## ***Descripción***

***Nombre:*** Tina_Russo (https://twitter.com/Hackers4F/status/1093990717502959616 )
***Related:*** Looney Tunes ( https://looneytunes.fandom.com/wiki/Tina_Russo )
***Fecha de liberación:*** 8 de febrero de 2019
***Autor:*** 1v4n
***Dificultad:*** Medio-Bajo

*Tina has discovered that someone has been looking at her laptop. Daffy Duck has been able to extract the traffic but it is encrypted. Can you help them?*

## Objetivo

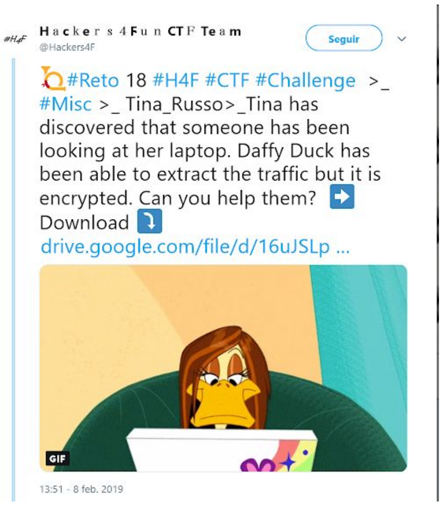
Formato de flag: H4F{text}

## Herramientas utilizadas

Firefox V. 60.3.0 https://www.mozilla.org/en-US/firefox/60.3.0/releasenotes/
gdown 3.6.0 https://pypi.org/project/gdown/
7z-crack https://github.com/kholia/7z-crack
SecLists https://github.com/danielmiessler/SecLists
TShark (Wireshark) 2.6.6 https://www.wireshark.org/#download
7-Zip [64] 16.02 https://www.7-zip.org/download.html

## Resumen:

Comenzamos por visitar la página del reto y descargamos el archivo comprimido *R3t0_18_H4F_T1n4_Russ0.7z (8c6aaa1d3494d741f9fc0a51a68e9718)* y que tiene la password una password desconocida.



Hackers4Fun CTF Team
@Hackers4F

#Reto 18 #H4F #CTF #Challenge >_ #Misc >_ Tina_Russo>_Tina has discovered that someone has been looking at her laptop. Daffy Duck has been able to extract the traffic but it is encrypted. Can you help them? ➡️ Download ⤵️ drive.google.com/file/d/16uJSLp ...

13:51 - 8 feb. 2019

```
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# gdown
https://drive.google.com/uc?id=16uJSLpnUdAvpt7l5eZ-VY5pdpQmUxLmn
Downloading...
From: https://drive.google.com/uc?id=16uJSLpnUdAvpt7l5eZ-VY5pdpQmUxLmn
To: /root/Escritorio/Desktop/C-Hackers4F/Reto18/R3t0_18_H4F_T1n4_Russ0.7z
100%|████████████████████████████████████| 2.64k/2.64k [00:00<00:00, 3.14MB/s]
```

```
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# file R3t0_18_H4F_T1n4_Russ0.
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# file
R3t0_18_H4F_T1n4_Russ0.7z
R3t0_18_H4F_T1n4_Russ0.7z: 7-zip archive data, version 0.4
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# 7z x
R3t0_18_H4F_T1n4_Russ0.7z

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,Utf16=on,HugeFiles=on,64 bits,1 CPU
Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz (406E3),ASM,AES-NI)

Scanning the drive for archives:
1 file, 2642 bytes (3 KiB)

Extracting archive: R3t0_18_H4F_T1n4_Russ0.7z
--
Path = R3t0_18_H4F_T1n4_Russ0.7z
Type = 7z
Physical Size = 2642
Headers Size = 178
Method = LZMA2:14 7zAES
Solid = -
Blocks = 1


Enter password (will not be echoed):
ERROR: Data Error in encrypted file. Wrong password? :
R3t0_18_H4F_T1n4_Russ0.pcap

Sub items Errors: 1

Archives with Errors: 1

Sub items Errors: 1
```

Pasamos a utilizar la herramienta 7z-crack para cracking de archivos comprimidos .7z generando el siguiente bash

```
#! /bin/bash

cat /usr/share/wordlists/rockyou.txt | grep "looney.*" | ./7z-crack/bin/7za t
R3t0_18_H4F_T1n4_Russ0.7z
```

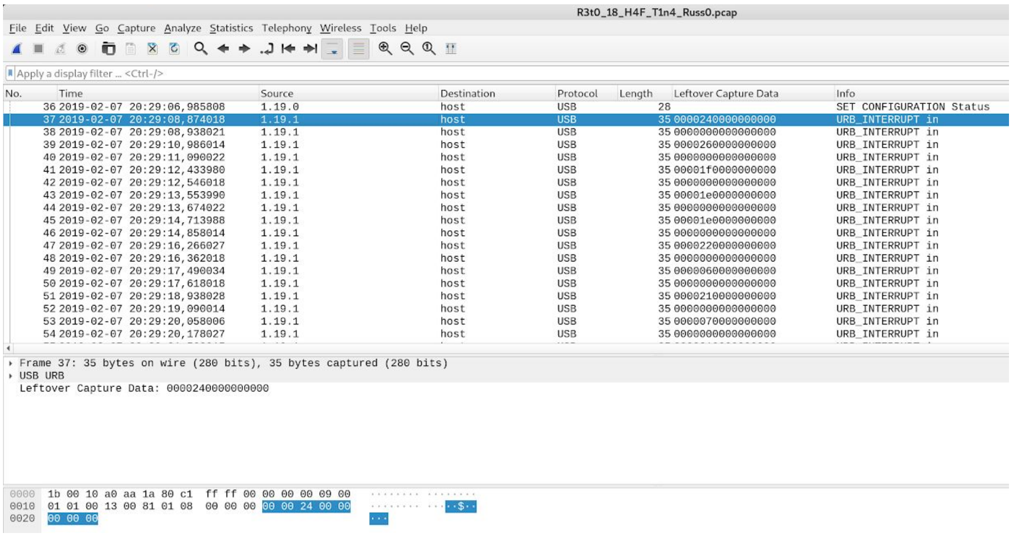
y lo ejecutamos obtenido la password ***looneytunes***:

```
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# chmod +x get_pass.sh
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# ./get_pass.sh
Everything is Ok
Password Found : looneytunes
```

Pasamos a descomprimirlo y descubrimos que nos esconde un archivo de paquetes de tráfico de red lamado R3t0_18_H4F_T1n4_Russ0.pcap (9898e05ea677409f781d91f8580de228) :

```
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# 7z x
R3t0_18_H4F_T1n4_Russ0.7z
...
Everything is Ok
Size:       15364
Compressed: 2642
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# md5sum
R3t0_18_H4F_T1n4_Russ0.pcap
9898e05ea677409f781d91f8580de228  R3t0_18_H4F_T1n4_Russ0.pcap
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# file
R3t0_18_H4F_T1n4_Russ0.pcap
R3t0_18_H4F_T1n4_Russ0.pcap: pcap capture file, microsecond ts (little-endian)
- version 2.4, capture length 262144)
```

Pasamos a analizar con Wireshark el archivo pcap y observamos que pertenece a una captura de tráfico USB


R3t0_18_H4F_T1n4_Russ0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

| No. | Time | Source | Destination | Protocol | Length | Leftover Capture Data | Info |
|---|---|---|---|---|---|---|---|
| 36 | 2019-02-07 20:29:06,985808 | 1.19.0 | host | USB | 28 | | SET CONFIGURATION Status |
| 37 | 2019-02-07 20:29:08,874018 | 1.19.1 | host | USB | 35 | 0000240000000000 | URB_INTERRUPT in |
| 38 | 2019-02-07 20:29:08,938021 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 39 | 2019-02-07 20:29:10,986014 | 1.19.1 | host | USB | 35 | 0000260000000000 | URB_INTERRUPT in |
| 40 | 2019-02-07 20:29:11,090022 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 41 | 2019-02-07 20:29:12,433980 | 1.19.1 | host | USB | 35 | 00001f0000000000 | URB_INTERRUPT in |
| 42 | 2019-02-07 20:29:12,546018 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 43 | 2019-02-07 20:29:13,553990 | 1.19.1 | host | USB | 35 | 00001e0000000000 | URB_INTERRUPT in |
| 44 | 2019-02-07 20:29:13,674022 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 45 | 2019-02-07 20:29:14,713988 | 1.19.1 | host | USB | 35 | 00001e0000000000 | URB_INTERRUPT in |
| 46 | 2019-02-07 20:29:14,858014 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 47 | 2019-02-07 20:29:16,266027 | 1.19.1 | host | USB | 35 | 0000220000000000 | URB_INTERRUPT in |
| 48 | 2019-02-07 20:29:16,362018 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 49 | 2019-02-07 20:29:17,490034 | 1.19.1 | host | USB | 35 | 0000060000000000 | URB_INTERRUPT in |
| 50 | 2019-02-07 20:29:17,618018 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 51 | 2019-02-07 20:29:18,938028 | 1.19.1 | host | USB | 35 | 0000210000000000 | URB_INTERRUPT in |
| 52 | 2019-02-07 20:29:19,090014 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 53 | 2019-02-07 20:29:20,058006 | 1.19.1 | host | USB | 35 | 0000070000000000 | URB_INTERRUPT in |
| 54 | 2019-02-07 20:29:20,178027 | 1.19.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |

```
▸ Frame 37: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
▸ USB URB
  Leftover Capture Data: 0000240000000000

0000  1b 00 10 a0 aa 1a 80 c1  ff ff 00 00 00 00 09 00   ........ ........
0010  01 01 00 13 00 81 01 08  00 00 00 00 00 24 00 00   ........ ....$..
0020  00 00 00                                           ...
```

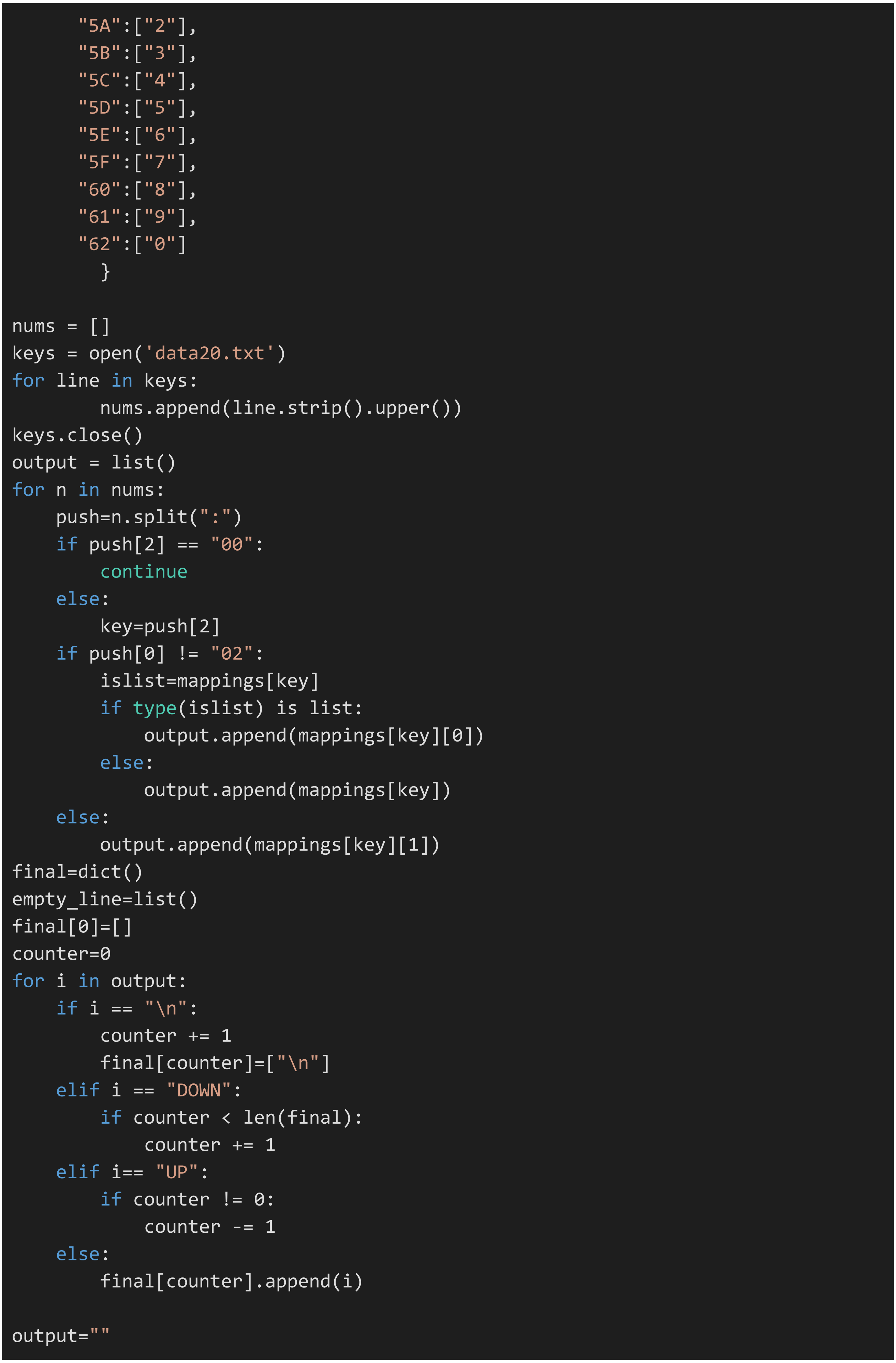
Detectamos que en la captura hay dos dispositivos USB con direcciones 1.19.1 y 1.20.1 con lo que vamos a separar las pulsaciones de cada uno a través de tshark:

```
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# tshark -r
R3t0_18_H4F_T1n4_Russ0.pcap -Y "usb.bus_id == 1 && usb.device_address == 20 &&
usb.transfer_type == 0x01" -T fields -e usb.capdata | awk -F: '{print $3}' |
grep -v 00 > data20.txt
```

```
root@kali:~/Escritorio/Desktop/C-Hackers4F/Reto18# tshark -r
R3t0_18_H4F_T1n4_Russ0.pcap -Y "usb.bus_id == 1 && usb.device_address == 19 &&
usb.transfer_type == 0x01" -T fields -e usb.capdata | awk -F: '{print $3}' |
grep -v 00 > data19.txt
```

Apoyandonós en USB keymap (http://www.mindrunway.ru/IgorPlHex/USBKeyScan.pdf ) pasamos a utilizar un script en python como el que se utilizó Kalrong en el reto de USB Ducker que nos facilitará la labor :

```
#!/usr/bin/python
mappings = {
        "04":["a","A"],
        "05":["b","B"],
        "06":["c","C"],
        "07":["d","D"],
        "08":["e","E"],
        "09":["f","F"],
        "0A":["g","G"],
        "0B":["h","H"],
        "0C":["i","I"],
        "0D":["j","J"],
        "0E":["k","K"],
        "0F":["l","L"],
        "10":["m","M"],
        "11":["n","N"],
        "12":["o","O"],
        "13":["p","P"],
        "14":["q","Q"],
        "15":["r","R"],
        "16":["s","S"],
        "17":["t","T"],
        "18":["u","U"],
        "19":["v","V"],
        "1A":["w","W"],
        "1B":["x","X"],
        "1C":["y","Y"],
        "1D":["z","Z"],
        "1E":["1","!"],
        "1F":["2","@"],
        "20":["3","#"],
        "21":["4","$"],
        "22":["5","%"],
        "23":["6","^"],
        "24":["7","&"],
        "25":["8","*"],
        "26":["9","("],
        "27":["0",")"],
        "28":"\n",
        "29":"ESC",
        "2A":"BKSPC",
        "2B":"  ",
        "2C":" ",
        "2D":["-","_"],
        "2E":["=","+"],
        "2F":["[","{"],
        "30":["]","}"],
        "31":["\\","|"],
        "32":"(INT 2)",
        "33":[";",":"],
        "34":[",",'"'],
        "35":["`","~"],
        "36":[",","<"],
        "37":[".",">"],
        "38":["/","?"],
        "39":"CAPSLOCK",
        "3A":"F1",
        "3B":"F2",
        "3C":"F3",
        "3D":"F4",
        "3E":"F5",
        "3F":"F6",
        "40":"F7",
        "41":"F8",
        "42":"F9",
        "43":"F10",
        "44":"F11",
        "45":"F12",
        "46":"PRTSCR",
        "47":"SCRLOCK",
        "48":"PAUSE",
        "49":"INS",
        "4A":"HOME",
        "4B":"PGUP",
        "4C":"DEL",
        "4D":"END",
        "4E":"PGDOWN",
        "4F":"RIGHT",
        "50":"LEFT",
        "51":"DOWN",
        "52":"UP",
        "53":"NUMLOCK",
      "54":["/"],
      "55":["*"],
      "56":["-"],
      "57":["+"],
      "58":"ENTER",
      "59":["1"],
```



```
      "5A":["2"],
      "5B":["3"],
      "5C":["4"],
      "5D":["5"],
      "5E":["6"],
      "5F":["7"],
      "60":["8"],
      "61":["9"],
      "62":["0"]
        }

nums = []
keys = open('data20.txt')
for line in keys:
        nums.append(line.strip().upper())
keys.close()
output = list()
for n in nums:
    push=n.split(":")
    if push[2] == "00":
        continue
    else:
        key=push[2]
    if push[0] != "02":
        islist=mappings[key]
        if type(islist) is list:
            output.append(mappings[key][0])
        else:
            output.append(mappings[key])
    else:
        output.append(mappings[key][1])
final=dict()
empty_line=list()
final[0]=[]
counter=0
for i in output:
    if i == "\n":
        counter += 1
        final[counter]=["\n"]
    elif i == "DOWN":
        if counter < len(final):
            counter += 1
    elif i== "UP":
        if counter != 0:
            counter -= 1
    else:
        final[counter].append(i)

output=""
```

```
for x in final.keys():
    for y in final[x]:
        output+=y
print output
```

Las dos cadenas que obtenemos son la correspondiente:
> `H0EFE2HjMmOMnx5BMSEJMyEHDayJER55IyEBp01RFxkAITj5`
> `792115c4d4ed3ea4b04a6af529a95d21`

La primera cadena la desencriptamos a través de CyberChef > https://gchq.github.io/CyberChef/#recipe=From_Base64('N-ZA-Mn-za-m0-9%2B/%3D',true)From_Base64('A-Za-z0-9%2B/%3D',true)&input=SDBFRkUySGpNbU9Nbng1Qk1TRUpNeUVIRGF5SkVSNTVJeUVCcDAxUkZ4a0FJVGo1 > `H4F{H4b3Mu5_M0rT3rU3l02K19}`

La segunda cadena que identificamos como hash md5 pasamos a reversearla > https://md5hashing.net/hash/md5/792115c4d4ed3ea4b04a6af529a95d21 > `FR33_M0rt3ru3l0CON_4j04rr13r0L4BS`

La flag es: ***H4F{H4b3Mu5_M0rT3rU3l02K19}***

Autor: 1v4n a.k.a. @1r0Dm48O
Twitter: https://twitter.com/1r0Dm48O