[redacted]

EMV commands

Notes on EMV works on Proxmark3

Table of Contents

- EMV commands
- Table of Contents
  - EMV Implemented parts
  - Working parts of qVSDC
  - `not implemented` parts of EMV
  - Commands
  - VISA(r) transactions
  - Mastercard(r) transactions
  - all commands
  - Useful links

- EMV kernels

EMV Implemented parts

^Top

- Get ATR|ATS
- Get AID by PSE (`emv pse`)
- Get AID by application list (`emv search`)
- Select application (`emv select`)
- Format PDOL (look at next part)
- Execute GPO (`emv gpo` this step and format PDOL)
- Get records from AFL (`emv readrec`)
- Make SDA (check records from GPO)
- Make DDA (`emv challenge` `emv intauth`)
- Check PIN (`not implemented`)
- Fill CDOL1 and CDOL2 (look at next part)
- Execute AC1 (with CDA support) (`emv genac`)
- Check ARQC (bank part) (`not implemented`)
- Make ARPC (bank part) (`not implemented`)
- Execute external authenticate (`not implemented`)
- Execute AC2 (with CDA support) (`not implemented`)
- Check ARQC cryptogram (`not implemented`)
- Issuer scripts processing (`not implemented`)

Working parts of qVSDC

^Top

- Get ATR|ATS
- Get AID by PSE (`emv pse`)
- Get AID by application list (`emv search`)
- Select application (`emv select`)

- Format PDOL (look at next part)
- Execute GPO (`emv gpo` this step and format PDOL)
- Get records from AFL (`emv readrec`)
- Make fDDA (`emv challenge` `emv intauth`)
- External authenticate command (`not implemented`)
- Issuer scripts processing (`not implemented`)

`not implemented` parts of EMV

^Top

They can be implemented, but it needs to know issuer's card keys (usually 3DES) and now this parts can be tested only on special test cards.

Commands

^Top

All this commands are parts of command `emv exec`. command `emv exec` executes EMV transaction. it have parameters:

```
    -j, -J, --jload      Load transaction parameters from
`emv/defparams.json` file.
    -f, -F, --forceaid   Force search AID. Search AID instead
of execute PPSE.
    By default:          Transaction type - MSD
    -v, -V, --qvsdc      Transaction type - qVSDC or M/Chip.
    -c, -C, --qvsdccda   Transaction type - qVSDC or M/Chip
plus CDA (SDAD generation).
    -x, -X, --vsdc       Transaction type - VSDC.
    -g, -G, --acgpo      VISA. generate AC from GPO.
    -w, -W, --wired      Send data via contact (iso7816)
interface. Contactless interface set by default.
```

It works for VISA(r) and Mastercard(r) transactions. It may work with other EMV payment system's card (and it works in general cases that is described in EMV).

VISA(r) transactions

^Top

MSD - Magnetic Stripe mode VSDC - contact transaction qVSDC - contactless transaction

Mastercard(r) transactions

^Top

MSD - Magnetic Stripe mode M/Chip - contact and contactless transaction

Different cards have different modes on/of and different behavior in them. So needs to check card in all this modes. MSD - compatibility mode. Now it work always. But it less secure and in near future it will be slowly) disabled.

all commands

^Top

```
exec             Executes EMV contactless transaction.
pse              Execute PPSE. It selects 2PAY.SYS.DDF01 or
1PAY.SYS.DDF01 directory.
search           Try to select all applets from applets list
and print installed applets.
select           Select applet.
gpo              Execute GetProcessingOptions.
readrec          Read files from card.
genac            Generate ApplicationCryptogram.
```



```
file for emulator.
test             Crypto logic test.
list             List ISO7816 history
roca             Extract public keys and run ROCA test
```

All main commands are parts of EMV specification. Commands than not described there:

`emv scan` - scans card and saves all records to json file. Can be executed with or without tags disassembly.

`emv roca` - extract public keys from cards (part of `emv scan`)

`emv test` - test all crypto code from emv part of proxmark.

Useful links

^Top

EMV specifications http://www.emvco.com/specifications.aspx?id=155

Excelent explanation of EMV
http://www.openscdp.org/scripts/emv/index.html

Fully working terminal written in Ruby. https://code.google.com/p/ruby-pboc2-lib/source/browse/trunk/lib/pboc.rb

EMV kernel written in C++
https://github.com/ntufar/EMV/tree/master/EMV_Library

C EMV library (part of this library uses proxmark)
https://github.com/lumag/emv-tools

Resources (keys, country codes, etc): https://github.com/binaryfoo/emv-bertlv/tree/master/src/main/resources

EMV kernels

^Top

POS terminal checks card and selects one of EMV kernels and launches it for EMV transaction. Different kernels have different rules to make EMV transaction.

This list from:

EMVco Architecture and General Requirement V2.4 volume A. EMVco Entry Point specification V2.4 volume B

- EMVco C-1 Kernel 1 V2.4 for some cards with JCB AIDs and some cards with Visa AIDs
- EMVco C-2 Kernel 2 V2.4 for MasterCards AIDs
- EMVco C-3 Kernel 3 V2.4 for Visa AIDs
- EMVco C-4 Kernel 4 V2.4 for American Express AIDs
- EMVco C-5 Kernel 5 V2.4 for JCB AIDs
- EMVco C-6 Kernel 6 V2.4 for Discover AIDs
- EMVco C-7 Kernel 7 V2.4 for UnionPay AIDs