(Un)Sicherheit von App-basierten TAN-Verfahren im Onlinebanking

Vincent Haupert und Tilo Müller

Friedrich-Alexander-Universität Erlangen-Nürnberg
{vincent.haupert, tilo.mueller}@cs.fau.de

***Zusammenfassung*—Die deutschen Kreditinstitute wenden sich zunehmend von den alten TAN-Verfahren ab. Als Motiv zur Erschließung neuer Techniken abseits der indizierten TAN-Liste, mTAN und chipTAN wird neben der Sicherheit auch der fehlende Komfort durch die Notwendigkeit dedizierter Hardware angeführt. Neue App-basierte TAN-Verfahren erlauben es dem Nutzer, eine Transaktion mit seinem mobilen Endgerät (Android oder iOS) auszulösen und *auf dem selben Gerät* zu bestätigen – und das bei vermeintlich höherer Sicherheit als bei den etablierten Verfahren. Wir haben die Sicherheit solcher App-basierten TAN-Verfahren am Beispiel des pushTAN-Verfahrens der Sparkassen ausgewertet und attestieren dem Verfahren gravierende konzeptionelle Schwächen. Der bewusste Verzicht auf eigenständige Hardware zur Transaktionsauslösung und -bestätigung macht das Verfahren für Schadsoftware zu einer leichten Beute. Zur Demonstration dieser Schwächen haben wir einen Angriff entwickelt, der vom Nutzer Transaktionen abfängt und vor ihrer Bestätigung nach Belieben manipulieren kann.**

***Index Terms*—Onlinebanking, TANs, Reverse Engineering**

I. Einleitung

Onlinebanking wird heute von über zwei Drittel der deutschen Internetnutzer zur Tätigung ihrer Bankgeschäfte genutzt [1]. Um den Onlinezugang eines Nutzers vor Missbrauch zu schützen, muss jede Transaktion durch eine Transaktionsnummer (TAN) bestätigt werden. Hierfür bieten die Kreditinstitute ihren Kunden meistens verschiedene Verfahren zur Auswahl an, die sich in ihrer Funktionsweise, ihrer Sicherheit und ihrem Komfort unterscheiden. Neue App-basierte TAN-Verfahren sollen die etablierten Verfahren ablösen und Onlinebanking bequemer und sicherer machen. Die Notwendigkeit von dedizierter Hardware entfällt und Transaktionen können mit einem einzelnen mobilen Endgerät durchgeführt werden. Sowohl die Sparkassen, Volksbanken-Raiffeisenbanken, die Hypovereinsbank, als auch die DKB und die ING-DiBa haben entsprechende Apps im Angebot. Es ist anzunehmen, dass die übrigen deutschen Kreditinstitute folgen werden, nicht nur um ihren Kunden den gleichen Komfort bieten zu können, sondern auch weil gegen das verbreitete mTAN-Verfahren vor Kurzem ein neuer Angriff bekannt geworden ist [2].

 Implementierungsdetails wurden geschwärzt und sind unabhängig von den konzeptionellen Schwächen der App-basierten TAN-Verfahren. Das pushTAN-Verfahren der Sparkassen dient ausdrücklich als ein mögliches Beispiel unserer Analyse. Ein Sicherheitsdefizit der Sparkassen gegenüber anderen Kreditinstituten, die ähnliche Verfahren einsetzen, lässt sich daraus nicht ableiten.

Der Mobilitätsaspekt von Produkten zur App-basierten TAN-Erzeugung wird von den Banken ebenso betont, wie eine durch den TÜV abgelegte Sicherheitszertifizierung. Das pushTAN-Verfahren der Sparkassen sei speziell gehärtet, arbeite mit kryptographischen Verfahren und Signaturen und werde zudem isoliert in einem separaten Kanal betrieben. Auf ihrer Internetpräsenz listet die Sparkasse „Mobile-Banking mit der pushTAN“ [3] sogar als erstes Verfahren und erweckt bei ihren Lesern damit den Eindruck, als sei pushTAN besonders zu empfehlen, oder zumindest gleichwertig gegenüber den etablierten Verfahren. Ein erhöhtes Gefahrenpotential durch pushTAN wird weder von den Sparkassen aufgeführt, noch ist uns eine unabhängige kritische Analyse durch Dritte bekannt. Im Gegenteil, nachdem der TÜV es als sicher eingestuft hat, die Banking-App zur Transaktionsauslösung und die TAN-App zur Transaktionsbestätigung auf ein und demselben Gerät zu verwenden, haben sich die Banken trotz aller Schutz- und Härtungsmaßnahmen de facto von der Zwei-Faktor-Authentifikation verabschiedet — ohne dass dies von Sicherheitskreisen angeprangert oder diskutiert worden wäre. In dieser Arbeit stufen wir App-basierte TAN-Verfahren erstmals als konzeptionell schwächer als die etablierten Verfahren ein und belegen unsere Einstufung durch konkrete Manipulationen von Auftragsdaten.

II. Die Zwei-Faktor-Authentifikation im Onlinebanking

Seit jeher ist Onlinebanking ein Zwei-Faktor-Verfahren. Schon als es 1980 die Verbraucherbank – damals noch BTX-basiert – einführte, musste der Kunde Transaktionen über Benutzername/Passwort auslösen und mit einer TAN (damals Geldtransaktionsnummern genannt) bestätigen [4]. Das Wissen über den Benutzernamen, und vor allem über das geheime Passwort, war dabei der erste Faktor, während der Besitz der physischen TAN-Liste den zweiten Faktor darstellte.

Die TAN-Liste blieb lange der unbestrittene zweite Faktor. Erst als sich Phishing-Angriffe mehrten, wurde das Verfahren bezüglich seiner Sicherheit in Frage gestellt und 2005 zunehmend durch die indizierte TAN-Liste (iTAN) abgelöst [5]. Bei dem iTAN-Verfahren kann nicht mehr länger eine beliebige Transaktionsnummer der TAN-Liste zur Transaktionsbestätigung genutzt werden, sondern nur noch eine bestimmte, die vom Kreditinstitut vorgegeben wird. Wie schon bei der klassischen TAN-Liste ist eine iTAN nach ihrer Verwendung verbraucht und kann danach nicht mehr genutzt werden.

Die indizierte TAN-Liste konnte gegenüber herkömmlichen TANs die Anzahl der Pishing-Angriffe merklich reduzieren, sorgte seitens der Angreifer jedoch dafür, dass neue Schadsoftware entwickelt wurde, die Transaktionen auf den Rechnern der Nutzer manipuliert — noch vor Eingabe der TAN [6]. Im Folgenden wurden deshalb Verfahren eingeführt, die eine TAN fest an Empfänger und Betrag binden. Ein solches TAN-Verfahren, das seit seiner großflächigen Adaptierung durch die Postbank 2005 auch heute noch Anwendung findet, ist das mTAN-Verfahren. Dabei erhält der Nutzer via SMS eine TAN samt Auftragsdetails auf sein Mobiltelefon. Ferner kann eine Transaktion gemäß Spezifikation nicht auf dem gleichen mobilen Endgerät ausgelöst werden, das die TAN per SMS empfängt. Letztendlich kann das mTAN-Verfahren aber, nicht nur wegen der unverschlüsselten Zustellung der TAN, sondern auch weil Angreifer ein Mobiltelefon und Überweisungsgerät gleichzeitig kapern könnten, nicht als ausreichend starker zweiter Faktor bezeichnet werden [7].

Noch einen Schritt weiter geht daher das auf der CeBIT 2006 vorgestellte chipTAN-Verfahren. Bei dem Verfahren wird mittels eines vertrauenswürdigen TAN-Generators und der persönlichen Bankkarte des Nutzers eine TAN generiert. Nachdem eine Transaktion im Onlinebanking-Portal angestoßen wurde, wird dem Benutzer ein Startcode angezeigt. Im Folgenden müssen in den TAN-Generator bei eingesteckter Bankkarte der dargestellte Startcode sowie die Überweisungsdetails eingegeben werden. Daraufhin generiert der TAN-Generator eine nur für diesen Auftrag gültige TAN, die der Nutzer zur Transaktionsbestätigung im Onlinebanking-Portal eingeben muss. Obwohl das vorgestellte Verfahren als sicher gilt, bietet es seinen Verwendern wenig Komfort, da neben dem Startcode die Überweisungsdetails doppelt eingegeben werden müssen. Erhöhten Komfort bietet das optische chipTAN-Verfahren, das über Sensoren am TAN-Generator den Startcode und die Überweisungsdetails mittels eines am Bildschirm dargestellten Flickercodes überträgt. Anschließend müssen der am Display des TAN-Generators angezeigte Empfänger und Betrag bestätigt werden, bevor die TAN angezeigt wird. Auch in dieser Ausführung gilt das Verfahren als sicher. Lediglich bei Sammelüberweisungen wurde bisher Missbrauchspotenzial nachgewiesen, da statt den Beträgen der Einzelüberweisungen nur der Gesamtbetrag aller Überweisungen angezeigt wird [8].

Während das chipTAN-Verfahren hervorragende Sicherheitseigenschaften besitzt und relativ portabel ist, wird sein Komfort oft kritisiert. Mit der zunehmenden Verbreitung von Android- und iOS-Geräten bieten die Sparkassen seit 2014 deswegen ein App-basiertes TAN-Verfahren mit dem Namen pushTAN an, das die TAN in einer eigenständigen App verschlüsselt empfängt und anzeigt. Würde das Verfahren ausschließlich in Verbindung mit eigenständigen Geräten zur Transaktionsauslösung und zum Empfang der TAN genutzt, wäre es in seiner Funktionsweise mit dem mTAN-Verfahren vergleichbar. Da bei pushTAN die TAN verschlüsselt auf das Smartphone übertragen wird, handelt es sich in diesem Fall sogar um einen stärkeren zweiten Faktor als die über SMS unverschlüsselt zugestellte TAN.

Im Gegensatz zum mTAN-Verfahren wird bei pushTAN jedoch *mobiles Onlinebanking auf einem einzigen Gerät* ausdrücklich ermöglicht und gefördert. So kann mit der Banking-App eine Transaktion zuerst ausgelöst und dann mit der TAN-App bequem auf dem selben Gerät empfangen und bestätigt werden. Dazu kommunizieren die beiden Apps in einer Art und Weise, die es dem Nutzer erspart die eingegangene TAN abzutippen oder manuell zu kopieren. Als einer der ersten Hersteller App-basierter TAN-Verfahren bieten die Sparkassen ihren Kunden diesen Service bundesweit mit der *S-pushTAN* an. Auch die Hypovereinsbank (*HVB Mobile B@nking*), ING-DiBa (*SmartSecure*) und DKB (*DKB-pushTAN-App*) führten 2014 jeweils ein App-basiertes TAN-Verfahren ein und seit kurzem sind die Volksbanken-Raiffeisenbanken mit *VR-SecureGo* ebenfalls mit einer solchen App vertreten.

In einem Rundschreiben an alle Zahlungsdienstleister der Bundesrepublik Deutschland hat die Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) im Mai 2015 eine *starke Kundenauthentifizierung* für TAN-Verfahren vorgeschrieben [9]. Diese sieht eine Kombination von mindestens zwei verschiedenen Faktoren aus den Kategorien Wissen (z.B. Benutzername und Passwort), Besitz (z.B. eine Smartcard, ein Token oder ein Smartphone) und Inhärenz (biometrische Merkmale) zur Durchführung von Online-Transaktionen vor. Ferner müssen die Faktoren voneinander unabhängig sein, so dass die „Verletzung eines Elements [...] keinen Einfluss auf das andere bzw. die anderen“ hat. Obwohl diese Richtlinie in Hinblick auf die etablierten Verfahren eher wie eine Bestandsaufnahme wirkt, da sogar die TAN-Liste diesen Anforderungen gerecht wird, kann ernsthaft angezweifelt werden, ob App-basierte TAN-Verfahren im mobilen Onlinebanking dieser Bedingung in allen Fällen Rechnung tragen.

Wegen seiner Vorreiterrolle lag es nahe, die *S-pushTAN-App* als Repräsentant für App-basierte TAN-Verfahren auszuwählen. Ferner haben die *S-pushTAN-App* und die *DKB-pushTAN-App* mit der Star-Finanz GmbH, eine Tochterfirma der zur Sparkassen-Finanzgruppe gehörenden Finanz Informatik GmbH, den gleichen Hersteller und weisen deshalb entscheidende Ähnlichkeiten auf. Demnach lassen sich getroffene Aussagen über die *S-pushTAN-App* wahrscheinlich auf die *DKB-pushTAN-App* übertragen.

III. Eine Sicherheitsanalyse der Sparkassen-Apps

Die beiden Apps *Sparkasse* und *S-pushTAN-App* haben zum Teil ähnliche, zum Teil unterschiedliche Sicherheitskonzepte, die im Folgenden dargestellt werden. Zum Zeitpunkt der Untersuchung war Version 2.7.1 (Build 27269) für die App *Sparkasse* [10] und Version 1.0.4 (Build 404) für die *S-pushTAN-App* [11] im Google Play Store aktuell. Als Gerät für die Untersuchungen diente ein LG Nexus 5 mit Android 5.1.1 (Lollipop).

Inbetriebnahme: Während die App *Sparkasse* auf beliebig vielen Geräten installiert und nur mit dem Wissen über die Legitimations-ID und das Passwort verwendet werden kann, muss das pushTAN-Verfahren bei der Sparkasse beantragt und im Anschluss über einen Registrierungsbrief und Erstzugangsdaten freigeschaltet werden. Erst nach Freischaltung kann die

S-pushTAN-App zur Generierung von TANs verwendet werden. Falls man die App auf mehreren Endgeräten verwenden will, muss jedes Gerät einzeln über einen separaten Registrierungsbrief freigeschalten werden.

PIN: Beide Apps verlangen zum Start die Eingabe einer PIN, die bei der initialen Verwendung der App festgelegt wird. Die PIN muss dabei mindestens aus acht Zeichen, einer Zahl, einem Buchstaben und einem Sonderzeichen bestehen. Bei der *S-pushTAN-App* darf die PIN außerdem höchstens fünfmal falsch eingegeben werden bevor die App sämtliche Applikationsdaten löscht. Danach ist eine erneute Inbetriebnahme erforderlich. Die recht restriktive Passwortrichtlinie für beide Apps gibt Grund zur Annahme, dass von vielen Nutzern der Einfachheit wegen die gleiche PIN für beide Apps vergeben wird. Die von uns aufgezeigten Schwächen der *S-pushTAN-App* richten sich aber unabhängig davon nicht gegen die PIN.

Eigene Tastatur: Unter Android gibt es seit der Einführung der virtuellen Tastatur die Möglichkeit, die Standardtastatur durch eine Tastatur eines Drittanbieters auszutauschen [12]. Die Verwendung einer solchen Tastatur birgt prinzipiell die Gefahr eines möglichen Keyloggers. Dadurch könnte beispielsweise die PIN, die beim Start der App abgefragt wird, auch ohne Super-User-Rechte ausgespäht werden. Aus diesem Grund liefert die *S-pushTAN-App* eine eigene Tastatur mit, die zur PIN-Eingabe beim Start verwendet wird. Die App *Sparkasse* ist hier weniger restriktiv und verzichtet auf eine eigene Tastatur.

Device-Fingerprinting: Um das Kopieren der App auf ein anderes Gerät zu erschweren, implementiert die *S-pushTAN-App* Device-Fingerprinting. Dadurch werden für die Hardware und die installierte Android-Version bestimmte, eindeutige Werte abgefragt und gespeichert. Diese Werte werden bei der Registrierung der App initial abgefragt und abgelegt, um sie bei folgender Verwendung der App wieder abzugleichen. Im Falle von Ungleichheit löscht die Applikation wiederum alle Daten und verlangt eine erneute Inbetriebnahme. Obwohl das Verfahren nachvollziehbar ist, sorgt es den Kommentaren im Google Play Store nach bei einigen Nutzern der *S-pushTAN-App* für Frust. Der Grund ist, dass einige Gerätehersteller bei der Auslieferung von Software-Updates Werte verändern, die von dem Fingerprinting-Algorithmus zur Validierung herangezogen werden.

Super-User: Während sich die *Sparkassen-App* auch unter `root` mit einem entsprechenden Warnhinweis betreiben lässt, verweigert die *S-pushTAN-App* komplett ihren Dienst, falls die Umgebung gerootet ist. Dass die *S-pushTAN-App* sich nicht unter `root` betreiben lässt, ist der zentrale Sicherheitsanker dieser App, der es ihr erlauben soll sich gegen Manipulationen durch Schadsoftware zu schützen.

Obfuskierung: Obwohl beide Apps mit *ProGuard* [13] obfuskiert wurden, sind sie problemlos dekompilierbar. Das bedeutet insbesondere, dass auf die Implementierung eines eigenen Class-Loaders und auf starke Möglichkeiten zur Verhinderung von Dekompilierung verzichtet wurde. Der Großteil der Klassen- und Methodennamen hat aber durch *ProGuard*s Umbenennungen seine Aussagekraft verloren.

Certificate Pinning: Beide Applikationen verwenden konsequent SSL/TLS-geschützte Verbindungen, um das Ausspähen und Modifizieren von Daten durch Dritte zu verhindern. Dennoch kann in solchen Fällen die Verbindung oft durch einen *Man-in-the-Middle*-Angriff (MITM) abgehört werden, indem auf dem Gerät ein Zertifikat für einen MITM-Proxy installiert wird. Um sich gegen solche Angriffe zu schützen, implementieren beide Apps *Certificate Pinning*. Dadurch wird nur bestimmten Zertifikaten das Vertrauen ausgesprochen, so dass ein MITM-Angriff nicht mehr ohne Weiteres möglich ist.

Repacking-Schutz: Repacking bezeichnet das Dekodieren, Modifizieren, Enkodieren und letztendlich Signieren einer App mit einem neuen Schlüssel. Zur Umgehung von sicherheitsrelevanten Funktionen ist Repacking eine übliche Vorgehensweise, die sich aber auch zum Reverse Engineering eignet. Beide Apps – insbesondere die *S-pushTAN-App* – implementieren Schutzmaßnahmen um Repacking zu verhindern. Zum einen soll somit verhindert werden, dass die App sich trivial modifizieren und verbreiten lässt, zum anderen erschwert dieser Schutz die dynamische Analyse.

Screenreader-Schutz: Unter Android gibt es die Möglichkeit sogenannte *Screenreader* zu installieren. Diese dienen der barrierefreien Verwendung des Androidgeräts, bieten aber auch einen legitimen Weg um sensitive Daten ohne `root` auszulesen. Ein Schutz gegen diese Möglichkeit wird nur von der *S-pushTAN-App* implementiert.

PROMON Shield: Ein gravierender Unterschied in der Sicherheit der beiden Apps ist die Verwendung einer *Native Library*. Während die *Sparkassen-App* ausschließlich auf Java zurückgreift, verwendet die *S-pushTAN-App* zusätzlich das native *PROMON Shield* [14] des proprietären Anbieters PROMON. Die Verzahnung mit dieser Bibliothek ist in der aktuellen Version der *S-pushTAN-App* äußerst eng. So wird ein Großteil der intern verwendeten Strings in die Bibliothek ausgelagert, die dann über UIDs abgefragt werden. Ferner setzt PROMON Shield statisch-öffentliche, eigentlich konstante Felder im Java-Code über Reflection, um die statische Analyse weiter zu erschweren. Die Bibliothek selbst ist zumindest zum Teil durch einen kryptographischen Schlüssel geschützt, der beim Laden zur eigenen Entschlüsselung verwendet wird. Nicht nur das Reverse Engineering der Bibliothek wird dadurch erschwert, sondern auch das Patchen der Bibliothek erfordert dadurch zusätzliche Schritte.

PROMON übernimmt für die *S-pushTAN-App* im Prinzip alle oben genannten, sicherheitsrelevanten Überprüfungen und ruft dafür Callbacks im Java-Code auf. Dabei wird periodisch überprüft ob ein Debugger verbunden ist, ob die App in einem Emulator läuft, ob sie neu gepackt wurde, ob ein Screenreader installiert ist oder ob versucht wird PROMON über *Native Code Hooks* auszuhebeln.

IV. Mögliche Angriffe gegen das App-basierte TAN-Verfahren

Wie zu Beginn erläutert, will die Sparkasse mit dem pushTAN-Verfahren eine Zwei-Faktor-Authentifizierung implementieren, die den Kunden auch bei unachtsamen Umgang mit

seinem Zugang schützen soll. Eine klassische Zwei-Faktor-Authentifizierung umfasst aber eine Wissens- und eine Besitzkomponente. Dieser Grundsatz wird mit dem pushTAN-Verfahren gebrochen, da „keine Zusatzgeräte nötig“ [3] sind. Eine Verbindung aus der App *Sparkasse* und *S-pushTAN-App* erlaubt Transaktionen von ein und dem selben Gerät zu veranlassen und zu bestätigen. Aus der *S-pushTAN-App* ist es sogar möglich, die angezeigte TAN direkt in die App *Sparkasse* zu übertragen, ohne diese abzutippen oder zu kopieren. Die starke gegenseitige Integration der Apps lässt die beworbene Entkopplung der Kanäle unserer Auffassung nach nicht erkennen und wirft die Frage auf, warum überhaupt noch zwei Applikationen verwendet werden. Der einzige – jedoch ebenfalls wissensbasierte – Zusatzschutz ist die PIN.

Letztendlich kann insbesondere die Implementierung der *S-pushTAN-App* zwar als technisch stark und hochwertig, deren grundsätzliche Konzeption aber als schwach und angreifbar bewertet werden. Aus diesem Grund ergeben sich vielfältige Angriffsmöglichkeiten, wobei im Folgenden die erfolgsversprechendsten vorgestellt werden.

A. Angreifermodell und Rahmenbedingungen

Das Ziel unserer Demonstration ist ein *technischer Angriff* gegen das App-basierte TAN-Verfahren der Sparkassen. Ein technisch erfolgreicher Angriff erlaubt es einem Angreifer, eine oder mehrere Transaktionen auf technischer Ebene auszulösen oder zu manipulieren. Wird eine Transaktion vom Angreifer ausgeführt, erfolgt dies ohne das Wissen oder den Willen des Opfers. Im Falle einer Transaktionsmanipulation verändert der Angreifer eine vom Opfer wissentlich und willentlich erstellte Transaktion, dessen Auftragsdaten jedoch frei von Fremdeinwirkung sind (kein *Social Engineering*).

Das Angreifermodell basiert auf einem Verwender der *S-pushTAN-App* in Verknüpfung mit der App *Sparkasse* nach dem vorgeschriebenen Nutzungsbild der Sparkasse. Konkret bedeutet dies, dass das Betriebssystem keine Modifikationen, insbesondere kein `root`, aufweist. Jedoch ist für die vom Nutzer verwendete Kombination aus Gerät und Androidversion ein `root`-Exploit bekannt, der es erlaubt das Betriebssystem zu rooten ohne Daten zu verlieren. Dadurch entsteht auch für einen Angreifer die Möglichkeit, einen solchen Exploit auszuführen und entsprechenden Schadcode zu platzieren.

B. Reverse Engineering der Transaktionsprotokolle

Der aus Angreifersicht attraktivste und auch schädlichste Angriff wäre das Reverse Engineering der Transaktionsprotokolle der *S-pushTAN-App* und die Implementierung eines eigenen Clients. Ist dieser Schritt erfolgreich abgeschlossen, müssten nur noch Opfer-spezifische Daten wie Session-Keys, die im Rahmen der Inbetriebnahme erzeugt werden, erlangt werden. Diese Daten könnten durch Kopieren bei physischen Zugriff, durch einen MITM-Angriff oder mittels Instrumentation ausgelesen werden. Da das Reverse Engineering des Protokolls Opfer-unspezifisch und separat erfolgen kann, beschränkt sich der Angriff auf die Daten der *S-pushTAN-App*. Im Anschluss könnte ein Angreifer beliebig viele gültige TANs erzeugen.

Obwohl der Angriff ein hohes Schadenspotenzial aufweist, verlangt er auf der Seite des Angreifers einiges an Entwicklungsaufwand. Das Certificate-Pinning und der Repacking-Schutz verhindern zumindest, dass das Protokoll einfach über einen Man-in-the-Middle-Angriff mitgelesen werden kann. Nicht zuletzt wird das Reverse Engineering durch die getroffenen Obfuskierungsmaßnahmen erschwert. Trotzdem konnten im Rahmen dieser Untersuchung Teile des Protokolls durch App-Interception ausgelesen werden, womit wir betonen wollen, dass die Entwicklung eines eigenen Clients im Bereich des Möglichen liegt.

C. Kopieren der TAN-App

Ein anderer naheliegender Gedanke ist das Klonen der TAN-App inklusive all ihrer Daten. Die Anwendungsdaten sind prinzipiell nur der App selbst zugänglich, weshalb der Angriff `root`-Rechte erfordert. Ein Vorteil des Angriffs wäre, dass die Integrität der App und der Daten vollständig erhalten bleiben kann. Auf der anderen Seite würden für die *S-pushTAN-App* auch Verfahren des Reverse Engineerings erforderlich, um deren Device-Fingerprinting zu verstehen. Die daraus gewonnen Erkenntnisse müssten zur Manipulation am Angreifer-Gerät eingesetzt werden, um die abgeprüften Werte für das Fingerprinting exakt nachzubilden. Neben dem reinen Kopierprozess der App und der Daten, würde also auch ein Skript zum Abfragen bestimmter Geräteeigenschaften notwendig sein. Ferner müsste die PIN erlangt werden. Die einzelnen Schritte des Angriffs könnten sich also wie folgt gestalten:

1) Der Angreifer platziert ein Skript auf dem Gerät des Opfers, das aktiv wird sobald der Nutzer die *S-pushTAN-App* öffnet. Als Resultat liefert das Skript die mitgeschnittene PIN, die App inklusive Daten, sowie die für das Device-Fingerprinting notwendigen Geräteeigenschaften. Je nach gewählter Platzierungsstrategie kann die Zustellung dieser Daten an den Angreifer über das Netzwerk oder auch über USB erfolgen.
2) Das Opfergerät wird nicht länger benötigt und entstandene Spuren sind möglichst zu beseitigen, mindestens in dem Umfang, dass der Eingriff vom Opfer unbemerkt bleibt.
3) Auf dem Angreifergerät muss nun die App sowie die ausgespähten Daten platziert werden. Außerdem muss der Fingerprinting-Algorithmus der App die gleichen Werte lesen wie es auf dem Opfergerät der Fall wäre.

Der eigentliche Realisierungsaufwand für diesen Angriff liegt in dem Verständnis über den Fingerprinting-Algorithmus und welche Werte dieser schlussendlich zur Verarbeitung abfragt. Um an die PIN zu gelangen, scheint nur ein Angriff sinnvoll der sie mitloggt. Ein Brute-Force-Angriff gegen die PIN wird aufgrund der Passwortrichtlinie vermutlich nicht zum Erfolg führen.

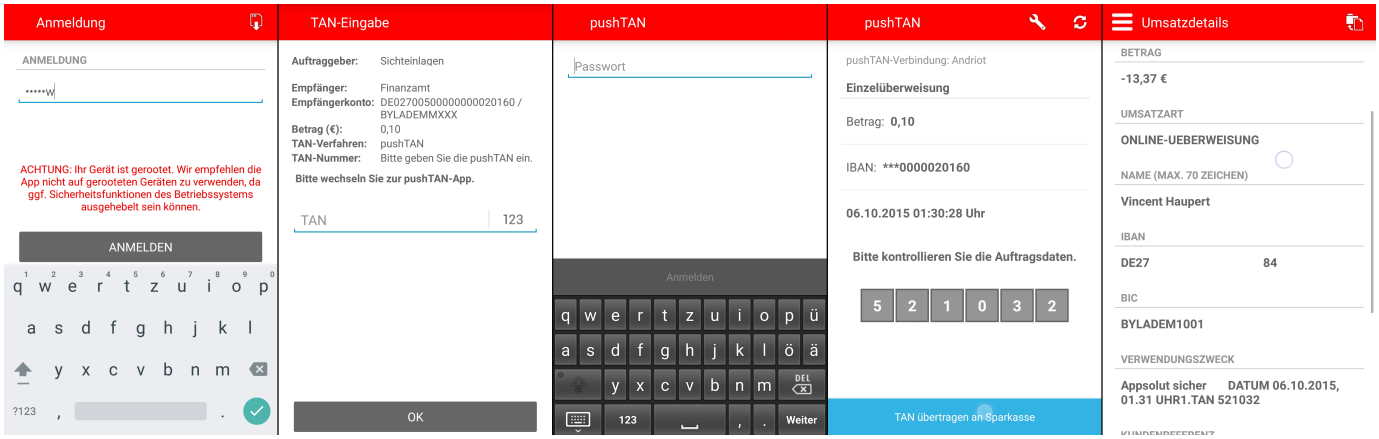

Abbildung 1: Die Transaktionsmanipulation im Überblick (v.l.n.r). Nachdem sich ein Nutzer im Onlinebanking anmeldet hat, gibt er eine Überweisung über 0,10 € an das Finanzamt auf, die per TAN bestätigt werden muss. Nach Anmeldung in der *S-pushTAN-App* kann der Nutzer die TAN 521033 per Click direkt an die App *Sparkasse* übertragen. Obwohl die Auftragsdaten auch in der *S-pushTAN-App* korrekt zu sein scheinen, zeigt ein Blick in die Umsatzdetails, dass mit der TAN 521033 nicht 0,10 € an das Finanzamt, sondern 13,37 € an Vincent Haupert überwiesen wurden.

D. *Instrumentalisierung beider Apps*

Während sich die beiden bereits beschriebenen Angriffe darauf konzentrieren würden, dass die Platform – in diesem Fall Android – trotz aller Schutzmaßnahmen nicht als sicher und vertrauenswürdig angenommen werden kann, sollen nun Möglichkeiten beschrieben werden, die das als sicher beworbene Mobilebanking direkt kompromittieren. Zu diesem Zweck werden sowohl die Banking- als auch die TAN-App instrumentalisiert, um entweder Nutzer-induzierte Auftragsdaten zu manipulieren oder von Seiten des Angreifers beliebige Transaktionen aufzugeben und zu bestätigen. Beide Angriffe haben gemeinsam, dass sie auf das Aktivwerden des Nutzers angewiesen sind, jedoch kein Social Engineering erfordern. Diese Angriffsart wurde letztendlich von uns konkret realisiert und stützt unsere Argumentation.

Eine erste Möglichkeit ist die nicht sichtbare Manipulation von Auftragsdaten. Sobald ein Nutzer eine Transaktion in der Banking-App aufgibt, wird diese, kurz bevor sie an den Server des Kreditinstituts übermittelt wird, von der Schadsoftware verändert. Würde im Folgenden dedizierte Hardware wie ein TAN-Generator verwendet, so würde die Manipulation durch die Anzeige des Empfängers und Betrags auffallen. Da die TAN-App aber auf dem gleichen Smartphone verwendet wird, ist es nicht notwendig, den zweiten Faktor gesondert zu kompromittieren. Die Schadsoftware speichert die durch den Nutzer zuvor in der Banking-App eingegeben Daten und manipuliert die Anzeige in der TAN-App derart, dass die erwarteten Werte präsentiert werden. Dadurch kann eine komplette Transaktion mit beliebigen Empfängerdetails ohne das Wissen des Nutzers durchgeführt werden.

Mit entsprechendem Zusatzaufwand könnte der Angriff ausgebaut werden, so dass die Schadsoftware die TAN-App automatisiert verwendet. Zuerst greift die Schadsoftware die Zugangsdaten zum Onlinebanking ab, wenn der Nutzer eine Transaktion ausführt. Danach wechselt der Nutzer in die TAN-App, die er durch Eingabe seiner PIN freigibt. Nun kann die Schadsoftware verhindern, dass sich die TAN-App beim App-Wechsel automatisch wieder sperrt. Im Folgenden könnte der Angreifer nicht nur beliebige Transaktionen anstoßen, sondern diese auch durch den Aufruf der entsprechenden Funktionen der TAN-App automatisiert bestätigen.

V. Der Angriff: Manipulation von Transaktionsdaten

Der von uns letztendlich realisierte Angriff instrumentalisiert und manipuliert die Apps *Sparkasse* und *S-pushTAN-App*. Hierfür wird der Angriff auf einem Gerät platziert, das der Nutzer zum Onlinebanking mit dem pushTAN-Verfahren nutzt. Der Angriff zeigt seine Wirkung sobald der Nutzer eine Überweisung über die App *Sparkasse* auslöst und über die *S-pushTAN-App* bestätigt, indem die vom Nutzer eingegeben Transaktionen durch die des Angreifers ersetzt werden. Dem Nutzer werden jedoch weiterhin in allen Transaktionsschritten beider Apps die von ihm erwarteten Daten präsentiert. Der Ablauf eines konkreten Angriffs ist in Abbildung 1 dargestellt.

Der Angriff ist als Modul für das Instrumentation-Framework *Xposed* [15] realisiert, das für alle Versionen ab Android 4.0.3 („Ice Cream Sandwich“) verfügbar ist (ausgenommen Android 6). *Xposed* nistet sich, wie in Abbildung 2 zu sehen, tief in Android ein und erlaubt es somit beliebigen Java-Code zu instrumentalisieren. Die Installation von *Xposed* benötigt zwingend `root`, während *Xposed* selbst keine inhärente Bedingung des realisierten Angriffs ist (weshalb er auch eigenständig verwirklicht werden kann). Dennoch bietet *Xposed* für einen *Proof of Concept* eine gute Basis, die uns erlaubt den Entwicklungsaufwand zu reduzieren.

A. *Angriffspunkte der App Sparkasse*

Die *Sparkassen-App* kann bei der Verwendung des pushTAN-Verfahrens zur Absetzung von Überweisungen genutzt werden

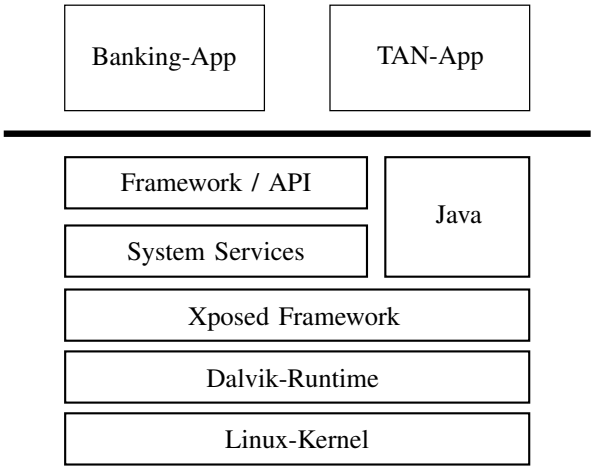

Abbildung 2: Das *Xposed* Framework in der Android-Architektur. Nach der Installation können mit dem Framework beliebige Klassen und Methoden in beliebigen Apps instrumentalisiert werden.

und ist deshalb der Ansatzpunkt unseres Angriffs. Grundsätzlich bleibt zu erwähnen, dass die App sich standardmäßig auch mit `root` verwenden lässt, lediglich ein Hinweis davor warnt, dass dies mit Risiken verbunden sei. In der Eingabemaske zur Erstellung einer Überweisung sind, nach entsprechender Navigation in der App, die folgenden Felder vom Nutzer auszufüllen: Begünstigter, IBAN, BIC, Betrag und Verwendungszweck.

Ziel dieses Angriffs ist es, diese Auftragsdaten zu manipulieren, bevor sie an den Server der Sparkasse gesendet werden, ohne diese Veränderungen zu irgendeinem Zeitpunkt für den Nutzer sichtbar zu machen. Durch Reverse Engineering wurde die Klasse [redacted] und dessen Methode [redacted] als zuständig für das Button-Event Einreichen identifiziert. Durch Instrumentation werden vor der Ausführung dieser Methode die Auftragsdaten des Angreifers vom Server des Angreifers abgerufen und mittels Reflection manipuliert. Nach der Ausführung der Methode werden diese Daten wieder auf die originalen Daten zurückgesetzt und zur späteren Verwendung in der *S-pushTAN-App* gespeichert.

Im Folgenden erscheint ein weiteres Fenster, das nochmals die Auftragsdaten anzeigt und die TAN aus der *S-pushTAN-App* verlangt. Es war im Rahmen des Angriffs nicht nötig, diese Daten erneut anzupassen, da der Server offenbar keine Bestätigung/Kopie der Auftragsdaten sendet oder diese von der *Sparkassen-App* schlichtweg nicht verwendet werden. Im nächsten Schritt muss zur *S-pushTAN-App* gewechselt werden.

B. Angriffspunkte der S-pushTAN-App

Die *S-pushTAN-App* zeigt im Wesentlichen nur die Auftragsdaten und die TAN an. Sie lässt sich allerdings nicht wie die App *Sparkasse* mit `root` betreiben und beendet sich im Gegensatz zu dieser mit einem entsprechenden Warnhinweis. Dieser Schutz musste zunächst deaktiviert werden. Der Hersteller Star-Finanz implementiert selbst keine eigene Überprüfung auf `root`-Zugriff, sondern verwendet hierfür das externe und native Modul *PROMON Shield*. Für *PROMON* gibt es im Java-Code allerdings entsprechende Callbacks, die für den Warnhinweis und das Beenden verwendet werden. Das Callback für die Überprüfung auf `root` in der Methode [redacted] der Klasse [redacted] wurde instrumentalisiert und abgebrochen. Dieses Vorgehen bleibt frei von Konsequenzen, was durchaus überrascht, da die aktuelle Version der *S-pushTAN-App* eine starke Verwebung mit dem *PROMON Shield* aufweist. So sind beispielsweise die Mehrzahl der Zeichenketten in *PROMON* ausgelagert und werden über eine ID abgerufen. Das *PROMON Shield* führt während der Verwendung mehrmals die Methode [redacted] aus, verweigert aber nicht das in ihr ausgelagerte Abrufen von Zeichenketten.

Nach dem Deaktivieren der `root`-Überprüfung lässt sich die App ganz normal verwenden und kann zum Abruf von TANs verwendet werden. Der Server der Sparkasse sendet allerdings die vom Angreifer manipulierten Auftragsdaten und zeigt diese in der *S-pushTAN-App* an. Die angezeigten Daten beschränken sich auf den Betrag und die (maskierte) IBAN. Beide Werte müssen auf die vom Nutzer ursprünglich eingegebenen Werte abgeändert werden. Hierfür stellt der Schadcode die zuvor in der App *Sparkasse* gespeicherten Transaktionsdaten des Opfers wieder her. Zur Manipulation wird die Klasse [redacted] und dessen Methode [redacted] instrumentalisiert. Sie erzeugt aus den vom Sparkassen-Server erhaltenen Daten mehrere Key-Value-Paare zur Anzeige in der App. Dabei werden die beiden Paare Betrag und IBAN entsprechend der abgerufenen Originaldaten des Nutzers abgeändert. Dem Nutzer wird also das erwartete Ergebnis präsentiert, weshalb er die TAN schlussendlich in die *Sparkassen-App* übertragen lässt und den Auftrag somit bestätigt und wirksam macht.

C. Anwendbarkeit für zukünftige Updates

Der demonstrierte Angriff funktioniert nur für die konkreten Versionen der App *Sparkasse* (2.7.1) und *S-pushTAN-App* (1.0.4), für die der Angriff entwickelt wurde. Das liegt darin begründet, dass das vom Obfuskator *ProGuard* eingesetzte Renaming für jeden Kompiliervorgang andere Namen für Klassen und Methoden vergibt. Die Anpassung des Exploits wäre deshalb noch nicht besonders aufwendig und der Angriff könnte sogar derart ausgebaut werden, dass Klassen- und Methodennamen abhängig von der Version nachgeladen werden. Tiefgreifendere Veränderungen an den instrumentalisierten Klassen und/oder Methoden, die bei neuen Versionen ebenfalls zu erwarten sind, würden eine Anpassung des Quelltextes aber zwingend notwendig machen.

Für die *S-pushTAN-App* lässt sich beobachten, dass sie von Version zu Version stärker obfuskiert wird. Das geht in erster Linie auf Verbesserungen des *PROMON Shields* zurück, das immer mehr Teile der Applikation übernimmt und in neuen Versionen besser versucht seine Wirkungsweise zu verschleiern. Mit Veröffentlichung dieser Arbeit wurde zum 16.10.2015 Version 1.0.5 der *S-pushTAN-App* veröffentlicht, die neue Wege bei der Bekämpfung von gerooteten Geräten geht. Diese Version macht unsere bestehenden Lösungsansätze zur Verwendung der *S-pushTAN-App* trotz `root`-Zugriffs zunächst

unwirksam, so dass keine unmittelbare Gefahr durch unsere Tools zu erwarten ist. Dennoch möchten wir darauf hinweisen und davor warnen, dass auch diese Version der *S-pushTAN-App* – und alle zukünftigen Versionen – mit entsprechendem Mehraufwand gebrochen werden kann. Stärkeren Maßnahmen zur Code-Obfuskierung kann immer durch stärkeres Reverse Engineering entgegengetreten werden, so dass lediglich der Realisierungsaufwand unseres Verfahrens steigen wird, während die konzeptionellen Schwächen des App-basierten TAN-Verfahrens nie beseitigt werden können.

VI. Mobile Schadsoftware im Play Store

Wie für jede Schadsoftware stellt sich bei den von uns präsentierten Angriffsmöglichkeiten die Frage, wie Angriffe auf den Geräten der Opfer platziert werden und wie die von uns angenommen Rahmenbedingungen – insbesondere das Erlangen von `root`-Rechten – hergestellt werden können. In einer aktuellen Studie der Universität Cambridge wird gezeigt, dass 87,7% aller Android-Geräte angreifbar gegenüber kritischen Sicherheitslücken sind [16]. Mit sieben von elf untersuchten Verwundbarkeiten können `root`-Rechte ohne physischen Zugang zu dem Gerät erlangt werden. Die Studie zeigt außerdem, dass Sicherheitslücken von Seiten der Hersteller nur langsam, wenn überhaupt, geschlossen werden, was dazu führt, dass viele Geräte auch nach der Entdeckung einer Lücke lange Zeit verwundbar bleiben.

Für die Platzierung des Angriffs sind verschiedene Strategien denkbar. Zum einen könnte die App durch einen Nutzer aus Dritt-Quellen installiert werden, wozu er durch Social-Engineering bewegt werden müsste. Wesentlich gefährlicher ist dagegen das Einschleusen von Schadsoftware in den offiziellen Play Store, das von Google nur unzureichend verhindert wird. Maier, Müller und Protsenko haben 2014 bereits gezeigt, dass sogenannte *Split-Personality-Malware*, also das Aufteilen von Schadsoftware in mehrere Teile, die für sich genommen jeweils unschädlich sind, nicht vom automatisierten Review-Prozess des Play Stores entdeckt wird [17].

Dass schadhafte Apps, die im Google Play Store verfügbar sind und Geräte zunächst rooten bevor sie beliebigen Schadcode zur Ausführung bringen, keine Fiktion sind, hat im September 2015 die App “Brain Test” gezeigt [18]. Diese App, welche heute als besonders gefährlich eingestuft wird, war lange Zeit im Play Store verfügbar und hat auf den Geräten ihrer Nutzer verschiedene `root`-Exploits zur Ausführung gebracht, um im Anschluss daran Schadsoftware nachzuladen und auszuführen. Einer Analyse durch Googles Review-Prozess hat sie sich entzogen, indem die automatisierte Analyseumgebung erkannt und sich innerhalb dieser Umgebung unauffällig verhalten wurde.

VII. Resümee

Der realisierte Angriff zur Manipulation der Auftragsdaten, der sich gegen die App *Sparkasse* und gegen die *S-pushTAN-App* richtet, zeigt deutlich die konzeptionelle Schwäche von App-basieren TAN-Verfahren, wenn auf getrennte Hardware für Transaktionsauslösung und -bestätigung verzichtet wird. Dabei lässt der Angriff es zu keiner Zeit zu, dass die Manipulation durch den Nutzer erkannt wird, weil die angezeigten Daten zu jeder Zeit des Transaktionsprozesses den eingegeben Werten entsprechen. Grundsätzlich könnte der Angriff – mit entsprechendem Mehraufwand – derart erweitert werden, dass er eigenständig auf einem Gerät (ohne `root`, aber mit verfügbaren `root`-Exploit) funktioniert. Da sich der demonstrierte Angriff nicht gegen eine technische, sondern gegen eine konzeptionelle Schwäche des Verfahrens richtet, lassen sich App-basierte TAN-Verfahren, und im Speziellen die *S-pushTAN-App*, durch rein technische Nachbesserungen nie vollständig absichern. Für sicheres Onlinebanking empfehlen wir daher, auf den Komfort des mobilen Onlinebankings auf einem einzigen Gerät zu verzichten und Verfahren vorzuziehen, die einen zweiten, vom ersten Faktor unabhängigen Authentifizierungsfaktor darstellen — wie beispielsweise das etablierte chipTAN-Verfahren.

Literatur

[1] B. 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] R. P. 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] B. 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

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

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

[12] G. 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

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

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

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

[16] D. R. Thomas, A. R. 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.

[17] 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.

[18] 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/