Solution Challenge

Old Memories
Steps

✔ 9 Steps
  - 5 « ordinaures »
  - 4 « technical » steps

✔ Ordinoaure steps separated by technical tasks
Step 1 - Frenchy

✓ Thomson MO5 (RAM cartridge helps)
Step 2 – Before the web

✓ We’re given a domain name and a port (70)
✓ 70/TCP == Gopher

$ nc 0lsSgLqAGPWi0pvinvP.grehack.fr 70

/  
OCE-22.txt  //CWE-22.txt  0lsSgLqAGPWi0pvinvP.grehack.fr 70
IGH16_logo.png  //GH16_logo.png  0lsSgLqAGPWi0pvinvP.grehack.fr 70
IGH16_logo_black.png  //GH16_logo_black.png  0lsSgLqAGPWi0pvinvP.grehack.fr 70
IGH16_logo_txt.png  //GH16_logo_txt.png  0lsSgLqAGPWi0pvinvP.grehack.fr 70
IGH16_logo_txt_black.png  //GH16_logo_txt_black.png  0lsSgLqAGPWi0pvinvP.grehack.fr 70
1camelid  //camelid  0lsSgLqAGPWi0pvinvP.grehack.fr 70
1passwd(5)_history  //passwd(5)_history  0lsSgLqAGPWi0pvinvP.grehack.fr 70
1spec  //spec  0lsSgLqAGPWi0pvinvP.grehack.fr 70

$ nc 0lsSgLqAGPWi0pvinvP.grehack.fr 70

//CWE-22.txt

# CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal').
## Description Summary
Step 2 – Before the web

✓ Path traversal

$ nc O1sSgD1qAGPWi0pvinvP.grehack.fr 70
/./../etc/passwd
root:*:0:0:Charlie &:/root:/bin/ksh
grehack:*:2016:2016:Thomomys bottae:/GreHack2016:/sbin/nologin
Step 3 – Hartmut Esslinger

✔ Apple IIc
  - Hartmut Esslinger was the designer
Step 4 – Welcome to 1985 forensic

- We’re given an OVA file containing an OVF and a VMDK
- VMDK can be converted to RAW using qemu-img
- Image contains a DOS
Step 4 – Welcome to 1985 forensic

✓ Sleuthkit FTW!

$ fls -o63 -r iggy.raw |grep -i flag
+ r/r * 6788: _ILLFLAG

✓ « _ » replaces the first character of a deleted file on FAT16

✓ Looks like a flag 😊

$ icat -o63 iggy.raw 6788|xxd|head
00000000: 4461 6e4d d002 e001 2600 3000 0500 0600 DanM....&.0.....
00000010: d002 e001 e400 f000 2b2c 01ff ff80 01ff ..........+,,.....

✓ Google => MS Paint 1.0 Image format (.MSP)
Step 4 – Welcome to 1985 forensic

✓ PIL MsplImagePlugin.py 😊
✓ Convert MSP to any format

20AF146596BEE7F086CCDB568211CD37

FLAG :-}
**Step 5 - MC6803P**

- Matra Hachette Alice
  - Uses the MC6803P processor
Step 6 – WTFBBQRCODE?!!

✓ Archive contains an encrypted ZIP file, an AES encrypted file, and a PNG file
✓ Definitly not a QRCODE 😊
Step 6 – WTFBBQRCODE?!!

✓ Seems to be slices of barcodes
✓ Now what?
  ▪ Read EAN-13 standard
  ▪ Write an EAN-13 decoder
  ▪ Apply the decoder on each pixel line
  ▪ ???
  ▪ Profit!
Step 6 – WTFBBQRCODE?!!

✅ There are only 6 different lines

✅ Only one can be decoded without error
  - Decodes to 4374335913103
  - Allows to open ZIP file!
  - Contains the 6 full barcodes (easier to handle)
  - Also tells that AES key is combination of all barcodes
    - Even gives hashes (bcrypt) for each barcode
Step 6 – WTFBBQRCODE?!!

✓ Seems the 5 remaining barcodes need to be fixed
  ▪ Barcode #0 : OK
  ▪ Barcode #1 : bits should be rotated
  ▪ Barcode #2 : apply not to 2\textsuperscript{nd} part bits
  ▪ Barcode #3 : fix first digit
  ▪ Barcode #4 : fix checksum
  ▪ Barcode #5 : try to fix digits one by one

\$ \texttt{openssl enc -d -aes-128-cbc -in EAN_13\_flag.aes -k} \hfill
437433591310381767165299985030952613534423187020536389143429357963145583521553

\texttt{FLAG = MD5("barcodesarefunarentthey")}
Step 7 – Little Brother

 Commodore 64
Step 7 – Little Brother

✓ Commodore Vic-20
Step 8 – Memory Base Register

✓ MBR to reverse
✓ 16-bits code, real mode
✓ First step is a XOR loop to decrypt next layer

Code snippet:
```
loc_36:
    lodsd
    xor eax, 0BADF00Dh
    stosd
```
```
loc_40:
    loop loc_36
```
Step 8 – Memory Base Register

✓ After decryption:

✓ OMG ROP!
Step 8 – Memory Base Register

✔ Clean all the things!

- Grab pushes 2 by 2 (segment and address)
- Compute segment*16 + address
- Grab basic block at computed address
- Replace add sp / sub sp by jmps
### Step 8 – Memory Base Register

✅ Cleaned 😊 (a bit)

```
; segment byte public `CODE' use16
assume cs:seg000
assume ds:nothing, ss:nothing, fs:nothing, gs:nothing

mov  si, OEE8Oh
mov ah, 0Eh
mov bx, 7

loc_10008:

    lods  ; CODE XREF: seg000:0020

    cmp al, 0
    jz loc_10030

    mov bp, [si]

    xor bp, bp

    mov ss, bp

    mov bp, [si]

    mov sp, 7C80h

    mov ds:OEE7Eh, bp

    int 10h

    ; VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (Ttv Write)
    ; AL = character, BH = display page (alpha modes)
    ; BL = foreground color (graphics modes)

    mov sp, ds:OEE7Eh

    mov bp, 9800h

    mov ss, bp

    assume ss:nothing

    jmp loc_10008

loc_10008:

    nop

    mov di, OEEAh

    dec di

    xor bx, bx
```
Step 8 – Memory Base Register

✓ Prompts password to user
✓ Jumps to somehow obfuscated code
  ▪ Push/pop emulated because of ROP, e.g. push 0x2e is:

```
mov bp, ds:0F339h
dec bp
dec bp
add bp, 0F33Bh
mov word ptr [bp+0], 2Eh ; .
sub bp, 0F33Bh
mov ds:0F339h, bp
```

  ▪ Some useless instructions

✓ Then transformations on the password (XOR, then stored 3 bits by 3 bits in an array, skipping some indexes)
✓ Finally, print Badboy message
✓ Whaaaat? No check?
Step 8 – Memory Base Register

 ✓ Let’s trace a bit...

 ✓ Instructions are executed before actual Badboy printing
  ▪ Someone has been messing with int 10h!
  ▪ Hooking!!

 ✓ In the hook function, new ROP chain!

| Push 12A0h |
| Push 2F 88h |
| Push 13E1h |
| Push 1C88h |
| Push 1274h |
| Push 956Ah |
| Push 1146h |
| Push 7808h |
| Push 1389h |
| Push 9208h |
| Push 1312h |
| Push 0B5D00h |
| Push 1954h |
| Push 1773h |
| Push 121Bh |
| Push 9B5Ah |
| Push 11F8h |
| Push 684Ah |
| Push 10C9h |
| Push 9D5Ch |
| Push 143Fh |
| Push 3A64h |
| Push 1300h |
| Push 0A55Eh |
| Push 12B0h |
| Push 00247h |
| Push 13A7h |
| Push 570Fh |
| Push 114Ceh |
| Push 3F79h |
| Push 118Ch |
| Push 5860h |
| Push 13DCh |
| Push 6F9Ch |
| Push 1032h |
| Push 92B8h |
| Push 1151h |
| Push 0A55Eh |
| Push 1011h |
| Push 0D461h |
Step 8 – Memory Base Register

✓ Clear all the things (again)!

```
sub 0 proc near
push ax
push bx
push cx
push si
push di
xor ax, ax
xor bx, bx
mov cx, 9
```
Step 8 – Memory Base Register

✓ Now what?
✓ Previously filled array is checked
  ▪ Size is 9x9
  ▪ Checks there’s only one occurrence of each digits between 0 and 8 for each:
    o Line
    o Column
    o 3x3 square
✓ This is a SUDOKU!
Step 8 – Memory Base Register

✓ Output the empty Sudoku:
Step 8 – Memory Base Register

✓ Solve!
Step 8 – Memory Base Register

✓ Revert algorithm (XOR etc.)
✓ ???
✓ PROFIT!

Flag: h00ked_ur_cod3!
Step 9 – Star Wars

✓ IBM 1130
Step 10 - Champagne

✔ Questions?