

# 实验吧 部分逆向题解

原创

pipixia233333 于 2019-04-06 18:42:19 发布 933 收藏 1

分类专栏: 逆向之旅

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/qq\\_41071646/article/details/89056010](https://blog.csdn.net/qq_41071646/article/details/89056010)

版权



[逆向之旅 专栏收录该内容](#)

128 篇文章 2 订阅

订阅专栏

最近搞pwn 搞得很自闭 不想看pwn了 准备看点 逆向 准备刷一些实验吧的一些题 来安慰安慰我 pwn不懂的心情! ~~~~~

有些题 已经刷过了 我也就不写了 那些刷过的 也没有整理成一篇博客 比较可惜吧

recursive

这个题 看出来是python的了 然后 在主函数里面发现了东西

```
IDA View-A Pseudocode-A Hex View-1 Structures Enums Imports Exports
0 int v9; // eax
1 _BOOL8 v10; // rbx
2 int v11; // eax
3
4 v2 = a2;
5 v3 = (unsigned int)a1;
6 Py_FrozenFlag = 1;
7 if ( Py_IgnoreEnvironmentFlag || (a1 = "PYTHONINSPECT", (v5 = getenv("PYTHONINSPECT")) == 0LL) )
8     v4 = 0;
9 else
10    v4 = *v5 != 0;
11 if ( !Py_IgnoreEnvironmentFlag )
12 {
13     a1 = "PYTHONUNBUFFERED";
14     v6 = getenv("PYTHONUNBUFFERED");
15     if ( v6 )
16     {
17         if ( *v6 )
18         {
19             setbuf((FILE *)stdin, 0LL);
20             setbuf((FILE *)stdout, 0LL);
21             a1 = (const char *)stderr;
22             a2 = 0LL;
23             setbuf((FILE *)stderr, 0LL);
24         }
25     }
}
https://blog.csdn.net/qq_41071646
```

看起来应该是让 getenv 不等于0的 但是确实不知道怎么搞 百度了一波wp

[https://blog.csdn.net/qq\\_42192672/article/details/82990655](https://blog.csdn.net/qq_42192672/article/details/82990655)

大概明白了 是 环境变量

然后 用 其中Linux export命令用于设置或显示环境变量

然后一个一个分析就好 用ida 或者linux 的字符匹配求解

```
zsh: no matches found: flag{.*}

# pipixia @ ubuntu in ~/桌面 [8:58:30] C:1
$ strings ./unstep_f67baaeb | grep -o 'flag{.*}'
flag{python_taken_2_far}

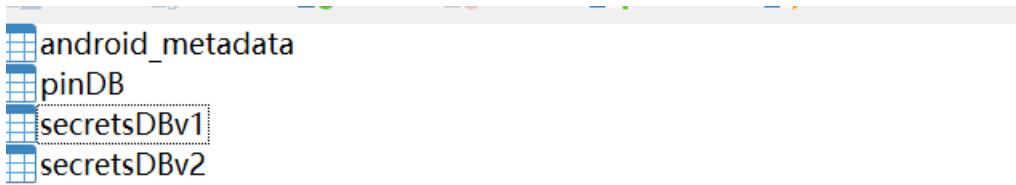
# pipixia @ ubuntu in ~/桌面 [8:58:40]
$
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

pinstore

这个题目太带劲了

还带数据库的 下了一个查看数据库的工具直接一把梭 看看是什么情况



可以看出很多的值

pinFromDB : d8531a519b3d4dfbece0259f90b466a23efc57b (SHA 1 解密结果 7498)

v1 : hcsvUnln5jMdw3GeI4o/txB5vaEf1PFAAnKQ3kPsRW2o5rR0a1JE54d0BLkzXPtqB

v2 : Bi528nDINBcX9BcCC+ZqGQo1Oz01+GOWSmvxRj7jg1g=

这里可以把值直接取出来

这个题 按道理来说应该有两种做法

第一种就是直接 把v1改成v2 让程序直接把flag打印出来

这样的做法 简单粗暴一把梭 我很喜欢 那么 下面我详细讲述第二种

第二种就是直接粘贴算法 然后模拟这个题目的算法

我们看一下他都做了什么

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView((int) R.layout.activity_main);
    Button button = (Button) findViewById(R.id.loginbutton);
    this.pinEditText = (EditText) findViewById(R.id.pinedittext);
    button.setOnClickListener(new OnClickListener() {
        public void onClick(View view) {
            String enteredPin = MainActivity.this.pinEditText.getText().toString();
            String pinFromDB = null;
            String hashOfEnteredPin = null;
            try {
                pinFromDB = new DatabaseUtilities(MainActivity.this.getApplicationContext()).fetchPin();
            } catch (IOException e) {
                e.printStackTrace();
            }
            try {
                hashOfEnteredPin = CryptoUtilities.getHash(enteredPin);
            } catch (NoSuchAlgorithmException e2) {
                e2.printStackTrace();
            } catch (UnsupportedEncodingException e3) {
                e3.printStackTrace();
            }
            if (pinFromDB.equalsIgnoreCase(hashOfEnteredPin)) {
                Intent intent = new Intent(MainActivity.this, SecretDisplay.class);
                intent.putExtra("pin", enteredPin);
                MainActivity.this.startActivity(intent);
                return;
            }
            MainActivity.this.pinEditText.setText("");
            Toast.makeText(MainActivity.this, "Incorrect Pin, try again", 1).show();
        }
    });
}
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

这里没啥用 直接输入 解密结果 7498 就可以进入下一个界面了

```
protected void SecretDisplay.onCreate(Bundle savedInstanceState)      //method@3fcf
{
    super.onCreate(savedInstanceState);
    this.setContentView(0x7f04001a);
    Context context = this.getApplicationContext();
    String pin = this.getIntent().getStringExtra("pin");
    DatabaseUtilities dbUtils = new DatabaseUtilities(this.getApplicationContext());
    CryptoUtilities cryptoUtils = new CryptoUtilities("v1", pin);
    tv.setText(cryptoUtils.decrypt(dbUtils.fetchSecret()));
    Toast.makeText(context, pin, 1);
    return;
}
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

这样看起来也没有做多少事情 分析里面的函数

```
public void CryptoUtilities.<init>(String version, String pin) //method@3fa2
{
    super();
    this.pin = pin;
    this.key = this.getKey(version);
    this.cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
    return;
}
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

```

public SecretKeySpec CryptoUtilities.getKey(String version) //method@3fa7
{
    MessageDigest md;
    SecretKeySpec keySpec;
    int iterations;
    byte[] salt;
    char[] pinArray;
    SecretKeyFactory secretKeyFactory;
    KeySpec ks;
    SecretKey secretKey;
    byte[] keyBytes = null;
    if (version.equalsIgnoreCase("v1")) {
        Log.d("Version", version);
        keyBytes = "t0ps3kr3tk3y".getBytes("UTF-8");
        md = MessageDigest.getInstance("SHA-1");
        keyBytes = md.digest(keyBytes);
        keyBytes = Arrays.copyOf(keyBytes, 16);
        keySpec = new SecretKeySpec(keyBytes, "AES");
    } else {
        Log.d("Version", version);
        iterations = 1000;
        salt = "SampleSalt".getBytes();
        pinArray = this.pin.toCharArray();
        secretKeyFactory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        ks = new PBEKeySpec(pinArray, salt, iterations, 128);
        secretKey = secretKeyFactory.generateSecret(ks);
        keySpec = new SecretKeySpec(secretKey.getEncoded(), "AES");
    }
    return keySpec;
}

```

[https://blog.csdn.net/tjq\\_41071646](https://blog.csdn.net/tjq_41071646)

最终解密算法。。。 重点是求出来key

```

public String decrypt(String ciphertext) throws Exception {
    byte[] ciphertextBytes = Base64.decode(ciphertext.getBytes(), 2);
    Log.d("Status", ciphertextBytes.toString());
    this.cipher.init(2, this.key);
    return new String(this.cipher.doFinal(ciphertextBytes), "UTF-8");
}

```

这里可以明显看出来 v1 v2 分开对待了

可以模拟一下这个算法

```
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.util.Arrays;
import java.util.Base64;
import java.util.Base64.Decoder;
import java.util.Base64.Encoder;
import java.util.Stack;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;

/*
 * v1: hcsvUnln5jMdW3GeI4o/txB5vaEf1PFAAnKQ3kPsRW2o5rR0a1JE54d0BLkzXPtqB
 * v2: Bi528nDlNBcX9BcCC+ZqGQo1Oz01+GOWSmvxRj7jg1g=
 */
 */

public class Main {

    public static void main(String[] args) throws NoSuchAlgorithmException, NoSuchPaddingException, UnsupportedEncodingException {
        int flag=2;
        SecretKeySpec key;
        byte[] ciphertextBytes=null;
        String pinFromDB="7498";
        Decoder decoder = Base64.getDecoder();
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        String dbv1="hcsvUnln5jMdW3GeI4o/txB5vaEf1PFAAnKQ3kPsRW2o5rR0a1JE54d0BLkzXPtqB";
        String dbv2="Bi528nDlNBcX9BcCC+ZqGQo1Oz01+GOWSmvxRj7jg1g=";
        if(flag==1)
        {
            ciphertextBytes = decoder.decode(dbv1.getBytes());
            key=new SecretKeySpec(Arrays.copyOf(MessageDigest.getInstance("SHA-1").digest("t0ps3kr3tk3y".getBytes()), 16));
        }
        else
        {
            ciphertextBytes = decoder.decode(dbv2.getBytes());
            byte[] salt = "SampleSalt".getBytes();
            key=new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(new PBEKeySpec("t0ps3kr3tk3y".toCharArray(), salt, 1000, 16)), "AES");
        }
        cipher.init(2,key);
        System.out.printf(new String(cipher.doFinal(ciphertextBytes)), "UTF-8");
    }
}
```

v1 的结果

```
27
28 public static void main(String[] args) throws NoSuchAlgorithmException {
29     int flag=1;
30     SecretKeySpec key;
31     byte[] ciphertextBytes=null;
32     String pinFromDB="7498";
33     Decoder decoder = Base64.getDecoder();
34     Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
35     String dbv1="hcsvUnln5jMdw3GeI4o/txB5vaEf1PFAAnKQ3kPsRW2c";
36     String dbv2="Bi528nDlNBcX9BcCC+ZqGQo10z01+GOWSmvxRj7jg1g";
37     if(flag==1)
```

#### Console

```
<terminated> main [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2019年7月17日 下午4:43:23)
```

Here is what the data will look like

v2

```
28 public static void main(String[] args) throws NoSuchAlgorithmException {
29     int flag=2;
30     SecretKeySpec key;
31     byte[] ciphertextBytes=null;
32     String pinFromDB="7498";
33     Decoder decoder = Base64.getDecoder();
34     Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
35     String dbv1="hcsvUnln5jMdw3GeI4o/txB5vaEf1PFAAnKQ3kPsRW2c";
36     String dbv2="Bi528nDlNBcX9BcCC+ZqGQo10z01+GOWSmvxRj7jg1g";
37     if(flag==1)
```

#### Console

```
<terminated> main [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2019年7月17日 下午4:43:48)
```

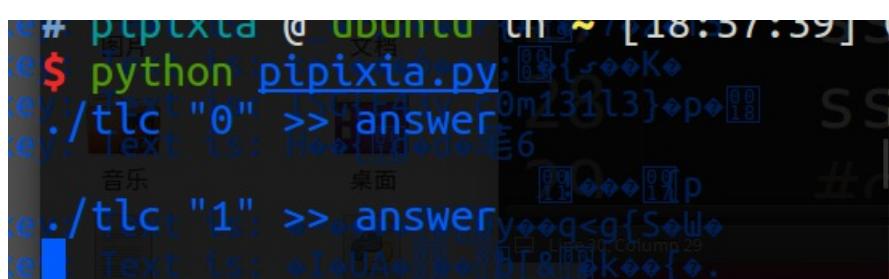
Flag:OnlyAsStrongAsWeakestLink

tcl

这个题目。。。真的比较狗了

我用的是暴力模拟 一开始的想法是用 os.system() 但是，，

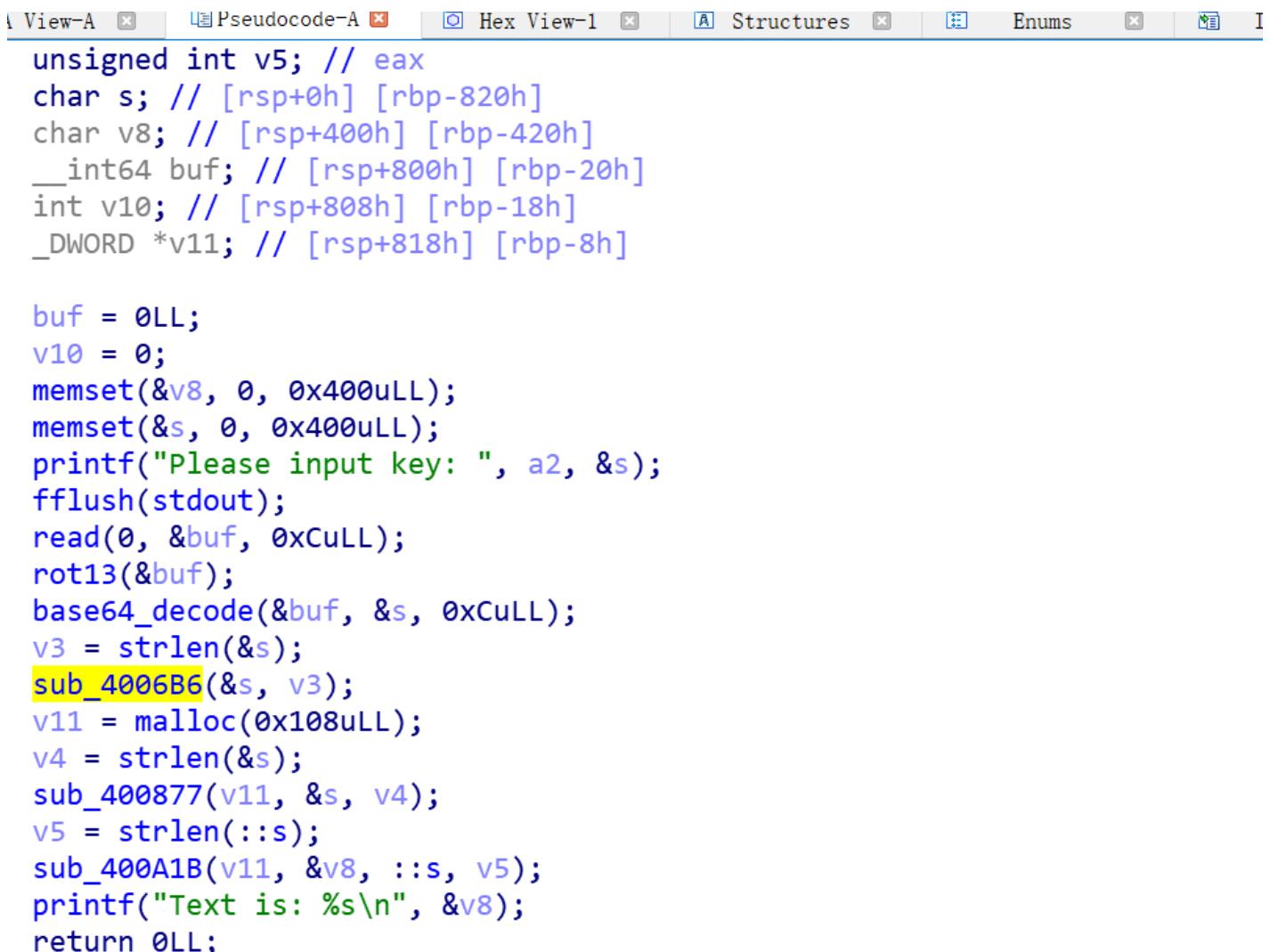
必须按回车才会进行下一步



```
# pipixia @ Ubuntu-18.04 ~ [18:57:59]
$ python pipixia.py
./tlc "0" >> answer
./tlc "1" >> answer
#
```

不清楚是怎么回事 如果有大佬知道 还请 指教。。

然后这个题目大概就是



The screenshot shows the Immunity Debugger interface with several tabs at the top: View-A, Pseudocode-A, Hex View-1, Structures, Enums, and Registers. The Pseudocode-A tab is active, displaying the following C-like pseudocode:

```
unsigned int v5; // eax
char s; // [rsp+0h] [rbp-820h]
char v8; // [rsp+400h] [rbp-420h]
__int64 buf; // [rsp+800h] [rbp-20h]
int v10; // [rsp+808h] [rbp-18h]
_DWORD *v11; // [rsp+818h] [rbp-8h]

buf = 0LL;
v10 = 0;
memset(&v8, 0, 0x400uLL);
memset(&s, 0, 0x400uLL);
printf("Please input key: ", a2, &s);
fflush(stdout);
read(0, &buf, 0xCuLL);
rot13(&buf);
base64_decode(&buf, &s, 0xCuLL);
v3 = strlen(&s);
sub_4006B6(&s, v3);
v11 = malloc(0x108uLL);
v4 = strlen(&s);
sub_400877(v11, &s, v4);
v5 = strlen(::s);
sub_400A1B(v11, &v8, ::s, v5);
printf("Text is: %s\n", &v8);
return 0LL;
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

其中

```
1 __int64 __fastcall sub_4006B6(char *a1, int a2)
2 {
3     __int64 result; // rax
4     int i; // [rsp+18h] [rbp-8h]
5     int v4; // [rsp+1Ch] [rbp-4h]
6
7     v4 = 0;
8     for ( i = 0; ; ++i )
9     {
10         result = i;
11         if ( i >= a2 )
12             break;
13         v4 = 10 * v4 + a1[i] - '0';
14         if ( v4 % (i + 1) )
15             exit(1);
16     }
17     return result;
18 }
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

这里是一个判断点，，

我们可以筛选这些数字 然后 进行暴力跑

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<vector>
#include<iostream>
#include<map>
#include<time.h>
#include<queue>
#include "windows.h"
using namespace std;
int pow(int n)
{
    int sum=0;
    while(n)
    {
        n/=10;
        sum++;
    }
    return sum;
}
int pows(int n)
{
    int ret=1;
    for(int i=0;i<n;i++)
    {
        ret*=10;
    }
    return ret;
}
```

```
j
// 0- 1000000000
bool kk(char *a1,int a2)
{
    int result;
    int i; // [rsp+18h] [rbp-8h]
    int v4; // [rsp+1Ch] [rbp-4h]
    v4 = 0;
    for ( i = 0; ; ++i )
    {
        result = i;
        if ( i >= a2 )
            break;
        v4 = 10 * v4 + a1[i] - '0';
        if ( v4 % (i + 1) )
        {
            printf("%d\n",i);
            return 0;
        }
    }
    return 1;
}

int pan(int n)
{
    int sum=0,i;
    int len=pow(n);
    int k=len;
    for(i=0;i<k;i++)
    {
        sum=sum*10+(n/pows(len-1)%10);
        len--;
        if(sum%(i+1))
        {
            return 0;
        }
    }
    return 1;
}
/*
243600
243606
243654
246000
246006
246054
246402
246408
246450
987654564
*/
int main()
{

FILE *fg=fopen("C:\\\\Users\\\\Lenovo\\\\Desktop\\\\1.txt","w");
if(fg==NULL)
{
    printf("打开文件失败! \\n");
    return 0;
}
```

```

}

int k=0;
for(int i=0;i<10000000000;i++)
{
    if(pan(i))
    {
        //printf("%d\n",i);
        fprintf(fg,"%d\n",i);
        k++;
    }
}
printf("sum:%d\n",k);
fclose(fg);
return 0;
}

```

然后 这里是python 脚本

```

# -*- coding:utf-8 -*-
import os
import base64
def rot(crypt_str,shift):
    crypt_list = list(crypt_str)
    plain_str = ""
    num = int(shift)
    for ch in crypt_list:
        ch = ord(ch)
        if ord('a') <= ch and ch <= ord('z'):
            ch = ch + num
            if ch > ord('z'):
                ch -= 26
        if ord('A') <= ch and ch <= ord('Z'):
            ch = ch +num
            if ch > ord('Z'):
                ch -= 26
        a=chr(ch)
        plain_str += a
    return plain_str

with open("flag.txt","r") as f:
    s=f.read()

s=s.split('\n')
for lists in s:
    ss=base64.b64encode(lists)
    ss=rot(ss,13)
    #command='./tlc "'+lists+'" >> answer'
    ret=os. popen('echo "'+ss +'" | ./tlc')
    ret_read=ret.read()
    for line in ret_read.splitlines():
        if line.find("{}")!=-1:
            print line

#os.system(command)

```

然后注释掉的 就是 system 不知道怎么回事，，，

然后 这个题目其实也可以把最后两个加密给模拟了 然后给直接算 这样会快点

但是 这里没有必要 因为数字够少。。 结果如下

debug

这个题 说起来惭愧

本来我以为 ida f5粘贴复制就行 但是我发现这个根本就不行 然后我进行了限制

j前面本来 我没有加(unsigned \_\_int8) 但是我发现结果跑的不对。。

加上去之后 就对了

太真实了。。

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<iostream>
using namespace std;
int main()
{
    int v3[5],k; // [esp+18h] [ebp-20h]
    puts("Printing flag");
    v3[0] = 0x1686F596;
    v3[1] = 0x5646F537;
    v3[2] = 0x76765726;
    v3[3] = 0x37F52756;
    v3[4] = 0xC6C696B6;
    for (int i = 0; i <= 4; ++i)
    {
        for ( int j= v3[i],k=0; k<4; j >>= 8,k++)
        {
            putchar(((unsigned __int8)j >> 4) | 16 * j);
        }
        //getchar();
    }

    putchar(10);
    return 0;
}
```

## catalyst-system

参考链接 <http://www.shiyanbar.com/ctf/writeup/5026>

如果要是知道linux 和windows 的随机数 竟然不一样

我也不不会。 .

思想了一下 应该是 windows linux 的随机策略不一样 毕竟 随机有很多 策略

阿西吧 比较可惜吧 这个题我先说分析到哪里

```
35}
36    putchar(10);
37    printf("Username: ");
38    __isoc99_scanf("%s", username);
39    printf("Password: ", username);
40    __isoc99_scanf("%s", password);
41    printf("Logging in", password);
42    fflush(stdout);
43    for ( j = 0; j <= 29; ++j )
44    {
45        v5 = rand();
46        sleep(v5 % (j + 1));
47        putchar(46);
48        fflush(stdout);
49    }
50    putchar(10);
51    checklen(username); // 这个函数限定了username 的内容
52    checkuser(username); // username字符串只能'a'-'z'或者'_'
53    checkusers(username);
54    check(username, password);
55    get_flag(username, password);
56    return 0LL;
57}
```

```
1 int64 __fastcall sub_400C41(int len)
2 {
3     int64 result; // rax
4
5     if ( 4 * (len >> 2) != len || 4 * (len >> 4) == len >> 2 || (result = (len >> 3), !result) || len >> 4 )//  
6         // 4*(len>>2)!=len    len是4的倍数  
7         //  
8         // len>>4==0  len<16  
9         //  
10        // len>>4 *4 ==len>>2  
11        //  
12        // len=8 /12
13    {
14        puts("invalid username or password");
15        exit(0);
16    }
17    return result;
18}
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

```
1 signed __int64 __fastcall checkuser(unsigned int *a1)
2 {
3     signed __int64 result; // rax
4     __int64 str[2]; // [rsp+10h] [rbp-20h]
5     __int64 str[1]; // [rsp+18h] [rbp-18h]
6     __int64 str[0]; // [rsp+20h] [rbp-10h]
7
8     str[0] = *a1;
9     str[1] = a1[1];
10    str[2] = a1[2];           // 在这里可以看的出来长度是 12
11    if ( str[0] - str[1] + str[2] != 0x5C664B56
12        || str[1] + 3 * (str[2] + str[0]) != 0x2E700C7B2LL
13        || (result = 0x32AC30689A6AD314LL, str[2] * str[1] != 0x32AC30689A6AD314LL) )
14    {
15        puts("invalid username or password");
16        exit(0);
17    }
18    return result;           // 可以列出方程式
19                                // str[0]-str[1]+str[2]==0x5C664B56
20                                // str[1] + 3 * (str[2] + str[0])==0x2E700C7B2LL
21                                // str[2] * str[1] != 0x32AC30689A6AD314LL
22                                // 可以解出
23                                // str[0]= 0x61746163
24                                // str[1]= 0x61746163
25                                // str[2]=0xF65635F
26}
27}
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

```
1 __int64 __fastcall checkusers(__int64 a1)
2 {
3     __int64 result; // rax
4     int i; // [rsp+1Ch] [rbp-4h]
5
6     for ( i = 0; ; ++i )
7     {
8         result = *(i + a1);
9         if ( !result )
10            break;
11         if ( (*(i + a1) <= ' ' || *(i + a1) > 'z') && *(i + a1) != '_' )
12         {
13             puts("invalid username or password");
14             exit(0);
15         }
16     }
17     return result;
18}
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

IDA View-A Pseudocode-A Hex View-1 Structures Enums Imports Exports

```
7 int v6; // ebx
8 int v7; // ebx
9 int v8; // ebx
.0 int v9; // ebx
.1 int v10; // ebx
.2 int v11; // ebx
.3 unsigned int v12; // ebx
.4 __int64 result; // rax
.5 int i; // [rsp+2Ch] [rbp-14h]
.6
.7 for ( i = 0; *(a2 + i); ++i )
.8 {
.9     if ( (*(a2 + i) <= 96 || *(a2 + i) > 122)
.0     && (*(a2 + i) <= 64 || *(a2 + i) > 90)
.1     && (*(a2 + i) <= 47 || *(a2 + i) > 57) )
.2     {
.3         puts("invalid username or password");
.4         exit(0);
.5     }
.6 }
.7 srand(a1[1] + *a1 + a1[2]);
.8 // password 只能0 -9 或者'a'-'z' 或者'A'-'Z'
.9 // 下面是 检查 password 和username的 内容是否符合要求
// srand(seek)
// seek=str[0]+str[1]+str[2]=0x454D3E2E
.0 v2 = *a2;
.1 if ( v2 - rand() != 1441465642 )
.2 {
.3     puts("invalid username or password");
.4     exit(0);
.5 }
.6 v3 = a2[1];
.7 if ( v3 - rand() != 251096121 )
.8 {
```

本来这里我感觉可以了

然后先把username 打印出来

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<iostream>
using namespace std;
int main()
{
    int username[4];
    username[0]= 0x61746163;
    username[1]= 0x61746163;
    username[2]=0x6F65635F;
    printf("%.12s",username);

    return 0;
}
```

```
catacata_ceo
Process returned 0 (0x0)   executable
```

然后在求 password 的时候发现出了问题

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main()
{
    unsigned int a[14];
    srand(0x454D3E2E); //前面解出来的x + y + z

    a[0] = 0x55EB052A + rand();
    a[1] = 0x0EF76C39 + rand();
    a[2] = 0xCC1E2D64 + rand();
    a[3] = 0xC7B6C6F5 + rand();
    a[4] = 0x26941BFA + rand();
    a[5] = 0x260CF0F3 + rand();
    a[6] = 0x10D4CAEF + rand();
    a[7] = 0xC666E824 + rand();
    a[8] = 0xFC89459C + rand();
    a[9] = 0x2413073A + rand();
    for(int i=0;i<10;i++)
        printf("%d:%x\n",i,a[i]);
    printf("%.40s",a);
    return 0;
}
```

```
0:55eb7477
1:ef76ea2
2:cc1e71d4
3:c7b7072b
4:2694477d
5:260d1223
6:10d52a08
7:c6671fa5
8:fc89b0ad
9:241307b5
*??g骗噏  } G?#□
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

我 ? ? ? ? ? ? ? ? ? ?

什么鬼

然后 看了那篇wp 我才知道我有那么 年轻。。

```

#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main()
{
    unsigned int a[14];
    srand(0x454D3E2E); //前面解出来的x + y + z

    a[0] = 0x55EB052A + rand();
    a[1] = 0x0EF76C39 + rand();
    a[2] = 0xCC1E2D64 + rand();
    a[3] = 0xC7B6C6F5 + rand();
    a[4] = 0x26941BFA + rand();
    a[5] = 0x260CF0F3 + rand();
    a[6] = 0x10D4CAEF + rand();
    a[7] = 0xC666E824 + rand();
    a[8] = 0xFC89459C + rand();
    a[9] = 0x2413073A + rand();

    unsigned char b[] = {0x42, 0x13, 0x27, 0x62, 0x41, 0x35, 0x6b, 0x0f, 0x7b, 0x46, 0x3c, 0x3e, 0x67, 0x0c, 0x
        for(int i=0;i<10;i++)
            printf("%d:%x\n",i,a[i]);
    char *aa = (char *)a;
    printf("password: %.40s",aa);
    printf("\n");
    printf("ALEXCTF{");
    for (int j = 0 ;j < 40 ;j++){
        printf("%c",b[j] ^aa[j]);
    }
    printf("}\n");
    return 0;
}

```

```

pipixia@pipixia-virtual-machine:~$ gcc -o a aa.c
pipixia@pipixia-virtual-machine:~$ ./a
0:56534c73
1:76345170
2:4763334b
3:38577957
4:5a694136
5:77676768
6:6a42484c
7:4339786d
8:56707352
9:6a676747
password: sLSVpQ4vK3cGWyW86AiZhggwLHBjmx9CRspVGggj
ALEXCTF{1_t41d_y0u_y0u_ar3_gr34t_reverser_533} https://log.csdn.net/qq_41071646

```

defcamp

```

7 unsigned __int64 v7; // [rsp+18h] [rbp-8h]
8
9 v7 = __readfsqword(0x28u);
10 for ( i = 1; i <= 10; ++i )
11 {
12     heap_point = (unsigned int *)malloc(16uLL);
13     *heap_point = i;
14     *((_BYTE *)heap_point + 4) = *heap_point + 109;
15     a3 = (char **)qword_601080;
16     *((_QWORD *)heap_point + 1) = qword_601080;
17     qword_601080 = (__int64)heap_point;
18 }                                // qword_601080 为上一个的堆块指针
19                                         // 每个堆块 都会有一个指针域指向下一个堆块
20                                         // 单向链表
21 printf("Enter the password: ", a2, a3);
22 if ( !fgets(&s, 7, stdin) )
23     return 0LL;
24 if ( (unsigned int)check((__int64)&s) )
25 {
26     puts("Incorrect password!");
27     result = 1LL;
28 }
29

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

Pseudocode-A Stack of check IDA View-A Hex View-1 Structures Enums Imports E

```

23 v12 = 2;
24 v13 = 5;
25 v14 = 6;
26 for ( i = 0; i <= 5; ++i )
27 {
28     v5 = qword_601080;                                // 初始化链表
29     v4 = 0;
30     while ( v5 )                                     // v5+4
31     {
32         // *heap_point = i;
33         // *((_BYTE *)heap_point + 4) = *heap_point + 109;
34         // 为了让 v9 的值等于那个值
35         // 输入的值 必须等于 i+109
36         // 即是 v9+109
37         if ( *((_BYTE *)v5 + 4) == *((_BYTE *)i + a1) )
38         {
39             v4 = *(DWORD *)v5;
40             break;
41         }
42         v5 = *(QWORD *)(v5 + 8);                         // 循环链表
43     }
44     *((DWORD *)&v6 + i) = v4;
45 }
46 for ( j = 0; j <= 5; ++j )
47 {
48     if ( *((DWORD *)&v6 + j) != (&v9 + j) )    // 判断点
49     {
50         return 1LL;
51     }
52 }
53 return 0LL;
54

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main()
{
    int s[6];
    for(int i=0;i<6;i++)
        scanf("%d",&s[i]);
    printf("\n");

    getchar();
    for(int i=0;i<6;i++)
        printf("%c",s[i]);
    getchar();
    return 0;
}
```

```
114 111 116 111 114 115
```

```
rotors
```

```
reversemeplz
```

这个题目有点好玩

```

1 int __cdecl main(int a1)
2 {
3     char dest; // [esp+0h] [ebp-118h]
4     char s; // [esp+80h] [ebp-98h]
5     int *v4; // [esp+10Ch] [ebp-Ch]
6
7     v4 = &a1;
8     gets(&s);
9     if ( sub_8048801(&s) )
10    {
11         strcpy(&dest, "The flag is flag{");
12         strcat(&dest, &s);
13         strcat(&dest, "}");
14         puts(&dest);
15     }
16     else
17     {
18         puts("Wrong input.");
19     }
20     return 0;
21 }

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

就一个验证函数

```

13
14 qmemcpy(&v11[1], &unk_8048980, 0x3Cu);
15 v1 = 0;
16 v2 = 0;
17 do
18 {
19     if ( a1[v2] <= 96 )
20         a1[v2] = sub_8048519(a1[1] & 1);
21     if ( a1[v2] > 122 )
22         a1[v2] = sub_8048519(a1[1] & 2);
23     v3 = sub_8048519(a1[v2]);
24     *(&v9 + v2) = v3;
25     if ( v3 > 0xCCu && v3 != -49 )
26         v1 = 1;
27     ++v2;
28 }
29 while ( v2 != 15 );
30 v4 = 0;
31 if ( v1 != 1 )
32 {
33     while ( 1 )
34     {
35         ++v4;
36         if ( *(&v9 + v4) - *(&v8 + v4) != v11[v4] )
37             break;
38         if ( v4 == 14 )

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

其中 sub\_8048519 是一个映射函数 把一个字符映射成另一个字符

然后我们找一下密文

密文

```
30 v4 = 0;
31 if ( v1 != 1 )
32 {
33     while ( 1 )
34     {
35         +v4;
36         if ( *(&v9 + v4) - *(&v8 + v4) != v11[v4] )
37             break;
38         if ( v4 == 14 )
39         {
40             if ( a1[15] )
41             {
42                 v6 = v10;
43                 for ( i = v9; i != -52; i ^= v6-- )
44                     ;
45                 return 0;
46             }
47             if ( sub_8048519(0) )
48                 return 0;
49             return sub_8048519(*a1) == 'b';
50         }
51     }
52 }
53 return 0;
54 }
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

这里就是验证密文的 然后 密文第一位就是b

然后我们求出密文 然后 把可见字符都映射一下 做成一个表 然后 把密文映射一下就可以了

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<vector>
#include<iostream>
#include<map>
#include<time.h>
#include<queue>
#include "windows.h"
using namespace std;
map<char,char>mapp;
int dword_8049CA4;
char sub_80484FB(char a1)
{
    ++dword_8049CA4;
    ++dword_8049CA4;
    return a1;
}
int sub_8048519(char a1)
{
    unsigned __int8 v1; // al
    signed int v2; // edi
    int v3; // ebx
    signed int v4; // edi
    int v5; // ebx
    signed int v6; // edi
```

```
signed int v0; // edi
int v7; // ebx
signed int v8; // edi
char v9; // cl
int v10; // edx
signed int v11; // edi
int v12; // edx
int v13; // edx
signed int v14; // edi
signed int v15; // ecx
int v16; // edx
signed int v17; // ecx
int v18; // edx
signed int v19; // ecx
int v20; // edx
signed int v21; // ebx
int v22; // edx
signed int v23; // ebx
int v24; // edx
signed int v25; // ebx
int v26; // edx
signed int v27; // ebx
int v28; // edx
signed int v29; // ecx
int v30; // edx
signed int v31; // ecx
int v32; // edx
signed int v33; // ecx
signed int v34; // esi
int v35; // edx
char v36; // cl
int v37; // edx
signed int v38; // esi
int v39; // edx
int v40; // edx
signed int v41; // edi
int v42; // edx
signed int v43; // edi
int v44; // edx
signed int v45; // ecx
signed int v46; // esi
int v47; // edx
int v48; // esi
bool v49; // zf
signed int v50; // eax
char v52; // [esp+4h] [ebp-10h]

v1 = sub_80484FB(a1);
v2 = 19;
if ( (v1 & 0x3F) != 38 )
    v2 = 0;
v3 = v2 | (v1 << 8) | 9 * ((v1 & 0x5F) == 86);
v4 = 71;
if ( (v1 & 0x77) != 116 )
    v4 = 0;
v5 = v4 | v3;
v6 = 84;
if ( (v1 & 0x3F) != 39 )
    v6 = 0;
v7 = v6 | v5;
```

```

v8 = 48;
if ( (v1 & 0x4F) != 4 )
    v8 = 0;
v9 = v1 & 0x1F;
v10 = 3 * ((v1 & 0x57) == 80) | 8 * (v9 == 1) | v7 | v8 | 2 * (v9 == 15) | 2 * ((v1 & 0x5B) == 83);
v11 = 114;
v52 = ~v1;
v12 = 8 * (v9 == 2) | 8 * (v9 == 11) | v10 | 2 * ((v1 & 0x57) == 66) | 8 * ((v1 & 0x2E) == 44);
if ( (v1 & 0x37) != 37 )
    v11 = 0;
v13 = v11 | v12;
v14 = 16;
v15 = 0;
if ( (v1 & 0x1C) == 8 )
    v15 = 16;
v16 = ((~v1 & 0x78u) < 1 ? 0x48 : 0) | v15 | v13;
v17 = 64;
if ( (v1 & 0x1D) != 16 )
    v17 = 0;
v18 = v17 | v16;
v19 = 0;
if ( (v1 & 0xF) == 11 )
    v19 = 16;
v20 = 4 * ((v1 & 0x55) == 64) | v19 | v18;
v21 = 72;
if ( (v1 & 0x4B) != 1 )
    v21 = 0;
v22 = v21 | v20;
v23 = 24;
if ( (v1 & 0x47) != 1 )
    v23 = 0;
v24 = v23 | v22;
v25 = 96;
if ( (v1 & 0x2B) != 34 )
    v25 = 0;
v26 = ((v52 & 0x55u) < 1 ? 0x48 : 0) | v25 | v24;
v27 = 0;
if ( (v1 & 0x31) == 16 )
    v27 = 16;
v28 = v27 | v26;
v29 = 0;
if ( (v1 & 0x55) == 81 )
    v29 = 68;
v30 = v29 | v28;
v31 = 0;
if ( (v1 & 0xE) == 8 )
    v31 = 32;
v32 = v31 | v30;
v33 = 97;
if ( (v1 & 0x59) != 72 )
    v33 = 0;
v34 = 81;
v35 = v33 | v32;
v36 = v1 & 0x17;
if ( (v1 & 0x17) != 4 )
    v34 = 0;
v37 = v34 | v35;
v38 = 37;
if ( (v1 & 0x47) != 66 )
    v38 = 0;

```

```

v38 = 0;
v39 = v37 | v38 | 8 * ((v1 & 0x43) == 2);
if ( (v1 & 0x46) != 2 )
    v14 = 0;
v40 = v14 | v39;
v41 = 80;
if ( v36 != 3 )
    v41 = 0;
v42 = v41 | v40;
v43 = 70;
if ( v36 != 1 )
    v43 = 0;
v44 = v43 | v42;
v45 = 40;
if ( (v1 & 0x70) != 64 )
    v45 = 0;
v46 = 0;
v47 = 4 * ((v1 & 0x41) == 1) | ((v52 & 0xBu) < 1 ? 0x60 : 0) | v45 | v44;
if ( (v1 & 0x48) == 64 )
    v46 = 32;
v48 = v47 | v46;
v49 = (v1 & 0x21) == 1;
v50 = 0;
if ( v49 )
    v50 = 68;
return v48 | v50;
}

```

```

//bargjbgursyntlb
int main()
{
    //求出密文...
//    int dword_8048980[] = {0xFFFFFFFF, 0x11, 0xFFFFFFFF5, 3, 0xFFFFFFFF8, 5, 0xE,
//                           0xFFFFFFFFD, 1, 6, 0xFFFFFFFF5, 6, 0xFFFFFFFF8, 0xFFFFFFFF6,
//                           0};
//    int l='b';
//    printf("b");
//    for(int i=0;i<14;i++)
//    {
//        l+=dword_8048980[i];
//        printf("%c",l);
//    }
    char s[]="bargjbgursyntlb";
    char ll;
    char k[]="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    for(int i=0;i<strlen(k);i++)
    {
        ll=sub_8048519(k[i]);
        mapp[k[i]]=ll;
    }
    printf("flag{");
    for(int i=0;i<strlen(s);i++)
    {
        printf("%c",mapp[s[i]]);
    }
    printf("}");
    return 0;
}

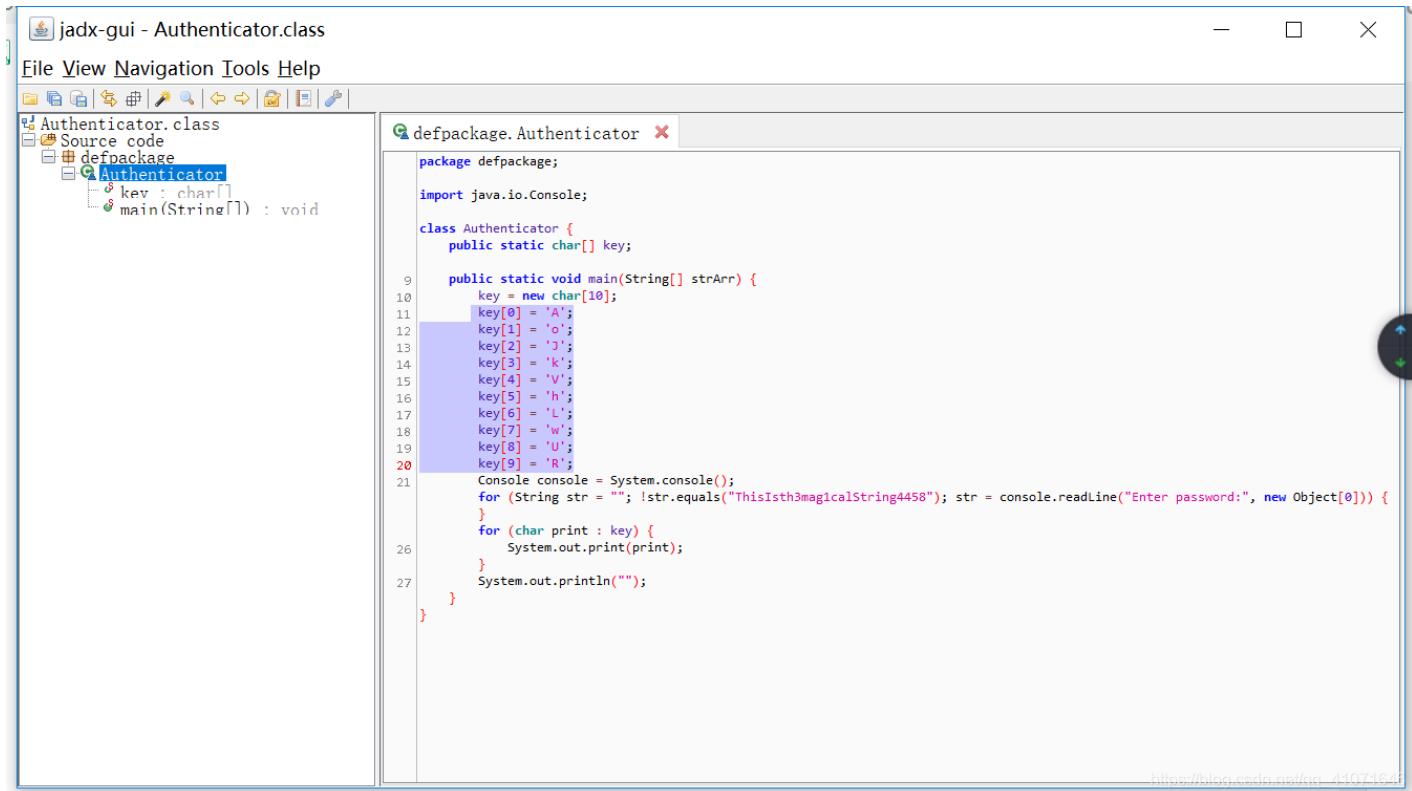
```

得出flag

```
选择D:\ConsoleApplication2\asd\bin\Debug\asd.exe
flag{onetwotheflagyo}
Process returned 0 (0x0) execution time : 0.315 s
Press any key to continue.
```

Byte Code 这个题 也是没有什么好讲的 这个题很简单

.class 多半就是java的 然后出现了



The screenshot shows the jadx-gui interface with the file 'Authenticator.class' selected in the left sidebar. The main window displays the decompiled Java code for the 'Authenticator' class:

```
package defpackage;
import java.io.Console;

class Authenticator {
    public static char[] key;

    public static void main(String[] strArr) {
        key = new char[10];
        key[0] = 'A';
        key[1] = 'o';
        key[2] = 'J';
        key[3] = 'k';
        key[4] = 'V';
        key[5] = 'h';
        key[6] = 'L';
        key[7] = 'w';
        key[8] = 'U';
        key[9] = 'R';
        Console console = System.console();
        for (String str = ""; str.equals("ThisIsTheMagicalString4458"); str = console.readLine("Enter password:", new Object[0])) {
        }
        for (char print : key) {
            System.out.print(print);
        }
        System.out.println("");
    }
}
```

然后看程序逻辑 应该就是 如果 输入的字符串等于 ThisIsTheMagicalString4458 就会输入key

那么 key 就是我们的flag。

## bitwise

一开始我想的是这个题 暴力跑 可是我发现了 &255 还有 左移右移 的和是 8 我感觉这个算法 还是可以逆一下的

#255 ==11111111 最后结果取 8个二进制 那么

#左移和右移 加起来是8 那么也就相当于 转个圈 最后在和 最后三位 异或一下

#那么逆推回去就是 先异或 111 然后 >>改成<< <<改成>> 即可

这个是我看源代码 给分析出来的

```
1 #coding:utf-8
2 if __name__ == '__main__':
3     verify_arr = [193, 35, 9, 33, 1, 9, 3, 33, 9, 225]
4     strs=""
5     for i in verify_arr:
6         i=i^111
7         temp=((i>>5)|(i<<3))&255
8         strs+=chr(temp)
9     print(strs)
10
```

运行 wp C:\Users\LENOVO\appdata\local\temp\ams\y\th\o\i\y\th\o\i\py\th\o\m\ e\c\ C:\Users\LENOVO\Desktop  
ub3rs3cr3t https://blog.csdn.net/qq\_41071646

这是跑出来的结果和脚本

```
#coding:utf-8
if __name__ == '__main__':
    verify_arr = [193, 35, 9, 33, 1, 9, 3, 33, 9, 225]
    strs=""
    for i in verify_arr:
        i=i^111
        temp=((i>>5)|(i<<3))&255
        strs+=chr(temp)
    print(strs)
```

这题算是很简单的

逆向观察

这个题就更简单了。

```

unsigned __int64 v10; // [rsp+68h] [rbp-18h]

v10 = __readfsqword(0x28u);
if ( argc <= 1 )
{
    puts("usage ./rev50 password");
}
else |
{
    src = 'sedecrem';
    v6 = 0;
    v7 = 0;
    v8 = 0;
    memcpy(&dest, &src, 9uLL);
    for ( i = 0; i <= 999; ++i )
    {
        if ( !strcmp(argv[1], (&dict)[i]) && !strcmp(&dest, (&dict)[i]) )
        {
            puts("Good password ! ");
            goto LABEL_10;
        }
    }
    puts("Bad ! password");
}
LABEL_10:
puts(&byte_40252A);

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

逻辑是如此 然后可能不知道 dict 是干啥的

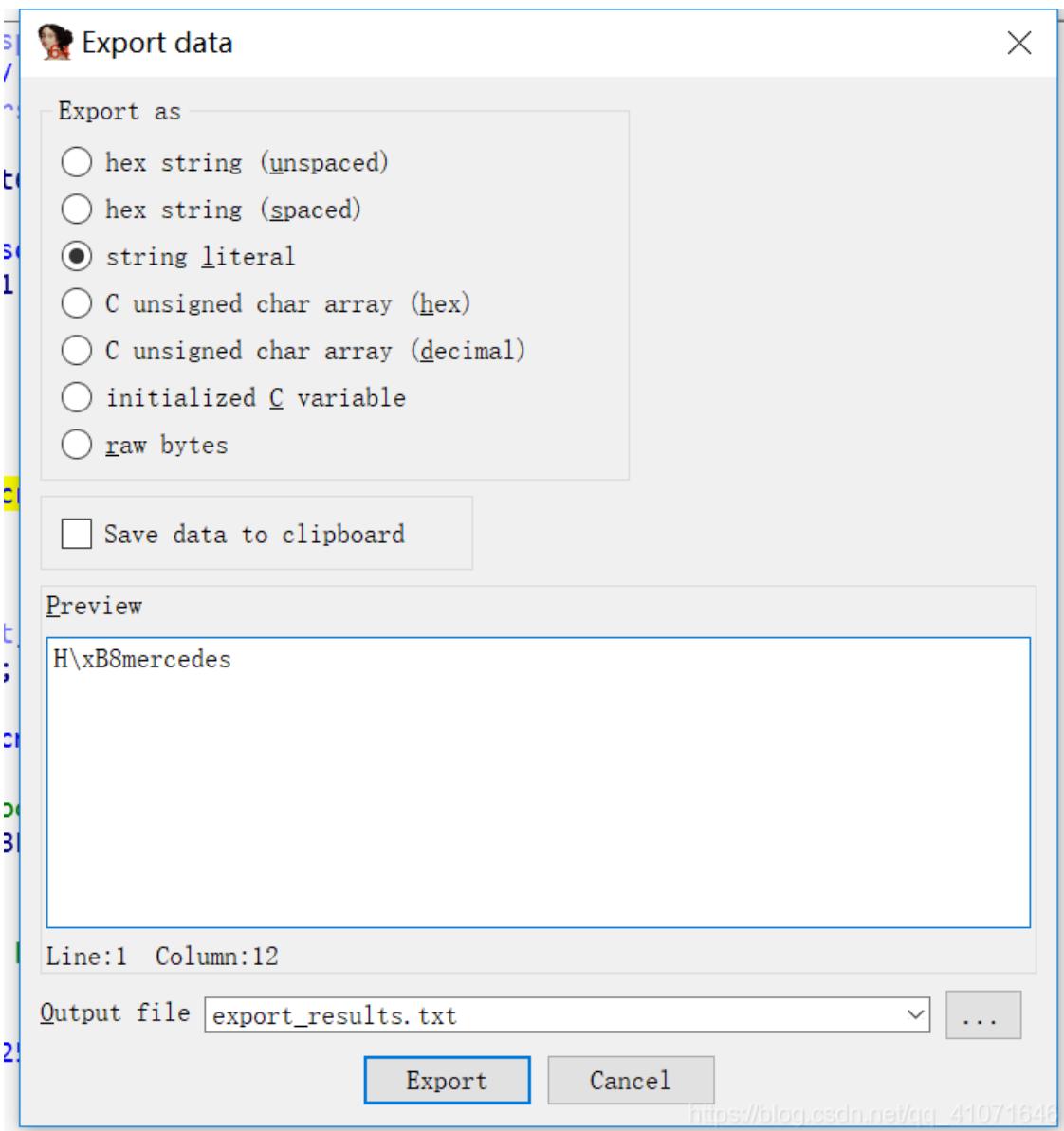
• .data:0000000000603080 dict	dq offset a123456	; DATA XREF: main+76↑o ; main+A8↑o ; "123456"
• .data:0000000000603080		"password"
• .data:0000000000603088	dq offset aPassword	; "12345678"
• .data:0000000000603090	dq offset a12345678	; "qwerty"
• .data:0000000000603098	dq offset aQwerty	; "123456789"
• .data:00000000006030A0	dq offset a123456789	; "12345"
• .data:00000000006030A8	dq offset a12345	; "1234"
• .data:00000000006030B0	dq offset a1234	; "111111"
• .data:00000000006030B8	dq offset a111111	; "1234567"
• .data:00000000006030C0	dq offset a1234567	; "dragon"
• .data:00000000006030C8	dq offset aDragon	; "123123"
• .data:00000000006030D0	dq offset a123123	; "baseball"
• .data:00000000006030D8	dq offset aBaseball	; "abc123"
• .data:00000000006030E0	dq offset aAbc123	; "football"
• .data:00000000006030E8	dq offset aFootball	; "monkey"
• .data:00000000006030F0	dq offset aMonkey	; "letmein"
• .data:00000000006030F8	dq offset aLetmein	; "696969"
• .data:0000000000603100	dq offset a696969	; "shadow"
• .data:0000000000603108	dq offset aShadow	; "master"
• .data:0000000000603110	dq offset aMaster	; "666666"
• .data:0000000000603118	dq offset a666666	; "qwertyuiop"
• .data:0000000000603120	dq offset aQwertyuiop	; "123321"
• .data:0000000000603128	dq offset a123321	; "mustang"
• .data:0000000000603130	dq offset aMustang	

0000307E 000000000060307E: .data:000000000060307E (Synchronized with Hex View-1)

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

这。。。 应该是一个数组 然后 暴力跑一遍???????? 不过反正我们输入的字符 和他们一样就行了 至于为什么 把我们转化的字符给反转 我感觉应该用ida的导出数据 好一点 不要轻易的 转化就直接用

因为我一开始 就犯了这个错误。。。



把前面的去掉就可以了

mercedes

回答正确。

1000

这个题 我是尽力了 这道题 做的我怀疑人生 我看着这个flag 不对 想着flag 也不会写到 文本里面 结果人家 还真的是 然后在后面添加一个Y就行。。。

1000 要不 是二进制 要不就是10进制 这个我是猜的出来了

但是后面的 我是真的 想不出来竟然是这个样子。 我断下了按钮

获得了 输入框 最后 断下了 writefile。。

唉

吾爱破解 - 1000.exe - [LCG - 主线程, 模块 - 1000]

文件(V) 查看(V) 调试(D) 插件(O) 选项(O) 窗口(W) 帮助(H) [+ 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder CMD Exit

寄存器 (FPU)

EAX	00728288	ASCII "G00dTh1sIske"
ECX	0019F338	
EDX	00600000	
EBX	00000002	
ESP	0019F324	ASCII "竞"
EBP	0019F360	
ESI	00000001	
EDI	0019F388	

EIP 00401C5A 1000.00401C5A

C 0	ES 002B	32位 0(FFFFFFFF)
P 1	CS 0023	32位 0(FFFFFFFF)
A 0	SS 002B	32位 0(FFFFFFFF)
Z 0	DS 002B	32位 0(FFFFFFFF)
S 0	FS 0053	32位 389900(FFF)
T 0	GS 002B	32位 0(FFFFFFFF)
D 0		

O 0 LastErr ERROR\_ALREADY\_EXISTS (00000007)

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)

ST0 empty 0.0  
ST1 empty 0.0  
ST2 empty 0.0  
ST3 empty 0.0  
ST4 empty 0.0  
ST5 empty 1.0000000000000000000000000000000

```

00401C2A 56 push esi
00401C2B 8D4D D8 lea ecx,dword ptr ss:[ebp-0x28]
00401C2E C645 FC 02 mov byte ptr ss:[ebp-0x4],0x2
00401C32 E8 27210700 call 1000.00401C0E
00401C37 83F8 01 cmp eax,0x1
00401C39 75 37 jnz short 1000.00401C73
00401C3C 885D 8C mov ebx,dword ptr ss:[ebp+0xC]
00401C3F 8BF0 mov esi,eax
00401C41 3BF3 cmp esi,ebx
00401C43 7D 1D jge short 1000.00401C62
00401C45 8D8C 0F lea ecx,dword ptr ds:[esi+esi*2]
00401C48 8B048F mov eax,dword ptr ds:[edi+ecx*4]
00401C4B 8B48 04 mov ecx,dword ptr ds:[eax+0x4]
00401C4E 83C0 08 add eax,0x8
00401C51 85C9 test ecx,ecx
00401C53 7E 0A jle short 1000.00401C5F
00401C55 51 push ecx
00401C56 50 push eax
00401C57 8D4D D8 lea ecx,dword ptr ss:[ebp-0x28]
00401C5A E8 56220700 call 1000.00473EB5
00401C5F 46 inc esi
00401C60 ^ EB DF jmp short 1000.00401C41
00401C62 8D4D D8 lea ecx,dword ptr ss:[ebp-0x28]

```

eax=00728288, (ASCII "G00dTh1sIske")

G00dTh1sIskeY

CTF{G00dTh1sIskeY}

提交

flag.txt - 记事本

文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

G00dTh1sIske

https://blog.csdn.net/cq\_41071635

迷路

太坑了 花里胡哨的

前面过的 花里胡哨 但是到后面就感觉差了很多。。。。。 是真的感觉不行啦

怎么说呢 出题人太骚了 特别是那个 整易语言 我都有点受不了 更别说其他的了 我都不知道 那些大佬是怎么发现这些骚套路的

这个题 我还以为是走方格什么的 ..... 结果不是 .... 我硬撸 算法 发现失败了.....

然后 参考链接

[https://blog.csdn.net/qq\\_40827990/article/details/83212779](https://blog.csdn.net/qq_40827990/article/details/83212779)

没错

我又看别人的博客了 呜呜呜 我哭了 没有想到 还会有一个按钮

具体看按钮情况 可以用

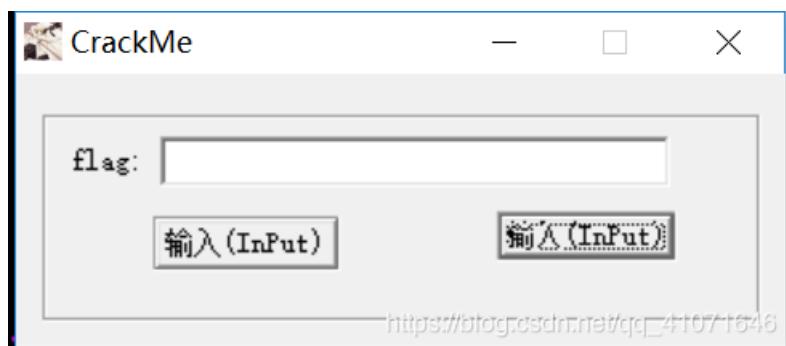
ShowWindow

来断点 看参数信息

```
0019F5F8 00412782 CALL 到 ShowWindow 来自 crackme_.0041277C
0019F5FC 00120FC4 hWnd = 00120FC4 ('输入(InPut)', class='Button', parent=001
0019F600 00000000 ShowState = SW_HIDE
```

点击 0 改成1就好

```
0019F5F8 00412782 CALL 到 ShowWindow 来自 crackme_.0041277C
0019F5FC 00120FC4 hWnd = 00120FC4 ('输入(InPut)', class='Button', parent=0
0019F600 00000001 ShowState = SW_SHOWNORMAL
0019F604 00412627 返回到 crackme_.0041277C 来自 crackme_.0041276E
```



然后 就多出了 哪一个按钮 断点 然后 静态分析就好

00401F8F	E8 55210100	call crackme_.004140E9
00401F94	8BCE	mov ecx,esi
00401F96	E8 A5F7FFFF	call crackme_.00401740
00401F9B	85C0	test eax,eax
00401F9D	74 33	je short crackme_.00401FD2
00401F9F	51	push ecx
00401FA0	8D56 60	lea edx,dword ptr ds:[esi+0x60]
00401FA3	8BCC	mov ecx,esp
00401FA5	896424 08	mov dword ptr ss:[esp+0x8],esp
00401FA9	52	push edx
00401FAA	E8 3A210100	call crackme_.004140E9
00401FAF	8BCE	mov ecx,esi
00401FB1	E8 AAF8FFFF	call crackme_.00401860
00401FB6	85C0	test eax,eax
00401FB8	74 18	je short crackme_.00401FD2
00401FBA	51	push ecx
00401FBB	83C6 64	add esi,0x64
00401FBE	8BCC	mov ecx,esp
00401FC0	896424 08	mov dword ptr ss:[esp+0x8],esp
00401FC4	56	push esi
00401FC5	E8 1F210100	call crackme_.004140E9
00401FCA	E8 E1FBFFFF	call crackme_.00401B80
00401FCF	83C4 04	add esp,0x4
00401FD0	EE	ret

这里直接就错误了

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

00401D6B	49	dec ecx
00401D6C	3BD1	cnp edx,ecx
00401D6E	72 E3	jb short crackme_.00401D53
00401D70	8D4C24 28	lea ecx,dword ptr ss:[esp+0x28]
00401D74	8D5424 08	lea edx,dword ptr ss:[esp+0x8]
00401D78	51	push ecx
00401D79	68 4CF14100	push crackme_.0041F14C
00401D7E	52	push edx
00401D7F	E8 4CD40000	call crackme_.0040F7D0
00401D84	8B4424 68	mov eax,dword ptr ss:[esp+0x68]
00401D88	8B4C24 14	mov ecx,dword ptr ss:[esp+0x14]
00401D8C	50	push eax
00401D8D	51	push ecx
00401D8E	E8 730F0000	call crackme_.00402D06
00401D93	83C4 14	add esp,0x14
00401D96	85C0	test eax,eax
00401D98	5F	pop edi
00401D99	53	push ebx
00401D9A	53	push ebx
00401D9B	75 87	jb short crackme_.00401D04
00401D9D	8D5424 10	lea edx,dword ptr ss:[esp+0x10]
00401D9F	52	push edx
00401DA1	E8 05	jmp short crackme_.00401D09
00401DA2	00401D09	ret

寄存器 (FPU)

EAX 0041F3F4 crackme\_.0041F3F4  
 ECX 026E5D08 ASCII "b5h760h64R867618bBwB48BrW92H4w5r"  
 EDX 026E5D09 ASCII "5h760h64R867618bBwB48BrW92H4w5r"  
 EBX 00000000  
 ESP 0019F3B0  
 EBP 0019F430  
 ESI 0019FE04 ASCII "热R"  
 EDI 0019F3D0  
 EIP 00401D8C crackme\_.00401D8C

C 0 ES 002B 32位 0(FFFFFFF)  
 P 1 CS 0023 32位 0(FFFFFFF)  
 A 0 SS 002B 32位 0(FFFFFFF)  
 Z 1 DS 002B 32位 0(FFFFFFF)  
 S 0 FS 0053 32位 254000(FFF)  
 T 0 GS 002B 32位 0(FFFFFFF)  
 D 0  
 O 0 LastErr ERROR\_SUCCESS (00000000)  
 EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)  
 ST0 empty 0.0  
 ST1 empty 0.0  
 ST2 empty 0.0  
 ST3 empty 0.0  
 ST4 empty 0.0  
 ST5 empty 12.0000000000000000 https://blog.csdn.net/qq\_41071646

发现这里都是直接跑飞的点 断下来 ida 静态分析一波

```

1 signed int __thiscall sub_401740(char *this, _BYTE *input)
2 {
3     char *v2; // esi
4     signed int result; // eax
5
6     v2 = this;
7     if ( *((_DWORD *)input - 2) < 0x27u
8         || *input != 'O'
9         || input[1] != 'O'
10        || input[2] != 'C'
11        || input[3] != 'T'
12        || input[4] != 'F'
13        || input[5] != '{'
14        || input[38] != '}' )
15     {
16         sub_414374(&input);
17         result = 0;
18     }
19     else
20     {
21         sub_40F42B((CString *)&input, 38, 1);
22         sub_40F42B((CString *)&input, 0, 6);
23         if ( *((_DWORD *)input - 2) )
24             sub_4143CB((CString *)(&v2 + 96), (int)&input);
25         CString::Empty((CString *)(&v2 + 92));
26         sub_414374(&input);
27         result = 1;
28     }
29     return result;
30 }

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

第一个函数限定了他们的flag格式

第二个就是算法了

像 v5 还有 v6 都可以调试调出来

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

b5h760h64R867618bBwB48BrW92H4w5r

那么可以写一个脚本出来

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<string.h>
#include<iostream>
using namespace std;
char s[]="b5h760h64R867618bBwB48BrW92H4w5r";
int main()
{
    int temp;
    printf("OCTF{");
    for(int i=0;i<strlen(s);i++)
    {
        if(s[i]>=48&&s[i]<=57)
        {
            printf("%c",s[i]);
            continue;
        }
        if(s[i]>=65&&s[i]<=90)
        {
            temp=65;
        }
        else
        {
            temp=97;
        }
        for(int j=0;j<26;j++)
        {
            if((j*5+28)%26==s[i]-temp)
            {
                printf("%c",j+temp);
            }
        }
    }
    printf("}");
    return 0;
}
```

## NSCTF Reverse 400

这个题 看着有点像 python打包的exe文件 以前接触过这种题

工具下载地址

<https://sourceforge.net/projects/pyinstallerextractor/>

直接试着能不能给直接搞定

```
D:\谷歌下载>python pyinstxtractor.py Reverse400.exe
[*] Processing Reverse400.exe
[*] Pyinstaller version: 2.0
[*] Python version: 27
[*] Length of package: 2551284 bytes
[*] Found 14 files in CArchive
[*] Beginning extraction...please standby
[*] Found 194 files in PYZ archive
[+] Possible entry point: _pyi_bootstrap
[+] Possible entry point: carchive
[+] Possible entry point: pyi_rth_encodings
[+] Possible entry point: Revesre03
[*] Successfully extracted pyinstaller archive: Reverse400.exe
```

You can now use a python decompiler on the pyc files within the extracted directory

D:\谷歌下载>

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

哦吼 好像搞定了

然后打开文件夹

名称	修改日期	类型
carchive	2019/4/8 10:28	文件
iu	2019/4/8 10:28	文件
out00-PYZ.pyz	2019/4/8 10:28	Python
pyi_rth_encodings	2019/4/8 10:28	文件
python27.dll	2019/4/8 10:28	应用程
Revesre03	2019/4/8 10:28	文件
Revesre03.exe.manifest	2019/4/8 10:28	MAN
select.pyd	2019/4/8 10:28	PYD
struct	2019/4/8 10:28	文件
unicodedata.pyd	2019/4/8 10:28	PYD

选定我光标的那个文件 然后 看一下文件都有什么内容

```

data = \
"\x1c\x7a\x16\x77\x10\x2a\x51\x1f\x4c\x0f\x5b\x1d\x42\x2f\x4b\x7e\x4a\x7a\x4a\x7b" +\
"\x49\x7f\x4a\x7f\x1e\x78\x4c\x75\x10\x28\x18\x2b\x48\x7e\x46\x23\x12\x24\x11\x72" +\
"\x4b\x2e\x1b\x7e\x4f\x2b\x12\x76\x0b"

...
char buf[] = "flag:{NSCTF_md5098f6bcd4621d373cade4e832627b4f6}";

int _tmain(int argc, _TCHAR* argv[])
{
    printf("%d\n", strlen(buf));
    char key = '\x0b';
    buf[47] ^= key;
    for (int i = 1; i < 48; i++)
    {
        buf[48 - i - 1] ^= buf[48 - i];
    }

    return 0;
}

print "Revese it?????????"
```

https://blog.csdn.net/qq\_41071646

哇 直接爆flag 正当我有点开心的时候

突然发现。 这个flag 不对

然后 我看了一下后面的算法 逆向了一下

用buf 还有data 都试了一下 结果 data 的对了

```

#!/usr/bin/env python
# encoding: utf-8
data = \
"\x1c\x7a\x16\x77\x10\x2a\x51\x1f\x4c\x0f\x5b\x1d\x42\x2f\x4b\x7e\x4a\x7a\x4a\x7b" +\
"\x49\x7f\x4a\x7f\x1e\x78\x4c\x75\x10\x28\x18\x2b\x48\x7e\x46\x23\x12\x24\x11\x72" +\
"\x4b\x2e\x1b\x7e\x4f\x2b\x12\x76\x0b"
flag=""
for i in range(len(data)-1):
    flag+=chr(ord(data[i]) ^ ord(data[i+1]))

print(flag)
```

flag:{NSCTF\_md540012655af49e803c68e165c  
9e5e1d9d}

做出来这个题 真的是 巧合

## Keylead (ASIS CTF 2015)

这个题应该算是简单的了 大致看了一下 我们输入的数据和解密的函数没有关系 那么直接修改rip 跳到解密函数即可

(sub\_4006B6) 动态调试一下 就可以

```
You have to roll 5 dices and get 3, 1, 3, 3, 7 in order.  
Press enter to roll.  
1 2 3 4 53  
ASIS{1fc1089e328eaf737c882ca0b10fcfe6}hi all -----  
Welcome to dice game!
```

得到flag

lol

终于。。我还是直接撸 llvm了 其实这个题我一开始再撸 610windows 那个题

但是他那个虚拟指令 我大概看懂了一点 但是 还是看的不是很明白 然后我们看一下这个题把

我们直接看 so库里面的代码 java 层代码 没有什么好说的 就那么多的东西

```
555         }      ''''  
556     }  
557     } else if ( v3 == 212135283 )  
558     {  
559 LABEL_37:  
560     v3 = 114562855;  
561     }  
562     } else if ( v3 == 114562855 )  
563     {  
564 LABEL_162:  
565     v3 = -2075160825;  
566     }  
567     } else if ( v3 == -64519298 )  
568     {  
569     v3 = -1436822662;  
570     if ( v26 )  
571     v3 = -1995193462;  
572     }  
573     }  
574     } else if ( v3 == -180004695 )  
575     {  
576     v3 = -1754276441;  
577     if ( v4 < v22 )  
578     v3 = -1308463181;  
579     }  
580     }  
581     }  
582     } while ( v3 != -273913548 );  
583     return _stack_chk_guard - v29;  
584 }  
585 }
```

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

很明显。。。。。要是我们直接撸 我感觉 起码一个星期才能撸出来 不过我听说 有关于pin 还有 符号执行能够解决这个问题 但是我现在还不是很明白怎么搞这个东西 比较好的是 这个题的 逻辑不是很难 （也是我瞎猫碰见死耗子 运气好一点吧）

我们先看一下主要的地方

```
if ( v16 == 1514569518 )  
{  
    v16 = -1698255010;  
    if ( v28[v15] != aNzRo168hvii8q[v15] )  
        v16 = -1091677274;  
}
```

```

        if ( v3 > -1410005754 )
            break;
        if ( v3 == -1436822662 )
        {
            v2 = v5;
            v7 = (unsigned __int8)(v24[v4] & 0xF6) | ~(unsigned __int8)v24[v4] & 9;
            v3 = 619176817;
            v28[v4] = v7;

            break;
        }
        if ( v3 == -1995193462 )
        {
            v2 = v5;
            v7 = (v24[v4] & 0x26) | ~(unsigned __int8)v24[v4] & 0xD9) ^ 0xDE;
            goto LABEL_16;
        }
    }
    if ( v3 <= -1754276442 )

```

那么我们看一下 跳转条件是什么

```

    {
        v3 = -1436822662;
        if ( v26 )
            v3 = -1995193462;
    }

    else if ( v3 == -64519298 )
    {
        v3 = -1436822662;
        if ( v26 )
            v3 = -1995193462;
    }
    else if ( v3 == -180004695 )

        v3 = -64519298;
        v6 = 0;
        if ( (~v4 | 0xFFFFFFFF) == -1 )
            v6 = 1;
        v26 = v6;
    }

```

$0xFFFFFFFF = 11111111111111111111111111111110 - 1 = 0xFFFFFFFF = 11111111111111111111111111111111$

那么  $\sim v4 | 11111111111111111111111111111110 == 11111111111111111111111111111111$

那么这里 是  $\sim v4$  只要最后一位是1 那么就会完成条件  $\sim$ 又是取反 那么只要是0那么就是完成条件

那么这个条件可以 简化成  $v4 \% 2 == 0$

在这里之后再也找不到 v26的变化地方了 那么我们尝试一下写一个脚本试试 看看是否是正确答案

```
#coding:utf-8
if __name__ == '__main__':
str1="NZ@rol68hViIs8qlX~7{6m&t"
flag=""
index=0
l=''
for i in str1:
    for j in range(256):
        if index%2==0:
            l=chr((j & 0x26 | ~j& 0xD9) ^ 0xDE)
        else:
            l=chr((j & 0xF6) | ~j& 9)
        if(l==i):
            flag+=chr(j)
            break
    index+=1
print(flag)
```

结果正确了~~