看雪.纽盾 KCTF 2019 Q2 | 第一题点评及解题思路

KCTF 看雪学院 6月25日

2019看雪纽盾KCTF晋级赛Q2经过十四天的激烈比拼，于昨日正午12点整正式宣告结束。

攻破难题时战士们豪情万丈，意气风发，未能攻破也不要灰心，我们一起来看下第一题的题目解析，积蓄力量，再接再厉！

本次比赛我们特意为大家设置了一个故事背景：

> 时间快进到100年后的地球，一束耀眼的白光划破长空，天空随即被撕开一个犹如黑洞般深不见底的黑洞，一艘巨大的宇宙飞船从这黑洞中冲出来，悬浮在高空中，随即来自外星球的宇宙大军也开始源源不断的涌入地球。
>
> 各个国家纷纷开启防御模式，拿出最强武器，对着来犯的外星人发起进攻。但显然，人类的武器撑不了多久，人类最后一片净土--新西兰，已经一片荒芜，正在缓慢沉入海底。
>
> 留给人类的时间不多了。最好的方法是逃离地球。你需要借助时间之轮，开启进入另一个平行时空的入口，拯救人类于水火之中。
>
> 而启动时间之轮，需要集齐9个能量宝石，方能开启。快点行动起来吧！

题目简介

▼

北京地球上的人们还在沉睡，此时的地球早已没有了996，每人每天只需要工作2～3小时，其余都交给人工智能，世界一片祥和。人们在这样和平安逸的生活中逐渐放松警惕。以至于面对突如其来的外来入侵，毫无防备。

你作为海军陆战队的一员，刚刚从与外星人对抗的一线撤离下来。手机信箱中突然出现了一封神秘来信，称里面含有时间之轮的秘密。但是必须成功解出这封信的密码才能看到……

本题围观人数高达5282人，人气颇高，攻破人数也达到了197人，为本次比赛开了个好彩头！

攻破此题的战队排名一览：

排名	战队名	破解时间	获取积分
1	打打酱油	760s	60.00
2	我是小垃圾	873s	46.76
3	AceHub	896s	46.21
4.	A2	1045s	43.18
5.	pwn_it	1180s	41.10
6.	金左手	1318s	39.42
7.	Ginkgo	1383s	38.74
8.	主人和他的女仆们	1435s	38.24
9.	ZhuGeliang	1503s	37.64
10.	gxkyrftx	1523s	37.48

题目名称 第一题：神秘来信

出题战队 Vagaeth

题目简介

时间快进到100年后的地球，一束耀眼的白光划破长空，天空随即被撕开一个犹如黑洞般深不见底的黑洞，一艘巨大的宇宙飞船从这黑洞中冲出来，悬浮在高空中，随即来自外星球的宇宙大军也开始源源不断的涌入地球。
各个国家纷纷开启防御模式，拿出最强武器，对着来犯的外星人发起进攻。但显然，人类的武器撑不了多久，人类最后一片净土--新西兰，已经一片荒芜，正在缓慢沉入海底。
留给人类的时间不多了。最好的方法是逃离地球。你需要借助时间之轮，开启进入另一个平行时空的入口，拯救人类于水火之中。
而启动时间之轮，需要集齐9个能量宝石，方能开启。快点行动起来吧！

第一题：神秘来信

地球上的人们还在沉睡，此时的地球早已没有了996，每人每天只需要工作2～3小时，其余都交给人工智能，

看雪CTF 评委 点评

▼

这道题难度较低，作为一道签到题被放在了第一题。其考察的内容是“异常（Exception）”的产生。

原理是：正确的flag可以触发除0异常。程序逻辑本身并不复杂。也有队伍采用穷举的办法完成此题。

解题思路

本题解题思路是将两位小伙伴的解题思路整合而成的。

第一部分由 **jackandkx** 提供：

签到题，F5一下。

输入长度为6，最后3位为353，前3位的和为149。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
int val; // esi
unsigned int v4; // kr00_4
unsigned int i; // ecx
unsigned __int8 input[6]; // [esp+10h] [ebp-3Ch]
CPPEH_RECORD ms_exc; // [esp+34h] [ebp-18h]

val = 0;
printf((int)"请输入序列号:\n");
scanf("%s", input);
v4 = strlen((const char *)input);
if ( v4 < 7 && input[5] == '3' && input[4] == '5' && input[3] == '3' && input[2] + inp
```

```
{
i = 0;
if ( v4 )
{
do
val = input[i++] + 16 * val - 48;
while ( i < v4 );
}
ms_exc.registration.TryLevel = 0;
printf((int)"error!\n");
while ( 1 )
;
}
printf((int)"error\n");
return 0;
}
```

F5的代码不完整，直接看汇编。函数开头设置异常处理函数：

```
.text:00401260 push ebp
.text:00401261 mov ebp, esp
.text:00401263 push 0FFFFFFFEh
.text:00401265 push offset stru_41CC98
.text:0040126A push offset __except_handler4
.text:0040126F mov eax, large fs:0
.text:00401275 push eax
```

处理函数显示"success"，所以需要产生异常。

```
.rdata:0041CC98 stru_41CC98 dd 0FFFFFFE4h ; GSCookieOffset
.rdata:0041CC98 ; DATA XREF: _main+5↑o
.rdata:0041CC98 dd 0 ; GSCookieXOROffset ; SEH scope table for function 401260
.rdata:0041CC98 dd 0FFFFFFB4h ; EHCookieOffset
.rdata:0041CC98 dd 0 ; EHCookieXOROffset
.rdata:0041CC98 dd 0FFFFFFFEh ; ScopeRecord.EnclosingLevel
.rdata:0041CC98 dd offset loc_401373 ; ScopeRecord.FilterFunc
.rdata:0041CC98 dd offset sucesss ; ScopeRecord.HandlerFunc
```

6位数字转化为16进制数：

```
.text:00401330 movzx eax, [ebp+ecx+input]
.text:00401335 shl esi, 4
.text:00401338 add esi, 0FFFFFFD0h
.text:0040133B add esi, eax
.text:0040133D inc ecx
.text:0040133E cmp ec
```

16进制数与地址0x401353相减，作为除数，让除数等于0就能产生异常。

```
.text:0040134E call loc_401354
.text:0040134E ; ---------------------------------------------------------------------------
.text:00401353 db 0EBh
.text:00401354 ; ---------------------------------------------------------------------------
.text:00401354
.text:00401354 loc_401354: ; CODE XREF: _main+EE↑j
.text:00401354 pop eax
.text:00401355 sub eax, 0
.text:00401358 sub esi, eax
.text:0040135A div esi
```

故key为401353

第二部分由 **微笑明天** 提供：

简单介绍一下SEH。

参考加密与解密

1、功能

SEH实际包含两个主要功能：结束处理（termination handling）和异常处理（exception handling)。

每当你建立一个try块，它必须跟随一个finally块或一个except块。一个try块之后不能既有finally块又有except块。但可以在try-except块中嵌套try-finally块，反过来也可以。

__try,__finally关键字用来标出结束处理程序两段代码的轮廓。

不管保护体（try块）是如何退出的。不论你在保护体中使用return，还是goto，或者是longjump，结束处理程序（finally块）都将被调用。在try使用__leave关键字会引起跳转到try块的结尾。

2、TIB结构

在用户模式下，TIB(ThreadInformationBlock)位于TEB的头部。而TEB是操作系统为了保存每个线程的私有数据创建的，每个线程都有自己的TEB。

```
nt!_TEB
+0x000 NtTib : _NT_TIB
+0x01c EnvironmentPointer : Ptr32 Void
```

我们看一下TIB的结构

```
typedef struct _NT_TIB //sizeof 1ch
{
00h struct _EXCEPTION_REGISTRATION *ExceptionList; //SEH链入口
04h PVOID StackBase; //堆栈基址
08h PVOID StackLimit; //堆栈大小
0ch PVOID SubSystemTib;
union {
PVOID FiberData;
10h DWORD Version;
};
14h PVOID ArbitraryUserPointer;
18h struct _NT_TIB *Self; //本NT_TIB结构自身的线性地址
}NT_TIB;
```

我们看到，ExceptionList在TIB的头部。而在X86下，TEB总是由fs:[0]指向的。

ExceptionList是一个链表的结构。画了一个流程图便于理解：

next是下一个链的地址。如果next的值是FFFFFFh,表示是链表的最后一个节点，该节点的回调函数是系统设置的一个终结处理函数，所有无人值守的异常都会到达这里。

异常处理函数可以是自定义的函数，系统有一个默认的函数，但我们可以自定义一个异常处理函数，让它来处理。

但是得先安装自定义函数才能使用。

我们可以写一个异常处理的例子：

```
//Powered by HAPPY
#include <Windows.h>
#include <iostream>
int exception_memory_access_violation(LPEXCEPTION_POINTERS p_exinfo)
{
```

```
if (p_exinfo->ExceptionRecord->ExceptionCode == EXCEPTION_ACCESS_VIOLATION)
{
return EXCEPTION_EXECUTE_HANDLER; //handle this exception
}
else return EXCEPTION_CONTINUE_SEARCH; //Do not handle this exception
}

int main()
{
char* mem = 0;
std::cout << "Hello World!\n";
__try {
*mem = 0; //throw exception
}
__except (exception_memory_access_violation(GetExceptionInformation())) //handler
{
puts("Memory error in except");
}
}
```

我们可以将其编译后反汇编研究下except的代码以及是如何安装SEH的，限于篇幅，我们不做深入探究。

看雪CTF晋级赛Q2 精彩回顾

1、终曲 · 看雪.纽盾 KCTF 2019 Q2 圆满落幕，精彩回顾！

2、 [看雪.纽盾 KCTF] 最后冲刺，前进吧！战士！

3、 [看雪.纽盾 KCTF] 赛况直播 | 谁能逆风翻盘？

4、赛况直播 | 当大佬开始发力后……

- End -

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