CheatEngine-实验吧CTF题库writeup



之前我是从来没听说过CE这个软件的,我一直是那种玩游戏都不敢开作弊器外挂的人×

这次去网上下载了CE,先找了CE自带的游戏攻略跟着做完了一遍,然后才做的这个题目,其实步骤都是一样的。

Virtual Game
欧迎你来到 Virtual Gmae。
顾名思义这是一个虚拟的游戏,你要做的就是使用CheatEngine工具把他当成真实的游戏并作弊通关。
游戏一共有9个步骤,当你完成所有步骤,你就会得到flag。
说问题外话这个虚拟的游戏是由CheatEngine的Tutorial改编而来的。
好了,点击"下一步"按钮进入下一个步骤。
Enjoy the game, and try to capture the flag.
By crr27

然后我们把CE打开,用左上角的按钮打开这个游戏,以便之后的修改

🐔 Cheat Engine 6.5		
File Edit Table D3D	Help 0000398C-Virtual_Game.exe	Ś
Foult 0 Address Value	 Process List 000033E0-ThunderPlatform.exe 00002DFC-SearchProtocolHost.exe 00002B30-MSDEV.EXE 00003068-sublime_text.exe 00003068-sublime_text.exe 00002DAC-WeChat.exe 00002DAC-WeChat.exe 000036F4-WeChatWeb.exe 00003528-vmware-usbarbitrator64.exe 000027A4-svchost.exe 0000398C-Virtual_Game.exe 	Undo Scan Settings Unrandomizer 0000 Enable Speedhack
3 Memory View 激活 描述	Open Cancel Attach debugger to process Create process Open file Process List Window List Network Process List(long)	iment Digits ng Add Address Manually
Advanced Options		Table Extras

开始闯关。

STEP2: 精确值扫描:

步骤2 步骤2	<u> </u>
步骤 2: 精确值扫描	*
现在你已经在 Cheat Engine 中打开了 训练教程 ,为我们进入到下一步作好了准备。	
在这个窗口的左下方的 健康:XXX ,每次你点击 "打我" 时,健康值便会减少。	
要进入下一关,你必须找到这个数值并把它改成 1000 。	
有不少方法都可以找到这个数值的位置,但我会告诉你一个最简单的方法,"精确数值扫描":	
首先确认数值类型设置为2字节或4字节,当然,设置成1字节也可以的,但最终修改它的时候便会有点麻烦了(虽然很容易解决),如果该地址后边的字节数值都是 O ,那么你设置成 8 字节也未尝不可,不过在这我们就不必尝试了。单浮点数,双浮点数, 以及其他的扫描方在这里行不通的,因为它们储存数值的方式不同。	
(注: 1个字节所表示的最大数值为十进制的 255 ,十六进制为 FF ,2个字节所表示的最大数值为十进制的 65535 ,十六进 制表示为 FF FF ;通常在游戏中很少有用8个字节表示数值)	Ŧ
健康: 100 丁一步	_

输入100,点击首次查询后按钮变成新的查询



点击游戏里的打我,然后看值的改变,把改变后的值填好,点继续搜索。

<u>F</u> ile <u>E</u> dit Tab	ole D3D	<u>H</u> elp	
風 🖻 🖪 🛛			0000398C-Virtual_Game.exe
Found: 1			~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Address	Value	Pre	新的扫描 Next Scan Undo Scan 📷 📷
020D00FC	96	96	Settings Settings
T			Hex 95
			Scan Type 精确数值 ▼
1			Value Type 4字节 👻
			Memory Scan Options 📃 Unrandomizer
			Start 00000000000000000000000000000000000
			Stop 7fffffffffff
			Viitable Executable
			CopyOnWrite
			Fast Scan 4
			Last Digits Pause the game while scanning

然后点中间下方的小箭头,添加到下面。把值修改为1000即可进入下一关。

激活	描述	地址	类型	数值	
	无描述	020D00FC	4 字节	1000	

€ 步骤3	٤
步骤 3: 未知的初始值	*
OK,看来你已经理解了怎样利用精确数值扫描并找到一个数值了,让我们进入下一步。	
在上一关中我们知道初始数值的大小,所以我们利用精确数值扫描,但这一关中我们仅有一个进度条,我们并不知道它开始时的数值。	ш
我们只知道这个数值在0到500之间,并且每次你点 "打我" 之后你会减些健康值,每次减的健康值会显示在进度条的上方。	
同样有好几种方法可以找这个数值,(例如使用"数值减少了"的扫描方式),但我只教你使用最简单的方法,"未知的初 始值"和"减少的数值"。	
因为你不知道现在数值的多少,利用精确数值进行扫描便派不上了用场,所以选择扫描方式为"未知初始数值",数值类型仍 然选择 4 字节(这是因为大多数WINDOWS应用程序都使用 4 字节存放数据)。	
点击"首次扫描"并等它扫描完成。	
	Ŧ
打我	
	_



因初始值未知,在此选择未知的初始值,之后在游戏中选择打我,提示减少值,在继续扫描中选择减少了,并填写减少值

	Found: 1				2
	Address	Value	Pre	新的扫描 Next Scan	Undo Scan Conference
L	020D0100	361	361	数值:	Settings
L				🗖 Hex 🥫 🔶	
				Scan Type 数值减少了…	▼ ■ 至少 xx%
				Value Type 4 字节	•
Ш				Memory Scan Options	

找到了地址,添加到下方修改即可。

STEP4: 浮点数

€ 步骤4	x
歩骤 4: 浮点数	*
在前面的教程中,我们使用字节的方式来扫描,但有些游戏使用了"浮点数"的来存储数值(这么做是为了给新手制造一些麻烦,让他们没那么容易修改游戏)。	
浮点数是带有小数点的数字(如5.12或11321.1)。	
正如本窗口中的健康和弹药,两者都以浮点方法储存数据,不同的是,健康为单精度浮点数,而弹药为双精度浮点数。	
点击 "打我" 可以减少一些健康值,而点击 "开火" 可以消耗掉 0.5 的弹药。	
你得把这两者都修改到5000或者更多才能过关。	
"猪确教值"扫描的方式虽然可以完成本关的工作,但也许你应该试试其它的扫描方式。	
	Ŧ
「 野約 000字点」: 100 <u> ガベ</u>	

首先查找健康值的地址,题目提示为浮点数,按下图设置后点击首次扫描,再按照第一关的方式点击"打我"按钮来改变健康值从 而寻找到地址。

(首次扫	描 Ne	ext Scan		Undo Scan	Setting:
		100				
Sca	n Type	精确数值		•	🔘 🔿 Rounded	l (default
Valu	e Type	浮点数🔺		•	C Rounded	l (extrem
	⊢ Memoi	ry Scan Ip	tions		🤊 🧿 Truncate	d
	Start	000	00000000	00000	Simple val	ues only
	Stop	7ff:	fffffff	fffff	📃 Unrandom	izer
	inW 🔽	table	Execut	able	📃 Enable Sp	eedhacł

修改火药数方法类似,两值添加到下方后把值改为5000即可过关。

新的挑	描 Next Scan	Undo Sc
	数值:	
	100	
Scar Type	精确数值 ▼	🔵 🔘 Rou
Value Type	双浮点数	🔘 Rou
Leino	ry Scan Options	🔍 💽 Trun
Start	000000000000000000000000000000000000000	Simpl
Stop	7fffffffffffffff	Unrar
	itable 🔲 Executable	Enab

STEP5: 代码查找

€ 步骤5
步骤 5:代码查找
某些游戏重新开始时,数据会存储在与上次不同的地方,甚至,游戏的过程中数据的存储位置也会变动,在这种情况下,你还是可以简单几步搞定它。
这次,我将尽里阐述如何运用 代码查找 的功能。
下方的数值在你每次启动 Tutorial.exe 的时候都会存放在不同的位置,所以正常的(固定的)地址列表就不管用了。
我们要先找到这个数值的存储地址(要如何去做,确信不用我再啰嗦了)。
当你找到地址后,右击 CB 中的这个地址,在菜单中选择 "找出是什么改写了这个地址"的选项,会弹出一个空白的窗口。
接着,点击 Iutorial.exe 窗口上的 "改变数值" 按钮,并且回到 CE ,如果操作没问题,在刚才弹出的空白窗口中,会出现 一些汇编代码,选中它,点击 "替换" 按钮,将它替换成什么也不做的代码,同时,修改后的代码也将放置在 高级选项 的 代码列 表 中去(当你保存地址列表时它将会同时保存)。
•
100 改变数值 下一步

在查找100这个数值时记得把查找类型从双精度型改回4字节

l	激活	描述	地址	类型	数值	
		无描述	020D0110	4字节	551	

在下方选中这个地址,然后右键"find out what writes to this address",即"找出是什么改写了这个地址",我这里电脑截图有问题,截不到右键的东西,点击后弹出一个对话框,问是否使用ce调试器加载当前进程,选择是。

此时出现一个新的对话框,我们需要通过改变值来找到访问了这个地址的代码,所以点击游戏里的"改变数值",然后对话框会显示出访问了这个地址的信息。如图

🐔 下列代码写入到 020D0110	×
C Instruction 1 00410885 - A3 10010D02 - mov (020D0110),eax	Replace
	Show disassembler
	Add to the codelist
	More information
	copy memory
۰	
004108AF - 89 45 F8 - mov [ebp-08],eax 004108AF - 89 45 FC - mov eax,[ebp-04] 004108B5 - A3 10010D02 - mov [020D0110],eax << 004108BA - A1 10010D02 - mov eax,[020D0110] 004108BF - 39 45 F8 - cmp [ebp-08],eax	
EAX=00000170 EBX=100E16D8 ECX=00000000 EDX=00000000	停止
<	

选中地址后点击右侧的replace,按题目要求把原来代码删除,改为nop,即什么都不做,然后点击右下方的停止使游戏继续下去即可。

😰 步骤6	x
步骤 6: 指针:	
上一步解释了如何使用 代码查找 功能对付变化位置的数据地址,但这种方法往往不能达到预期的效果,所以,我们需要用到 指针。	=
在本关的 Tutorial.exe 窗口下面有两个按钮,一个会改变数值,另一个不但能改变数值而且还会改变数值在内存中存储的位置。	
在这一步,你不需要懂汇编,但如果你懂的话会很有帮助。	
首先找到数值的地址,然后再查找是什么改写了这个地址。再次改变数值,CB 便可以将找到的汇编代码列出来,双击一行汇编 代码(或选择它 并点击 "详细信息"),打开的 详细信息窗口 将显示详细的信息,来告诉你当这个指令运行时发生了什么事情。	Ā
如果在这条汇编指令中,没看到方括号([])的存在,我们就应该换另外一条汇编代码查看详细信息,如果看到了方括号,那 就表示我们可能找到了需要的指针。	
除了 详细信息窗口我们暂时保留以外(如果你已经关掉了,那么要记好方栝号中间的代码),剩余的窗口请关闭掉,回到CB白	E +
100 改变数值 下一步 改变指针	

€ 步骤6	×
当扫描完成时,它可能返回一个或几百个地址。大多数情况下你想要的会排在地址列表中的最上面。	*
点击 手工添加地址 按钮,并在勾选"指针"选项。	
添加地址窗口 将发生变化,多出了"Address of Pointer"和"Offset (Hex)"的文本框,在"Address of Pointer"那一栏 中填入刚才扫描到的地址。	
如果汇编指令中的方栝号中存在计算(例如:[esi+12]),那么应当把数值部分填在"Offset Ofex)"的文本框中,如果不存 在,则让它保持为 O 。	Е
如果看上去是更复杂的计算指令的话,举例说明一下:	
[BAX*2+EDX+00000310] eax=4C 并且 edx=00801234 (在详细信息窗口的下方,会列出寄存器中的值)	
这种情况下 EDX 便是数值的指针,而 BAX*2+00000310 则是它的偏移里,所以你要填在 "Offset (Hex)" 将是 2*4C +00000310=3A8 (这些都是在十六进制下计算的,你可以使用WIXDOWS的计算器,在科学方式下用十六进制计算)。	Ŧ
	_
100 改变数值 下一步 改变指针	

€ 步骤6	
[EAX*2+EDX+00000310] eax=4C 并且 edx=00801234 (在详细信息窗口的下方,会列出寄存器中的值) 这种情况下 EDX 便是数值的指针,而 EAX*2+00000310 则是它的偏移里,所以你要填在"Offset (Hex)"将是 2*4C +00000310=3A8 (这些都是在十六进制下计算的,你可以使用WINDOWS的计算器,在科学方式下用十六进制计算)。 点击确定,这个地址便会加到 CE主窗口下方的列表上,如果没做错,在地址处将显示 P->xxxxxxxx,而 xxxxxxx 会和你扫描 到的地址数值是一致的,如果不一致,那么你可能哪里出错了。 现在,改变那条指针地址的数值 5000 并锁定它,,然后点击Tutorial.exe窗口上的"改变指针"按钮,如果一切正确,那么 "下一步"按钮将变成可点击状态了。 提示: 你也可以使用"指针扫描"的方式来找到这个指针地址。	
100 改变数值 下一步 改变指针	

题目要求略长。首先我们按照前面的方式寻找到数值的地址添加到下面,然后按照上一步的方式启动调试器。

	🐔 下列代码写入到 005AE008	×
	C Instruction 1 035B10B1 - 89 14 81 - mov [ecx+eax*4],edx	Replace
	Ť	Show disassembler
l	'	Add to the codelist
		More information
		copy memory
	035B10AB - 8B 4D 08 - mov ecx.[ebp+08] 035B10AE - 8B 55 0C - mov edx.[ebp+0C] 035B10B1 - 89 14 81 - mov [ecx+eax*4].edx << 035B10B4 - 8B 45 0C - mov eax.[ebp+0C] 035B10B7 - 5F - pop edi	
	EAX=00000030 EBX=100E16D8 ECX=005ADF48 EDX=000000E5	停止

然后按照题目要求点击详细信息。

🐔 Extra info		
035B10AB - mov 035B10AE - mov >> 035B10B1 - mov 035B10B4 - mov 035B10B7 - pop	ecx,[ebp+08] edx,[ebp+0C] [ecx+eax*4],edx eax,[ebp+0C] edi	
要查找的地址指针值	可能是 005ADF48 🔶	
EAX=00000030	EDX=000000E5	EBP=0018F398 F
EBX=100E16D8	ESI=100E9F78	ESP=0018F34C
ECX=005ADF48	EDI=0018F398	EIP=035B10B4 S
The registers shown here	are AFTER the instruction	has been executed

标红的部分用来算偏移量,在题目中按照规律求即可。本题中偏移量为eax*4,使用电脑自带计算器求得偏移量为C0。要查找的地址指针值处右键,可将地址复制下来。复制之后点击OK关闭界面。将刚才的地址放到上面搜索。

首次扫描 Next Scan Undo Scan
数值:
Mex 005ADF48
Scan Type 精确数值 ▼
Value Type 4字节 ▼
Memory Scan Options
Start 00000000000000000000000
Stop 7ffffffffffff
Vritable Executable
CopyOnWrite
V Fast Scan 4
Last Digits
Add Address M.

扫描后左边出现两个值,依次试一下。选中第一个值,点击右下角"Add Address Manually",先勾选POINT,然后在下方填入查找到的地址和偏移量



这时我们观察到指向的数值与下方的数值是一致的



将第二条同样添加到下方。并且选中前方的锁定,然后把数值改为5000.在游戏中选择改变指针,过关。



STEP7: 代码注入

€ 步骤7					×
步骤 7: 代码注入:					•
代码注入是将一小段代码注入到目标说	4程中的技巧,然后他	使进程执行你写入的	代码。		
在这一步教程中,你将有一个健康值利 增加2点的健康值。	口——个每按——次将减少	少1点健康值的按钮,	你的任务是利用 代码	注入, 使得每按-	-次按钮 ≡
开始查找这个地址,然后查看是什么在	E改写它("找出是什	么改写了这个地址")。		
当你看到那条减少数值的汇编代码后, 快捷键 Ctrl+a),选择"模板"中的"代 的地址,你可以手工输入它)。	_选择 "显示反汇编税 码注入" 。CII将自动	呈序″, 然后打开」 住成一部分汇编代码	自动汇编窗口 (菜单- 好为你输入指令做好准	工具->自动汇编 9 售备(如果CE没有约	妭 按下 合出正确
注意 alloc 这部分代码,它会为你的 ,很可能导致系统崩溃,幸运的是,这种情	代码分配出——小块空 \$况在 win2000 以后	白的内存,过去,在 的操作系统得到改善	Win2000 之前的系统 。	,这种行为存在安	全隐患
也要注意 line newmem 这部分代码,	原代码,以及用文本	"此处放置你的代码	9″ 标示出的空白部分,	正如你猜测的,有	主这儿可 👻
健康: 100		下一步			

▶ 步骤7	
推荐你使用"ADD"汇编指令,下面是一些示例:	1
"ADD [OO901234],9″ 使 [OO901234] 地址的值增加9 "ADD [ESP+4],9″ 使地址指针 [ESP+4] 的值增加9	
在本关的情况下,你可以使用相同的手法去处理 减健康值那条原代码方栝号之间的部分。	
提示:	
,推荐你从原代码中删除减健康值的那行代码, 否则你得加3点健康值(你增加了3点,原代码减去1点,最终结果才会增加2点) ,这样看上去很容易让人迷惑,但最终方案还是由你来决定好了。	
提示:	
某些游戏中,原代码可能在多条指令之外,有时候(并非一向如此),它可能由不同的地方跳转至你的指令中并结束运行,其 结果可能引起未知的错误,如果出现了这种情况,通常应当查看附近的那些跳转指令,进行修改,或者在尝试使用不同地址进行 代	
码注入,确认无误后便可以将你修改的代码注入原代码中了。	1
健康 : 100	

首先寻找到数值的地址。右键打开调试。根据题目提示,找到减少数值的汇编代码后,选择右侧的"Show disassembler"即"显示 反汇编程序"



打开后选择上方tools栏中的最后一项, auto assmble即自动汇编。同没办法截图。打开后选择模板(template)中的代码注入(code injection)。



其中第一个箭头指向的代码,会为代码分配出一块内存。在win2000后的操作系统改善的,否则容易存在安全隐患使系统崩溃。 下方箭头指向的代码是减少血量的代码。



我们添加一句代码,使每点击一次"打我"按钮,健康值加三,这样健康值加三减一,最后得到题目要求的摁一次增加两点的要求。当然你可以把dec那句减少血量的代码删除。

保存后关闭界面,在游戏中点击"打我"按钮,可以看到每次点击加两点血,进入下一关。

STEP8: 多级指针

🗊 步骤8 📃	
歩骤 8: 多级指针:	-
此步骤,将解释如何使用多级指针。	
在第 6 步,你已经清楚1级指针的概念和用途,并可以利用数值的首个地址找到存放数据真正的基址。	Ξ
在本关中,你将看到4级指针,它由第一个指针指向第二个指针,再由第二个指针指向第三个指针,由第三个指针指向第四个指针,最终指向健康值的真正地址。	
开始的几步与在第 6 步中的操作基本相同。找出是什么访问了这个地址,然后分析汇编指令,查找指针地址中的数值,以及它 的偏移里,将它们记下来。但这次你按数值找出的仍然是一个指针,你得依据这些数值,使用同样的操作方法找出指向这个指针的 指针。	
看看是什么访问了你发现的那个指针地址,分析汇编指令, 留意可能的代码和偏移里,并加以利用。	
持续这种过程,直到不能更进一步查找为止(通常基址为静态时,地址将以绿色标示)。	
	Ŧ
100 改变数值 下一步 改变指针	

n 🕑 步骤8 🐂 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘	x
点击 "改变数值" 改变健康值,如果你发现列表中那些指针地址所指向的值发生同样的变化时,那表示你可以试着将基址中的 值更改为 5000,并锁定它,以便完成本关的任务了。	•
备注1: 本步骤也可以使用自动汇编程序脚本或者使用指针扫描器加以解决。	
备注2: 在某些情况下,可以改变CII软件 "代码查找" 的相关设置,当你遇到像 mov eax,[eax] 的指令时,调试程序将显示改变之后的 寄存器中的值,也许利用它更容易找出指针的位置。	
备注3: 你还在读 ?! 当你查看汇编指令时你可能已经注意到,这些指针是在相同的代码块(相同的程序,如果你懂汇编,可以查看程 序的起始代码)位置被读写。这种情况并不总会发生,但是当你在查找某个指针遇到问题的时候,没准能起到很大的用处。	A H
100 改变数值 下一步 改变指针	

采用指针扫描的方式,首先找到数值的地址。然后右键在菜单里选择"Generate Pointermap"。在弹出的新对话框如图选择,保存 文件,然后多次改变指针,筛选得最后的地址。

🐔 指针扫描器 扫描选项			
◎ 地址查找: ◎	数值查找:		
008F5500			
 ✓ 地址必须是32位的 ✓ 仅查找静态地址路径 □ 发现堆栈时停止遍历 	路径		
指针路径可能在此区域	内:		
Ж	至: 		
00000000	7FFFFFFF		
■ 使用数据堆收集改善 □ 在路径中仅允许堆	指针扫描 转和堆地址		
│ └──── │ ☑ 使用上次的指针查找约	は果 /		
🔲 指针必须以具体的偏移	8里结束		
正在扫描的线程:	3	标准	•
最大的偏移里:	2048	最大等级 5	
	确定	取消	

扫描后得到约11w个结果。最小化窗口。在游戏中点击改变指针,然后重新找到数值对应的地址,在指针扫描器中选择重新扫描,输入地址为第二次找到的数值地址。



多次重复操作。然后选中第一个,锁定改5000即可过关。

文件(Y) 指针扫描器(Z)							
4字节 ▼	▼ pointercount:40						
Base Address	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Points to:	-
Virtual_Game.exe+0000F1A8	0	44	64	4D8	160	08489558 = 5000	
Virtual_Game.exe+0000F308	0	44	64	4D8	160	08489558 = 5000	
Virtual_Game.exe+0000FD	0	44	64	4D8	160	08489558 = 5000	
Virtual_Game.exe+00010B54	0	44	64	4D8	160	08489558 = 5000	
Virtual Game.exe+00010C	0	44	64	4D8	160	08489558 = 5000	

ps: 其实这步是pcat大神教的。。在我问为啥是第一个时候,大神说因为爱。。啊啊啊好可爱!!!!

STEP9: 注入++

€ 步骤9	x
│ 步骤 9: 注入++:	-
这一步基本上和步骤 7 的做法是相似的,不过稍稍有点难度。	
本关你必须用一小段代码来修改减少健康值的原代码,并且做到,如果当前时间的秒数大于或等于 30 将健康值设置为1000, 如果当前时间的秒数小于 30 将健康值设置为 2000。	-
我们可以用 API 调用函数获取当前的时间,并利用 自动汇编 完成这项任务,但这里好象有更简单的做法:C语言脚本注入。	
找到健康值的地址,然后打开 CE 脚本引擎 (内存浏览器窗口 快捷键: ctrl+alt+a ,或是点击 工具->脚本引擎)。	
然后,和其它的教程不同的是,这在里我会多给你一些提示(假设你从未接触讨C语言)。	
#include <time.h></time.h>	
struct tm *timep;	Ŧ
健康: 100 下一步	

-
=
-

😢 步骤9	×
点击 "执行" ,点击 Tutorial.exe 窗口上 "打我" 。	•
如果没什么问题的话,健康值会按照当前的时间变动。	
额外信息:	
正如前面所说,CE 支持标准的汇编语言。所以,你可以在代码中使用函数名称来调用函数。	
你也可以在脚本中引用 DLL 文件,例如:	
injectdll(mydll.dll)// 可以使用你喜欢的语言去写DLL文件	
codecave: call functionofmydll jmp exit	=
健康:100打我 下一步	

然后找到数值对应的地址,查看反汇编程序。

T					
a second s	文件(U) 搜索(V) 衫	z件(U) 搜索(V) 视图(W) 调试(X) 工具(Y) 内核工具(Z)			
	00410D3E				
itoriaLiexe 窗口上 打衣 。					
建康值会按照当前的时间变动。	00410D3E ff 0d 20 0	1d dec [021d0120] [00000062]			
	00410D44 68 00 00 0	0 00 push 00000000			
下列代码更改了洗择的地址	×	00 mov ebx,00000210			
1 / 31 04 33C 8X 3 / 25 + 8 3 / 28 /		00 call 004111d4 [->10028E11]			
10d3e - ff 0d 20 01 1d 02 - dec [021d0120]		add esp,04			
	替換	80 push 80000003			
		push edx			
	見示反汇编程序	push eax			
		00 push 0000001			
	S-#+n∞1/4570 Tul ≠	00 call 004111d4 [->10028E11]			
	※加到15時列表	add esp. 10			
		mov [ebp-18],eax			
	详细信息	cmp dword ptr [ebp-18], 1e			
		00jl 00410ef1			
	减 1	. 1 cmp [021d0120],000003e8	-		
		0 inc.00410coc)at 1			
		1112星址=00400000 范围大小=1000 模块=Virtual_Game.exe			
		IB / 5 9F 49 UB / 5 FU I3 UB / 5 82 I2 UB / 5 ° ▼I ID 76 54 27 0D 76 93 3F 0B 76 B1 17 0B 76 &Ã ++T'			
		B 76 91 14 0B 76 46 82 0D 76 99 17 0B 76 ∎S v′			
		ID 76 B6 E1 C1 77 A9 14 0B 76 45 12 0B 76 9¼ v¶á	Áw© vE v		
		B 76 69 51 0B 76 2F 44 0B 76 D8 79 0B 76 vic) v∕D vØy v		
		IB 76 F5 49 UB 76 4E 18 UB 76 36 18 UB 76 w5 vol	. VN V6 V		
		E 76 93 51 0E 76 ED 16 0E 76 0F 77 0D 76 arwey	vevwv) vív) a v		
	停止	B 76 F1 CA 0B 76 7B 51 0B 76 F1 34 0B 76 «Q vñf	v{Q vñ4 v		
		D 76 51 51 0B 76 7C 17 0B 76 A1 D1 0D 76 ÃN vQC) v [viÑ_v		
	00406000 65 35	4E 76 26 19 0E 76 00 00 00 00 E6 FD 59 76	ν æýΥν		
the state of the second second second	004060C0 12 00				
- ALLER AND ALLER	004060D0 09 00	00 00 06 00 00 00 00 00 00 00 00 05 00 00 00			
AND A DESCRIPTION OF A	004060E0 0B 00				
	004060F0 0D 00		···· +		

然后在工具中选择脚本引擎,打开界面、并在窗口中填入提示中的代码。并替换变量为查找到的地址。注意地址前要写上0x,如 图。然后从注入栏中选择"注入到当前进程"

		×1+(1)	(니) (카비카(니	(二)(土(下)	(II) (AL 4H			
-		🗌 🛃 🖻				00000778-Vi	rtual_Game.e	xe
(ED-BOD)		Eound: 1						
响讯(X)	⊥具(Y) 内核⊥具(Z)	- Odridi I				1		
	00410D3E	地址		值		新的扫	1描 再次	扫描
铃	注释	021D01	20	98			值·	
c [021d01	20] [00000062]						00	
sh 000000	000					I nex	99	
v ebx,000	000210					扫描类型	精确数值	
1004111	22 脚本引擎			X		*눈/ㅎㅋﺩㅠ!		
a esp,ue						刻且尖型	47T	
sh edv	文件(X) 视图(Y) 注入(Z)					一内存	扫描选项	
sheax	1 #include <time.h></time.h>					0 6	日本式	
sh 0000	2					©л	м. 	至
v ebx,0	3 struct tm *timen.					00	000000	76666666
00411:	A time t c:							///////
d esp, 10	E c=time(0):						时扫描只读内	内存
v [ebp-:	s c-time(0);	0120.		=		快i	速扫描	深度扫描
p dword	6 Int addresstochange-oxo210	0120;					描时暂停游汉	ť
0410ef1	<pre>/ timep=localtime(&c);</pre>					X		
p [021d	8						\mathbf{O}	
00/10/	9 if (timep->tm_sec>=30)							
	<pre>10 *(int *)addresstochange</pre>	=1000;				地址	类型	值
=00400	11 else					021D0120	4字节	98
F 49 U A 27 D	12 */int *laddragetochange	=2000.						
1 14 d	1			P				
6 E1 C 9 51 C 5 49 C	执行脚本							

```
★ 自动汇编
文件(X) 视图(Y) 模板(Z)
1 //Call this code to execute the script from assembler
2 call 032F00F8
3
4 //eax==0 when successfully executed
5 //'call underc_geterror' to get a pointer to the last generated error buffer
```

点击模板,选择代码注入。然后在弹出对话框中填入第一项的地址。如图

00410D3E	
地址 字节 指令 注释	
00410D3E ff 0d 20 01 1d dec [021d0120] [000000 🐇 Cheat	: Engine 5
0041 44 68 00 00 00 push 00000000	(存得(F)
00410 49 bb 10 02 00 00 mov ebx,00000210	3冊相(E)
004101 4E 00410053 從 自动汇编	
00410D5B 又仟(X) 视图(Y) 模板(Z)	
00410D5C 1 //Call this code to execute the script from a	ssembl
00410D5D 2 call 032F00F8	
00410D62	
00410D67 175時注入模版	
00410D6C	the la
00410D6F 打算在哪条地址上跳转?	CHE IA
00410D72 00410D3E	
00410070	
分配保护=2	
0040600	
0040601	
0040604	
0040609	
0040606	
0040607	

然后得到如下的代码。其中第一个箭头指向的是源代码。第二个箭头指向我们输入的代码。

13	
14	
15	originalcode:
16	dec [021d0120
17	
18	exit:
19	jmp returnhere
20	//Call this code to execute the script from assembler
21	call 032F00F8
22	
23	<pre>//eax==0 when successfully executed</pre>
24	//'call underc_geterror' to get a pointer to the last generated error buffer
۰.	4
	执行

然后我们把call这一条代码放在newmem区域,并且把源代码注释,如图。

文件(X) 视图(Y) 模板(Z)	
7 jmp newmem	
8 nop	
9 returnhere:	
10	
11 newmem: //this is allocated memory, you have read, write, execute access	
12 //place your code here	11
13 call 032F00F8	
14	
15 originalcode:	
16 //dec [021d0120]	
17	
18 exit:	
19 jmp returnere	
20 //Call this code to execute the script from assembler	
22	
22 23 (/cauto the suggestively suggestively	
24 //early unders actessibility executed	
K	
	1
执行	

点击执行,原来代码已被改写,如图。

、 内存浏	览器	C	100.00	And Designed	· Brinsen		
文件(U)	搜索(V) 视图(W	/) 调试(X) 工具(Y)	内核工具(Z)				
	00410D3E						
地址	字节	指令		注释			
00410D3E	e9 bd f2 21 03	jmp 03630000					
00410D43	90	nop					
00410D44	68 00 00 00 00	push 00000000					
00410D49	bb 10 02 00 00	mov ebx,00000210					
00410D4E	e8 81 04 00 00	call 004111d4		[->10028E11]			
00410D53	83 c4 04	add esp,04					
00410D56	68 03 00 00 80	push 80000003					
00410D5B	52	push edx					
00410D5C	50	push eax					
00410D5D	68 01 00 00 00	push 00000001					
00410D62	bb 08 02 00 00	mov ebx,00000208					
00410D67	e8 68 04 00 00	call 004111d4		[->10028E11]			
00410067	20 00 04 00 00	Call 00411104		[->10020E11]			

回到游戏窗口点击"打我"按钮,得到通关的对话框。耶!





创作打卡挑战赛 赢取流量/现金/CSDN周边激励大奖