NFC Payments: The Art of Relay & Replay Attacks

Salvador Mendoza

August 14, 2018

Disclaimer

This white paper is a shortened version of the actual research. Unfortunately, some techniques and exploitation bugs are not fixed yet. It is inadequate to release them without the proper fix from the vendors. However, some bug cases are already closed such as Google Pay. So it is completely detailed.

Content

Introduction

In the last few years, digital payment methods have had an incredible adoption rate in consumer devices around the world. Many big companies are adding NFC(Near Field Communication) support to all sorts of devices to allow consumers to make monetary transactions. Some of these companies are protecting themselves by implementing tokenization as part of the payment technology. However, it is well documented that it is possible to bypass these technologies using simple mechanisms. With all these changes in the NFC ecosystem, the information security field is not well prepared to protect against the increasing new attacks in this area.

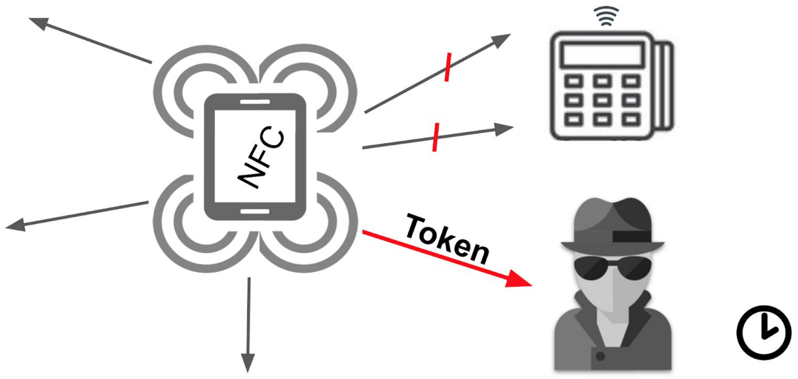
Relay and replay attacks are becoming more common in the payment industry. Getting more complex and sophisticated day by day. We are not just seeing simple skimming techniques but complex attack vectors that are a combination of technologies and implementations involving SDR(Software-Defined Radio), NFC, APDU(Application Protocol Data Unit), hardware emulation design, specialized software, tokenization protocols and social engineering.

In this talk, we will discuss what these attacks are, or what kind of hardware or software could be implemented. Also we will talk about how anyone already has the hardware necessary to carry out one of these attacks or for $35 dollars someone can create a device to do so. Adding that we will show real scenarios where these technologies combined with RFID(Radio Frequency Identification) emulation could exploit any type of NFC transaction. But even worse, how the same attack methods could exploit new NFC implementations for years to come.

This talk uses exploitation hardware and demos; the presentation will include SDR communication, RFID emulation, APDU communication, extraction of data from physical and digital cards.

Replay Attacks

Replay attack is a technique where a malicious user could implement a device to intercept a NFC transaction and redeem it later, using other device or even in different location.



Google Pay is the new version of Android Pay. This application implements cloud-validated cryptograms to make NFC transactions. On March, I found an important issue in the application which could be exploitable for malicious users to intercept and make fraudulent transactions. The applied solution of the vendor, in this case, Google Pay, was that “my report has been closed without providing a fix.”

Vulnerability

NFC(Near Field Communication) protocol is the technology that Google Pay integrated to make digital payments. By default, the protocol has different layers of security to avoid or protect its customers against replay or downgrade attacks. However, the actual implementation of Google Pay does not apply this countermeasure.

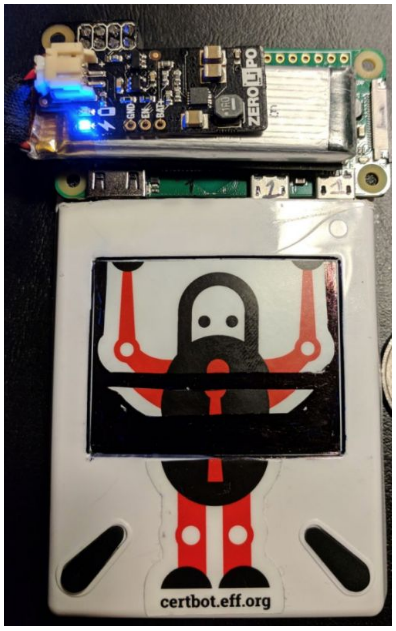
When a person wants to make a NFC transaction with Google Pay, an attacker could intercept the NFC transaction. The malicious user can manipulate the intercepted transaction in different location and with different hardware as well. So, that intercepted transaction could be saved and replay it later with another device to make a payment, spending the money from the original Google Pay customer.

Google Pay Tokens

Each Google Pay transaction is unique. Meaning that every token id is unique as well. If a transaction is not completed in a real PoS or terminal, the token id will keep increasing anyway.
Example of a token incremental:

Token 1: 12 34 56 78 12 34 56 78 D2 40 32 01 48 54 00 00 **03** 71 6F
Token 2: 12 34 56 78 12 34 56 78 D2 40 32 01 48 54 00 00 **04** 14 1F

Important: If a token is not used in a real transaction, that token could be used later because it is not expired or disposed which means a malicious user could implement it with another device such as NFCopy.



Personal NFCopy Project

The AIP

The Application Interchange Profile(82) is in charge to state which level of security the NFC transaction will be implemented to validate a transaction. Basically, the AIP is a tag in the APDU communication which tells the PoS what kind of security the mobile device supports in NFC transaction. This type of information is generated in the communication process between Google Pay and the PoS using APDUs.

The Issue

If an attacker intercepts a Google Pay transaction, he/she could be able to use a third-party application such as SwipeYours from Google Play store to generate a mag-stripe mode transaction. This Android app will generate a mag-stripe transaction(Visa MSD) by changing the AIP values in the NFC connection with the minimum level of security.

82 02 20 40 9f 36 02 00 06 9f ...

82 02 00 80 ...

"Turn the magstripe bit on (set AIP bytes to 0x0080)"

Because the intercepted Google Pay token implements a cloud-validated cryptogram and the EMV system accepts this type of data, the transaction can go through using this mechanism.

Hardware

How difficult is to design and make a tool that intercepted a Google Pay transaction and replay it? A malicious user can program an Arduino and a PN532 board to read the transaction and then emulate it in the terminal. The cost of this device is less than 10 dollars.

Damage Range

This issue in the payment network might affect different NFC wallets from different banks if they are applying the "Intended Behavior" methodology, and even worse, this could affect different countries as well.

Proof of Concept

In this PoC, I am running a Raspberry Pi Zero W with an Acr122u NFC reader which also works as emulator to behave as smart card after a Google Pay token is intercepted.
NFCopy tool: https://salmg.net/2018/03/17/nfcopy-project/

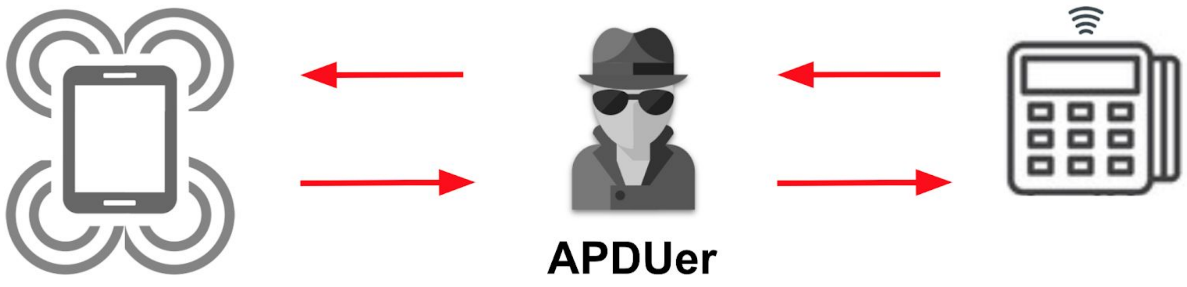
Demo: https://www.youtube.com/watch?v=pomyDepFie0

Timeline:

2018–03–17: Discovered
2018–04–10: Retest in different PoS
2018–04–21: Google Pay team notified
2018–04–21: Google received my bug report
2018–04–23: Bug accepted – Assigned Priority: 1 Severity: 2
2018–04–26: Google feedback, bug not qualified for reward
2018–07–06: Google feedback, “Won’t Fix (Intended Behavior)”
2018–07–24: Public report

Relay Attack

The relay attack is a technique where a malicious user implements a man in the middle attack. The attacker(APDUer) is capable to intercept, manipulate and change the transaction in real time to take advantage of it.



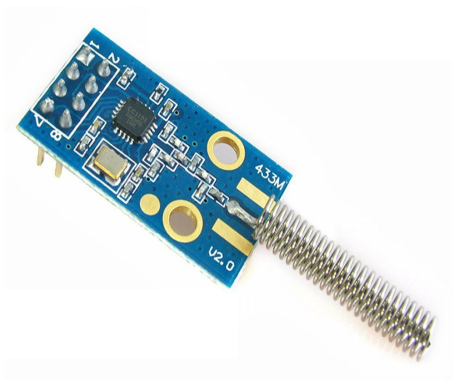
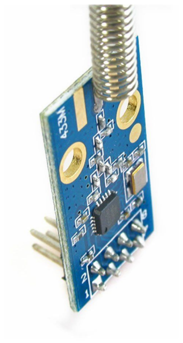
In the left side, we have a device with NFC technology, capable to make digital transactions. In the right side, we have a PoS(Point of Sale System) with NFC technology as well.

The APDUer could design an exclusive communication channel implementing SDR(Software-defined radio) instead of WiFi to avoid interruptions, lag or delays. EMV states that a NFC transaction has to be completed in 500ms, but the terminal is not restricted to finished the transaction if it takes longer.

So this flexibility could be abused with many variable factors; the most important is the difficulty to design a time bounding protocol to protect this.

Transceiver: CC1101

For this research, I used the CC1101 radio transmitter. It is cheap, small and easy to use with different devices like Arduino or Raspberry Pi.

The price of this device is between 4 and 6 dollars in different online stores. It also handles different frequencies and modulations:

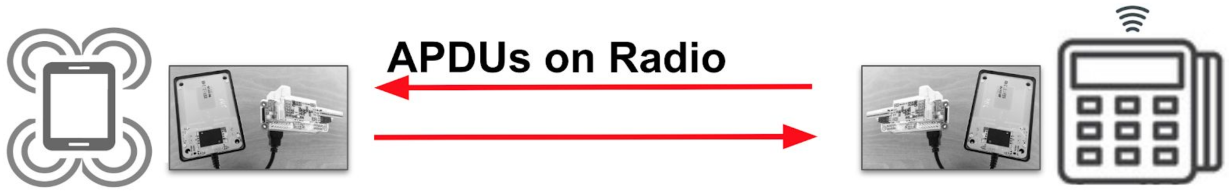
Frequencies(MHz):

- 315
- 433
- 868
- 915

Modulations:

- GFSK(Default)
- MSK
- OOK

The main idea of using a transceiver is to avoid any delays in the communication and to make it as faster as possible.



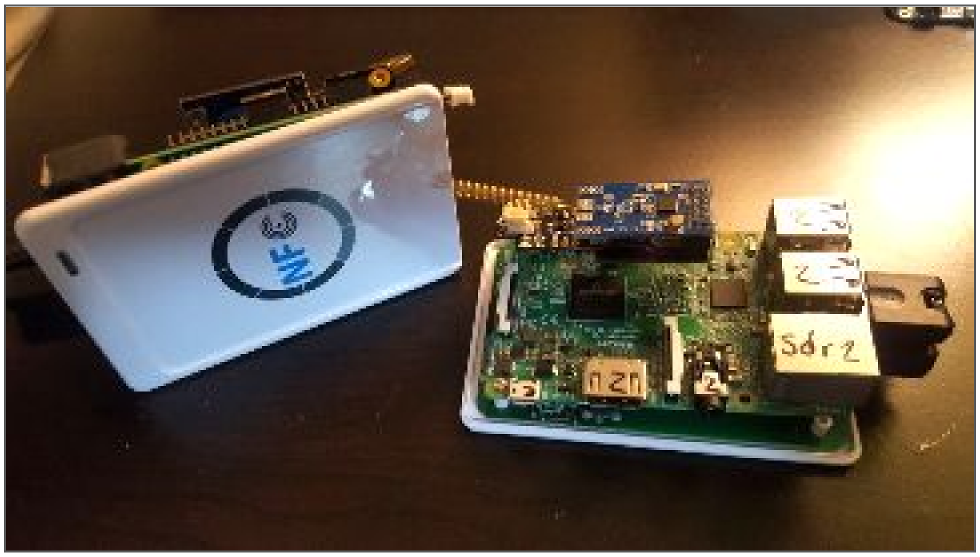
In this graph, we noticed two devices, one close to the PoS and the second close to the mobile device or smart card. Making a real-time connection requires time and the CC1101 has some limitations: it can only transmit 60 bytes per packet. This means that if we have a 200 APDU command, we should cut it in different chunks:

77 60 82 02 20 40 9f 36 02 00 06 9f 26 08 05 81 c8 11 14 17 25 ba 9f 10 20 1f 4a 01 32 a0
00 00 00 00 10 03 02 73 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 5f 34 01
00 9f 6c 02 00 80 57 13 41 36 93 00 20 39 02 71 d2 31 22 01 00 00 05 12 99 99 5f 9f 6e 04
23 88 00 00 9f 27 01 80 90 00 = **Length 200**

Chunks <= 60 bytes

77 60 82 02 20 40 9f 36 02 00 06 9f 26 08 05 81 c8 11 14 17 25 ba 9f 10 20 1f 4a 01 32 a0	**Payload 1**
00 00 00 00 10 03 02 73 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 5f 34 01	**Payload 2**
00 9f 6c 02 00 80 57 13 41 36 93 00 20 39 02 71 d2 31 22 01 00 00 05 12 99 99 5f 9f 6e 04	**Payload 3**
23 88 00 00 9f 27 01 80 90 00	**Payload 4**

For Proof of Concept, I designed the Centinelas project:



Hardware

- Raspberry Pi
- ZERO-LiPO
- Acr122 USB NFC Reader
- LiPo 3.7v 500mAh
- ZERO-LiPO
- CC1101 Transceiver

The library implemented is https://github.com/SpaceTeddy/CC1101

The actual code to make the relay attack is not public, and it will not be until the vendors could have a countermeasure.

Demo: https://www.youtube.com/watch?v=zk-eIJWd14A

References

Peter Fillmore:
https://www.blackhat.com/docs/us-15/materials/us-15-Fillmore-Crash-Pay-How-To-Own-And-Clone-Contactless-Payment-Devices-wp.pdf

Adam Laurie: https://github.com/AdamLaurie/RFIDIOt

Michael Roland:
https://www.usenix.org/system/files/conference/woot13/woot13-roland.pdf