

Reverse 部分 WP/出题思路

Babymath

Checkin

二十一元一次方程，随机数生成的式子，有多解，上网找个 z3 解多解的脚本就行。

babysmc

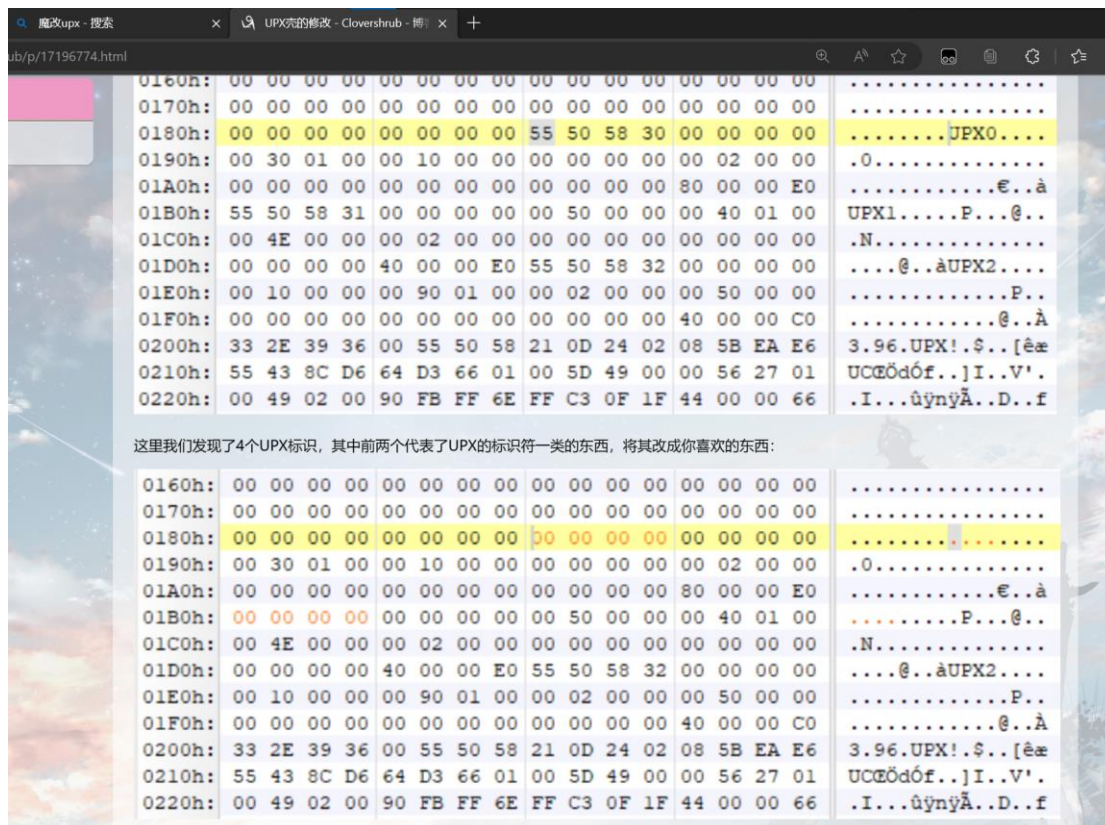
checkin

解出来的人数没有想象中的多

UPX 加壳，稍微改了一下

0070	6D 6F 64 65	2E 0D 0D 0A	24 00 00 00	00 00 00 00	mode....\$......
0080	50 45 00 00	4C 01 03 00	74 A7 9F 65	00 82 00 00	PE..L...tšŸe.,..
0090	00 02 00 00	E0 00 07 01	0B 01 02 20	00 40 00 00à...... @..
00A0	00 10 00 00	00 00 01 00	40 46 01 00	00 10 01 00@F.....
00B0	00 50 01 00	00 00 40 00	00 10 00 00	00 02 00 00	.P....@.....
00C0	04 00 00 00	01 00 00 00	04 00 00 00	00 00 00 00
00D0	00 60 01 00	00 02 00 00	00 00 00 00	03 00 00 00	.`.....
00E0	00 00 20 00	00 10 00 00	00 00 10 00	00 10 00 00
00F0	00 00 00 00	10 00 00 00	00 00 00 00	00 00 00 00
0100	00 50 01 00	B4 00 00 00	00 00 00 00	00 00 00 00	.P..`.....
0110	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0130	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0140	00 48 01 00	18 00 00 00	00 00 00 00	00 00 00 00	.H.....
0150	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0160	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0170	00 00 00 00	00 00 00 00	2E 2E 2E 30	00 00 00 000....
0180	00 00 01 00	00 10 00 00	00 00 00 00	00 02 00 00
0190	00 00 00 00	00 00 00 00	00 00 00 00	80 00 00 E0€..à
01A0	2E 2E 2E 31	00 00 00 00	00 40 00 00	00 10 01 00	...1.....@.....
01B0	00 3A 00 00	00 02 00 00	00 00 00 00	00 00 00 00	.:.....
01C0	00 00 00 00	40 00 00 E0	2E 2E 2E 32	00 00 00 00@..à...2....
01D0	00 10 00 00	00 50 01 00	00 02 00 00	00 3C 00 00P.....<..
01E0	00 00 00 00	00 00 00 00	00 00 00 00	40 00 00 C0@..À
01F0	34 2E 30 32	00 2E 2E 2E	21 0D 09 02	08 E9 FE A4	4.02....!....éþ

对 UPX 壳最基本的改动，能删的貌似不止这些，设想是做题多点应该能遇见过这玩意，或者上网搜魔改 upx 能搜出来一大堆：



这题是将 UPX 改成了 0x2e0x2e0x2e，改回来就能一键脱壳了。

当然也可以手动脱，@Libr 看的一个教程 <https://www.anquanke.com/post/id/272639>。

脱完后拖进 ida 看，有个 smc （Self Modifying Code）函数

```
int sub_4014C8()
{
    int result; // eax
    DWORD f10ldProtect[2]; // [esp+1Ch] [ebp-1Ch] BYREF
    int v2; // [esp+24h] [ebp-14h]
    int (__usercall *v3)@<eax>(char@<pf>); // [esp+28h] [ebp-10h]
    int i; // [esp+2Ch] [ebp-Ch]

    f10ldProtect[0] = (DWORD)malloc(8u);
    v3 = sub_401410;
    v2 = 184;
    f10ldProtect[1] = (unsigned int)sub_401410 & 0xFFFFF000;
    VirtualProtect((LPVOID)((unsigned int)sub_401410 & 0xFFFFF000), 0x1000u, 0x80u, f10ldProtect);
    for ( i = 0; ; ++i )
    {
        result = i;
        if ( i >= v2 )
            break;
        *((_BYTE *)v3 + i) ^= 0x2Eu;
    }
    return result;
}
```

就是对 check 函数进行异或 0x2e 的操作，异或回来就行。

异或又有两种办法，一种是用 ida 脚本静态，一种是直接调试下断点到 check 函数让程序自行解密，解密完长这样：

```

.text:0040140E 90 90          align 10h
.text:00401410
.text:00401410      ; ===== S U B R O U T I N E =====
.text:00401410      ; Attributes: bp-based frame
.text:00401410      ; void sub_401410()
.text:00401410      sub_401410 proc far          ; CODE XREF: sub_40154E+3A1p
.text:00401410                                     ; DATA XREF: sub_4014C8+151o
✓.text:00401410 55          push     ebp
.text:00401410
.text:00401410      ; -----
.text:00401411 89          db      89h
.text:00401412
.text:00401412 E5 83      in       eax, 83h          ; DMA page register 74LS612:
.text:00401412      sub_401410 endp ; sp-analysis failed ; Channel 1 (address bits 16-23)
.text:00401412
.text:00401414 EC          in       al, dx
.text:00401415 28 C7      sub     bh, al
.text:00401417 45          inc     ebp
.text:00401418 F4          hlt
.text:00401418
.text:00401419 00          db      0
.text:0040141A 00          db      0
.text:0040141B 00          db      0
.text:0040141C 00          db      0
.text:0040141D EB          db      0EBh
.text:0040141E 4C          db      4Ch ; L
.text:0040141F 8B          db      8Bh
.text:00401420 45          db      45h ; E
.text:00401421 F4          db      0F4h
.text:00401422 C1          db      0C1h
.text:00401423 E0          db      0E0h
.text:00401424 02          db      2
.text:00401425 89          db      89h
.text:00401426 C2          db      0C2h
.text:00401427 8B          db      8Bh
.text:00401428 45          db      45h ; E
.text:00401429 08          db      8
.text:0040142A 01          db      1
.text:0040142B D0          db      0D0h
.text:0040142C 8B          db      8Bh
.text:0040142D 10          db      10h
.text:0040142E 8B          db      8Bh
.text:0040142F 45          db      45h ; E
.text:00401430 F4          db      0F4h
.text:00401431 C1          db      0C1h
.text:00401432 E0          db      0E0h
.text:00401433 02          db      2
.text:00401434 89          db      89h
.text:00401435 C1          db      0C1h
.text:00401436 8B          db      8Bh
.text:00401437 45          db      45h ; E
.text:00401438 08          db      8
.text:00401439 01          db      1
.text:0040143A C8          db      0C8h
.text:0040143B 81          db      81h
.text:0040143C C2          db      0C2h
.text:0040143D 78          db      78h ; x
.text:0040143E 56          db      56h ; V
.text:0040143F 34          db      34h ; 4
.text:00401440 13          db      13h

```

把这一段代码选中按 C 键分析，然后 P 键生成函数
 注意要先把函数 sub_401410 undefine 了不然会干扰 ida 的分析
 生成函数长这样：

```

1 int __cdecl sub_401410(int a1)
2 {
3     int j; // [esp+18h] [ebp-10h]
4     int i; // [esp+1Ch] [ebp-Ch]
5
6     for ( i = 0; i <= 9; ++i )
7     {
8         *(_DWORD *)(4 * i + a1) += 305419896;
9         *(_DWORD *)(4 * i + a1) ^= 0x87654321;
10    }
11    for ( j = 0; j <= 9; ++j )
12    {
13        if ( *(_DWORD *)(4 * j + a1) != dword_405020[j] )
14        {
15            printf("Wrong!");
16            exit(0);
17        }
18    }
19    return printf("Right!");
20 }

```

用密文逆回去就行了。

Babyhash

原本感觉太简单了，已经出了两个签到题，强行撸成了 c++ 增大分析难度，用 "12345" + 6 位 key1 + "67890" 作为 rc4 加密的 key 对 flag 进行加密，6 位 key1 限制了是小写字母，python 脚本爆五分钟能爆出来

```

import hashlib
for a1 in range(97,123):
    print(a1)
    for a2 in range(97,123):
        for a3 in range(97,123):
            for a4 in range(97,123):
                for a5 in range(97,123):
                    for a6 in range(97,123):
                        key=''
                        key='12345'+chr(a1)+chr(a2)+chr(a3)+chr(a4)+chr(a
5)+chr(a6)+'67890'
                        key_md5 = hashlib.md5(key.encode(encoding='utf-
8')).hexdigest()
                        if(key_md5=='f27d71f53ae17679fb352baa5ea326db'):
                            print(key)

```

```
107
108
109
12345maikei67890
110
```

得到 key 后对密文进行 rc4 加密即可。

Ezdebug

elf 文件加了点简单的反调试（难的不会）

一开始 base64 码表直接露出来的，后来有点害羞，决定加个盒隐藏一下。

```
void alarmHandler(int sig)
{
    //    printf("time is up, quicker next time\n");
    exit(1);
}
__attribute__((constructor))void setupSig(void)
{ //设置程序一开始就执行
    signal(SIGALRM, alarmHandler);
    alarm(3);
}
```

用来限时的，ida 调试时能忽略掉。

```
int checkdebug()
{
    if(ptrace(PTRACE_TRACEME, 0, 0, 0) == -1)
    {
        return 1;
    }
    return 0;
}
```

原理是一个程序只能被 PTRACE_TRACEME 一次，之后就返回-1 了。

挺好绕，nop 掉就行，但是这题去了符号表+静态编译不好直接看，估计得调试过程中发现。

根据 checkdebug 得到的正在调试与否的信息，对 tea 加密的 key 和 s 盒进行相应的操作

```
152 void Debug()
153 {
154     int debugflag=checkdebug();
155     if(debugflag==0)//正常
156     {
157         for(int i=0;i<4;i++)
158         {
159             teakey[i]^=0x2e;
160         }
161         for(int i=0;i<256;i++)
162         {
163             S[i]^=0xff;
164         }
165         strncpy(realtable,table1,64);
166         box2char(realtable);
167         printf("good boy :)\n");
168     }
169     if(debugflag==1)//调试
170     {
171         for(int i=0;i<4;i++)
172         {
173             teakey[i]^=0xe2;
174         }
175         strncpy(realtable,table2,64);
176         box2char(realtable);
177         printf("bad boy :(\n");
178     }
179 }
180 }
```

然后是对 input 的加密操作，先 tea 加密，再查 s 盒得出 base64 码表，然后进行 base64 加密，再把加密结果换回 s 盒对应值，结果与 flag 加密值进行比较。

```
181  ✓ int check(char *input)
182      {
183          char *base64enc_flag;
184      ✓  for(int i=0;i<4;i++)
185          {
186              teaencrypt((int*)&input[i*8],teakey);
187          }
188
189          strncpy(realtable,table1,64);
190          box2char(realtable);
191          base64enc_flag=base64_encode(input);
192          char2box(base64enc_flag);
193
194          if(strcmp(boxencflag,base64enc_flag)==0)
195              return 1;
196          return 0;
197      }
```