Los pergaminos del engaño ninja

El primer fichero parece contener datos codificados en base64, pero los valores solo contienen mayúsculas y finalizan con tres “=”, esto no puede suceder en base64. Los datos están codificados en base32.

```
 1 MI4WKY3FGE4GGOJVGBQWMYTGME3GEMDGMRRGMYJUMZTDOMZRMQZQU===
 2 GI2TCMDDGM4TAMJRMM2WEZJXGA2DCOBSGQZDGZJTME3DSNLFHEYQU===
 3 MUYTMNZRG44TOYZVGJSTCNLGG43DGMZYGBRDINLFHA2DCZLDGMZAU===
 4 HAYDANRRHA4TIMZQGI2TGMJVMY4DMOLFGRSTCZRQHE2DOMJQGEZAU===
 5 GJSGEOJVMU4GKMLBHEZDMN3CG5QTCMJYHA2TKNTCGIYDCM3CGMZQU===
 6 GBRWGMJXGVRDSYZQMYYWENTBHAZTCYZTHE4WKMRWHE3TOMRWGYYQU===
 7 MIZGMNLGMY2DONBTGY3DOMLCGZSTKMZTMQ4GIYZTGYYTIOBUGVSAU===
 8 MRSDONJTGY3TSNDCGYZWEZRZGBSWGY3GMQZTOZRZMIYTIN3EG5TAU===
 9 GAZWGN3DGBQWGZJTHE2WIOBQGE4DEZDCGA3WCZJSMMZTAZRQGM2AU===
10 MMYWIOLGGUYGMOBWHAZDKYJRMEZDGMBSMVRTENBUHFRTCNZRHE3AU===
11 HA3DKYZQMMYGENDBMIYGKMBWGNSTKY3BMEZTGOBXMMYWCOBXGQYQU===
12 HAZDON3FGA4TCMDEG42TAMJZGVRDINBYG44TONRRGZSTAOJRMFSAU===
13 HAZDON3FGA4TCMDEG42TAMJZGVRDINBYG44TONRRGZSTAOJRMFSAU===
14 MUYTMNZRG44TOYZVGJSTCNLGG43DGMZYGBRDINLFHA2DCZLDGMZAU===
15 G5RDQYRZGY2WCZBUMJRWCMDFGQYWCYRVGFSGKN3CGMYTGNRTMEYQU===
16 MRSDONJTGY3TSNDCGYZWEZRZGBSWGY3GMQZTOZRZMIYTIN3EG5TAU===
17 G5RDQYRZGY2WCZBUMJRWCMDFGQYWCYRVGFSGKN3CGMYTGNRTMEYQU===
18 GAZWGN3DGBQWGZJTHE2WIOBQGE4DEZDCGA3WCZJSMMZTAZRQGM2AU===
19 HA3DKYZQMMYGENDBMIYGKMBWGNSTKY3BMEZTGOBXMMYWCOBXGQYQU===
20 HAZDON3FGA4TCMDEG42TAMJZGVRDINBYG44TONRRGZSTAOJRMFSAU===
```

Lo decodificamos:

```
import base64

with open("message1.txt", "r") as fin, open("b32decode.txt", "w") as fout:
for line in fin:
line = line.strip()
if not line:
continue
line += "=" * (-len(line) % 8)
decoded = base64.b32decode(line)
fout.write(decoded.decode("utf-8", errors="ignore"))
```

```
┌──(vfric㉿vfric)-[~/…/CTF-cybermin
└─$ cat b32decode.txt
b9ece18c950afbfa6b0fdbfa4ff731d3
2510c39011c5be704182423e3a695e91
e1671797c52e15f763380b45e841ec32
8006189430253l5f869e4e1f09471012
2db95e8e1a9267b7a1188556b2013b33
0cc175b9c0f1b6a831c399e269772661
b2f5ff47436671b6e533d8dc3614845d
dd7536794b63bf90eccfd37f9b147d7f
03c7c0ace395d80182db07ae2c30f034
c1d9f50f86825a1a2302ec2449c17196
865c0c0b4ab0e063e5caa3387c1a8741
8277e0910d750195b448797616e091ad
8277e0910d750195b448797616e091ad
e1671797c52e15f763380b45e841ec32
7b8b965ad4bca0e41ab51de7b31363a1
dd7536794b63bf90eccfd37f9b147d7f
7b8b965ad4bca0e41ab51de7b31363a1
03c7c0ace395d80182db07ae2c30f034
865c0c0b4ab0e063e5caa3387c1a8741
8277e0910d750195b448797616e091ad
e1671797c52e15f763380b45e841ec32
```

El archivo creado contiene un montón de valores ilegibles, pero haciendo uso de hash-identifier comprobamos que se trata de hashes MD5.

```
┌──(vfric㉿vfric)-[~/…/CTF-cyberminds-1er-Edici-n/Crypto/Los Pergaminos del Er
└─$ hash-identifier
   ###########################################################################
   #                                                                         #
   #                                                                         #
   #                                                                         #
   #                                                                         #
   #                                                                         #
   #                                                                         #
   #                                                                         #
   #                                                                v1.2     #
   #                                                             By Zion3R   #
   #                                                          www.Blackploit.com #
   #                                                         Root@Blackploit.com #
   ###########################################################################
--------------------------------------------------

 HASH: b9ece18c950afbfa6b0fdbfa4ff731d3

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

Mediante el uso de la herramienta John the Ripper obtenemos el valor de los hashes contenidos en el fichero.

```
└─$ john --format=raw-md5 b32decode.txt
Created directory: /home/vfric/.john
Using default input encoding: UTF-8
Loaded 306 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=16
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
1                (?)
1                (?)
1                (?)
1                (?)
1                (?)
1                (?)
```

```
└─$ cat /home/vfric/.john/john.pot
$dynamic_0$c4ca4238a0b923820dcc509a6f75849b:1
$dynamic_0$0cc175b9c0f1b6a831c399e269772661:a
$dynamic_0$e1671797c52e15f763380b45e841ec32:e
$dynamic_0$865c0c0b4ab0e063e5caa3387c1a8741:i
$dynamic_0$8277e0910d750195b448797616e091ad:d
$dynamic_0$03c7c0ace395d80182db07ae2c30f034:s
$dynamic_0$2db95e8e1a9267b7a1188556b2013b33:l
$dynamic_0$cfcd208495d565ef66e7dff9f98764da:0
$dynamic_0$b2f5ff47436671b6e533d8dc3614845d:g
$dynamic_0$7b8b965ad4bca0e41ab51de7b31363a1:n
$dynamic_0$2510c39011c5be704182423e3a695e91:h
$dynamic_0$dd7536794b63bf90eccfd37f9b147d7f:I
$dynamic_0$b9ece18c950afbfa6b0fdbfa4ff731d3:T
$dynamic_0$c1d9f50f86825a1a2302ec2449c17196:H
$dynamic_0$8006189430253l5f869e4e1f09471012:F
$dynamic_0$853ae90f0351324bd73ea615e6487517::
```

Representamos las cadenas de hashes md5 a ascii(puedes usar crackstation)

```python
import hashlib
import string

# Archivo con hashes MD5 (uno por línea)
FILE = "b32decode.txt"

# Conjunto de caracteres a probar (ajusta si es necesario)
charset = (
string.ascii_lowercase +
string.ascii_uppercase +
string.digits +
string.punctuation +
" "
)
```

```python
def md5(s):
return hashlib.md5(s.encode()).hexdigest()

decoded = ""

with open(FILE, "r") as f:
for line in f:
hash_line = line.strip()
found = False

for c in charset:
if md5(c) == hash_line:
decoded += c
found = True
break

if not found:
decoded += "?"

print("Frase decodificada:")
print(decoded)
```

```
└─$ python3 hashmd5.py
Frase decodificada:
TheFlagIsHiddenInsideThis:0101001100111011001000110000110100100000101100110001011001100010100100101011111100010110001
1001101110001010000101100100010000111000101010011100011001010101001111110001000110000010101101101110010010101100100
0011001000110000100110010110010101000011011000011010010101100010101101110010
```

Hacemos un xor entre el codigo binario anterior y el message2.txt.