CTF杂项之BubbleBabble加密算法

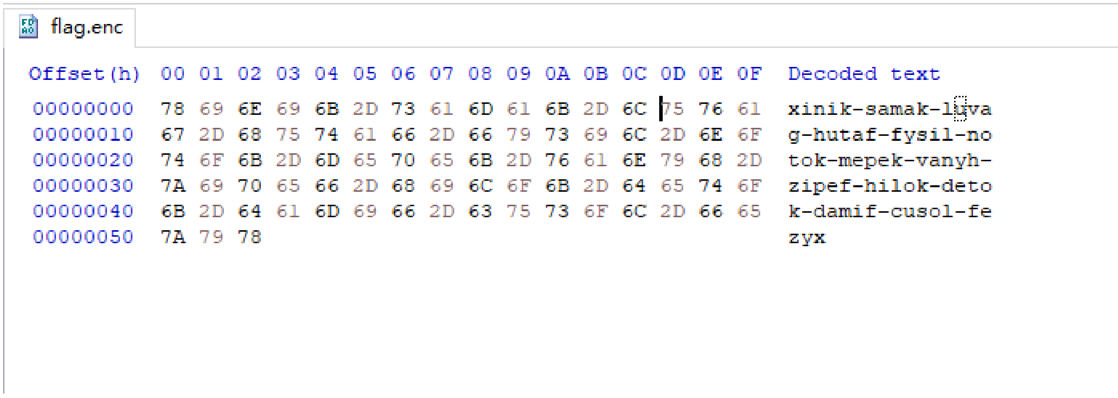
转载

digupang7059 于 2019-03-28 17:44:00 发布 835 收藏 1
文章标签： python 运维
原文链接：http://www.cnblogs.com/mke2fs/p/10616588.html

这题很坑，刚开始我拿到就分析不出来了（/无奈），关键是不知道是什么加密算法，后来看题目描述的bubble,猜测是bubble

这种算法（听都没听说过。。。）

上图



这串编码

```
xinik-samak-luvag-hutaf-fysil-notok-mepek-vanyh-zipef-hilok-detok-damif-cusol-fezyx
```

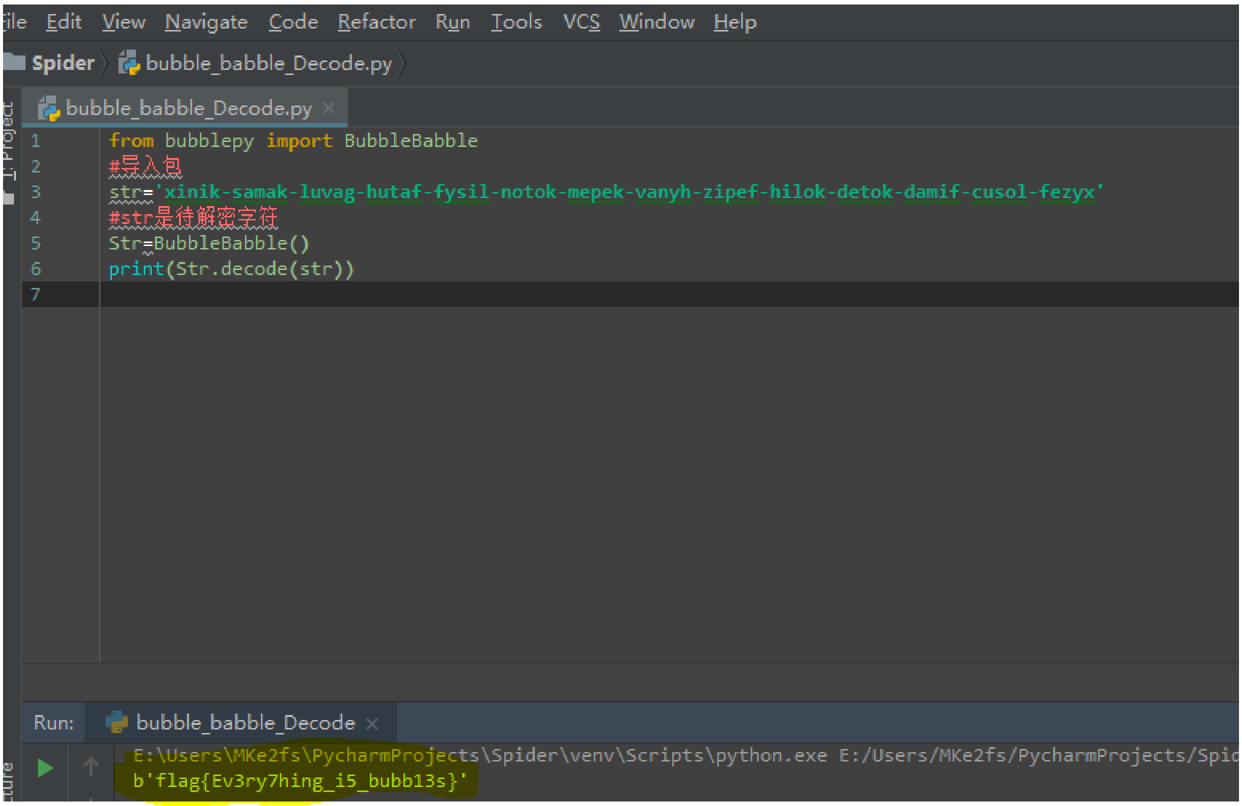
百度也没找到哪里有，后面看到有大佬写的writeup，于是就借鉴一下下，233

这里是这种加密算法的解释（为了方便大家，加密算法我复制到文末），链接传送：**http://wiki.yak.net/589/Bubble_Babble_Encoding.txt**

附上解题代码，楼主用的python3.7,另外bubblepy这个库需要导入一下，网址附上**https://pypi.python.org/pypi/bubblepy/**

```python
from bubblepy import BubbleBabble
#导入包bubblepy
str='xinik-samak-luvag-hutaf-fysil-notok-mepek-vanyh-zipef-hilok-detok-damif-cusol-fezyx'
#str是待解密字符
Str=BubbleBabble()
print(Str.decode(str))
```

最后，get Flag



这里是BubbleBabble加密算法

```
authors==Huima
status==Experimental
title==The Bubble Babble Binary Data Encoding
number==Internet Draft
date==April 2000
Network Working Group                                    Antti Huima
Internet Draft                              SSH Communications Security
draft-huima-babble-01.txt                                 April 2000


  The Bubble Babble Binary Data Encoding

Status of this Memo

   This memo provides information for the Internet community.  It does
   not specify an Internet standard of any kind.  Distribution of this
   memo is unlimited.

Copyright Notice

   Copyright (C) The Internet Society (2000).  All Rights Reserved.

Abstract

   This document describes a new encoding method for binary data that is
   intended to be used in conjunction with fingerprints of
   security-critical data.

1. Introduction

   Hash values of certificates and public keys, known as fingerprints
   or thumbprints, are commonly used for verifying that a received
   security-critical datum has been received correctly.  Fingerprints
```

are binary data and typically encoded as series of hexadecimal digits. However, long strings hexadecimal digits are difficult for comprehend and cumbersome to translate reliably e.g. over phone.

The Bubble Babble Encoding encodes arbitrary binary data into pseudowords that are more natural to humans and that can be pronounced relatively easily. The encoding consumes asymptotically the same amount of space as an encoding of the form

```
HH HH HH HH ...
```

where `H' is a hexadecimal digit, i.e. carries 16 bits in six characters. However, the Bubble Babble Encoding includes a checksumming method that can sometimes detect invalid encodings. The method does not increase the length of the encoded data.

2. Encoding

Below, _|X|_ denotes the largest integer not greater than X.

Let the data to be encoded be D[1] ... D[K] where K is the length of the data in bytes; every D[i] is an integer from 0 to 2^8 - 1. First define the checksum series C[1] ... C[_|K/2|_] where

```
C[1] = 1

C[n] = (C[n - 1] * 5 + (D[n * 2 - 3] * 7 + D[n * 2 - 2])) mod 36
```

The data is then transformed into _|K/2|_ `tuples' T[1] ... T[_|K/2|_] and one `partial tuple' P so that

```
T[i] = <a, b, c, d, e>
```

where

```
a = (((D[i * 2 - 3] >> 6) & 3) + C[i]) mod 6
b = (D[i * 2 - 3] >> 2) & 15
c = (((D[i * 2 - 3]) & 3) + _|C[i] / 6|_) mod 6
d = (D[i * 2 - 2] >> 4) & 15; and
e = (D[i * 2 - 3]) & 15.
```

The partial tuple P is

```
P = <a, b, c>
```

where if K is even then

```
a = (C[i]) mod 6
b = 16
c = _|C[i] / 6|_
```

but if it is odd then

```
a = (((D[K] >> 6) & 3) + C[i]) mod 6
b = (D[K] >> 2) & 15
c = (((D[K]) & 3) + _|C[i] / 6|_) mod 6
```

The `vowel table' V maps integers between 0 and 5 to vowels as

```
0 - a
```

```
1 - e
2 - i
3 - o
4 - u
5 - y
```

and the `consonant table' C maps integers between 0 and 16 to consonants as

```
 0 - b
 1 - c
 2 - d
 3 - f
 4 - g
 5 - h
 6 - k
 7 - l
 8 - m
 9 - n
10 - p
11 - r
12 - s
13 - t
14 - v
15 - z
16 - x
```

The encoding E(T) of a tuple T = <a, b, c, d, e> is then the string

```
V[a] C[b] V[c] C[d] `-' C[e]
```

where there are five characters, and `-' is the literal hyphen.

The encoding E(P) of a partial tuple P = <a, b, c> is the three-character string

```
V[a] C[b] V[c].
```

Finally, the encoding of the whole input data D is obtained as

```
`x' E(T[1]) E(T[2]) ... E(T[_|K/2|_]) E(P) `x'
```

where `x's are literal characters.

3. Decoding

Decoding is obviously the process of encoding reversed.

To check the checksums, when a tuple <a, b, c, d, e> or partial tuple <a, b, c> has been recovered from the encoded string, an implementation should check that ((a - C[i]) mod 6) < 4 and that ((c - C[i]) mod 6) < 4. Otherwise the encoded string is not a valid encoding of any data and should be rejected.

4. Checksum Strength

Every vowel in an encoded string carries 0.58 bits redundancy; thus the length of the `checksum' in the encoding of an input string containing K bytes is 0.58 * K bits.

5. Test Vectors

ASCII Input	Encoding
`' (empty string)	`xexax'
`1234567890'	`xesef-disof-gytuf-katof-movif-baxux'
`Pineapple'	`xigak-nyryk-humil-bosek-sonax'

6. Author's Address

Antti Huima
SSH Communications Security, Ltd.
[XXX]

7. Full Copyright Statement