xctf reverse

原创

Y0n1an 于 2021-02-16 14:24:15 发布 46 收藏

分类专栏： re wp 文章标签： 安全

本文链接：https://blog.csdn.net/qq_46441427/article/details/113815384

oprn-source

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    if (argc != 4) {                        //判断传入给argc是不是4
     printf("what?\n");
     exit(1);
    }

    unsigned int first = atoi(argv[1]);
    if (first != 0xcafe) {
     printf("you are wrong, sorry.\n");  //first = 0xcafe
     exit(2);
    }

    unsigned int second = atoi(argv[2]);
    if (second % 5 == 3 || second % 17 != 8) {   //second = 25
     printf("ha, you won't get it!\n");
     exit(3);
    }

    if (strcmp("h4cky0u", argv[3])) {
     printf("so close, dude!\n");             //直接把h4cky0u给argv[3]
     exit(4);
    }

    printf("Brr wrrr grr\n");

    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;

    printf("Get your key: ");
    printf("%x\n", hash);
    return 0;
}
```

我们粗略看一下源码，是四个if，然后把赋的值给三个变量，最后的flag是hash的值，那我们一段段来分析

（1）

```
int main(int argc, char *argv[]) {
    if (argc != 4) {                    //判断是不是四个参数，传入到agrc里面
     printf("what?\n");
     exit(1);
```

这里表示的是传入main函数参数的个数，argc是一个整型变量，也就是有四个参数传入main

（2）

```
unsigned int first = atoi(argv[1]);
    if (first != 0xcafe) {
     printf("you are wrong, sorry.\n");  //first = 0xcafe
     exit(2);
    }
```

先把argv[1]的值传给first。这里，如果first不等于0xcafe，那就退出程序了所以argv【1】=0xcafe

（3）

```
unsigned int second = atoi(argv[2]);
    if (second % 5 == 3 || second % 17 != 8) {   //second = 25
     printf("ha, you won't get it!\n");
     exit(3);
    }
```

把argv【2】的值给second变量，如果second模5等于3或者second模17不等于8，只要满足其中一个就会退出进程，那这里我们让second = 25

（4）

```
if (strcmp("h4cky0u", argv[3])) {
     printf("so close, dude!\n");
     exit(4);
    }
```

下面是 strcmp() 函数的声明。

> int strcmp(const char *str1, const char *str2)
> 该函数返回值如下：
> 如果返回值小于 0，则表示 str1 小于 str2。
> 如果返回值大于 0，则表示 str1 大于 str2。
> 如果返回值等于 0，则表示 str1 等于 str2。

令if条件里的值为0，所以让h4ckyou和argv【3】相等

（5）

根据这一段编写exp `unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;`
exp:

```
a = int('0xcafe',16)
payload = a * 31337 + (25 % 17)*11 + 7 -1615810207
print(hex(hash))
```

python trade

打开，是个pyc文件
这里需要进行反编译查看源码，其实平常使用ida也是进行反编译，只不过反编译的C
用在线工具

```
# uncompyle6 version 3.5.0
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.5 (default, Aug  7 2019, 00:51:29)
# [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]
# Embedded file name: 1.py
# Compiled at: 2017-06-03 10:20:43
import base64

def encode(message):
    s = ''
    for i in message:
        x = ord(i) ^ 32
        x = x + 16
        s += chr(x)

    return base64.b64encode(s)


correct = 'XlNkVmtUI1MgXWBZXCFeKY+AaXNt'  #定义了一个字符串
flag = ''   #定义一个空的变量
print 'Input flag:'  # 在交互界面打印：INput flag
flag = raw_input() # 接收字符串 1*
if encode(flag) == correct:  #如果调用encode函数，参数是flag后的返回值等于correct 就对了
    print 'correct'
else:
    print 'wrong'
```

> *1：python raw_input() 用来获取控制台的输入。
> raw_input() 将所有输入作为字符串看待，返回字符串类型。
> 注意：input() 和 raw_input() 这两个函数均能接收 字符串 ，但 raw_input() 直接读取控制台的输入（任何类型的输入它都可以接收）。而对于 input() ，它希望能够读取一个合法的 python 表达式，即你输入字符串的时候必须使用引号将它括起来，否则它会引发一个 SyntaxError 。
> 除非对 input() 有特别需要，否则一般情况下我们都是推荐使用 raw_input() 来与用户交互。
> 注意：python3 里 input() 默认接收到的是 str 类型。

看encode函数

```
def encode(message):        # 参数是message
    s = ''                  #s的局部变量先定义为空
    for i in message:       #参数message里面的每一个字符
        x = ord(i) ^ 32     #把该字符的ascii码与32异或，值传给局部变量x
        x = x + 16          #x+16
        s += chr(x)         #返回当前整数对应的ascii符号传给s，然后每一个字符拼接

    return base64.b64encode(s) #返回值是s的base64编码
```

编写exp

```
mport base64
correct = 'XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
message=base64.b64decode(correct)
s=' '
for i in message:
    x = (ord(i) -16)^32       #这里是encode函数的逆运算，注意这里异或逆运算还是异或哈
    s += chr(x)
print(s)
```

得到flag

```
franex@franex-virtual-machine:~/桌面$ python reexp.py
 nctf{d3c0mpil1n9_PyC}
franex@franex-virtual-machine:~/桌面$
```