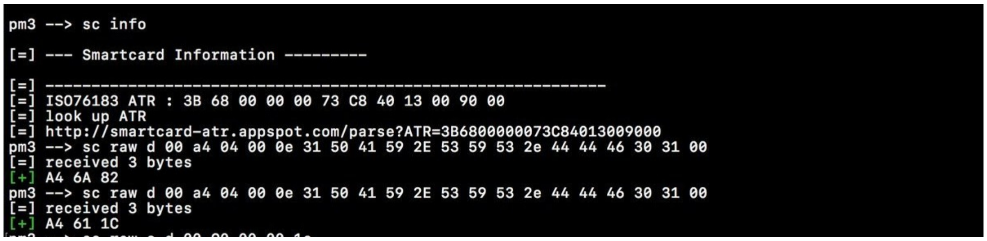
Salvador Mendoza

If you are not making something or inspiring something, you are doing something… wrong.

CONFERENCES, IDEAS, NFC, TOOLS

Proxmark3 RDV4: Extracting Data from Chip-And-PIN Cards



```
pm3 --> sc info

[=] --- Smartcard Information ---------

[=] -------------------------------------------------------------
[=] ISO76183 ATR : 3B 68 00 00 00 73 C8 40 13 00 90 00
[=] look up ATR
[=] http://smartcard-atr.appspot.com/parse?ATR=3B6800000073C84013009000
pm3 --> sc raw d 00 a4 04 00 0e 31 50 41 59 2E 53 59 53 2e 44 44 46 30 31 00
[=] received 3 bytes
[+] A4 6A 82
pm3 --> sc raw d 00 a4 04 00 0e 31 50 41 59 2E 53 59 53 2e 44 44 46 30 31 00
[=] received 3 bytes
[+] A4 61 1C
```

Date: October 18, 2018Author: Salvador Mendoza □ 0 Comments

Intro

The new version of Proxmark3 family(RDV4) contains special features which might help to understand and analyze Chip-And-PIN cards. This new connector is “hidden” under the base case and can be implemented with the new version of the RDV4 repository based on iceman fork.

This slideshow requires JavaScript.

The command is the “SC” (Smart Card). It has certain functions that can be used to take information from the chip. Specifically, I will write about how I did it manually using the raw command initially to extract data from financial chip credit cards. This method could be applied to many different SIM technologies to interact with the chips.

Story

I bought the Proxmark3 RVD4 at defcon this year from HackerWarehouse. I tested certain functions such as Payment NFC cards support, EMV exploitation and raw commands. Also, I changed antennas to test distance. Everything worked as expected.

The only thing left was to test the Chip connector. I did not have the adapter, so I started researching where could I get it; and thanks to Philippe Teuwen, I bought it at Aliexpress:

> *this one seems even more alike https://t.co/VWzCcZU4Kk pic.twitter.com/2w7H4ZlQIi*
>
> *— Philippe Teuwen (@doegox) August 14, 2018*

After I ordered this adapter, I left US to participate in HITB at Singapore. Surprisingly, I had the opportunity to met part of the great team who design and support the new version of Proxmark3:

Dennis and Proxgrind

Personally, I congratulated them for this impressive artwork, and I promised that I will work on the SC command to extract data from PSE(Payment System Environment) cards. Moreover, they had chip card adapters, so I could start researching about that right away.

The Process

The main issue was that we "did not have" enough support for SC commands in the Proxmark3 repository to read specific information from financial Chip cards.

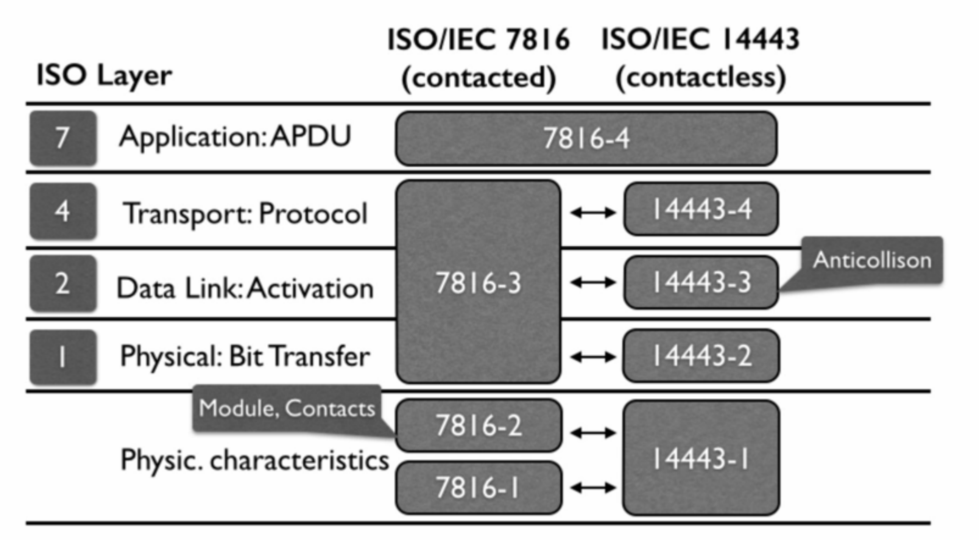
Initially, we had a command which extract the ATR(Answer To Reset) information:

```
pm3 –> sc info
```

As a bonus, the SC command has the RAW parameter to inject specific ISO-7816 commands depending of the chip type that we are playing with.

```
pm3 –> sc raw
Usage: sc raw [h | r | c] d

h       :  this help
r       :  do not read response
a       :  active signal field ON without select
s       :  active signal field ON with select
t       :  executes TLV decoder if it possible
d       :  bytes to send

Examples:

sc raw d 11223344
```

Without specific details, I assumed that I could send raw hexadecimal values using the PSE idea remembering the ISOs relation between contact(chip) and contactless(NFC) technology:



We noticed that they share the same APDU layer which means we can interact them using the similar orders and structure relating the NFC commands. Changing from PPSE to PSE easily;

for more reference you can take a look at Intro to Analyze NFC Payment Methods & Contactless Cards:

Initial changes:

From: 00 A4 04 00 0E **32** 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00 (PPSE)

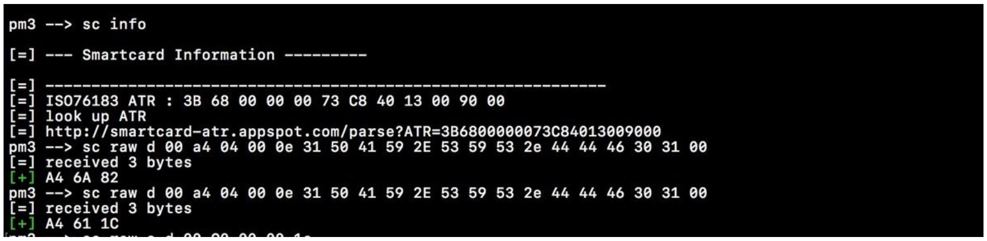
To: 00 A4 04 00 0E **31** 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00 (PSE)

Class byte = 00 (CLA)
Select command = A4 (INS)
Select by name = 04 (P1)
The first record = 00 (P2)

The data is a PPSE constant “filename” value, 2PAY.SYS.DDF01, so we have to convert it:

2PAY.SYS.DDF01 = PPSE = [**0x32**, 0x50, 0x41, 0x59, 0x2E, 0x53, 0x59, 0x53, 0x2E, 0x44, 0x44, 0x46, 0x30, 0x31]

1PAY.SYS.DDF01 =   PSE = [**0x31**, 0x50, 0x41, 0x59, 0x2E, 0x53, 0x59, 0x53, 0x2E, 0x44, 0x44, 0x46, 0x30, 0x31]



```
pm3 --> sc info

[=] --- Smartcard Information ---------

[=] -------------------------------------------------------------
[=] ISO76183 ATR : 3B 68 00 00 00 73 C8 40 13 00 90 00
[=] look up ATR
[=] http://smartcard-atr.appspot.com/parse?ATR=3B6800000073C84013009000
pm3 --> sc raw d 00 a4 04 00 0e 31 50 41 59 2E 53 59 53 2e 44 44 46 30 31 00
[=] received 3 bytes
[+] A4 6A 82
pm3 --> sc raw d 00 a4 04 00 0e 31 50 41 59 2E 53 59 53 2e 44 44 46 30 31 00
[=] received 3 bytes
[+] A4 61 1C
```

At this point, the Proxmark3 received this response:

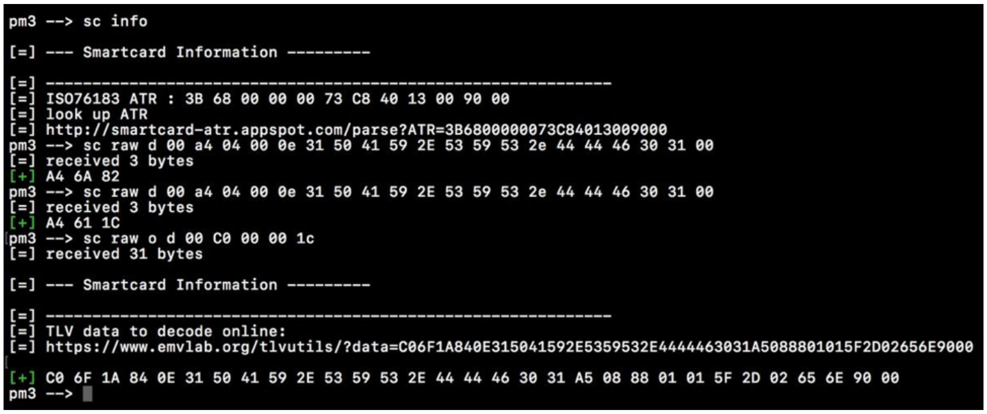
> *[+] A4 61 1C*

Which means “…If the card answers with a “61 XX”, means that the card has “XX” bytes waiting.” to obtain that information, we should send a new command:

> *there is always a way* 🙂
>
> *the key with the iso-7816 is the "GET RESPONSE" after a succeed command. If the card answers with a "61 XX", means that the card has "XX" bytes waiting. We must call the "GET RESPONSE"(00 c0 00 00 XX) where "XX" should be used as "le"*
>
> *— Salvador Mendoza (@Netxing) September 19, 2018*

So we have to request with the "GET RESPONSE" command. Being "1C", the answered number of bytes that the initial "sc raw" obtained.

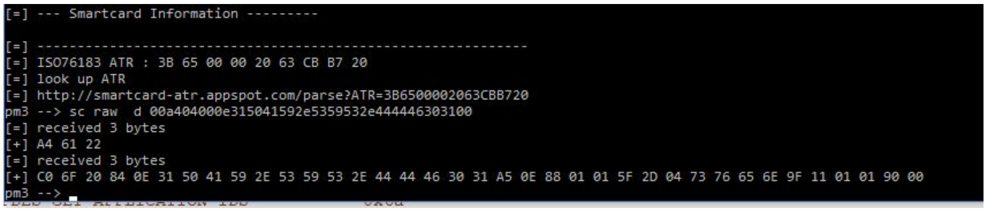
> *pm3 –> sc raw d 00 C0 00 00* ***1C***



```
pm3 --> sc info

[=] --- Smartcard Information ---------

[=] -------------------------------------------------------------
[=] ISO76183 ATR : 3B 68 00 00 00 73 C8 40 13 00 90 00
[=] look up ATR
[=] http://smartcard-atr.appspot.com/parse?ATR=3B6800000073C84013009000
pm3 --> sc raw d 00 a4 04 00 0e 31 50 41 59 2E 53 59 53 2e 44 44 46 30 31 00
[=] received 3 bytes
[+] A4 6A 82
pm3 --> sc raw d 00 a4 04 00 0e 31 50 41 59 2E 53 59 53 2e 44 44 46 30 31 00
[=] received 3 bytes
[+] A4 61 1C
pm3 --> sc raw o d 00 C0 00 00 1c
[=] received 31 bytes

[=] --- Smartcard Information ---------

[=] -------------------------------------------------------------
[=] TLV data to decode online:
[=] https://www.emvlab.org/tlvutils/?data=C06F1A840E315041592E5359532E4444463031A5088801015F2D02656E9000

[+] C0 6F 1A 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 08 88 01 01 5F 2D 02 65 6E 90 00
pm3 -->
```

With this in mind, we are able to automatize scripts or functions to do the "GET RESPONSE" automatically:

> *You mean like this?? pic.twitter.com/Io73lTX2m5*
>
> *— iceman (@herrmann1001) September 19, 2018*



```
[=] --- Smartcard Information ---------

[=] -------------------------------------------------------------
[=] ISO76183 ATR : 3B 65 00 00 20 63 CB B7 20
[=] look up ATR
[=] http://smartcard-atr.appspot.com/parse?ATR=3B6500002063CBB720
pm3 --> sc raw  d 00a404000e315041592e5359532e444446303100
[=] received 3 bytes
[+] A4 61 22
[=] received 3 bytes
[+] C0 6F 20 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 0E 88 01 01 5F 2D 04 73 76 65 6E 9F 11 01 01 90 00
pm3 -->
```

During my trip to Ekoparty, Iceman was able to integrate certain functions in the "SC" module to brute force SFIs files using Adam Laurie work for MasterCard:

> *#eko14 is over, but our curiosity is not. After working to get some info from the chips using the new #Proxmark3 rdv4, using this methodology: https://t.co/bOYNcO9BPJ@herrmann1001 integrated a new command to brute force using some work from Adam Laurie for MasterCard chips.*
>
> *— Salvador Mendoza (@Netxing) September 29, 2018*

And the code could be easily modified to adapt it to another type of cards such as Visa:

> *But it is easy to adapt the code to another type of technology such as Visa.*
>
> *Even we can adapt the code to use the PDOL creation using the same methods from the NFC tech… pic.twitter.com/B8PRTwd2BS*
>
> *— Salvador Mendoza (@Netxing) September 29, 2018*

Now, we have a new functionality in the Proxmark3 that could assist with the creation of new analytical techniques or attacks approaches against Chip-And-PIN cards.

CHIP **CHIP-AND-PIN** **DEFCON** **HITB** **ICEMAN** **ISO7816** **NFC** **PIN** **PROXMARK3** **RDV4** **RIFD** **SC**

Published by Salvador Mendoza

View all posts by Salvador Mendoza