SYNACKTIV

Resolution of the Challenge THCon

20/04/2023

$ whoami

- **Fabien PERIGAUD (@0xf4b)**
- **Reverse Engineering tech lead**
  - Synacktiv
- **VR, Exploit, RE**
- **Likes challenges :)**

How did it start?

How did it start?

How did it start?

How did it start? (2)

RULEZ

One should seek mail addresses of the following form: **<large-random-string-flag>@m0rgan.net.**

Please report your achievements by sending a mail to the given addresses. Make sure to use a consistent from fiel
allowing your identification throughout the challenge.

The first challenger reporting the last flag will be considered as the winner, hence scalping fame and rewards.
Nevertheless the remaining most proficient participents will also be rewarded with valuable swag.

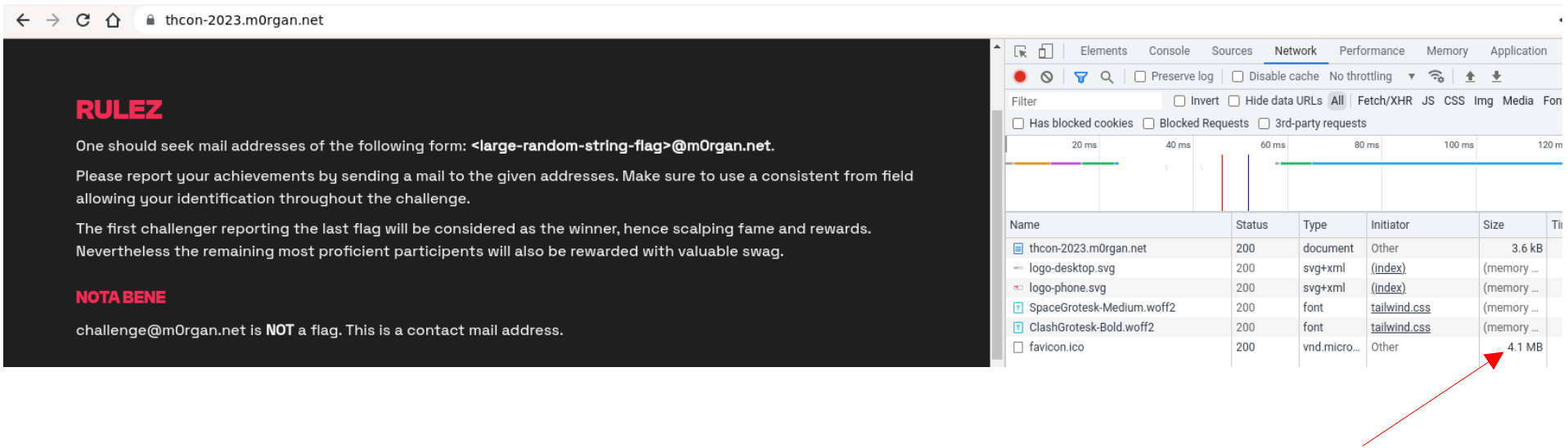
NOTA BENE

challenge@m0rgan.net is **NOT** a flag. This is a contact mail address.

GETTIN' STARTED

Something's hidden here. Good luck.

Step 0


thcon-2023.m0rgan.net
RULEZ
One should seek mail addresses of the following form: <large-random-string-flag>@m0rgan.net.
Please report your achievements by sending a mail to the given addresses. Make sure to use a consistent from field allowing your identification throughout the challenge.
The first challenger reporting the last flag will be considered as the winner, hence scalping fame and rewards. Nevertheless the remaining most proficient participents will also be rewarded with valuable swag.
NOTA BENE
challenge@m0rgan.net is NOT a flag. This is a contact mail address.
Elements
Console
Sources
Network
Performance
Memory
Application
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Has blocked cookies
Blocked Requests
3rd-party requests
20 ms
40 ms
60 ms
80 ms
100 ms
Name
Status
Type
Initiator
Size
thcon-2023.m0rgan.net 200 document Other 3.6 kB
logo-desktop.svg 200 svg+xml (index) (memory ...
logo-phone.svg 200 svg+xml (index) (memory ...
SpaceGrotesk-Medium.woff2 200 font tailwind.css (memory ...
ClashGrotesk-Bold.woff2 200 font tailwind.css (memory ...
favicon.ico 200 vnd.micro... Other 4.1 MB

Step 0

```
$ binwalk favicon.ico

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Linux EXT filesystem, blocks count: 4000,
image size: 4096000, rev 1.0, ext2 filesystem data, UUID=d38aeea5-4f70-
41f1-899a-011ff547f547
```

Step 0

```
$ binwalk favicon.ico

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Linux EXT filesystem, blocks count: 4000,
image size: 4096000, rev 1.0, ext2 filesystem data, UUID=d38aeea5-4f70-
41f1-899a-011ff547f547

$ fls favicon.ico | grep -i flag
r/r 741:  FLAG
$ icat favicon.ico 741
efs1csztiwc2aqentuzlxurfceeelv41bv1tzor4gf2v9778wxf9jztx6zxi@m0rgan.net
```

Step 1

```
$ fls -d favicon.ico
d/d * 742(realloc): thcon-2023
```

Step 1

```
$ fls -d favicon.ico
d/d * 742(realloc): thcon-2023

$ fls favicon.ico 742
r/r 743:  THCON-2023
r/b 744:  step2.tgz
r/r 244:  despair.nohope
```

Step 1

```
$ fls -d favicon.ico
d/d * 742(realloc): thcon-2023

$ fls favicon.ico 742
r/r 743:  THCON-2023
r/b 744:  step2.tgz
r/r 244:  despair.nohope

$ icat favicon.ico 743
its@tr@p
```

Step 1

```
$ fls -d favicon.ico
d/d * 742(realloc): thcon-2023

$ fls favicon.ico 742
r/r 743:  THCON-2023
r/b 744:  step2.tgz
r/r 244:  despair.nohope

$ icat favicon.ico 743
its@tr@p

$ icat favicon.ico 744 > step2.tgz
Error in metadata structure (unix: Indirect block address too large:
659103860)
```

Step 1 - inodes

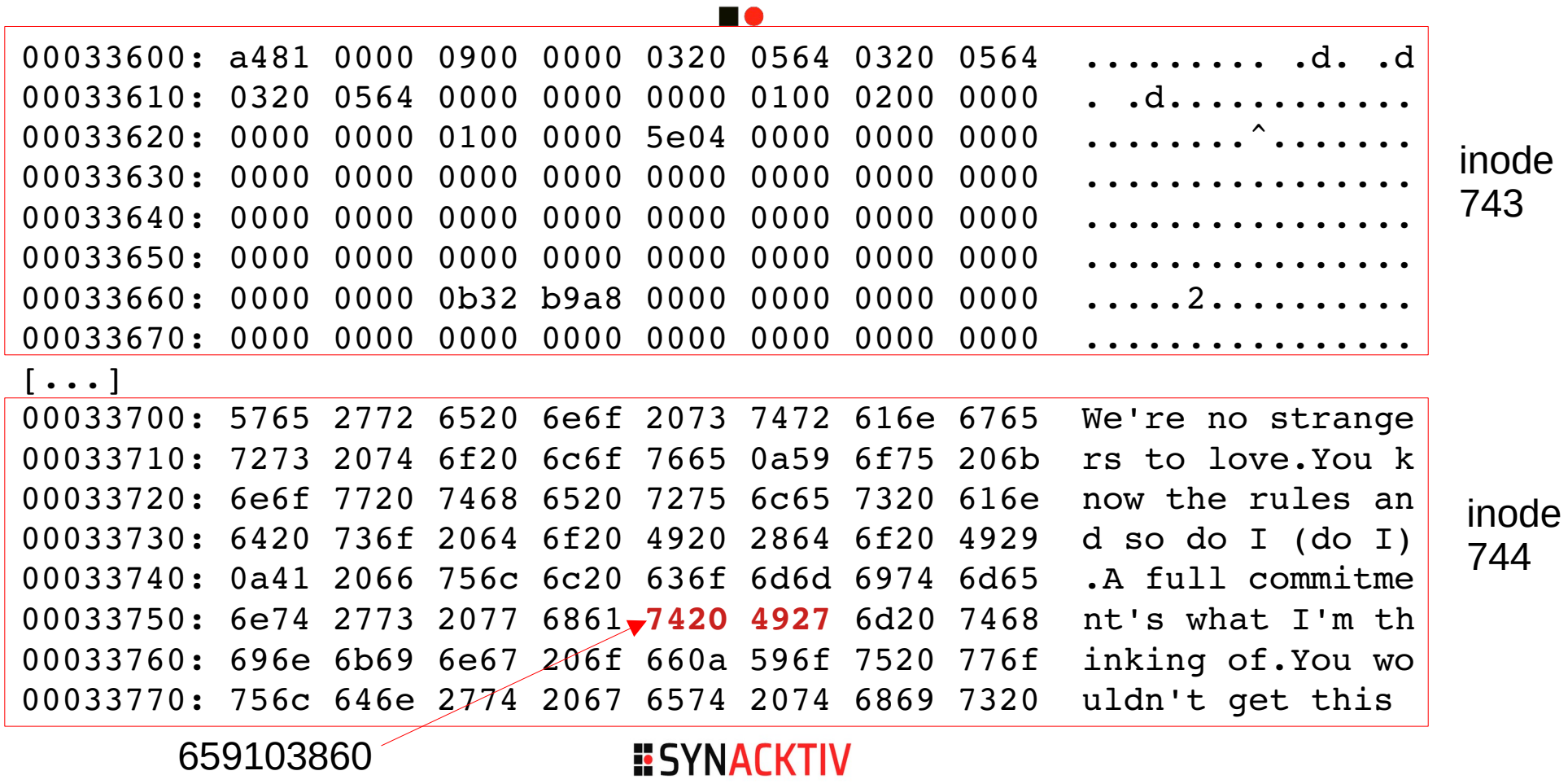
```
$ istat favicon.ico 742
inode: 742
Allocated
Group: 0
Generation Id: 1672958236
uid / gid: 0 / 0
mode: drwxr-xr-x
size: 1024
num of links: 2

Inode Times:
Accessed:  2023-03-06 00:04:35 (CET)
File Modified: 2023-03-06 00:04:35
(CET)
Inode Modified:    2023-03-06 00:04:35
(CET)

Direct Blocks:
1116
```

```
$ istat favicon.ico 743
inode: 743
Allocated
Group: 0
Generation Id: 2830709259
uid / gid: 0 / 0
mode: rrw-r--r--
size: 9
num of links: 1

Inode Times:
Accessed:  2023-03-06 00:04:35 (CET)
File Modified: 2023-03-06 00:04:35
(CET)
Inode Modified:    2023-03-06 00:04:35
(CET)

Direct Blocks:
1118
```

Step 1 - inodes

```
$ istat favicon.ico 744
inode: 744
Allocated
Group: 0
Generation Id: 1864394606
uid / gid: 1952805415 / 1948280182
mode: br-x-wSrwx
Device Major: 117   Minor: 114
[...]

Direct Blocks:

Error reading file:  Error in metadata structure (unix: Indirect block
address too large: 659103860)
```

Step 1 - inodes



SYNACKTIV

Step 1 - inodes

2023-03-06 00:04:35 (CET)

```
00033600: a481 0000 0900 0000 0320 0564 0320 0564  ......... .d. .d
00033610: 0320 0564 0000 0000 0000 0100 0200 0000  . .d............
00033620: 0000 0000 0100 0000 5e04 0000 0000 0000  ........^.......
00033630: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00033640: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00033650: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00033660: 0000 0000 0b32 b9a8 0000 0000 0000 0000  .....2..........
00033670: 0000 0000 0000 0000 0000 0000 0000 0000  ................
[...]
00033700: 5765 2772 6520 6e6f 2073 7472 616e 6765  We're no strange
00033710: 7273 2074 6f20 6c6f 7665 0a59 6f75 206b  rs to love.You k
00033720: 6e6f 7720 7468 6520 7275 6c65 7320 616e  now the rules an
00033730: 6420 736f 2064 6f20 4920 2864 6f20 4929  d so do I (do I)
00033740: 0a41 2066 756c 6c20 636f 6d6d 6974 6d65  .A full commitme
00033750: 6e74 2773 2077 6861 7420 4927 6d20 7468  nt's what I'm th
00033760: 696e 6b69 6e67 206f 660a 596f 7520 776f  inking of.You wo
00033770: 756c 646e 2774 2067 6574 2074 6869 7320  uldn't get this
```

inode 743

inode 744

659103860

SYNACKTIV

Step 1 – find the date

```
00197400: a481 0000 630f 1900 0320 0564 0320 0564  ....c.... .d. .d
00197410: 0320 0564 0000 0000 0000 0100 980c 0000  . .d............
00197420: 0000 0000 0100 0000 5702 0000 5802 0000  ........W...X...
00197430: 5902 0000 5a02 0000 5b02 0000 5c02 0000  Y...Z...[...\...
00197440: 5d02 0000 5e02 0000 5f02 0000 6002 0000  ]...^..._...`...
00197450: 6102 0000 6202 0000 550c 0000 490e 0000  a...b...U...I...
00197460: 0000 0000 15af 8c59 0000 0000 0000 0000  .......Y........
00197470: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Step 1 – find the date

```
00197400: a481 0000 630f 1900 0320 0564 0320 0564  ....c.... .d. .d
00197410: 0320 0564 0000 0000 0000 0100 980c 0000  . .d............
00197420: 0000 0000 0100 0000 5702 0000 5802 0000  ........W...X...
00197430: 5902 0000 5a02 0000 5b02 0000 5c02 0000  Y...Z...[...\...
00197440: 5d02 0000 5e02 0000 5f02 0000 6002 0000  ]...^..._...`...
00197450: 6102 0000 6202 0000 550c 0000 490e 0000  a...b...U...I...
00197460: 0000 0000 15af 8c59 0000 0000 0000 0000  .......Y........
00197470: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

- **Replace previously found data at 0x33700!**

Step 1 - solved

```
$ icat fav.ico 744 > step2.tgz

$ tar tvzf step2.tgz
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step2/
-rwxr-xr-x bmorgan/users 11143720 2023-03-06 00:04 step2/nop
-rw-r--r-- bmorgan/users       72 2023-03-06 00:04 step2/zFLAG

$ cat zFLAG
aun9dwzu6wxd7spcg5k59pj6wnanmy0kasohykz5q9qbwiyocdyl26axazwg@m0r
gan.net
```

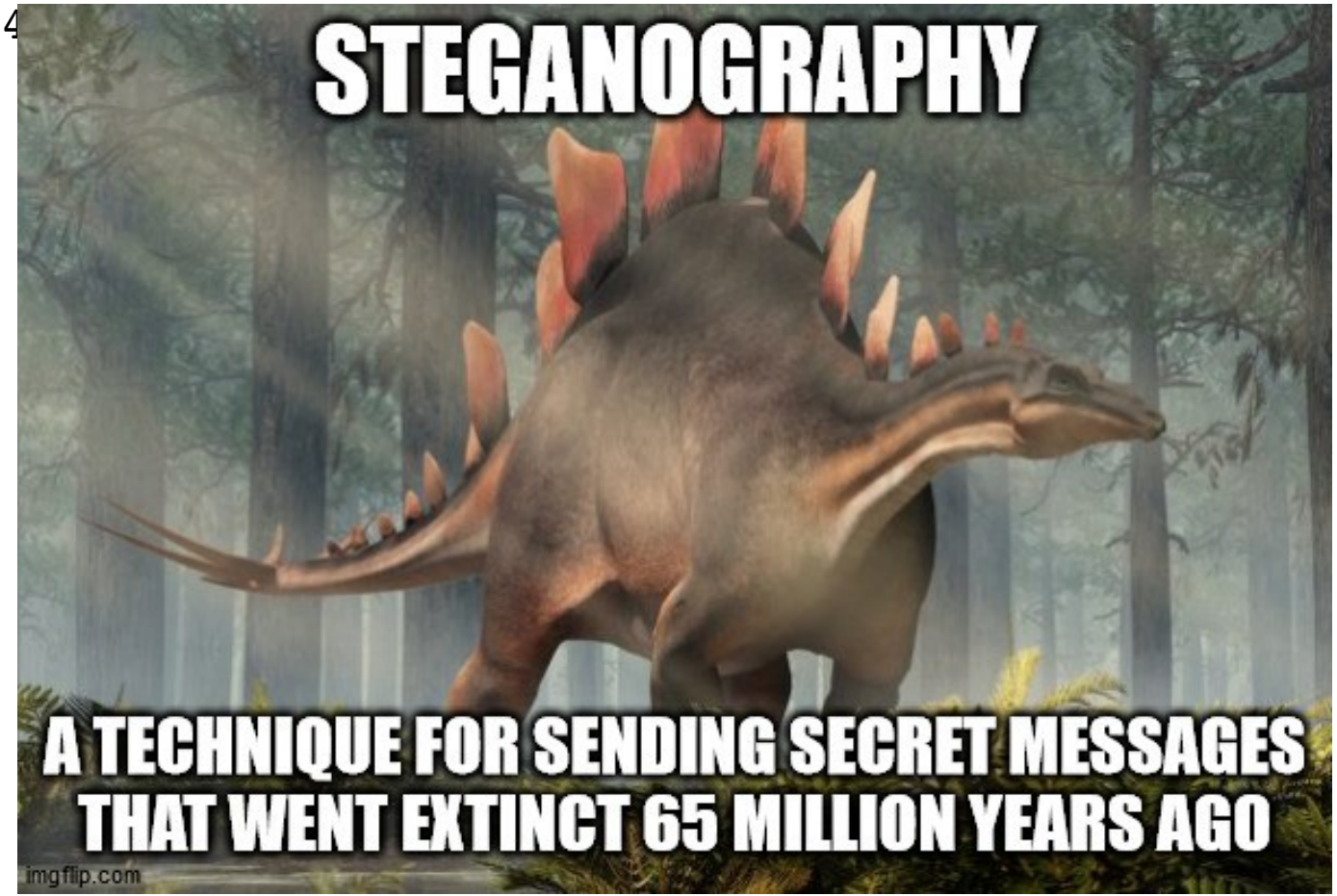
Step 2

```
$ file nop
nop: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
statically linked, not stripped
```

```
.text:0000000000401000
.text:0000000000401000                          public _start
.text:0000000000401000 _start                   proc near               ; DATA XREF: LOAD:0000000000400018↑o
.text:0000000000401000                                                  ; LOAD:0000000000400088↑o
.text:0000000000401000 E8 15 00 00 00           call    nopnop
.text:0000000000401005 E8 35 00 00 00           call    nojoke
.text:000000000040100A 48 C7 C0 3C 00 00 00     mov     rax, 3Ch ; '<'
.text:0000000000401011 48 C7 C7 00 00 00 00     mov     rdi, 0          ; error_code
.text:0000000000401018 0F 05                    syscall                 ; LINUX - sys_exit
.text:0000000000401018 _start                   endp
.text:0000000000401018
.text:000000000040101A
.text:000000000040101A                          ; =============== S U B R O U T I N E =======================================
.text:000000000040101A
.text:000000000040101A
.text:000000000040101A nopnop                   proc near               ; CODE XREF: _start↑p
.text:000000000040101A 90                       nop
.text:000000000040101B 66 90                    xchg    ax, ax
.text:000000000040101D 0F 1F 00                 nop     dword ptr [rax]
.text:0000000000401020 0F 1F 40 00              nop     dword ptr [rax+00h]
.text:0000000000401024 0F 1F 44 00 00           nop     dword ptr [rax+rax+00h]
.text:0000000000401029 66 0F 1F 44 00 00        nop     word ptr [rax+rax+00h]
.text:000000000040102F 0F 1F 80 00 00 00 00     nop     dword ptr [rax+00000000h]
.text:0000000000401036 0F 1F 84 00 00 00 00 00  nop     dword ptr [rax+rax+00000000h]
.text:000000000040103E C3                       retn
.text:000000000040103E nopnop                   endp
.text:000000000040103E
.text:000000000040103F
.text:000000000040103F                          ; =============== S U B R O U T I N E =======================================
.text:000000000040103F
.text:000000000040103F
.text:000000000040103F nojoke                   proc near               ; CODE XREF: _start+5↑p
.text:000000000040103F 0F 1F 84 00 00 00 00 00  nop     dword ptr [rax+rax+00000000h]
.text:0000000000401047 0F 1F 40 00              nop     dword ptr [rax+00h]
.text:000000000040104B 0F 1F 44 00 00           nop     dword ptr [rax+rax+00h]
.text:0000000000401050 66 0F 1F 44 00 00        nop     word ptr [rax+rax+00h]
.text:0000000000401056 90                       nop
.text:0000000000401057 66 90                    xchg    ax, ax
.text:0000000000401059 0F 1F 00                 nop     dword ptr [rax]
.text:000000000040105C 90                       nop
.text:000000000040105D 90                       nop
.text:000000000040105E 90                       nop
.text:000000000040105F 90                       nop
.text:0000000000401060 90                       nop
```

Step 2

```
$ file nop
nop: ELF 64
statically
```



Step 2 – no RE, just stegano

- **8 different NOP encodings**
- **Used as a way to encode 3 bits of data**
  - 1 byte → 000
  - 2 bytes → 001
  - 3 bytes → 010
  - …
  - 8 bytes → 111

Step 2 – solve

```
$ objdump --disassemble=nojoke --insn-width=8 nop|head
000000000040103f <nojoke>:
  40103f:    0f 1f 84 00 00 00 00 00   nopl   0x0(%rax,%rax,1)
  401047:    0f 1f 40 00               nopl   0x0(%rax)
  40104b:    0f 1f 44 00 00            nopl   0x0(%rax,%rax,1)
```

Step 2 – solve

```
$ objdump --disassemble=nojoke --insn-width=8 nop|head
000000000040103f <nojoke>:
  40103f:    0f 1f 84 00 00 00 00 00   nopl   0x0(%rax,%rax,1)
  401047:    0f 1f 40 00               nopl   0x0(%rax)
  40104b:    0f 1f 44 00 00            nopl   0x0(%rax,%rax,1)

$ objdump --disassemble=nojoke --insn-width=8 nop | grep -P '\t' | awk -F '\t' '{print
$2}' > out

$ head out
0f 1f 84 00 00 00 00 00
0f 1f 40 00
0f 1f 44 00 00
66 0f 1f 44 00 00
90
66 90
0f 1f 00
90
90
90
```

SYNACKTIV

Step 2 – solve

```
$ objdump --disassemble=nojoke --insn-width=8 nop|head
000000000040103f <nojoke>:
  40103f:    0f 1f 84 00 00 00 00 00   nopl   0x0(%rax,%rax,1)
  401047:    0f 1f 40 00               nopl   0x0(%rax)
  40104b:    0f 1f 44 00 00            nopl   0x0(%rax,%rax,1)

$ objdump --disassemble=nojoke --insn-width=8 nop | grep -P '\t' | awk -F '\t' '{print $2}' > out

$ head out
0f 1f 84 00 00 00 00 00
0f 1f 40 00
0f 1f 44 00 00
66 0f 1f 44 00 00
90
66 90
0f 1f 00
90
90
90
```

$ **python solve_step2.py out**

```python
final=0
cnt=0
for x in f.readlines():
    if x.startswith(b"c3"):
        break
    x=x.replace(b" ",b"").replace(b"\n",b"")
    y=len(x)//2 - 1
    final = (final) | (y<<(3*cnt))
    cnt+=1
    if (cnt == 8):
        cnt=0
        v=pack("<L",final)
        wf.write(v[:3])
        final=0
```

SYNACKTIV

Step 2 – solve

```
$ tar tvzf step3
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step3/
-rw-r--r-- bmorgan/users 32873 2023-03-06 00:04 step3/chal.bin
-rw-r--r-- bmorgan/users    72 2023-03-06 00:04 step3/FLAG
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step3/var/
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step3/var/ez/
-rw-r--r-- bmorgan/users   120 2023-03-06 00:04 step3/var/ez/README
-rw-r--r-- bmorgan/users 922608 2023-03-06 00:04 step3/var/ez/a.e
-rwxr-xr-x bmorgan/users  16608 2023-03-06 00:04 step3/emu

$ cat step3/FLAG
jwrwwijfo58vsj8okmfmyr0lx1lbadpeez4dxdctqvetjwna6dvajqwjpbyk@m0rgan.net
```

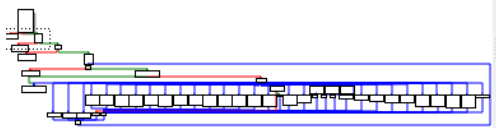
Step 3

- Emulator + “chal.bin”
- 0x150 bytes of code

```
$ xxd -a chal.bin
00000000: 0000 0dd0 0010 00f3 0000 00d0 0000 00c3  ................
00000010: 0200 0ad0 0a0d 0d02 0e0d 00e3 2080 00d0  ............ ...
00000020: 0000 01d0 0000 02d0 0000 00c2 0000 08d1  ................
00000030: 2080 00d0 0000 01d0 0100 08e3 2080 00d0   ........... ...
00000040: 4102 01d0 a401 02d0 0000 00c2 0000 09d1  A...............
00000050: 0100 00d0 2b80 01d0 1300 02d0 0000 00c1  ....+...........
00000060: 0000 00d0 0080 01d0 1000 02d0 0000 00c0  ................
00000070: 0000 0bd1 0100 00d0 0b00 0106 0198 00f1  ................
00000080: 0200 00d0 3f80 01d0 2a00 02d0 0000 00c1  ....?...*.......
00000090: 0100 00d0 0000 00c3 0008 00d1 1080 01d0  ................
000000a0: 000b 02d1 0000 00c0 0000 03d1 0003 00d1  ................
000000b0: 0000 01d0 0100 0002 0040 01f1 0404 040c  .........@......
000000c0: 0707 070c 1080 00d0 0004 05e4 0080 01d0  ................
000000d0: 0104 06e4 0005 00d1 0006 01d1 ff00 02d0  ................
000000e0: 0200 0006 0201 0106 0100 0002 0200 0006  ................
000000f0: 0007 0705 0005 00d1 0006 01d1 00ff 02d0  ................
00000100: 0200 0006 0201 0106 0100 0002 0200 0006  ................
00000110: 0007 0705 1080 00d0 0700 04e5 0200 00d0  ................
00000120: 0004 0401 0403 000a 00c0 00f1 0009 00d1  ................
00000130: 1080 01d0 0003 02d1 0000 00c1 0098 00f0  ................
00000140: 0d00 0ee2 0200 0ad0 0a0d 0d01 0000 0ef2  ................
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
*
00008020: 7661 722f 657a 2f61 2e65 0045 6e74 6572  var/ez/a.e.Enter
00008030: 2070 6173 7370 6872 6173 653a 200a 0054   passphrase: ..T
00008040: 6865 206b 6579 2073 697a 6520 6e65 6564  he key size need
00008050: 7320 746f 2062 6520 6120 6d75 6c74 6970  s to be a multip
00008060: 6c65 206f 6620 320a 00                   le of 2..
```

SYNACKTIV

Step 3 - VM



```
56  v22 = data;
57 LABEL_8:
58  while ( (pc & 3) == 0 )
59  {
60    current_insn = &data[(unsigned __int16)pc];
61    switch ( current_insn[3] )
62    {
63      case 0:
64        goto LABEL_44;
65      case 1:
66        regs[(unsigned __int8)current_insn[2]] = regs[(unsigned __int8)current_insn[1]]
67                                               + regs[(unsigned __int8)*current_insn];
68        goto LABEL_44;
69      case 2:
70        regs[(unsigned __int8)current_insn[2]] = regs[(unsigned __int8)current_insn[1]]
71                                               - regs[(unsigned __int8)*current_insn];
72        goto LABEL_44;
73      case 3:
74        regs[(unsigned __int8)current_insn[2]] = regs[(unsigned __int8)current_insn[1]] << regs[(unsigned __int8)*current_insn];
75        goto LABEL_44;
76      case 4:
77        regs[(unsigned __int8)current_insn[2]] = (int)regs[(unsigned __int8)current_insn[1]] >> regs[(unsigned __int8)*current_insn];
78        goto LABEL_44;
79      case 5:
80        regs[(unsigned __int8)current_insn[2]] = regs[(unsigned __int8)current_insn[1]] | regs[(unsigned __int8)*current_insn];
81        goto LABEL_44;
82      case 6:
83        regs[(unsigned __int8)current_insn[2]] = regs[(unsigned __int8)current_insn[1]] & regs[(unsigned __int8)*current_insn];
84        goto LABEL_44;
85      case 7:
86        regs[(unsigned __int8)current_insn[2]] = regs[(unsigned __int8)current_insn[1]] == regs[(unsigned __int8)*current_insn];
87        goto LABEL_44;
```

SYNACKTIV

Step 3 - VM

- **Quite simple 16 bits VM**
- **32 bits instructions**
- **Registers : PC, 13 GP, SP, LR**
- **Classical instructions:**
  - MOVs
  - ADD/SUB/AND/XOR/OR…
  - JCC/CALL/RET
- **Special instructions: open, read, write**

Step 3 – Disass.py

```
pc=0

while True:
    b = prog[pc:pc+4]
    ins=b[3]
    #print(b)
    if ins == 0:
        break
    elif ins == 0xD0:
        print("%04x: MOV R%d, 0x%04x" % (pc, b[2], (b[1]<<8 | b[0])))
    elif ins == 0xD1:
        print("%04x: MOV R%d, R%d" % (pc, b[2], b[1]))
    elif ins == 0xF3:
        print("%04x: CALL 0x%04x" % (pc, (b[2]<<8 | b[1])))
    elif ins == 0xF1:
        print("%04x: BEQ 0x%04x" % (pc, (b[2]<<8 | b[1])))
    elif ins == 0xF0:
        print("%04x: JMP 0x%04x" % (pc, (b[2]<<8 | b[1])))
    elif ins == 0xF2:
        print("%04x: BR R%d" % (pc, b[2]))
    elif ins == 0xC3:
        print("%04x: EXIT" % (pc))
    elif ins == 1:
        print("%04x: ADD R%d, R%d, R%d" % (pc, b[2], b[1], b[0]))
    elif ins == 2:
        print("%04x: SUB R%d, R%d, R%d" % (pc, b[2], b[1], b[0]))
    elif ins == 6:
        print("%04x: AND R%d, R%d, R%d" % (pc, b[2], b[1], b[0]))
    elif ins == 0xa:
        print("%04x: TST R%d, R%d < R%d" % (pc, b[2], b[1], b[0]))
    elif ins == 0xc:
        print("%04x: XOR R%d, R%d, R%d" % (pc, b[2], b[1], b[0]))
    elif ins == 0x5:
        print("%04x: OR R%d, R%d, R%d" % (pc, b[2], b[1], b[0]))
    elif ins == 0xe2:
        print("%04x: MOV R%d, [R%d + %x]" % (pc, b[2], b[0], b[1]))
    elif ins == 0xe3:
        print("%04x: MOV [R%d + %x], R%d" % (pc, b[1], b[2], b[0]))
    elif ins == 0xe4:
        print("%04x: MOV R%d, [R%d + R%d]" % (pc, b[2], b[0], b[1]))
    elif ins == 0xe5:
        print("%04x: MOV [R%d + R%d], R%d" % (pc, b[1], b[2], b[0]))
    elif ins == 0xC2:
        print("%04x: OPEN [R%d], R%d, R%d" % (pc, b[0], b[1], b[2]))
    elif ins == 0xC1:
        print("%04x: WRITE R%d, [R%d], R%d" % (pc, b[0], b[1], b[2]))
    elif ins == 0xC0:
        print("%04x: READ R%d, [R%d], R%d" % (pc, b[0], b[1], b[2]))
    else:
        print(b)
        print("[-] Unk ins %x" % ins)
        break

    pc += 4
```

Step 3 – Simple code

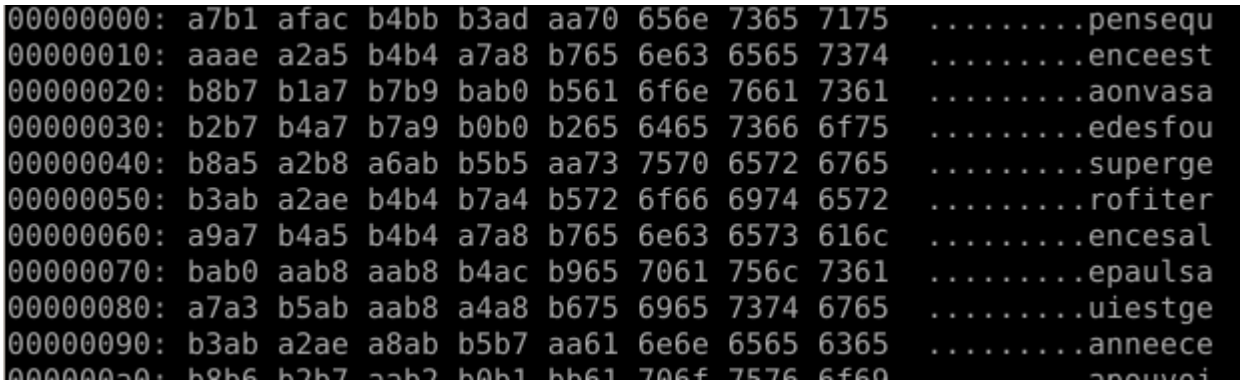
- Open var/ez/a.e
- Ask passphrase (16 chars)
- Applies ciph[i] – passphrase[i%16]

```
$ python x.py chal.bin
0000: MOV R13, 0x0000
0004: CALL 0x0010
0008: MOV R0, 0x0000
000c: EXIT
0010: MOV R10, 0x0002
0014: SUB R13, R13, R10
0018: MOV [R13 + 0], R14
001c: MOV R0, 0x8020
0020: MOV R1, 0x0000
0024: MOV R2, 0x0000
0028: OPEN [R0], R0, R0
002c: MOV R8, R0
0030: MOV R0, 0x8020
0034: MOV R1, 0x0000
0038: MOV [R0 + 8], R1
003c: MOV R0, 0x8020
0040: MOV R1, 0x0241
0044: MOV R2, 0x01a4
0048: OPEN [R0], R0, R0
004c: MOV R9, R0
[...]
```

Step 3 – Passphrase guess

```
00000540: d9c3 d5d1 dad9 6b82 f06a 6162 6566 6163  ......k..jabefac

000e13e0: 6469 8afe 972a 613b 7362 6162 6566 6163  di...*a;sbabefac
```

- Try passphrase “ babefac”
- Look for patterns



Step 3 – Passphrase guess

```
00000540: d9c3 d5d1 dad9 6b82 f06a 6162 6566 6163  ......k..jabefac
000e13e0: 6469 8afe 972a 613b 7362 6162 6566 6163  di...*a;sbabefac
```

- Try passphrase “ babefac”
- Look for patterns
- Guess remaining characters → “ebabefacebabefac”
- Output file has an ASCII header, followed by a GZIP file

Step 3 - solved

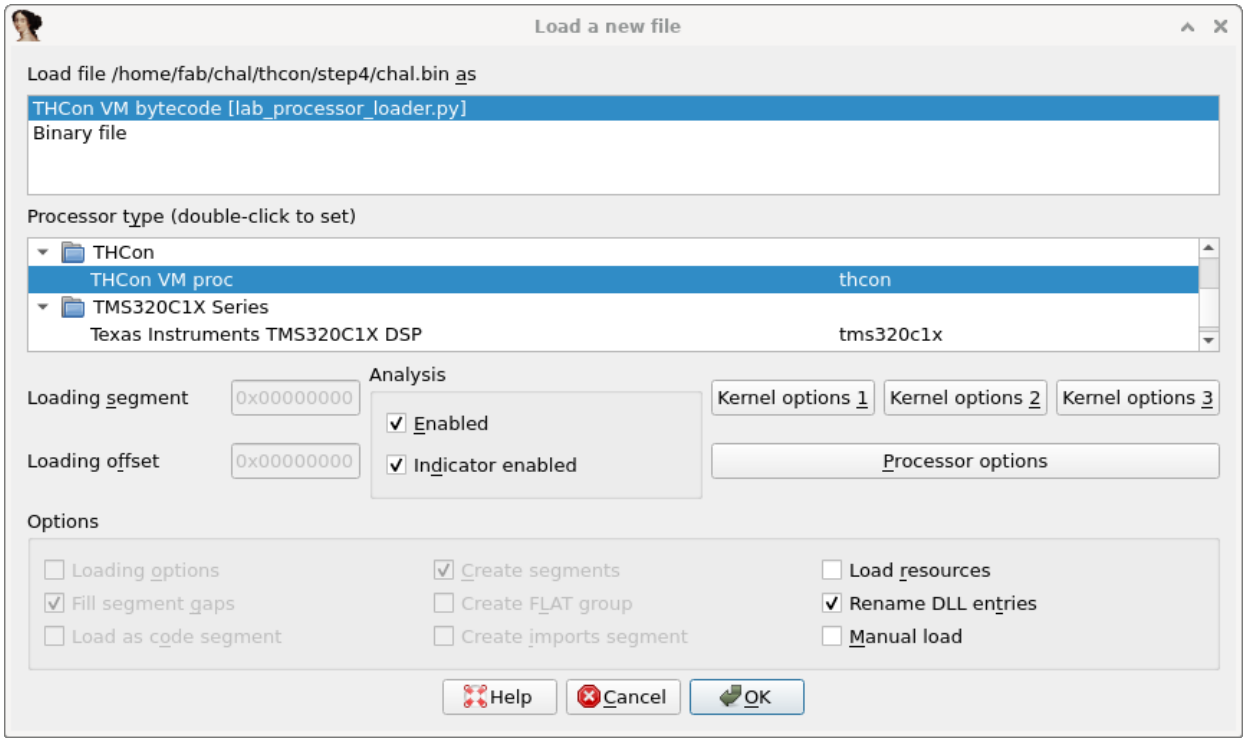
```
$ file step4
step4: gzip compressed data, from Unix, original size modulo 2^32 0

$ tar tvzf step4
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step4/
-rw-r--r-- bmorgan/users 33542 2023-03-06 00:04 step4/chal.bin
-rw-r--r-- bmorgan/users    72 2023-03-06 00:04 step4/FLAG
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step4/var/
drwxr-xr-x bmorgan/users     0 2023-03-06 00:04 step4/var/container/
-rw-r--r-- bmorgan/users 911765 2023-03-06 00:04 step4/var/container/file.bin.enc
-rw-r--r-- bmorgan/users     16 2023-03-06 00:04 step4/var/container/file.expected.hash
-rwxr-xr-x bmorgan/users  16608 2023-03-06 00:04 step4/emu

$ cat step4/FLAG
2kv0iayavqir6ybnfnipcryc6cr5r22zvmsnmys7eye6fgi1k1qjlndsxyeb@m0rgan.net
```

Step 4

- **Emulator + chal.bin**
  - Encrypted file and expected hash
- **Same VM ...**
  - … but 22kB of code
- **Simple disassembler is not enough**
  - We need a better tool

Step 4 – The elegant way

- **IDA Pro custom processor and loader (not during challenge)**



Step 4 – The elegant way

```
store   [sp], lr
mov     r0, 1
store   [sp+var_30], r0
mov     r0, 0x8000
store   [sp+var_32], r0
mov     r0, 0x15
store   [sp+var_34], r0
store   [sp+var_36], sp
mov     r0, 0x38
sub     sp, sp, r0
call    write
load    sp, [sp+0x38+var_36]
store   [sp+var_30], r0
load    r0, [sp+var_30]
store   [sp+var_A], r0
mov     r0, aVarContainerFi_0 ; "var/container/file.bin.enc"
store   [sp+var_30], r0
mov     r0, 0
store   [sp+var_32], r0
mov     r0, 0
store   [sp+var_34], r0
store   [sp+var_36], sp
mov     r0, 0x38
sub     sp, sp, r0
call    open
load    sp, [sp+0x38+var_36]
store   [sp+var_30], r0
load    r0, [sp+var_30]
store   [sp+var_2], r0
mov     r0, aVarContainerFi ; "var/container/file.bin"
store   [sp+var_30], r0
```

SYNACKTIV

Step 4 – The elegant way

```
store   [sp], lr
mov     r0, 1
store   [sp+var_30], r0
mov     r0, 0x8000
store   [sp+var_32], r0
mov     r0, 0x15
store   [sp+var_34], r0
store   [sp+var_36], sp
mov     r0, 0x38
sub     sp, sp, r0
call    write
load    sp, [sp+0x38+var_36]
store   [sp+var_30], r0
load    r0, [sp+var_30]
store   [sp+var_A], r0
mov     r0, aVarContainerFi_0 ; "var/container/file.bin.enc"
store   [sp+var_30], r0
mov     r0, 0
store   [sp+var_32], r0
mov     r0, 0
store   [sp+var_34], r0
store   [sp+var_36], sp
mov     r0, 0x38
sub     sp, sp, r0
call    open
load    sp, [sp+0x38+var_36]
store   [sp+var_30], r0
load    r0, [sp+var_30]
store   [sp+var_2], r0
mov     r0, aVarContainerFi ; "var/container/file.bin"
store   [sp+var_30], r0
```

SYNACKTIV

Step 4 – The dirty way

- **Writing a custom proc is complicated**
- **Writing a disassembler script in Python is easy…**

Step 4 – The dirty way

- Writing a custom proc is complicated
- Writing a disassembler script in Python is easy…

- Output valid ARM code instead of custom asm…
- … then assemble with as …
- … then open in IDA Pro!

Step 4 – The dirty way

```
$ python vm.py chal.bin
```

```
$ head x.asm
_label0000:
mov sp, #0xfffe
_label0004:
bl _label5010
_label0008:
bl _exit
_label000c:
strh lr, [sp, #0]

$ arm-none-eabi-as x.asm
$ arm-none-eabi-strip -w --strip-symbol='_label*' a.out

$ idapro a.out
```

Step 4 – The dirty way

```
STRH            LR, [SP,#arg_0]
MOV             R0, #1
STRH            R0, [SP,#var_30]
MOV             R0, #0x8000
STRH            R0, [SP,#var_32]
MOV             R0, #0x15
STRH            R0, [SP,#var_34]
STRH            SP, [SP,#var_36]
MOV             R0, #0x38 ; '8'
SUB             SP, SP, R0
BL              write
LDRH            SP, [SP,#0x38+var_36]
STRH            R0, [SP,#0x38+var_68]
LDRH            R0, [SP,#0x38+var_68]
STRH            R0, [SP,#0x38+var_42]
MOVW            R0, #aVarContainerFi ; "var/container/file.bin.enc"
STRH            R0, [SP,#0x38+var_68]
MOV             R0, #0
STRH            R0, [SP,#0x38+var_6A]
MOV             R0, #0
STRH            R0, [SP,#0x38+var_6C]
STRH            SP, [SP,#0x38+var_6E]
MOV             R0, #0x38 ; '8'
SUB             SP, SP, R0
BL              open
LDRH            SP, [SP,#0x70+var_6E]
STRH            R0, [SP,#0x70+var_A0]
LDRH            R0, [SP,#0x70+var_A0]
STRH            R0, [SP,#0x70+var_72]
MOVW            R0, #aVarContainerFi_2 ; "var/container/file.bin"
STRH            R0, [SP,#0x70+var_A0]
MOV             R0, #0x40 ; '@'
STRH            R0, [SP,#0x70+var_A2]
```

SYNACKTIV

Step 4 – The dirty way

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // r1
  int v4; // r2
  int v5; // r1
  int v6; // r1
  int v7; // r1
  int v8; // r2
  int v9; // r1
  int v10; // r2
  int v11; // r1
  int v12; // r1
  int v13; // r1

  write();
  open(56, v3, v4);
  open(56, 512, 577);
  write();
  if ( (unsigned __int16)read(56, v5) != 4 )
  {
    write();
    exit(52, v6);
  }
  unk_81E4 = unk_81E4 & 0xFF00 | (unsigned __int8)*(_WORD *)((unsigned int)&STACK[0x102B6] & 0xFFFE);
  *(_WORD *)((unsigned int)&tabX[4] & 0xFFFE) = *(_WORD *)((unsigned int)&tabX[4] & 0xFFFE) & 0xFF00 | (unsigned __int8)*(_WORD *)((unsigned int)&STACK[0x102B7] & 0xFFFE);
  *(_WORD *)((unsigned int)&tabX[8] & 0xFFFE) = *(_WORD *)((unsigned int)&tabX[8] & 0xFFFE) & 0xFF00 | (unsigned __int8)*(_WORD *)((unsigned int)&STACK[0x102B8] & 0xFFFE);
  *(_WORD *)((unsigned int)&tabX[12] & 0xFFFE) = *(_WORD *)((unsigned int)&tabX[12] & 0xFFFE) & 0xFF00 | (unsigned __int8)*(_WORD *)((unsigned int)&STACK[0x102B9] & 0xFFFE);
```

SYNACKTIV

Step 4

- **Hex-Rays not so useful...**
  - … but disassembly is sufficient!
- **Ask a 4 alpha characters passphrase**
- **“Derivation” to produce an AES key**
- **Decrypt file in AES-128-CTR**
- **Pad output (PKCS#7)**
- **Encrypt in AES-128-CBC**
- **Check if last 16 bytes match the expected “hash”**

Step 4 - Bruteforce

- **Bruteforce can be done**
  - 26**4 possible “keys” (twice if we try lower then upper)
  - Each try requires a full file decryption then encryption
  - Test all lower alpha characters → correct key found in ~5min \o/

Step 4 - Bruteforce

- **Bruteforce can be done**
  - 26**4 possible “keys” (twice if we try lower then upper)
  - Each try requires a full file decryption then encryption
  - Test all lower alpha characters → correct key found in ~5min \o/

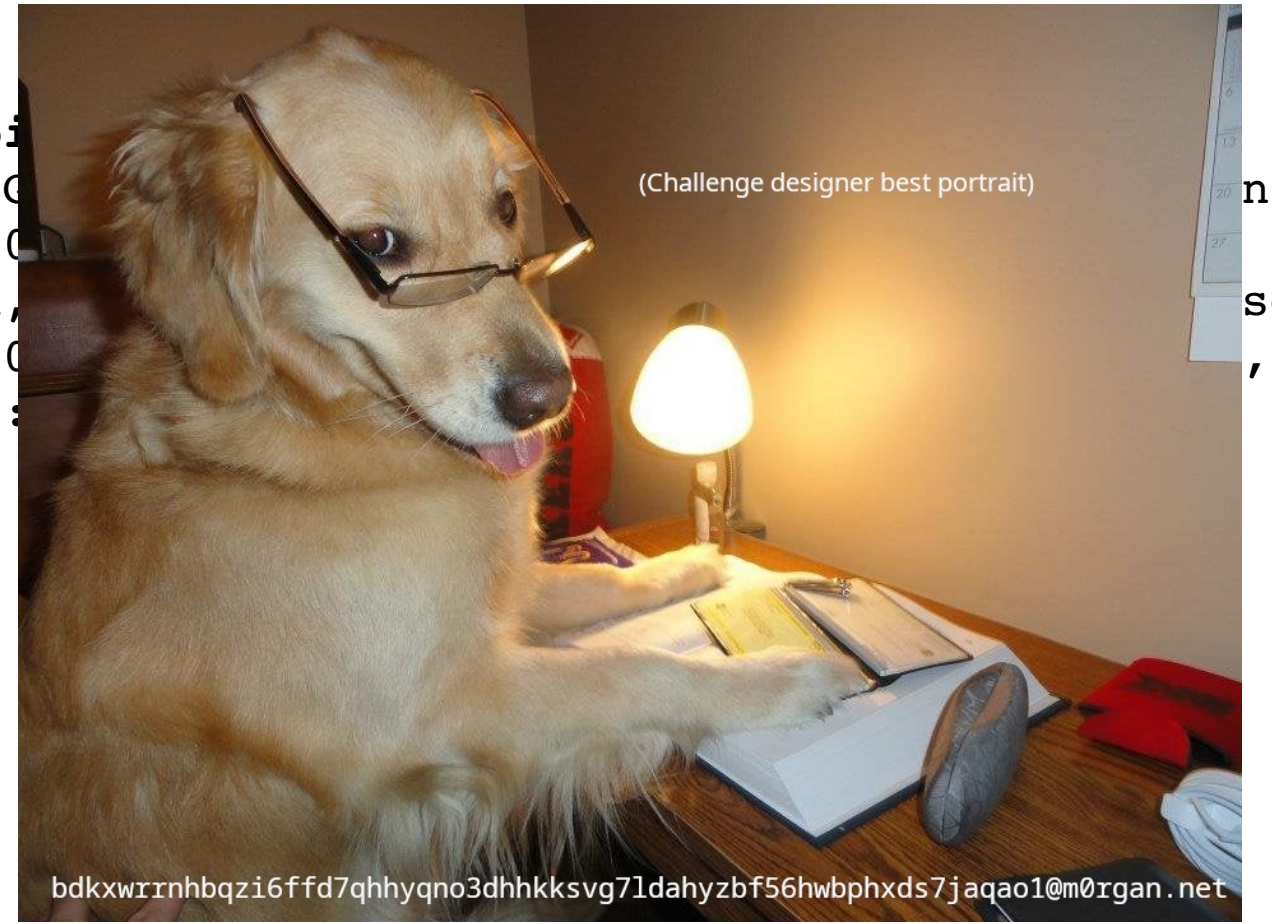
```
$ time python break_alpha_low.py var/container/file.bin.enc var/container/file.expected.hash
Trying aaXX
Trying abXX
[...]
Trying thXX
Trying tiXX
[+] FOUND tisa

real 4m32.248s
```

Final file

```
$ file file.bin
file.bin: JPEG image data, JFIF standard 1.01, resolution (DPI),
density 300x300, segment length 16, Exif Standard: [TIFF image data,
little-endian, direntries=7, orientation=upper-left, xresolution=98,
yresolution=106, resolutionunit=2, software=GIMP 2.10.34,
datetime=2023:03:07 02:57:10], progressive, precision 8, 960x720,
components 3
```

Final file

```
$ file file.bi
file.bin: JPEG                                    n (DPI),
density 300x30                                     image data,
little-endian,                                    solution=98,
yresolution=10                                    ,
datetime=2023:                                     960x720,
components 3
```



https://www.linkedin.com/company/synacktiv
https://twitter.com/synacktiv
Nos publications sur : https://synacktiv.com