BUUCTF 每日打卡 2021-7-27

原创

Σ2333! 于 2021-07-27 21:16:32 发布 55 收藏

分类专栏： crypto 文章标签： 密码学

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/weixin_52446095/article/details/119147361

版权

crypto 专栏收录该内容

79 篇文章 1 订阅

订阅专栏

引言

无

[ACTF新生赛2020]crypto-des

题目给了一串浮点数：

```
721432389920416410000000.000000,
771353571780065040000000000000000.000000,
112586834561643540000000.000000,
67378029765916820000000.000000,
755534860921847030000000000000.000000,
4397611913739958700000.000000,
76209378028621039000000000000000000.000000
```

还有一个提示：

To solve the key, Maybe you know some interesting data format about C language?

数据类型？double？有什么用啊？

还有一个加密的压缩包，试了试爆破，没成功

直接找wp，原来考的是关于数据在内存中的存储（完全不会呢）

对wp中的解题代码做了一些修改：

```python
from Crypto.Util.number import *
import struct

s = [72143238992041641000000.000000, 771353571780065040000000000000000.000000, 11258683456164354000000000.000000,
6737802976591682000000.000000, 75553486092184703000000000000.000000, 4397611913739958700000.000000, 76209378028
62103900000000000000.000000]
a = ''
b = ''
for i in s:
    a += struct.pack('<f', i).hex()        # 小端
print(a)

for j in s:
    b += struct.pack('>f', j).hex()        # 大端
print(b)

print(long_to_bytes(int(a, 16)))
print(long_to_bytes(int(b, 16)))
```

wp中的注释“大端”“小端”（wp中写成两个“小端”了）还有`struct.pack('<f', i)`让我挺在意的
于是就找了struct库的官方文档，有这么一段话：

字节顺序，大小和对齐方式

默认情况下，C类型以机器的本机格式和字节顺序表示，并在必要时通过跳过填充字节进行正确对齐（根据C编译器使用的规则）。

或者，根据下表，格式字符串的第一个字符可用于指示打包数据的字节顺序，大小和对齐方式：

字符	字节顺序	大小	对齐方式
@	按原字节	按原字节	按原字节
=	按原字节	标准	无
<	小端	标准	无
>	大端	标准	无
!	网络（=大端）	标准	无

如果第一个字符不是其中之一，则假定为 '@'。

本机字节顺序可能为大端或是小端，取决于主机系统的不同。例如， Intel x86 和 AMD64 (x86-64) 是小端的；Motorola 68000 和 PowerPC G5 是大端的；ARM 和 Intel Itanium 具有可切换的字节顺序（双端）。 请使用 `sys.byteorder` 来检查你的系统字节顺序。

本机大小和对齐方式是使用 C 编译器的 `sizeof` 表达式来确定的。 这总是会与本机字节顺序相绑定。

https://blog.csdn.net/weixin_52446095

不难猜想，解题代码同时写了大端和小端是因为不同的计算机是不一样的
而后面的"f"是格式字符：

格式字符

格式字符具有以下含义；C 和 Python 值之间的按其指定类型的转换应当是相当明显的。 ‘标准大小’列是指当使用标准大小时以字节表示的已打包值大小；也就是当格式字符串以 '<'，'>'，'!' 或 '=' 之一开头的情况。 当使用本机大小时，已打包值的大小取决于具体的平台。

格式	C 类型	Python 类型	标准大小	注释

x	填充字节	无		
c	char	长度为 1 的字节串	1	
b	signed char	整数	1	(1), (2)
B	unsigned char	整数	1	(2)
?	_Bool	bool	1	(1)
h	short	整数	2	(2)
H	unsigned short	整数	2	(2)
i	int	整数	4	(2)
I	unsigned int	整数	4	(2)
l	long	整数	4	(2)
L	unsigned long	整数	4	(2)
q	long long	整数	8	(2)
Q	unsigned long long	整数	8	(2)
n	ssize_t	整数		(3)
N	size_t	整数		(3)
e	(6)	浮点数	2	(4)
f	float	浮点数	4	(4)

https://blog.csdn.net/weixin_52446095

那么什么是大端和小端呢？参考知乎文章：什么是大端序和小端序，为什么要有字节序？

字节的排列方式有两个通用规则:

- 大端序（Big-Endian）将数据的低位字节存放在内存的高位地址，高位字节存放在低位地址。这种排列方式与数据用字节表示时的书写顺序一致，符合人类的阅读习惯。
- 小端序（Little-Endian），将一个多位数的低位放在较小的地址处，高位放在较大的地址处，则称**小端序**。小端序与人类的阅读习惯相反，但更符合计算机读取内存的方式，因为CPU读取内存中的数据时，是从低地址向高地址方向进行读取的。

https://blog.csdn.net/weixin_52446095

简单来说一个是正着写（按照人类阅读习惯），一个是倒着写（符合计算机读取习惯）
解题代码输出结果为：

```
496e7465726573747269e67204964656120746f20656e6372797074
65746e497473657267e6972656449206f742061636e652074707972
b'Interestring Idea to encrypt'
b'etnItsergniredI ot acne tpyr'
```

得到压缩包密码：Interestring Idea to encrypt
解压之后得到一个加密程序：

```
import pyDes
import base64
from FLAG import flag
deskey = "********"
DES = pyDes.des(deskey)
DES.setMode('ECB')
DES.Kn = [
   [1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0],
   [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0],
   [0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0],
   [1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1],
   [0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
   [0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],
   [0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0],
   [0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0],
   [1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],
   [0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0],
   [0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1],
   [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0],
   [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0],
   [1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1],
   [1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1],
   [1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]
  ]
cipher_list = base64.b64encode(DES.encrypt(flag))
#b'vrkgBqeK7+h7mPyWujP8r5FqH5yyVlqv0CXudqoNHVAVdNO8ML4lM4zgez7weQXo'
```

照应了标题的crypto-des
DES和AES一样属于块加密，深究起来的话也很麻烦
这道题只要照着加密脚本写解密脚本就行了
解密代码如下：

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Time    : 7/27/2021 4:19 PM
# @Author  : Σ2333!
# @File    : solution.py
# @Software: PyCharm
import pyDes
import base64

deskey = "********"
DES = pyDes.des(deskey)
DES.setMode('ECB')
DES.Kn = [
   [1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0],
   [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0],
   [0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0],
   [1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1],
   [0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
   [0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],
   [0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0],
   [0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0],
   [1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],
   [0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0],
   [0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1],
   [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0],
   [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0],
   [1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1],
   [1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1],
   [1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]
  ]

# cipher_list = base64.b64encode(DES.encrypt(flag))
c = b'vrkgBqeK7+h7mPyWujP8r5FqH5yyVlqv0CXudqoNHVAVdNO8ML4lM4zgez7weQXo'
m = DES.decrypt(base64.b64decode(c))
print(m)
```

结语

希望继续坚持