NFCulT

An easy and nice tool that will make
you have fun, or... make profit!

Who I am

- **Matteo Beccaro (aka bughardy)**
  - Student at Politecnico di Torino – Computer Engineering
  - Security Engineer and researcher at Secure Network S.r.l.
  - Contacts:
    - bughardy@cryptolab.net
    - Twitter: **@_bughardy_**
    - Blog: https://bughardy.me

- Speaker at:
  - **DefCon 21** - «OPT it won't save you from free rides»
  - **30C3** - «How to build a safe NFC ticketing system»
  - **BlackHat** US 14's Arsenal
  - **DefCon 22** SkyTalks
  - **BlackHat EU 14's Arsenal**
  - And now here at **TetCon 2015**

Who I am

- **Technical Research Leader** at **OPFOR,** a Physical Security Dep. of Secure Network with focus on:
  - Electronic Access Gate
  - Ticketing System Security
  - Physical Penetration Test
  - Device Vulnerability Research

- **Member of PhySec OPFOR's RedTeam**

What we are dealing with

- **MIFARE ULTRALIGHT** tags as tickets
  - Designed to work @13.56 MHz
  - Manufactured by NXP semiconductors
  - Worldwide largest installed platform
    - More than 50 million MIFARE reader core components
    - More than 5 billion ICs

Image source: Google® images

Use of NFC on transportation systems

- Ticketing systems that work at very high speeds to reduce queues at tube and train stations
- Usually, absolutely not safe

  – **Why?**

Image source: Google® images

Memory Layout

- **512 bits of memory** (64 bytes) arranged in 16 *pages* of 4 bytes

- **Different** memory **areas:**
  - UID and Internal Byte
  - The Lock Bytes
  - OTP Area
  - Data Area

Page Address		Byte #			
DEC	*HEX*	*0*	*1*	*2*	*3*
0	*0x00*	*UID*			
1	*0x01*	*UID*			
2	*0x02*	*UID*	*Internal*	*Lock Byte*	*Lock Byte*
3	*0x03*	*OTP*			
From 4 to 15	*0x04 to 0x0F*	*Data*			

UID and Internal Bytes

- 7 bytes for ***Serial Number***
  - SN0 to SN6
- 1 byte for ***Internal***
- 2 ***Check bytes*** as a result of XOR operations
  - CB0 and CB1

- All of them are programmed by the manufacturer: **read only**

Page Address		Byte #			
DEC	*HEX*	*0*	*1*	*2*	*3*
0	*0x00*	*SN0*	*SN1*	*SN2*	*CB0*
1	*0x01*	*SN3*	*SN4*	*SN5*	*SN6*
2	*0x02*	*CB1*	*Internal*	*Lock Byte*	*Lock Byte*
3	*0x03*	*OTP*			
From 4 to 15	*0x04 to 0x0F*	*Data*			

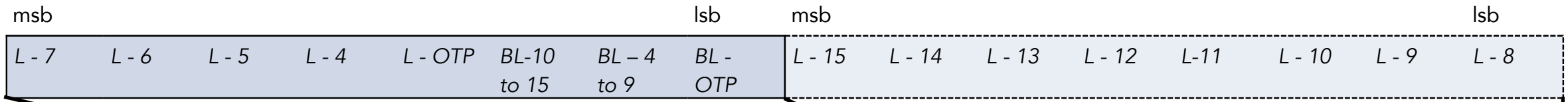
UID and Internal Bytes

- How CB0 and CB1 are evaluated?
  - CB0 = 0x88 ⊕ SN0 ⊕ SN1 ⊕ SN2
  - CB1 = SN3 ⊕ SN4 ⊕ SN5 ⊕ SN6

Page Address		Byte #			
DEC	*HEX*	*0*	*1*	*2*	*3*
0	*0x00*	*SN0*	*SN1*	*SN2*	*CB0*
1	*0x01*	*SN3*	*SN4*	*SN5*	*SN6*
2	*0x02*	*CB1*	*Internal*	*Lock Byte*	*Lock Byte*
3	*0x03*	*OTP*			
From 4 to 15	*0x04 to 0x0F*	*Data*			

Lock Bytes

- 2 bytes



Page Address		Byte #			
DEC	HEX	0	1	2	3
0	0x00	UID			
1	0x01	UID			
2	0x02	UID	Internal	Lock Byte	Lock Byte
3	0x03	OTP			
From 4 to 15	0x04 to 0x0F	Data			

- Possibility of making a whole page (4 bytes) read-only
- Possibility of making the Lock Bytes themselves read-only

Lock Bytes

- They cannot be edited as you want
  - What you are about to write is simply bitwise *ORed* with the present data

OR	0	1
0	*0*	*1*
1	*1*	*1*

- One bit in state **1** cannot be turned into **0** anymore

Page Address		Byte #			
DEC	*HEX*	*0*	*1*	*2*	*3*
0	*0x00*	*UID*			
1	*0x01*	*UID*			
2	*0x02*	*UID*	*Internal*	*Lock Byte*	*Lock Byte*
3	*0x03*	*OTP*			
From 4 to 15	*0x04 to 0x0F*	*Data*			

OTP Bytes

- The **only** security function in MIFARE ULTRALIGHT tags
- 4 bytes, 0x00 by default
- As the Lock Bytes, what you are about to write is *ORed* with the present data
- *One-Time Programmable*
  - **Not** *One-Time Password* !

Page Address		Byte #			
DEC	*HEX*	*0*	*1*	*2*	*3*
0	*0x00*	*UID*			
1	*0x01*	*UID*			
2	*0x02*	*UID*	*Internal*	*Lock Byte*	*Lock Byte*
3	*0x03*	*OTP*			
From 4 to 15	*0x04 to 0x0F*	*Data*			

Data Sector

- Largest sector
  - 48 bytes, arranged in 12 pages
- Read / Write mode
- For transportation system applications this sector will store
  - Time of last stamp
  - Validator Machine ID
  - Bus line or underground stop

Page Address		Byte #			
DEC	*HEX*	*0*	*1*	*2*	*3*
0	*0x00*	*UID*			
1	*0x01*	*UID*			
2	*0x02*	*UID*	*Internal*	*Lock Byte*	*Lock Byte*
3	*0x03*	*OTP*			
From 4 to 15	*0x04 to 0x0F*	*Data*			

Pros & Cons

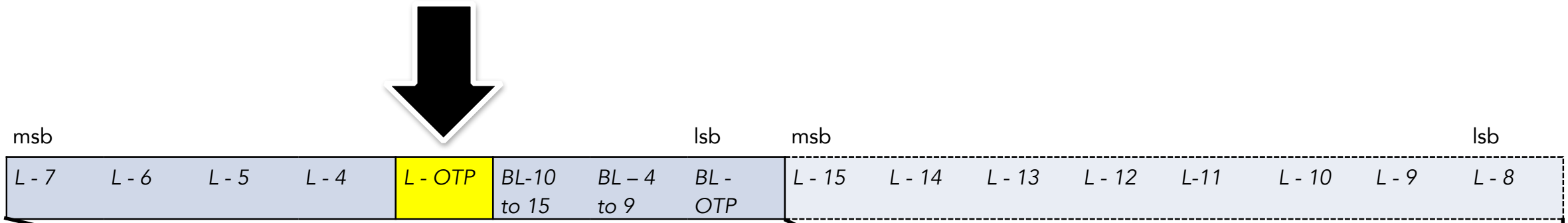
- **Pros**
  - Cheap
  - Possibility of creating limited tickets
    - Expiration after a finite number of times
    - Good for public transportation systems

- **Cons**
  - **No hardware encryption**
  - Usually not well implemented on public transportation systems:
    - Reset Attack [2012]
    - Lock Attack [2013]
    - Time Attack [2013]
    - Replay Attack [2014]

How easily is to exploit the related vulnerabilities?

Here comes NFCulT !

Attack 1 - Lock Attack

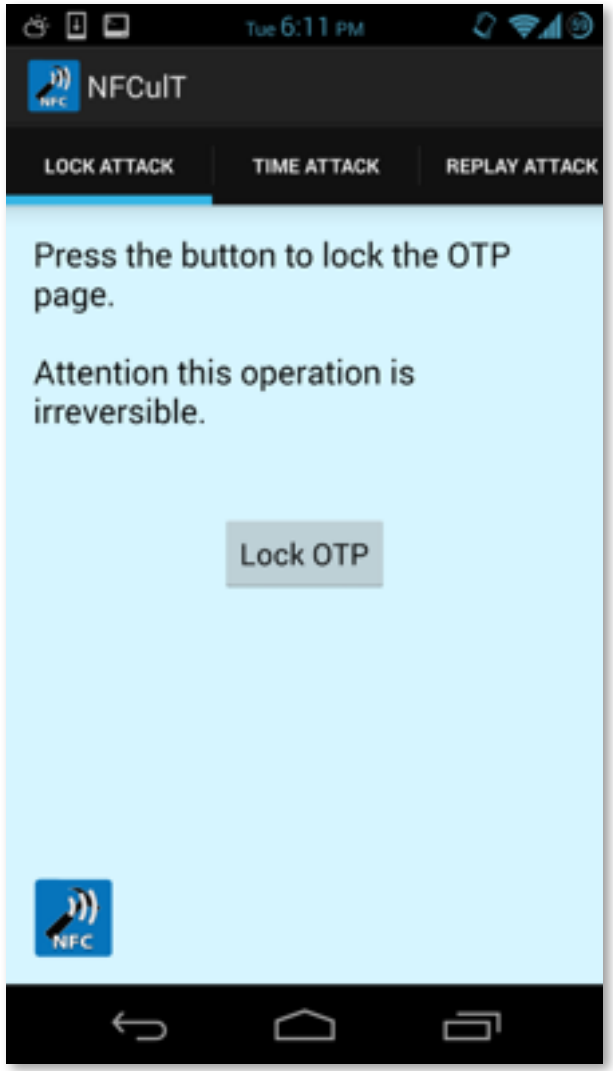


- **Lock Attack**
  - Mostly overcome
- **Extremely easy task to perform**
  - Step 1: set the correct lock bit
  - Step 2: have fun!

Page Address		Byte #			
DEC	HEX	0	1	2	3
0	0x00	UID			
1	0x01	UID			
2	0x02	UID	Internal	Lock Byte	Lock Byte
3	0x03	OTP			
From 4 to 15	0x04 to 0x0F	Data			

Attack 1 - Lock Attack

- Acting on the lock bits
- We can make a sector of the ticket read-only
  - Just turn a bit from 0 into 1 (L-OTP)
- Lock attack with our app
  - **just press a button!**
- The stamping machine tries to validate the ticket
  - No feedback for the (failed) write operation
- The number of rides is freezed!



Attack 1 - Lock Attack

- Check if last time stamp is older than *n* minutes:
  - No → the ticket is still valid
  - Yes → let's stamp the ticket
    - Check if there are rides left:
      - No → the ticket is useless
      - Yes → let's stamp it
        - » Write Timestamp → OK
        - » Write other stuff → OK
        - » Write the new number of rides left → **KO**
- No feedback → WIN
- Do not forget to take one ride off !
- Easily fixable
  - Just check the state of L-OTP bit and decide whether to validate the ticket or not

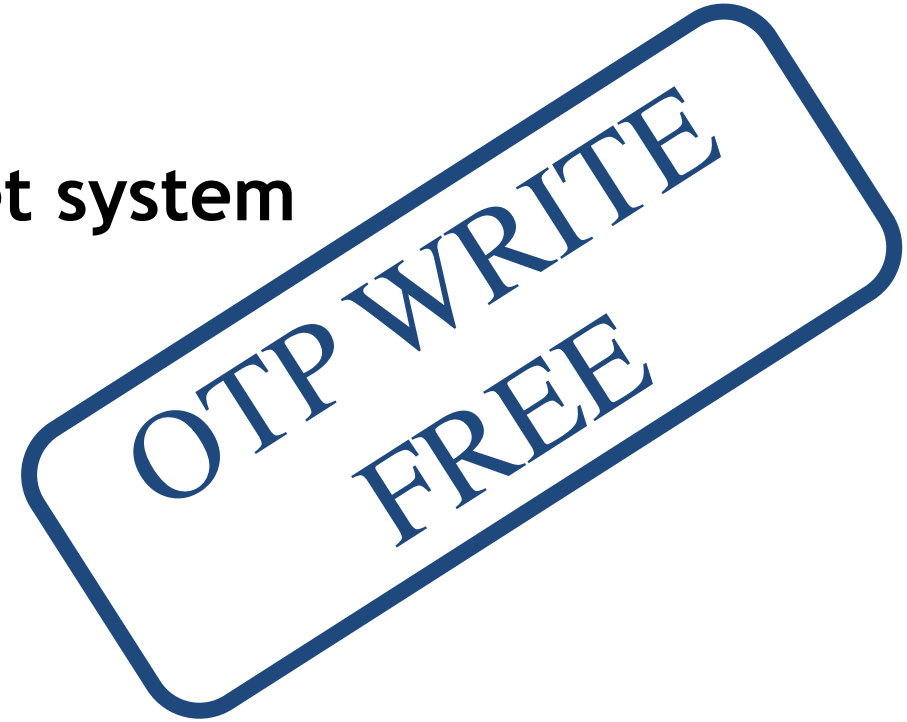
Type of multiple ride ticket: 5 rides ticket

Ticket is valid. 5 Rides left

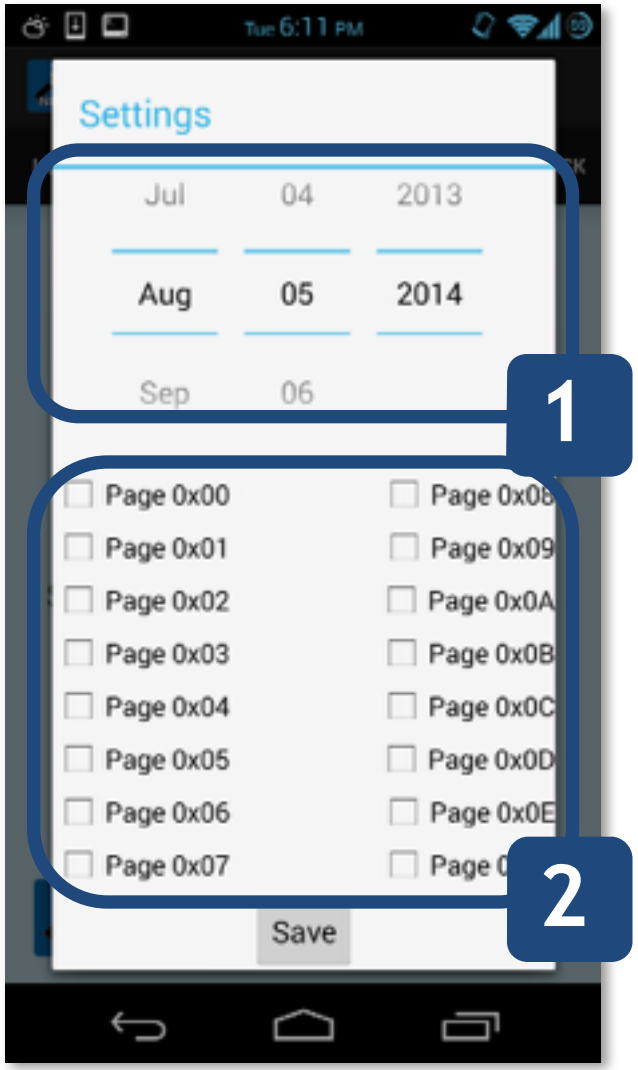
Attack 2 - Time Attack

- **Time Attack**
  - Attacker is required to understand the *format* of the timestamp
  - The *location* where the timestamp is stored must be identified as well
  - The timestamp *is not encrypted*

- **Four (4) simple steps to own the ticket system**
  - Set the right page
  - Set the starting timestamp's date
  - Set your desired timestamp's time
  - Have fun with your band new ticket!



Attack 2 - Time Attack



1. Set the date of the timestamp

1. Choose the right data page

3. Setting the desired time of timestamp

4. Write data to the ticket

Attack 3 – Reply Attack

- **Replay Attack**
  - Assumptions
    - No online database of tickets are available
      - Attacker needs a valid source ticket
    - The attacker owns a *clone* MIFARE Ultralight tag

- **Three (3) simple steps**
  - Dump the ticket
  - Edit key information
  - Write to the blank UL tag

Attack 3 – Reply Attack

- We love Chinese stuff!
  - **Perfect clone** of UL ticket
  - No read-only pages

- **Just clone your real ticket!**
  - Add a ride to the cloned ticket
  - Stamp your brand new cloned ticket
  - Clone it back on the real ticket
    - Or directly use the clone
    - Pay attention to the number of rides

  - Have fun!

Attack 3 – Reply Attack

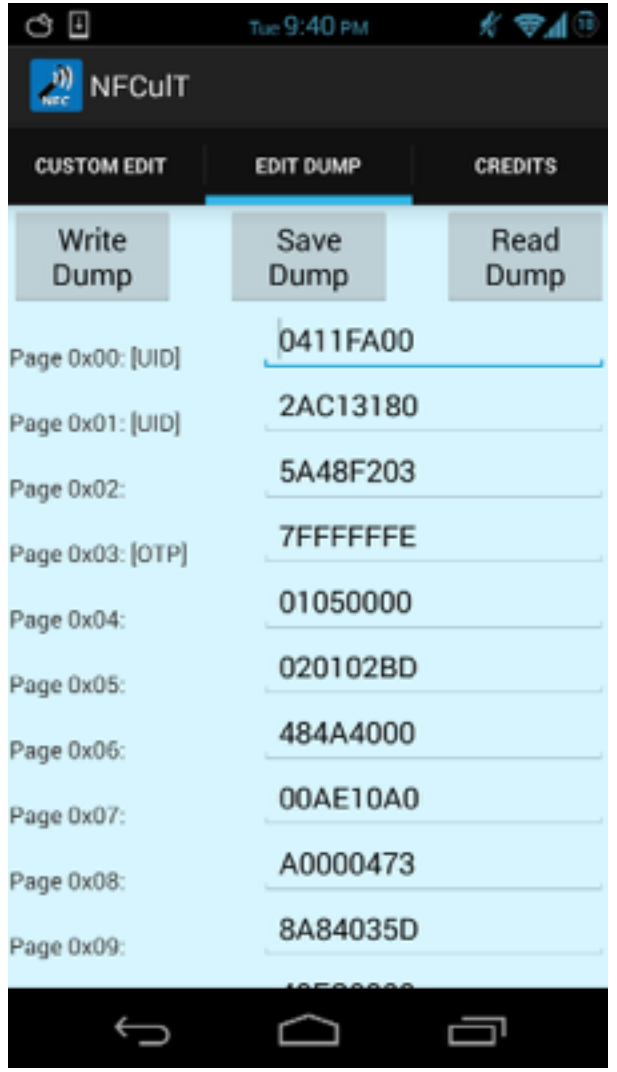
- **Considerations**
  - Encrypted ticket
    - Encryption can be circumnavigated by cloning the whole encrypted data
      - It works if the whole data area is encrypted
      - It works if the timestamp is encrypted
    - It works because
      - The stamping machine is validating a cloned ticket, but with a ride more
      - The clone tag has the same number or rides left, but an updated timestamp

What if you could simply edit your tag as you prefer?

Custom Edit

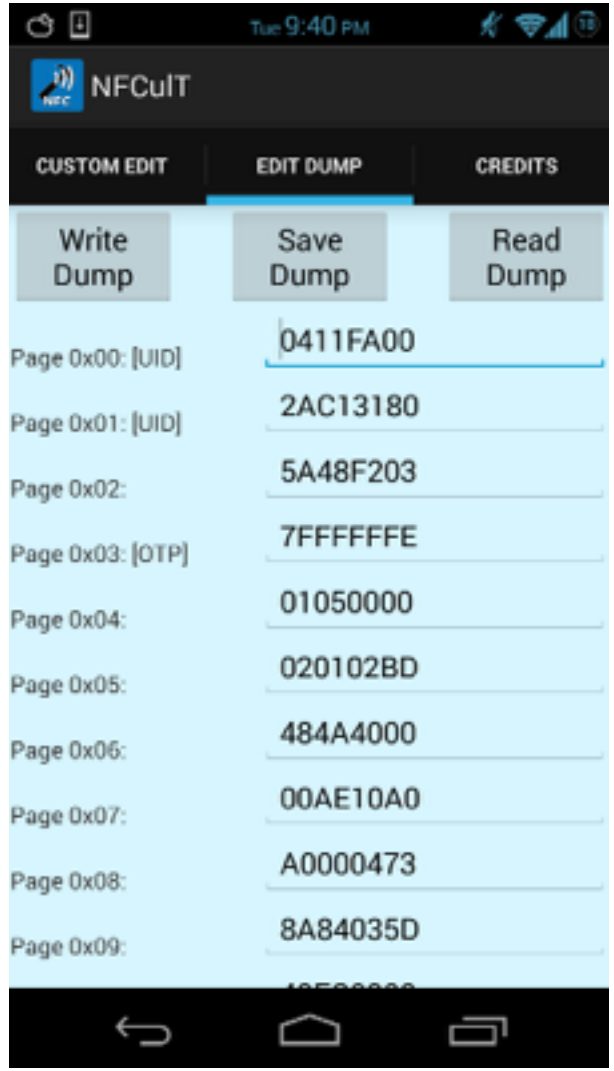
- Just write whatever you want wherever you prefer:
  - Fix UID bytes (if you broke them)
  - Write arbitrary data to pages

- **Be careful!**
  - If you are not using a clone UL, you could lock forever some pages



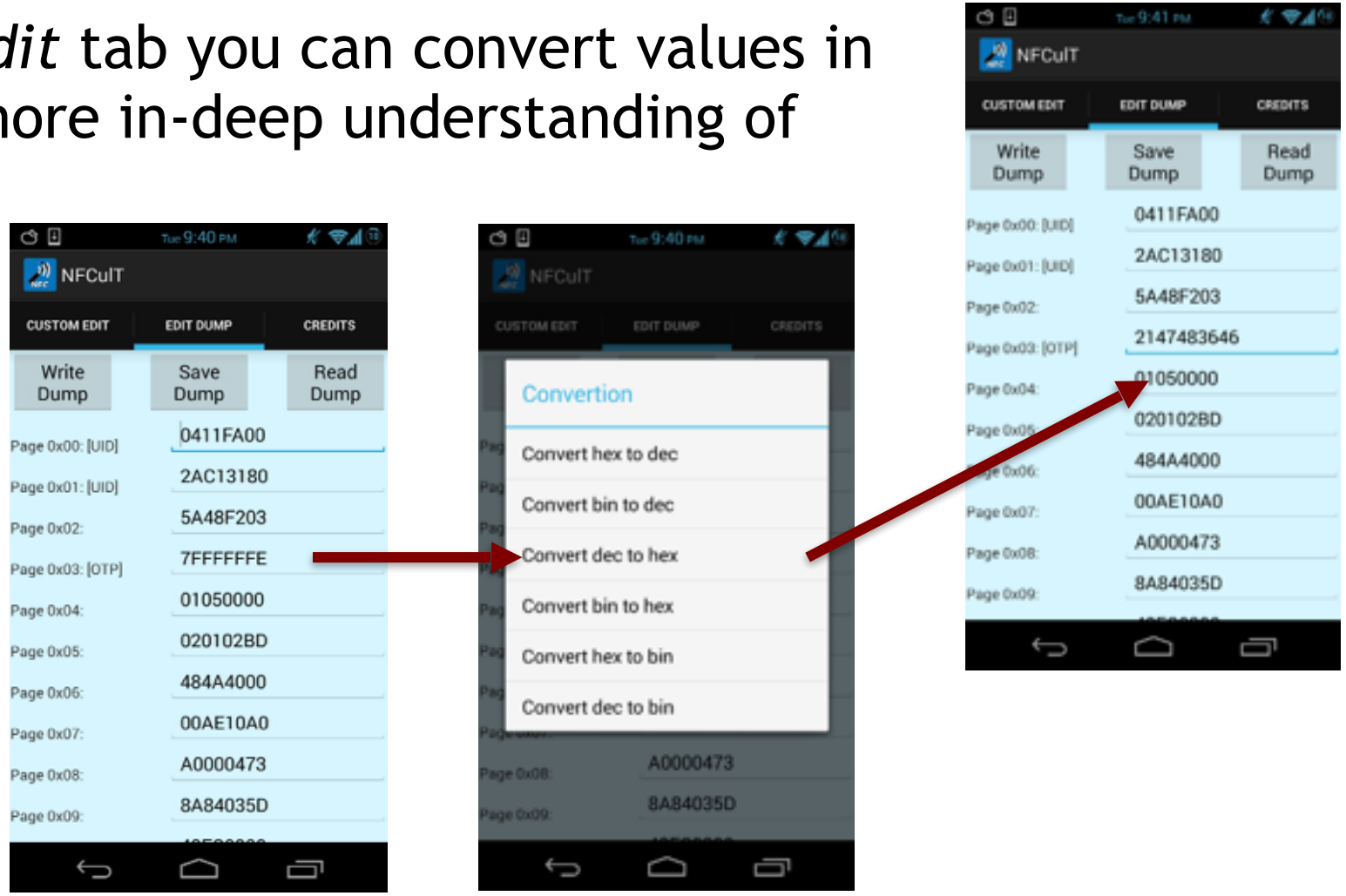
Dump Edit

- Now you can edit your saved dump:
  - For **privacy** and **security** reasons dumps are stored in private directory of the app located in **/data/data/org.bughardy.nfcult/files/**
  - **Without root** permission an external app can not access to those files
  - Now you can edit them within the NFCulT app
- **Why it is useful?**
  - This is a must-have features for reply attack, since very often you need to add a ride to your dump
  - You can **reply** the edited dump **directly** to your ticket or save it as a new one



Base Conversion

- Now from the *Dump Edit* tab you can convert values in different bases for a more in-deep understanding of data on the ticket.
- **It supports:**
  - Hex —> Bin
  - Hex —> Dec
  - Bin —> Hex
  - Bin —> Dec
  - Dec —> Hex
  - Dec —> Bin



ART Ready

- Now NFCulT supports also **ART**
- That means it **is ready to use** also on **Android 5.0** smartphones
- In the next release I'll probably work on a tablet version

Where to find

- Source code and dev build are on github:
  - https://github.com/securenetwork/nfcult

- APK and official signed builds are available on Play Store:
  - https://play.google.com/store/apps/details?id=org.bughardy.nfcult