RingZer0 Team Online CTF

A solution for the challenge 55

Is it a secure string?

Category

Cryptography

By

garigouster

For this cryptography challenge, we have a password string and a key.

Given the set of characters for the password string, it is probably encoded with Base64. But looking more closely at this string, we can see that the first part of this string is rather composed of hexadecimal characters:

```
$ p=76492d1116743f0423413b16050a5345MgB8AEEAYQBNAHgAZQAxAFEAVABIAEEAcABtAE4ATgBV
AFoAMwBOAFIAagBIAGcAPQA9AHwAZAAyADYAMgA2ADgAMwBlADcANAA3ADIAOQA1ADIAMwA0ADMAMwBl
ADIAOABmADIAZABlAGMAMQBiAGMANgBjADYANAA4ADQAZgAwADAANwA1AGUAMgBlADYAMwA4AGEAZgA1
AGQAYgA5ADIAMgBkAGIAYgA5AGEAMQAyADYAOAA=
$ p1=$(cut -c 1-32 <<< $p)
$ p2=$(cut -c 33- <<< $p)
$ echo $p1
76492d1116743f0423413b16050a5345
$ echo $p2
MgB8AEEAYQBNAHgAZQAxAFEAVABIAEEAcABtAE4ATgBVAFoAMwBOAFIAagBIAGcAPQA9AHwAZAAyADYA
MgA2ADgAMwBlADcANAA3ADIAOQA1ADIAMwA0ADMAMwBlADIAOABmADIAZABlAGMAMQBiAGMANgBjADYA
NAA4ADQAZgAwADAANwA1AGUAMgBlADYAMwA4AGEAZgA1AGQAYgA5ADIAMgBkAGIAYgA5AGEAMQAyADYA
OAA=
```

See the second part of the string Base64 encoded:

```
$ openssl enc -d -A -base64 <<< $p2 | hexdump -C
00000000  32 00 7c 00 41 00 61 00  4d 00 78 00 65 00 31 00  |2.|.A.a.M.x.e.1.|
00000010  51 00 54 00 48 00 41 00  70 00 6d 00 4e 00 4e 00  |Q.T.H.A.p.m.N.N.|
00000020  55 00 5a 00 33 00 4e 00  52 00 6a 00 48 00 67 00  |U.Z.3.N.R.j.H.g.|
00000030  3d 00 3d 00 7c 00 64 00  32 00 36 00 32 00 36 00  |=.=.|.d.2.6.2.6.|
00000040  38 00 33 00 65 00 37 00  34 00 37 00 32 00 39 00  |8.3.e.7.4.7.2.9.|
00000050  35 00 32 00 33 00 34 00  33 00 33 00 65 00 32 00  |5.2.3.4.3.3.e.2.|
00000060  38 00 66 00 32 00 64 00  65 00 63 00 31 00 62 00  |8.f.2.d.e.c.1.b.|
00000070  63 00 36 00 63 00 36 00  34 00 38 00 34 00 66 00  |c.6.c.6.4.8.4.f.|
00000080  30 00 30 00 37 00 35 00  65 00 32 00 65 00 36 00  |0.0.7.5.e.2.e.6.|
00000090  33 00 38 00 61 00 66 00  35 00 64 00 62 00 39 00  |3.8.a.f.5.d.b.9.|
000000a0  32 00 32 00 64 00 62 00  62 00 39 00 61 00 31 00  |2.2.d.b.b.9.a.1.|
000000b0  32 00 36 00 38 00                                 |2.6.8.|
```

The decoded data contains null characters probably due to Unicode characters. With bash, we can easily remove null characters for an ASCII string:

```
$ p2d=$(openssl enc -d -A -base64 <<< $p2)
$ echo $p2d
2|AaMxe1QTHApmNNUZ3NRjHg==|d262683e74729523433e28f2dec1bc6c6484f0075e2e638af5db9
22dbb9a1268
```

These decoded data contain 3 fields separated by a pipe character. The first field is probably a flag on the format of password string or the cryptography algorithm used. The third field is another hexadecimal string.

For the second field, data are again Base64 encoded. Let's examine this more closely:

```
$ p2d=($(IFS='|' ; echo -n $p2d))
$ openssl enc -d -A -base64 <<< ${p2d[1]} | hexdump -C
00000000  01 a3 31 7b 54 13 1c 0a  66 34 d5 19 dc d4 63 1e  |..1{T...f4....c.|
00000010
```

Nothing in particular. The decoded data are any.

Let's do a little review of size of the various data:

```
$ p22=$(openssl enc -d -A -base64 <<< ${p2d[1]} | xxd -p)
$ echo $p22
01a3317b54131c0a6634d519dcd4631e
$ echo -n $p22 | wc -c
      32
$ p23=${p2d[2]}
$ echo -n $p23 | wc -c
```

```
      64
$ k=$(perl -pe '$_ = join "",map{sprintf "%02x",$_} split ","' <<< 3,4,2,3,56,34
,254,222,205,34,2,23,42,64,33,223,1,34,2,7,6,5,35,12)
$ echo $k
030402033822fedecd2202172a4021df012202070605230c
$ echo -n $k | wc -c
      48
```

Firstly the “key” has a length of 24 (=48/2) bytes/numbers. This is a valuable indication because there are not many cryptographic algorithms using this key size: it is certainly AES (192) which is used (or Rijndael).

Secondly there are 2 hexadecimal data (remember the first part of the password string) with a length of 16 (=32/2) bytes. This can be the encrypted data, but also a seed (initial vector) which have a size of 16 bytes with AES.

The last data have a size of 32 (=64/2) bytes. This is surely not a seed for AES, or a MD5 or SHA-1 digest: these are probably the encrypted data.

With these indications and after some tests, then we succeed easily decode data:

```
$ xxd -r -p <<< $p23 | openssl enc -d -aes192 -K $k -iv $p22 ; echo
FLAG-.....
```

A small pleasure in solving this challenge only with deduction and UNIX shell…

A quick search (with the keyword: SecureString) on Internet allows us to understand that this secure string is a Windows standard to protect passwords. There are many ways to program the decoding of a secure string (class of .NET or C#, or other tools).