monitors and compares the running behavior of an application (e.g., system calls, file accesses, API calls) against malicious and/or normal behavior profiles through the use of machine learning techniques. It is more resilient to polymorphic worms and code obfuscation and has the potential to defeat zero-day worms.

These techniques for desktop computers are still considered too time consuming for resource-constrained mobile devices operated on battery. Most existing research focuses on optimizing desktop solutions to fit on mobile devices. The work of [38] tries to speed up the signature lookup process in static analysis by using hashes. Several collaborative analysis techniques have been proposed to distribute the work of analysis by a network of devices [34, 37]. Remote server assisted analysis techniques have also been proposed to reduce the overhead of computation on individual devices [9, 6].

2.2 Gesture Recognition and Security

Gesture recognition has been extensively studied to support spontaneous interactions with consumer electronics and mobile devices in the context of pervasive computing [4, 8, 25]. 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 misuse of stolen devices [19, 11]. In [19], a mobile device gets unlocked for use when it detects the gait (walking pattern) of the legitimate owner. In [11], a smartphone gets locked when it does not detect the "picking-up phone" gesture which the owner naturally performs to answer a phone. Both works provide transparent user authentication and do not require explicit user involvement. [26] 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.

Related to our work, gesture recognition has also been suggested to defend against unauthorized reading and ghost-and-leech relay attacks in RFID systems [12, 21]. The secret handshakes scheme proposed in [12] allows an RFID tag to respond to reader query selectively when the tag owner moves the tag in a certain pattern (i.e., secret handshake). In contrast to our hand waving gesture recognition, secret handshakes requires a pre-stored template, and the underlying gestures themselves may not be very intuitive for the user. The work of [21] uses posture as a valid context to unlock an implanted RFID device without changing the underlying user usage model. This approach, however, can have high false positive rate in practice and is not applicable in the context of mobile devices that are not implanted.

The use of unique key press gestures or secure attention sequences (SASs), such as CTRL-ALT-DEL, may also serve as a means to defend against malware and unauthorized reading attacks. However, we are not aware of SASs being currently used on mobile phones. SASs need to be unique and usually require multiple key presses simultaneously (e.g., CTRL-ALT-DEL). Such sequences will be very hard for the user to perform on phones. The hand waving gesture proposed in our paper can be viewed as a form of novel and user-friendly SAS suitable for phones.

Recently published paper [20] is also relevant to our work. Like ours, it focuses on the hand-wave gesture but utilizes reflected sound waves instead of light. Specifically, it applies "Doppler Effect" to sense hand waving gestures. It uses speaker to generate inaudible sound waves and microphone to receive the reflected frequency-shifted wave. The approach involves calculating the frequency of the received signal to infer various gestures such hand wave, double-tap, and two-handed seesaw movements. Based on these "in-air" gestures, the paper suggests different non-security use cases and applications. It might be possible to use this gesture recognition scheme for the purpose of protecting sensitive mobile device services. However, there are a few caveats. First, an adversary can send inaudible sound waves to a victim's smartphone with varying frequency so as to mimic one of the gestures. This will undermine the security of this scheme. Second, the sound wave generated by the speaker can be annoying for children and pets who can hear the high frequency sound waves. Furthermore, compared to our gesture recognition mechanism, this approach is computationally-intensive, and power-exhaustive, for mobile phones.

3 Threat Model and Design Goals

3.1 Threat Model

In our security model, we consider both insider attacks and outsider attacks against mobile devices. In the context of an insider attack, the attacker is assumed to be a malicious application or malware. We assume that attackers use malware to access sensitive services (such as phone call, SMS, NFC, or GPS) for various malicious intentions. The malware can be hidden in a normal application. Malware can spread through various paths to the phone via various communication channels such as Bluetooth, WiFi, and GSM. We assume that the malware has already been downloaded and installed on the phone without any user suspicion. This can happen, for instance, when a user downloads an application from an untrustworthy source that looks like a game but contains malicious code. How to prevent malware from being installed on the phone is beyond the scope of our model.

In the context of an outsider attack against the NFC service running on a mobile device, the attacker is a device that can read the contents stored on the NFC chip and can later use this information for illegitimate purposes. The attacker could also constitute two colluding entities who can launch a ghost-and-leech attack against NFC.

We assume the mobile device OS kernel itself is healthy and immune to malware infection; hardening the kernel is an orthogonal problem [35, 31]. So the malware is not able to maliciously alter the kernel control flow. The malware is not able to alter data values of on-board sensors too. Otherwise, the malware can supply fake sensor data to escape detection. However, the action from malware is neither human triggered nor can it maliciously alter the kernel control flow.

We assume the attacker may be physically near the user. The attacker is unable to persuade the user to perform the gesture to access a particular service. However, she may coerce/fool the user into moving a particular manner with a hope that such movement can generate similar motion as a valid gesture. 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 security approach to be useful in practice, it must satisfy the following properties:

– The approach should be *lightweight* in terms of the various resources available on the phone, such as memory, computation and battery power. A biometrics-based approach does not satisfy our goal as it is not lightweight and can be time-consuming [24].
– 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.
– 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 a 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 a 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 malicious entity to evade detection.
– The solution should *require little changes to the usage model* of existing smartphone applications. An intuitive gesture should be required from the user that may involve simple hand movements defined by that gesture. In this case, only minor changes to the adopted usage model will be imposed.

4 Hand Wave Detection

The primary sensor that we use to detect the hand waving gesture is an ambient light sensor, commonly available on smartphones and tablets.[1] A light sensor measures the intensity of ambient light. The light intensity is measured in *lux* which defines how bright or dark the surrounding environment is. The primary reason the light sensors are deployed on smartphones, and tablets and laptops, is for prolonging battery life. The brightness of the screen display of the phone is adjusted according to the intensity of the surrounding light measured by the light sensor. For example, in a dark environment, the display is dimmed which helps reduce the battery consumption. This is the reason the light sensor is located in the front of the mobile phone at the top of its display (see Figure 1). This is true for most, if not all, smartphones and tablets, including the Androids and iPhones. We note that this is a property that we carefully leverage in developing our waving gesture mechanism aimed at improving the security of mobile phones. Specifically, our gesture interfaces with the light sensor, and due to the location of the light sensor, it does not interfere with the gestures made by the user while interacting with the device's (touch screen) display, thereby significantly reducing the False Positive Rate (FPR).

In order to utilize the light sensor for our purpose, we needed a human gesture that can "trigger" the sensor in some way and is not likely to be exhibited in daily activities. We chose waving (depicted in Figure 2) as a simple and a convenient gesture mechanism since it can be easily executed by a human user

[1] A proximity sensor may also be used to detect a hand wave as suggested in [27], but tablets do not commonly come equipped with this sensor.

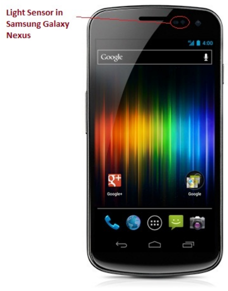

Fig. 1. Location of the Light Sensor (Samsung Galaxy Nexus)

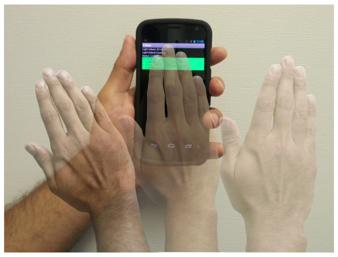

Fig. 2. Permission Granting via a Hand Waving Gesture. In order to access a sensitive service, the user is asked to simply wave her hand in front of the phone; this unlocks the service for use. A malicious application, in contrast, would fail to exhibit such a gesture and will not be able to access the service.

and easily detected by a smartphone. The algorithm to detect quick fluctuations in the reading of the light sensor is very simple and straightforward, and, unlike many other gesture recognition algorithms, do not even need any pre-established templates. This makes our approach extremely lightweight, satisfying one of our design goals (Section 3.2).

To detect the hand wave gesture, whenever there is a change in reading from the light sensors, we record the light sensor readings along with their respective timestamps. We then analyze this light data and time recorded to determine the fluctuation in the light intensity. If the light value fluctuates beyond a given threshold for certain number of times within an allocated time, then we consider such fluctuations in light values as being triggered by the hand wave gesture. The threshold to determine if the light has fluctuated depends upon the current ambient light intensity. When it is dark, i.e, the ambient light intensity is below 200 lux, then using a threshold of 20 lux is optimal to detect the fluctuation as per our measurements. However, when ambient light is around 60,000 lux, i.e, in the presence of bright sunlight, optimum value of threshold is around 15,000 lux.

We used eight different thresholds for eight different ranges of light intensity to accurately determine the wave gesture.[2] For the analysis of fluctuation, we used certain number of light readings $WINDOW\_SIZE\_FOR\_LIGHT$ (16). If we detect the change in light intensity beyond the $LIGHT\_THRESHOLD$ (20 – 15,000 lux), then we add up the light change count ($extremaCount$). After analysis of $WINDOW\_SIZE\_FOR\_LIGHT$ readings of data, if the light change count is greater than $CHANGE\_COUNT\_FOR\_LIGHT$ (6) and all the light under analysis is within $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ (2

[2] All the thresholds and range buckets were determined through active experimentation.

Algorithm 1 Wave Detection using Light Sensor (and Accelerometer)

```
1: IF sensors are locked THEN wait for $MOVEMENT\_LOCK\_TIME$
   ELSE get accelerometer sensor readings x, y and z.
2: IF
       $\sqrt{x^2 * y^2 * z^2} > ACC\_THRESHOLD$
   THEN lock the sensors for $MOVEMENT\_LOCK\_TIME$ and RETURN to step 1.
3: IF sensors are not locked THEN get light sensors reading to check if wave gesture is detected.
    1. Analyze $WINDOW\_SIZE\_FOR\_LIGHT$ data to find out how many extremas (maximas and minimas) were there using $LIGHT\_THRESHOLD$.
    2. IF $extremaCount > CHANGE\_COUNT\_FOR\_LIGHT$ AND All the light data are recorded within $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ THEN
       SET $unlockAttempted = true$,
       RECORD first unlock attempted time
       DISPLAY Message "Stop Waving" for $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$.
    3. IF $unlockAttempted$ THEN
        (a) IF another unlockAttempt is obtained within less than $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ THEN Do not unlock, reset everything and start over, i.e., return to Step 2.
        (b) IF another $unlockattempt$ is not obtained within $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ THEN $Unlock$ the phone for $UNLOCK\_TIME\_FRAME$.
```

seconds; the maximum duration for the hand wave gesture), then we determine it as a wave gesture. However, sometimes environmental effects may trigger the light sensors to detect it as wave gestures. So instead of unlocking the phone straightaway (i.e., allowing access to the requested service), we delay the unlock for certain time $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$. If no gesture is detected within this timeframe, then we unlock the phone for certain time $UNLOCK\_TIME\_FRAME$ (1 second).

In our scheme, the light sensor data is used in conjunction with the accelerometer data to detect the wave gesture. The accelerometer sensor is used for the purpose of reducing FPRs. Whenever the phone is moved, there will be a relative change in the position of the phone with respect to the light source triggering a change in light intensity. This will in turn be detected as a wave gesture, leading to a high FPR (since the user did not wave in front of the phone). In order to reduce this effect, if the phone detects movement, greater than a certain threshold ($ACC\_THRESHOLD$), as per the accelerometer data, it does not register it as a hand wave gesture; further it locks the light sensor as well. Thus, when an application requests for the permission to access the resource/service, the algorithm will first check if the sensors are locked. If the sensors are locked, then the algorithm will wait for certain time ($MOVEMENT\_LOCK\_TIME$) before it starts reading the light and accelerometer signal again. Note that when the algorithm is first executed, both sensors are active, and the corresponding data is read. A detailed pseudocode for this simple procedure is outlined as Algorithm 1.

5 Implementation and Evaluation

5.1 Test Prototype: Wave-to-Call

To evaluate the feasibility of the wave gesture detection mechanism, we developed our prototype in Android Operating System using Motorola Droid X2. The project build target was chosen for Android 2.3.3 platform or above using API level 10. A simple UI was created to emulate the unlocking of a service using the hand waving gesture. The goal of the prototype was to add a layer of security to Android permission model where user needs to provide a gesture to use a resource/service. In our prototype, we specifically asked user to wave their hand in front of the phone as a gesture to make an *outgoing call*. If the gesture is not received within 10 seconds, the application will not allow a call.

We created a service that intercepts all the outgoing calls and two activities, one for turning on/off the service and another for receiving the wave gesture. When the service is turned on, whenever there is an intent to make a phone call, our service will intercept that intent and start the activity to receive the gesture. This activity turns the sensors on and reads the sensor value to analyze it using our gesture recognition algorithm (Algorithm 1). If the readings from the sensor satisfy the algorithm, then it will return true and provides an approval token to make a call. If the sensor data does not satisfy the algorithm within certain time duration, then the service will shut down the activity waiting for gesture along with the sensors, i.e., sensors are only activated whenever there is an intent to use a service and not all the time. These steps while making a call are shown in Figure 3.

The service to intercept outgoing calls is turned on when user sets his preference via one of the activities mentioned above. However, the service must also be turned on when the device boots up from being shutdown. We utilized the BroadcastReceiver from Android SDK for this purpose. This is explained in Figure 4.

Our approach does not involve any modification to the Android OS, rather we created an application with about 500 lines of Java Code to turn on the service as well as intercept the intent and make a call. Adding another intent intercept will require few additional lines of code. Android OS provides limited number of intents that we can intercept. Available intents can be found in [13].

5.2 Hand Wave Detection Experiments and Results

In this section, we report on the evaluation of our hand wave gesture recognition scheme. This scheme is designed to protect against the malware and other malicious entities trying to access phone's resources or services without user awareness. We conducted several experiments to evaluate our prototype implementing

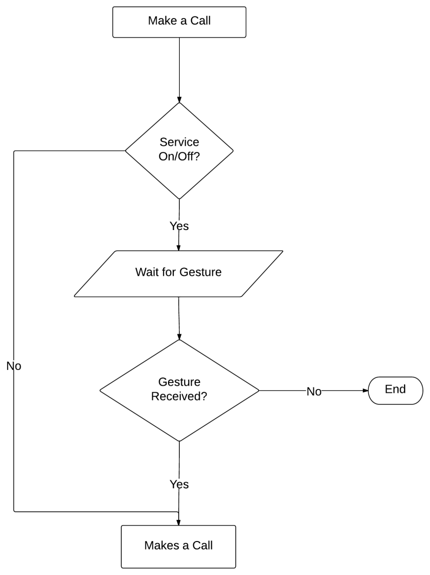

Fig. 3. Process while making a call

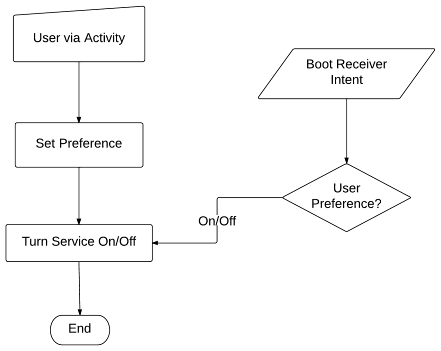

Fig. 4. Turning the service on/off. Service can either be turned on by user via setting up the preference or when the device boots up after checking the user preference

the hand-wave detection mechanism. The goal of our tests was to primarily estimate the error rates, i.e., FNR and FPR. A summary of our experimental results is depicted in Table 1, and the details are explained below.

FNR Experiments: In order to determine the FNR, one of the authors attempted to unlock the phone himself using the hand waving gesture. Out of forty trials performed, the user failed to unlock the device only 3 times. Most of these failures occurred when the user tried to wave the hand far away from the phone (farther than 30 cm), resulting in an average recognition rate of 92.5% (or FNR of 7.5%).

Since multiple trials may have trained this user significantly, likely leading to a bias, we further conducted our tests with multiple other users. These volunteers were drawn from our Department (Computer Science) and were mostly students at undergraduate and graduate levels. The users were first explained the purpose of the study and then demonstrated the gestures using which they were to unlock a service on the phone. The users were specified the location of the light sensor on the phone. Although in real life, users may not be aware of the location of the light sensor, they can be easily provided with this information using a simple interface. For example, an arrow pointer could be provided on the screen which points to the light sensor, and user could be asked to execute the wave gesture accordingly.

A total of 20 volunteers participated in our study. Each of them was requested to perform the hand waving based unlocking procedure 10 times and the results were recorded automatically by our program. The resulting average recognition

Table 1. Recognition Rates for Hand Waving Detection. White cells: hand waving attempts correctly detected as hand waving; Gray cells: other activities falsely detected as hand waving

Activity	Hand Wave	Phone Unlock Count	Light Readings Change (Attempt) Count
Hand Wave (> 700 lux)	**95.71%**	**67**	**70**
Hand Wave (350-700 lux)	87.00%	87	100
Hand Wave (< 350 lux)	83.33%	25	30
Hand Wave (average conditions)	**90.50%**	**181**	**200**
Walking, phone in backpack	0.00%	0	155
Walking with phone held in hand	**0.08%**	**4**	**5138**
Walking, phone in pocket	0.00%	0	103
Car, phone on passenger seat	0.00%	0	7098
Car, phone on dashboard	**0.15%**	**14**	**9053**
Routine Usage	0.00%	1	8729
Infront of TV	**0.67%**	**3**	**449**
Watching movie in theater	0.00%	0	64
Monitor Blinking 10 times/sec	**0.32%**	**2**	**623**
Monitor Blinking 5 times/sec	**0.47%**	**3**	**633**
Monitor Blinking 3.3 time/sec	**1.15%**	**5**	**435**
Monitor Blinking 1.67 times/sec	0.00%	0	257
Monitor Blinking 1 times/sec	0.00%	0	183
Manually flickering the light 40 times/min	0.00%	0	176
Manually flickering the light 30 times/min	**0.49%**	**2**	**408**
Drop Test	0.00%	0	43
Game Playing Landscape 1	**0.44%**	**4**	**911**
Game Playing Landscape 2	**0.43%**	**4**	**925**
Game Playing O3 Portrait	**0.23%**	**2**	**871**

rate observed was 90.5% (181/200; FNR of 9.5%). Most of these FNR occurred when there was less ambient light. When the light intensity was greater than 700 lux, recognition rate was observed to be 95.71% (67/70) whereas when it was between 350 lux and 700 lux, the rate was 87% (87/100), and when it was lower than 350 lux, it was observed to be 83.3% (35/40).

In general, these error rates can be deemed to be fairly low and are in line with prior research on gesture recognition (e.g., [12, 21]). We expect them to further reduce significantly as users become more and more familiar with the hand waving gesture.

FPR Experiments: Next, we set out to evaluate the likelihood of false unlocking under different activities. These activities might be just routine user activities, or activities coerced by a nearby attacker. The experimenter conducted several tests emulating different user activities that have the potential of triggering the light sensor fluctuations. The phone recorded the number of times it has been unlocked, i.e., when the activity is recognized as a wave gesture by our algorithm, out of a total number of registered light fluctuations.

First, to simulate a walking activity, the mobile phone was stowed in a backpack and the experimenter, carrying the backpack on the shoulders, walked around for 20 minutes. No unlocking events occurred in this case, although 155 light fluctuations were observed. This experiment was repeated at a later point of time, but the phone was held in hand emulating the reading of text messages while walking (for 20 minutes). This time, the phone was unlocked 4 times, out of 5138 light fluctuations, leading to an FPR of 0.08%. The phone was unlocked when sunlight was coming from behind and shadow from the moving shoulder was partially obstructing the phone. We further continued this walking experiment for a duration of another 20 minutes but this time the phone was kept in a pocket. No unlocking events occurred even in this case.

To verify if the phone will get unlocked when it is carried by its owner inside a car, the phone was first kept on the passenger seat of a car, while the car was driven. The phone did not get unlocked in this scenario. Next, the phone was placed on the dashboard of the car underneath the windshield and the car was driven around for one hour. In this case, we noticed that the phone got unlocked 14 times out of a total of 9053 light reading changes, which yields an FPR of 0.15%. Note that once the car moves at a constant speed, the accelerometer reading will not change significantly, allowing the light sensor to detect the wave gesture. Although not frequent, in general, we can see a potential for the phone to get unlocked when there is bright light from windshield coming onto the phone, while the car passes beneath a tree. The phone was further carried in the car in the same way as above in dark (i.e. light intensity below 300 lux) for 5 minutes, but, as expected, no unlocking was observed.

Next, the phone was treated as a user's own phone for about one day (20 hours). Different routine activities were performed during this experiment, such as walking with the phone in pocket, going upstairs/downstairs, and sending messages and making/taking calls. The phone was also placed on a desk alongside the user's laptop. Only 1 unlock was registered in this case, out of a total of 8729 changes in the light values. This suggests that normal usage of the phone will only have a little likelihood of unlocking.

Another experiment was conducted to see if a bright and dynamic light source, such as a Television, can trigger the phone's unlocking. Here, the phone was placed in front of a Television, 6ft away from it, while watching a program for 1 hour. In this case, we found that the phone got unlocked 3 times, out of a total of 449 light fluctuation events, equivalent to an FPR of 0.67%. Interestingly, we also noticed that such event occurs only if the surrounding is dark enough such that the threshold is low enough for the TV to fluctuate the light intensity. Also, to unlock, the TV screen must display a bright light and flicker quickly. Extending this general experiment further, the phone was also carried to a movie theater, where it was kept outside the user's pocket while the user watched a 2 hour long movie. In this case, interestingly, no unlocking was registered.

A similar experiment was conducted using a computer monitor. Here again, we held the phone in front of a flickering monitor. We flickered the monitor with different frequencies. The phone got unlocked twice, out of 623 light change events (FPR of 0.32%), when the monitor was flickering with a frequency of 10 times/second. It unlocked thrice when the frequency was 5 times/second (633 light changes; FPR 0.47%), and five times when time interval for flickering rate was 3.3 times/second (435 light changes; FPR 1.15%). When the frequency of flickering was reduced, it could not unlock the phone since time interval exceeded $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$.

To trigger sudden fluctuations to the light sensor readings, we next conducted a "drop and fall" test. This mimicked a situation where the phone accidentally drops on, or is thrown at, a surface. Clearly, we could not just drop or throw our test device on the floor to avoid damaging it. To do this meaningfully, therefore, we first threw our test device on a bed from a height of around 1 meter. Number of trials of this test were performed for two minutes. Phone was thrown straight to the bed as well as rolled over so that there is change in the relative position of the light source. No unlocking events were recorded over this set of tests since the change in accelerometer readings was large enough to even trigger the light reading detection.

We also conducted tests to simulate a nearby adversary who may (deliberately) try to change the surrounding ambient light and unlock the phone. Although, this may create suspicion, we simulated such a scenario as the attacker would have a high incentive to exploit. For example, the attacker can flicker lights in a building which may enable all malware-infected mobile devices in that building to access the phone's resources. To do this, the light was turned on and off in the evening when the primary source of light was the fluorescent lamps. When the light was switched on/off slowly, i.e., 30 times per min, it did not unlock the phone. However, when the light was turned on/off 40 times per min, it unlocked the phone twice, out of a total of 408 light change events, leading to a FPR of 0.49%.

Finally, we analyzed a scenario where the user would be playing a game on her smartphone. We were interested in finding out the likelihood of unlocking the phone by hand movements which may trigger the light sensor fluctuation mimicking hand waving. We emulated the game play activity on the phone under portrait and landscape orientations. When the phone was held in portrait mode, fingers are far away from the light sensor and may not trigger the light sensor. However, when the phone is held in either of the landscape orientations using two hands, fingers plays a vital role in light sensor readings. The game playing activity was mimicked by our experimenter for two minutes. It unlocked the phone four times in each landscape orientation (out of 911 and 925 light changes; FPR of 0.44% and 0.45%), and twice in portrait (out of 871 light changes; FPR of 0.23%). Indeed, this confirmed our hypothesis that game play under portrait mode is less likely to unlock the phone than under landscape modes.

6 Discussion

Overall, our experiment results in previous section show that hand waving can be effectively used to infer the "right" human activity in order to unlock the use of sensitive services/resources on smartphones, thus preventing unauthorized and stealthy access by malicious entities. The low FNR (less than 10%) and short delay (up to 2 seconds) demonstrate the usability of our approach. Note that, in practice, the user will be given up to 3 attempts to perform the gesture correctly, which would mean that the effective FNR will be nearly 0%, and it will still take only a few seconds to perform the gestures. We believe that the FNR can be further reduced as users become more and more familiar with the underlying gesture. The low FPR (less than 1% in most cases), on the other hand, shows that our approach will provide a high degree of security in practice. Although there exists some potential for unlocking the phone (e.g., while the user is watching a TV or riding a car), the likelihood is extremely low. Moreover, it will be very hard for the malicious attacker to constantly wait for, and synchronize with, scenarios in which unlocking is possible, further confirming the robustness of our approach.

In the rest of this section, we further analyze and interpret the performance of our hand waving gesture recognition scheme, and discuss other relevant aspects.

Ease of Use and Convenience: Hand waving is a gesture that captures users' intent to access a mobile device service. It simply requires user to perform a simple hand movement to unlock a desired service/resource. This does require an extra user effort to access a service. However, hand wave gesture is quite intuitive and can be easily performed unlike traditional passwords and PINs which user has to memorize and input diligently, and add an extra burden when the user forgets them. Certainly, passwords and PINs are more secure compared to gestures, but we are, in this work, concerned about protection against the malware and unauthorized reading attacks rather than against theft and unauthorized usage by other person. The explicit gesture is a minimal cost required to add security against such attacks. We believe that the hand waving gesture is as easy as a "finger-swiping" gesture commonly deployed on many smartphones.

The hand wave gesture is user-independent as shown by our experiment results. That means, the service/phone can be shared by multiple users without registering his/her own template. In fact, there is no template employed in this scheme and there is no need to train the device. This scheme therefore offers a high level of convenience to the users, and might be easily adoptable. A mentioned above, it does not prevent unauthorized use of such service when the phone is stolen. However, allowing the user to change the threshold and wave time limit parameters are future modifications to personalize and detect the wave gesture accurately for a specific device owner.

Battery Consumption and Efficiency: Another important issue to cover in our work is the power consumed by the sensors while trying to capture the user gesture. Since the battery-life is one of the most important factors to be considered for user's day to day activity, our design needs to be battery-friendly.

The waving gesture proposed by our design is very short (up to 2 seconds). When there is a request for the sensitive service/resource which requires users gestures, only then the sensors will be turned on and the gesture detection algorithm will be executed. Once the kernel captures the required gesture, the permission will be granted to the active application and sensor will be turned off. If kernel fails to capture the gesture for certain duration, the sensor will be turned off and application will be denied to use the resources.

For the sake of our experiments, we have turned on the sensor all the time. This was done so as to determine the FPRs, i.e., to calculate the rate at which the algorithm will fail to provide security.

Effect of Light: Since we are using the light sensor, the ambient light plays a crucial role in the detection of hand wave gesture. We can see from our experiment that, as the surrounding light intensity decreased, the FNR increased. As soon as the light intensity drops below a certain level, the hand wave gesture will not be able to alter the light intensity even by a minimum threshold. Hence, when it is completely dark, our hand wave detection will not work. This situation can be remedied by resorting to a more complex touch screen gesture from the user, such as tapping on the screen a few times in succession, whenever the phone detects the surrounding to be dark enough.

Targeted Attacks: Our experiments demonstrate very low FPR, which means that there is little probability that an application will gain access to resources without the knowledge of its owner. This is based on an assumption that the attacker can not create the required gesture. However, a malware can fool a user by launching a social-engineering attack. For example, a malware developer can design a game such that user has to move his hand in certain ways mimicking the hand wave gesture. While such attacks are likely, they still require the malware program to constantly wait for, and synchronize with, the desired user gesture, which may make these programs easily detectable by the OS. Nevertheless, our approach still significantly raises the bar against many existing malware attacks, a prominent advancement in state-of-the-art in smartphone malware prevention.

Sensitivity of Sensors: We are using two sensors to detect the hand wave gesture, namely a light sensor and an accelerometer. There are different types of these sensors available on different devices. The frequency at which the sensor feeds the data to the kernel not only depends upon the kind of sensor but also on the processor speed, and number of application the phone is running, among other things. When we compared the sensor of our prototype device (Motorola Droid X2 running Android 2.3.3 on a Dual-core 1 GHz Cortex-A9 processor) with other devices (Samsung Galaxy Nexus (Android 4.0.3, Dual-core 1.2 GHz Cortex-A9), we found that light sensor reading on our device (Droid X2) changes quite frequently, i.e., the light sensor on this device is highly sensitive. On the other hand, the accelerometer readings changes more rapidly on the other device (Galaxy Nexus). For the hand wave gesture to be recognized accurately on a given device, the threshold and the wave time limit should be modified according to that device's configuration and the sensitivity of its sensors.

Extending to RFID Tags: Our scheme is also applicable for preventing unauthorized reading and relay attacks against standalone RFID tags (such as contactless credit cards or access cards). In this case, the RFID tag will need an on-board ambient light sensor and an accelerometer, and the user will simply need to wave in front of the tag to access it. Unlike prior security mechanisms that use sensor-equipped tags (e.g., [12, 21]), our approach is very simple and lightweight, and can be easily accommodated within the constraints of typical RFID tags.

7 Conclusion and Future Work

In this paper, we presented a novel approach to protecting sensitive mobile device services against many prominent attacks. The approach captures user's intent to access a given service via a lightweight hand waving gesture. This gesture is very simple, quick and intuitive for the user, but would be very hard for the attacker to exhibit without user's knowledge. We presented the design and implementation of the hand waving gesture using an ambient light sensor, already available on most smartphones and tablets. We also reported on our experiments to analyze the performance of our approach. Our results indicate the approach to be quite effective in preventing the misuse of sensitive resources with a very little user effort. Our future work constitutes further evaluating our approach on different devices and integrating it with services/resources beyond voice dialing, such as SMS and NFC.

Acknowledgments

The authors thank the CANS'13 anonymous reviewers for their feedback. This work was funded in part by the NSF CNS-1201927 grant.

References

[1] A. Juels. RFID Security and Privacy: A Research Survey. In *Journal on Selected Areas in Communications*, 2006.

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

[3] M. Ballano. Android threats getting steamy, 2011. http://www.symantec.com/connect/blogs/android-threats-getting-steamy.

[4] T. Baudel and B.-L. Michel. Charade: remote control of objects using free-hand gestures. *Communication of ACM*, 36:28–35, 1993.

[5] A. Bose, X. Hu, K. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In *MobiSys'08*, 2008.

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

[7] L. Cai and H. Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Proc. of USENIX HotSec*, 2011.

[8] X. Cao and R. Balakrishnan. VisionWand: interaction techniques for large display using a passive wand tracked in 3D. In *ACM UIST'03*, 2003.

[9] 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.

[10] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *12th Conference on USENIX Security Symposium*, 2003.

[11] 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 *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, 2011.

[12] A. Czeskis, K. Koscher, J. R. Smith, and T. Kohno. Rfids and secret handshakes: defending against ghost-and-leech attacks and unauthorized reads with context-aware communications. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 479–490, New York, NY, USA, 2008. ACM.

[13] A. Developers. Intent. Available online at http://developer.android.com/reference/android/content/Intent.html.

[14] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia. A behavioral approach to worm detection. In *ACM Workshop on Rapid malcode (WORM)*, 2004.

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

[16] F-Secure. Trojan:symbos/viver.a. Available online at http://www.f-secure.com/v-descs/trojan_symbos_viver_a.shtml.

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

[18] G. Hancke. Practical Attacks on Proximity Identification Systems. In *Symposium on Security and Privacy*, 2006.

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

[20] S. Gupta, D. Morris, S. Patel, and D. Tan. Soundwave: using the doppler effect to sense gestures. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, 2012.

[21] 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.

[22] 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.

[23] S. Kolesnikov-Jessop. Hackers go after the smartphone, 2011. www.nytimes.com/2011/02/14/technology/14iht-srprivacy14.html.

[24] H. Li, D. Ma, N. Saxena, B. Shrestha, and Y. Zhu. Tap-wave-rub: Lightweight malware prevention for smartphones using intuitive human gestures. *CoRR*, abs/1302.4010, 2013.

[25] 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.

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

[27] D. Ma, N. Saxena, B. Shrestha, T. Xiang, and Y. Zhu. Tap-wave-rub: Lightweight malware prevention for smartphones using intuitive human gestures (short paper). In *ACM Conference on Wireless Network Security (WiSec)*, 2013.

[28] 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.

[29] Microsoft. What is user account control?, 2011. http://windows.microsoft.com/en-US/windows-vista/What-is-User-Account-Control.

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

[31] N. L. Petroni, Jr. and M. Hicks. Automated detection of persistent kernel control-flow attacks. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 103–115, New York, NY, USA, 2007. ACM.

[32] 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 *IEEE Symposium on Security and Privacy*, 2012.

[33] R. Schlegel, K. Zhang, X. yong Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Proc. of NDSS*, 2011.

[34] 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.

[35] A. Seshadri, M. Luk, N. Qu, and A. Perrig. Secvisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 335–350, New York, NY, USA, 2007. ACM.

[36] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Inf. Secur. Tech.*, 14:16–29, Feb. 2009.

[37] 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.

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

[39] D. Venugopal, G. Hu, and N. Roman. Intelligent virus detection on mobile devices. In *PST'06*, 2006.

[40] M. Ward. Smartphone security put on test, 2010. http://www.bbc.com/news/technology-10912376.

[41] X. Liang and X. Zhang and Jean-Pierre Seifert and S. Zhu. pBMDS: A behavior-based malware detection system for cellphone devices. In *WiSec'10*, 2010.

[42] Z. Kfir and A. Wool. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. In *Security and Privacy for Emerging Areas in Communications Networks*, 2005.