Owning and Cloning NFC Payment Cards

Crash and Pay

About me

- Security Jerk.
- Failed “Musician”
- Does stuff with computers.

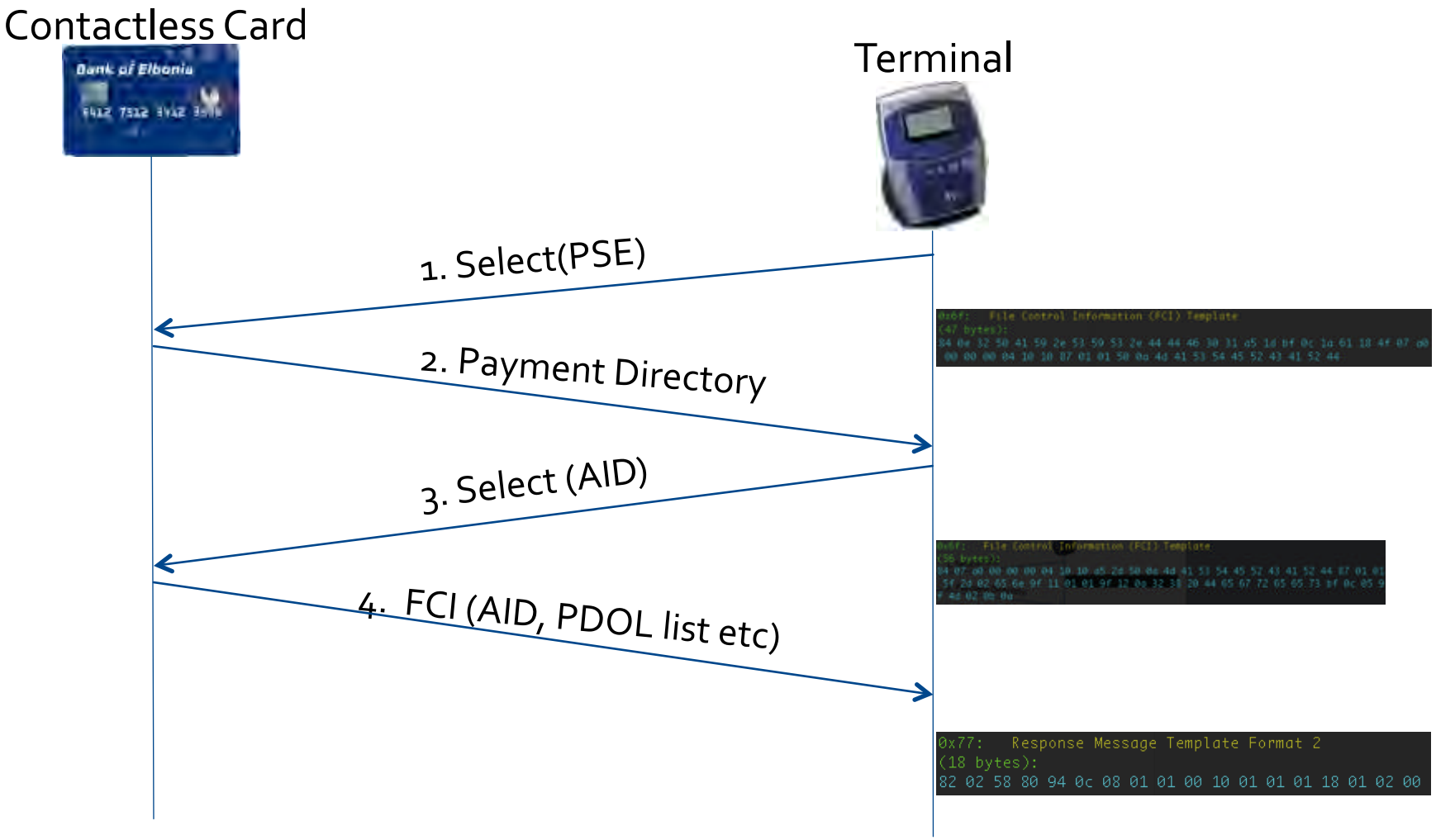
Stuff I'm assuming you know / you can ignore for this talk

- Low level NFC/RFIDs stuff (initialization etc)
- What a credit card is.
- How to do a contactless transaction
- How tradition magstripe cloning is performed.
- Basics of crypto
- The inanity of writing parsers in C.

Handy Definitions!

ATC	Application Transaction Counter	Monotonic counter of transactions performed
UN	Unpredicable Number	Random number used in trandaction
CVV/ CVC	Card Verification Value (VISA)/ Card Verfication Code (Mastercard)	Used to prevent alteration of data on the card.
dCVV/ CVC3	CVV3(Mastercard)/ dynamic CVV(Visa)	Used to prevent alteration of card data and prevent cloning of cards.
TTQ	Terminal Transaction Qualifiers (Visa)	Indicates what kind of card verification the terminal supports
PAN	Personal Account Number	Account Number assigned to the user
PSE	Payment Systems Environment	Tells terminal that the card is a banking card
AID	Application Identifier	Tells terminal what brands the card supports (Mastercard, Visa etc)
PDOL	Processing Data Options List	List of tags we need the terminal to send the card (amount, UN etc).
AFL	Application File Locator	Indicate what records the terminal needs to read.
AIP	Application Interchange Profile	Field to tell the terminal what authentications the card supports

Overview of an NFC Payment (First bit)



How to clone a VISA card

The proceeding information is for education use only.
This is not a license to clone other peoples cards.
Don't blame me if someone does this to you, buy a roll of tin foil and wrap your card in it.

Verification Modes Supported By VISA

- Dynamic CVV - dCVV
  - Legacy magstripe equivalent mode
  - Terrible, broken on release
- Cryptogram Version Number 17 - CVN17
  - Updated to magstripe equivalent mode
  - Lot better than dCVV
- Quick Visa Smart Debit/Credit - qVSDC
  - Reduced EMV mode
  - Defined in standard for speed
- Visa Smart Debit/Credit - VSDC
  - Full EMV mode (i.e CDA)
  - Slower – requires card to be in field for complete transaction

How does the card/terminal know what mode to use?

- 9F66 - TTQ – Terminal Transaction Qualifier

Table 1 – Summary of Possible Card / Reader Interactions

Reader Configuration \ Contactless Card Capability	MSD and qVSDC	MSD, qVSDC, and VSDC
MSD and qVSDC	qVSDC	qVSDC
qVSDC only	qVSDC	qVSDC
qVSDC and VSDC	qVSDC	VSDC
MSD, qVSDC, and VSDC	qVSDC	VSDC
MSD and VSDC	MSD	VSDC
MSD	MSD	MSD

Table 3 – Terminal Transaction Qualifiers (Tag '9F66')

Byte	Bit	Definition
1	8	'1' – Contactless magnetic stripe (MSD) supported '0' – Contactless magnetic stripe (MSD) not supported
	7	'1' – Contactless VSDC supported '0' – Contactless VSDC not supported
	6	'1' – Contactless qVSDC supported '0' – Contactless qVSDC not supported
	5	'1' – Contact VSDC supported '0' – Contact VSDC not supported
	4	'1' – Reader is Offline Only '0' – Reader is Online Capable
	3	'1' – Online PIN supported '0' – Online PIN not supported
	2	'1' – Signature supported '0' – Signature not supported
	1	RFU – b'x'
2	8	'1' – Online cryptogram required '0' – Online cryptogram not required
	7	'1' – CVM required '0' – CVM not required
	6-1	RFU – b'xxxxxx'
3	8-1	RFU – b'xxxxxxxx'
4	8-1	RFU – b'xxxxxxxx'

VISA Authentication flow dCVV mode

Terminal

Contactless Card

9f66: Terminal Transaction Qualifiers (TTQ)
9f02: Amount, Authorised (Numeric)
9f37: Unpredictable Number
5f2a: Transaction Currency Code

5. Get Processing Options(PDOL)

6. AIP, AFL

```
57: Track 2 Equivalent Data (19 bytes): 48 62 70 00 57 22 49 45 d1 51 02 21
00 00 00 37 70 90 0f
5f20: Cardholder Name (2 bytes):  /
9f1f: Track 1 Discretionary Data (17 bytes): 00000000709000000
```

7. Read Records

Track Datas

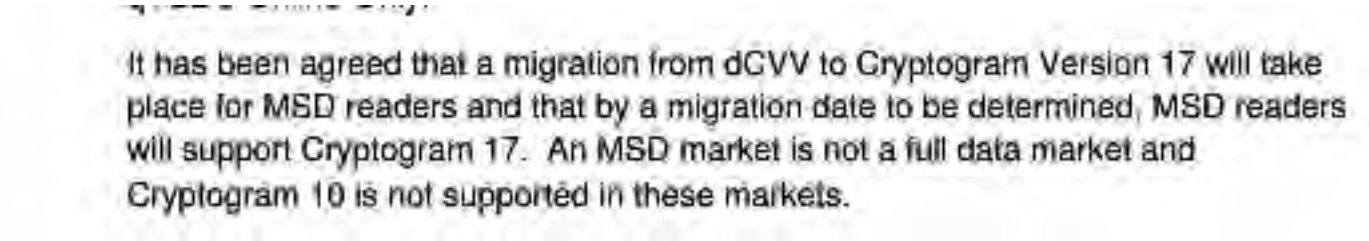
Generate track 1 and Track 2 datas

So lets clone a VISA card in dCVV

- We have to use dCVV mode.
- All other modes appear to use long random numbers.
- dCVV is a legacy mode
- So limits amount of devices in the field.
- dCVV sucks. No random number, not use of transaction specific data.
- Plus its optional – issuers can set static CVVs if they want here!

How to prevent cloning (VISA)

- Issuers can disable dCVV Support – my ING debit card doesn't support this.
- Processors must not support dCVV mode
- Terminals must use strong cryptographic secure random number generation.



It has been agreed that a migration from dCVV to Cryptogram Version 17 will take place for MSD readers and that by a migration date to be determined, MSD readers will support Cryptogram 17. An MSD market is not a full data market and Cryptogram 10 is not supported in these markets.

Cloning a Mastercard!

Hey – I'm an equal opportunity troll!

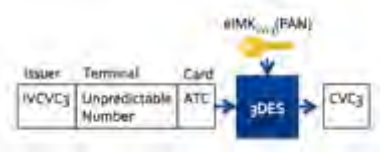
What does Mastercard Support?

- Magstripe Mode
  - – generate dynamic CVV code and insert into track data
- MChip Mode
  - EMV – validate records and calculate application cryptogram.

NFC Payments (MAGSTRIPE modes only)

Contactless Card

Terminal



Generate UN
(hopefully random…)

7. Compute Cryptographic Checksum

CVCs, ATC

```
77: Response Message Template Format 2 (15 bytes):
9f61: CVC3 Track 2 (2 bytes): 46 2f
9f60: CVC3 Track 1 (2 bytes): 7e 3f
9f36: Application Transaction Counter (2 bytes): 02 44
```

9. Read Records

10. Track Datas

Generate track 1 and
Track 2 datas

Great – lets clone a card.

1. Criminal scans victims card and application performs $10^{length(UN)}$ transactions

2. Criminal takes app to a retailer and buy stuff

3. Terminal send UN to "card"

4. App looks up the supplied UN, responds with correct CVC and ATC codes

5. Criminal runs off with loot.

How we determine the required length of the UN?

Length of UN = NumBitsSet(KtrackX) – TtrackX

However CVV/CVC has to be formatted to fit in Track Data.
So we format it as Binary Coded Decimal.
And issuer limits length of UN to suit their systems
Take a UN of 4 bytes:

- 4 bytes binary = 2^{32} values = 4,294,967,296
- 4 bytes BCD = 10^{8} values = 100,000,000
- 2 numbers(1 byte BCD) = 10^{2} values = 100

Solutions!

- Issuers need to program cards to support long UN numbers (my GE cards have this)
- Processors need to monitor ATC codes – if you jump ATCs by a bit you should flag it as fraud.
- Terminal Manufacturers need to use proper cryptographically strong RNGs
- Monitor "Generate Cryptographic Checksum" commands.
- Request 2nd factor when ATCs jump – I get this when I've tested in Australia
- Wrap yourself in tin foil?

So I made an app to test your own cards

- https://github.com/peterfillmore/Check-Paypass-Random-Number
- Scan your card and find the time it takes to clone!
- Doesn't bork your card – i.e no transactions take place – just a reads the appropriate record.

Tools

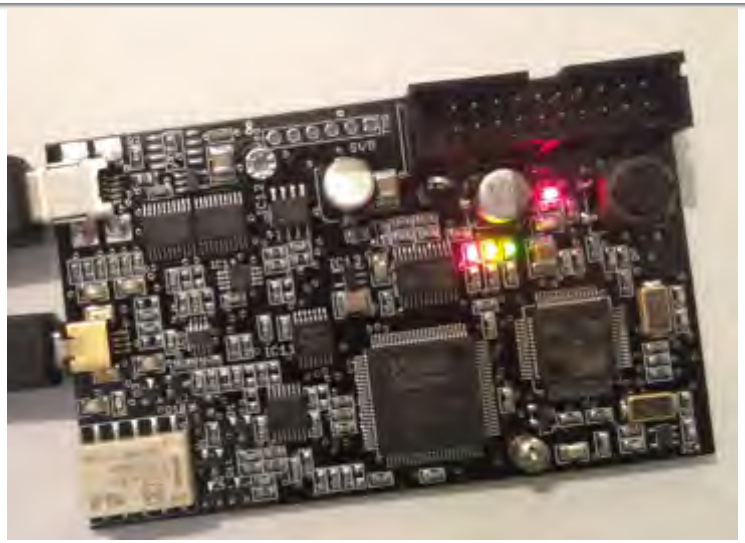
ACR-122U, PCSCd, PyScard and RFIDiot

- ~AUD$50 on ebay
- https://pypi.python.org/pypi/pyscard
- Adam Laurie's RFIDiot: https://github.com/AdamLaurie/RFIDIOt
- My fork used in this talk: https://github.com/peterfillmore/RFIDIOt/tree/preplayattack
- Crappy driver – PCSCd pegs core at 100% - frequent resets needed.
- See Charlie Millers talk Don't Stand So Close To Me: An Analysis of the NFC Attack Surface ': https://www.youtube.com/watch?v=qRFpjLIhoXo

Android phone with HCE support

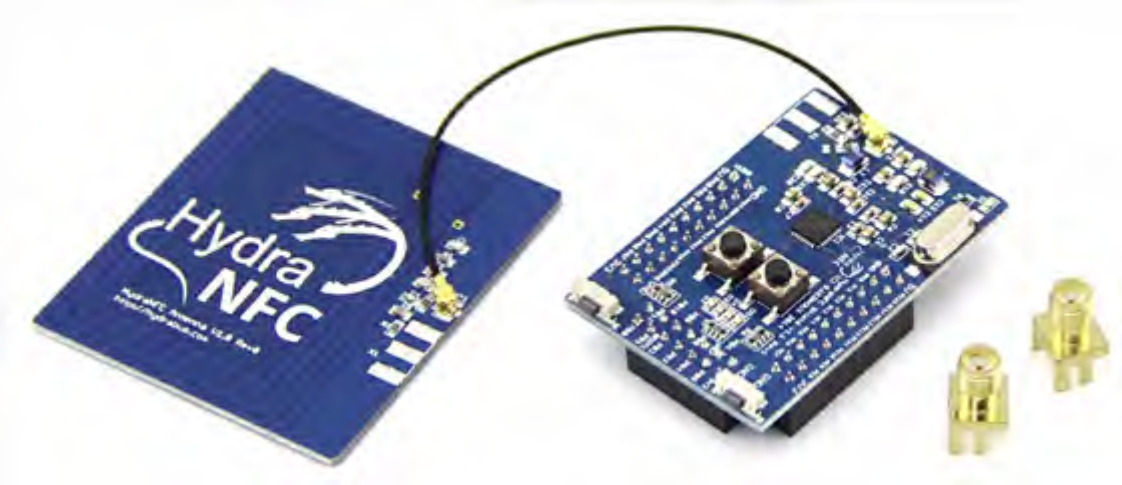
- I love HCE – Host Card Emulation – added Android 4.4 (KitKat)
- Best tool I've found – takes care of a lot of pain.
- Can't emulate low level stuff (i.e UID, card parameters)
- 1 stop shop to cloning a card!
- Great tool for in field testing – no one questions a phone in your hand.
- https://github.com/peterfillmore/Check-Paypass-Random-Number
- Not releasing my cloner tool – that's a can of worms I'm not going to open.

Proxmark3



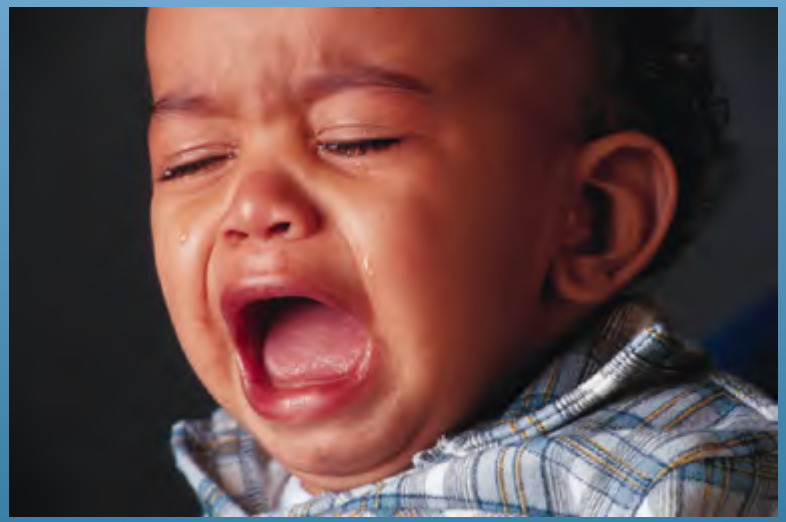
- Lots of improvements in the last few months.
- You can control everything!
- EMV support added in my git branch: https://github.com/peterfillmore/proxmark3

HydraNFC+HydraBus



- Benjamin Vernoux's Project.
- http://hydrabus.com/
- In between a Proxmark3 and an Android phone.
- Hopefully will provide a good fuzzing platform (i.e. hardware control and easy software).

My Failures



Failures / how not to do hardware exploitation

- Thought I had a buffer overflow when fuzzing – actually just the watchdog timer resetting ☹
- Lots of pain getting jtag sorted. Make sure you disable watchdog timer!
- Even after getting jtag – can't get it to boot when connected – I think its related to the timers.
- Tried to QEMU it – got it running but stopped at trying to emulate all the hardware.

JTAG! OpenOCD! Still Failed!

Steps:

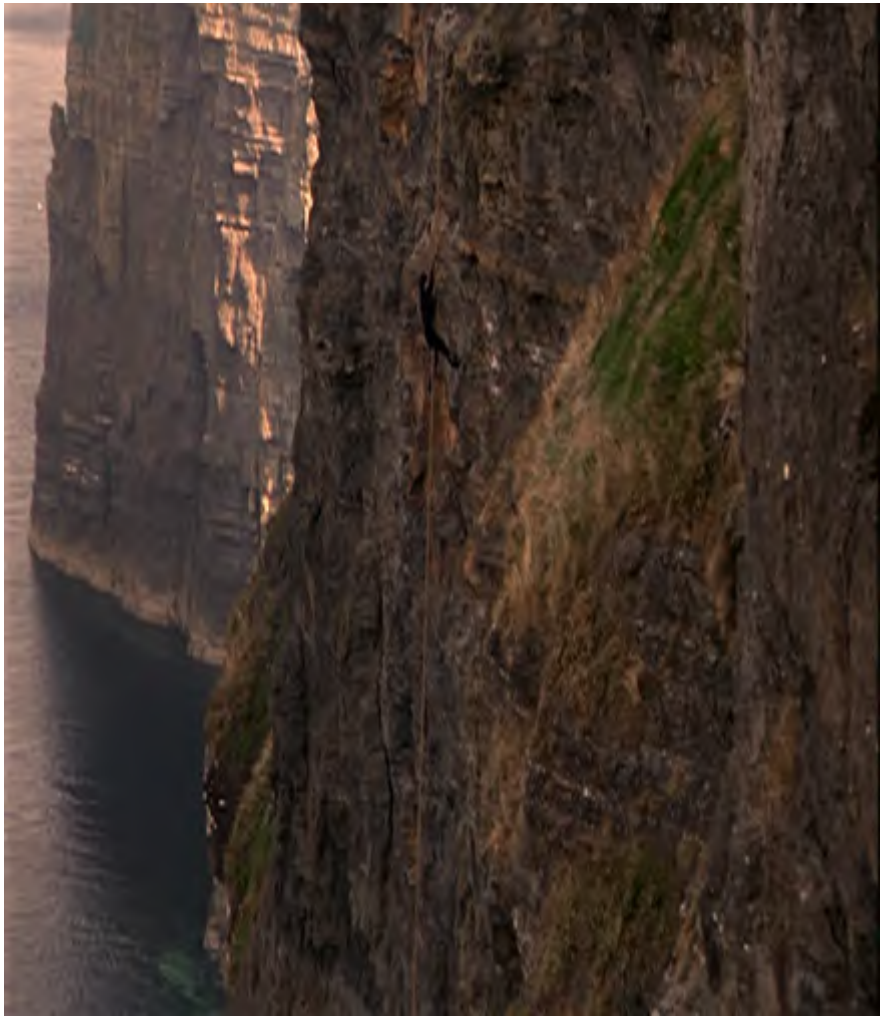
1. Look for unpopulated pads
2. Download data sheet
3. Trace pads to chip
4. Patch firmware to enable JTAG and disable watchdog timers
5. OpenOCD and connect to GDB

```
seg000:00003D6C      setupWatchdogTimers
seg000:00003D6C
seg000:00003D6C 000                  LDR     R2, =0x2F9D05D
seg000:00003D6E 000                  LDR     R3, =WDTC
seg000:00003D70 000                  STR     R2, [R3]
seg000:00003D72 000                  MOVS    R3, WDMOD
seg000:00003D76 000                  MOVS    R2, #0
seg000:00003D78 000                  STRB    R2, [R3]
seg000:00003D7A
seg000:00003D7A      initWDFEED
seg000:00003D7A 000                  LDR     R2, =WDFEED
seg000:00003D7C 000                  MOVS    R3, #0xAA
seg000:00003D7E 000                  NEGS    R3, R3
seg000:00003D80 000                  STRB    R3, [R2]
seg000:00003D82
seg000:00003D82      reloadWatchdog
seg000:00003D82 000                  MOVS    R3, #0x55 ; 'U'
seg000:00003D84 000                  STRB    R3, [R2]
seg000:00003D86 000                  BX      LR
seg000:00003D86      ; End of function setupWatchdogTimers
```

Failures continued (IDA Pro)

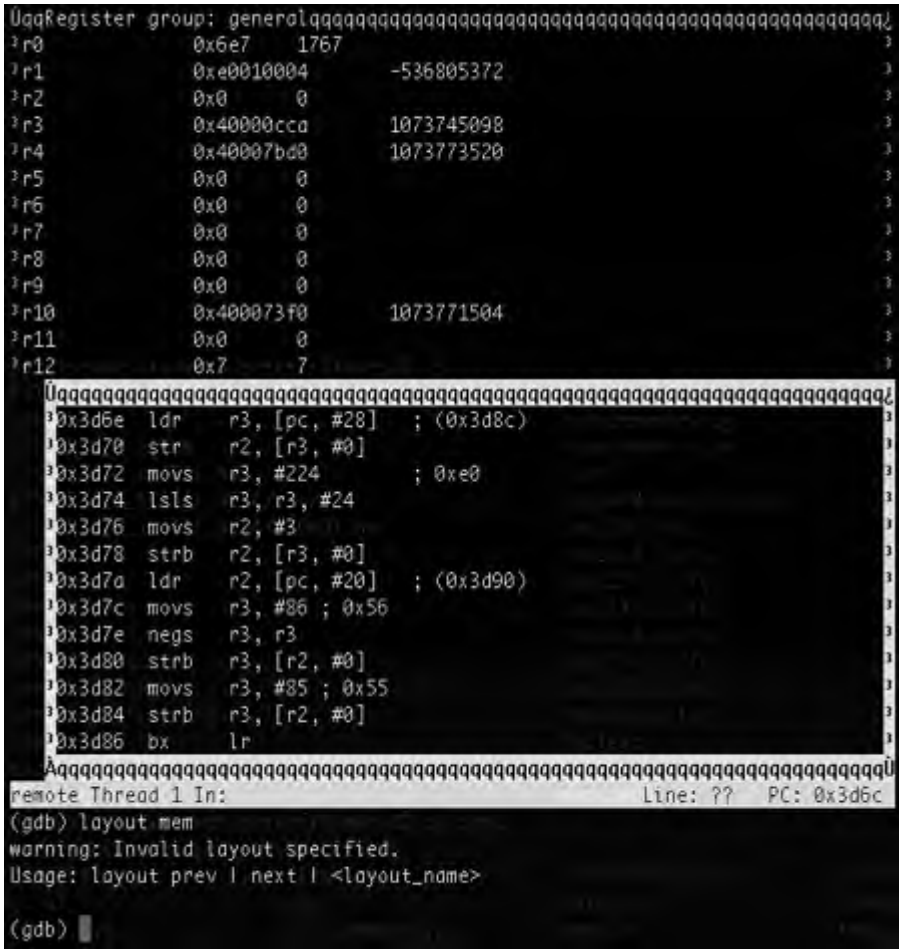
- Threw the firmware update binary into IDA pro
- No identified signatures ☹
- Tried installing
- Thumb and ARM code mixed is fun!
- Worked out lots of basic system functions.
- But still a bit of a mystery as to many key functions

Here's some IDA graphs



Qemu To the Rescue?

- Semi-success.
- Had to customise the platform.
- Injected code to provide Bootloader functions
- Makes reversing a lot easier!



EMV Prevents Cloning right?

- Yes (however…)
- As long as you do everything right.
- And everyone in the payment flow supports it.
- And the reader is not comprimised
- And your card doesn't support MSD mode(VISA)
  - Or your card implements a long UN (Mastercard)
- And the issuer is checking ATCs for large jumps (Mastercard)
- And your issuer programmed the card right.
- And the NSA/FSB/PETA haven't copied your master crypto keys (gemalto I'm looking at you)

Conclusions

- Yes – you can clone Contactless Payment Cards
- Or more correctly Contactless Payment Transactions
- Requires legacy modes to be supported by hardware and processors
- Requires issuers to set low random number lengths(Mastercard).
- Requires old hardware and systems that don't support EMV messaging (Visa)
- Above all depends on proper random numbers being generated by terminals!

Conclusions 2

- Able to be performed in practice.
- Fraud detection doesn't seem to occur
- Contactless interfaces don't get the love they need in testing.
- EMV relies on complex processing – is highly susceptable to security issues creeping in.
- EMV testing does not perform security testing on kernel code.
- Somebody please let me sniff an Apple Pay transaction!

Check my github for stuff

- Check your Paypass Random Number
- https://github.com/peterfillmore/Check-Paypass-Random-Number
- Forked proxmark to do EMV stuff
- https://github.com/peterfillmore/proxmark3
- RFIDiot with additions used in this talk
- https://github.com/peterfillmore/RFIDIOt/tree/preplayattack
- Slides:
- XXXTBAXXX

References – aka who have I plagerized off

- "Don't Stand So Close To Me, An analysis of the NFC attack surface" – Charlie Miller 2012
- "PinPadPwn" – Nils & Rafael Dominguez Vega Pin Pads, 2012
- "Credit Card Fraud - The Contactless Generation" Kristian Paget, 2012
- "Mission Mpossible" –Nils and Jon Butler 2013
- "Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless" - Michael Roland 2013
- Standards - http://www.emvco.com
- Utilities - http://www.emvlab.org/tlvutils/
- http://www.cl.cam.ac.uk/research/security/

Thanks

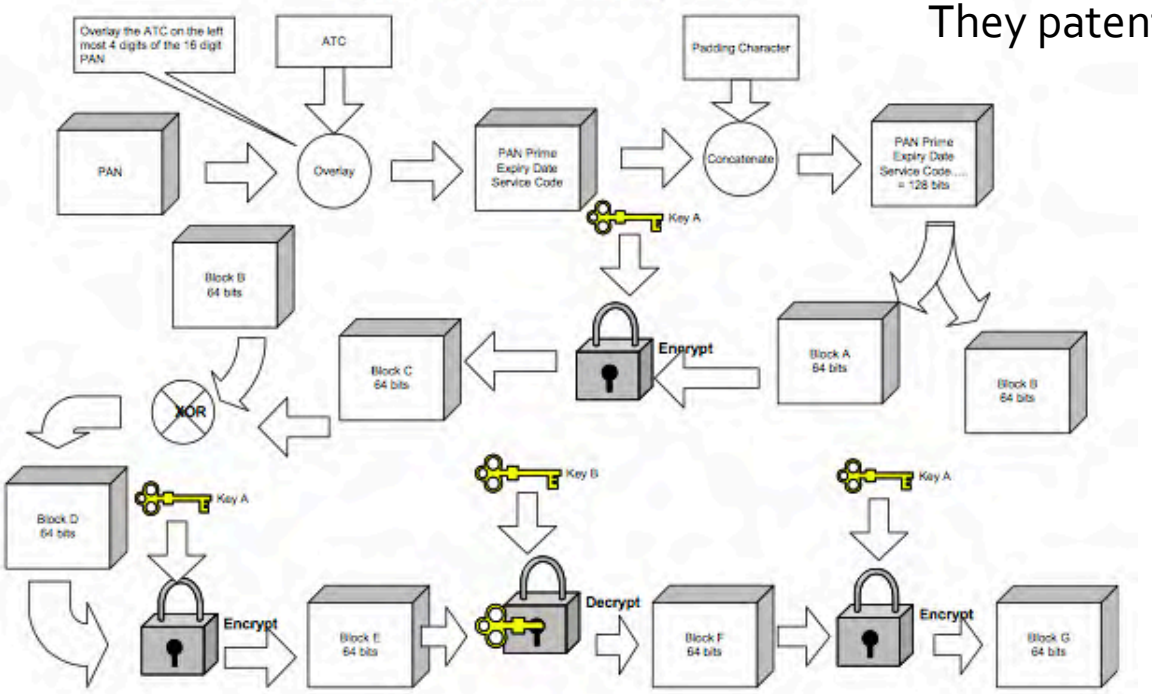
- Pwpiwi@proxmark3 – Putting up with my complaining and fixing the Proxmark3 code
- Peter Maydell@linaro – Not insulting me for asking a stupid question on the qemu mailing list.
- Dodgy Chinese Document share sites – for allowing me to skip signing NDAs
- Benjamin Vernoux@hydrabus – buy it!

So lets clone a VISA card in dCVV mode (DEMO)

- Its rather easy – too easy in fact.
- Just have to read the records off the card
- If the card supports dCVV, then it'll copy that into the track data.
- If the card supports only static CVV, then it'll copy it into place.
- If the card doesn't support this method, then it'll return an error code (ING debit – 6984)

Why dCVV sucks

Figure 5 – Dynamic CVV Algorithm



They patented this crap?!?

It has been agreed that a migration from dCVV to Cryptogram Version 17 will take place for MSD readers and that by a migration date to be determined, MSD readers will support Cryptogram 17. An MSD market is not a full data market and Cryptogram 10 is not supported in these markets.

VISA Authentication flow (CVN17 mode)

Terminal

Contactless Card

9f66: Terminal Transaction Qualifiers (TTQ)
9f02: Amount, Authorised (Numeric)
9f37: Unpredictable Number
5f2a: Transaction Currency Code

5. Get Processing Options(PDOL)

6. AIP, AFL, Cryptogram

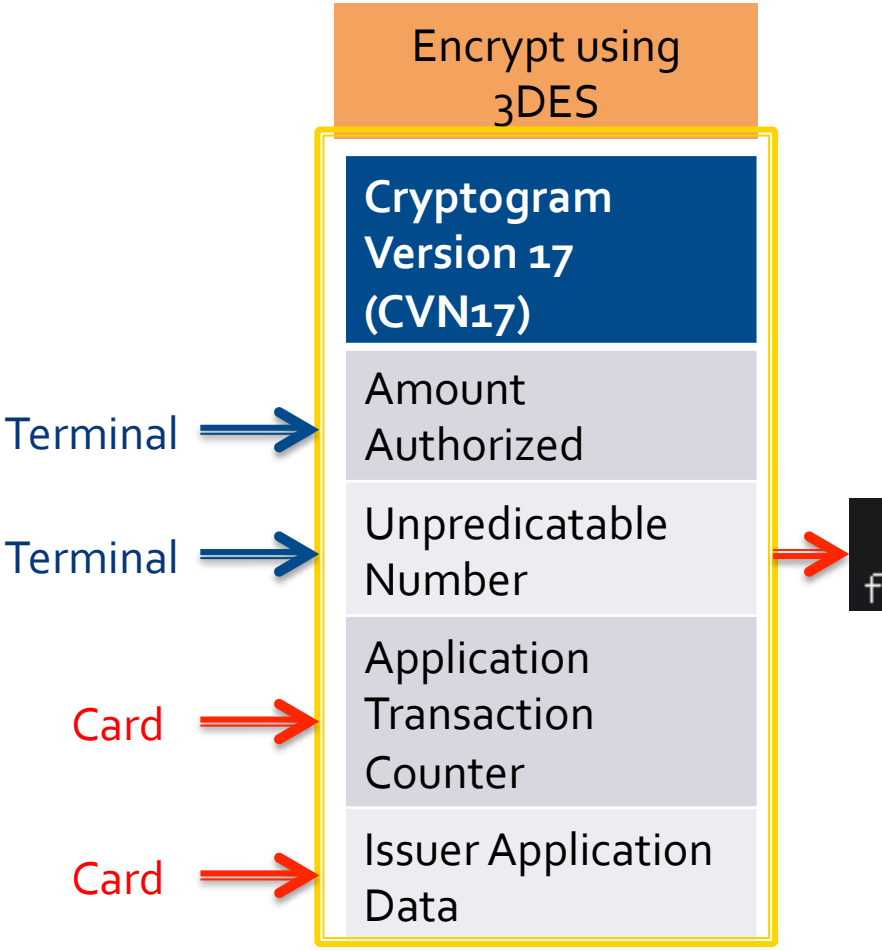
```
9f10: Issuer Application Data (7 bytes): 06 0c 1
57: Track 2 Equivalent Data (19 bytes): 48 62 70
9f1f: Track 1 Discretionary Data (17 bytes): 000
5f34: Application Primary Account Number (PAN) 5
82: Application Interchange Profile (2 bytes): 0
9f36: Application Transaction Counter (2 bytes):
9f26: Application Cryptogram (8 bytes): ea 41 92
```

7. Read Records

Track Datas

Generate track 1 and Track 2 datas

CVN17 Generation (how we prevent cloning in VISA cards)




```
  9f26: Application Cryptogram (8 bytes):
f1 f0 79 4e 46 d5 18 10
```

Tools I used in this research

- Proxmark3 – lots fixed since I started doing this stuff. Can finally emulate ISO14443a tags properly
- My fork with EMV stuff – http://github.com/peterfillmore/proxmark3
- RFIDiot + ACR122U – good for learning basics quickly. Beware of the 64 byte limit when trying to emulate tags with proxmark3
- Fork with EMV stuff :
- https://github.com/peterfillmore/RFIDIOt/tree/preplayattack