Breaking the (FAQin) badge

Rodolfo García Peñas (kix)

kix@kix.es | @thekix

October 2019

Outline

Outline

Introduction

- This badge was the present for the attendees to the last **FAQin Congress** (2019)
- The badge is an NFC card (tip: see the waves in the right area ;-))
- We need read the card contents!! -> HW and SW tools

Outline

Reading the NFC Card

ACR122U

Proxmark

Readers

- You can use some NFC readers.
- I used ACR122U and Proxmark ('like') devices

Identify the NFC Card

nfc-list

- Using nfc-list we can read the NFC Card info
- Card is an ISO 14443A, standard for RFID, 13.56 MHz
- This card is a MiFare Classic (MFC) Card

```
kix@seahorse:~/src/nfc/faqin/script$ nfc-list -v
nfc-list uses libnfc 1.7.1
NFC device: ACS / ACR122U PICC Interface opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
* UID size: single
* bit frame anticollision supported
       UID (NFCID1): 3a  92  ed  1a
      SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092

Fingerprinting based on MIFARE type Identification Procedure:
* MIFARE Classic 1K
* MIFARE Plus (4 Byte UID or 4 Byte RID) 2K, Security level 1
* SmartMX with MIFARE 1K emulation
```

Identify the NFC Card

nfc-list

- Using nfc-list we can read the NFC Card info
- Card is an ISO 14443A, standard for RFID, 13.56 MHz
- This card is a MiFare Classic (MFC) Card

```
kix@seahorse:~/src/nfc/faqin/script$ nfc-list -v
nfc-list uses libnfc 1.7.1
NFC device: ACS / ACR122U PICC Interface opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
* UID size: single
* bit frame anticollision supported
       UID (NFCID1): 3a  92  ed  1a
      SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092

Fingerprinting based on MIFARE type Identification Procedure:
* MIFARE Classic 1K
* MIFARE Plus (4 Byte UID or 4 Byte RID) 2K, Security level 1
* SmartMX with MIFARE 1K emulation
```

Identify the NFC Card

nfc-list

- Using nfc-list we can read the NFC Card info
- Card is an ISO 14443A, standard for RFID, 13.56 MHz
- This card is a MiFare Classic (MFC) Card

```
kix@seahorse:~/src/nfc/faqin/script$ nfc-list -v
nfc-list uses libnfc 1.7.1
NFC device: ACS / ACR122U PICC Interface opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
* UID size: single
* bit frame anticollision supported
       UID (NFCID1): 3a  92  ed  1a
      SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092

Fingerprinting based on MIFARE type Identification Procedure:
* MIFARE Classic 1K
* MIFARE Plus (4 Byte UID or 4 Byte RID) 2K, Security level 1
* SmartMX with MIFARE 1K emulation
```

MiFare Classic structure 1/2

MFC 1K structure

- 16 sectors of 4 blocks of 16 bytes:
- Every sector (4 blocks) include:
  - 3 data blocks: 48 bytes
  - 1 card info block: 2 passwords (6B + 6B) + access bytes (4B)
- First block in sector zero is the manufacturer block ("not writable"). It includes:
  - Card ID (UUID), 4 bytes
  - ATQA: Answer To reQuest Action (00 04 to MFC), 2 bytes
  - SAK: Select AcKnowledge (08 to MFC), 1 byte
  - CHECK: Checksum, 1 byte
  - 2 * 6 bytes of passwords + access data
- Data:
  - Max user data: 752 bytes: (1sec * 2blk + 15sec * 3blk) * 16 bytes
  - Usually, empty blocks contains zeroes

https://www.nxp.com/docs/en/application-note/AN10833.pdf

MiFare Classic structure 2/2

Example:

- This is an example with the FAQin badge (yes, we need read it first)

```
kix@seahorse: ~/src/nfc/faqin/script
kix@seahorse:~/src/nfc/faqin/script$ python3 faqin.py 20190907_faqin.mfc
----------------------------------------------------------------
Checking the size, the ATQA, SAK or size is not a valid
method to detect the NFC cart type. Please use the communication
with the card to get the right info.
----------------------------------------------------------------
Mifare Classic has 320, 1.024 or 4.096 bytes
This dump has a size of 1024 bytes, so it seems to be a
valid Mifare Classic dump. Checkin it.
UUID: 3A 92 ED 1A
ATQA: 00 04
CHECK: 5F | Computed Check: 5F | OK, are equal
SAK: 08 [SAK's 6th bit is 0: Mifare protocol]
Header seems to be a valid for a MFC Card
0x000000: 3A 92 ED 1A 5F 08 04 00 62 63 64 65 66 67 68 69        :..._...bcdefghi
0x000010: AB BC 1C 0E 86 E6 0E 97 E6 0E C5 E7 59 08 9B D6        ............Y...
0x000020: 26 5F E6 E7 15 4C 59 38 9B 5F C6 E6 15 4C 66 1F        &_...LY8._...Lf.
0x000030: 56 4A 0C 9B 9E F7 08 77 8F 69 04 3A F4 E1 E8 93        VJ.....w.i.:....
0x000040: FE 92 05 58 89 9B 0E 8E E6 5E 08 9B 0E 93 E6 0E        ...X.....^......
0x000050: 8D E6 58 08 9B 66 DA E6 92 34 58 08 9B 59 94 9B        ..X..f...4X..Y..
0x000060: 0E 71 E6 E9 64 1A E6 59 90 9B 0E 6B E6 59 98 9B        .q..d..Y...k.Y..
0x000070: 2C 83 80 99 3C 87 08 77 8F 69 49 7C E4 36 80 88        ,...<..w.iI|.6..
0x000080: E9 64 02 E6 59 9C 9B 0E 66 E6 59 77 9B E9 64 31        .d..Y...f.Yw..d1
0x000090: E6 0E 6B E6 0D 40 5E E5 E6 2B F6 0E E7 E6 25 5E        ..k..@^..+....%^
0x0000a0: C6 EF 5F E6 F6 55 E4 2B F6 25 0E 0F 19 5E C6 EF        .._..U.+.%...^..
0x0000b0: 77 D7 F7 4B 4B C3 08 77 8F 69 44 75 03 E3 EB A7        w..KK..w.iDu....
0x0000c0: 5F B6 E6 55 66 2B F6 58 84 9B 0E E2 E6 0E EB E6        _..Uf+.X........
0x0000d0: 25 52 E8 4A DA E6 92 E2 2B F6 0D 11 25 58 B9 9B        %R.J....+...%X..
0x0000e0: 0E 08 19 25 86 6F 21 D7 2F 0E F8 E6 DA EB 92 F3        ...%.o!./.......
0x0000f0: C9 D9 95 EC 93 9A 08 77 8F 69 6B 2C 8B 1A 1D 67        .......w.ik,...g
...
0x0003a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x0003b0: 86 80 09 68 65 75 08 77 8F 69 46 F6 51 F1 9C 85        ...heu.w.iF.Q...
0x0003c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x0003d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x0003e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x0003f0: D8 6C A4 CD 99 21 08 77 8F 69 65 2D F8 30 17 C6        .l...!.w.ie-.0..
kix@seahorse:~/src/nfc/faqin/script$
```

Read the NFC Card

Reading the badge...

Read the card using **mfoc** 1/4

mfoc

- **mfoc** is an utility to read MFC cards
- It uses default passwords or dictionaries to read the contents

```
kix@seahorse:~/src/nfc/faqin/script$ mfoc -h
Usage: mfoc [-h] [-k key] [-f file] ... [-P probnum] [-T tolerance] [-O output]

  h     print this help and exit
  k     try the specified key in addition to the default keys
  f     parses a file of keys to add in addition to the default keys
  P     number of probes per sector, instead of default of 20
  T     nonce tolerance half-range, instead of default of 20
        (i.e., 40 for the total range, in both directions)
  O     file in which the card contents will be written (REQUIRED)
  D     file in which partial card info will be written in case PRNG is not...

Example: mfoc -O mycard.mfd
Example: mfoc -k ffffeeeedddd -O mycard.mfd
Example: mfoc -f keys.txt -O mycard.mfd
Example: mfoc -P 50 -T 30 -O mycard.mfd
```

Read the card using **mfoc** 1/4

mfoc

- **mfoc** is an utility to read MFC cards
- It uses default passwords or dictionaries to read the contents

```
kix@seahorse:~/src/nfc/faqin/script$ mfoc -h
Usage: mfoc [-h] [-k key] [-f file] ... [-P probnum] [-T tolerance] [-O output]

  h     print this help and exit
  k     try the specified key in addition to the default keys
  f     parses a file of keys to add in addition to the default keys
  P     number of probes per sector, instead of default of 20
  T     nonce tolerance half-range, instead of default of 20
        (i.e., 40 for the total range, in both directions)
  O     file in which the card contents will be written (REQUIRED)
  D     file in which partial card info will be written in case PRNG is not...

Example: mfoc -O mycard.mfd
Example: mfoc -k ffffeeeedddd -O mycard.mfd
Example: mfoc -f keys.txt -O mycard.mfd
Example: mfoc -P 50 -T 30 -O mycard.mfd
```

Read the card using **mfoc** 2/4

mfoc

- Trying `mfoc` with default passwords:

```
kix@seahorse:~/src/nfc/faqin/script$ mfoc -P 50 -T 30 -O mycard.mfd
-----8<----- snip -----8<-----

Try to authenticate to all sectors with default keys...
Symbols: '.' no key found, '/' A key found, '\' B key found, 'x' both keys found
[Key: ffffffffffff] -> [................]
[Key: a0a1a2a3a4a5] -> [................]
[Key: d3f7d3f7d3f7] -> [................]
[Key: 000000000000] -> [................]
[Key: b0b1b2b3b4b5] -> [................]
[Key: 4d3a99c351dd] -> [................]
[Key: 1a982c7e459a] -> [................]
[Key: aabbccddeeff] -> [................]
[Key: 714c5c886e97] -> [................]
[Key: 587ee5f9350f] -> [................]
[Key: a0478cc39091] -> [................]
[Key: 533cb6c723f6] -> [................]
[Key: 8fd0a4f256e9] -> [................]

-----8<----- snip -----8<-----
```

Read the card using **mfoc** 3/4

mfoc

- Or using with default dictionaries, like [1]

```
kix@seahorse:~/src/nfc/faqin/script$ mfoc -P 50 -T 30 -O mycard.mfd
-----8<----- snip -----8<-----

Sector 00 - Unknown Key A               Unknown Key B
Sector 01 - Unknown Key A               Unknown Key B
Sector 02 - Unknown Key A               Unknown Key B
Sector 03 - Unknown Key A               Unknown Key B
Sector 04 - Unknown Key A               Unknown Key B
Sector 05 - Unknown Key A               Unknown Key B
Sector 06 - Unknown Key A               Unknown Key B
Sector 07 - Unknown Key A               Unknown Key B
Sector 08 - Unknown Key A               Unknown Key B
Sector 09 - Unknown Key A               Unknown Key B
Sector 10 - Unknown Key A               Unknown Key B
Sector 11 - Unknown Key A               Unknown Key B
Sector 12 - Unknown Key A               Unknown Key B
Sector 13 - Unknown Key A               Unknown Key B
Sector 14 - Unknown Key A               Unknown Key B
Sector 15 - Unknown Key A               Unknown Key B
mfoc: ERROR:

No sector encrypted with the default key has been found, exiting..
```

[1] https://github.com/ikarus23/MifareClassicTool/blob/master/Mifare%20Classic%20Tool/app/src/main/assets/key-

Read the card using **mfoc** 4/4

mfoc

- It returns an unsuccessful result, unable to read the card
- We need other methods

Read the card using **mfcuk**

mfcuk

- `mfcuk` MiFare Classic Universal toolKit
- `mfcuk` is another utility to read MFC cards
- It uses different methods to read the card info:
  - Default passwords or dictionaries
  - Proxmark sniffed conversations
  - Nested, hardnested and darkside attacks

Read the card using **mfcuk**

mfcuk

- `mfcuk` MiFare Classic Universal toolKit
- `mfcuk` is another utility to read MFC cards
- It uses different methods to read the card info:
  - Default passwords or dictionaries
  - Proxmark sniffed conversations
  - Nested, hardnested and darkside attacks

Read the card using **mfcuk**

mfcuk

- `mfcuk` MiFare Classic Universal toolKit
- `mfcuk` is another utility to read MFC cards
- It uses different methods to read the card info:
  - Default passwords or dictionaries
  - Proxmark sniffed conversations
  - Nested, hardnested and darkside attacks

Read the card using **mfcuk**

mfcuk

- Attacks using default passwords and dictionaries is like with **mfoc**: :-(
- nested and hardnested attacks needs at least one key: :'(
- Then... darkside attack!

Read the card using **mfcuk**

Darkside Attack

- Is a "brute force" attack
- You can run it with something like these:

```
$ mfcuk -C -R 0:A -s 50 -S 50 -O faqin.dmp -v 3
mfcuk - 0.3.8
Mifare Classic DarkSide Key Recovery Tool - 0.3
...
------------------------------------------------
Let me entertain you!
    uid: 3a92ed1a
   type: 08
    key: 000000000000
  block: 03
diff Nt: 0
  auths: 0
------------------------------------------------
...
```

It takes some time... about 20-30 minutes

Read the card using **mfcuk**

It takes some time more... about 1-3 hours

```
...
-------------------------------------------------------
Let me entertain you!
    uid: 3a92ed1a
   type: 08
    key: 000000000000
  block: 03
diff Nt: 1416
  auths: 20851
-------------------------------------------------------

-------------------------------------------------------
Let me entertain you!
    uid: 3a92ed1a
   type: 08
    key: 000000000000
  block: 03
diff Nt: 1480
  auths: 32851
-------------------------------------------------------
...
```

Read the card using **mfcuk**

Three/four days later... wtf!
It doesn't finish! (20 min expected)

Read the card using **Proxmark**

Proxmark time...

Read the card using **Proxmark**

DarkSide with Proxmark

Trying the Darkside attack using Proxmark:

```
proxmark3> hf mf mifare
--------------------------------------------------------------------------
Executing command. Expected execution time: 25sec on average
Press button on the proxmark3 device to abort both proxmark3 and client.
--------------------------------------------------------------------------
.....Card is not vulnerable to Darkside attack (its random number generator seems to be based on the wellknown
generating polynomial with 16 effective bits only, but shows unexpected behaviour.
proxmark3>
...
```

Bruce force the NFC card?

- There is no real option with standard tools
- I created a modified version of **mfoc** to make brute force, but it takes too much time.

- Is not possible to break it. Perhaps the new Proxmark?
- Game ends here... waiting my bruteforce attack...

WhiskyLeaks or TwitterLeaks

40 days later...

Dump card keys

We have one key. We can recover all keys with this key using **mfoc**

Dump card keys

Dump the keys using **mfoc**

- With one key (666dd3de6cc1) is possible to recover all keys and dump the badge data
- We run: `mfoc -k 666dd3de6cc1 -O 20190907_faqin.mfc`

```
Sector 00 - Key A: 564a0c9b9ef7 - Key B: 043af4e1e893
Sector 01 - Key A: 2c83b0993c87 - Key B: 497ce4368088
Sector 02 - Key A: 77d7f74b4bc3 - Key B: 447503e3eba7
Sector 03 - Key A: c9d995ec939a - Key B: 6b2c8b1a1d67
Sector 04 - Key A: 02662056aecd - Key B: 666dd3de6cc1
Sector 05 - Key A: 37ab2cb03ec3 - Key B: fbdcb5468b84
Sector 06 - Key A: 8749f086d780 - Key B: a941914a824e
Sector 07 - Key A: 619585c75e3c - Key B: 349486e5b52f
Sector 08 - Key A: 704205d57dd4 - Key B: 2c59d41ecb80
Sector 09 - Key A: 020aa9d31304 - Key B: cc3e710e8bd6
Sector 10 - Key A: bd2baa246881 - Key B: 6cdb55b3603b
Sector 11 - Key A: dd27e3b0af89 - Key B: e3fab0a0f1b5
Sector 12 - Key A: 9a3cbca3cc9f - Key B: f8aa5d7591bc
Sector 13 - Key A: 5c1045cce0be - Key B: fe79d0fbb4ec
Sector 14 - Key A: 868009686575 - Key B: 46f651f19c85
Sector 15 - Key A: d86ca4cd9921 - Key B: 652df83017c6
```

Dump card contents

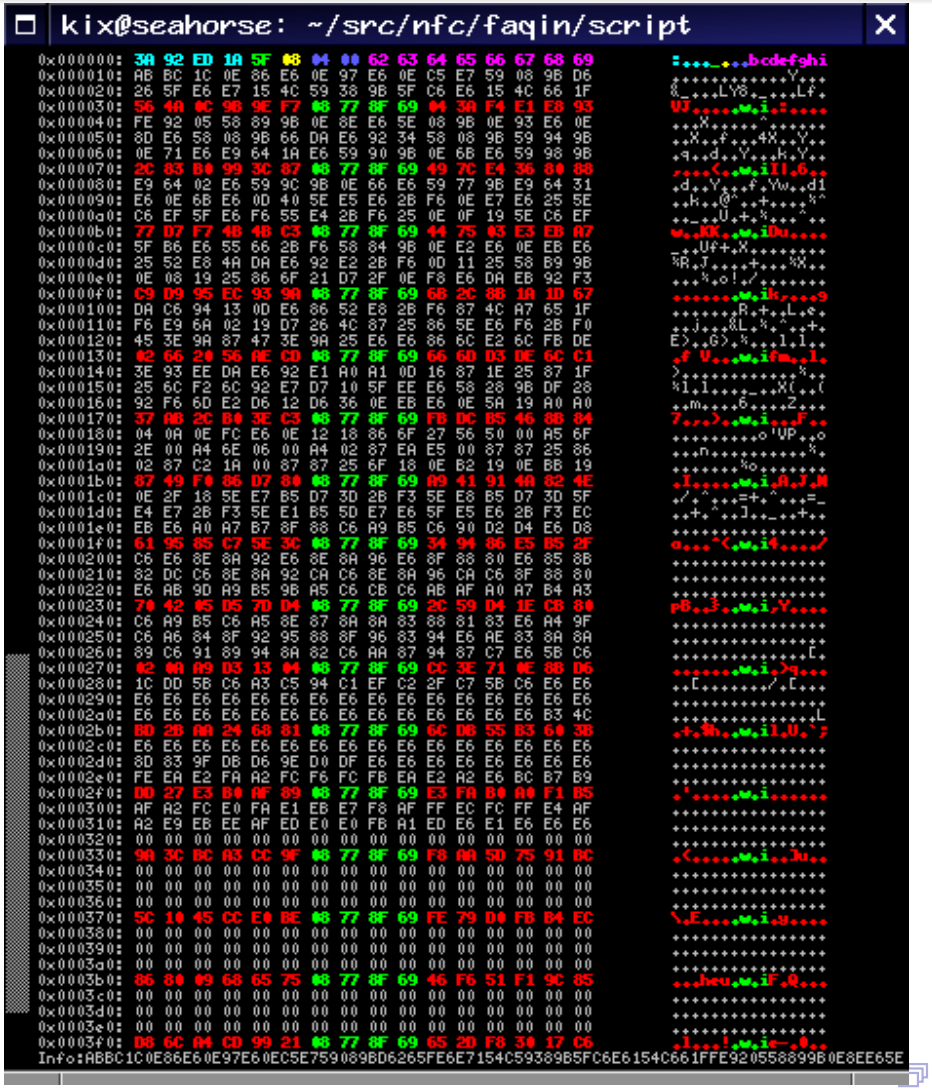
Now we have the data in 20190907_faqin.mfc

```
kix@seahorse:~/src/nfc/faqin/script$ hexdump -vC 20190907_faqin.mfc
0000  3a 92 ed 1a 5f 08 04 00  62 63 64 65 66 67 68 69  |:..._...bcdefghi|
0010  ab bc 1c 0e 86 e6 0e 97  e6 0e c5 e7 59 08 9b d6  |............Y...|
0020  26 5f e6 e7 15 4c 59 38  9b 5f c6 e6 15 4c 66 1f  |&_...LY8._...Lf.|
0030  56 4a 0c 9b 9e f7 08 77  8f 69 04 3a f4 e1 e8 93  |VJ.....w.i.:....|
0040  fe 92 05 58 89 9b 0e 8e  e6 5e 08 9b 0e 93 e6 0e  |...X.....^......|
0050  8d e6 58 08 9b 66 da e6  92 34 58 08 9b 59 94 9b  |..X..f...4X..Y..|
0060  0e 71 e6 e9 64 1a e6 59  90 9b 0e 6b e6 59 98 9b  |.q..d..Y...k.Y..|
0070  2c 83 b0 99 3c 87 08 77  8f 69 49 7c e4 36 80 88  |,...<..w.iI|.6..|
0080  e9 64 02 e6 59 9c 9b 0e  66 e6 59 77 9b e9 64 31  |.d..Y...f.Yw..d1|
0090  e6 0e 6b e6 0d 40 5e e5  e6 2b f6 0e e7 e6 25 5e  |..k..@^..+....%^|
00a0  c6 ef 5f e6 f6 55 e4 2b  f6 25 0e 0f 19 5e c6 ef  |.._..U.+.%...^..|
00b0  77 d7 f7 4b 4b c3 08 77  8f 69 44 75 03 e3 eb a7  |w..KK..w.iDu....|
00c0  5f b6 e6 55 66 2b f6 58  84 9b 0e e2 e6 0e eb e6  |_..Uf+.X........|
00d0  25 52 e8 4a da e6 92 e2  2b f6 0d 11 25 58 b9 9b  |%R.J....+...%X..|
00e0  0e 08 19 25 86 6f 21 d7  2f 0e f8 e6 da eb 92 f3  |...%.o!./.......|
00f0  c9 d9 95 ec 93 9a 08 77  8f 69 6b 2c 8b 1a 1d 67  |.......w.ik,...g|
0100  da c6 94 13 0d e6 86 52  e8 2b f6 87 4c a7 65 1f  |.......R.+..L.e.|
0110  f6 e9 6a 02 19 d7 26 4c  87 25 86 5e e6 f6 2b f0  |..j...&L.%.^..+.|
0120  45 3e 9a 87 47 3e 9a 25  e6 e6 86 6c e2 6c fb de  |E>..G>.%...l.l..|
0130  02 66 20 56 ae cd 08 77  8f 69 66 6d d3 de 6c c1  |.f V...w.ifm..l.|
0140  3e 93 ee da e6 92 e1 a0  a1 0d 16 87 1e 25 87 1f  |>............%..|
0150  25 6c f2 6c 92 e7 d7 10  5f ee e6 58 28 9b df 28  |%l.l...._..X(..(|
0160  92 f6 6d e2 d6 12 d6 36  0e eb e6 0e 5a 19 a0 a0  |..m....6....Z...|
0170  37 ab 2c b0 3e c3 08 77  8f 69 fb dc b5 46 8b 84  |7.,.>..w.i...F..|
0180  04 0a 0e fc e6 0e 12 18  86 6f 27 56 50 00 a5 6f  |.........o'VP..o|
0190  2e 00 a4 6e 06 00 a4 02  87 ea e5 00 87 87 25 86  |...n..........%.|
01a0  02 87 c2 1a 00 87 87 25  6f 18 0e b2 19 0e bb 19  |.......%o.......|
01b0  87 49 f0 86 d7 80 08 77  8f 69 a9 41 91 4a 82 4e  |.I.....w.i.A.J.N|
01c0  0e 2f 18 5e e7 b5 d7 3d  2b f3 5e e8 b5 d7 3d 5f  |./.^...=+.^...=_|
01d0  e4 e7 2b f3 5e e1 b5 5d  e7 e6 5f e5 e6 2b f3 ec  |..+.^..]..._..+..|
01e0  eb e6 a0 a7 b7 8f 88 c6  a9 b5 c6 90 d2 d4 e6 d8  |................|
01f0  61 95 85 c7 5e 3c 08 77  8f 69 34 94 86 e5 b5 2f  |a...^<.w.i4..../|
0200  c6 e6 8e 8a 92 e6 8e 8a  96 e6 8f 88 80 e6 85 8b  |................|
0210  82 dc c6 8e 8a 92 ca c6  8e 8a 96 ca c6 8f 88 80  |................|
0220  e6 ab 9d a9 b5 9b a5 c6  cb c6 ab af a0 a7 b4 a3  |................|
0230  70 42 05 d5 7d d4 08 77  8f 69 2c 59 d4 1e cb 80  |pB..}..w.i,Y....|
0240  c6 a9 b5 c6 a5 8e 87 8a  8a 83 88 81 83 e6 a4 9f  |................|
0250  c6 a6 84 8f 92 95 88 8f  96 83 94 e6 ae 83 8a 8a  |................|
0260  89 c6 91 89 94 8a 82 c6  aa 87 94 87 c7 e6 5b c6  |..............[.|
0270  02 0a a9 d3 13 04 08 77  8f 69 cc 3e 71 0e 8b d6  |.......w.i.>q...|
0280  1c dd 5b c6 a3 c5 94 c1  ef c2 2f c7 5b c6 e6 e6  |..[......./.[...|
0290  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02a0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 b3 4c  |...............L|
02b0  bd 2b aa 24 68 81 08 77  8f 69 6c db 55 b3 60 3b  |.+.$h..w.il.U.`;|
02c0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02d0  8d 83 9f db d6 9e d0 df  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02e0  fe ea e2 fa a2 fc f6 fc  fb ea e2 a2 e6 bc b7 b9  |................|
02f0  dd 27 e3 b0 af 89 08 77  8f 69 e3 fa b0 a0 f1 b5  |.'.....w.i......|
0300  af a2 fc e0 fa e1 eb e7  f8 af ff ec fc ff e4 af  |................|
0310  a2 e9 eb ee af ed e0 e0  fb a1 ed e6 e1 e6 e6 e6  |................|
0320  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0330  9a 3c bc a3 cc 9f 08 77  8f 69 f8 aa 5d 75 91 bc  |.<.....w.i..]u..|
0340  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0350  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0360  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0370  5c 10 45 cc e0 be 08 77  8f 69 fe 79 d0 fb b4 ec  |\.E....w.i.y....|
0380  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0390  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
03a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
03b0  86 80 09 68 65 75 08 77  8f 69 46 f6 51 f1 9c 85  |...heu.w.iF.Q...|
03c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
03d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
03e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
03f0  d8 6c a4 cd 99 21 08 77  8f 69 65 2d f8 30 17 c6  |.l...!.w.ie-.0..|
```

Outline

Card dump

This is the dump (hex), including the keys (red), access control (green),...



Card dump - Only data

This is the dump (hex). Only data.

```
0010  ab bc 1c 0e 86 e6 0e 97  e6 0e c5 e7 59 08 9b d6  |............Y...|
0020  26 5f e6 e7 15 4c 59 38  9b 5f c6 e6 15 4c 66 1f  |&_...LY8._...Lf.|
0040  fe 92 05 58 89 9b 0e 8e  e6 5e 08 9b 0e 93 e6 0e  |...X.....^......|
0050  8d e6 58 08 9b 66 da e6  92 34 58 08 9b 59 94 9b  |..X..f...4X..Y..|
0060  0e 71 e6 e9 64 1a e6 59  90 9b 0e 6b e6 59 98 9b  |.q..d..Y...k.Y..|
0080  e9 64 02 e6 59 9c 9b 0e  66 e6 59 77 9b e9 64 31  |.d..Y...f.Yw..d1|
0090  e6 0e 6b e6 0d 40 5e e5  e6 2b f6 0e e7 e6 25 5e  |..k..@^..+....%^|
00a0  c6 ef 5f e6 f6 55 e4 2b  f6 25 0e 0f 19 5e c6 ef  |.._..U.+.%...^..|
00c0  5f b6 e6 55 66 2b f6 58  84 9b 0e e2 e6 0e eb e6  |_..Uf+.X........|
00d0  25 52 e8 4a da e6 92 e2  2b f6 0d 11 25 58 b9 9b  |%R.J....+...%X..|
00e0  0e 08 19 25 86 6f 21 d7  2f 0e f8 e6 da eb 92 f3  |...%.o!./.......|
0100  da c6 94 13 0d e6 86 52  e8 2b f6 87 4c a7 65 1f  |.......R.+..L.e.|
0110  f6 e9 6a 02 19 d7 26 4c  87 25 86 5e e6 f6 2b f0  |..j...&L.%.^..+.|
0120  45 3e 9a 87 47 3e 9a 25  e6 e6 86 6c e2 6c fb de  |E>..G>.%...l.l..|
0140  3e 93 ee da e6 92 e1 a0  a1 0d 16 87 1e 25 87 1f  |>............%..|
0150  25 6c f2 6c 92 e7 d7 10  5f ee e6 58 28 9b df 28  |%l.l...._..X(..(|
0160  92 f6 6d e2 d6 12 d6 36  0e eb e6 0e 5a 19 a0 a0  |..m....6....Z...|
0180  04 0a 0e fc e6 0e 12 18  86 6f 27 56 50 00 a5 6f  |.........o'VP..o|
0190  2e 00 a4 6e 06 00 a4 02  87 ea e5 00 87 87 25 86  |...n..........%.|
01a0  02 87 c2 1a 00 87 87 25  6f 18 0e b2 19 0e bb 19  |.......%o.......|
01c0  0e 2f 18 5e e7 b5 d7 3d  2b f3 5e e8 b5 d7 3d 5f  |./.^...=+.^...=_|
01d0  e4 e7 2b f3 5e e1 b5 5d  e7 e6 5f e5 e6 2b f3 ec  |..+.^..]..._..+..|
01e0  eb e6 a0 a7 b7 8f 88 c6  a9 b5 c6 90 d2 d4 e6 d8  |................|
0200  c6 e6 8e 8a 92 e6 8e 8a  96 e6 8f 88 80 e6 85 8b  |................|
0210  82 dc c6 8e 8a 92 ca c6  8e 8a 96 ca c6 8f 88 80  |................|
0220  e6 ab 9d a9 b5 9b a5 c6  cb c6 ab af a0 a7 b4 a3  |................|
0240  c6 a9 b5 c6 a5 8e 87 8a  8a 83 88 81 83 e6 a4 9f  |................|
0250  c6 a6 84 8f 92 95 88 8f  96 83 94 e6 ae 83 8a 8a  |................|
0260  89 c6 91 89 94 8a 82 c6  aa 87 94 87 c7 e6 5b c6  |..............[.|
0280  1c dd 5b c6 a3 c5 94 c1  ef c2 2f c7 5b c6 e6 e6  |..[......./.[...|
0290  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02a0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 b3 4c  |...............L|
02c0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02d0  8d 83 9f db d6 9e d0 df  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02e0  fe ea e2 fa a2 fc f6 fc  fb ea e2 a2 e6 bc b7 b9  |................|
0300  af a2 fc e0 fa e1 eb e7  f8 af ff ec fc ff e4 af  |................|
0310  a2 e9 eb ee af ed e0 e0  fb a1 ed e6 e1 e6 e6 e6  |................|
0320  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
0340  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
0350  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
0360  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
0380  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
0390  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
03a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
03c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
03d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
03e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| ----> Empty!
```

Card dump - Valid data only

Valid data only, without the 00's (probably no info)

```
0010  ab bc 1c 0e 86 e6 0e 97  e6 0e c5 e7 59 08 9b d6  |............Y...|
0020  26 5f e6 e7 15 4c 59 38  9b 5f c6 e6 15 4c 66 1f  |&_...LY8._...Lf.|
0040  fe 92 05 58 89 9b 0e 8e  e6 5e 08 9b 0e 93 e6 0e  |...X.....^......|
0050  8d e6 58 08 9b 66 da e6  92 34 58 08 9b 59 94 9b  |..X..f...4X..Y..|
0060  0e 71 e6 e9 64 1a e6 59  90 9b 0e 6b e6 59 98 9b  |.q..d..Y...k.Y..|
0080  e9 64 02 e6 59 9c 9b 0e  66 e6 59 77 9b e9 64 31  |.d..Y...f.Yw..d1|
0090  e6 0e 6b e6 0d 40 5e e5  e6 2b f6 0e e7 e6 25 5e  |..k..@^..+....%^|
00a0  c6 ef 5f e6 f6 55 e4 2b  f6 25 0e 0f 19 5e c6 ef  |.._..U.+.%...^..|
00c0  5f b6 e6 55 66 2b f6 58  84 9b 0e e2 e6 0e eb e6  |_..Uf+.X........|
00d0  25 52 e8 4a da e6 92 e2  2b f6 0d 11 25 58 b9 9b  |%R.J....+...%X..|
00e0  0e 08 19 25 86 6f 21 d7  2f 0e f8 e6 da eb 92 f3  |...%.o!./.......|
0100  da c6 94 13 0d e6 86 52  e8 2b f6 87 4c a7 65 1f  |.......R.+..L.e.|
0110  f6 e9 6a 02 19 d7 26 4c  87 25 86 5e e6 f6 2b f0  |..j...&L.%.^..+.|
0120  45 3e 9a 87 47 3e 9a 25  e6 e6 86 6c e2 6c fb de  |E>..G>.%...l.l..|
0140  3e 93 ee da e6 92 e1 a0  a1 0d 16 87 1e 25 87 1f  |>............%..|
0150  25 6c f2 6c 92 e7 d7 10  5f ee e6 58 28 9b df 28  |%l.l...._..X(..(|
0160  92 f6 6d e2 d6 12 d6 36  0e eb e6 0e 5a 19 a0 a0  |..m....6....Z...|
0180  04 0a 0e fc e6 0e 12 18  86 6f 27 56 50 00 a5 6f  |.........o'VP..o|
0190  2e 00 a4 6e 06 00 a4 02  87 ea e5 00 87 87 25 86  |...n..........%.|
01a0  02 87 c2 1a 00 87 87 25  6f 18 0e b2 19 0e bb 19  |.......%o.......|
01c0  0e 2f 18 5e e7 b5 d7 3d  2b f3 5e e8 b5 d7 3d 5f  |./.^...=+.^...=_|
01d0  e4 e7 2b f3 5e e1 b5 5d  e7 e6 5f e5 e6 2b f3 ec  |..+.^..]..._..+..|
01e0  eb e6 a0 a7 b7 8f 88 c6  a9 b5 c6 90 d2 d4 e6 d8  |................|
0200  c6 e6 8e 8a 92 e6 8e 8a  96 e6 8f 88 80 e6 85 8b  |................|
0210  82 dc c6 8e 8a 92 ca c6  8e 8a 96 ca c6 8f 88 80  |................|
0220  e6 ab 9d a9 b5 9b a5 c6  cb c6 ab af a0 a7 b4 a3  |................|
0240  c6 a9 b5 c6 a5 8e 87 8a  8a 83 88 81 83 e6 a4 9f  |................|
0250  c6 a6 84 8f 92 95 88 8f  96 83 94 e6 ae 83 8a 8a  |................|
0260  89 c6 91 89 94 8a 82 c6  aa 87 94 87 c7 e6 5b c6  |..............[.|
0280  1c dd 5b c6 a3 c5 94 c1  ef c2 2f c7 5b c6 e6 e6  |..[......./.[...|
0290  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................| <---- e6 e6 :-?
02a0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 b3 4c  |...............L|
02c0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................| <---- e6 e6 :-?
02d0  8d 83 9f db d6 9e d0 df  e6 e6 e6 e6 e6 e6 e6 e6  |................|
02e0  fe ea e2 fa a2 fc f6 fc  fb ea e2 a2 e6 bc b7 b9  |................|
0300  af a2 fc e0 fa e1 eb e7  f8 af ff ec fc ff e4 af  |................|
0310  a2 e9 eb ee af ed e0 e0  fb a1 ed e6 e1 e6 e6 e6  |................|
```

Discovering data

Looking the ciphered info

- We can see a lot of "e6" in the end of the valid data
- Probably these info is "00", so we can apply an XOR

```
01d0  e4 e7 2b f3 5e e1 b5 5d  e7 e6 5f e5 e6 2b f3 ec  |..+.^..]..._..+..|
01e0  eb e6 a0 a7 b7 8f 88 c6  a9 b5 c6 90 d2 d4 e6 d8  |................|
0200  c6 e6 8e 8a 92 e6 8e 8a  96 e6 8f 88 80 e6 85 8b  |................|
0210  82 dc c6 8e 8a 92 ca c6  8e 8a 96 ca c6 8f 88 80  |................|
0220  e6 ab 9d a9 b5 9b a5 c6  cb c6 ab af a0 a7 b4 a3  |................|
0240  c6 a9 b5 c6 a5 8e 87 8a  8a 83 88 81 83 e6 a4 9f  |................|
0250  c6 a6 84 8f 92 95 88 8f  96 83 94 e6 ae 83 8a 8a  |................|
0260  89 c6 91 89 94 8a 82 c6  aa 87 94 87 c7 e6 5b c6  |..............[.|
0280  1c dd 5b c6 a3 c5 94 c1  ef c2 2f c7 5b c6 e6 e6  |..[......./.[...| <---- e6 e6
0290  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................| <---- e6 e6
02a0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 b3 4c  |...............L| <---- e6 e6
02c0  e6 e6 e6 e6 e6 e6 e6 e6  e6 e6 e6 e6 e6 e6 e6 e6  |................| <---- e6 e6
02d0  8d 83 9f db d6 9e d0 df  e6 e6 e6 e6 e6 e6 e6 e6  |................| <---- e6 e6
02e0  fe ea e2 fa a2 fc f6 fc  fb ea e2 a2 e6 bc b7 b9  |................|
0300  af a2 fc e0 fa e1 eb e7  f8 af ff ec fc ff e4 af  |................|
0310  a2 e9 eb ee af ed e0 e0  fb a1 ed e6 e1 e6 e6 e6  |................| <---- e6 e6
```

Discovering data

```
Decripting using password E6E6E6E6E6E6E6E6, Function or
0x000010: 4D 5A FA E8 60 00 E8 71 00 E8 23 01 BF EE 7D 30   MZ..‘..q..#...}0 ----> Block 1
0x000020: C0 B9 00 01 F3 AA BF DE 7D B9 20 00 F3 AA 80 F9   ........}. .....
0x000030: 18 74 E3 BE 6F 7D E8 68 00 B8 EE 7D E8 75 00 E8   .t..o}.h...}.u..
0x000040: 6B 00 BE EE 7D 80 3C 00 74 D2 BE EE 7D BF 72 7D   k...}.<.t...}.r}
0x000050: E8 97 00 0F 82 FC 00 BF 76 7D E8 8D 00 BF 7E 7D   ........v}....~}
0x000060: 0F 82 E4 00 BF 7A 7D E8 80 00 BF 91 7D 0F 82 D7   .....z}.....}...
0x000070: 00 E8 8D 00 EB A6 B8 03 00 CD 10 E8 01 00 C3 B8   ................
0x000080: 20 09 B9 00 10 B3 02 CD 10 C3 E8 E9 FF B8 20 09    ............. .
0x000090: B9 50 00 B3 80 CD 10 BE 62 7D E8 04 00 E8 0D 00   .P......b}......
0x0000a0: C3 B4 0E AC 3C 00 74 04 CD 10 EB F7 C3 BE 5F 7D   ....<.t......._}
0x0000b0: E8 EE FF C3 60 89 C7 31 C9 E8 1E 00 3C 0D 74 15   ....‘..1....<.t.
0x0000c0: 3C 20 72 F5 EB 00 60 B4 0E CD 10 61 AA 41 83 F9   < r...‘....a.A..
0x0000d0: 10 0F 8C E4 FF 31 C0 AA 61 C3 60 B8 00 10 CD 16   .....1..a.‘.....
0x0000e0: A3 D8 7C 61 A1 D8 7C C3 00 00 60 8A 04 8A 1D 38   ..|a..|...‘....8
0x0000f0: D8 75 08 3C 00 74 07 46 47 EB F0 61 F8 C3 61 F9   .u.<.t.FG..a..a.
0x000100: C3 8A 14 8A 74 01 31 F6 B9 08 00 BE CE 7D 39 CE   ....t.1......}9.
0x000110: 74 10 8B 04 30 F4 30 D0 E8 0D 00 E8 BC FF 46 46   t...0.0.......FF
0x000120: E2 EC E8 1A 00 E8 F4 FE 60 89 C1 B0 B6 E6 43 89   ........‘.....C.
0x000130: C8 E6 42 88 E0 E6 42 E4 61 0C 03 E6 61 61 C3 60   ..B...B.a...aa.‘
0x000140: E4 61 24 FC E6 61 61 C3 89 FE E8 54 FF E8 5D FF   .a$..aa....T..].
0x000150: E8 C9 FE B8 01 53 31 DB CD 15 B8 0E 53 31 DB B9   .....S1.....S1..
0x000160: 02 01 CD 15 B8 07 53 BB 01 00 B9 03 00 CD 15 0A   ......S.........
0x000170: 0D 00 46 41 51 69 6E 20 4F 53 20 76 34 32 00 3E   ..FAQin OS v42.>
0x000180: 20 00 68 6C 74 00 68 6C 70 00 69 6E 66 00 63 6D    .hlt.hlp.inf.cm
0x000190: 64 3A 20 68 6C 74 2C 20 68 6C 70 2C 20 69 6E 66   d: hlt, hlp, inf
0x0001a0: 00 4D 7B 4F 53 7D 43 20 2D 20 4D 49 46 41 52 45   .M{OS}C - MIFARE
0x0001b0: 20 4F 53 20 43 68 61 6C 6C 65 6E 67 65 00 42 79    OS Challenge.By
0x0001c0: 20 40 62 69 74 73 6E 69 70 65 72 00 48 65 6C 6C    @bitsniper.Hell
0x0001d0: 6F 20 77 6F 72 6C 64 20 4C 61 72 61 21 00 BD 20   o world Lara!..
0x0001e0: FA 3B BD 20 45 23 72 27 09 24 C9 21 BD 20 00 00   .;. E#r’.$.!. ..
0x0001f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x000200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA   ..............U.
0x000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x000220: 6B 65 79 3D 30 78 36 39 00 00 00 00 00 00 00 00   key=0x69........ ----> Block 2
0x000230: 18 0C 04 1C 44 1A 10 1A 1D 0C 04 44 00 5A 51 5F   ....D......D.ZQ_ ----> Block 3
0x000240: 49 44 1A 06 1C 07 0D 01 1E 49 19 0A 1A 19 02 49   ID.......I.....I
0x000250: 44 0F 0D 08 49 0B 06 06 1D 47 0B 00 07 00 00 00   D...I....G......
0x000260: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x000270: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x000280: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x000290: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x0002a0: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x0002b0: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x0002c0: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x0002d0: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x0002e0: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
0x0002f0: E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6 E6   ................ ----> Empty
```

Data discovered. Review

- Block 1: Seems to be a valid data. It starts with 4D5A!
- Block 2: key=0x69. A key for something...
- Block 3: ??

Block 1 - MS-DOS Executable

DOS MZ executable

From Wikipedia, the free encyclopedia

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.

Find sources: "DOS MZ executable" – news · newspapers · books · scholar · JSTOR *(April 2015) (Learn how and when to remove this template message)*

The **DOS MZ executable** format is the executable file format used for .EXE files in DOS.

The file can be identified by the ASCII string "MZ" (hexadecimal: 4D 5A) at the beginning of the file (the "magic number"). "MZ" are the initials of Mark Zbikowski, one of leading developers of MS-DOS.[1]

The MZ DOS executable file is newer than the COM executable format and differs from it. The DOS executable header contains relocation information, which allows multiple segments to be loaded at arbitrary memory addresses, and it supports executables larger than 64k; however, the format still requires relatively low memory limits. These limits were later bypassed using DOS extenders.

The environment of an EXE program run by DOS is found in its Program Segment Prefix.

DOS MZ executable

Filename extension	.exe
Magic number	MZ or ZM
Type of format	Binary, executable
Extended to	New Executable Linear Executable Portable Executable

- As it starts with 4D 5A it seems to be a MS-DOS exe file
- I tried to run it with **dosbox**, but nothing happens...
- Check the info again!

Block 1 - Check the info again...

If we look this block again...

```
0x000010: 4D 5A FA E8 60 00 E8 71 00 E8 23 01 BF EE 7D 30        MZ..‘..q..#...}0
0x000020: C0 B9 00 01 F3 AA BF DE 7D B9 20 00 F3 AA 80 F9        ........}. .....
0x000030: 18 74 E3 BE 6F 7D E8 68 00 B8 EE 7D E8 75 00 E8        .t..o}.h...}.u..
0x000040: 6B 00 BE EE 7D 80 3C 00 74 D2 BE EE 7D BF 72 7D        k...}.<.t...}.r}
0x000050: E8 97 00 0F 82 FC 00 BF 76 7D E8 8D 00 BF 7E 7D        ........v}....~}
0x000060: 0F 82 E4 00 BF 7A 7D E8 80 00 BF 91 7D 0F 82 D7        .....z}.....}...
0x000070: 00 E8 8D 00 EB A6 B8 03 00 CD 10 E8 01 00 C3 B8        ................
0x000080: 20 09 B9 00 10 B3 02 CD 10 C3 E8 E9 FF B8 20 09         .............. .
0x000090: B9 50 00 B3 80 CD 10 BE 62 7D E8 04 00 E8 0D 00        .P......b}......
0x0000a0: C3 B4 0E AC 3C 00 74 04 CD 10 EB F7 C3 BE 5F 7D        ....<.t........_}
0x0000b0: E8 EE FF C3 60 89 C7 31 C9 E8 1E 00 3C 0D 74 15        ....‘..1....<.t.
0x0000c0: 3C 20 72 F5 EB 00 60 B4 0E CD 10 61 AA 41 83 F9        < r...‘....a.A..
0x0000d0: 10 0F 8C E4 FF 31 C0 AA 61 C3 60 B8 00 10 CD 16        .....1..a.‘.....
0x0000e0: A3 D8 7C 61 A1 D8 7C C3 00 00 60 8A 04 8A 1D 38        ..|a..|...‘....8
0x0000f0: D8 75 08 3C 00 74 07 46 47 EB F0 61 F8 C3 61 F9        .u.<.t.FG..a..a.
0x000100: C3 8A 14 8A 74 01 31 F6 B9 08 00 BE CE 7D 39 CE        ....t.1......}9.
0x000110: 74 10 8B 04 30 F4 30 D0 E8 0D 00 E8 BC FF 46 46        t...0.0.......FF
0x000120: E2 EC E8 1A 00 E8 F4 FE 60 89 C1 B0 B6 E6 43 89        ........‘.....C.
0x000130: C8 E6 42 88 E0 E6 42 E4 61 0C 03 E6 61 61 C3 60        ..B...B.a...aa.‘
0x000140: E4 61 24 FC E6 61 61 C3 89 FE E8 54 FF E8 5D FF        .a$..aa....T..].
0x000150: E8 C9 FE B8 01 53 31 DB CD 15 B8 0E 53 31 DB B9        .....S1.....S1..
0x000160: 02 01 CD 15 B8 07 53 BB 01 00 B9 03 00 CD 15 0A        ......S.........
0x000170: 0D 00 46 41 51 69 6E 20 4F 53 20 76 34 32 00 3E        ..FAQin OS v42.>
0x000180: 20 00 68 6C 74 00 68 6C 70 00 69 6E 66 00 63 6D         .hlt.hlp.inf.cm
0x000190: 64 3A 20 68 6C 74 2C 20 68 6C 70 2C 20 69 6E 66        d: hlt, hlp, inf
0x0001a0: 00 4D 7B 4F 53 7D 43 20 2D 20 4D 49 46 41 52 45        .M{OS}C - MIFARE
0x0001b0: 20 4F 53 20 43 68 61 6C 6C 65 6E 67 65 00 42 79         OS Challenge.By
0x0001c0: 20 40 62 69 74 73 6E 69 70 65 72 00 48 65 6C 6C         @bitsniper.Hell
0x0001d0: 6F 20 77 6F 72 6C 64 20 4C 61 72 61 21 00 BD 20        o world Lara!..
0x0001e0: FA 3B BD 20 45 23 72 27 09 24 C9 21 BD 20 00 00        .;. E#r'.$.!. ..
0x0001f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ................
0x000200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA        ..............U. ---> Ends with 55 AA
```

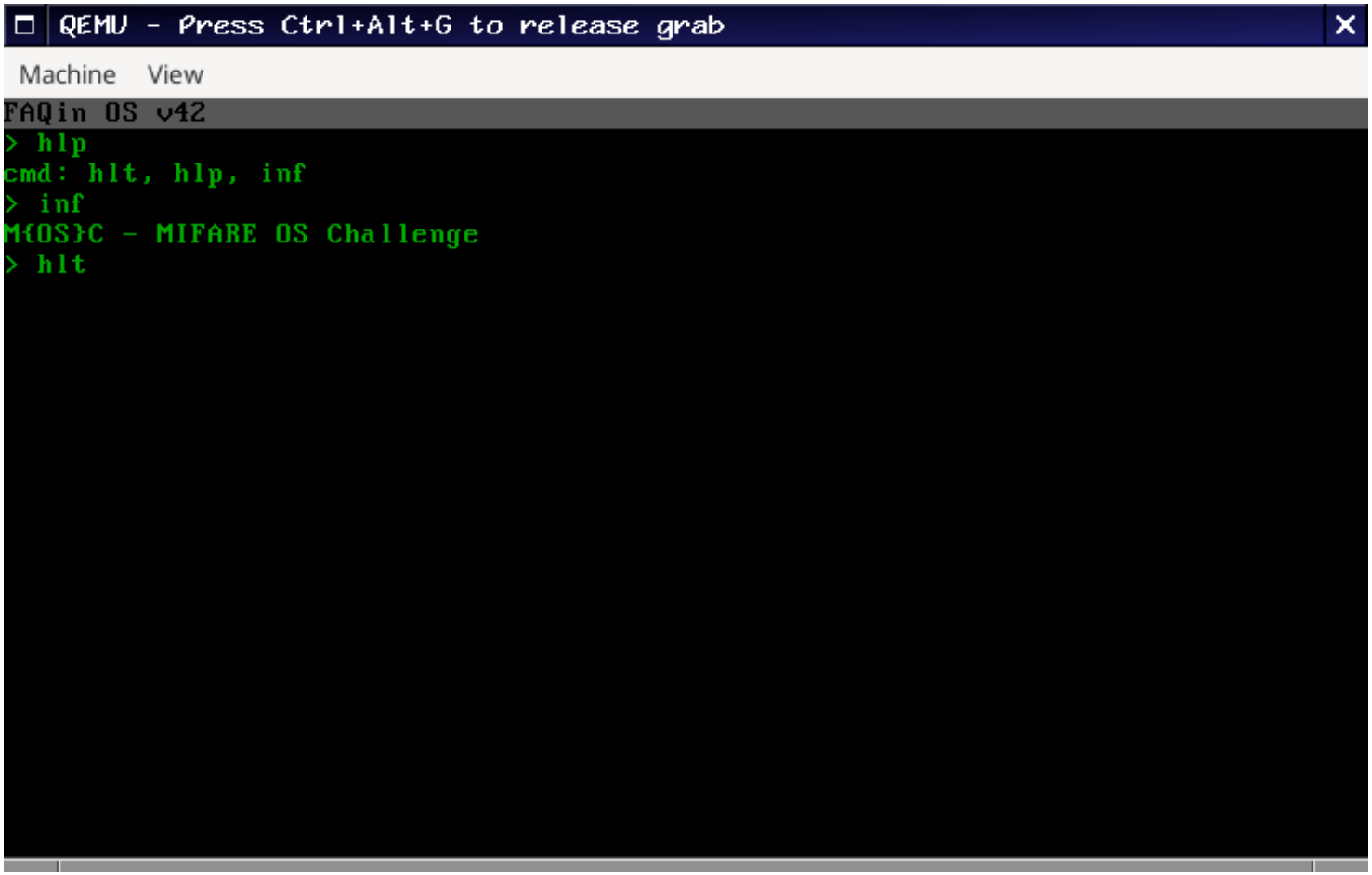
Block 1 - Check the info again... it is an MBR!

Is it an MBR!

```
kix@seahorse:~/src/nfc/faqin$ file 20190907_faqin.iso
20190907_faqin.iso: DOS/MBR boot sector
kix@seahorse:~/src/nfc/faqin$
```

Test it!!! `qemu-system-i386 -fda 20190907_faqin.iso`

Block 1 - Check the info again...



Remember the commands in the code... But... nothing happens :-(

```
0x000170: 0D 00 46 41 51 69 6E 20 4F 53 20 76 34 32 00 3E        ..FAQin OS v42.>
0x000180: 20 00 68 6C 74 00 68 6C 70 00 69 6E 66 00 63 6D         .hlt.hlp.inf.cm
0x000190: 64 3A 20 68 6C 74 2C 20 68 6C 70 2C 20 69 6E 66        d: hlt, hlp, inf
0x0001a0: 00 4D 7B 4F 53 7D 43 20 2D 20 4D 49 46 41 52 45        .M{OS}C - MIFARE
0x0001b0: 20 4F 53 20 43 68 61 6C 6C 65 6E 67 65 00 42 79         OS Challenge.By
0x0001c0: 20 40 62 69 74 73 6E 69 70 65 72 00 48 65 6C 6C         @bitsniper.Hell
0x0001d0: 6F 20 77 6F 72 6C 64 20 4C 61 72 61 21 00 BD 20        o world Lara!..
```

Block 1 - Image behaviour

Image behaviour

- Typing `hlp` -> get `'cmd:  hlt, hlp, inf'`
- Typing `inf` -> get `'M{OS}C - MIFARE OS Challenge'`
- Typing `hlt` -> system is halted
- Typing other things -> system seems to be stuck. Hitting `Enter` key some times, prompt comes again

Blocks - Review

- Block 1: MBR. Nothing happens... I need to disassembly the code.
- Block 2: key=0x69. A key for something...
- Block 3: Unknown —> I will try now with this block!

Block 3

Check the Block 3 using Block 2 (key=0x69) with xor again

```
0x000230: 71 65 6D 75 2D 73 79 73 74 65 6D 2D 69 33 38 36        qemu-system-i386
0x000240: 20 2D 73 6F 75 6E 64 68 77 20 70 63 73 70 6B 20         -soundhw pcspk
0x000250: 2D 66 64 61 20 62 6F 6F 74 2E 62 69 6E 69 69 69        -fda boot.biniii
```

Voila! How to launch the qemu image :-)

Block 1 - Image behaviour full

Image behaviour

- Typing `hlp` -> get `'cmd:  hlt, hlp, inf'`
- Typing `inf` -> get `'M{OS}C - MIFARE OS Challenge'`
- Typing `hlt` -> system is halted
- Typing other things -> after some key inputs (about 16) system start to play noise in the PC Speaker. I need press 8 keys to stop the noise and the prompt returns.

Block 1 - Review

- Block 1: MBR: It needs disassembly
- Block 2: key=0x69
- Block 3: How to launch the mbr image in qemu

Block 1 - Disassembly

To disassembly, we can use some different tools:

- Ida Pro
- Radare
- ...
- objdump + extras (I used this, because the code is small)

Block 1 - Disassembly

It is a MBR: Important things

- Debug in real mode
- x86 CPU with 16 bit
- start address 0x7c00

Block 1 - Disassembly

Main blocks (brief)

- 7C00 - 7D5D: Code
- 7D5F - 7DFD: Data (ASCII strings)

Block 1 - Disassembly - Code

Start	Length	Function name
7C0C	5Ah	Clean memory + main menu
7C66	09h	Set Video Mode
7C6F	0Bh	Print cursor
7C7A	17h	Print title "FaqinOS"
7C91	0Ch	Print text data on screen
7C9D	07h	Print CR + LF
7CA4	26h	Read text from prompt
7CCA	0Eh	Read key and save in AX
7CDA	17h	Compare text
7CF1	27h	Do sound loop
7D18	17h	Turn on PC speaker
7D2F	09h	Turn off PC speaker
7D38	0Bh	Print Text CR LF and restart
7D43	1Ch	PC shutdown using APM

Block 1 - Disassembly - Data

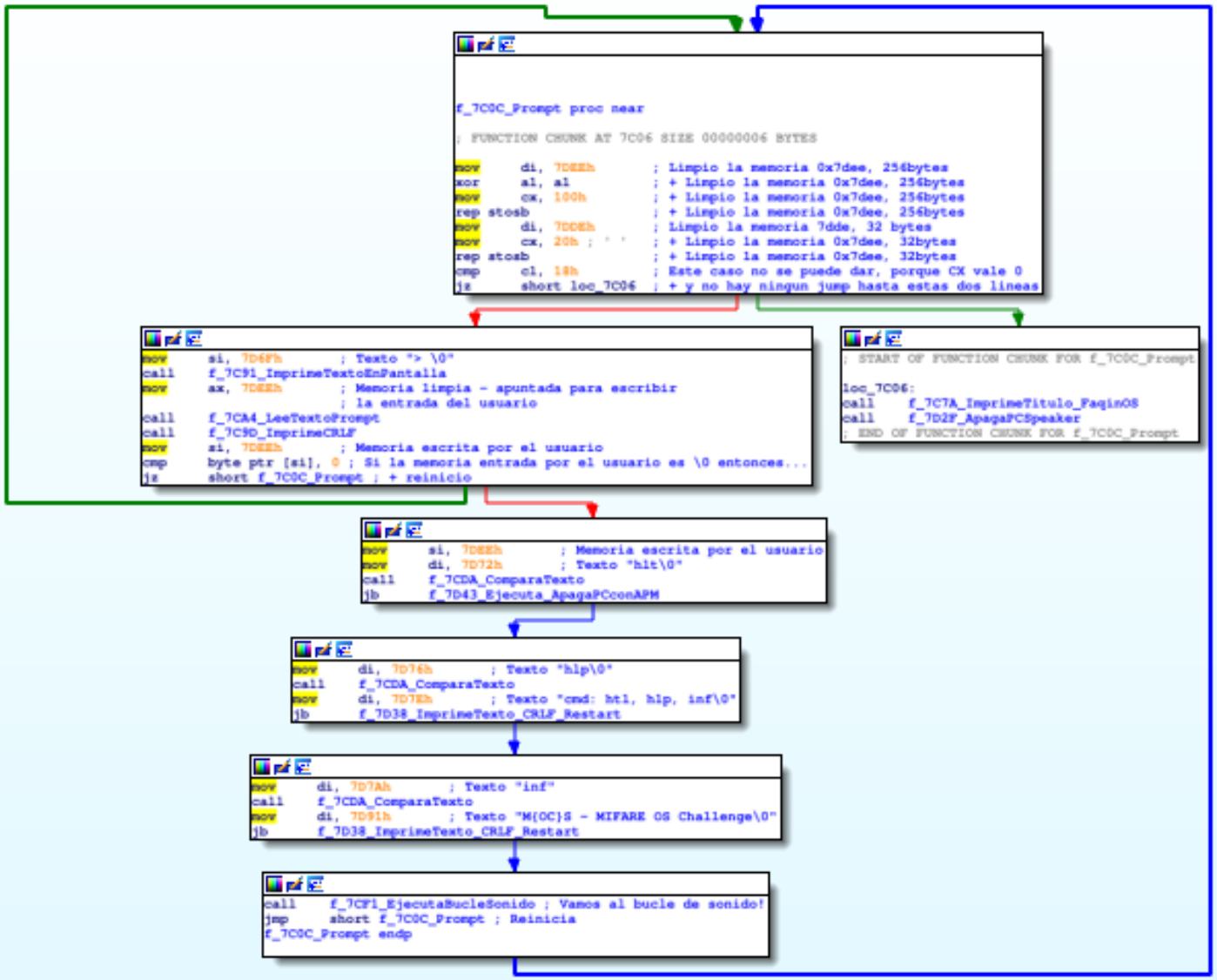
This is the data block, starting at 7D5F:

```
Data:7D5F Data            segment byte public '' use16
Data:7D5F                 assume cs:Data
Data:7D5F                 ;org 7D5Fh
Data:7D5F                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
Data:7D5F                 db 0Ah
Data:7D5F                 db 0Dh,0
Data:7D62 aFaqinOsV42     db 'FAQin OS v42',0
Data:7D6F                 db '> ',0
Data:7D72 aHlt            db 'hlt',0
Data:7D76 aHlp            db 'hlp',0
Data:7D7A aInf            db 'inf',0
Data:7D7E aCmdHltHlpInf   db 'cmd: hlt, hlp, inf',0
Data:7D91 aMOsCMifareOsCh db 'M{OS}C - MIFARE OS Challenge',0
Data:7DAE aByBitsniper    db 'By @bitsniper',0
Data:7DBC aHelloWorldLara db 'Hello world Lara!',0
Data:7DCE                 db 0BDh,' ',0FAh,';',0BDh,' E#r',27h,9,'$',0C9h,21h,0BDh,' ',0
Data:7DDF                 db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,'U',0AAh
Data:7DDF Data            ends
Data:7DDF
```

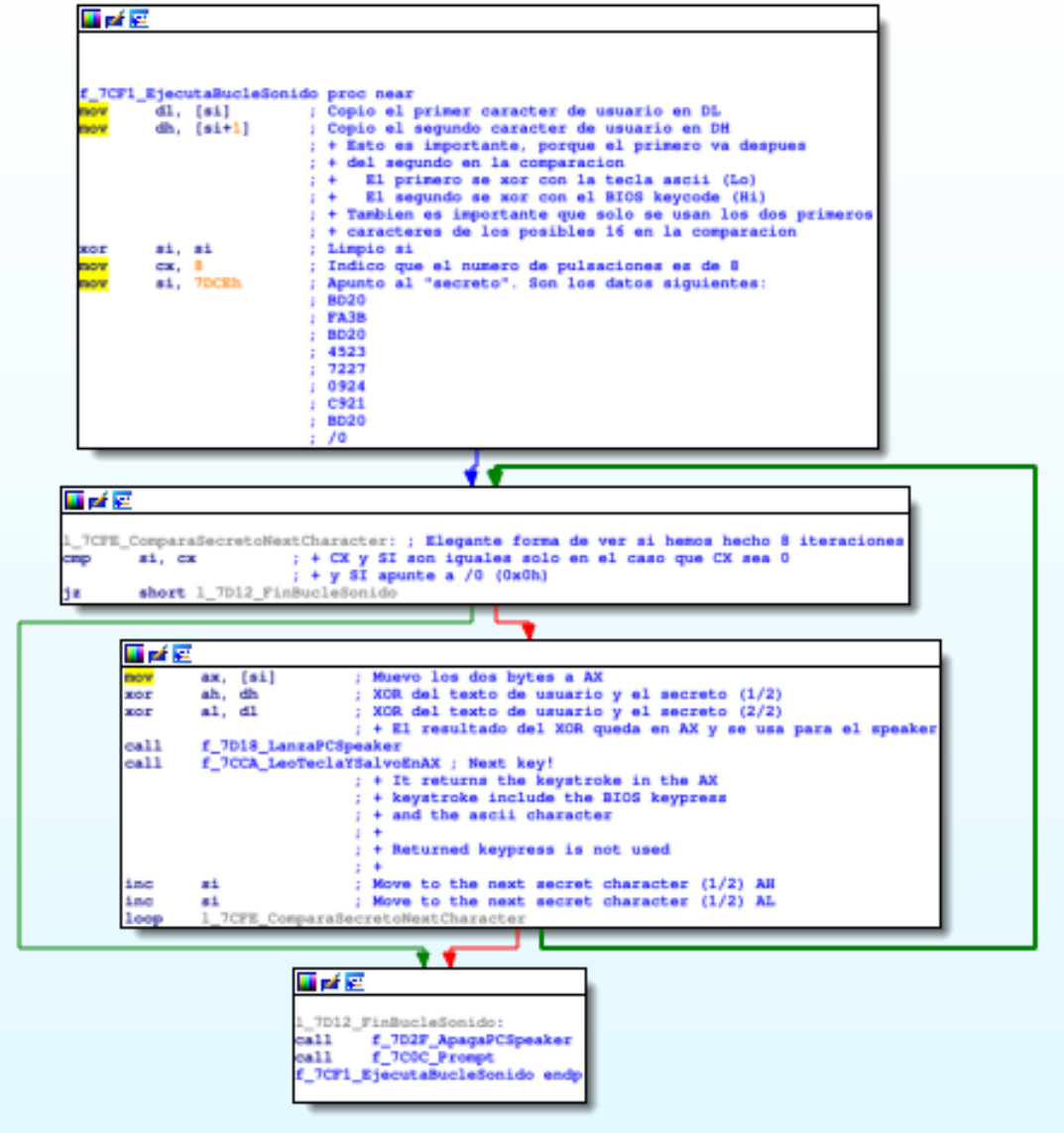
Running the code

- In main menu, the user can type:
  - hlp + Enter -> Show help -> main menu again
  - inf + Enter -> Show info -> main menu again
  - hlt + Enter -> Halt machine
  - less than 16 characters + Enter -> Enter sound loop
  - 16 characters -> Enter sound loop
- In sound loop:
  - PC speaker play a continuous tone
  - User can type 8 characters -> main menu again
- Nothing is stored in memory. Nothing is printed. Only sound.
- The sound is the key.

Block 1 - Main menu workflow



Block 1 - Sound loop workflow



Analyzing the sound loop

- Sound loop is the important block
- It compares the first two characters typed (ONLY TWO!)...
- ...with the stored datapairs in the data section (7DCE-7DDD)
  - BD 20 | FA 3B | BD 20 | 45 23 | 72 27 | 09 24 | C9 21 | BD 20
- The sound loop loads the first two characters in reg DX (DL+DH)
- The sound loop loads the data pairs characters in reg AX (AL+AH)
- It makes an XOR of AX with DX and call the PC speaker function
- PC speaker plays a set of eight tones, one per loop

We need the two typed characters to decode the datapairs!

Block 1 - Sound loop initialization

```
Code_Part_2:7CF1 f_7CF1_EjecutaBucleSonido proc near     ; CODE XREF: f_7C0C_Prompt+55↑p
Code_Part_2:7CF1                 mov     dl, [si]        ; Copio el primer caracter de usuario en DL
Code_Part_2:7CF3                 mov     dh, [si+1]      ; Copio el segundo caracter de usuario en DH
Code_Part_2:7CF3                                         ; + Esto es importante, porque el primero va despues
Code_Part_2:7CF3                                         ; + del segundo en la comparacion
Code_Part_2:7CF3                                         ; +   El primero se xor con la tecla ascii (Lo)
Code_Part_2:7CF3                                         ; +   El segundo se xor con el BIOS keycode (Hi)
Code_Part_2:7CF3                                         ; + Tambien es importante que solo se usan los dos primeros
Code_Part_2:7CF3                                         ; + caracteres de los posibles 16 en la comparacion
Code_Part_2:7CF6                 xor     si, si          ; Limpio si
Code_Part_2:7CF8                 mov     cx, 8           ; Indico que el numero de pulsaciones es de 8
Code_Part_2:7CFB                 mov     si, 7DCEh       ; Apunto al "secreto". Son los datos siguientes:
Code_Part_2:7CFB                                         ; BD20
Code_Part_2:7CFB                                         ; FA3B
Code_Part_2:7CFB                                         ; BD20
Code_Part_2:7CFB                                         ; 4523
Code_Part_2:7CFB                                         ; 7227
Code_Part_2:7CFB                                         ; 0924
Code_Part_2:7CFB                                         ; C921
Code_Part_2:7CFB                                         ; BD20
Code_Part_2:7CFB                                         ; /0
Code_Part_2:7CFE
```

Block 1 - Sound loop

```
Code_Part_2:7CFE l_7CFE_ComparaSecretoNextCharacter:     ; CODE XREF: f_7CF1_EjecutaBucleSonido+1F↓j
Code_Part_2:7CFE                 cmp     si, cx          ; Elegante forma de ver si hemos hecho 8 iteraciones
Code_Part_2:7CFE                                         ; + CX y SI son iguales solo en el caso que CX sea 0
Code_Part_2:7CFE                                         ; + y SI apunte a /0 (0x0h)
Code_Part_2:7D00                 jz      short l_7D12_FinBucleSonido
Code_Part_2:7D02                 mov     ax, [si]        ; Muevo los dos bytes a AX
Code_Part_2:7D04                 xor     ah, dh          ; XOR del texto de usuario y el secreto (1/2)
Code_Part_2:7D06                 xor     al, dl          ; XOR del texto de usuario y el secreto (2/2)
Code_Part_2:7D06                                         ; + El resultado del XOR queda en AX y se usa para el speaker
Code_Part_2:7D08                 call    f_7D18_LanzaPCSpeaker
Code_Part_2:7D0B                 call    f_7CCA_LeoTeclaYSalvoEnAX ; Next key!
Code_Part_2:7D0B                                         ; + It returns the keystroke in the AX
Code_Part_2:7D0B                                         ; + keystroke include the BIOS keypress
Code_Part_2:7D0B                                         ; + and the ascii character
Code_Part_2:7D0B                                         ; +
Code_Part_2:7D0B                                         ; + Returned keypress is not used
Code_Part_2:7D0B                                         ; +
Code_Part_2:7D0E                 inc     si              ; Move to the next secret character (1/2) AH
Code_Part_2:7D0F                 inc     si              ; Move to the next secret character (1/2) AL
Code_Part_2:7D10                 loop    l_7CFE_ComparaSecretoNextCharacter
Code_Part_2:7D12
Code_Part_2:7D12 l_7D12_FinBucleSonido:                  ; CODE XREF: f_7CF1_EjecutaBucleSonido+F↑j
Code_Part_2:7D12                 call    f_7D2F_ApagaPCSpeaker
Code_Part_2:7D15                 call    f_7C0C_Prompt
Code_Part_2:7D15 f_7CF1_EjecutaBucleSonido endp
```

Block 1 - Sound loop - xor code

Status

- We need two characters, typed by the user (first two typed)
- xor between the two characters typed and the 16 bytes stored
  - BD 20 | FA 3B | BD 20 | 45 23 | 72 27 | 09 24 | C9 21 | BD 20
- Output will be 16 bytes.
- These 16 bytes are sent in pairs to the PC speaker
- PC speaker plays 8 tones (16 bits, 8 times)
- BD 20 is three times,...
- Probably the key is the PC speaker output!

We need the two typed characters to decode the datapairs!
We need think about the PC speaker output codification!!

Block 1 - Sound loop - PC Speaker

- Computer PC speaker is based in old Intel 8253/8254 chips
- These chips (PITs) uses 16 bits to create a beep as input.
- The PIT operates a frequency of 1.19318 MHz
- Output frequency is the PIT frequency divide by the AX value (16bit)
- For example, to beep with a frequency of 261.63 Hz (middle C on a piano keyboard), we need set 4560h in the AX register, because 1.119.180 MHz divide by 4560h is 261.63 Hz.

Wait! 'middle C on piano keyboard' Yes! we can check piano codes!

Block 1 - Sound loop - Brute force

- We can do bruteforce with the two characters input
- User can type ASCII values only between 32 and 128
- We can get the piano code and the frequencies with this code:

```
218 # This block is when the user type two characters
  1 for y in range(32, 128):
  2     i = hex(y)
  3     inp1 = [i, i, i, i, i, i, i, i]
  4     ret1 = hex_or(TOCHECK1, inp1) # TOCHECK1 = ["BD", "FA", "BD", "45", "72", "09", "C9", "BD"]
  5
  6     for x in range(32, 128):
  7         j = hex(x)
  8         inp2 = [j, j, j, j, j, j, j, j]
  9         ret2 = hex_or(TOCHECK2, inp2) # TOCHECK2 = ["20", "3B", "20", "23", "27", "24", "21", "20"]
 10
 11         fres = []
 12         ltrs = []
 13         for k in range(8):
 14             # Careful here, sort matter!
 15             hexfreq = [ret2[k], ret1[k]]
 16
 17             myfreq = get_frecuency(hexfreq)
 18             fres.append(myfreq)
 19             ltrs.append(get_note(myfreq))
 20
 21         print("{} {}  {} {}".format(chr(y), chr(x), ltrs, fres))
NORMAL  hexpass4.py | +
```

Block 1 - Brute force outputs

- There are 9.312 outputs in the brute force attack.
- Most of the outputs are not an exact piano key frequency (Printed as '')

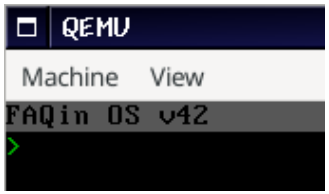
```
kix@inle: ~/src/nfc/faqin/script
  \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [6313.12, 166.6, 6313.12, 1425.54, 626.01, 1155.06, 2610.9, 6313.12]
! \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [2681.3, 172.77, 2681.3, 2053.67, 723.14, 925.66, 5936.22, 2681.3]
" \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [1702.11, 179.43, 1702.11, 3671.32, 855.94, 772.28, 1231.35, 1702.11]
# \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [1246.79, 186.61, 1246.79, 17292.46, 1048.49, 662.51, 1673.46, 1246.79]
$ \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [983.66, 145.76, 983.66, 641.15, 1352.81, 132575.56, 805.66, 983.66]
% \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [812.24, 150.46, 812.24, 743.41, 1906.04, 4502.57, 974.02, 812.24]
& \0 ['.', 'D#3/Eb3', '.', '.', '.', '.', '.', '.'] [691.7, 155.48, 691.7, 884.49, 3224.81, 2290.17, 598.69, 691.7]
' \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [602.31, 160.85, 602.31, 1091.66, 10466.49, 1535.62, 686.92, 602.31]
( \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [533.38, 233.32, 533.38, 413.58, 301.77, 387.27, 476.32, 533.38]
) \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [478.61, 245.61, 478.61, 453.85, 322.66, 357.56, 530.54, 478.61]
* \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [434.04, 259.27, 434.04, 502.82, 346.65, 332.08, 395.49, 434.04]
+ \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [397.06, 274.55, 397.06, 563.62, 374.51, 310.0, 432.16, 397.06]
, \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [365.89, 194.39, 365.89, 305.24, 407.23, 580.06, 338.11, 365.89]
- \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [339.26, 202.85, 339.26, 326.63, 446.22, 515.86, 364.55, 339.26]
. \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [316.24, 212.08, 316.24, 351.25, 493.46, 464.45, 295.27, 316.24]
/ \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [296.15, 222.19, 296.15, 379.87, 551.89, 422.36, 315.24, 296.15]
0 \0 ['.', '.', '.', '.', '.', '.', '.', '.'] [278.46, 389.17, 278.46, 241.88, 198.8, 232.63, 262.06, 278.46]
1 \0 ['.', '.', '.', '.', 'G#3/Ab3', '.', '.', '.'] [262.76, 424.62, 262.76, 255.12, 207.65, 221.57, 277.68, 262.76]
```

There are only one key with all frequencies...

The flag is...

```
kix@inle: ~/src/nfc/faqin/script
kix@inle:~/src/nfc/faqin/script$ python3 hexpass4.py | grep -v "'\.'"
* *  ['A4', 'C4', 'A4', 'B4', 'F4', 'E4', 'G4', 'A4'] [440.13, 261.66, 440.13, 494.07, 349.29, 329.7, 392.11, 440.13]
kix@inle:~/src/nfc/faqin/script$
```

The input is: ** (42d 42d) -> 42 is the FAQin OS version!



The flag is: ACABFEGA

Number 42

Popular culture [edit]

The Hitchhiker's Guide to the Galaxy [edit]

The Answer to the Ultimate Question of Life, The Universe, and Everything.

The number 42 is, in *The Hitchhiker's Guide to the Galaxy* by Douglas Adams, the "Answer to the Ultimate Question of Life, the Universe, and Everything", calculated by an enormous supercomputer named Deep Thought over a period of 7.5 million years. Unfortunately, no one knows what the question is. Thus, to calculate the Ultimate Question, a special computer the size of a small planet was built from organic components and named "Earth". The Ultimate Question "What do you get when you multiply six by nine"[25] was found by Arthur Dent and Ford Prefect in the second book of the series, *The Restaurant at the End of the Universe*. This appeared first in the radio play and later in the novelization of *The Hitchhiker's Guide to the Galaxy*. The fact that Adams named the episodes of the radio play "fits", the same archaic title for a chapter or section used by Lewis Carroll in *The Hunting of the Snark*, suggests that Adams was influenced by Carroll's fascination with and frequent use of the number. The fourth book in the series, the novel *So Long, and Thanks for All the Fish*, contains 42 chapters. According to the novel *Mostly Harmless*, 42 is the street address of Stavromula Beta. In 1994 Adams created the *42 Puzzle*, a game based on the number 42.

The book *42: Douglas Adams' Amazingly Accurate Answer to Life, the Universe and Everything* (2011)[26] examines Adams' choice of the number 42, and contains a compendium of some instances of the number in science, popular culture, and humour.

https://en.wikipedia.org/wiki/42_(number)

Remember the important keys...

Hello world Lara!

Questions?

Questions?

Thanks a lot to @bitsniper for this amazing game

Hello world Martín!