(In)Security of App-based TAN Methods in Online Banking

Vincent Haupert and Tilo Müller
University of Erlangen-Nuremberg
{vincent.haupert, tilo.mueller}@cs.fau.de

***Abstract*—German banks are increasingly turning away from the established TAN methods. Their incentives for developing new technologies to replace the indexed TAN list, mTAN and chipTAN are both improved security and usability, which cannot be met with dedicated hardware. New app-based methods allow the user to initiate a transaction with his mobile device (Android or iOS), and to confirm it *on the same device*, with supposedly more security than in the established methods. We have evaluated the security of such app-based methods using the pushTAN method of the *Sparkassen-Finanzgruppe* savings banks as an example, and we can certify that the method has serious conceptual weaknesses. The deliberate decision not to use independent hardware for the transaction initiation and confirmation makes the method an easy prey for malware. To demonstrate these weaknesses, we developed an attack that captures transactions from the user and manipulates them at whim before their confirmation.**

***Index Terms*—Online Banking, TANs, Reverse Engineering**

I. Introduction

More than two thirds of German internet users carry out their banking transactions via online banking [1]. To protect the user's online access against misuse, each transaction has to be confirmed by a transaction authentication number (TAN). To this end, credit institutions frequently offer their customers several methods to choose from, which differ in how they work, as well as in their security and usability. New app-based methods will replace the established methods, and make online banking more convenient and more secure. The need for dedicated hardware is eliminated and transactions can be performed using a single mobile device. Today, Sparkassen, Volksbanken-Raiffeisenbanken, and Hypo-Vereinsbank, as well as DKB and ING-DiBa offer corresponding apps. It is likely that other German banks will follow, not only in order to offer the same convenience to their customers but also because a new attack has been launched recently against the widespread mTAN method [2].

The mobility aspect of products for app-based TAN generation is emphasized by the banks and by the security evaluation carried out by TÜV (a German security assessment company). They claim that the pushTAN method of the savings banks was specially hardened, using cryptographic methods and signatures, and that it operates isolated in a separate channel. On its website, Sparkasse lists "mobile banking with pushTAN" [3] as the first suggested method, giving readers the impression that pushTAN is highly recommended or, at least, equivalent to the established methods. However, any increased risk potential of pushTAN neither has been stated by the savings banks, nor — to our knowledge — has any independent critical analysis been carried out by third parties. On the contrary, TÜV has confirmed the use of the banking app for transaction initiation and the TAN method to confirm the transaction, on the same device, to be secure. Despite all the protection and hardening measures, the banks have de facto stepped away from two-factor authentication without any discussion in the relevant security communities. In this paper, we classify app-based methods for the first time as conceptually weaker than the established methods, and prove our evaluation by a concrete manipulation of transaction data.

II. Two-factor Authentication in Online Banking

Since its initial days, online banking has used a two-step verification procedure. Even in 1980, when Verbraucherbank introduced online banking based on BTX (interactive videotext), the customer had to initiate transactions via user name and password, and to confirm them with a TAN (at that time called money transaction numbers) [4]. The knowledge of the user name, and especially of the secret password, was the first factor, while the ownership of the physical TAN list constituted the second factor.

For a long time, the TAN list was the undisputed second factor. The security of the method was challenged only when phishing attacks started to grow. By 2005, it had been increasingly replaced by the indexed TAN list (iTAN) [5]. The iTAN method differs, as an arbitrary transaction number from the TAN list can no longer be used for the confirmation of a transaction, but only a specific TAN requested by the bank. As with the classic TAN, an iTAN is consumed upon transaction confirmation and cannot be used a second time.

While the indexed TAN list considerably reduced the number of phishing attacks as compared to conventional TANs, new malicious software was developed on the attackers' side to manipulate the transactions on the user's computer right before the TAN was entered [6]. As a result, the following methods

This is the official translation of the German publication "(Un)Sicherheit von App-basierten TAN-Verfahren im Onlinebanking". Apart from the translation, it has been slightly edited and extended with regard to new results presented at the 32C3 in December 2015. All company names, brand names, and trademarks used are the property of their respective owners, and are used only for identification and description of products and services. Implementation details have been blacked out and are independent of the conceptual weaknesses of app-based methods. The pushTAN method of Sparkasse serves explicitly as a possible example of our analysis. The security deficits of the *Sparkasse* app, as compared to other financial institutions that use similar methods, is out of scope.

were introduced in order to tie a TAN to a transaction's recipient and amount. An example of such a TAN method, which has been in use since its large-scale adaptation by Postbank in 2005, is the mTAN method. Using this method, a text message (SMS) containing the TAN and the transaction details is sent to the user's cell phone. Furthermore, according to its specification, one must not receive the TAN on the same device that is used to trigger the transaction. Ultimately, the mTAN method cannot be regarded as a strong second factor, not least because of the unencrypted delivery of the TAN, but also because attackers can hijack a mobile phone and the transfer-initiating device at the same time [7].

The chipTAN method presented at CeBIT 2006 goes a step further. In this method, a TAN is generated using the personal debit card of the user, together with a trusted TAN generator. After a transaction has been initiated in the online banking portal, a start code is displayed to the user. This start code and the transaction details have to be entered into the TAN generator with the bank card inserted. The TAN generator then generates a TAN that is valid only for this transaction. The user has to enter this TAN in the online banking portal to confirm the transaction. Although the described method is considered secure, it offers low usability because, in addition to the start code, the transaction details must be entered twice. The optical chipTAN method offers increased usability as it transmits the start code and the transaction details via sensors on the TAN generator using a "Flickercode." As a result, the receiver and the amount shown on the display of the TAN generator merely need to be confirmed before the TAN is displayed. This optical technique to transfer transaction details as well as more recent implementations using Bluetooth are considered secure. Only in the case of batch file transfers was a potential abuse scenario identified, as in the place of the amounts for individual transfers, only the total amount of all transactions will be displayed [8].

While the chipTAN method has excellent security features and is portable to some extent, its convenience has been frequently criticized. Given the increasing popularity of Android and iOS devices, Sparkassen launched an app-based method named *pushTAN* in 2014, which receives the TAN encrypted and displays it in a standalone app. If this method were used exclusively in conjunction with independent devices for initiating transactions and receiving the TAN, it would be comparable in functionality with the mTAN method. Since the pushTAN method only delivers the TAN encrypted to the smartphone, it is an even stronger second factor than mTAN, which is sent as an unencrypted text message.

In contrast to the mTAN method, mobile online banking using the pushTAN method expressly *allows and promotes using a single device* for transaction initiation and confirmation. Thus, a transaction is initiated first with the banking app and then confirmed conveniently on the same device with the TAN app. These two apps communicate so that the user does not need to retype the received TAN or copy it manually. As one of the first manufacturers of app-based methods, Sparkassen offer this service nationwide to its customers utilizing the *S-pushTAN* method. In addition, HypoVereinsbank (*HVB Mobile B@nking*), ING-DiBa (*Smart Secure*), and DKB (*DKB-pushTAN*) have introduced app-based methods in 2014, and more recently, Volksbanken Raiffeisenbanken has also developed such an app, *VR SecureGo*. A special case is the increasingly popular Fintech Number26, which operates as a mobile-first solution backed by the universal Wirecard bank [9]. Number26 was founded in 2013 as a one-device, TAN-less banking solution that does not offer any alternatives to its app-based method.

In May 2015, the German Banking Authority — the Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) — stipulated *strong customer authentication* for TAN methods in a circular to all payment service providers in the Federal Republic of Germany [10]. This circular demands a combination of at least two different factors from the categories of knowledge (e.g. username and password), possession (e.g. a smart card, a token or a smartphone), and inherence (biometrics) to carry out online transactions. Furthermore, the factors must be independent of each other, so that the "breach of an element [. . .] does not affect any other". Although this circular sounds like a description of the current situation with regard to the established methods — even the TAN list complies with these requirements — it is doubtful whether app-based methods for mobile online banking fulfill these requirements in all cases.

Due to its pioneering role, it was natural to select the *S-pushTAN* app as a representative of app-based methods. Furthermore, the *S-pushTAN* method and the *DKB-pushTAN* app have the same manufacturer, Star-Finanz GmbH, a subsidiary company belonging to Finanz IT GmbH of Sparkassen-Finanzgruppe; therefore, they have decisive similarities. Accordingly, statements made about the *S-pushTAN* app can probably be transferred to the *DKB-pushTAN* app.

III. A Security Assessment of the Sparkassen Apps

The two apps, the *Sparkasse* banking app and the *S-pushTAN* app, have partly similar, partly different security policies, which are outlined below. At the time of the investigation, the current versions in the Google Play Store were version 2.7.1 (Build 27269) for the *Sparkasse* app [11] and version 1.0.4 (Build 404) for the *S-pushTAN* app [12]. As the device for the investigations, an LG Nexus 5 with Android 5.1.1 ("Lollipop") was used.

Setup: While the *Sparkasse* app can be installed on any number of devices and can be used only with the knowledge of the user name and the password, the pushTAN method has to be requested at the savings bank, and must be unlocked in connection with a registration letter and initial login data. The *S-pushTAN* app can be used for the generation of TANs only after activation. If the *S-pushTAN* app is to be used on several devices, each device must be registered individually and unlocked with a separate registration letter.

PIN: Both apps require entering a PIN that is set during the initial use of the app. The PIN has to consist of at least eight characters, a number, a letter, and a special character. In the *S-pushTAN* app, an incorrect PIN may be entered five times; if this limit is exceeded, all application data is deleted and another setup procedure is required. The restrictive password policy for both apps is justified by the assumption that many users — for reasons of simplicity — might choose the same

PIN for both apps. However, the weaknesses of the *S-pushTAN* app identified by us are independent of this issue, and do not address the PIN policy. Still, the nature of the password policy likely causes users to choose the same PIN for both applications, thus making it an easier target for password-sniffing attacks.

Own keyboard: Since its introduction, Android has the possibility to replace the default virtual keyboard with a keyboard from a third party [13]. In principle, the use of such a keyboard induces the risk of potential keyloggers. By using a keylogger, the required PIN upon app start could be accessed even without super-user privileges. For this reason, the *S-pushTAN* app delivers its own keyboard, which is used to enter the PIN when starting the app. The *Sparkasse* app is less restrictive and does not provide a keyboard of its own. This is especially relevant in connection with the aforementioned PIN policy.

Device fingerprinting: In order to inhibit copying the app to another device, the *S-pushTAN* app implements device fingerprinting to bind a user to a specific device. The app extracts and stores unique values that are specific to the hardware and the installed Android version. When registering the app, these values are determined and stored in order to compare them on upcoming usage of the app. In the case of inequality, the application deletes all personalized data and requires a new setup. Although this process is comprehensible, it is also a source of frustration for many users of the *S-pushTAN* app, going by their comments in the Google Play Store. When delivering software updates, some device manufacturers change the values that are validated by the fingerprinting algorithm and, therefore, yield false positives.

Super-user: While the *Sparkasse* app can be used on a rooted device with only a warning message displayed to the user, the *S-pushTAN* app refuses to work when the environment is rooted. The central security anchor of the *S-pushTAN* app is that it cannot be used on a rooted device, as core security features might have been disabled.

Obfuscation: Although both apps are obfuscated with ProGuard [14], they can easily be decompiled. In particular, the implementation of an own class loader and strong features to prevent decompilation have been omitted. Still, the majority of the class and method names have lost their meaning due to the renaming done by *ProGuard*.

Certificate pinning: Both applications consistently use SSL/TLS-protected connections to prevent sniffing and modification of data by third parties. Nevertheless, connections still might be intercepted using a man-in-the-middle attack (MITM), which requires that the certificate for the MITM proxy be installed on the device. To protect against such attacks, both apps implement *certificate pinning*. As a result, both apps only trust certain certificates, thus making MITM attacks more difficult.

Repacking protection: Repacking means decoding, modifying, encoding, and eventually signing an app with a new key. To bypass security-related functions, repacking is a common practice, but it is also suitable for reverse engineering. Both apps — particularly the *S-pushTAN* app — implement protective measures to prevent repacking. For one thing, this prevents the app from being modified and distributed easily; for another thing, this feature makes the dynamic analysis more difficult.

Screen reader protection: Android offers the possibility to install so-called *screen readers*. Along with increased accessibility, screen readers provide a legitimate way to read sensitive data without root.

Promon Shield: A significant difference in the security of the two apps is the use of a dedicated security solution in the form of a native library. While the *Sparkasse* app relies solely on Java, the *S-pushTAN* app additionally uses the native *Promon Shield* [15] from the proprietary software supplier Promon. The interlocking with this library is tight in the current version of the *S-pushTAN* app. The majority of the strings used internally are outsourced to the library, which then can be queried via UIDs. Beyond that, the Promon Shield changes statically public, but actually constant, fields in Java code through reflection in order to complicate the static analysis further. The library itself is, at least partly, protected by a static, cryptographic key that is used for decryption when loading. Those features not only make the reverse engineering of the library difficult but also complicate the patching of the library.

For the *S-pushTAN* app, Promon, in principle, carries out all of the above security checks and executes call-backs in Java code. It checks periodically whether a debugger is connected, whether the app is running in an emulator, whether it has been repackaged, whether a screen reader is installed or if an attempt is made to bypass Promon by the means of *native code hooks*.

IV. Possible Attacks Against the App-based TAN Procedure

As explained in the beginning, with the pushTAN method, Sparkasse wants to implement a two-factor authentication mechanism that protects the customer even when he is careless in giving access. However, classic two-factor authentication includes a knowledge component and a possession component. This principle has been broken in the pushTAN method because "no additional device is necessary" [3]. A connection between the *Sparkasse* app and the *S-pushTAN* app allows the user to initiate and confirm transactions on one and the same device. From the *S-pushTAN* app, it is even possible to transmit the displayed TAN directly into the *Sparkasse* app without retyping it or copying it manually. The strong mutual integration of these apps is in contradiction to the advertised decoupling of the channels and begs a question: Why are two applications used? The only — but also knowledge-based — additional protection is the PIN.

Ultimately, while the implementation of the *S-pushTAN* app is technically strong and of high value, the underlying concept has to be rated as weak and vulnerable. For this reason, a variety of attacks is possible; the most promising are presented hereinafter.

A. Attacker Model and Framework

The goal of our demonstration is a *technical attack* against the app-based method used by Sparkasse. A technically successful attack may allow an attacker to initiate one or more transactions at a technical level or to manipulate them. If a transaction is executed by the attacker, it is done without the knowledge or the intention of the victim. In the case of a manipulated transaction, the attacker changes a transaction created knowingly and willingly by the victim whose transaction data is free from outside interference (i.e. no *social engineering*).

The attacker model is based on usage of the *S-pushTAN* app in conjunction with the *Sparkasse* app in accordance with the terms of use prescribed by Sparkasse. Specifically, this means that the operating system has no modifications — in particular, no root. However, for the used combination of device and Android version, a root exploit is known, which allows to root the operating system without losing data. Consequently, an attacker has the ability to run such an exploit and, afterward, might place and execute arbitrary malicious code in a super-user context.

B. Reverse Engineering of the Transaction Protocols

The most attractive and harmful attack from the attackers' point of view would be reverse engineering the transaction protocols used by the *S-pushTAN* app and developing an own client. Once this step has been completed successfully, only victim-specific data, such as session keys that are generated during the initialization of the app, remain to be obtained. This data could be read through copying them by physical access, by an MITM attack or by instrumentation. Since the reverse engineering of the protocol can be made victim-unspecific and separately, the attack on the data is limited to the victim-specific data of the *S-pushTAN* app. Afterward, an attacker can create an unlimited number of valid TANs.

Although this attack has a high potential for damage, it requires some development effort by the attacker. The certificate pinning and repacking protection at least prevent the protocol from being easily read via an MITM attack. Finally, reverse engineering is made more difficult by the implemented obfuscation measures. Nevertheless, in the present investigation, parts of the protocol could be read by app interception, whereby we wish to emphasize that the development of a malicious client is within the realm of possibility.

C. Copying of the TAN App

Another obvious idea is to clone the TAN app, including all its data. In general, the application data is only accessible by the app itself, which is why the attack requires root privileges. An advantage of the attack would be that the integrity of the app and the data remain unaltered. However, methods for reverse engineering would be required to understand the device fingerprinting implemented in the *S-pushTAN* app. The results of this analysis are used by the attacker's device in order to replicate precisely the fingerprint with the manipulated app. In addition to the pure copying of the app and data, a script to query specific device characteristics would be necessary. In addition, the PIN would have to be obtained. The individual steps of the attack could be designed as follows:

1) The attacker places a script on the device of the victim, which is triggered when the user starts the *S-pushTAN* app. As a result, the script returns the recorded PIN, the app including data, as well as the device properties necessary for the device fingerprinting. Depending on the placement strategy, the delivery of this data to the attacker can be done over the network or via USB.
2) The victim's device is no longer needed and any traces resulting from the attack should be eliminated as much as possible, at least to the extent that the intervention is not noticed by the victim.
3) Now, the app and the sniffed data have to be placed on the attacker's device. In addition, the fingerprinting algorithm of the app needs to read the same values as it would have on the victim's device.

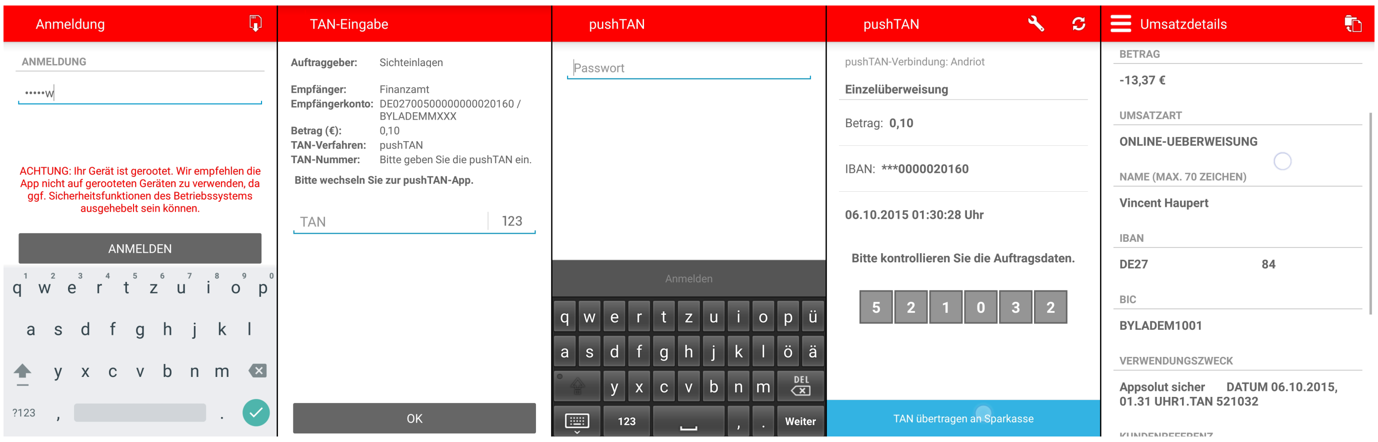
The main implementation effort for this attack lies in the understanding of the fingerprinting algorithm and the values that are used for processing. To gain access to the PIN, only a logging attack seems feasible. A brute-force attack against the PIN will probably not be successful because of the password policy.

D. Instrumentation of Both Apps

While the two attacks described earlier would focus on the fact that the platform — in this case, Android — despite all protection measures, cannot be regarded as safe and trustworthy, we will now describe attacks that compromise the assumed secure mobile banking app directly. To this end, both the banking app and the TAN app are instrumented in order to either manipulate transactions initialized by the user or allow the attacker to initiate or confirm any transaction. Both attacks have in common that they rely on the activities of the user, but do not require social engineering. This type of attack has been concretely realized by us and supports our argumentation.

A first option is the invisible manipulation of transaction data. Once a user enters a transaction in the banking app, it is modified by the malware just before it is sent to the banking server. If, in the next step, a dedicated hardware like a TAN generator is used, manipulation would become evident by displaying the recipient and amount. But as the TAN app is used on the same smartphone, it is not necessary to compromise the second factor separately. The malicious software stores the transaction data previously entered by the user in the banking app and manipulates the display in the TAN app presenting only expected values. This allows for carrying out transactions with arbitrary recipient details without the user's knowledge.

With appropriate additional effort, the attack could be expanded in such a way that the malicious software processes the TAN app in an automated way. As a first step, the malicious software captures the confidential data granting access to the online banking platform when the user performs a transaction. The user then unlocks the TAN app by entering his PIN. Now, the malicious software can prevent the TAN app from locking itself again automatically when the user changes the app. In this way, the attacker could not only initiate any transactions, but also confirm them automatically by calling the corresponding functions of the TAN app.



ANMELDUNG

••••••w

ACHTUNG: Ihr Gerät ist gerootet. Wir empfehlen die App nicht auf gerooteten Geräten zu verwenden, da ggf. Sicherheitsfunktionen des Betriebssystems ausgehebelt sein können.

Auftraggeber:	Sichteinlagen
Empfänger:	Finanzamt
Empfängerkonto:	DE02700500000000020160 / BYLADEMMXXX
Betrag (€):	0,10
TAN-Verfahren:	pushTAN
TAN-Nummer:	Bitte geben Sie die pushTAN ein.

Bitte wechseln Sie zur pushTAN-App.

TAN 123

Passwort

pushTAN-Verbindung: Andriot

Einzelüberweisung

Betrag: 0,10

IBAN: ***0000020160

06.10.2015 01:30:28 Uhr

Bitte kontrollieren Sie die Auftragsdaten.

5 2 1 0 3 2

BETRAG

-13,37 €

UMSATZART

ONLINE-UEBERWEISUNG

NAME (MAX. 70 ZEICHEN)

Vincent Haupert

IBAN

DE27 84

BIC

BYLADEM1001

VERWENDUNGSZWECK

Appsolut sicher DATUM 06.10.2015, 01.31 UHR1.TAN 521032

Figure 1: Overview of the transaction manipulation (left to right): Once a user has logged into the online banking app, he or she initiates a credit transfer worth € 0.10 to the tax administration, which must be confirmed by a TAN. After logging into the *S-pushTAN* app, the user can transmit the TAN 521033 with a single click directly into the *Sparkasse* app. Although the transaction data appear to be correct in the *S-pushTAN* app, the account statement details show that instead of transmitting 0.10 € to the tax administration, 13.37 € was sent to Vincent Haupert with the TAN 521033.

V. The Attack: Manipulation of Transaction Data

The attack realized by us instrumented and manipulated the *Sparkasse* app and the *S-pushTAN* app. Ultimately, the attack is placed on a device that the banking customer uses for online banking in conjunction with the pushTAN method. At the moment that the user initiates a credit transfer, the entered values are replaced by the details of the attacker and the transaction becomes effective as soon as the victim confirms the transaction using the *S-pushTAN* app. The expected data will still be presented to the user in all transaction steps of both apps. The sequence of this specific attack is shown in Figure 1.

The attack is implemented as a module for the instrumentation framework *Xposed* [16], which is available for all versions starting with Android 4.0.3 (“Ice Cream Sandwich”). As shown in Figure 2, *Xposed* integrates itself deeply inside the Android system; thus, it is able to hook and modify any Java code. In order to install *Xposed*, root or access to a custom recovery (i.e. an unlocked bootloader) is required. *Xposed* itself, however, is not required to realize the attack and there are various other approaches to do so. Nevertheless, *Xposed* provides a good basis for a *proof of concept*, which allows us to reduce the development effort. Still, we will not tire of emphasizing that a real attack will likely neither install *SuperSU* nor *Xposed*, rendering all safeguards based on their detection useless against real-world malware.

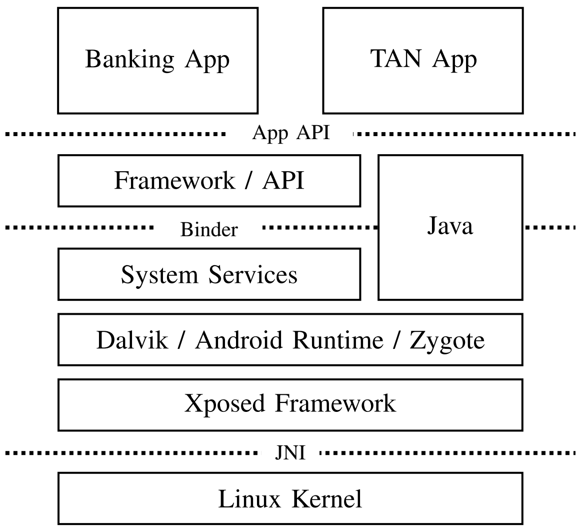


Figure 2: The *Xposed* framework in Android’s architecture [17]. After its installation, the framework can instrument any Java classes and methods in arbitrary apps.

A. Attack Points of the Sparkasse App

When online banking transactions are authorized with the pushTAN method, the *Sparkasse* app is used for initiating credit transfers and, therefore, is the starting point for our attack. First of all, it is worth mentioning that the app can be used by default with root; only a notice warns that this may pose risks. After the user has navigated to the appropriate mask inside the app to place a credit transfer, he has to enter the following fields: beneficiary, international bank account number (IBAN), bank identifier code (BIC), amount, and intended use.

The goal of this attack is to manipulate the transaction data before it is sent to the banking server *without* making the changes visible to the user at any time. Through reverse engineering, we have identified the class [redacted] and the method [redacted] responsible for the button event *submit*. Prior to the execution of this method, the transaction data of the attacker is retrieved from the attacker’s server through instrumentation and manipulated by means of reflection. After execution of the method, the transaction details revert to the original ones and are saved for later use in the *S-pushTAN* app.

In the next step, another window pops up, which again displays the transaction data and requests the TAN from the *S-pushTAN* app. In the context of the attack, it was not necessary to adjust the displayed data again because the banking server sends no confirmation or copy of the transaction data or the data is just not processed by the *Sparkasse* app. The next step of the attack targets the *S-pushTAN* app.

B. *Attack Points of the S-pushTAN-App*

The main task of the *S-pushTAN* app is to display the transaction details and the TAN to complete the transaction. In contrast to the *Sparkasse* app, the *S-pushTAN* app cannot be run on a rooted device and terminates itself after showing a brief notice. This protection must first be deactivated. The manufacturer Star-Finanz has not implemented its own checking for root access, but relies on the external and native module *Promon Shield*. For *Promon*, however, appropriate call-backs exist in Java code, which are used for displaying the warning and terminating the app afterward. The call-back to check for root in the method of the respective class was instrumented and deactivated. This procedure is free of consequences, which is surprising, as the current version of the *S-pushTAN* app has a strong interconnection with *Promon Shield*. For example, the majority of strings are stored in *Promon* and can only be accessed via an ID. The *Promon Shield* executes its security methods repeatedly, but does not refuse the retrieval of the outsourced strings when root was detected. Moreover, after disabling the root checking the app can be used as normal for the retrieval of TANs.

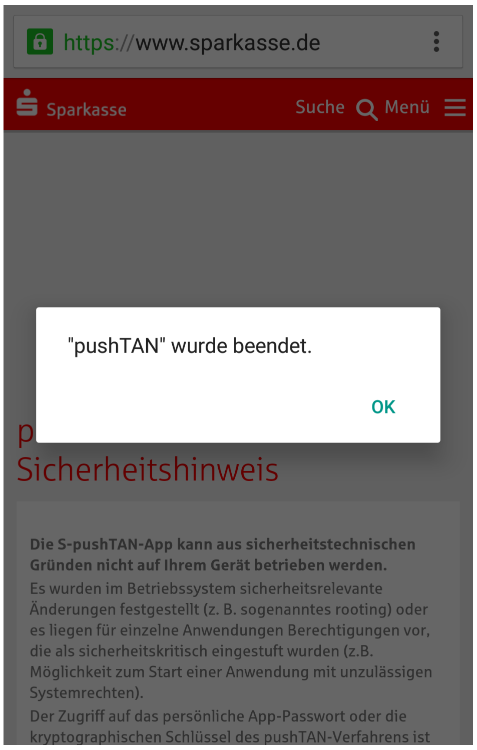
Still, the banking server sends the transaction data manipulated by the attacker and displays it in the *S-pushTAN* app. The data displayed is limited to the amount and the (masked) IBAN. Both values must be changed to the original values entered by the user. Therefore, the malicious code recovers the transaction data from the victim previously stored in the *Sparkasse* app. For this manipulation, the respective class [redacted] and its method [redacted] were instrumented. This method generates several key-value pairs from the data obtained from the banking server to be displayed inside the app. The two pairs *amount* and *IBAN* are changed according to the retrieved original transaction data input by the user. The user will see the expected values, which is why he agrees to transmit the TAN into the *Sparkasse* app and, thus, confirms the transaction, making it effective.

C. *Relevance for Future Updates*

The demonstrated attack works only for the specific versions of the *Sparkasse* app (2.7.1) and the *S-pushTAN* app (1.0.4) for which the attack has been developed. The reason for this is that the renaming generated by the obfuscator *ProGuard* assigns different names for classes and methods for each compilation. Nevertheless, the adaptation of the exploit would not be particularly complex. The attack could even be expanded in such a way that class and method names are loaded depending on the version. More profound changes to the instrumented classes and/or methods that are to be expected with new versions would also require an adaptation of the source code.

In the case of the *S-pushTAN* app, its obfuscation has become stronger from version to version. This is primarily the result of improvements to the *Promon Shield*, which has taken over more and more parts of the application, and disguises its operation more effectively in new versions. With the initial publication of this paper (in German), on October 16, 2015, the *S-pushTAN* app has been released in version 1.0.5, which contains new ways for combating rooted devices and disabling the demonstrated solution. In their statement, the German association of savings banks (*Deutscher Sparkassen- und Giroverband* [DSGV]) rejected the described attack as irrelevant because it was carried out targeting an outdated version of the *S-pushTAN* app. Nevertheless, we wish to point out and warn that the latest version of the *S-pushTAN* app — and all future versions — can be broken with a corresponding effort. Stronger measures for code obfuscation can always be countered by stronger reverse engineering, so that only the implementation effort of the attack will increase, while the conceptual weaknesses of the app-based method can never be eliminated.

In order to demonstrate that this is not just a mere statement, we realized the same attack shown before for the more recent version 1.0.7 of the *S-pushTAN* app. Starting from version 1.0.5, the *S-pushTAN* app no longer invokes Java call-backs to deal with a detected issue. Instead, the *S-pushTAN* app ostensibly crashes with the usual Android system dialog, as shown in Figure 3. Right after, the browser is invoked and opens the site https://sparkasse.de, which explains that the *S-pushTAN* app was terminated because of detected security issues. New countermeasures are solely introduced by an updated version of the *Promon Shield* and, apart from the elimination of Java callbacks, they mainly focus on the detection of common hooking frameworks like *Xposed* and *Cydia Substrate* [18].



"pushTAN" wurde beendet.

p
Sicherheitshinweis

Die S-pushTAN-App kann aus sicherheitstechnischen Gründen nicht auf Ihrem Gerät betrieben werden.

Es wurden im Betriebssystem sicherheitsrelevante Änderungen festgestellt (z. B. sogenanntes rooting) oder es liegen für einzelne Anwendungen Berechtigungen vor, die als sicherheitskritisch eingestuft wurden (z.B. Möglichkeit zum Start einer Anwendung mit unzulässigen Systemrechten).

Der Zugriff auf das persönliche App-Passwort oder die kryptographischen Schlüssel des pushTAN-Verfahrens ist

Figure 3: Starting from version 1.0.5, the *S-pushTAN* app crashes if it detects a security issue and opens a web page explaining the behavior.

Both the *Xposed* detection and the root detection could be defeated, thus enabling the attack carried out against version 1.0.4 once again. More detailed information as well as a video demonstration can be found in the talk (in German) given at the *32th Chaos Communication Congress* [19].

VI. Mobile Malware in the Play Store

The presented possibilities of attack — as for any malicious software — raise the question of how attacks are placed on victim's devices and how the conditions assumed by us — particularly gaining root privileges — can be achieved. A recent study by Cambridge University shows that 87.7% of all Android devices are vulnerable to critical security breaches [20]. With seven of the eleven vulnerabilities examined, root privileges can be obtained without physical access to the device. The study also shows that vulnerabilities resulting from the phone manufacturers' software are closed only slowly, if at all, which means that many devices remain vulnerable for a long time, even after the discovery of a vulnerability.

Regarding the placement of the attack, different strategies are conceivable. First, the app could be installed by a user from a third source. To do this, the user may have to be motivated by social engineering. Substantially more dangerous is the injection of malicious software in the official Play Store, which cannot defend itself appropriately against sophisticated malware. Maier, Müller, and Protsenko have already shown in 2014 that so-called *split-personality malware* — i.e. the splitting of malicious software into several parts, where each of them taken by itself is harmless — is not detected by the automated review process used by the Play Store [21].

It is not fiction that malicious apps that initially root the device before they place arbitrary malicious code for execution are available in the Google Play Store. This was demonstrated in September 2015 by the app *Brain Test* [22]. This app, which today is classified as particularly dangerous, has been available in the Play Store for a long time, and has run different root exploits on users' devices in order to subsequently install malicious software and execute it. It escaped from the Google review analysis process by recognizing the automated analysis environment, and showing non-suspicious profile in this environment.

VII. Conclusion

The realized attack directed the *Sparkasse* app and the *S-pushTAN* app to manipulate the transaction data, which clearly demonstrates the conceptual weakness of the app-based method, when transaction initiation and confirmation on separate hardware is not applied. Here, the attack does not permit the detection of the manipulation by the user at any time, as the data displayed corresponds to the values entered by the user during every phase of the transaction process. Basically, the attack could — with appropriate additional effort — be extended in such a way that it works independently on a device (without root, but with an available root exploit). As the demonstrated attack is directed not against a technical, but against a conceptual weakness of the procedure, an app-based method, and in particular the *S-pushTAN* app, can never be fully secured by pure technical improvements. For secure online banking, we recommend that you give up the comfort of mobile online banking on a single device. Instead, choose methods that rely on a second authentication factor independent from the first factor, such as the established chipTAN procedure.

References

[1] Bitkom e.V. (2015, 08) Online-Banking ist bequem und sicher. [Online]. Available: https://www.bitkom.org/Presse/Presseinformation/Online-Banking-ist-bequem-und-sicher.html

[2] H. Freiberger. (2015, Oct.) Betrugsserie beim Online-Banking. [Online]. Available: http://www.sueddeutsche.de/geld/online-banking-betrugsserie-beim-online-banking-1.2700184

[3] W. Block. pushTAN Mobile-Banking - Sparkasse.de. [Online]. Available: https://www.sparkasse.de/service/sicherheit-im-internet/tan-verfahren.html

[4] D. Borchers. (2015, 11) Vor 30 Jahren: Online-Banking startet in Deutschland. [Online]. Available: http://heise.de/-1135331

[5] D. Grell. (2005, 08) Postbank mit neuem TAN-System gegen Phishing. [Online]. Available: http://heise.de/-121126

[6] D. Bachfeld, "Mit guten Karten," *c't*, vol. 19, pp. 92–95, 2009.

[7] J. Schmidt. (2011, 04) Angriffe auf deutsche mTAN-Banking-User. [Online]. Available: http://heise.de/-1221951

[8] RedTeam Pentesting GmbH. (2009, 11) Man-in-the-Middle-Angriffe auf das chipTAN comfort-Verfahren im Online-Banking. [Online]. Available: https://www.redteam-pentesting.de/publications/2009-11-23-MitM-chipTAN-comfort_RedTeam-Pentesting.pdf

[9] NUMBER26 GmbH. (2016, 01) Customer 100000 – NUMBER26. [Online]. Available: https://number26.eu/our-customer-100000/

[10] Bundesanstalt für Finanzdienstleistungsaufsicht. (2015, 05) Mindestanforderungen an die Sicherheit von Internetzahlungen. [Online]. Available: https://www.bafin.de/SharedDocs/Veroeffentlichungen/DE/Rundschreiben/2015/rs_1504_ba_MA_Internetzahlungen.html

[11] Star Finanz GmbH. Sparkasse – Android-Apps auf Google Play. [Online]. Available: https://play.google.com/store/apps/details?id=com.starfinanz.smob.android.sfinanzstatus&hl=de

[12] ——. S-pushTAN – Android-Apps auf Google Play. [Online]. Available: https://play.google.com/store/apps/details?id=com.starfinanz.mobile.android.pushtan&hl=de

[13] Google Inc. (2009, 04) Android 1.5 Platform Highlights. [Online]. Available: https://www.redteam-pentesting.de/publications/2009-11-23-MitM-chipTAN-comfort_RedTeam-Pentesting.pdf

[14] E. Lafortune. Proguard. [Online]. Available: http://proguard.sf.net

[15] Promon AS. Promon - True App Security. [Online]. Available: http://www.promon.no

[16] rovo89 and Tungstwenty. (2015) Xposed Installer | Xposed Module Repository. [Online]. Available: http://repo.xposed.info/module/de.robv.android.xposed.installer

[17] K. Yaghmour, *Embedded Android: Porting, Extending, and Customizing*, 1st ed. O'Reilly Media, Inc., 2013.

[18] SaurikIT, LLC. (2016) Cydia Substrate. [Online]. Available: http://www.cydiasubstrate.com/

[19] V. Haupert. (2015, 12) (Un)Sicherheit von App-basierten TAN-Verfahren im Onlinebanking. 32. Chaos Communication Congress. [Online]. Available: https://media.ccc.de/v/32c3-7360-un_sicherheit_von_app-basierten_tan-verfahren_im_onlinebanking

[20] D. Thomas, A. Beresford, and A. Rice, "Security metrics for the android ecosystem," in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, ser. SPSM '15. New York, NY, USA: ACM, 2015, pp. 87–98.

[21] D. Maier, T. Müller, and M. Protsenko, "Divide-and-conquer: Why android malware cannot be stopped," in *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*, Sept 2014, pp. 30–39.

[22] A. Polkovnichenko and A. Boxiner. (2015, 09) BrainTest – A New Level of Sophistication in Mobile Malware. [Online]. Available: http://blog.checkpoint.com/2015/09/21/braintest-a-new-level-of-sophistication-in-mobile-malware/