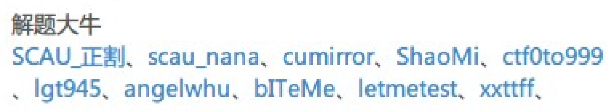
IDF 实验室部分题目WriteUp

转载

dingjin9321 于 2015-01-02 04:13:00 发布 101 收藏
文章标签： python 开发工具 php
原文链接：http://www.cnblogs.com/lastavengers/p/4198095.html
版权


解题大牛
SCAU_正割、scau_nana、cumirror、ShaoMi、ctf0to999
、lgt945、angelwhu、bITeMe、letmetest、xxttff、

前天花了一个下午的时间刷了几道IDF实验室的题目, 这个网站实在是有点冷清, 题目也比较少, 所以就被我和师兄们刷榜了2333...

因为我最先开始做, 所以就干脆刷到第一去了.

题目很水, 切莫见怪.

牛刀小试

http://ctf.idf.cn/index.php?g=game&m=list&a=index&id=16

莫尔斯密码: 网上有转换器, 转换后去空格全小写就是flag. flag: wctf{morseode}

ASCII码而已: 这是Unicode码好吧...随便找个语言Print一下即可，我用的是java, 为什么那么执着于粉丝呢？ flag: wctf{moremore_weibo_fans}

最简单的题目: F12, 在源码里面搜索, flag: wctf{woldy_s_weibo}

啥: 把图片用WinHex打开, 找flag: wctf{mianwubiaoqing__}

被改错的密码: 注意到给出的密文里面出现了一个不合时宜的字母l, 去掉后在cmd5查询得到flag: wctf{idf}

-天孤剑-的微博: 在首页「寒 近 嗜 好 酒　剑 动 秀 清 风」里面的「剑」就能找到flag: wctf{@无所不能的魂大人} (广告真是无处不在

包罗万象

http://ctf.idf.cn/index.php?g=game&m=list&a=index&id=17

图片里的英语: 好吧这题是我脑洞不够大, 是同学告诉我做法的, 图片改名成任意压缩文件, 解压得到赵本山的剧照, flag就是剧照里面的台词首字母, 图片本来是没有字幕, 百度之

台词是「May the force be with you」, flag: wctf{Mtfbwy}

逆天而行

http://ctf.idf.cn/index.php?g=game&m=list&a=index&id=21

逆向题起了这么个名字…

只做了三题,

Python BtyeCode: 好像是用Cpython编译的, 手头没工具, 用的是师兄给的反编译出来的代码:

```python
def encrypt(key, seed, string):
    rst = []
    for v in string:
        rst.append((ord(v) + seed ^ ord(key[seed])) % 255)
        seed = (seed + 1) % len(key)
    return rst
if __name__ == '__main__':
    print "Welcome to idf's python crackme"
    flag = input('Enter the Flag: ')
    KEY1 = 'Maybe you are good at decryptint Byte Code, have a try!'
    KEY2 = [
        124, 48, 52, 59, 164, 50, 37, 62, 67, 52, 48, 6, 1,
        122, 3, 22, 72, 1, 1, 14, 46, 27, 232]
    en_out = encrypt(KEY1, 5, flag)
    if KEY2 == en_out:
        print 'You Win'
    else:
        print 'Try Again !'
```

正常人反着推肯定是写: arr[i] - (key[seed] ^ seed);

不过这样肯定推不出来, 因为出题人写错了… (师兄想出来的… 打死我也不知道是题目错了啊…), 出题的用了正确的算法给出密文之后, 把

rst.append((ord(v) ^ ord(key[seed]) + seed) % 255) 写成了:

rst.append((ord(v) + seed ^ ord(key[seed])) % 255)

所以没必要深究什么了:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
int main(){
    unsigned char arr[] = {
        124, 48, 52, 59, 164, 50, 37,
        62, 67, 52, 48, 6, 1, 122, 3,
        22, 72, 1, 1, 14, 46, 27, 232, 0};
    const char key[] = "Maybe you are good at decryptint Byte Code, have a try!";
    char flag[100];
    int tmp;
    int seed = 5;
    for (int i = 0; i < 23; i++){
        tmp = (arr[i] ^ key[seed]) - seed;
        flag[i] =abs(tmp)%127; //事实上为什么要%127也不太清楚
        while (flag[i] < 32 || flag[i] > 127){
            if (flag[i] < 32) flag[i] = flag[i] + 255;
            if (flag[i] > 127) flag[i] = flag[i] - 255;
        }
        printf("arr[%d] = %d; key[%d] = %d; ^ = %d flag[%d] = %d;\n",
                i,arr[i],seed, key[seed],key[seed]^seed, i, flag[i]);
        seed = (seed + 1)%strlen(key);
    }
    puts(flag);
    printf("\n");
    system("pause");
    return 0;
}
```

flag: WCTF{ILOVEPYTHONSOMUCH}

简单的ELF逆向:

这题是ELFx64位的CrackMe, 只能用IDA啦, 载入之,师兄叫我用F4 F5,不过64位的IDA好像没有F5, 找到main函数, F4, 得到代码:

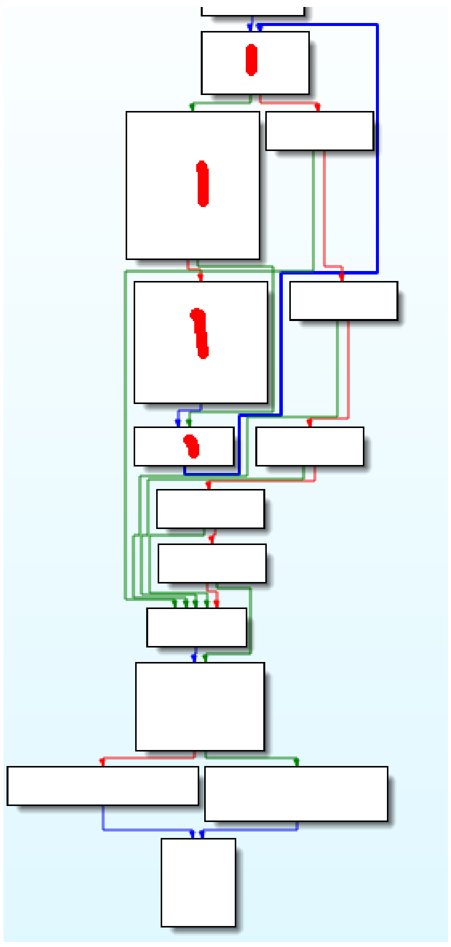
```
addr_0x400900_12:
{
    v13 = 0;
    if (v3 != 22) {
        v13 = 1;
    }
    v14 = 0;
    while ((unsigned char)(uint1_t)(v14 <= 16) != 0) {
        eax15 = (int32_t)(uint32_t)(unsigned char)v8;
        if ((int32_t)*(signed char*)&eax15 != 0) {
            v13 = 1;
            *(int32_t*)&rsi = 0;
            *((int32_t*)&rsi + 1) = 0;
            printf("%d", 0);
        }
        ++v14;
    }
    eax16 = (int32_t)(uint32_t)(unsigned char)v17;
    if (*(signed char*)&eax16 != 48
            || ((eax18 = (int32_t)(uint32_t)(unsigned char)v19, *(signed char*)&eax18 != 56)
                || ((eax20 = (int32_t)(uint32_t)(unsigned char)v21, *(signed char*)&eax20 != 50)
                    || ((eax22 = (int32_t)(uint32_t)(unsigned char)v23, *(signed char*)&eax22 != 51)
                        || (eax24 = (int32_t)(uint32_t)(unsigned char)v25, *(signed char*)&eax24 != 0x7d))))) {
        v13 = 1;
    }

puts("\r", rsi);
/* v13 应该是一个标志变量 */
    if (v13 != 0) {
        puts("u r wrong\r\n\r", rsi);
        rax26 = main("u r wrong\r\n\r", rsi);
    } else {
        puts("u r right!\r", rsi);
    }
    return 0;
addr_0x4008ff_7:
    goto addr_0x400900_12;
}
```

果然代码的可读性不是很好, 前面的printf之类的被我省去了, 重点放在while循环和那个if上, 可以看到if要求的是几个变量必须分别为 0, 8, 2, 3,} 应该就是flag 的后部分了, 从最后的判断right和wrong可以看出v13是判断正确与否的变量.

while循环是在是难懂, 乖乖回去看汇编好了.

右键选择Graphic View模式, 这样汇编代码显得很清晰, 把重点放在while循环对应的那部分, 简单分析得到, 红笔标注的地方就是程序内为数不多的循环了, 循环之后多条并排的绿线那里是多路if,最后的是正确与否的判断以及输出.



关键代码如下，interator 对应var_14, arr_1 对应var_40, arr_2 对应 var_c0:

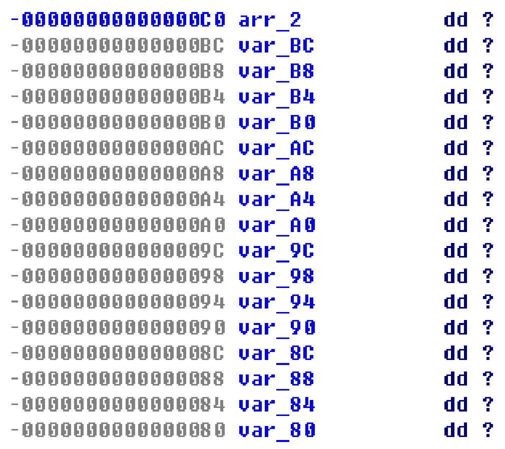
```
loc_40097C:
cmp     [rbp+iterator], 10h    ; 循环总次数
setle   al
test    al, al
jnz     short loc_40091D

loc_40091D:
mov     eax, [rbp+iterator]    ; 装入循环变量
cdqe
movzx   eax, [rbp+rax+arr_1]
movsx   edx, al         ; 取出(unsigned char)arr_1[iterator], 数组元素只有一个字节
mov     eax, [rbp+iterator]
cdqe
mov     eax, [rbp+rax*4+arr_2] ; 取出(int)arr_2[iterator], 四个字节
sub     eax, 1         ; eax = eax - 1
mov     ecx, eax      ; ecx = eax
shr     ecx, 1Fh     ; ecx = ecx >> 0x1f
lea     eax, [rcx+rax]    ; 装入地址其实就是 eax = ecx + eax;
sar     eax, 1          ; eax = eax >> 1
cmp     edx, eax      ; 比较arr_2[iterator]经过运算的值是否等于arr_1[iterator]
jz      short loc_400978    ; 等于则跳

loc_400978:
add     [rbp+iterator], 1
```

经过以上分析可以知道 arr_1 应该是我们输入的key, 所以有必要知道arr_2 的值,

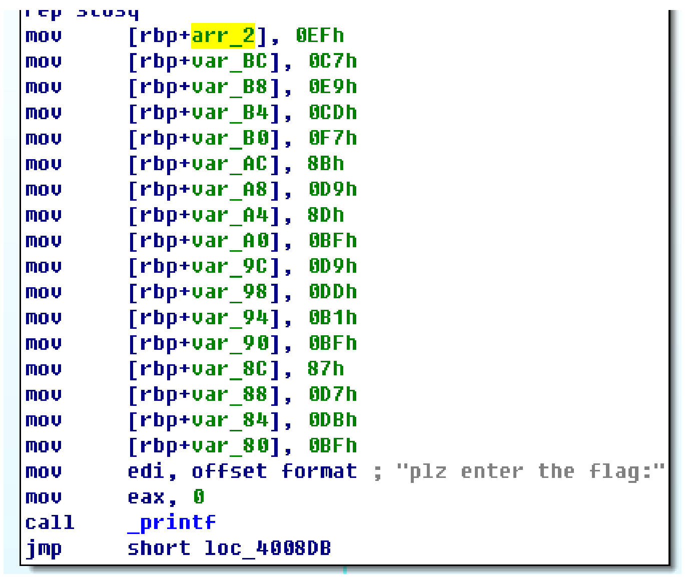
跳转到arr_2的定义:



```
-00000000000000C0 arr_2           dd ?
-00000000000000BC var_BC          dd ?
-00000000000000B8 var_B8          dd ?
-00000000000000B4 var_B4          dd ?
-00000000000000B0 var_B0          dd ?
-00000000000000AC var_AC          dd ?
-00000000000000A8 var_A8          dd ?
-00000000000000A4 var_A4          dd ?
-00000000000000A0 var_A0          dd ?
-000000000000009C var_9C          dd ?
-0000000000000098 var_98          dd ?
-0000000000000094 var_94          dd ?
-0000000000000090 var_90          dd ?
-000000000000008C var_8C          dd ?
-0000000000000088 var_88          dd ?
-0000000000000084 var_84          dd ?
-0000000000000080 var_80          dd ?
```

是空的…

但是我们回到代码中, 对arr_2有这样的操作:



```
rep stosq
mov     [rbp+arr_2], 0EFh
mov     [rbp+var_BC], 0C7h
mov     [rbp+var_B8], 0E9h
mov     [rbp+var_B4], 0CDh
mov     [rbp+var_B0], 0F7h
mov     [rbp+var_AC], 8Bh
mov     [rbp+var_A8], 0D9h
mov     [rbp+var_A4], 8Dh
mov     [rbp+var_A0], 0BFh
mov     [rbp+var_9C], 0D9h
mov     [rbp+var_98], 0DDh
mov     [rbp+var_94], 0B1h
mov     [rbp+var_90], 0BFh
mov     [rbp+var_8C], 87h
mov     [rbp+var_88], 0D7h
mov     [rbp+var_84], 0DBh
mov     [rbp+var_80], 0BFh
mov     edi, offset format ; "plz enter the flag:"
mov     eax, 0
call    _printf
jmp     short loc_4008DB
```

刚好17个项(0-10h),

所以说 arr_i[i] = ((arr_2[i] – 1) + (arr_2[i] – 1)>>0x1f)>>1

(忽略了shr 和 sar 以及各种细节问题… 所幸没有出错)

(vim 来处理这些最爽了)

代码:

```
#include <cstdio>
#include <cstdlib>
#define N 17
int arr_2[N] = {
    0x0EF, 0x0C7, 0x0E9, 0x0CD, 0x0F7, 0x8B, 0x0D9,
    0x8D, 0x0BF, 0x0D9, 0x0DD, 0x0B1, 0x0BF, 0x87,
    0x0D7, 0x0DB, 0x0BF
};
int main(){
    for (int i = 0; i < N; i++){
        int ch = ((arr_2[i] - 1) + ((arr_2[i] - 1) >> 0x1f))>>1;
        /*注意一下 >> 的优先级*/
        printf("%c",ch);
    }
    printf("0823}\n");
    system("pause.");
    return 0;
}
```

flag: wctf{ElF_lnX_Ckm_0823}

简单的PE文件逆向:

x86平台, 双击没法运行, 应该需要某个古老的C++运行时, 那就放弃用OD了, IDA载入, 稍微翻一翻(其实是不知道如何有效定位), 0x4113a0处就是关键处, F5之, 这次代码好看多了, 可以看出和上一个CrackMe基本相同...

```
flag = 0;
for ( i = 0; i < 17; ++i ){
    if ( v76[i] != byte_415768[*(&v53 + i)] )
        flag = 1;
}
if ( v77 != 49 || v78 != 48 || v79 != 50 || v80 != 52 || v81 != 125 )
    flag = 1;
    v76[v75] = 0;
    printf("\r\n");
    sub_411136();
if ( flag )
{
    printf("u r wrong\r\n\r\n");
    sub_411136();
    sub_41113B();
}
else
{
    printf("u r right!\r\n");
    sub_411136();
}
system("pause");
```

同样是把flag分成两部分, 后面五个必须是1024},前面的在一个for循环里算出:

v76[i] != byte_415768[*(&v53 + i)]

通过一个数组v53[]运算出下标, 再用下标从另一个数组byte_415768[]取出值来, 数组是:

```
v53 = 1;
v54 = 4;
v55 = 14;
v56 = 10;
v57 = 5;
v58 = 36;
v59 = 23;
v60 = 42;
v61 = 13;
v62 = 19;
v63 = 28;
v64 = 13;
v65 = 27;
v66 = 39;
v67 = 48;
v68 = 41;
v69 = 42;
byte_415768 db 73h
                   db 'wfxc{gdv}fwfctslydRddoepsckaNDMSRITPNsmr1_=2cdsef66246087138',0
```

要注意byte_415768[]的一个元素s(73h)没有被识别.

所以:

```
#include <cstdio>
#include <cstdlib>
int v53[] = {
    1, 4, 14, 10, 5, 36, 23, 42, 13,
    19, 28, 13, 27, 39, 48, 41, 4
};
char byte_415768[] = "swfxc{gdv}fwfctslydRddoepsckaNDMSRITPNsmr1_=2cdsef66246087138\0";

int main(){
    for (int i = 0; i < 17; i++){
        printf("%c", byte_415768[v53[i]]);
    }
    printf("1024}\n");
    system("pause");
}
```

flag: wctf{Pe_cRackme1c1024}

凯撒加密:

给出一个字符串, 写个程序按127为周期跑一圈, 得到二十多个处在合法范围内的串, 最正常的一个如下, 像是base64, 解码一下得出flag.

dGhlIGZsYWcgaXMgd2N0ZntrYWlzYV9qaWFhYWFhbWl9LHBseiBmbG93IG15IHdlaWJvLGh0dHA6Ly93ZWliby5jb

代码:

```c
#include <cstdio>
#include <cstdlib>
#include <cstring>
char kaiser[] = "U8Y]:8KdJHTXRI>XU#?!K_ecJH]kJG*bRH7YJH7YSH]*=93dVZ3^S8*$:8\"&:9U]RH;g=8Y!U92'=j*$KH]ZSj&[S#!gU#*dK9\\.";
/* 记得转义 */
int main(){
    char ch[1000];
    bool flag = true;
    for (int i = 0; i < 127; i++){
        memset(ch, 0, sizeof(ch));
        for (int j = 0; j < strlen(kaiser); j++){
            ch[j] = (kaiser[j] + i)%127;
            if (ch[j] <= 32 || ch[j] > 127) flag = false;
        }
        if (flag) puts(ch);
    }
    system("pause");
    return 0;
}
```

flag: wctf{kaisa_jiaaaaami}

天罗地网

http://ctf.idf.cn/index.php?g=game&m=list&a=index&id=22

一种编码而已: Jother编码, JSのBrainFuck, 前几天刚在Wooyun看到…直接在F12控制台执行得到:

flag: WCTF{H3110_J0t4er}

超简单的JS题: 查看源码发现script标签里有这一段:

```javascript
var p1 ='%66%75%6e%63%74%69%6f%6e%20%63%68%65%63%6b%53%75%62%6d%69%74%28%29%7b%76%61%72%20%61%3d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%22%70%61%73%73%77%6f%72%64%22%29%3b%69%66%28%22%75%6e%64%65%66%69%6e%65%64%22%21%3d%74%79%70%65%6f%66%20%61%29%7b%69%66%28%22%65%66%66%35%62%66%38%63%63';
var p2 ='%31%61%30%64%38%33%35%36%32%63%35%22%3d%3d%61%2e%76%61%6c%75%65%29%72%65%74%75%72%6e%21%30%3b%61%6c%65%72%74%28%22%45%72%72%6f%72%22%29%3b%61%2e%66%6f%63%75%73%28%29%3b%72%65%74%75%72%6e%21%31%7d%7d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%42%79%49%64%28%22%6c%65%76%65%6c%51%75%65%73%74%22%29%2e%6f%6e%73%75%62%6d%69%74%3d%63%68%65%63%6b%53%75%62%6d%69%74%3b';
eval(unescape(p1) + unescape('%33%66%61%37%38%32%66%32%36%31%61%35' + p2));
```

JS我是完全不懂…

查了一下unescape()发现这个函数是用来解码字符串的, 大概是为了让URL参数支持Unicode吧… eval()用来执行参数指定的JS代码, 有点MetaPrograming(元编程)的意思.

找个在线解码: http://tool.chinaz.com/Tools/Escape.aspx

```
var p1 = 'function checkSubmit(){var a=document.getElementById("password");if("undefined"!=typeof a){if("eff5bf8cc';
var p2 ='1a0d83562c5"==a.value)return!0;alert("Error");a.focus();return!1}}document.getElementById("levelQuest").onsubmit=checkSubmit;';
eval(unescape(p1) + unescape('3fa782f261a5' + p2));
```

合并起来就是:

```
function checkSubmit(){
    var a=document.getElementById("password");
    if("undefined"!=typeof a){
        if("eff5bf8cc3fa782f261a51a0d83562c5"==a.value)
            return!0;
        alert("Error");
        a.focus();
        return!1
    }
}
document.getElementById("levelQuest").onsubmit=checkSubmit;
```

这下就清晰了, 输入eff5bf8cc3fa782f261a51a0d83562c5提交:

flag: wctf{webclieNt_c0py}

古老的邮件编码: 不太清楚邮件编码什么的, 稍微搜索后发现它比较符合 Uuencode编码的特征, 找在线转换工具: http://web.chacuo.net/charsetuuencode

flag: wctf{uuuuuencode__}

剩下的题就没做了(其实是不会做).

转载于:https://www.cnblogs.com/lastavengers/p/4198095.html