Hardened Setup of Personalized Security Indicators to Counter Phishing Attacks in Mobile Banking

Claudio Marforio
ETH Zurich
claudio.marforio@inf.ethz.ch

Ramya Jayaram Masti
ETH Zurich
ramya.masti@inf.ethz.ch

Claudio Soriente
Telefonica
claudio.soriente@telefonica.com

Kari Kostiainen
ETH Zurich
kari.kostiainen@inf.ethz.ch

Srdjan Capkun
ETH Zurich
srdjan.capkun@inf.ethz.ch

ABSTRACT

Application phishing attacks are rooted in users inability to distinguish legitimate applications from malicious ones. Previous work has shown that personalized security indicators can help users in detecting application phishing attacks in mobile platforms. A personalized security indicator is a visual secret, shared between the user and a security-sensitive application (e.g., mobile banking). The user sets up the indicator when the application is started for the first time. Later on, the application displays the indicator to authenticate itself to the user. Despite their potential, no previous work has addressed the problem of how to securely setup a personalized security indicator — a procedure that can itself be the target of phishing attacks.

In this paper, we propose a setup scheme for personalized security indicators. Our solution allows a user to identify the legitimate application at the time she sets up the indicator, even in the presence of malicious applications. We implement and evaluate a prototype of the proposed solution for the Android platform. We also provide the results of a small-scale user study aimed at evaluating the usability and security of our solution.

Keywords

Mobile applications; phishing; security indicators

1. INTRODUCTION

User-application interaction provides a subtle and often very stealthy attack vector to mobile malware. Phishing applications can violate the confidentiality of user input to steal login credentials. For example, a phishing application can show a mobile banking login screen, to steal the victim's credentials. Alternatively, a malicious application can violate the integrity of other application outputs to provide the user with fabricated data. For example, a malicious application may mimic the UI of a legitimate trading application and show fabricated stock prices, to induce the user into buying or selling stocks.

Despite many security mechanisms employed on mobile devices, attacks that target user-application interaction remain an open problem [1, 8, 34]. Such attacks are mostly rooted in the user inability to distinguish a malicious application from the genuine one. In particular, application phishing attacks that target mobile banking and payment applications, have become a recurring threat with several incidents reported [6, 9, 10, 21, 29, 32].

Previous work has shown that personalized security indicators can help an alert user to detect phishing attacks on mobile devices [19]. A personalized security indicator (or *indicator* for short) is a visual secret shared between the user and an application. The first time the user starts the application, she chooses an indicator (e.g., a picture) for that particular application. Every time the application asks for the user's credentials, it authenticates to the user by showing the indicator. A user should input her credentials only if the application shows the correct indicator.

Personalized security indicators do not fully prevent phishing attacks, as a careless user may forget to verify the indicator when entering her credentials [19]. However, indicators can help users to detect otherwise stealthy phishing attacks and provide a complementary security mechanism to strengthen security-critical applications.

Confidentiality of the indicator is key to prevent phishing applications from masquerading as legitimate ones. If the indicator chosen by the user for the legitimate application is phished, malware on the device can masquerade, later on, as the legitimate application and phish the user credentials. Previous research proposals [4, 31, 34] ask the user to set up the indicator upon the first execution of the application, assuming that the adversary is not present at this time. This is typically referred to as the "trust on first use" (TOFU) assumption. In mobile platforms, the TOFU assumption is only reasonable early in a device lifetime, when no malicious applications are present on the device. In reality, users install applications at different times during the device lifetime. A phishing attack at the time when the user sets up the indicator for an application, may leak the indicator to the adversary and void any security guarantees provided by indicators.

In this paper, we propose a novel scheme to set up indicators more securely. We use mobile banking and phishing protection as a case study, but the security mechanism we propose is applicable also to other applications and can be

SPSM'16, October 24 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2994459.2994462

used to protect other forms of user-application interaction (e.g., integrity of the application output). Our solution helps an alter user to withstand malicious apps on the device at the time of indicator setup, and hence improves the security properties of indicators.

Our solution requires minor modifications to the mobile platform and to the infrastructure of the application provider (e.g., the bank). Our scheme incurs extra user interaction when the indicator is set up, but no extra burden when the application is used. We implement a prototype for the Android platform and show that the increase in the platform TCB is small.

We also present results of a small-scale user study. The majority of our 30 study participants were able to complete the indicator setup procedure, but few (4) participants fell for subtle attacks. We conclude that our solution helps an alert user to set up indicators more securely.

To summarize, in this paper we make the following contributions:

- We propose a novel scheme to securely set up personalized security indicators in the presence of malicious applications on the user's device.
- We implement a prototype of our solution for the Android platform and evaluate its performance.
- We conduct a small-scale user study that shows that majority of the users are able to complete the proposed procedure and that the procedure improves the security of indicator setup.

The rest of this paper is organized as follows. Section 2 provides background information on mobile application phishing and Section 3 explains the problem of secure indicator setup. In Section 4 we describe our indicator setup scheme. We evaluate the performance and usability in Section 5. In Section 6 we discuss our solution in a broader context. Section 7 surveys related work, and Section 8 concludes the paper.

2. BACKGROUND

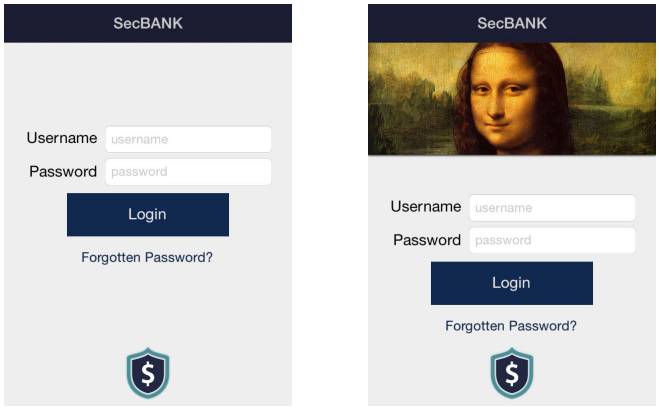
An adversary that wants to attack the interaction between a user and an application on a mobile platform can violate the confidentiality of the user input or the integrity of the application output. In this paper we focus on application phishing attack, and therefore on the protection of user input confidentiality, as such attacks are becoming increasingly commonplace [6, 9, 10, 21, 29, 32].

2.1 Application Phishing Techniques

A simple phishing technique is to clone the entire user interface of the target application. Such phishing applications are often spotted in application marketplaces. We call this phishing technique a *similarity attack*.

In a more advanced phishing scenario, malware runs in the background of the device and leverages the platform APIs to observe which application is scheduled to the foreground. When the user starts the target application, the malware activates to the foreground and presents the user with a phishing screen. Such phishing technique is called a *background attack* [8].

A malicious application can also present a button to make a payment via another app, but instead of forwarding the user to the legitimate payment or banking application, the malicious application shows a phishing screen. We refer to this phishing technique as a *forwarding attack*.



(a) (b)

Figure 1: (a) Screenshot of a banking application that does not use security indicators. (b) Screenshot of a banking application that makes use of security indicators (in this case the user chose the Mona Lisa as her security indicator)

For a more extensive discussion on known application phishing techniques, we refer the reader to [1, 8, 19].

2.2 Known Countermeasures

Several countermeasures have been proposed to the problem of mobile application phishing. However, all known countermeasure have their own limitations [19].

Signature-based code analysis can detect background attacks [1]. However, some phishing techniques, such as forwarding attacks, do not require any specific system APIs or permissions. The only precondition of successful phishing is the ability to draw on the mobile device screen.

The mobile platform could also be enhanced to show the identity of the currently running application: similar to address bars on web browsers, the mobile platform could allocate a part of its screen for the application name and developer identity [1, 28]. However, mobile devices have limited screen estate. Additionally the adversary could use similar looking application name and developer identity, as in many web phishing attacks.

Personalized security indicators are a well-known phishing countermeasure from the context of websites [4, 31]. An indicator is a visual shared secret between the user and the application. In the context of mobile application [34], the setup of the indicator happens the first time the application is started. At this time the user picks an indicator (e.g., a picture) for that particular application. From that moment on, every time the application asks for the user's credentials, it displays the indicator as a menas to authenticate to the user.

Figure 1a shows a sample banking application that does not use security indicators and, in contrast, Figure 1b shows the same application displaying the indicators chosen by the user (the Mona Lisa picture). The user should input her credentials only if the application is displaying the correct indicator. Obviously, the effectiveness of indicators relies on user alertness. While, prior studies in the context of web

phishing have shown that the majority of users forget to check the presence of the correct indicator [27], we recently evaluated the effectiveness of personalized indicators in the context of mobile banking applications and found that deployment of indicators prevented many (50%) attacks [19]. (We emphasize that the attack success rates reported in [27] and [19] are not directly comparable, due to differences in their respective studies. The goal of our recent study was not to compare or contradict previous results, but rather to evaluate the effectiveness of indicators in the new context of mobile banking applications.)

3. PROBLEM STATEMENT

Importantly, the anti-phishing benefits of personalized security indicators are void in case the indicator itself is phished. For example, the first time the user starts a banking application and chooses the indicator, she may not be able to tell if the application in foreground (i.e., the one receiving the user-chosen indicator) is the legitimate banking application or a malicious one. If the user-chosen indicator is leaked to a malicious application, that application can, later on, authenticate to the user as the legitimate banking application and phish her credentials.

In this paper we focus on how to avoid that malicious software on the device phishes the indicator that a user chooses for a legitimate application. We focus on mobile banking as it is a scenario where phishing attacks may cause severe monetary losses for the victim user or the targeted financial institution. Also, a mobile banking application asks for user credentials every time the application is started (differently from applications that keep session cookies) and, therefore, increases the chances of a phishing attack. We design our solution for the Android platform because it provides extensive capabilities for an attacker to implement phishing and because malicious apps are often found in the Android ecosystem [36].

3.1 System Model

Figure 2 illustrates the system model we consider. Mobile applications run on top of an OS that, together with the underlying hardware, constitute the device TCB. The TCB guarantees that a malicious (e.g., phishing) application cannot tamper with the execution of another application (isolated execution) or read its stored data (application-specific storage). Typically, the application active in foreground controls the entire device screen and receives all user inputs. The TCB enforces that applications running in the background do not intercept user inputs intended for the foreground application and cannot capture its output (user-interaction isolation).

Applications are primarily distributed through a centralized repository operated by the mobile platform provider, known as the marketplace (e.g., Google Play). Applications can also be downloaded directly from the developer or from third-party marketplaces. Installing applications from other sources than the main marketplace is called sideloading.

We consider a mobile banking application distributed by the bank to its customers through the marketplace. The application uses personalized security indicators and the user must set up an indicator before being able to use the application. Regular mail is normally used by banks as a secure and authentic channel to transfer PIN codes and other credentials to their customers. We, therefore, assume the availability of an out-of-band-channel between the bank and the user.

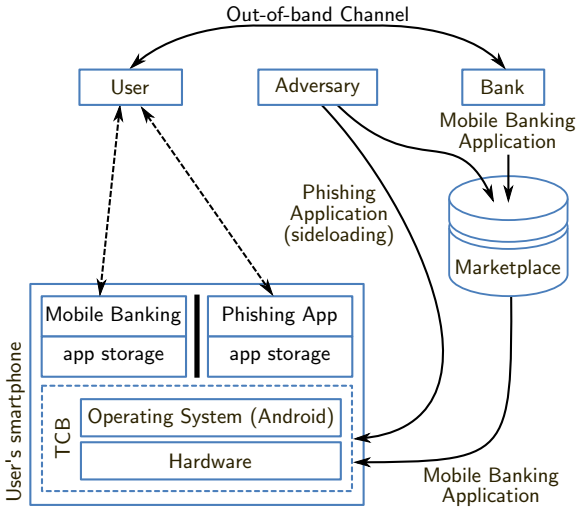


Figure 2: System model. Each application running on the smartphone is sandboxed and cannot access other applications memory or storage. Applications are installed from a marketplace or via sideloading. Banks have an out-of-band channel (e.g., regular mail) to their customers.

3.2 Adversary Model

The goal of the adversary is to succeed in a phishing attack to steal the user login credentials for the banking application. In order to mimic the GUI of the banking application, the adversary must phish the user-chosen indicator when it is set up. The adversary can induce the user to install a rogue application on her device (e.g., using social engineering). The malicious application may be installed either from the marketplace or via sideloading (as shown in Figure 2). The adversary can control all network traffic to and from the mobile device. However, the adversary cannot compromise the device TCB or the out-of-band channel between the bank and the user.

4. HARDENED INDICATOR SETUP

In this section we describe a new scheme to set up indicators more securely. We start by giving an overview of the scheme and then explain its details, security properties, and implementation. We focus on scenario where the user sets up an indicator for a mobile banking application.

4.1 Solution Overview

Setting up an indicator securely, in the presence of malicious applications, requires the user to identify the legitimate banking application during the indicator setup process. That is, the user must tell if the application asking to set up the indicator is the legitimate banking application or a malicious application that looks exactly like the banking application. We note that the only party that can tell the legitimate banking application from a malicious one is the application provider (i.e., the bank). We therefore leverage the out-of-band channel to the user in order to bootstrap the secure identification of the banking application.

Figure 3 shows in gray the components that we add to the mobile device TCB. A system component that we call the

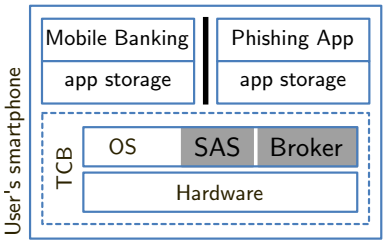


Figure 3: Our solution enhances the mobile platform with two additional components (shown in gray). The Broker can measure installed applications and has a GUI to receive user inputs. A Secure Attention Sequence (SAS) is used to start the Broker.

Broker can measure applications (e.g., compute a hash of the application binary and its configuration files) installed on the device and has a GUI to receive user inputs. The mobile OS is also enhanced with a Secure Attention Sequence (SAS), which is a common approach to start a particular software component of the TCB [11, 16, 22]. (A popular SAS is the "ctrl-alt-del" sequence in the Windows systems which generates a non-maskable interrupt that starts the user-logon process.) In our solution, the SAS operation is implemented as two repeated presses of the home button and it starts the Broker. When the Broker is running, the OS ensures that no application can activate to the foreground.

The bank sends a short PIN to the user via the out-of-band channel. The user starts the Broker via the SAS operation and enters the PIN. The Broker and the bank use the PIN to establish an authenticated channel and to verify the integrity of the banking application installed on the device. If the protocol succeeds, the Broker transfers the PIN to the banking application. When the application is started, it displays the PIN to the user and asks her to set up the application indicator. The user can identify the legitimate banking application, by comparing the PIN received from the bank with the one displayed on the screen.

4.2 Protocol Details

Figure 4 illustrates the steps to securely set up a personalized security indicator. Here we explain them in detail.

1. The bank uses the out-of-band channel (e.g., regular mail) to send the PIN together with the instructions to install the application to the user.

2. The user installs the application downloading it from the marketplace. Based on the instructions, when the installation completes, the user performs the SAS operation to start the Broker. While the Broker is running, the mobile OS prevents third-party applications from activating to the foreground.

3. The user inputs the PIN to the Broker (see Figure 5a).

4. The Broker and the bank use the PIN to run a Password Authenticated Key Exchange (PAKE) protocol [14] and establish a shared key.

5. The bank sends the measurement of the legitimate banking application to the Broker. The measurement can be the hash of the application installation package. The message is authenticated using the key established during the previous step.

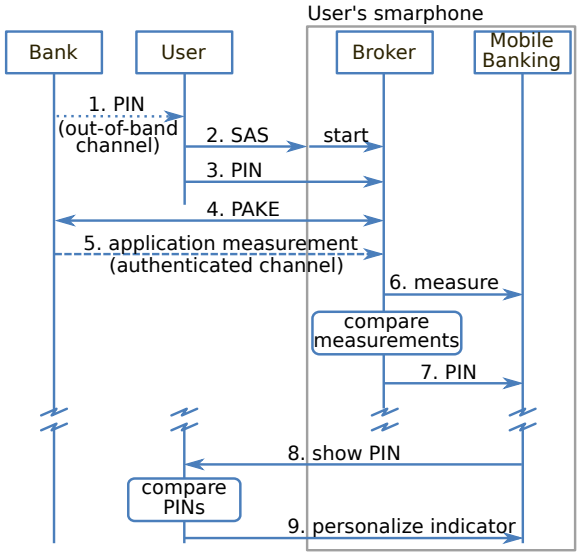


Figure 4: The application indicator setup protocol. The user performs an SAS operation to start the Broker and enters the PIN. The Broker verifies the installed banking application with the help of the bank. If the installed application is the legitimate one, the Broker starts it and the user can choose a custom indicator.

6. The Broker checks the authenticity of the received message, measures the banking application on the device, and matches its measurement against the reference value received from the bank (i.e., compares two hash values).

7. If the two measurements are identical, the Broker securely transfers the PIN to the banking application (e.g., writes the PIN to the application-specific storage). Otherwise the Broker aborts the protocol and notifies the user.

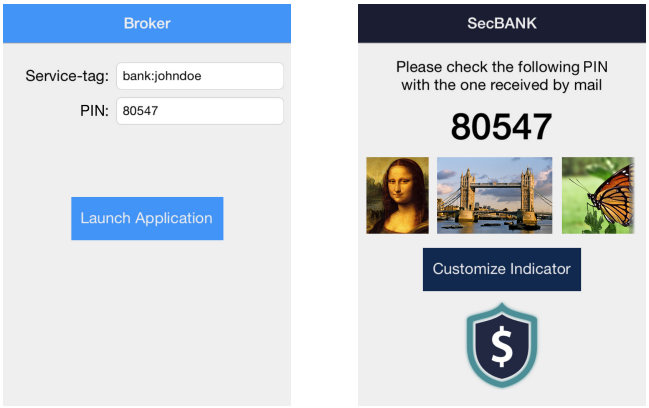
8. The Broker starts the banking application and the mobile OS restores the functionality that allows background applications to activate to the foreground. The banking application displays the PIN which serves as a confirmation to the user that the application in foreground is the legitimate banking application (see Figure 5b).

9. The user can now choose an indicator for the banking application (Figure 5b). The latter stores the indicator in its application-specific storage and displays it every time the user must input his credentials.

4.3 Security Analysis

At the beginning of the setup process, the user must input the PIN only after performing the SAS operation. At the end of the protocol, the user can identify the legitimate banking application comparing the PIN received from the bank with the one displayed by the application.

A phishing application on the device will not receive the PIN from the Broker because its measurement differs from the one of the legitimate banking application. Similarly, the adversary cannot impersonate the bank server to the Bro-



(a) (b)

Figure 5: (a) Screenshot of the Broker prototype. (b) Banking application started by the Broker. The user must verify the displayed PIN and can customize the application indicator.

ker without knowledge of either the PIN or the key derived through the PAKE protocol.

The SAS operation and the device TCB guarantee that no information is leaked when the user inputs the PIN to the Broker. The PIN is used as a one-time password to derive a key through the PAKE protocol. The derived key is only used to authenticate one message (from the bank to the Broker). The security properties of the PAKE protocol guarantee that, given a transcript of the protocol, the adversary can neither learn the PIN, nor brute-force the set of possible PINs [17].

The isolated storage functionality, provided by the mobile OS, guarantees that the PIN received by the banking application can not be read by other applications. Similarly, when the banking application stores the indicator chosen by the user, no other application can access it.

A malicious application could try to steal the user's indicator choice using an *overlay attack*. In the overlay attack, the malicious application draws partially on top of the victim application. For example, the malicious application could draw an indicator choice element on top of the banking application, leaving the correct PIN visible to the user. Implementation of such attacks requires the `SYSTEM_ALERT_WINDOW` permission. Overlay attacks can be addressed such that the banking application hides the PIN or displays a warning message it looses user focus (i.e., the user taps a UI element of another application).

4.4 Implementation

We build a prototype of our solution for the Android platform. We use a Samsung Galaxy S3 and develop against the CyanogenMod Android OS (version 4.3 JellyBean). Cryptographic operations are based on the Bouncy Castle library. Message authentication uses HMAC-SHA256 with a 128-bit key. The bank server is implemented in Python, using the CherryPy Web Framework and SQLite. Client-server communication works over a standard HTTP channel using messages in the JSON format. Our implementation adds two Java files, to implement the Broker as a privileged application, and modifies four system Java files. A total of 652 lines of code were added to the device TCB.

TCB increment	652 LoC
Communication overhead	705 bytes
Execution time (WiFi)	421ms (±21ms)
Execution time (3G)	2042ms (±234ms)
Execution time (Edge)	2957ms (±303ms)

Table 1: Performance evaluation summary for the application indicator setup prototype.

We add an extra tag (`<secureapk>`) to the Android Manifest file of the banking application. The tag contains two fields indicating a URL and an application handle. When the banking application is installed, the `PackageParser` reads the extra information in the `<secreapk>` tag and stores it for later use.

We assign the SAS operation to a double-tap on the "home" button. The SAS operation starts the Broker that asks the user to enter the PIN received from the bank (see Figure 5a). In our implementation, the bank sends to the user a 5-digit PIN (e.g., "80547") and a *service tag*. The service tag contains an application handle used by the Broker to search through the registered handles and identify the application to measure. The service tag also contains a user ID, required by the bank to retrieve the correct PIN from its database. An example of service tag is "bank:johndoe", where "bank" is the application handle and "johndoe" is the user ID. When the user has input the service tag and the PIN, the Broker uses the handle to identify the application to measure. At this time the Broker also fetches the URL stored by the `PackageParser`, to contact the bank server.

We use SPEKE [14] as an instantiation of the key establishment protocol. SPEKE uses 4 messages, including a key confirmation step, as mandated in [14]. The first SPEKE message sent by the Broker also contains the user ID that allows the bank's server to select the correct PIN from its database. When the key has been established, the server uses it to compute a MAC over the hash of the legitimate application. Both the hash and the authentication tag are sent to the Broker. The Broker verifies the authentication tag, hashes the installed application, and compares the hash computed locally with the received one. If the two hashes match, the Broker writes the PIN to the application's folder so that it can only be read by that application. Otherwise, the Broker aborts and notifies the user.

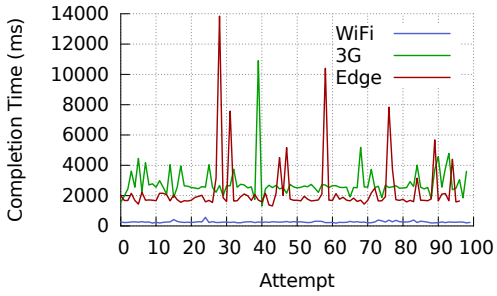
Figure 5b shows the banking application started by the Broker. The user is asked to compare the displayed PIN with the one received via mail. The user can select a custom application indicator by clicking on the "Customize Indicator" button.

5. EVALUATION

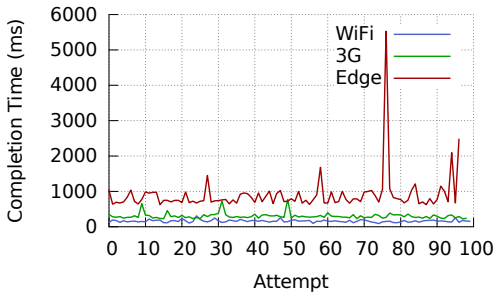
In this section we evaluate the protocol performance, and the usability and security of the setup procedure through a small-scale user study.

5.1 Protocol Performance

We use a sample banking application of 264 KB. Communication overhead totals to 705 bytes, of which 641 bytes account for the SPEKE protocol. (Communication overhead



(a)



(b)

Figure 6: (a) Completion time for 100 runs of the SPEKE protocol. (b) Time required to complete 100 runs of the setup protocol after the shared key has been established.

is independent of the application size.) We test the protocol over a WiFi connection (hosting the server in a remote location, i.e., not on a local network), as well as over 3G and EDGE cellular data connections. Figure 6a shows the time required to establish a key with the SPEKE protocol over 100 runs. Figure 6b shows the additional time required to complete the setup protocol, i.e., the time to send the authenticated hash, to measure the application on the phone, to compare the two hash values, and to transfer the PIN to the application.

Table 1 summarizes the performance evaluation in terms of added lines of code, communication overhead, and average protocol completion times. We note that hashing the banking application takes 25 ms on average, with a standard deviation of 2 ms. The time to hash an application increases linearly with the application's size and remains lower than the network delay for applications up to 100 MB.

5.2 User Study

The indicator setup protocol requires the user to follow instructions from the bank (i.e., perform the SAS operation, input the PIN, and configure the indicator). This is comparable to enrolling typical second-factor authentication tokens that customers receive from their banks.

In the following we present the result of a small-scale user study that aims at evaluating the usability and security of the proposed setup protocol. In particular, our goal was to understand how easy it is for users to setup indicators by following written instructions, like the ones a bank customer would receive from her bank. We also wanted to verify how attentive users are when comparing the PIN shown by the application with the one received from the bank, i.e., does our solution prevent subtle phishing attacks at the time of indicator setup.

We invited participants to our lab and asked them to carry out the setup procedure as if they were customers of a bank that uses indicators in its application. We provided participants with a phone and a letter from the bank with instructions on how to setup the indicator. We assigned participants to three different groups. One group carried out the set up protocol in benign settings. For the remaining two groups, we simulated variants of a background attack at the time when the banking application displays the PIN (step 8 of the setup protocol). In a background attack, the malicious application waits in the background and activates to the foreground right when the legitimate app is about to be displayed [19]. We recorded whether participants pressed the "Customize Indicator" button and chose a personalized indicator while under attack.

Recruitment and group assignment.

We advertised the study as a user study "to evaluate the usability of a setup protocol for a mobile banking application". Participants were asked to come to our lab and setup the application on a smartphone that we provided. We informed participants that we would not collect any personal information and offered a compensation of $20. We selected 30 participants and assigned them to one of three groups in a round-robin fashion. The difference among the groups was the PIN shown by the banking application right before participants were asked to choose a personalized indicator, as explained below:

- In Group A, participants were shown the same PIN that appeared on the letter from the bank.
- In Group B, participants were shown a random PIN.
- In Group C, participants where shown no PIN.

Participants of group A reflected the user experience of the setup protocol under no attack. The purpose of this group was to measure how well users are able to complete the setup procedure. Participants of groups B and C reflected the user experience of the setup protocol under a background attack where the malicious application is either guessing the PIN or displaying no PIN. The purpose of these groups was to measure how attentive users are during the indicator setup procedure.

Task details.

The task started with a pre-test questionnaire to collect demographic information. After the questionnaire, participants were given a smartphone and a letter from a fictitious bank with instructions to setup the indicator. Appendix B shows the contents of the letter given to the participants.

The letter included a 5-digit PIN and a service tag to be used during the setup protocol. The procedure was carried out on a Galaxy S3. We did not interact with the participants for the duration of the setup protocol. Participants were also asked to fill in a post-test questionnaire to evaluate the procedure they had just completed.

Results.

Table 2 shows the demographics of the participants. 40% were males and 60% females. Most of them had a university degree (84%), and were aged between 21 and 40 (90%).

Gender	
Male	12 (40%)
Female	18 (60%)
Age	
21 – 30	15 (50%)
31 – 40	12 (40%)
41 – 50	3 (10%)
Education	
High School	2 (6%)
Bachelor	5 (17%)
Master	20 (67%)
Other	3 (10%)
Use mobile banking	
Yes	9 (30%)
No	18 (60%)
Do not have	3 (10%)
total	**30**

Table 2: User study participant demographics.

	setup not completed	setup completed
Group A (no attack)	0 (0%)	10 (100%)
	attack not detected	attack detected
Group B (wrong PIN)	0 (0%)	10 (100%)
Group C (missing PIN)	4 (40%)	6 (60%)

Table 3: User study results. All participants in Group A were able to complete the procedure. In Group B, all participants detected the attack. In Group C, some participants fell for the attack.

The user study results are shown in Table 3. All participants in Group A managed to complete the task successfully. This suggests that users are able to complete the procedure by following the given instructions. All participants of Group B (wrong PIN) aborted the setup procedure because they detected a mismatch between the PIN displayed by the banking application and the one in the letter from the bank. This suggests that attacks where the adversary presents a randomly chosen PIN are not effective. In Group C (missing PIN), four participants (40% of the group) failed to detect the missing PIN and completed the setup process, thereby leaking the indicator to the phishing application. This implies that despite the given instructions, attacks with missing PIN may still work on some users.

Post-test questionnaire.

The post-test questionnaire showed that 90% of the participants rated the instruction sheet easy to understand (Q1) and 95% of them rated the task easy to complete (Q2). 87% of the participants believed that they had completed the task successfully (Q3) and 67% of them would use the mechanism in a mobile banking application (Q4). Figure 7 shows the distribution of the answers, while Appendix A provides the full text of items Q1–Q4.

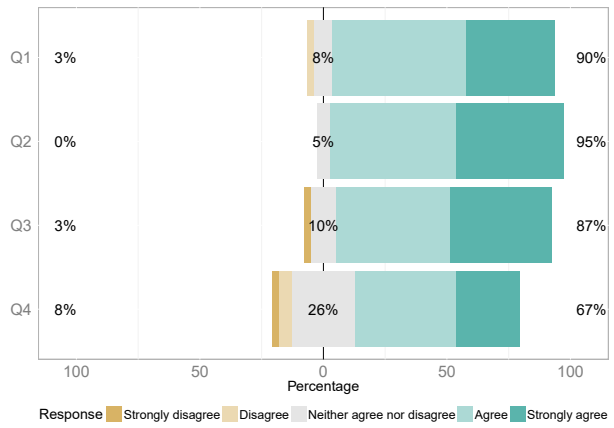


Figure 7: Answers to the post-test questionnaire for the user study on the indicator setup protocol. Percentages on the left side include participants that answered "Strongly disagree" or "Disagree". Percentages in the middle account for participants that answered "Nor agree, nor disagree". Percentages on the right side include participants that answered "Agree" or "Strongly agree".

Conclusions.

Our study shows that the setup procedure was simple enough to be completed by average users. Regarding attack detection, the "missing PIN" attack was the only one that went undetected by four participants. These participants may have judged the absence of the PIN as a temporary bug and decided to continue with the setup protocol. Similarly to previous work [19] this result shows that users are acquainted with software bugs and are likely to tolerate a temporary bug even during security-sensitive operations. These results suggest that our solution helps an alert user to setup indicators securely, but careless users might still fall for subtle attacks.

6. DISCUSSION

Benefit of our solution.

Because some of the study participants fell for attacks during the indicator setup procedure, it becomes natural to ask, what is the benefit of our solution? Simply put, our indicator setup solution has similar security guarantees as indicator usage [19]. Our solution allows an alert user to setup (and use) indicators securely, but a careless user might fall for attacks, either at the time of indicator setup or use. Without our solution, attacks at the time of indicator setup are practically impossible to detect. Therefore, our solution allows providers of security-critical mobile applications, such as banks, to improve the security of their applications (while not fully eliminate all attacks).

User study ecological validity.

When evaluating the usability of the setup protocol we could not leverage a real deployment of personalized security indicators, since no application is currently using them. Our user study was based on role-playing and we tried to keep the experiment as close to a real-world deployment as possible. For example, we edited the text provided to par-

ticipants as the one customers from a bank would receive (see Appendix B), and we did not interfere with participants during the experiments. Previous work has shown that role-playing can negatively impacts the effect of security mechanisms [27]. Therefore, the attack success rate in a real-world scenario, where the customer's money are at stake, may be even lower. Nevertheless, a user study in cooperation with a bank willing to deploy personalized security indicators in its mobile application would provide more accurate results.

Another factor that hinders the validity of our user study is the population sample and size. We only recruited 30 participants and we focused on affiliates to our institution (students, professors and staff members). They were mostly young female subjects with a university degree. Further studies with a larger number of participants and with a more diverse population are required to assess our results.

Deployment aspects.

Our scheme requires the application provider to setup an online server. Given that a large number of applications communicate with the infrastructure of their providers (to consolidate user rankings, to store user's profiles, etc.), we argue that the needed infrastructure already exists for many application scenarios. For mobile application providers that do not run online servers, the cost of adopting our solution is naturally higher.

From the results of our evaluation (see Section 4), we conclude that the setup of an application indicator is inexpensive in terms of computation and communication; hence, the solution can easily scale for a large number of users.

The setup mechanism also requires an out-of-band (OOB) communication between the application provider and the user. Examples of OOB channels include postal mail, web or email communication via a trusted device like a PC. The mobile platform itself cannot be used as the OOB channel because a malicious application could inject fabricated messages, for example, by mounting a background attack. The indicator setup mechanism can easily be adopted by application providers that already have established OOB channels to users (e.g., banks regularly send letters to their customers).

Application output protection.

Personalized security indicators allow the user to identify the application running in foreground. A careful user can, therefore, tell which application is receiving her input and detect phishing attacks.

Identifying the application in foreground also allows to protect the integrity of the application-to-user output. For example, if the application in foreground is displaying an account balance, the presence of the correct indicator can assure the user that the balance is displayed by the legitimate banking application. Nevertheless, further research is required to assess when and how to show an application indicator while retaining their effectiveness and without affecting the user experience. Personalized security indicators take screen estate and may hinder the user experience if displayed at all times. If the indicator is always visible to the user, there may be little space left for other application data. We speculate that personalized indicators are suited for login screens where in most cases only the application logo and the user input fields are displayed (most mobile app login screens follow this model [18]).

7. RELATED WORK

Application phishing. Application phishing attacks have been described in recent research [3, 8, 34] and several attacks have been spotted in the wild [6, 9, 10, 21, 29, 32]. Proposed solutions are primarily ad-hoc, i.e., they identify specific attacks and try to prevent them by restricting the device functionality that those attacks exploit [1, 3, 34].

Personalized security indicators are a well-known concept from the context of website phishing and can provide an holistic solution to application phishing attacks in mobile platforms [19]. Similar to previous work on website phishing, the authors of [34] assume that the adversary cannot mount a phishing attack when the indicator is set up. In this paper we described a protocol that allows secure setup of application indicators in the presence of an adversary. We are not aware of any previous work that provides the same security property.

Web phishing. Anti-phishing techniques for the web have been deployed in practice [5, 13]. Proposals include automated comparison of website URLs [20], visual comparison of website contents [2, 35], use of a separate and trusted authentication device [24], security indicators [4, 27], multi-stage authentication [12], and attention key sequences to trigger security checks on websites [33]. While some of these mechanisms are specific to the web environment, others could be adapted also for mobile application phishing detection. Website phishing in the context of mobile web browsers has been studied in [23, 26].

Mobile malware. Malicious mobile applications typically exploit platform vulnerabilities (e.g., for privilege escalation) or use system APIs that provide access to sensitive user data. A malicious application can, for example, leak the user location or send SMS messages to premium numbers without user consent. For reviews on mobile malware, we refer the reader to recent surveys [7, 36]. Infection rates of mobile malware are reported in [30].

Mobile malware detection is typically based on the analysis of system call patterns and detection of known platform exploits. The authors of [15] propose to detect malware by analyzing network traffic patterns. The detection mechanisms used for mobile malware are ill-suited for detection of mobile phishing applications, as many phishing attacks do not exploit any platform vulnerability, do not require access to security-sensitive system APIs, and do not generate predictable network traffic patterns.

Electronic voting. Our protocol has some similarities to electronic voting protocols, such as [25], that send a voting sheet with a unique code to each voter using regular mail. The purpose of the code is to allow the voter to authenticate the voting application or server.

8. CONCLUSION

In this paper we have addressed the problem of secure setup of personalized indicators. The anti-phishing benefits of indicators are void in case the indicator itself is phished at setup time. We introduced a novel protocol that leverages an out-of-band communication channel between the user and the service provider (e.g., the bank). Our solution is practical to deploy and it enables an alert user to set up indicators securely, even in the presence of malicious applications on the same device. A careless users might fall for attacks, either at the time of indicator setup or usage (as shown by

previous studies [19]). We see our solution as a complementary mechanism to harden security-critical mobile applications, such as banking and payment. Our solution is also the first work on secure indicator setup and we hope it encourages further research on secure user-application interaction on smartphone platforms — a problem that remains largely unsolved.

9. ACKNOWLEDGEMENTS

This work was partially supported by the Zurich Information Security Center (ZISC). It represents the views of the authors.

10. REFERENCES

[1] Bianchi, A., Corbetta, J., Invernizzi, L., Fratantonio, Y., Kruegel, C., and Vigna, G. What the app is that? deception and countermeasures in the android user interface. In *IEEE Symposium on Security and Privacy (SP)* (2015).

[2] Chen, T.-C., Dick, S., and Miller, J. Detecting visually similar web pages: Application to phishing detection. *ACM Transactions on Internet Technologies (TOIT)* (2010).

[3] Chin, E., Felt, A. P., Greenwood, K., and Wagner, D. Analyzing inter-application communication in android. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)* (2011).

[4] Dhamija, R., and Tygar, J. D. The battle against phishing: Dynamic security skins. In *Symposium on Usable Privacy and Security (SOUPS)* (2005).

[5] Dhamija, R., Tygar, J. D., and Hearst, M. Why phishing works. In *Conference on Human Factors in Computing Systems (CHI)* (2006).

[6] ESET. Android Trojan Targets Customers of 20 Major Banks. http://www.eset.com/int/about/press/articles/malware/article/android-trojan-targets-customers-of-20-major-banks/, 2016.

[7] Felt, A. P., Finifter, M., Chin, E., Hanna, S., , and Wagner, D. A survey of mobile malware in the wild. In *Workshop on security and privacy in smartphones and mobile devices (SPSM)* (2011).

[8] Felt, A. P., and Wagner, D. Phishing on mobile devices. In *Web 2.0 Security and Privacy Workshop (W2SP)* (2011).

[9] FireEye. An evolving Android trojan family targeting users of worldwide banking apps, 2015. https://www.fireeye.com/blog/threat-research/2015/12/slembunk_an_evolvin.html.

[10] FireEye. The latest android overlay malware spreading via sms phishing in europe, 2016. https://www.fireeye.com/blog/threat-research/2016/06/latest-android-overlay-malware-spreading-in-europe.html.

[11] Gligor, V. D., Burch, E. L., Chandersekaran, C. S., Chapman, R. S., Dotterer, L. J., Hecht, M. S., Jiang, W.-D., Luckenbaugh, G. L., and Vasudevan, N. On the design and the implementation of secure xenix workstations. In *IEEE Symposium on Security and Privacy (SP)* (1986).

[12] Herzberg, A., and Margulies, R. My authentication album: Adaptive images-based login mechanism. In *Information Security and Privacy Research* (2012).

[13] Hong, J. The state of phishing attacks. *Communications of the ACM* (2012).

[14] Jablon, D. The SPEKE Password-Based Key Agreement Methods, 2003. IETF Internet Draft.

[15] Lever, C., Antonakakis, M., Reaves, B., Traynor, P., and Lee, W. The core of the matter: Analyzing malicious traffic in cellular carriers. In *Network and Distributed System Security (NDSS)* (2013).

[16] Libonati, A., McCune, J. M., and Reiter, M. K. Usability testing a malware-resistant input mechanism. In *Network and Distributed System Security (NDSS)* (2011).

[17] MacKenzie, P. On the security of the speke password-authenticated key exchange protocol. In *In IACR ePrint archive* (2001).

[18] Malisa, L. Detecting mobile application spoofing attacks by leveraging user visual similarity perception, 2015. Cryptology ePrint Archive: Report 2015/709.

[19] Marforio, C., Masti, R., Soriente, C., Kostiainen, K., and Capkun, S. Evaluation of personalized security indicators as an anti-phishing mechanism for smartphone applications. In *Conference on Human Factors in Computing Systems (CHI)* (2016).

[20] Maurer, M.-E., and Hofer, L. Sophisticated Phishers Make More Spelling Mistakes: Using URL Similarity against Phishing. In *International Conference on Cyberspace Safety and Security (CSS)* (2012).

[21] McAfee. Phishing Attack Replaces Android Banking Apps With Malware, 2013. https://blogs.mcafee.com/mcafee-labs/phishing-attack-replaces-android-banking-apps-with-malware.

[22] McCune, J. M., Perrig, A., and Reiter, M. K. Safe passage for passwords and other sensitive data. In *Network and Distributed System Security (NDSS)* (2009).

[23] Niu, Y., Hsu, F., and Chen, H. iPhish: Phishing Vulnerabilities on Consumer Electronics. In *Conference on Usability, Psychology, and Security (UPSEC)* (2008).

[24] Parno, B., Kuo, C., and Perrig, A. Phoolproof phishing prevention. In *Financial Cryptography and Data Security* (2006).

[25] Ryan, P. Y., and Teague, V. Pretty good democracy. In *International Workshop on Security Protocols* (2009).

[26] Rydstedt, G., Gourdin, B., Bursztein, E., and Boneh, D. Framing attacks on smart phones and dumb routers: Tap-jacking and geo-localization attacks. In *Workshop on Offensive Technologies (WOOT)* (2010).

[27] Schechter, S. E., Dhamija, R., Ozment, A., and Fischer, I. The emperor's new security indicators. In *IEEE Symposium on Security and Privacy (SP)* (2007).

[28] Selhorst, M., Stüble, C., Feldmann, F., and Gnaida, U. Towards a trusted mobile desktop. In

International Conference on Trust and Trustworthy Computing (TRUST) (2010).

[29] Symantec. Android banking Trojan delivers customized phishing pages straight from the cloud, 2015. http://www.symantec.com/connect/blogs/android-banking-trojan-delivers-customized-phishing-pages-straight-cloud.

[30] Truong, H. T. T., Lagerspetz, E., Nurmi, P., Oliner, A. J., Tarkoma, S., Asokan, N., and Bhattacharya, S. The company you keep: Mobile malware infection rates and inexpensive risk indicators. In *International Conference on World Wide Web (WWW)* (2014).

[31] Tygar, J. D., and Whitten, A. Www electronic commerce and java trojan horses. In *Workshop on Electronic Commerce (WOEC)* (1996).

[32] WeLiveSecurity. Android banking trojan masquerades as Flash Player and bypasses 2FA, 2016. http://www.welivesecurity.com/2016/03/09/android-trojan-targets-online-banking-users/.

[33] Wu, M., Miller, R. C., and Little, G. Web wallet: Preventing phishing attacks by revealing user intentions. In *Symposium on Usable Privacy and Security (SOUPS)* (2006).

[34] Xu, Z., and Zhu, S. Abusing notification services on smartphones for phishing and spamming. In *Workshop on Offensive Technologies (WOOT)* (2012).

[35] Zhang, Y., Hong, J. I., and Cranor, L. F. Cantina: A content-based approach to detecting phishing web sites. In *International Conference on World Wide Web* (2007).

[36] Zhou, Y., and Jiang, X. Dissecting android malware: Characterization and evolution. In *IEEE Symposium on Security and Privacy (SP)* (2012).

APPENDIX

A. POST-TEST QUESTIONNAIRE

We report the items of the post-test questionnaire. All items were answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*.

Q1 *The instruction sheet was clear and easy to understand*

Q2 *The task was easy to complete*

Q3 *I believe that I have successfully completed the task*

Q4 *I would use this mechanism to improve the security of my mobile banking application*

B. INSTRUCTIONS LETTER

Figure 8 reports the exact text and format of the letter showed to the participants of our user study. We call personalized security indicators as "personal images".

Re: *Secure set up of your personal image.*

Dear Customer,

The Android Bank application uses personal images to improve the security of its mobile banking platform. Before using our application, you must set up a personal image.

Please read carefully the instructions below and follow them to securely set up your personal image. We advise you to read this letter until the end before starting. Note that during the procedure you will need the following information:

Service ID:	**ax:timross**
PIN:	**94455**

Instructions to securely set up your personal image.

1. Double-tap the button "HOME" to start the *Secure Verifier* application.
2. Type in your **Service ID** and **PIN** shown above, then tap the button "Launch Appliction".
3. The *Android Bank* application displays your PIN (**94455**). **Attention!** If the PIN is **missing** or **different** from **94455**, tap the button "HOME". The installed application is not the legitimate Android Bank application. You **must not** select any personal image and should retain from using the application. The test is over.
4. If the PIN displayed is **identical** to your PIN (94455), tap the button "Select Personal Image". The installed application is the legitimate Android Bank application. You can safely select your personal image.
5. Select one of the images displayed and tap the button "Confirm Selected Image". The test is over.

Figure 8: The instruction letter presented to the participants of the user study