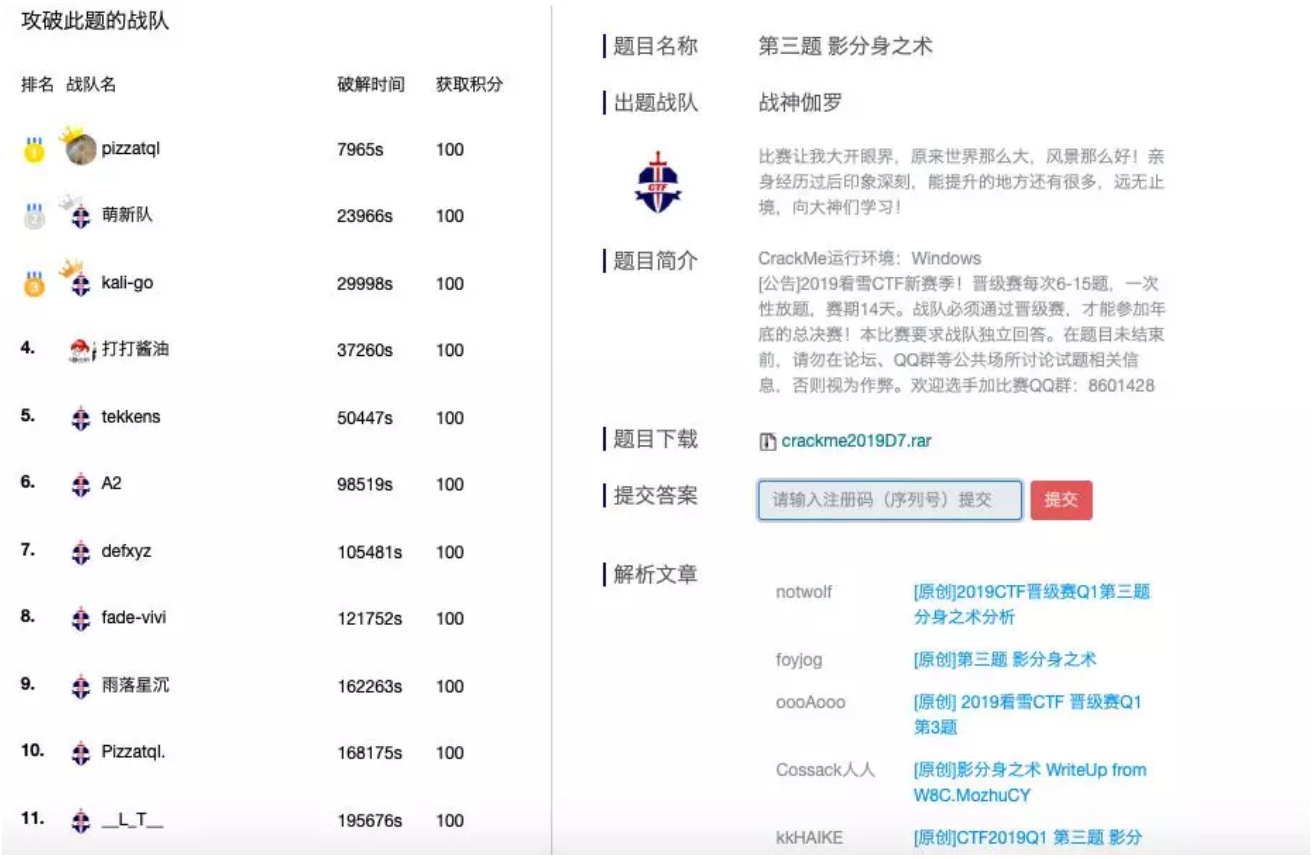
2019 KCTF 晋级赛Q1 | 第三题点评及解题思路

小雪 看雪学院 1周前

影分身之术是火影忍者中-鸣人的招牌忍术，火影迷们想必并不陌生了。与简单的分身术相比，这个由二代火影千手扉间开创的忍术，其分出来的分身不再是幻影，而是实打实的实体。

你能成功识破哪个是真身，哪个是幻影吗？接下来，让我们一起来看看这道《影分身之术》，是如何被大家破解的吧！

第三题 影分身之术



攻破此题的战队

排名	战队名	破解时间	获取积分
	pizzatql	7965s	100
	萌新队	23966s	100
	kali-go	29998s	100
4.	打打酱油	37260s	100
5.	tekkens	50447s	100
6.	A2	98519s	100
7.	defxyz	105481s	100
8.	fade-vivi	121752s	100
9.	雨落星沉	162263s	100
10.	Pizzatql.	168175s	100
11.	__L_T__	195676s	100

题目名称 第三题 影分身之术

出题战队 战神伽罗

比赛让我大开眼界，原来世界那么大，风景那么好！亲身经历过后印象深刻，能提升的地方还有很多，远无止境，向大神们学习！

题目简介 CrackMe运行环境：Windows
[公告]2019看雪CTF新赛季！晋级赛每次6-15题，一次性放题，赛期14天。战队必须通过晋级赛，才能参加年底的总决赛！本比赛要求战队独立回答。在题目未结束前，请勿在论坛、QQ群等公共场所讨论试题相关信息，否则视为作弊。欢迎选手加比赛QQ群：8601428

题目下载 crackme2019D7.rar

提交答案 请输入注册码（序列号）提交 提交

解析文章

notwolf	[原创]2019CTF晋级赛Q1第三题分身之术分析
foyjog	[原创]第三题 影分身之术
oooAooo	[原创] 2019看雪CTF 晋级赛Q1第3题
Cossack人人	[原创]影分身之术 WriteUp from W8C.MozhuCY
kkHAIKE	[原创]CTF2019Q1 第三题 影分

此题共有 **35** 支战队破解，围观人数达 **3443** 人。

出题团队

战神伽罗

战神伽罗

战队信息 战队成员(1) 成员动态

战队名称: 战神伽罗

战队签名: wannaCrack

创建者: simpower

战队总分: 400

战队介绍: 比赛让我大开眼界，原来世界那么大，风景那么好！亲身经历过后印象深刻，能提升的地方还有很多，远无止境，向大神们学习！

注册时间: 2018-10-20

战队成员看雪ID：**simpower**

个人主页：https://bbs.pediy.com/user-177594.htm

简介：

野生程序猿，临床医学专业转行，现为北京某网络安全公司研发，从事恶意软件动态行为分析产品的开发，对黑科技以及产品架构有很大兴趣，设计出了超稳定的大规模挂钩引擎，kctf中的战神伽罗系列题目某种程度上反应了本猿平时的工作特征，本系列赛题将会和你一起不断进化，成长，欢迎继续关注，感谢看雪！感谢有你参与！

看雪CTF 评委 crownless 点评

《影分身之术》这道题采用了javascript加密，并考察了简单的汇编。然而，不同的战队采用了不同的解题手法和工具，让人大开眼界，值得学习和借鉴。

题目设计思路

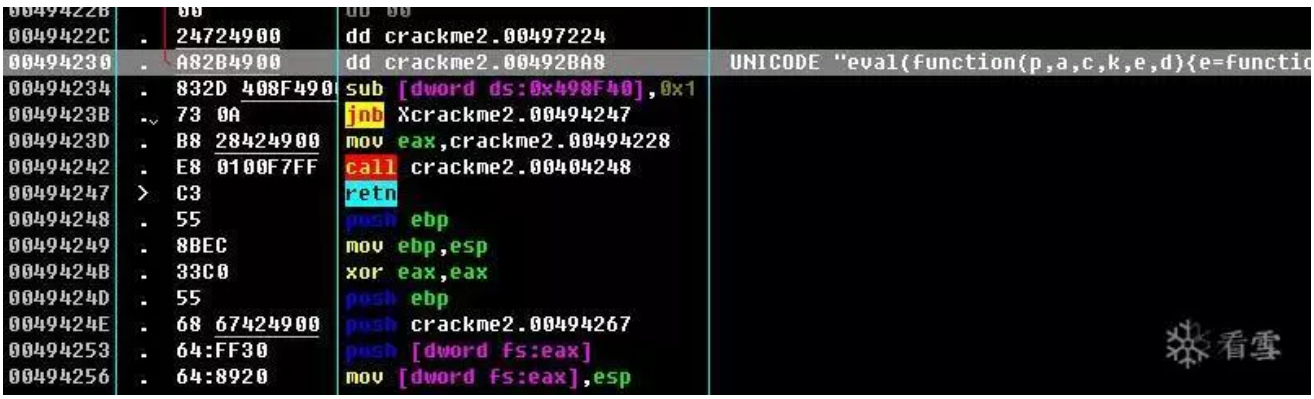
1. 首先将一部分密码封装在javascript中，通过javascript将自身进行加密。

2. 通过简单的汇编代码变形算法 (加减固定数值)，将一部分密码代码编译到可执行区域，通过指令跳转和对硬编码的变形对这部分密码进行恢复比对。

3. 将恢复的代码注入到IE内核当中，并显示出来。

破解思路

1. 搜索内存可以搜到javascript代码，并将代码中的一部分密码获取出来。



2. 解密后代码， 密码为：simpower91

```
function ckpswd() {
key="simpower91";
a = document.all.pswd.value;
if (a.indexOf(key) ==0) {
l=a.length;
i=key.length;
sptWBCallback(a.substring(i,l));
} else {
alert("wrong!<" + a + "> is not my GUID ;-)");
return "1234";
}
}
function ok(){
alert("congratulations!");
}
```

3. 剩下4位很容易就可以跟踪到比对代码的那个call处，有4个很怪异的指令被跳过，将 每个字节-7F就是剩余的ascii码 。

其中E0-7F='a'

B0-7F='1'

B1-7F='2'

B2-7F='3'

界面如下所示(注册成功后):

题目破解思路

本题破解思路由 **oooAooo** 提供

一、初探

初看程序是由dephi写的，从出题团队来看，可能是去年的加强版。用DEDE反编译一下看看，虽然注册了几个按钮事件，但并不是核心校验函数。可能是dephi+script形式。打开IDA可以发现类似如下字符串:

```
CODE:00493628 db 'function sptWBCallback(spt_wb_id,spt_wb_name,optionstr){url=',27h; Te
CODE:00493628 db '#sptWBCallback:id=',27h,';url=url+spt_wb_id+',27h,';eventName=',27h;
CODE:00493628 db '+spt_wb_name;if(optionstr) url=url+',27h,';params=optionstr',27h,';';
CODE:00493628 db 'location=url;}',0 ; Text

CODE:00493868 db '<center><br><br><br><input value="" id="pswd" size=39></input><br'; T
CODE:00493868 db '><br><br><input type=button value="checkMyFlag" onclick="ckpswd()'; T
CODE:00493868 db ';"></center>',0 ; Text
CODE:004938FF align 10h
```

从字符串上看，存在一个html，里面有个onclick的处理函数 ckpswd函数，首先需要找到ckpswd函数体。不想一步一步分析，启动程序，用ce在内存总搜索，很快可以发现如下脚本函数：

```
function ckpswd()
{
key="simpower91";
a = document.all.pswd.value;
if(a.indexOf(key) ==0)
{
l=a.length;
i=key.length;
sptWBCallback(a.substring(i,l));
}
else
{
alert("wrong!<" + a + "> is not my GUID;-)");
return "1234";
}
}
```

可以看出要求输入的sn开头部分必须是 "simpower91"，否则会提示 'worng!'，测试一下确实如此，如果输入simpower91111，没有任何提示，看起来仍然有进一步的判断。从上面的脚本可知为sptWBCallback函数，看起来像个回调函数。先找到sptWBCallback函数体。从前面的字符串中可以看到其影子，在内存中搜索下，发现如下：

```
function sptWBCallback(spt_wb_id,spt_wb_name,optionstr)
{
url='#sptWBCallback:id=';
url=url+spt_wb_id+';
eventName='+spt_wb_name;
```

```
if(optionstr)
url=url+';
params=optionstr';
location=url;
}
```

上面代码应该是WebBrowser JS回调delphi的方法。因此真正的检测应该仍然在dephi主程序中。需要找到注册时间的回调函数。分析IDA反汇编代码发下如下：

```
CODE:00491DCC sub_491DCC proc near ; CODE XREF: _Tfrmcrackme_FormCreate+81↓p
CODE:00491DCC push ebx
CODE:00491DCD mov ebx, eax
CODE:00491DCF mov eax, ebx
CODE:00491DD1 call sub_491B78 ; call function
CODE:00491DD6 mov eax, [ebx+40h]
CODE:00491DD9 mov [eax+2A4h], ebx
CODE:00491DDF mov dword ptr [eax+2A0h], offset sub_492088
CODE:00491DE9 pop ebx
CODE:00491DEA retn
CODE:00491DEA sub_491DCC endp
```

其中sub_492088函数比较可疑。设置个断点,输入“ simpower91111 ”果然断了下了。

二、回调 sub_492088

```
int __userpurge sub_492088@<eax>(int a1@<eax>, int a2@<edx>, int a3@<ecx>, int a4@<ebx>
{
int v12; // ebx
int v13; // esi
int v14; // eax
int v15; // eax
Classes::TStrings *v16; // esi
int v17; // ST14_4
int v18; // ST10_4
unsigned int v20; // [esp-14h] [ebp-34h]
void *v21; // [esp-10h] [ebp-30h]
int *v22; // [esp-Ch] [ebp-2Ch]
int v23; // [esp-8h] [ebp-28h]
int v24; // [esp-4h] [ebp-24h]
int v25; // [esp+0h] [ebp-20h]
int v26; // [esp+4h] [ebp-1Ch]
```

```
int v27; // [esp+8h] [ebp-18h]
char v28[4]; // [esp+Ch] [ebp-14h]
int v29; // [esp+10h] [ebp-10h]
int System::AnsiString; // [esp+14h] [ebp-Ch]
int v31; // [esp+18h] [ebp-8h]
int v32; // [esp+1Ch] [ebp-4h]
int savedregs; // [esp+20h] [ebp+0h]

System::AnsiString = 0;
v29 = 0;
*(_DWORD *)v28 = 0;
v27 = 0;
v26 = 0;
v25 = 0;
v24 = a4;
v23 = a5;
v31 = a3;
v32 = a2;
v12 = a1;
v22 = &savedregs;
v21 = &loc_4921D1;
v20 = __readfsdword(0);
__writefsdword(0, (unsigned int)&v20);
Variants::__linkproc__ VarToLStr(&v29, a11);
v13 = sub_465C88((int)&str__sptWBCallback_[1], v29);
if ( v13 > 0 )
{
*a6 = -1;
Variants::__linkproc__ VarToLStr(&System::AnsiString, a11);
v14 = GetJSLen(System::AnsiString);
v15 = System::__linkproc__ LStrCopy(System::AnsiString, v13 + 15, v14 - v13 - 14, (int)
LOBYTE(v15) = 1;
unknown_libname_161(System::AnsiString, (int)&str___41[1], (int)&str____19[1], v12, edi
System::__linkproc__ LStrLAsg(&System::AnsiString, *(signed __int32 *)v28);
v16 = (Classes::TStrings *)TStreamCreate((int)cls_Classes_TStringList, 1);
(*(void (__fastcall **)(Classes::TStrings *, int))(*(_DWORD *)v16 + 44))(v16, System::A
if ( *(_WORD *)(v12 + 50) )
{
Classes::TStrings::GetValue(v16, (const int)&str_params[1], (int)&v27);
v17 = v27;
Classes::TStrings::GetValue(v16, (const int)&str_eventName[1], (int)&v26);
v18 = v26;
Classes::TStrings::GetValue(v16, (const int)&str_id[1], (int)&v25);
(*(void (__fastcall **)(_DWORD, int, int, int))(v12 + 48))(*(_DWORD *)(v12 + 52), v25,
}
```

```
}
if ( *(_WORD *)(v12 + 58) )
(*(void (__fastcall **)(_DWORD, int, int, int, int, int, int, int, _WORD *))(v12 + 0x38
*(_DWORD *)(v12 + 60),
v32,
v31,
a11,
a10,
a9,
a8,
a7,
a6);
__writefsdword(0, v20);
v22 = (int *)&loc_4921D8;
return System::__linkproc__ LStrArrayClr(&v25, 6);
}
```

核心处理函数为 V12+0X38，地址为;493F70

三、493F70函数

```
loc_493F70: ; DATA XREF: _Tfrmcrackme_FormCreate+87↑o
CODE:00493F70 push ebp
CODE:00493F71 mov ebp, esp
CODE:00493F73
CODE:00493F73 loc_493F73: ; CODE XREF: CODE:00493FC5↓j
CODE:00493F73 add esp, 0FFFFFBD8h
CODE:00493F79 push ebx
CODE:00493F7A push esi
CODE:00493F7B xor ebx, ebx
CODE:00493F7D mov [ebp-428h], ebx
CODE:00493F83 mov [ebp-424h], ebx
CODE:00493F89 mov [ebp-420h], ebx
CODE:00493F8F mov [ebp-14h], edx
CODE:00493F92 mov ebx, eax
CODE:00493F94 mov eax, [ebp-14h]
CODE:00493F97 call @System@@LStrAddRef$qqrpv ; System::__linkproc__ LStrAddRef(void *)
CODE:00493F9C mov eax, [ebp+8]
CODE:00493F9F call @System@@LStrAddRef$qqrpv ; System::__linkproc__ LStrAddRef(void *)
CODE:00493FA4 xor eax, eax
CODE:00493FA6 push ebp
```

```
CODE:00493FA7 push offset loc_4941BA
CODE:00493FAC push dword ptr fs:[eax]
CODE:00493FAF mov fs:[eax], esp
CODE:00493FB2 mov eax, [ebp-14h]
CODE:00493FB5 call GetJSLen ; BDS 2005-2007 and Delphi6-7 Visual Component Library
CODE:00493FBA cmp eax, 4
CODE:00493FBD jnz loc_494181
CODE:00493FC3 jmp short loc_493FE6
CODE:00493FC5 ; ---------------------------------------------------------------------------
CODE:00493FC5 loopne near ptr loc_493F73+4
CODE:00493FC7 mov cl, 0B2h
CODE:00493FC9 mov edx, offset byte_494045
CODE:00493FCE mov [ebp-4], edx
CODE:00493FD1 mov edx, offset loc_4940D0
CODE:00493FD6 mov [ebp-8], edx
CODE:00493FD9 mov edx, offset dword_494168
CODE:00493FDE mov [ebp-0Ch], edx
CODE:00493FE1 jmp loc_4940DA
CODE:00493FE6 ; ---------------------------------------------------------------------------
CODE:00493FE6
CODE:00493FE6 loc_493FE6: ; CODE XREF: CODE:00493FC3↑j
CODE:00493FE6 mov eax, offset _str_data_txt_1.Text
CODE:00493FEB call @Sysutils@FileExists$qqrx17System@AnsiString ; Sysutils::FileExists(
CODE:00493FF0 test al, al
CODE:00493FF2 jz short loc_494010
CODE:00493FF4 mov dl, 1
CODE:00493FF6 mov eax, off_416648
CODE:00493FFB call TStreamCreate ; BDS 2005-2007 and Delphi6-7 Visual Component Library
CODE:00494000 mov esi, eax
CODE:00494002 mov edx, offset _str_data_txt_1.Text
CODE:00494007 mov eax, esi
CODE:00494009 call @Ibsql@TIBXSQLVAR@LoadFromFile$qqrx17System@AnsiString ; Ibsql::TIBX
CODE:0049400E jmp short loc_494025
CODE:00494010 ; ---------------------------------------------------------------------------
CODE:00494010
CODE:00494010 loc_494010: ; CODE XREF: CODE:00493FF2↑j
CODE:00494010 or ecx, 0FFFFFFFFh
CODE:00494013 mov edx, offset _str_data_txt_1.Text
CODE:00494018 mov eax, [ebx+330h]
CODE:0049401E call GetVmpCode
CODE:00494023 mov esi, eax
CODE:00494025
CODE:00494025 loc_494025: ; CODE XREF: CODE:0049400E↑j
CODE:00494025 mov edx, offset byte_494045
CODE:0049402A mov [ebp-4], edx
```

```
CODE:0049402D mov eax, [ebp-4]
CODE:00494030 push eax
CODE:00494031 mov eax, [esi+4]
CODE:00494034 push eax
CODE:00494035 mov eax, ds:dword_498F3C
CODE:0049403A push eax
CODE:0049403B call sub_473A04
CODE:0049403B ; ---------------------------------------------------------------------------
CODE:00494040 dd 0
CODE:00494044 db 0
CODE:00494045 byte_494045 db 0 ; DATA XREF: CODE:00493FC9↑o
CODE:00494045 ; CODE:loc_494025↑o
CODE:00494046 dd 0
CODE:0049404A align 4
CODE:0049404C dd 1Ah dup(0)
CODE:004940B4 db 2 dup(0)
CODE:004940B6 dd 0
CODE:004940BA align 4
CODE:004940BC dd 5 dup(0)
```

这个函数与去年的类似，程序对地址 494045代码进行解密，里面存在一个反汇编引擎，对代码进行重定位，并一条一条执行，同时存在一个简单的VMP。

```
73 69 6D 76 6D 01 00 05 00 .......simvm....
00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 C5 3F 49 00 05 ............I..
01 00 07 00 00 F0 00 00 00 00 00 00 00 00 01 00 ................
00 00 01 05 00 00 F0 00 00 00 00 00 00 00 00 00 ................
00 00 00 02 01 01 02 00 00 F0 00 00 00 00 F3 FF ................
FF FF 00 00 00 00 00 07 00 00 F0 00 00 00 00 00 ................
00 00 00 00 00 00 00 03 01 00 07 00 00 F0 00 00 ................
00 00 00 00 00 00 01 00 00 00 01 05 00 00 F0 00 ................
00 00 00 01 00 00 00 00 00 00 00 03 01 01 02 00 ................
00 F0 00 00 00 00 F2 FF FF FF 00 00 00 00 00 07 ................
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 03 ................
01 00 07 00 00 F0 00 00 00 00 00 00 00 00 01 00 ................
00 00 01 05 00 00 F0 00 00 00 00 02 00 00 00 00 ................
00 00 00 03 01 01 02 00 00 F0 00 00 00 00 F1 FF ................
FF FF 00 00 00 00 00 07 00 00 F0 00 00 00 00 00 ................
00 00 00 00 00 00 00 03 01 00 07 00 00 F0 00 00 ................
00 00 00 00 00 00 01 00 00 00 01 05 00 00 F0 00 ................
00 00 00 03 00 00 00 00 00 00 00 03 01 01 02 00 ................
00 F0 00 00 00 00 F0 FF FF FF 00 00 00 00 00 07 ................
```

```
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 03 ................
01 00 07 00 00 F0 00 00 00 00 00 00 00 00 00 00 ................
00 00 04 02 00 00 F0 00 00 00 00 EC FF FF FF 00 ................
00 00 00 03 01 00 07 00 00 F0 00 00 00 00 00 00 ................
00 00 00 00 00 00 01 07 00 00 F0 00 00 00 00 00 ................
00 00 00 00 00 00 00 03 6F 72 69 67 6E 83 C0 7F ........orign勐 .
33 D2 73 69 6D 76 6D 01 00 05 00 00 F0 00 00 00 3襻 imvm.........
00 00 00 00 00 01 00 00 00 01 02 00 00 F0 00 00 ................
00 00 F3 FF FF FF 00 00 00 00 03 6F 72 69 67 6E ...........orign
3B C2 75 52 73 69 6D 76 6D 01 00 07 00 00 F0 00 ;聊 Rsimvm.......
00 00 00 00 00 00 00 00 00 00 00 04 02 00 00 F0 ................
00 00 00 00 EC FF FF FF 00 00 00 00 03 01 00 07 ................
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 01 ................
07 00 00 F0 00 00 00 00 01 00 00 00 00 00 00 00 ................
04 6F 72 69 67 6E 83 C0 7F 33 D2 73 69 6D 76 6D .orign勐 .3襻 imvm
01 00 05 00 00 F0 00 00 00 00 00 00 00 00 01 00 ................
00 00 01 02 00 00 F0 00 00 00 00 F2 FF FF FF 00 ................
00 00 00 03 6F 72 69 67 6E 3B C2 75 3F 73 69 6D ....orign;聊 ?sim
76 6D 01 00 07 00 00 F0 00 00 00 00 00 00 00 00 vm..............
00 00 00 00 04 02 00 00 F0 00 00 00 00 EC FF FF ................
FF 00 00 00 00 03 01 00 07 00 00 F0 00 00 00 00 ................
00 00 00 00 00 00 00 00 01 07 00 00 F0 00 00 00 ................
00 02 00 00 00 00 00 00 00 04 6F 72 69 67 6E 83 ..........orign.
C0 7F 33 D2 73 69 6D 76 6D 01 00 05 00 00 F0 00 ..3襻 imvm.......
00 00 00 00 00 00 00 01 00 00 00 01 02 00 00 F0 ................
00 00 00 00 F1 FF FF FF 00 00 00 00 03 6F 72 69 .............ori
67 6E 3B C2 75 2C 73 69 6D 76 6D 01 00 07 00 00 gn;聊 ,simvm.....
F0 00 00 00 00 00 00 00 00 00 00 00 00 04 02 00 ................
00 F0 00 00 00 00 EC FF FF FF 00 00 00 00 03 01 ................
00 07 00 00 F0 00 00 00 00 00 00 00 00 00 00 00 ................
00 01 07 00 00 F0 00 00 00 00 03 00 00 00 00 00 ................
00 00 04 6F 72 69 67 6E 83 C0 7F 33 D2 73 69 6D ...orign勐 .3襻 im
76 6D 01 00 05 00 00 F0 00 00 00 00 00 00 00 00 vm..............
01 00 00 00 01 02 00 00 F0 00 00 00 00 F0 FF FF ................
FF 00 00 00 00 03 6F 72 69 67 6E 3B C2 75 19 8D ......orign;聊 ..
85 E0 FB FF FF 50 33 C9 73 69 6D 76 6D 01 00 05 呸 ...P3蒺 imvm...
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 E8 41 49 00 ............鍰 I.
05 01 00 07 00 00 F0 00 00 00 00 00 00 00 00 00 ................
00 00 00 04 04 00 00 F0 00 00 00 00 34 03 00 00 ............4...
00 00 00 00 06 6F 72 69 67 6E E8 27 DD FF FF EB .....orign.....
09 73 69 6D 76 6D 65 6E 64 00 .simvmend
```

上面是指令码，其中带有 simvm 标识的为VMP代码，带有orgin的为原始代码，对于simvm的代码走虚拟机执行引擎，对于orgin标识通过反编译引擎，并重定位执行。

493F70 会调用473A04 上面的逻辑主要在本函数执行。

```
CODE:00473A04 sub_473A04 proc near ; CODE XREF: CODE:0049403B↓p
CODE:00473A04
CODE:00473A04 var_2C = dword ptr -2Ch
CODE:00473A04 var_8 = dword ptr -8
CODE:00473A04 var_4 = dword ptr -4
CODE:00473A04 var_s0 = dword ptr 0
CODE:00473A04 arg_4 = dword ptr 0Ch
CODE:00473A04 arg_8 = dword ptr 10h
CODE:00473A04
CODE:00473A04 push ebp
CODE:00473A05 mov ebp, esp
CODE:00473A07 leave
CODE:00473A08 push dword ptr [esp+0]
CODE:00473A0B push [esp+var_s0]
CODE:00473A0E push [esp+4+var_4]
CODE:00473A11 pusha
CODE:00473A12 pushf
CODE:00473A13 call sub_4723EC
CODE:00473A18 mov ebx, eax
CODE:00473A1A xor eax, eax
CODE:00473A1C mov [ebx+1050h], eax
CODE:00473A22 mov eax, 0FFFFFFFFh
CODE:00473A27 mov [ebx+1054h], eax
CODE:00473A2D mov [esp+2Ch+var_8], ebx
CODE:00473A31 mov eax, [esp+2Ch+arg_8] ; eax = encryptCodeAddr
CODE:00473A35 mov [esp+2Ch+var_4], eax
CODE:00473A39 mov eax, [esp+2Ch+arg_4]
CODE:00473A3D mov [esp+2Ch+var_s0], eax
CODE:00473A41 mov eax, [ebx+1090h]
CODE:00473A47 cmp eax, 0
CODE:00473A4A jnz short loc_473A57
CODE:00473A4C mov eax, offset codeCrypt
CODE:00473A51 mov [ebx+1090h], eax
CODE:00473A57
CODE:00473A57 loc_473A57: ; CODE XREF: sub_473A04+46↑j
CODE:00473A57 popf
CODE:00473A58 popa
CODE:00473A59 call HandleCode
CODE:00473A5E pushf
```

```
CODE:00473A5F pushf
CODE:00473A60 pushf
CODE:00473A61 pusha
CODE:00473A62 pushf
CODE:00473A63 jmp short loc_473A89
CODE:00473A65 ; ---------------------------------------------------------------------------
CODE:00473A65
CODE:00473A65 loc_473A65: ; CODE XREF: sub_473A04+98↓j
CODE:00473A65 mov eax, [ebx+1058h]
CODE:00473A6B mov [esp+2Ch+var_s0], eax
CODE:00473A6F mov eax, [ebx+1050h]
CODE:00473A75 mov [esp+2Ch+var_4], eax
CODE:00473A79 mov [esp+2Ch+var_8], ebx
CODE:00473A7D popf
CODE:00473A7E popa
CODE:00473A7F call HandleCode
CODE:00473A84 pushf
CODE:00473A85 pushf
CODE:00473A86 pushf
CODE:00473A87 pusha
CODE:00473A88 pushf
CODE:00473A89
CODE:00473A89 loc_473A89: ; CODE XREF: sub_473A04+5F↑j
CODE:00473A89 call sub_4723EC
CODE:00473A8E mov ebx, eax
CODE:00473A90 mov eax, [ebx+1050h]
CODE:00473A96 cmp eax, [ebx+1054h]
CODE:00473A9C jb short loc_473A65
CODE:00473A9E mov eax, [ebx+1050h]
CODE:00473AA4 mov [esp+30h], eax
CODE:00473AA8 mov eax, [esp+2Ch+var_2C]
CODE:00473AAB mov [esp+2Ch+var_s0], eax
CODE:00473AAF popf
CODE:00473AB0 popa
CODE:00473AB1 add esp, 8
CODE:00473AB4 popf
CODE:00473AB5 retn 0Ch
CODE:00473AB5 sub_473A04 endp
CODE:00473AB5
```

而其核心函数为HandleCode（472EAC）

```
int __usercall HandleCode@<eax>(int a1@<ebx>, int a2@<edi>, int a3@<esi>, int a4, int a
{
unsigned int v15; // et0
int (__fastcall *v16)(unsigned int *); // ST10_4
unsigned int v18; // [esp+10h] [ebp-30h]
int v19; // [esp+30h] [ebp-10h]
void **v20; // [esp+34h] [ebp-Ch]
int *v21; // [esp+38h] [ebp-8h]
int v22; // [esp+3Ch] [ebp-4h]
unsigned int vars0; // [esp+40h] [ebp+0h]
void *retaddr; // [esp+44h] [ebp+4h]

vars0 = a3;
v22 = a2;
v21 = (int *)&vars0;
v20 = &retaddr;
v19 = a1;
v15 = __readeflags();
*(_DWORD *)(a13 + 4188) = retaddr;
DecryptData(a13, a14, a12, (char *)a15, &v21, (int *)&v20, &v19);
vars0 = v18;
__writeeflags(v18);
__writeeflags(v18);
return v16(&vars0);
}
```

其中 "16(&vars0);"语句为执行重定位后的代码。而虚拟机以及原始代码重定位在DecryptData函数（472EAC）如下：

```
DWORD *__stdcall DecryptData(int a1, int orgAddr, int a3, char *dataOfFile, _DWORD *a5,
{
_DWORD *global; // ebx
int v8; // esi
int v9; // edi
int v10; // eax
int v11; // ecx
int v12; // esi
int v13; // esi
int v14; // eax
int v15; // ST18_4
unsigned int v17; // [esp+1Ch] [ebp-77Ch]
void *v18; // [esp+20h] [ebp-778h]
int *v19; // [esp+24h] [ebp-774h]
```

```
int v20; // [esp+34h] [ebp-764h]
int v21; // [esp+38h] [ebp-760h]
char data[1024]; // [esp+3Fh] [ebp-759h]
int v23; // [esp+440h] [ebp-358h]
char v24; // [esp+444h] [ebp-354h]
char v25; // [esp+544h] [ebp-254h]
unsigned __int8 orgCodeLen; // [esp+77Ah] [ebp-1Eh]
char v27; // [esp+77Bh] [ebp-1Dh]
char *nextData; // [esp+77Ch] [ebp-1Ch]
char *data1; // [esp+780h] [ebp-18h]
_BYTE *v30; // [esp+784h] [ebp-14h]
int outDataLen; // [esp+788h] [ebp-10h]
int v32; // [esp+78Ch] [ebp-Ch]
int opCodeLen; // [esp+790h] [ebp-8h]
_DWORD *v34; // [esp+794h] [ebp-4h]
int vars0; // [esp+798h] [ebp+0h]

v21 = 0;
v20 = 0;
v30 = 0;
v19 = &vars0;
v18 = &loc_4732B2;
v17 = __readfsdword(0);
__writefsdword(0, (unsigned int)&v17);
global = sub_4723EC();
v34 = (_DWORD *)(vars0 + 8);
data1 = data;
v8 = *(_DWORD *)off_497644;
*(_DWORD *)(*(_DWORD *)off_497644 + 52) = 0;
v27 = 0;
while ( 1 )
{
callCodeDecrypt((int)global, dataOfFile, data, &outDataLen);
while ( 1 )
{
while ( *(_DWORD *)(v8 + 52) == 1 )
{
orgCodeLen = 0;
nextData = callVmpHandle(v8, (_BYTE *)orgAddr, data, (int)&orgCodeLen);
opCodeLen = nextData - data;
GetNextValueByLen(&dataOfFile, nextData - data);
GetNextValueByLen(&orgAddr, orgCodeLen);
callCodeDecrypt((int)global, dataOfFile, data, &outDataLen);
}
if ( ifCode_simvm_orign(v8, &data1) != 1 )// 不是 simvm
```

```
break;
data1 = data;
v27 = 1;
simvm_Handle(global, v34);
}
data1 = data;
if ( !v27 )
break;
v27 = 0;
orgin_Handle(global, v34);
}
outDataLen = off_498CA0(data, outDataLen, 0x400000, &v23, 4, v17);// 执行原始code
*a5 = outDataLen;
v9 = unknown_libname_29(32 - outDataLen - 6);
data1 = (char *)global + v9 + 4144;
sub_464FA4(data1, data, outDataLen);
*((_BYTE *)global + v9 + outDataLen + 4144) = 104;
GetCharFormTstring((int)&v30, &v25);
if ( (unsigned __int8)sub_472520(v30, global[1047]) )
{
GetCharFormTstring((int)&v30, &v24);
opCodeLen = sub_465C88((int)&str___29[1], (int)v30);
v10 = GetJSLen(v30);
System::__linkproc__ LStrCopy((int)v30, opCodeLen + 1, v10 - opCodeLen, (int)&v30);
opCodeLen = sub_465CEC(v30);
if ( outDataLen - 1 >= 0 )
{
v11 = outDataLen;
v32 = 0;
do
{
*((_BYTE *)global + v9 + v32++ + 4144) = -112;
--v11;
}
while ( v11 );
}
opCodeLen += outDataLen;
v12 = sub_471F04(v8, data, opCodeLen);
}
else if ( sub_465C88((int)&str_CALL_0[1], (int)v30) == 1 )
{
GetCharFormTstring((int)&v30, &v24);
if ( sub_465C88((int)&str_FF_0[1], (int)v30) != 1 )
{
opCodeLen = sub_465C88((int)&str___29[1], (int)v30);
```

```
v32 = opCodeLen - 1;
v13 = (opCodeLen - 1) / 2;
data1 = (char *)global + v9 + v13 + 4144;
v14 = GetJSLen(v30);
System::__linkproc__ LStrCopy((int)v30, opCodeLen + 1, v14 - opCodeLen, (int)&v30);
opCodeLen = sub_465CEC(v30);
v32 = opCodeLen + orgAddr - ((_DWORD)global + v9 + 4144);
sub_464FA4(data1, &v32, outDataLen - v13);
}
opCodeLen = outDataLen;
v12 = outDataLen;
}
else
{
opCodeLen = outDataLen;
v12 = outDataLen;
}
*a7 = opCodeLen;
opCodeLen = a3;
data1 = (char *)global + v9 + outDataLen + 4145;
sub_464FA4(data1, &opCodeLen, 4u);
*((_BYTE *)global + v9 + outDataLen + 4149) = 0xC3u;
*a6 = (int)global + v9 + 0x1030;
global[1044] = orgAddr + *a7;
global[1046] = &dataOfFile[v12];
GetCharFormTstring((int)&v21, &v25);
v15 = v21;
GetCharFormTstring((int)&v20, &v24);
sub_4736D4((int)global, *a6, orgAddr, v15, v20, outDataLen);
__writefsdword(0, v17);
v19 = (int *)&loc_4732B9;
System::__linkproc__ LStrArrayClr(&v20, 2);
return System::__linkproc__ LStrClr(&v30);
}
```

其中如下代码为执行VMP代码：

```
while ( 1 )
{
callCodeDecrypt((int)global, dataOfFile, data, &outDataLen);
while ( 1 )
{
while ( *(_DWORD *)(v8 + 52) == 1 )
```

```
{
orgCodeLen = 0;
nextData = callVmpHandle(v8, (_BYTE *)orgAddr, data, (int)&orgCodeLen);
opCodeLen = nextData - data;
GetNextValueByLen(&dataOfFile, nextData - data);
GetNextValueByLen(&orgAddr, orgCodeLen);
callCodeDecrypt((int)global, dataOfFile, data, &outDataLen);
}
if ( ifCode_simvm_orign(v8, &data1) != 1 )// 不是 simvm
break;
data1 = data;
v27 = 1;
simvm_Handle(global, v34);
}
data1 = data;
if ( !v27 )
break;
v27 = 0;
orgin_Handle(global, v34);
}
```

后半部分为执行原始代码，进行反编译从定位。

下面是虚拟程序相关函数：

```
char *__fastcall callVmpHandle(int unknowGlogal, _BYTE *orgAddr, _BYTE *data, int globa
{
int unknowGlogal_1; // ebx
char *data_2; // [esp+4h] [ebp-8h]
char *data_1; // [esp+8h] [ebp-4h]

data_1 = data;
unknowGlogal_1 = unknowGlogal;
data_2 = data;
if ( *(_DWORD *)(unknowGlogal + 52) == 1 )
{
if ( ifCode_simvm_orign(unknowGlogal, &data_1) == 2 )
{
data_2 = data_1; // orgin
}
else
{
data_2 = data_1;
if ( *data_1 == byte_497214 )
```

```
data_2 = vmpHandle(unknowGlogal_1, data_1, (char *)global);
else
*(_DWORD *)(unknowGlogal_1 + 52) = 2;
}
}
return data_2;
}

char *__fastcall vmpHandle(int unknowGlogal, char *data, char *a3)
{
char *v3; // ST1C_4
int unknowGlogal_1; // edi
char num0Data; // bl
int num6Data; // ebp
char *dataEnd; // [esp+4h] [ebp-38h]
char num5Data; // [esp+8h] [ebp-34h]
int num7Data; // [esp+Ch] [ebp-30h]
int num8Data; // [esp+10h] [ebp-2Ch]
int num9Data; // [esp+14h] [ebp-28h]
int num1Data; // [esp+18h] [ebp-24h]
int num2Data; // [esp+1Ch] [ebp-20h]
int num3Data; // [esp+20h] [ebp-1Ch]
int num4Data; // [esp+24h] [ebp-18h]
char *data_1; // [esp+28h] [ebp-14h]

v3 = a3;
unknowGlogal_1 = unknowGlogal;
dataEnd = data + 0x24;
data_1 = data;
GetNextValueByLen(&data_1, 1);
num0Data = *data_1;
GetNextValueByLen(&data_1, 1);
num1Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
num2Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
num3Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
num4Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
num5Data = *data_1;
GetNextValueByLen(&data_1, 1);
num6Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
num7Data = *(_DWORD *)data_1;
```

```
GetNextValueByLen(&data_1, 4);
num8Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
num9Data = *(_DWORD *)data_1;
GetNextValueByLen(&data_1, 4);
*v3 = *data_1;
if ( num0Data || num5Data )
{
if ( num0Data && num5Data )
{
sub_471CDC(num0Data, num9Data, num8Data, num7Data, num6Data, num4Data, num3Data);
}
else if ( !num0Data || num5Data )
{
if ( !num0Data && num5Data )
call_vmp_add_reg_value(
unknowGlogal_1,
num1Data,
num2Data,
num3Data,
num4Data,
num6Data,
num7Data,
num8Data,
num9Data,
num5Data);
}
else
{
call_vmp_memcpy_regAndReg_num(
unknowGlogal_1,
num1Data,
num2Data,
num3Data,
num4Data,
num6Data,
num7Data,
num8Data,
num9Data,
num0Data);
}
}
else
{
sub_471C08(unknowGlogal_1, num1Data, num2Data, num3Data, num4Data, num6Data, num7Data,
```

```
}
return dataEnd;
}
```

```
int __fastcall call_vmp_add_reg_value(int unknowGlogal, int num1Data, int num2Data, int
{
if ( num1Data && num6Data )
return vmp_add_reg_value(unknowGlogal, num1Data, num3Data, num6Data, num8Data, num5Data
if ( num1Data && num7Data )
return vmp_add_reg_value(unknowGlogal, num1Data, num3Data, num7Data, num8Data, num5Data
if ( num1Data && num9Data )
return vmp_add_reg_value(unknowGlogal, num1Data, num3Data, num9Data, 0, 0);
if ( num2Data && num6Data )
return vmp_add_reg_value(unknowGlogal, num2Data, num3Data, num6Data, num8Data, num5Data
if ( num2Data && num7Data )
return vmp_add_reg_value(unknowGlogal, num2Data, num3Data, num7Data, num8Data, num5Data
if ( num2Data )
{
if ( num9Data )
unknowGlogal = vmp_add_reg_value(unknowGlogal, num2Data, num3Data, num9Data, 0, 0);
}
return unknowGlogal;
}
```

就是一些简单的寄存器读取以及加减乘除运算。

下面的函数是针对代码解密的，就是异或0xFF.

```
DWORD *__fastcall codeCrypt(int a1, int dataOfFile, char *outData, _DWORD *a4)
{
int v4; // ecx
char v5; // zf
_DWORD *v6; // eax
unsigned int v8; // [esp+8h] [ebp-20h]
void *v9; // [esp+Ch] [ebp-1Ch]
int *v10; // [esp+10h] [ebp-18h]
int v11; // [esp+20h] [ebp-8h]
char *outData1; // [esp+24h] [ebp-4h]
int savedregs; // [esp+28h] [ebp+0h]

v11 = 0;
```

```
v10 = &savedregs;
v9 = &loc_473839;
v8 = __readfsdword(0);
__writefsdword(0, (unsigned int)&v8);
*a4 = 0x400;
outData1 = outData;
v4 = 0;
do
{
outData1[v4] = ~*(_BYTE *)(dataOfFile + v4);
++v4;
}
while ( v4 != 0x400 );
GetCharFormTstring((int)&v11, outData1 + 2);
System::__linkproc__ LStrCmp(v11, &str_simvmend[1]._top);
if ( v5 ) // 判断代码是否结束
{
v6 = sub_4723EC();
v6[0x415] = v6[0x414];
}
__writefsdword(0, v8);
v10 = (int *)&loc_473840;
return System::__linkproc__ LStrClr(&v11);
}
```

三、sn

如下vm指令的主要功能就是解密最后的字符串"a123"

```
73 69 6D 76 6D 01 00 05 00 .......simvm....
00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 C5 3F 49 00 05 ............I..
01 00 07 00 00 F0 00 00 00 00 00 00 00 00 01 00 ................
00 00 01 05 00 00 F0 00 00 00 00 00 00 00 00 00 ................
00 00 00 02 01 01 02 00 00 F0 00 00 00 00 F3 FF ................
FF FF 00 00 00 00 00 07 00 00 F0 00 00 00 00 00 ................
00 00 00 00 00 00 00 03 01 00 07 00 00 F0 00 00 ................
00 00 00 00 00 00 01 00 00 00 01 05 00 00 F0 00 ................
00 00 00 01 00 00 00 00 00 00 00 03 01 01 02 00 ................
00 F0 00 00 00 00 F2 FF FF FF 00 00 00 00 00 07 ................
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 03 ................
01 00 07 00 00 F0 00 00 00 00 00 00 00 00 01 00 ................
```

```
00 00 01 05 00 00 F0 00 00 00 00 02 00 00 00 00 ................
00 00 00 03 01 01 02 00 00 F0 00 00 00 00 F1 FF ................
FF FF 00 00 00 00 00 07 00 00 F0 00 00 00 00 00 ................
00 00 00 00 00 00 00 03 01 00 07 00 00 F0 00 00 ................
00 00 00 00 00 00 01 00 00 00 01 05 00 00 F0 00 ................
00 00 00 03 00 00 00 00 00 00 00 03 01 01 02 00 ................
00 F0 00 00 00 00 F0 FF FF FF 00 00 00 00 00 07 ................
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 03 ................
01 00 07 00 00 F0 00 00 00 00 00 00 00 00 00 00 ................
00 00 04 02 00 00 F0 00 00 00 00 EC FF FF FF 00 ................
00 00 00 03 01 00 07 00 00 F0 00 00 00 00 00 00 ................
00 00 00 00 00 00 01 07 00 00 F0 00 00 00 00 00 ................
00 00 00 00 00 00 00 03 6F 72 69 67 6E 83 C0 7F ........orign勉 .
33 D2 73 69 6D 76 6D 01 00 05 00 00 F0 00 00 00 3襻 imvm.........
00 00 00 00 00 01 00 00 00 01 02 00 00 F0 00 00 ................
00 00 F3 FF FF FF 00 00 00 00 03 6F 72 69 67 6E ...........orign
3B C2 75 52 73 69 6D 76 6D 01 00 07 00 00 F0 00 ;聊 Rsimvm.......
00 00 00 00 00 00 00 00 00 00 00 04 02 00 00 F0 ................
00 00 00 00 EC FF FF FF 00 00 00 00 03 01 00 07 ................
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 01 ................
07 00 00 F0 00 00 00 00 01 00 00 00 00 00 00 00 ................
04 6F 72 69 67 6E 83 C0 7F 33 D2 73 69 6D 76 6D .orign勉 .3襻 imvm
01 00 05 00 00 F0 00 00 00 00 00 00 00 00 01 00 ................
00 00 01 02 00 00 F0 00 00 00 00 F2 FF FF FF 00 ................
00 00 00 03 6F 72 69 67 6E 3B C2 75 3F 73 69 6D ....orign;聊 ?sim
76 6D 01 00 07 00 00 F0 00 00 00 00 00 00 00 00 vm..............
00 00 00 00 04 02 00 00 F0 00 00 00 00 EC FF FF ................
FF 00 00 00 00 03 01 00 07 00 00 F0 00 00 00 00 ................
00 00 00 00 00 00 00 00 01 07 00 00 F0 00 00 00 ................
00 02 00 00 00 00 00 00 00 04 6F 72 69 67 6E 83 ..........orign.
C0 7F 33 D2 73 69 6D 76 6D 01 00 05 00 00 F0 00 ..3襻 imvm.......
00 00 00 00 00 00 00 01 00 00 00 01 02 00 00 F0 ................
00 00 00 00 F1 FF FF FF 00 00 00 00 03 6F 72 69 .............ori
67 6E 3B C2 75 2C 73 69 6D 76 6D 01 00 07 00 00 gn;聊 ,simvm.....
F0 00 00 00 00 00 00 00 00 00 00 00 00 04 02 00 ................
00 F0 00 00 00 00 EC FF FF FF 00 00 00 00 03 01 ................
00 07 00 00 F0 00 00 00 00 00 00 00 00 00 00 00 ................
00 01 07 00 00 F0 00 00 00 00 03 00 00 00 00 00 ................
00 00 04 6F 72 69 67 6E 83 C0 7F 33 D2 73 69 6D ...orign勉 .3襻 im
76 6D 01 00 05 00 00 F0 00 00 00 00 00 00 00 00 vm..............
01 00 00 00 01 02 00 00 F0 00 00 00 00 F0 FF FF ................
FF 00 00 00 00 03 6F 72 69 67 6E 3B C2 75 19 8D ......orign;聊 ..
85 E0 FB FF FF 50 33 C9 73 69 6D 76 6D 01 00 05 呸 ...P3蒺 imvm...
00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 E8 41 49 00 ............鍒 I.
```

```
05 01 00 07 00 00 F0 00 00 00 00 00 00 00 00 00 ................
00 00 00 04 04 00 00 F0 00 00 00 00 34 03 00 00 ............4...
00 00 00 00 06 6F 72 69 67 6E E8 27 DD FF FF EB .....orign.'....
09 73 69 6D 76 6D 65 6E 64 00 .simvmend
```

sn为 ： simpower91a123

> 明日将是第四题《拯救单身狗》的解析，请大家守时关注哦~

看雪CTF晋级赛Q1 题解列表

1. 2019KCTF 晋级赛Q1 | 第一题点评及解题思路
2. 2019KCTF 晋级赛Q1 | 第二题点评及解题思路
3. 英雄榜 | 2019 看雪CTF 晋级赛Q1 排行榜出炉!

- End -

开奖啦!

昨日的赠书活动详情戳👉开奖 | Android软件安全权威指南【签名版】，下图为截止至今天下午5点的留言点赞截图：

恭喜第一名 **极目楚天舒** 获得 《**Android软件安全权威指南** 》签名版一本，请在微信公众号后台发送您的收件信息：收件人、收件地址、联系电话，小编安排给您发货 ~

注：中奖后一周内未发来获奖信息者将视为自动放弃。（本环节为福利环节，奖品不重复赠送，望谅解。点赞截止时间及中奖用户以截图为准。）

公众号ID：ikanxue

官方微博：看雪安全

商务合作：wsc@kanxue.com


戳原文，查看更多精彩writeup！

阅读原文