# Mobile Pickpocketing: Exfiltration of Sensitive Data through NFC-enabled Mobile Devices

Ryan Caney, Christopher Dorros, Stuart Kennedy, Gregory Owens, and Patrick Tague

December 5, 2013

CMU-CyLab-13-015

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

# Mobile Pickpocketing: Exfiltration of Sensitive Data through NFC-enabled Mobile Devices

Ryan Caney, Christopher Dorros, Stuart Kennedy, Gregory Owens, and Patrick Tague
Carnegie Mellon University
{rcaney,cdorros,sekenned,gowens,tague}@cmu.edu

## ABSTRACT

With the increasing popularity of Near field communication (NFC) in consumer-off-the-shelf devices, more and more applications are taking advantage of the technology in innovative ways. Unfortunately, with the rise of NFC applications, there emerges a variety of vulnerabilities that could leave an unwitting user vulnerable to a data breach. One such potentially devastating attack is mobile pickpocketing, in which an attacker uses a standard NFC-enabled device to read, store, and transmit unprotected personally identifiable information from cards carried by unsuspecting bystanders.

In this paper, we detail the mobile pickpocketing threat, describe inherent vulnerabilities in today's NFC landscape, and explain how easy it is for a malicious user to exploit them. We define physical and distributed models of the attack. We walk through our experience developing a mobile pickpocketing application, including the capabilities of the application on particular NFC-enabled devices. Finally, we explore short-term and long-term defenses against mobile pickpocketing attacks.

## 1. INTRODUCTION

Near Field Communication (NFC) is a wireless communication technology built on the IEEE ISO 14443 Radio Frequency Identification (RFID) standard. NFC was first standardized by the ISO/IEC in 2003 [27]. Until recently, it has remained widely unused outside the commercial sector. Now, with the advent of NFC-enabled smartphones, the technology is increasingly deployed in consumer mobile devices. As NFC hardware becomes more readily available to end users, companies are exploring novel ways to leverage the technology in new and enhanced services.

Foremost among these companies is Google, Inc. (Google). In recent years, arguably no other major company has done as much to facilitate rapid, widespread adoption and deployment of NFC hardware and software in the U.S. market. In early 2011, the company released a streamlined application programming interface (API) to integrate newly-available NFC capabilities in mobile phone hardware with its Android platform [15], and encouraged programmers to develop compelling new NFC applications. Their profit motive soon became clear with the announcement of Google Wallet [14], an NFC-based payment system enabling mobile devices to function as payment cards at NFC-enabled payment terminals. Google argues [38] that a lockable, encrypted payment system using a secure element embedded in a mobile device is safer than a standard credit card. Whether or not these assertions are true remains to be seen. While Google Wallet has suffered from high-profile security flaws [1], none of these resulted directly from its use of NFC.

Although Google's release of Google Wallet positions the tech company as a new and powerful force in consumer adoption of NFC technology, it is only one of the latest companies to enable U.S. consumers to transfer of financial information through RF-enabled devices and tags [11]. Credit card companies have added RFID technology to their traditional magnetic stripe cards, adding a new vector to access the data stored within. These RFID tags embedded in credit cards lack the necessary resources to offer the sophisticated defenses of a smartphone. As we will see, these so-called smart cards will divulge their data to any RFID reader that sends a correct sequence of commands.

The susceptibility of such tags to RFID-enabled attacks is not new [24, 34, 33, 30]; however, the limitations of previously available technologies force an attacker to carry out these exploits with conspicuous, often custom-built RFID equipment. The ability to read personally identifiable information (PII) via RFID with an inconspicuous consumer off-the-shelf device increases the stealthiness of an attack while simultaneously lowering a would-be attacker's barrier to the world of cybercrime. Security researchers analyzing this new variant of RFID attacks dubbed it *mobile pickpocketing*.

A mobile pickpocketing attack leverages the likely proximity between NFC-enabled phones and RFID-tagged cards to gather PII, thereby granting the attacker unauthorized access to the victim's sensitive data. In mid-2011, ID Stronghold, the self-proclaimed "#1 manufacturer of RFID blocking wallets" [6] made a proof-of-concept application: a tic-tac-toe game equipped with malicious code that would surreptitiously exfiltrate data to a remote server [26]. Along with this distributed pickpocketing application, ID Stronghold ported the functionality of its custom-built RFID credit card reader into a mobile application. Demonstrating similar research at Shmoocon 2012, Kristin Paget of the security consulting group Recursion demonstrated a similar attack [36]. Both sets of research were presumably done to promote the two companies' RFID-blocking card cases and wallets.

In addition to mobile applications able to capture and parse credit card information [6], other mobile applications expose

transit card data, such as trip information (departure and arrival locations, dates, times, etc.), and account information (when the card was refilled, how much was added, and the card's current balance [21]). The use of such an application could easily serve the needs of mobile pickpocket and legitimate user alike.

Although these sensational attacks seem viable, few records document their actual occurrence in the wild. Yet, as NFC capabilities become a standard feature on consumer mobile devices, attacks become easier to commit (the tools are more accessible) and more profitable (more victims posses the vulnerable technology). Even so, companies that profit from NFC devices and companies that sell the security products that complement them cannot be trusted to give consumers a complete understanding of the potential risks of mobile pickpocketing. Our research aims to fill that void.

This work discusses the building process for a proof-of-concept Android application that uses a consumer off-the-shelf mobile device's NFC features to read PII from both credit and transit cards. We presume a reasonably skilled attacker could build such an application to exploit naïve RFID communications, exfiltrating sensitive data from unsuspecting users within the attacker's immediate proximity. We developed the application for 2011 Samsung Google Nexus S phones and targeted PNC Visa payWave Virtual Wallet debit cards and Clipper transit cards. We discuss our results and explore ways to secure RFID PII transactions in both the short and long term.

# 2. BUILDING A MOBILE PICKPOCKETING APPLICATION

Inspired by ID Stronghold's Electronic Pickpocket application [28], we developed and tested an application of our own in order to better understand the level of difficulty an attacker might face in doing the same, as well as to see firsthand the strengths and weaknesses of a mobile pickpocketing attack. We worked on the application incrementally, beginning with simple NFC tag interactions, then moving on to communicate with tags using Application Protocol Data Unit (APDU) commands. ISO 7816-4 gives ready access to file system navigation commands, but eliciting valid responses (i.e. retrieving data) from the credit/transit card required a knowledge of proprietary commands. This additional information is unique to the card vendor and, as far as we can tell, not publicly documented. Therefore, in order to identify the specific commands, we reverse engineered ID Stronghold's Electronic Pickpocket credit card reading application [28] and analyzed the code of FareBot, an open source program [9] that reads transit cards. After analysis of the applications, we identified many of these proprietary commands. Applying that list of commands, we began interrogating the PNC and Clipper cards through a process of trial and error. Once we were able to navigate the file system and extract the data, we saw that the proprietary codes were the sole means of security: once the commands are executed, the data emerges in the clear with no encryption whatsoever. From that point, exposing sensitive data was a simple matter of isolating the relevant output, applying labels to better identify the data, and returning the PII for display.

## 2.1 NFC Technologies

NFC is an overarching term for a wide variety of related RFID technologies, each offering its own set of attributes: memory, file structures, and communication protocols. For our initial foray into NFC Android development, we began interacting with MIFARE Ultralight. It was a natural choice: it only holds a maximum of 64 data bytes, does not support encryption, and readable/writable Ultralight tags are cheap (to the point of disposability, making it ideal for adding data to posters or event tickets). Following the Android API's ample documentation and examples [18, 19, 20], building simple applications to read and write data to an NFC tag was straightforward. Even with its limited resources, MIFARE Ultralight is not too small for vulnerability [39], but the chief interest of this work is RFID implementations used to steal private, personal data, requiring exploration of more sophisticated tags.

MIFARE Classic employs the same basic file structure as MIFARE Ultralight, but offers two larger memory sizes, 1024 bytes or 4096 bytes (minus identification data such as serial number, keys and access controls); these MIFARE Classic 1K and 4K tags have enough power to use a modicum of cryptographic protection for stored data, and do it quickly. MIFARE Classic also uses two 48-bit keys for tag authentication, one with read/write permission and one limited to read-only. Incidentally, this cryptography can be broken in about 200 seconds on a laptop [10]. Both the MIFARE Ultralight and Classic tags used in our experiments were factory fresh, and thus had no data PII apart from our own.

Having mastered the simplest protocols for programming present-day NFC technology in Android, we took aim at the embedded RFID chips in Visa payWave credit cards. The Android NFC API made the initial interface with the credit card tags remarkably similar to MIFARE interactions. After establishing our initial connection, we were able to obtain basic information about the card, such as its particular type of NFC tag technology, and being working with Application Protocol Data Unit (APDU) codes to gather PII.

The tags in the tested credit cards follow the standard implemented by Europay Mastercard and Visa (EMV). This standard is loosely based on ISO 14443 and 7816-4. We found that the tags use two primary technologies: NFC A/B and IsoDep. These technologies are defined in ISO 14443 Parts 1 through 4, which describe the subtle differences between NFC A/B technologies, mostly radio frequency modulation in ISO 14443-2 [2] and initialization procedures in ISO 14443-3 [3]. These RFID chips follow the guidelines of ISO 14443 to access the smart card, but each credit card company has its own Application ID (AID) used to access information.

The final tag technology that we encountered is MIFARE DESFire, used in the Clipper transit card. These tag types support more robust, industry-standard encryption (3DES and AES), and hold more information in memory (2, 4 and 8 kilobytes) than the comparatively lighter-weight MIFARE technologies. While DESFire provides a reasonable measure of security, its implementation does not necessarily prevent unauthorized access to PII, as we will soon see.

| Value | Command Description |
|---|---|
| '0E' | ERASE BINARY |
| '20' | VERIFY |
| '70' | MANAGE CHANNEL |
| '82' | EXTERNAL AUTHENTICATE |
| '84' | GET CHALLENGE |
| '88' | INTERNAL AUTHENTICATE |
| 'A4' | SELECT FILE |
| 'B0' | READ BINARY |
| 'B2' | READ RECORD(S) |
| 'C0' | GET RESPONSE |
| 'C2' | ENVELOPE |
| 'CA' | GET DATA |
| 'D0' | WRITE BINARY |
| 'D2' | WRITE RECORD |
| 'D6' | UPDATE BINARY |
| 'DA' | PUT DATA |
| 'DC' | UPDATE DATA |

**Table 1: Interindustry codes defined by ISO 7816 [4] are included for reference in context of our interactions with the Visa payWave card.**

## 2.2 APDU Codes

ISO 7816-4 defines APDU codes as the means of communication between an NFC reader and tag. Since NFC is a half-duplex communication system, APDU codes are exchanged by command and response. The NFC reader sends a command APDU containing a 4-byte header along with 0 to 256 bytes of data; the NFC tag then responds with an APDU containing a 2-byte response status and 0 to 256 bytes of data [4]. Since most NFC tags hold more than 256 bytes of data, these interactions are typically exchanged multiple times before the reader finishes a complete data transaction.

Although credit card RFID chips use unpublished, proprietary commands, any NFC-readable data store understands a common set of interindustry APDU codes: codes for file selection, reading, writing, updating, erasing and appending data, as well as codes asking for the next section of successive bytes in a file. After connecting to an NFC tag with Android's NFC API, sending a succession of APDU codes allow a reader to navigate a card's file system and reveal the data it stores. We saw these inter-industry codes, as defined by ISO 7816 [4], included in Table 1 for reference, as we interacted with Visa payWave card.

The Clipper card uses MIFARE DESFire, which uses a different set of APDU commands, given in Table 2. DESFire EV1, the successor to DESFire supports the interindustry ISO 7816 APDU commands and command structures, which could possibly be seen used in other transit smart cards.

## 2.3 Vulnerable Proprietary APDU Codes

Using knowledge of proprietary APDU codes as the sole defense for smart card RFID authentication is by no means secure, but the challenge of determining those codes remains. Card manufacturers and issuers understandably keep these codes private, yet we saw identical APDU codes reused in each PNC Visa payWave card we tested. Learning one set of codes is a small investment to gain access to an entire class of payment card. But how does one break the code?

| Value | Command Description |
|---|---|
| 0x6f | Get File |
| 0xf5 | Get File Settings |
| 0xbd | Read File Data (followed with file location) |
| 0xbb | Read File Record (requires authentication) |
| 0x60 | Get Manufacturing Data |
| 0x6a | Get Application Directory |
| 0x5a | Select Application (must be followed with proprietary application code) |
| 0xaf | Get Next 256 bytes (If at the beginning of a response it means file has more than 256 bytes) |

**Table 2: MIFARE DESFire APDU commands are included for reference.**

The most straightforward option is brute force, attempting to access the card's data by exhausting all possible APDU code options until the expected result is produced. The space of all possible APDUs is easily narrowed by eliminating the bytes that are not proprietary: the APDU headers. Of course, even a brute force attack with the leverage of a smaller keyspace is not necessarily the most efficient approach. A resourceful engineer can always build on the work of others.

With NFC's popularity on the rise, developers are creating applications to provide useful RFID functionality. Some of those applications are open source. A quick search on Google Play led us to FareBot [9], an application that allows mass transit card users read access to NFC data, which, in the case of Clipper card, exposes a current balance and recent trip information. Reading the freely available code of this open-source application made it simple to retrieve enough information to profile a victim's everyday transit habits after just a single read, as demonstrated in Figure 1. It should be noted that the author of the application does explicitly advocate for better use of the security features in transit cards, arguing that merely keeping the procedures secret is not adequate security. To our gain (and to a potential attacker's), he protests the current state of smart card security through his mobile-pickpocketing-vulnerable open source code.

## 2.4 Reverse Engineering to Expose Proprietary APDU Codes

Some may see increasing security through forced transparency as rash and irresponsible; however, this position implies that protecting the source code of a mobile application provides ample protection for the proprietary codes used to read unencrypted RFID tags. We decided to test this theory against a closed-source application: ID Stronghold's Electronic Pickpocket. As explained in Section 1, this application uses a smartphone's NFC hardware to read sensitive information from RFID-tagged credit cards. The free application limits the exposed payment card data - revealing just enough to encourage a user to consider ID Stronghold's RFID-blocking payment card accessories. While it's thoughtful of ID Stronghold to sanitize the output of their freely available Android application, its inner workings use all of the correct APDUs required to obtain the entire credit card number and expiration date. Turning this free application into a potential gold mine just takes a little reverse engineering.


**Figure 1: Code from the open-source FareBot application provided a straightforward method to read a victim's transit habits through a single reading from the NFC chip embedded in the transit card.**

The application's first layer of protection is its resistance to Android Application Package (APK) backup utilities that require root privileges on the mobile device to run. "Rooting" the phone to gain total control of a device allows code to run in privileged mode, which allows a user to call an application intended to provide legitimate remote backup of phone data to access the APK. Once the APK is isolated, it can be extracted with "APKTool" provided by Google [17]. Next, the classes.dex jar file is extracted using a combination of the standard jar utility and the "APKTool". This file is the the compiled binary of the application a user is attempting to extract, organized in the Android DEX format. Extracted from the classes.dex file are the Java .class files. Finally, the "JD-Gui" tool [5] can be used to inspect the java classes and export the source code. While the code output by JD-Gui is somewhat obfuscated (the originally-chosen variable names are replaced by sequentially assigned characters), the function names remain, leaving proprietary APDU codes completely exposed to analysis.

## 2.5 Limitations of Android's NFC

Our mobile pickpocketing application was developed on 2011 Samsung Google Nexus S phones running Android versions 2.3.6 and 4.0.4. We performed basic usability tests using PNC Visa payWave debit cards issued in 2011 and a Clipper transit card issued in 2012.

The Android operating system imposes limitations on the use of NFC hardware. An Android device will only conduct NFC interactions when the phone is powered on and the screen is unlocked. The device must also have NFC enabled in its wireless and network settings. Provided the above are true, an NFC-enabled Android phone is listening for NFC signals, and will react automatically when it senses a tag. NFC connections are low-range and low-power, making them somewhat tenuous. To heighten usability, in its 2011 NFC API enhancements [31], Google presumed that a user might have multiple NFC-enabled applications on her mobile device. To avoid interrupting a brittle NFC connection by requiring manual selection of an appropriate receiver NFC application, the update introduced streamlining features to direct detected NFC information to the presumed-correct application automatically. After clicking through a selection menu once, the application will then respond to sensed NFC data in the same way for each subsequent interaction without further consulting the user. This enhancement to the user experience also enhances a clandestine attack.

# 3. ATTACK MODEL

Attacks using RFID are well documented [24, 34], but until the advent of NFC in commonly available consumer hardware, these attacks required conspicuous, custom-built readers, in many cases limiting the attacker's stealth. While many of the devices implementing these attacks have the advantage of functioning at ranges far greater than NFC's ten centimeter limitation [32], the mobile pickpocket's weapon is an inconspicuous, everyday object. This is the attack model we assume: a malicious user armed only with an off-the-shelf, NFC-enabled mobile device running mobile pickpocketing software, such as the one described in Section 2. We now consider two vectors: a physical attack, where the attacker uses his own device in close physical proximity to a victim's RFID card; and a distributed attack, where the attacker surreptitiously loads his pickpocketing software on a user's device, thereby enabling remote exfiltration [30]. Let us first consider the former.

As NFC phones continue to permeate the market, a close-range physical attack becomes increasingly plausible. While NFC devices do not yet saturate the market, the number of mobile devices using the technology is increasing rapidly, from 50 million NFC-enabled phones in 2010 to an estimated 300 million in 2013 [12]. The current cost of a Samsung Google Nexus S, the phone we used in our research and development, is less than $200 (not a prohibitive investment for an aspiring data thief). The cost of NFC phones will continue to decrease [8], further easing entry into the world of mobile pickpocketing. While the Android platform does not provide direct access to more sophisticated NFC protocols, a savvy hacker has ample access to documentation and RFID-reading code posted on the Internet, even if she is not able to leverage reverse engineering. Moreover, as the technology becomes more pervasive, more applications will be built, and more open Android RFID programming documentation will certainly follow. Access to hardware and specialized software clearly provides no serious deterrent to an attacker.

Relying on the short range of NFC communications for security is a similarly dubious defense. A would-be pickpocket still has ample opportunity to perform a stealthy attack, despite the distance limitations of an NFC antenna. Our physical attack model includes any attacker who can easily come within close proximity of a victim without raising suspicion.

Given the speed of a potential mobile pickpocketing attack, even a user giving up an RFID-enabled card for legitimate purposes is vulnerable to attack, even if the victim's card never leaves her sight. For instance, a cashier might appear to be using his phone's SMS service while furtively stealing credit card information from an unsuspecting patron. No greater stealth is needed to gain illegitimate access. In everyday crowded spaces, such as public transportation, even a wary eye would have trouble differentiating between benign user and attacker. Clearly the physical limitations of the attack are trivial to surmount.

Physical limitations can also be sidestepped entirely. In recent years, embedded data exfiltration trojans have been observed in Android programs, sending device data to foreign servers [20]. Google Play requires these malicious applications to state their permissions. As a malicious application requires permissions above and beyond those required by its ostensible legitimate functions, a careful user's suspicions might rightfully be raised. However, research shows a user is likely to ignore this subtle discrepancy and simply agree to an application's permissions without critical examination [29], thereby enabling stealth exfiltration of data. The types of seemingly legitimate applications that host NFC-enabled exfiltration trojans are carefully chosen to sidestep the limitations placed on NFC by the Android operating system (see Section 2.5), including active wallpaper and popular games [13]. While these applications run in the foreground, they give an attacker stealth access to any NFC tag within 10 cm of the infected device. Given the frequent proximity of cell phones and wallets (purses, pockets, nightstands, etc.), a successful exploit from an infected device seems all but inevitable: the attack only needs to work once.

# 4. DEFENSES

In the sections above, we have defined a mobile pickpocketing attack and described its viability. We now turn to defenses currently available to a concerned user, as well as larger-scale defenses to better protect all NFC-enabled mobile device users in the long term.

## 4.1 Short-term Defenses

Seemingly, the most obvious defense against mobile pickpocketing is for an RFID-embedded card user to keep her tags more than 10 cm away from any NFC-reading capable device at all times. Even for a wary user, this defense is impossible. Not only do NFC tags and their readers under the user's control naturally tend to inhabit the same physical space, but as in the case of the physical attack modeled in Section 3, it is clear than a user cannot control the NFC readers around her. Any unencrypted data on an NFC-readable card within physical proximity of a reader is at risk of exposure. These measures provide no real security.

In our investigation of more active defensive measures available to current users, we did not bother to test the defensive card sleeves or wallets from either ID Stronghold or Recursion discussed in Section 1. It seemed unnecessary. From our first attempts to read any NFC tag, we observed that any metal in contact with an RFID tag would stop all NFC communications (a fact we recalled frequently, working at our metal desks). This solution is easily extended to a consumer's wallet: aluminum foil is both inexpensive and effective. Provided the foil remains in contact with the RFID enabled section of a card, NFC interactions are entirely stopped. For an attacker limited to everyday, off-the-shelf mobile devices, the aluminum foil defense against mobile pickpocketing is simple, cheap and apt.

For the user who gets value from a card's RFID functionality, aluminum foil is a fine solution. But, for a user who is uninterested in taking advantage of embedded RFID chips, destruction of the chip is another simple, effective data defense that, when correctly implemented, eliminates any possible risk from an attack. We will consider two methods of destruction.

Cooking the card in a microwave oven is an effective way to disable an RFID chip, but the method is risky. Microwaving the card for too long may destroy the non-RFID functionality of the card; microwaving it longer still increases the likelihood of a fire. This is not the safest way to disable an RFID chip.

Any equally effective, less flammable option is to crush the RFID chip inside the card. While safer than the microwave option, it requires a modicum of aim. The RFID chip can be tricky to find. We illustrate this proces in the example of a PNC payWave card in Figure 2, showing both the card itself and a diagram of the location of the chip within the card. Once the RFID chip is discovered, disabling the chip is simple: smashing it with a hammer should render the chip useless [37]. Once destroyed, the RFID chip will no longer communicate, thereby ensuring perfect data privacy.

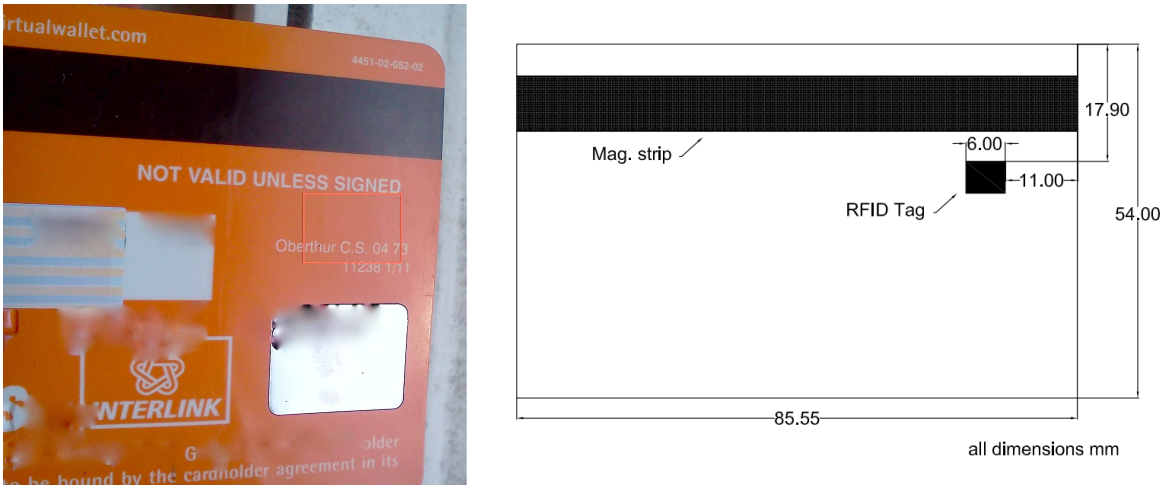
While our recommended measures are cheap, simple, and effective, any defensive measure postulates user awareness of the inherent risks of RFID-enabled cards in the current mobile landscape. For a user to decide on the most suitable defensive measures against mobile pickpocketing, she must first be aware of the technology she possesses and understand the implications of its settings to her data's security. Even a conservative estimate still puts the number of RFID-enabled cards in circulation well over 100 million [36, 12]. Many users are unaware that these vulnerabilities are already part of their everyday lives. An ignorant user is a less wary user, and an easier target for a mobile pickpocket, suggesting longer-term defenses are needed.

## 4.2 Long-term Defenses

We entertain two broad ideas of long-term defenses when it comes to NFC enabled smart cards. The first deals with the smart card transaction itself and ways that could enhance its security. Second, we talk about the removal of RFID tags in physical cards altogether and moving the access vector to a more secure agent, a smartphone with a secure application to transmit data on behalf of the user.

### *4.2.1 Securing Smart Card Transactions*

Current RFID credit and transit cards do not encrypt PII, nor do they require any means of authentication to read PII. Anyone with the correct APDU command sequence can read sensitive data (the ease of obtaining those commands is discussed in Section 2.3). With no built-in defenses to overcome, mobile pickpockets who use an application to read




(a) (b)

**Figure 2: We illustrate the process of locating and targeting the RFID chip within a defunct PNC payWave card, showing both an image of the card and a diagram illustrating the location of the chip.**

sensitive information may not even qualify as hackers; they simply implement an open protocol [35]. With payment and transit cards ready to divulge their data to any inquisitor with the proper sequence of commands, consumer data stored on these RFID cards is at continual risk of exfiltration. As consumer NFC technology becomes more pervasive in mobile devices, that risk will only increase.

This level of exposure is surprising, considering that measures to secure RFID transactions of sensitive information are currently implemented in other domains. Although RFID cards lack the power to perform the necessary functions for data security analogous to those seen in Transport Layer Security (TLS) communications for example, some RFID cards do provide better than non-existent security. E-passports, for instance, implement a PKI, facilitating secure encryption. They also link optical and RFID information, requiring information from both sources to access chip data, which renders the pickpocketing attacks described in this work moot [9]. The Navigo card used in Paris' mass transit system stores no PII; instead of a user or account number, each card is assigned an id number [35]. Using authenticating and encrypting data transactions, a point-of-service (PoS) terminal can take the information stored on a Navigo card and use it to access sensitive information through a secure channel between the company and the point of service. This scheme keeps private data entirely within the control of the company, not out on the streets alongside its vulnerable user. By securing a service, rather than the NFC protocol used to initiate it, sensitive data is protected, despite the limited resources of a smart card.

In this case, increased security comes with a tradeoff in performance, requiring a fair amount of overhead for the PoS terminals and the authorization servers. Placing a greater burden of authentication and encryption on PoS terminals requires more interactions with a remote server, leading to higher computational overhead. As a result, the quickness benefit of using a touchless smart card would become a time consuming hassle that could lead to payment outages if PoS terminals go down.

An authentication system to begin encrypted transactions using preshared keys may appear promising, as a smart card's limited resources make the generation of secure keys difficult, but a single key (or even a set of keys) kept secret between manufacturers and vendors does not provide authentication: it simply moves the problem of discovering secret knowledge (i.e. the discovery of proprietary APDUs in Section 2.3) into a card's keyspace. An attacker could easily acquire a PoS system (they are under $100 on eBay) and extract the keys used to interact with credit cards.

Some suggest replacing the credit card number stored on an RFID tag with a different identification number assigned by a card company. While this certainly thwarts a mobile pickpocket from collecting a credit card number, it still leaves a victim's data compromised. An attacker could use an application such as ours to read this new ID from someone's credit card and subsequently perform a replay attack, emulating the tag to initiate fraudulent payments at any one of many NFC-enabled payment kiosks.

#### 4.2.2 *Moving Consumer NFC from Smart Cards to Smart Phones*

Perhaps U.S. credit card companies are getting wise to their card vulnerabilities. As of January 2012, U.S. credit card companies are looking to not only further secure contactless cards, but "...[move] toward a world beyond plastic..." [23], better able to meet a variety of payment needs. As mobile hardware moves toward widespread implementation in consumer mobile devices, working within the computational confines of RFID chips becomes a far less attractive option. By comparison, industry and consumers alike will grow to favor more robust device-based interactions over similar implementations using smart cards. In this way, the profusion of mobile devices that make mobile pickpocketing such a dangerous problem are simultaneously the most attractive solution.

The advantages of mobile devices over RFID cards are many. With current technology, such devices already possess formidable computational power, which, by Moore's Law, will only continue to grow through the foreseeable future. A typical smart phone has all the resources necessary to implement authentication, cryptography, and key management, allowing handheld devices to perform secure transactions, such as Internet browsing, just as easily as a personal computer. In Google Wallet, these security functions are delegated to a "secure element," further cordoning off the application's cryptographic functions from internal or external access. As in any modern operating system, mobile devices ship with native security features as well, including encryption, certificate storage, and password-protected locking screens. Combining these standard OS features with those additional measures implemented in an NFC payment application, such as Google Wallet [38], provides a degree of defense-in-depth, requiring a user to pass several thresholds of authentication before access to sensitive information is granted. This depth, variety, and strength of security features makes mobile application-based payment an obvious alternative to unauthenticated, unencrypted RFID cards.

While smart phones do offer greater security capabilities than smart cards, it's worth noting that these devices are not without their own security flaws. Google Wallet in particular has had several known problems [25, 7]. Such flaws are to be expected: the technology is still maturing. Better understanding the current security shortcomings in NFC-enabled mobile devices will undoubtedly pave the way for future improvement. As NFC applications endure further scrutiny from the security community, NFC payment and transit application security will advance.

Some important large-scale solutions to better secure users against mobile pickpocketing are possible now. Application market management appears highly effective [16]. While the open Android market provides more variety than its Apple counterpart, its unfettered freedom makes fertile ground for malware. Some attribute Apple's paucity of malicious iOS applications to its policy of strict capability limitations and application review. Amazon recently opened a curated Android market [18]; the impact of such curation on the security of Amazon-provided applications will be interesting to watch.

A regulated marketplace would do well to identify always-on applications, such as live wallpaper, that use NFC functionality. Devices implementing NFC for no clear application-related purpose should be blocked.

# 5. FUTURE WORK

Given the primary motivation for owning a mobile device is convenient, anytime, anywhere data connectivity, this study focused on researching and implementing a physical attack using these increasingly common devices as a vector. Having achieved that goal, we assume that we can replicate Eric Lee's work [30] and simply send captured data. Now that we have developed card-reading capability, the next logical step would be to follow ID Stronghold's track and embed that capability in a seemingly benign application (likely a live wallpaper or game), documenting the process. We could use that legitimate portion of the application as a front for reading and stealthily exfiltrating RFID information gathered by the device on which our application is installed.

Sending data over NFC in Android is becoming even easier. In Android version 4.0, Google added Android Beam [7] to its NFC library to further facilitate peer-to-peer NFC communications. While the Google API suggests Beam will help users "share contacts, web pages, and videos with other devices," [22] any further encouragement of NFC data transfer can also serve a mobile pickpocket. Could a malicious program allow for piggybacking unauthorized, sensitive NFC data along with transmissions from legitimate applications? An indepth look at those possibilities leveraging Android Beam would be an interesting progression from this research.

We found no balance data stored on the Visa payWave card; however, we have found the files where balance data is stored on the Clipper card. As the Clipper is a MIFARE DESFire card, which has yet to be fully broken [9], a next step in our research could involve implementing a card balance alteration attack and assessing any defensive measures currently in place along with any necessary improvements. Along with the current balance updated with each fare interaction, the card's files contain a card ID number. Is the balance on the card associated with a balance record stored on the server side? What other fraud prevention measures are in place? What measures could be implemented? As most mass-transit systems are woefully underfunded, a widespread free-ride hack could prove extremely damaging to the public.

# 6. CONCLUSION

We have shown how easily an attacker with some programming knowledge and access to an NFC-enabled mobile device can create her own application on an Android device that will allow her to read sensitive data from a victim's smart cards. Despite any evidence or lack thereof of mobile pickpocketing in the wild, such attacks pose a legitimate threat. The very nature of the current smart card-NFC reader landscape makes these attacks difficult to defend against without introducing some outside physical mechanism. The companies responsible for deploying smart cards should reconsider security measures in light of the capabilities of these now commonly available NFC readers. Users should be made aware of this emerging technology and its risks, along with known defenses, so they can choose a course of action most appropriate to their individual needs. Given the current climate of NFC implementation, the longer the public is ignorant, the more vulnerable it will become.

# 7. REFERENCES


[1] Identity stronghold. `http://www.idstronghold.com/`, accessed 1-Nov-2012.

[2] ISO 14443-2: Identification cards - contactless integrated circuit(s) cards - proximity cards - part 2: Radio frequency power and signal interface. `http://www.waazaa.org/download/fcd-14443-2.pdf`, accessed 28-Oct-2012.

[3] ISO 14443-3: Identification cards - contactless integrated circuit(s) cards - proximity cards - part 3: Initialization and anticollision. `http://www.waazaa.org/download/fcd-14443-3.pdf`, accessed 28-Oct-2012.

[4] ISO 7816-4: Interindustry command for interchange iso7816 4 smart card standard. http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4.aspx, accessed 28-Oct-2012.

[5] Jd-gui java decompiler. http://java.decompiler.free.fr/?q=jdgui, accessed 28-Oct-2012.

[6] Amazon. Amazon appstore for android. http://www.amazon.com/b?node=2350149011, accessed 1-Nov-2012.

[7] A. Asthana and R. G. S. Asthana. iOS 5, android 4.0 and windows 8 - a review. *IEEE Beacon*, 31:33–43, Mar. 2012.

[8] C. Boulton. Google augments NFC API in android 2.3, Feb. 2011. http://www.eweek.com/c/a/Application-Development/GoogleRefreshes-NFC-ReaderWriter-API-in-Android-23-739296/, eWeek.

[9] E. Butler. Farebot: Read data from public transit cards with your NFC-equipped android phone, Feb. 2011. http://codebutler.com/announcing-farebot-for-android.

[10] N. T. Courtois, K. Nohl, and S. O'Neil. Algebraic attacks on the crypto-1 stream cipher in mifare classic and oyster cards, 2008. IACR ePrint Archive.

[11] dcmugen88. Black epik case cover gel skinfr samsung nexus s i9020. http://www.ebay.com/itm/BLACK-Epik-Case-Cover-Gel-Skinfr-Samsung-Nexus-S-I9020-/250784951850, accessed 1-Nov-2012.

[12] Deloitte. NFC and mobile devices: Payments and more! http://www.deloitte.com/view/en_GX/global/industries/technology-media-telecommunications/tmt-predictions-2012/telecommunications/b421068df67a4310VgnVCM1000001a56f00aRCRD.htm, accessed 28-Oct-2012.

[13] Dominic. Mifare classic detection on android, Apr. 2011. http://mifareclassicdetectiononandroid.blogspot.com/2011/04/reading-mifare-classic-1k-from-android.html.

[14] ECMA International. ECMA welcomes ISO/IEC adoption of NFC standard for short range wireless communication, Dec. 2003. http://www.ecma-international.org/news/Ecma-340-NFCIP-1.htm, press release.

[15] EmvX Blog. Mastercard aligns with visa's U.S. EMV migration plans by publishing its own EMV implementation roadmap, Feb. 2012. http://blog.level2kernel.com/mastercard-us-emv-migration-roadmap/.

[16] eSecurity Planet. Google wallet hacked, Feb. 2012. http://www.esecurityplanet.com/mobile-security/google-wallet-hacked.html.

[17] Google. android-apktool: A tool for reverse engineering android apk files. https://code.google.com/p/android-apktool/, accessed 28-Oct-2012.

[18] Google. Android developers: Advanced nfc. https://developer.android.com/guide/topics/nfc/advanced-nfc.html, accessed 1-Nov-2012.

[19] Google. Android developers: android.nfc package. https://developer.android.com/reference/android/nfc/package-summary.html, accessed 28-Oct-2012.

[20] Google. Android developers: Nfc basics. http://developer.android.com/guide/topics/nfc/nfc.html, accessed 1-Nov-2012.

[21] Google. Android developers: Sample code: Nfcdemo. http://developer.android.com/resources/samples/NFCDemo/index.html, accessed 7-May-2012.

[22] Google. Beaming ndef messages to other devices. https://developer.android.com/guide/topics/nfc/nfc.html#p2p, accessed 1-Nov-2012.

[23] Google. Google wallet: How it works: Security. http://www.google.com/wallet/why/security.html, accessed 1-Nov-2012.

[24] Google. Coming soon: Make your phone your wallet, May 2011. http://googleblog.blogspot.com/2011/05/coming-soon-make-your-phone-your-wallet.html.

[25] A. Greenberg. Hacker's demo shows how easiy credit cards can be read through clothes and wallets, Jan. 2012. http://www.forbes.com/sites/andygreenberg/2012/01/30/hackers-demo-shows-how-easily-credit-cards-can-be-read-through-clothes-and-wallets/.

[26] G. Hancke. Eavesdropping attacks on high-frequency RFID tokens. In *4th Workshop on RFID Security (RFIDSec)*, July 2008.

[27] Identity Stronghold. Google play: Electronic pickpocket RFID. https://play.google.com/store/apps/details?id=com.idstronghold.CCReaderMkt, accessed 1-Nov-2012.

[28] InformationWeek. Curated app markets could help android security, Aug. 2011. http://www.informationweek.com/news/security/mobile/231300342, accessed 1-Nov-2012.

[29] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A conundrum of permissions: Installing applications on an android smartphone. In *Proc. Workshop on Usable Security*, Mar. 2012.

[30] E. Lee. NFC hacking: The easy way. In *DEFCON 20*, July 2012.

[31] Lookout Mobile Security. The lookout blog: Search results for: chinese data phone. http://blog.mylookout.com/?s=chinese+data+phone, accessed 1-Nov-2012.

[32] McAfee. Android malware surges 76%, iphone untouched, Aug. 2011. http://www.electronista.com/articles/11/08/23/mcafee.shows.android.facing.huge.spike.in.malware.

[33] C. Miller. Exploring the NFC attack surface. In *Black Hat USA 2012*, July 2012.

[34] Near Field Communications World. Google adds more NFC features with android 2.3.3, Feb. 2011. http://www.nfcworld.com/2011/02/09/35866/google-adds-more-nfc-features-with-android-2-3-3.

[35] Near Field Communications World. A definitive list of NFC phones, May 2012. http://www.nfcworld.com/nfc-phones-list/.

[36] Smart Card Alliance. New visa paywave issuers and merchants sign up for faster, more convenient payments, Sept. 2007. http://www.

smartcardalliance.org/articles/2007/09/20/new-visa-paywave-issuers-and-merchants-sign-up-for-faster-more-convenient-payments.

[37] The Smartphone Champ. Second major security flaw found in google wallet...rooted or not no one is safe (video), Feb. 2012. http://thesmartphonechamp.com/second-major-security-flaw-found-in-google-wallet-rooted-or-not-no-one-is-safe-video/.

[38] University of Pittsburgh. Panther card. http://www.pc.pitt.edu/card/services.html, accessed 1-Nov-2012.

[39] J. Wu, L. Qi, N. Kumar, R. S. Siva Kumar, and P. Tague. S-SPAN: Secure smart posters in android using NFC. In *13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012.