Mobile Point of Scam: Attacking the Square Reader

Alexandrea Mellen
Undergraduate

John Moore
Undergraduate

Artem Losev
Undergraduate

Department of Electrical and Computer Engineering
Boston University
Boston, MA
{almell, jmoore15, artlosev}@bu.edu

Abstract

We consider the security of Square, Inc.'s mobile card-reading device, the Square Reader, across multiple models, as well as the associated Square Register app where relevant. In doing so, we identify a number of vulnerabilities in the device that allow malicious merchants to initiate fraudulent transactions and, with minor device modification, skim credit card information of unsuspecting customers. We highlight that since mobile card-reading devices like the Square Reader are necessarily compact, cheap, and compatible with a broad range of commodity smartphones, they pose new security challenges over traditional payment-processing hardware. These challenges in turn expose an attack surface that is relatively new and unexplored given the infancy of mobile point-of-sale systems compared to their non-mobile counterparts. We investigate this attack surface and find a number of vulnerabilities that confirm that even current "secure" mobile point-of-sale systems suffer from software and hardware design flaws, leaving them vulnerable to both third parties and malicious merchants.

I. Introduction

The rise of smartphones with Internet capability has enabled the recent formation and surge of a new mobile transaction processing market. A number of providers including Square, Inc., PayPal, and Intuit offer mobile transaction processing services and have created mobile point-of-sale systems that allow merchants to process transactions using phones and tablets in place of expensive retail hardware [1][2][3]. Often these providers offer their hardware to new merchants for free and validate new merchants with a simple and largely automated process. This new convenience and lower barrier to entry of transaction processing especially appeals to street vendors, coffee shops, salons, and other small businesses, who have bolstered the mobile payment industry with hundreds of millions of dollars and growing annual sales volume [4][5]. As a testament to the prevalence of Square's services alone, and by extension its mobile card-reading devices, it is worth noting that Square processed $20 billion in transactions in 2013 and was expected to process $30 billion in 2014 [4]. In fact, Square has processed as much as $100 million in a single day [6].

The convenience and lower barrier to market entry brought on by the rise of mobile payment processing has not come without fresh security risks posed by new hardware designs. Consider that mobile card-reading devices such as the Square Reader are subject to size and cost constraints unparalleled in their similar, non-mobile counterparts. Achieving a small design is necessary to appeal to merchants with limited space, such as street vendors or food truck owners. Minimizing hardware costs ensures a majority of merchants, including smaller, less established ones, can afford the requisite hardware. However, a small and cheap design is at odds with security, since properly implementing encryption at the point-of-swipe becomes more challenging when circuit size and cost allowances are limited.

Also, consider that achieving basic encryption of card data at the point-of-swipe is only half the battle. Mobile card-reading devices must interface with many different smartphone "hosts," i.e. phones running an app such as Square Register, that act as proxies between customers and a provider such as Square. These smartphones may be used for many tasks unrelated to payment processing, including personal use; may run old or insecure operating systems that are prone to mobile malware; may interface with insecure networks either immediately during or interspersed with periods of transaction processing; and may even be intentionally modified by malicious merchants to scam customers. In general, the smartphone hosts present a much less controlled environment than that of dedicated payment hardware, and mobile card-reading devices should implement measures to mitigate the problems that arise when interfacing with untrusted smartphones.

In our investigation of attacks on mobile card-reading devices, we focus on hardware offered by Square, Inc. Founded in 2009, Square is one of the first and main competitors in the relatively new mobile payment processing industry. Since demonstrating its first Square Reader device, the model S1, at the 2010 SXSW conference [7], Square has released three additional revisions of the device [8][9][10]. The Square Reader was criticized as insecure for lacking point-of-swipe encryption until the later S3 and S4 models [11]. Given Square's early entry to the industry, response to initial criticism, and closing of over 200 bugs as

part of a bug bounty program since then [12], we expect that our results are also applicable to hardware and software offered by other, newer industry competitors who have had less time to iterate.

II. Background

Credit cards encode information on magnetic stripes with a varying magnetic field. When a customer swipes a credit card, a magnetic head outputs a voltage signal that is modulated by the magnetic information on the stripe (based on the current induced in a coil of wire within the magnetic head). This varying voltage can be considered audio, and is herein referred to as a "swipe." A microcontroller or other device can decode the signal, which typically follows a Manchester Coding scheme.

The initial models of the Square Reader, models S1 and S2, are quite simple and do not contain any integrated circuitry. The devices consist of a magnetic head connected to a headphone jack with a microphone output, which is sufficient to read a magnetic stripe. By sampling a phone's microphone input fast enough, an application is able to read the small voltages produced by the magnetic head and, by examining the zero-crossings in the signal, decode them into unencrypted credit card information.

Later models of the Square Reader, models S3 and S4, contain integrated circuitry that can read and modify the signal before transmitting it to the phone in order to provide encryption and amplification. However, the signal is still transmitted as a varying voltage, recorded by an app, and decoded into binary digits that represent encrypted or unencrypted data. In the case of encrypted data, the encrypted bits can then be sent to external servers for decryption.

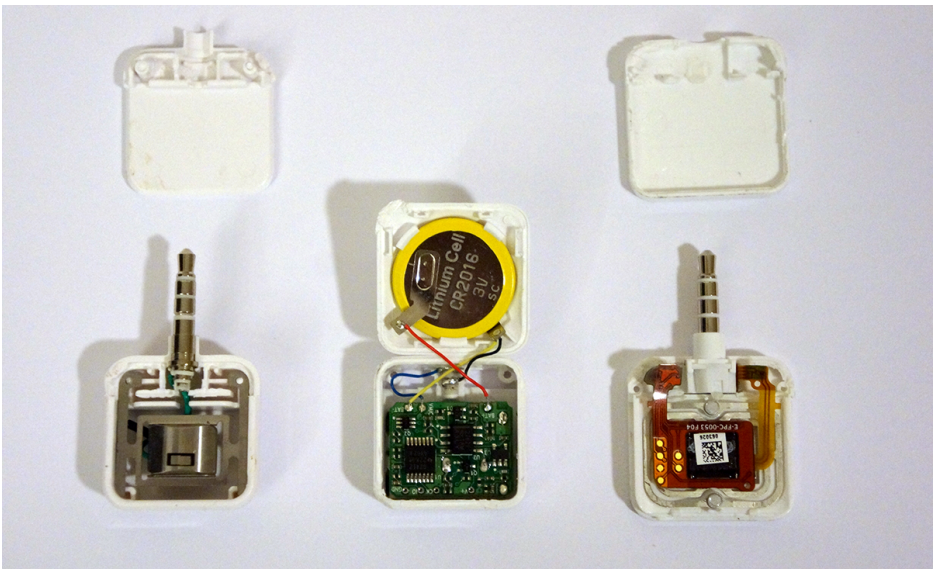

Fig. 1. (Left) The model S2 Square Reader. The device consists of a magnetic tape head connected directly to the microphone and ground terminals of a 3.5mm audio jack. (Center) The model S3 Square Reader. Circuitry including a small microprocessor powered by a battery encrypts card swipes before outputting via the audio jack. (Right) The model S4 Square Reader. Encryption circuitry is smaller and no longer requires a battery.

III. Vulnerabilities

In our analysis, we have uncovered several attack vectors in software and hardware: (1) we have identified that older, deprecated Readers were still usable in the wild until July 2015 despite lacking encryption; (2) we have discovered a way to perform a playback attack to initiate unauthorized transactions; (3) we have implemented a hardware encryption bypass that allows a malicious merchant to successfully convert the latest encrypted Square Reader, the model S4, into an unencrypted Reader without tamper evidence.

A. Software

Given the unencrypted, passive nature of the initial Square Reader device and related prior work discussed in Section V, we first set out to determine the relevancy of attacks on old Square Readers, which we were able to acquire with non-trivial effort on eBay. Since Square officially deprecates old Reader devices and replaces them with new models for free [13], we expected the old, unencrypted Readers would fail when used with the most recent Square Register app. Surprisingly, we found that original Reader devices still worked with the Register app until July 2015 despite replacement of unencrypted Readers in 2012 [9]. That is, merchants with unencrypted Readers were still able to process transactions normally.

Incomplete deprecation of old Reader devices exposes merchants and customers to many of the attack vectors outlined by Frisby et al [14], as discussed in Section V, including malicious operating systems and fake point-of-sale apps. In some cases,

merchants may not be aware of an attack, for example if using a compromised mobile device. However, it is possible for a malicious merchant to intentionally skim credit card information with an unencrypted reader in day-to-day operations if the merchant is able to record swipes of cards. Fortunately, as of July 2015, Square ultimately completely deprecated the Readers in part due to our research [15].

Next we tested for replay vulnerabilities in the current, encrypted model of the Square Reader, the S4 model, along with the official Square Register app. We first recorded several card swipes using the Reader connected to a computer's microphone input. We analyzed the swipe audio from the S4 Reader with the open-source decoding tool SWipe to confirm that the swipes were encrypted [16]. Next we implemented a circuit to feed audio from a computer through a phone's microphone port to the Square app [17]. We attempted to play back the recorded swipes to the Square app running on a smartphone. We found that an initial playback of the swipe successfully initiated a transaction as expected, but that further replays of the recorded audio were rejected. Thus we were able to confirm that Square takes measures to prevent against simple replay attacks.

To verify that replay attack prevention is properly implemented using a transaction counter at the device level [18], we attempted to confirm that replayed swipes were rejected based on their decoded digital contents, i.e. the transaction counter, as opposed to their analog signature or some hash thereof. We modified the swipe waveforms so that the decoded bits were identical but the waveform peaks different using Audacity [19]. The modified swipes were rejected when replayed, but the Square Register app still displayed the associated credit card's last four digits before denying the transaction. Thus we are confident that features such as amplitude and frequency of the analog signal are not used to distinguish swipes from each other, leaving the digital decoded bits as the distinguishing factors.

However, in our analysis of replay attacks, we noticed that we were able to play back recorded swipes out of order. That is, if we sequentially recorded ten swipes labeled 1 through 10 on a computer, we were able to play back swipe 10 and then swipe 1, and both would respectively be accepted. The implication is that although each Reader device contains a transaction counter, Square is not checking whether swipes decrypted on its servers are occurring in the proper order. It is thus possible to stockpile the audio recordings of encrypted swipes and later play them back to initiate transactions, even many days after the swipes are recorded, and even after having processed an arbitrary number of transactions with the same reader in the intervening days.

As a concrete example of an attack based on this vulnerability, consider a malicious merchant Alice and a customer Bob. Bob attempts to make a $5 purchase from Alice, who accepts payments with Square. Alice then opens a recording application on her phone, takes Bob's credit card, and swipes it through the card reader dongle, pretending to attempt a transaction in the Register app. Having recorded a swipe, Alice exits the recording app, quickly switches to the Square Register app, initiates a transaction for $5, and charges Bob by swiping again. To Bob, it appears his card was simply misread on the first swipe, an error he accepts without question. However, many days or weeks later, Alice plays back the stockpiled swipe to complete a $500 transaction on Bob's card. Bob is not expecting the transaction at this time and may or may not notice it on his credit card statement.

We have designed a proof-of-concept application, "Swordphish," that enables easy exploitation of the vulnerabilities outlined in this paper. Swordphish includes a "Playback Attack" mode. This feature records Square Reader swipes and transmits the audio to an external server. The server provides a web interface that allows recorded swipes to be tracked and later played back into the Square Register app in order to initiate and complete delayed transactions.

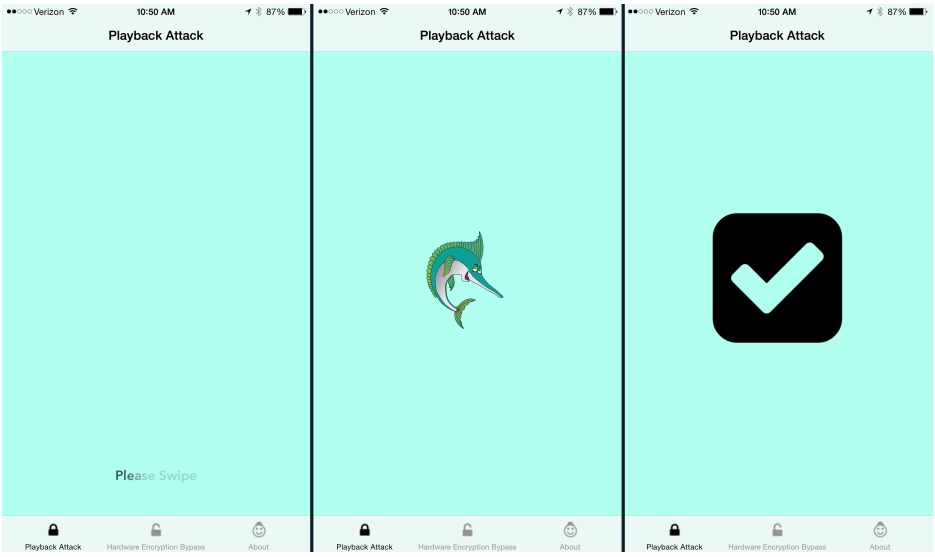

Fig. 2. (Left) Swordphish waits for a card to be swiped in "Playback Attack" mode. (Center) Swordphish detects a swipe and records the swipe's audio, sending it to an external server for storage. If the encoded data is encrypted, the server stores the swipe. (Right) A completed swipe with the Reader yields a check mark to indicate success.

B. Hardware

After analyzing software issues with Square Reader devices, we set out to examine the Reader hardware. The current Square Readers implement point-of-swipe encryption via a chip located inside the Reader as shown in Figure 3. This chip encrypts credit card information continually as the swipe takes place and passes it through the headphone jack into the mobile device and Register app. If it were possible to break this encryption, a malicious merchant could feasibly record a customer's unencrypted credit card information for their own purposes while still maintaining the implicit trust associated with an encrypted device.

Fig. 3. The location of the model S4 Square Reader encryption chip.

In order to analyze the model S4 Reader, we opened it and inspected the internals. Initially, we were unable to open the Reader without effectively destroying the outer casing. The outer casing of the Reader is made up of a soft, white plastic which easily deforms under pressure from external tools that may be used for prying the assembly open such as knives, screwdrivers, etc. The Reader is designed to strongly defend against hardware tampering, or at the very least make it extremely evident. Breakages upon opening the Reader included tearing the headphone jack off of the ribbon cable, tearing the ribbon cable off of the spring base, and permanently deforming the external housing. The case is secured internally with several clips, pins, and adhesive.

Once we were able to successfully open a Square Reader without damaging the internal hardware, which took several careful tries, we inspected the hardware assembly. A ribbon cable connects all main hardware components together: The magnetic head is centered within the assembly and is connected to the chip responsible for encryption, which in turn is connected to the headphone jack. Along the right-hand side of the magnetic head on the ribbon cable are two parallel sets of connection points. Upon probing them with a voltmeter, we were able to identify that the left-hand set of connection points correspond to the magnetic head's tracks, while the right-hand set correspond to the encryption chip's output. Therefore, the latter set of connection points are positioned between the encryption chip and the headphone jack. By jumpering from a connection point corresponding to the magnetic head's tracks to one corresponding to the headphone jack, we are able to bypass the hardware encryption chip.

However, this does not completely downgrade the Reader to an unencrypted version. We were able to identify that noise interference is still present in the signal despite the bypass of the encryption chip. Originally, we attempted to merely cut the portion of the ribbon cable with the encryption chip off, but this created an open circuit in the system by removing other important circuitry. Instead, we applied even, medium pressure lengthwise along the chip and were able to effectively crush the chip to remove all extraneous noise. Alternatively, we were able to produce similar results in an easier fashion by severing the connections to the encryption chip located on the opposing side of the ribbon cable. Both of these methods resulted in a completely unencrypted and useable Square Reader.

Fig. 4. The model S4 Square Reader with both stages of the hardware encryption bypass implemented.

In order to identify hardware encryption bypass as a potential attack vector, we constructed a simple, two-phase system that takes under ten minutes to implement. First, by jumpering from the magnetic head reader to the input of the headphone jack directly, as shown in Figure 4, we are able to bypass the encryption chip but not remove the noise associated with the encryption circuitry. Second, in order to remove this noise, it is necessary to crush the encryption chip with a pair of pliers or sever the connection to the encryption chip with a pair of wire cutters. By placing medium pressure lengthwise along the encryption chip, it is possible to break it but still maintain the connections necessary to successfully read a swipe. Alternatively, placing medium pressure lengthwise along the ribbon cable on the opposite side to the encryption chip successfully breaks the connections to the chip while still maintaining the connections necessary to successfully read a swipe. These two alterations remove the hardware encryption and convert the model S4 Square Reader into an unencrypted device.

Due to Square's current tamper-evident plastic, it is fairly difficult to open a Reader without destroying at least one half of the plastic container. Therefore, the easiest solution we found was to harvest one back piece and one front piece from two Square Readers. We mangled the front piece of one Reader in order to obtain a pristine back piece, then sacrificed the back piece on a separate Reader to obtain a pristine front piece. Once we secured these pieces and implemented the hardware encryption bypass, we were able to reconstruct the assembly using superglue with minimal tamper evidence.

In order to facilitate the exploitation of the hardware encryption bypass vulnerability, Swordphish, first introduced in Section III-A, includes a "Hardware Encryption Bypass" mode. This feature records unencrypted swipes from a modified reader and transmits the audio to an external server for decoding. Once decoding is complete, the server stores the credit card information and transmits it back to the iOS app.

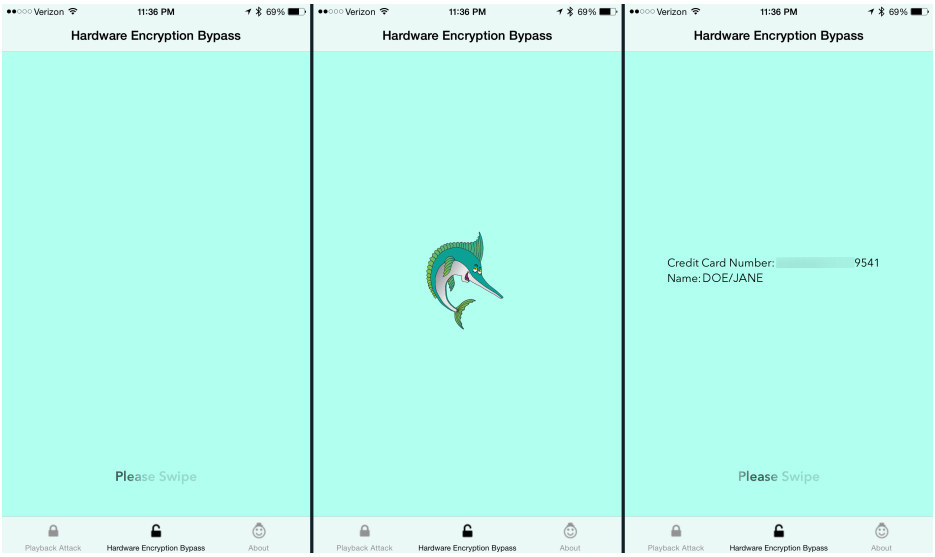

Fig. 5. (Left) Swordphish waits for a card to be swiped in "Hardware Encryption Bypass" mode. (Center) Swordphish detects a swipe and records the swipe's audio, sending it to an external server for storage. If the encoded data is unencrypted, the server extracts available credit card information. (Right) A swipe with the modified Reader yields complete credit card information.

As an example of a practical attack, consider a malicious merchant Alice and a customer Bob. Bob attempts to make a $5 purchase from Alice, who accepts payments through Square. Alice then opens Swordphish on her phone, takes Bobs credit card, and swipes it through the modified reader, pretending to attempt a transaction in the Register app. Having collected a swipe, Alice has at least two potential social engineering routes:

1) Alice pretends the swipe has charged the customer's credit card and allows the customer to leave. This way, the merchant leaves no record of the credit card being used through the merchant, yet has still successfully stolen the user's credit card information.
2) Alice exits Swordphish and quickly switches to the Square Register app. Alice feigns that the Square Reader has not been working as expected lately, and pulls out an encrypted Square Reader to use instead. She then initiates a transaction for $5 and charges Bob by swiping again. To Bob, it appears as though the first Square Reader had merely failed, an error he accepts without question due to his inherent trust in the Square Reader and Square, Inc. However, Alice has actually stolen his credit card information and can use it however she deems fit.

Alternatively, this attack may be used to facilitate the creation of credit card skimmers. The Square Reader is small, compact, and cheap, and the hardware encryption bypass vulnerability is very easy and quick to implement. A malicious third party could easily and cheaply create their own, small credit card skimmer, without much knowledge of engineering required.

IV. Countermeasures

A. Software

One of the simplest and most efficient ways to increase the security and integrity of the entire Square transaction system is to enforce deprecation of the old models of Square Readers, i.e. the unencrypted models that can still be used to initiate transactions and that provide easy access to credit card information. Until May 2015, Square claimed that "All previous readers continue to be secure," [20][21] however, in Section III-A we confirmed the lack of encryption of earlier Reader models and demonstrated their resulting insecurity through mid-July 2015. As long as deprecation of unencrypted Square Readers remained unenforced, malicious merchants had an easy way of acquiring credit card information in the form of audio data [11][22][23]. As of July 2015, Square has officially deprecated old readers in part due to our research [15].

As for protecting against stockpiling encrypted swipe data, implementing and enforcing a transaction counter should prove effective. Enforcing such a transaction counter would involve preventing Square's servers from accepting out-of-order transactions. The S4 model of the Square Reader seems to contain a transaction counter that is unique to a single Reader device and is sent to Square's servers along with each encrypted swipe [15]. However, Square does not currently verify that the transaction count of an arriving transaction's swipe is greater than that of the last swipe processed by their infrastructure. This lack of verification allows for the playback attack described in Section III-A.

B. Hardware

In order to prevent the hardware encryption bypass attack in the future, Square should mount the encryption chip directly on the magnetic head as an assembly. By currently placing the chip directly on the ribbon cable, the Reader is vulnerable to attacks that directly affect the encryption but do not directly affect the ability to record swipes. By designing the magnetic head with the chip directly built-in, the encryption would occur precisely at the point-of-swipe. This way, whenever someone attempted to remove or somehow damage the encryption chip, the magnetic head would be permanently damaged. This would prevent the Square Reader from malicious exploitation via the damaging or destruction of the chip.

Fig. 6. In order to properly implement point-of-swipe encryption, Square should move the encryption chip to the magnetic head as shown.

V. Related Work

Much prior work has been conducted on traditional point-of-sale devices and systems, which we distinguish from mobile ones. Traditional point-of-sale devices tend to be large, are usually mounted to a table or other surface, and are typically found in larger businesses. Their mobile counterparts, in contrast, are typically small dongles or Unix-based systems that are connected to a host smartphone directly. In our summary of related work we focus specifically on mobile variants.

In 2011, Adam Laurie and Zac Franken of Aperture Labs demonstrated the misuse of the Square Register app by passing credit card information through the headphone jack and into the app without a Square Reader [24]. Their goal was to show how a malicious user could purchase credit card information online and launder it with the Square app to convert stolen card data into cash. Since Square requires merchant verification, we expect this attack to be limited and traceable in practice.

In 2012, Frisby et al. [14] performed a security analysis of multiple smartphone point-of-sale devices, which they dubbed audio-jack magnetic stripe readers (AMSRs), and their associated apps. They identified several potential AMSR attack vectors, including network adversaries, fake point-of-sale apps, malicious operating systems, third-party apps, firmware, and hardware. In analyzing the Square Reader, they confirmed that earlier devices are unencrypted and thus exposed to attack from malicious operating systems. They further found that Square added encryption to later devices with reasonable protections against firmware tampering.

During Blackhat 2014, Nils & Jon Butler presented on mobile point-of-sale vulnerabilities, concentrating on "Linux devices, which you pair with a mobile device" [25]. They mentioned Square and other dongles very briefly, calling them "mobile skimming devices" and saying that "swiping your card information into some dude's phone is not a great idea." However, they revealed a number of attack vectors on Unix mobile point-of-sale devices that do not fall within the scope of our analysis.

VI. Conclusion

We have examined the security of the Square Reader, one of many mobile card-reading devices designed to allow merchants to more easily enter the market of processing transactions. In our analysis, we have demonstrated a number of vulnerabilities in the Square Reader, including unenforced deprecation of old hardware, allowance of out-of-order transactions, and insufficient tamper-proof hardware features. We suggest that similar attacks could possibly be performed on other mobile point-of-sale competing systems such as Intuit GoPayments and PayPal Here, which utilize similar end-to-end encryption [2][26]. We emphasize that mobile card-reading devices face additional challenges beyond traditional point-of-sale hardware, given that they are smaller, cheaper, and compatible with commodity hardware. These challenges are manifest in the vulnerabilities that we have identified and in the responses we received to our disclosure reports outlined in Section VII.

VII. Disclosure

In January 2015, we contacted Square about the playback vulnerability described in Section III-A, filing a bug report and suggesting that they enforce transaction counters on their servers. In the following dialogue it became apparent there are several difficulties in doing so with Square's current infrastructure. First, implementing a synchronized counter across the data centers in several timezones is a challenging problem which Square claimed was delaying a timely fix. Next, Square offers a feature where users can store transactions offline and then upload and process them the next time the user goes online. Square claimed this feature further complicated the possibility of enforcing a transaction counter, and opted to simply use out-of-order transactions as a metric in fraud search and investigation as opposed to implementing hard restrictions on out-of-order transactions. As of May 2015, Square has marked the report as resolved and awarded a bounty for it.

Also in January 2015, we filed a bug report about the hardware encryption bypass vulnerability described in Section III-B. Square claimed to be aware that S4 Readers may be modified by bypassing the encryption circuitry without tamper evidence, but did not triage the bug or indicate they would take any steps to fix the vulnerability.

ACKNOWLEDGMENT

We would like to thank Prof. Ari Trachtenberg for his suggestions of potential attack vectors, assistance in editing of the final paper, and guidance in disclosing our findings to Square and the security community.

We would like to acknowledge the assistance of William Vangos on an earlier version of our analysis.

REFERENCES

[1] "Square: Credit card processing & business solutions." [Online]. Available: https://squareup.com/

[2] A. West, "What the paypal here mobile payment system promises," 2012. [Online]. Available: http://www.pcworld.com/article/252962/what_the_paypal_here_mobile_payment_system_promises.html

[3] "Intuit: Easy mobile credit card processing." [Online]. Available: https://payments.intuit.com/mobile-credit-card-processing/

[4] S. Buhr, "Square closes $150 million round at $6 billion valuation," 2014. [Online]. Available: http://techcrunch.com/2014/10/05/square-closes-150-round-at-6-billion-valuation/

[5] O. Kharif, "Square finds itself encircled," 2014. [Online]. Available: http://www.bloomberg.com/bw/articles/2014-10-30/square-mobile-payments-market-catches-up-to-and-passes-pioneer

[6] "Square sellers just made $100m in sales in one day." [Online]. Available: https://squareup.com/townsquare/100m-day/

[7] "Demo of square at sxsw," 2010. [Online]. Available: https://vimeo.com/10216371/

[8] J. Carr, "New square credit card readers begin to arrive," 2010. [Online]. Available: http://eciov.com/tech/new-square-credit-card-readers-begin-to-arrive/

[9] R. Agrawal, "Square adds encryption to its square reader," 2012. [Online]. Available: http://venturebeat.com/2012/03/26/square-adds-encryption-to-its-square-reader/

[10] N. Lee, "Square reveals thinner and more accurate mobile credit card reader," 2013. [Online]. Available: http://www.engadget.com/2013/12/09/new-square-reader/

[11] A. D. Hart, "An open letter to the payment industry." [Online]. Available: http://www.magtek.com/V2/an-open-letter-to-the-payment-industry/

[12] "Square - hackerone." [Online]. Available: https://hackerone.com/square

[13] S. Guarino, "Square notifying existing customers to update new card reader for free." [Online]. Available: http://9to5mac.com/2014/03/29/square-notifying-existing-customers-to-update-to-new-card-reader-for-free/

[14] W. Frisby, B. Moench, B. Recht, and T. Ristenpart, "Security analysis of smartphone point-of-sale systems," in *6th USENIX Workshop on Offensive Technologies*. USENIX, 2012.

[15] "Hackerone: Delayed, fraudulent transactions possible with encrypted square reader devices due to lack of server-side verification of device transaction counter." [Online]. Available: https://hackerone.com/reports/38682/

[16] J. Malone, "Swipe." [Online]. Available: https://github.com/ieatlint/SWipe/

[17] T. Engdahl, "Connecting speaker signals to line level inputs," 1997. [Online]. Available: http://www.epanorama.net/circuits/speaker_to_line.html/

[18] J. Stapleton, *Security without Obscurity: A Guide to Confidentiality, Authentication, and Integrity*. Auerbach Publications, 2014.

[19] "Audacity." [Online]. Available: http://audacityteam.org/

[20] "Square reader options," May 2015. [Online]. Available: https://squareup.com/help/us/en/article/5202-square-reader-options/

[21] "Square reader security," March 2015. [Online]. Available: https://web.archive.org/web/20150304003042/https://squareup.com/help/us/en/article/5202-square-reader-security

[22] M. J. Schwartz, "ipad credit card reader hacked as skimmer," 2011. [Online]. Available: http://www.darkreading.com/vulnerabilities-and-threats/ipad-credit-card-reader-hacked-as-skimmer/d/d-id/1099397

[23] A. Bromberg, "Weekend project: Hacking the square reader," 2013. [Online]. Available: http://andybromberg.com/credit-cards/

[24] P. Roberts, "Researchers: Square card reader provides straight line to illicit cash?" 2011. [Online]. Available: http://threatpost.com/researchers-square-card-reader-provides-straight-line-illicit-cash-080411/75513

[25] Nils and J. Butler, "Mission mpossible," 2014. [Online]. Available: https://www.youtube.com/watch?v=iwOP1hoVJEE

[26] "Gopayment goes global," 2012. [Online]. Available: http://investors.intuit.com/press-releases/press-release-details/2012/GoPayment-Goes-Global--First-Stop-Canada-Unveils-Newly-Designed-Free-Credit-Card-Reader/default.aspx