ENCODING FUN

AUTHOR:

CYBEXER TECHNOLOGIES

International Cyber Security Challenge

SEPT 2021

1. DESCRIPTION

The target file contains an encrypted flag. The task is to figure out the encryption and/or obfuscation method(s), there can be multiple of them embedded into each other.

2. CHALLENGE SPECIFICATIONS

- Categroty: Crypto
- Difficulty: Easy
- Estimated time: 5-10 min

3. QUESTIONS AND ANSWERS

3.1 WHAT TEXT IS IN THE FILE?

f[abcdefgh]

4. SETUP INSTRUCTIONS

Dockerfile and *docker-compose.yml* are provided to run the task in a container. FLAG and PORT can be given as build-time arguments to Docker, there are defaults in Dockerfile.

With docker-compose, they are also populated from environment, as shown in **.env** file.

docker-compose build

docker-compose up

The file is generated during build time with commands in *Dockerfile*. The container is just serving it over HTTP at given port.

5. ARTIFACTS PROVIDED

File	SHA-256
encoding-fun.tar.gz	9348661bd802968959f36ec4c294fe7d9a03868fbd410c86fb75caee7172d726

6. TOOLS NEEDED

- Common Linux utilities

7. WALKTHROUGH

A file called **encoded.txt** is served to us. First let's determine the file type:

```
$ file encoded.txt
encoded.txt: gzip compressed data, from Unix, original size 508
```

Despite the *.txt* extension, it is actually not a text file. Let's unpack it:

```
$ mv encoded.txt encoded.txt.gz
$ gunzip encoded.txt.gz
$ file encoded.txt
encoded.txt: ASCII text
$ cat encoded.txt
10,20,4d55496f6864652b637a6575724a46515a424176
1,20,516c706f4f54464257535a54576151557a6b7341
9,20,51674d68504c4d516f3748416142434d4466654c
8,20,6b6b5149413773443076797639506859745 84d55
7,20,336b5653786768515871797 94d66655a57663232
11,20,4e75356f774851753549707 76f5346494b5a7957
6,20,534e674674737772635977 4b344e2b455749434a
5,20,453177546149766f69416c6c6e4f627131414c53
2,20,4143467a67476f3441414413559414577414141
3,20,4244414132434b5a4d54494d6a436d5445794449
4,20,775a6f6971656f615053667054355358693 76b53
```

Now we have some CSV text. We can notice that first column has all numbers from 1 to 11, second is always 20 and finally there is some hexprint. Let's make assumptions that the content must be sorted according to first column and hexprint must be converted to binary. This can be achieved with few commands:

```
$ sed 's/,/ /g' encoded.txt | sort -n > sorted.txt
$ cut -d' ' -f3 sorted.txt | tr -d '\n' | perl -e 'print pack("H*",<>);'
QlpoOTFBWSZTWaQUzksAACFzgGo4AADA5YAEwAAABDAA2CKZMTIMjCmTEyDIwZoiqeoaPSfpT5SXi7kSE1wTaIvoiAllnObq1A
LSSNgFtswrcYwK4N+EWICJ3kVSxghQXqyyMfeZWf22kkQIA7sD0vyv9PhYtXMUQgMhPLMQo7HAaBCMDfeLMUIohde+czeurJFQ
ZBAvNu5owHQu5IpwoSFIKZyW
```

This looks like Base64. Let's try to decode it:

```
$ cut -d' ' -f3 sorted.txt | tr -d '\n' | perl -e 'print pack("H*",<>);' > encoded.b64
$ base64 -d encoded.b64 > encoded.bin
$ file encoded.bin
encoded.bin: bzip2 compressed data, block size = 900k
```

Obvious next step is to unpack the file:

```
$ mv encoded.bin encoded.bz2
$ bunzip2 encoded.bz2
$ file encoded
encoded: ASCII text
$ cat encoded
  __  ___       _               _         __  ___
 / _||  _|     | |             | |       / _||_  |
| |_ | |  __ _ | |__    ___  __| |  ___ | |_   | |
|  _|| | / _` || '_ \  / __|/ _` | / _ \|  _|  | |
| |  | || (_| || |_) || (__| (_| ||  __/| |    | |
|_|  | |_\__,_||_.__/  \___|\__,_| \___||_|   _| |
     |___|                                  |___|
```

Done.

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str.
151 24 Marousi, Attiki, Greece

Heraklion Office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu

ISBN xxx-xx-xxxx-xxx-x
doi:xx.xxxx/xxxxxx
TP-xx-xx-xxx-EN-C