

VB的变量类型

实战分析

第一部分 基础校验

第二部分 根据用户名计算结果

第三部分 除以圆周率

第四部分 干扰代码

第五部分 关键比较

写出注册机

总结

附上分析过程

【软件名称】：CyberBlade.2.exe

【软件大小】：61.0 KB

【下载地址】：<https://github.com/TonyChen56/160-Crackme>

【加壳方式】：未加壳

【保护方式】：Name/Serial

【编译语言】：VB P-Code

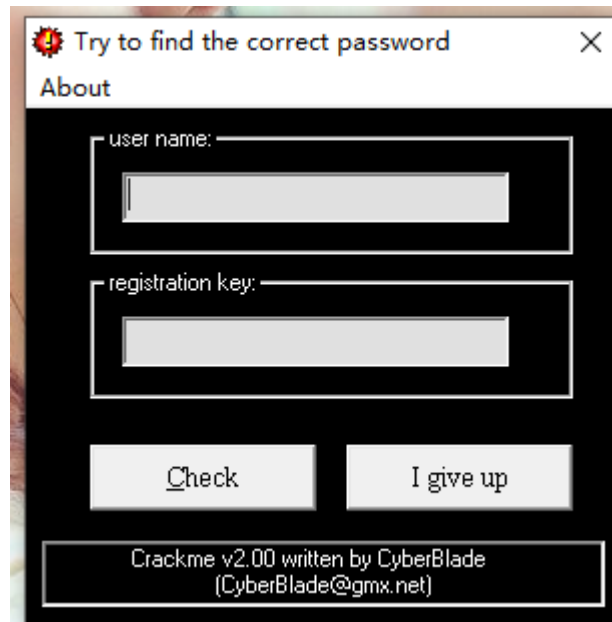
【调试环境】：W10 x64

【使用工具】OD, VBExplorer, VB Decompiler

【破解日期】：2019-5-1

【破解目的】：学习分析P-Code类型的程序，理解P-Code虚拟机的解释过程

【目标程序】：



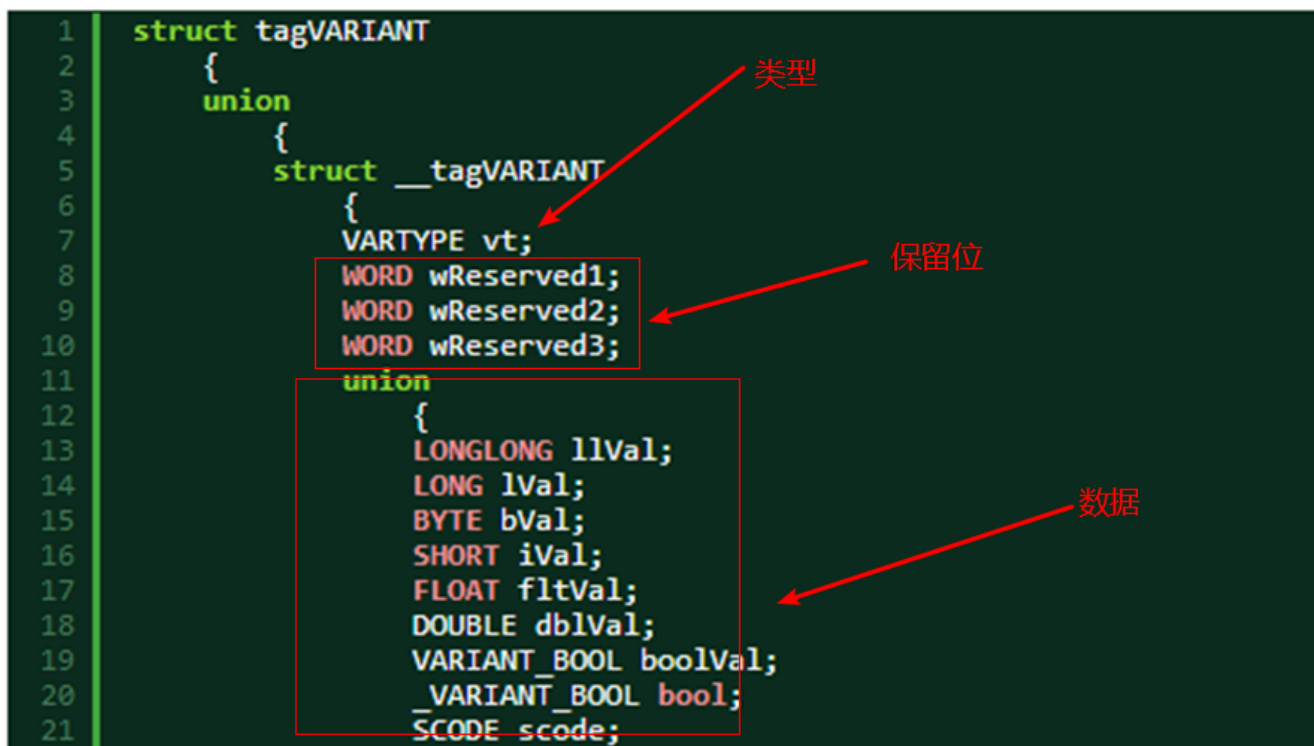
这一次的目标程序的这个Crackme，是160个Crackme里面的第38个。运行时需要 Visual Basic 5.0 运行库支持。这个Crackme的分析，用到了三个工具，每个工具都有各自的用途：

- OD：用于跟踪P-Code伪指令的具体细节及在静态分析过程中无法查看的数据
- VBExplorer 用于查看P-Code伪指令及注释
- VB Decompiler 用于静态查看反汇编的伪代码，减少OD跟踪伪指令的工作量

## VB的变量类型

想要分析这个Crackme，首先需要了解VB变量类型在内存中的存储方式。

VB中的variant类型属于一种结构体,该结构体的前两个字节表示变量的类型,后面有3个WORD是保留的,接下来才是其真正的值，如下图：

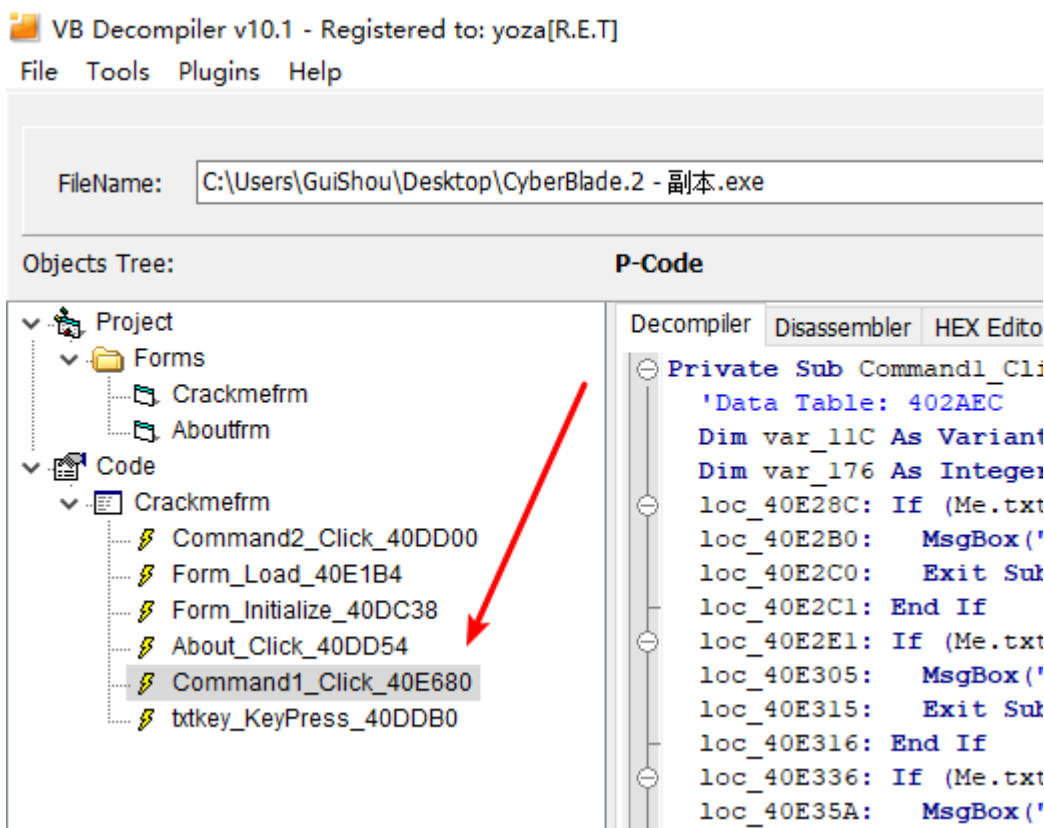


也就是说VB变量中真正的数据是存储在首地址+8的位置处，下图显示了VB的所有的变量类型及含义

Data Type	VB Constant	Value
Empty ( <i>uninitialised</i> )	vbEmpty	0
Null ( <i>no valid data</i> )	vbNull	1
Integer	vbInteger	2
Long integer	vbLong	3
Single-precision floating-point number	vbSingle	4
Double-precision floating-point number	vbDouble	5
Currency value	vbCurrency	6
Date value	vbDate	7
String	vbString	8
Object	vbObject	9
Error value	vbError	10
Boolean value	vbBoolean	11
Variant ( <i>used only with arrays of variants</i> )	vbVariant	12
Byte value	vbByte	17
User-Defined Types ( <i>UDT</i> )	vbUserDefinedType	36
Array	vbArray	8192

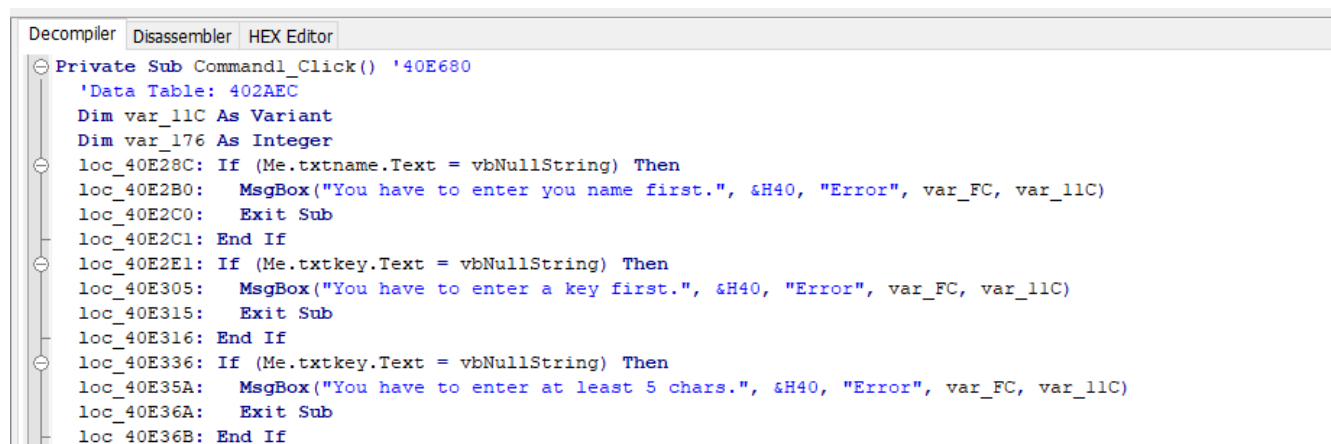
## 实战分析

首先用 VB Decompiler反编译目标程序，找到Check按钮的点击事件，分析整个点击事件的校验过程



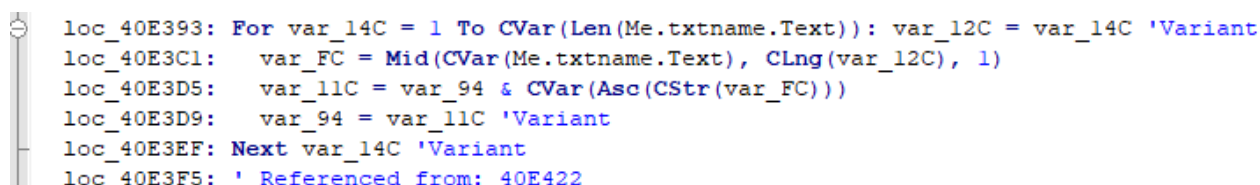
这个程序的校验过程分为五个部分，下面讲解每一个部分的校验过程

## 第一部分 基础校验

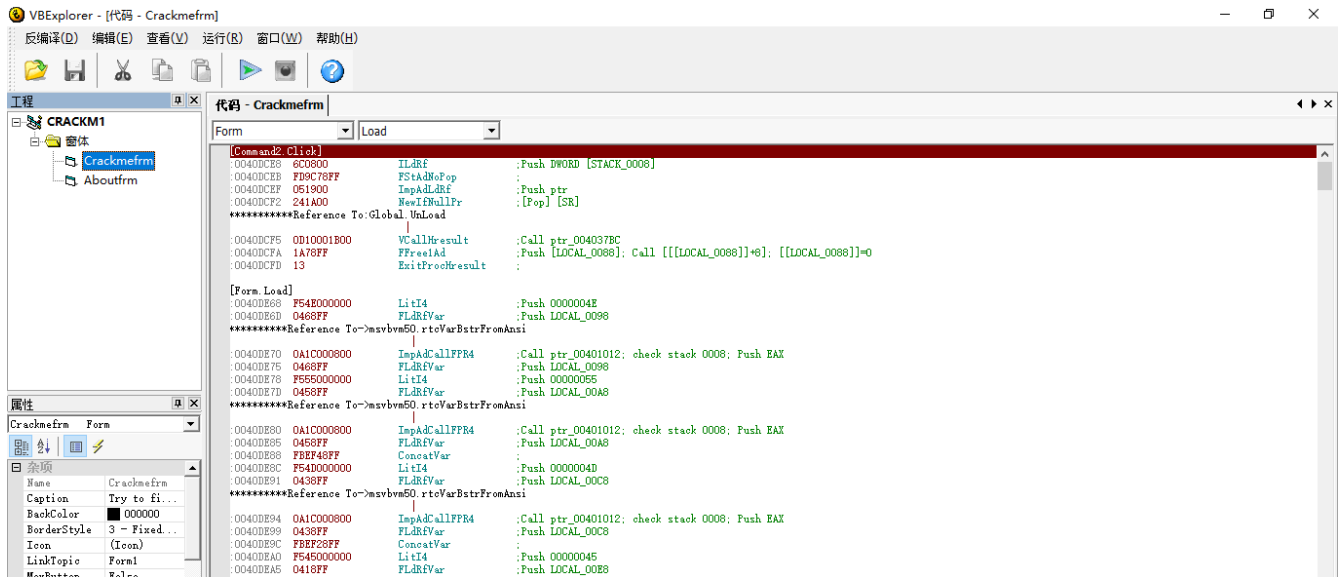


首先根据静态分析的结果可以看到，该程序首先会校验用户名和序列号是否为空，然后判断序列号长度是否小于5个字节，否则提示错误。即使看不懂反汇编后的VB代码，也可以通过字符串知道整个过程。

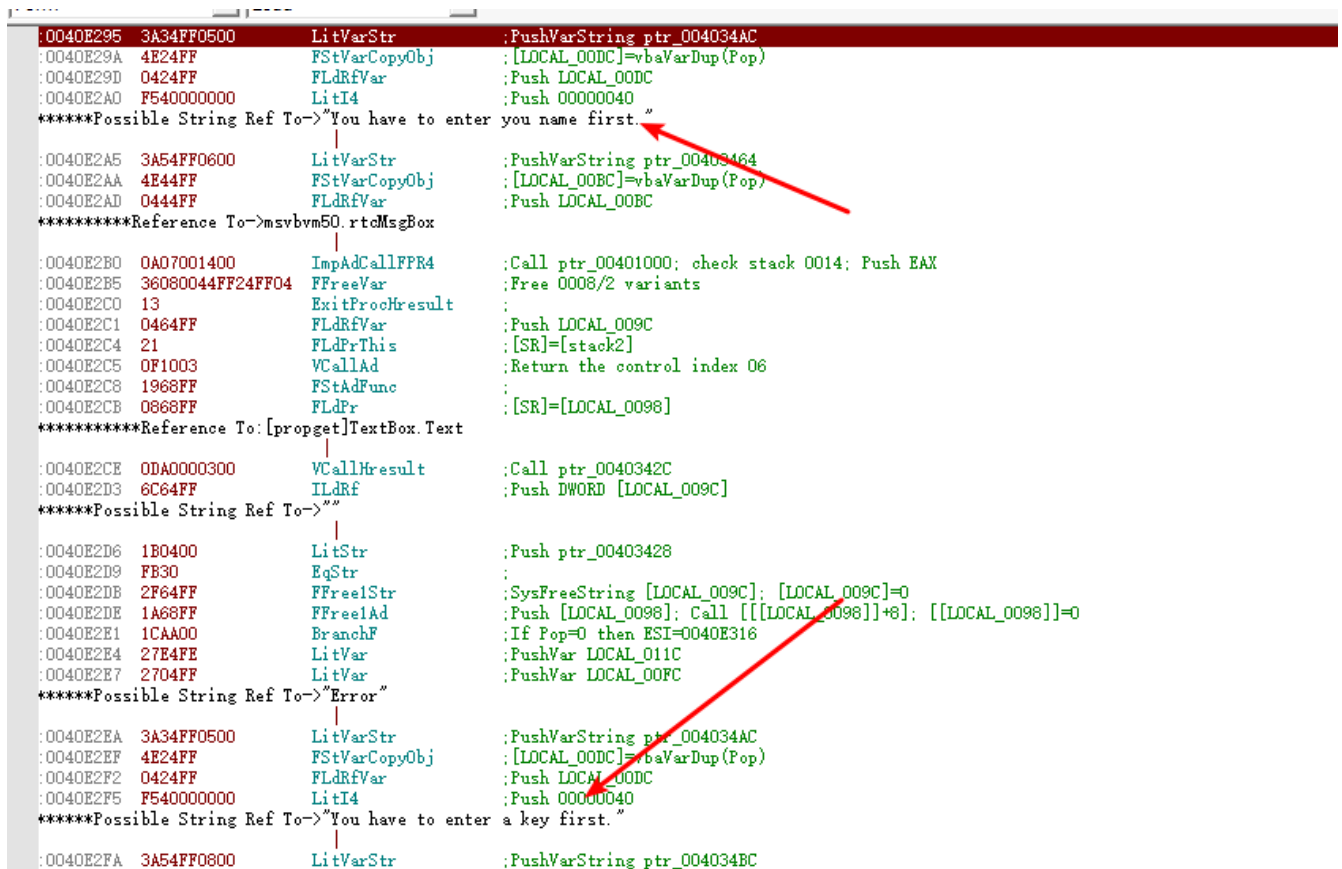
## 第二部分 根据用户名计算结果



第二部分的校验过程看的就不那么清晰了，需要利用OD动态跟踪每一个伪指令的具体操作流程。



用VBExplorer对目标程序进行反编译，一直往下拉，根据字符串直接忽略第一部分的基础校验，来到0040E380的位置



```

0040E37D 0868FF FldPr      :[SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text
0040E380 0DA0000300 VCallHresult :Call ptr_0040342C
0040E385 0864FF ILdRf      :Push DWORD [LOCAL_009C]
0040E388 4A FLenStr  :vbaLenBstr
0040E389 FD6934FF CVarI4     :
0040E38D 2F64FF FFreeIAd   :SysFreeString [LOCAL_009C]; [LOCAL_009C]=0
0040E390 1A68FF FFreeIAd   :Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_0098]]=0
0040E393 FE6B4FE8901 CVar      :
0040E399 0464FF FLdRfVar  :Push LOCAL_009C
0040E39C 21 FLdPrThis :[SR]=[stack2]
0040E39D 0F0003 VCallAd     :Return the control index 02
0040E3A0 1968FF FStAdFunc  :
0040E3A3 0868FF FldPr      :[SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text
0040E3A6 0DA0000300 VCallHresult :Call ptr_0040342C
0040E3AB 046CFF FLdRfVar  :Push LOCAL_0094
0040E3AE 2824FF0100 LitVarI2     :PushVarInteger 0001
0040E3B3 04D4FE FLdRfVar  :Push LOCAL_012C
0040E3B6 FC22 CVarI4     :vbaI4Var
0040E3B8 3E64FF FLdZeroAd :Push DWORD [LOCAL_009C]; [LOCAL_009C]=0
0040E3BB 4644FF CVarStr    :
0040E3BE 0404FF FLdRfVar  :Push LOCAL_00FC
*****Reference To: msvbvm50.rtcMidCharVar
0040E3C1 0A0A001000 ImpAdCallFPR4 :Call ptr_00401006; check stack 0010; Push EAX
0040E3C6 0404FF FLdRfVar  :Push LOCAL_00FC
0040E3C9 FDFE00FE CStrVarVal  :
*****Reference To: msvbvm50.rtcAnsiValueBstr
0040E3CD 080B000400 ImpAdCallI2   :Call ptr_0040100C; check stack 0004; Push EAX
0040E3D2 4434FF CVarI2       :
0040E3D5 FBFEFAFE ConcatVar    :
0040E3D9 FCF66CFF FStVar      :
0040E3DD 2F60FE FFreeIAd   :SysFreeString [LOCAL_0150]; [LOCAL_0150]=0
0040E3E0 1A68FF FFreeIAd   :Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_0098]]=0
0040E3E3 36080044FF24FF04 FFreeVar    :Free 0006/2 variants
0040E3EC 04D4FE FLdRfVar  :Push LOCAL_012C

```

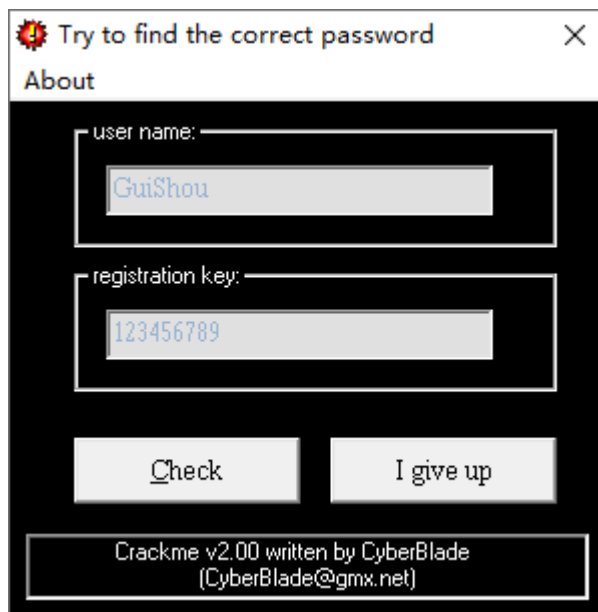
我们可以看到第一个被执行的伪指令是0040E380处的0D，接下来将程序载入OD，数据窗口跟随->0040E380



然后给第一个字节0D下内存访问断点，F9运行

地址	HEX	数据	ASCII
0040E380	0D	A0 00 03 00 6C 64 FF 4A FD 69 34 FF 2F 64 FF	.?□.ld J齣4
0040E390	1A 68 FF FE	68 B4 FE 89 01 04 64 FF 21 0F 00 03	□h 柄逮?□d
0040E3A0	19 68 FF 08	68 FF 0D A0 00 03 00 04 6C FF 28 24	□h □h .?□.
0040E3B0	FF 01 00 04	D4 FE FC 22 3E 64 FF 46 44 FF 04 04	□. □軋?>d F
0040E3C0	FF 0A 0A 00	10 00 04 04 FF FD FE B0 FE 0B 0B 00	... □. □□
0040E3D0	04 00 44 34	FF FB EF E4 FE FC F6 6C FF 2F B0 FE	□.D4 濼 1
0040E3E0	1A 68 FF 36	06 00 44 FF 24 FF 04 FF 04 D4 FE FE	□h 6□.D \$
0040E3F0	7E B4 FE 2D	01 04 6C FF FB EB 44 FF 28 54 FF 09	~逮-□□1 D
0040E400	00 5D FB 74	1C B9 01 04 6C FF FE C4 54 FF 50 45	.]鸛?□1 T
0040E410	52 54 FB 21	09 40 FB BC 44 FF FB E1 24 FF FC F6	RT?.@ D \$
0040E420	6C FF 1E 89	01 04 6C FF FE C1 54 FF 78 56 F8 30	1 ?□1 T x
0040E430	FB 17 44 FF	FC F6 6C FF 04 6C FF 08 08 00 8A 4C	?D 1 □1 □
0040E440	00 FD 69 54	FF FB 9C 44 FF FC F6 6C FF 28 14 FF	.齣T 麥D 1

然后随便输入一个用户名和序列号，点击Check



程序首先会读取一个字节的操作码到AL，

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BE3E1	8945 B4	mov dword ptr ss:[ebp-0x4C], eax		EAX 0000000D
741BE3E4	33C0	xor eax, eax		ECX 00000000
741BE3E6	8A06	mov al, byte ptr ds:[esi]		EDX 0E2330B8
741BE3E8	46	inc esi	CyberBla. 0040E380	EBX FFFFFFFF
741BE3E9	FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]	MSVBVM50. 741BE829	ESP 0019F04C
741BE3F0	8B7D B4	mov edi, dword ptr ss:[ebp-0x4C]		EBP 0019F1D0
741BE3F3	0FB706	movzx eax, word ptr ds:[esi]		ESI 0040E380 Cy
741BE3F6	83C6 02	add esi, 0x2		EDI 0019F0E4
741BE3F9	8B0438	mov eax, dword ptr ds:[eax+edi]		EIP 741BE3E8 MS
741BE3FC	EB DB	jmp short MSVBVM50. 741BE3D9		C 0 ES 002B 32
741BE3FE	0FBF3E	movsx edi, word ptr ds:[esi]		P 1 CS 0023 32
741BE401	0FB746 02	movzx eax, word ptr ds:[esi+0x2]		A 0 SS 002B 32
741BE405	83C6 04	add esi, 0x4		Z 1 DS 002B 32
esi=0040E380 (CyberBla. 0040E380)				
地址	HEX 数据	ASCII	地址	数值
0040E380	0D A0 00 03 00 6C 64 FF 4A FD 69 34 FF 2F 64 FF	?[.ld J詢4	0019F04C	0019F134
0040E390	1A 68 FF FE 68 B4 FE 89 01 04 64 FF 21 0F 00 03	[h 树逮?d	0019F050	0019F0A4
0040E3A0	19 68 FF 08 68 FF 0D A0 00 03 00 04 6C FF 28 24	[h [h ?	0019F054	0019F0F4

:0040E380	0DA0000300	VCallHresult	;Call ptr_0040342C
:0040E385	6C64FF	ILdRf	;Push DWORD [LOCAL_009C]
:0040E388	4A	FnLenStr	;vbaLenBstr
:0040E389	FD6934FF	CVarI4	;
:0040E38D	2F64FF	FFreeIStr	;SysFreeString [LOCAL_009C]; [LOCAL_009C]=0
:0040E390	1A68FF	FFreeIAd	;Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_
:0040E393	FE68B4FE8901	ForVar	;
:0040E399	0464FF	FLdRfVar	;Push LOCAL_009C
:0040E39C	21	FLdPrThis	;[SR]=[stack2]
:0040E39D	0F0003	VCallAd	;Return the control index 02
:0040E3A0	1968FF	FStAdFunc	;
:0040E3A3	0868FF	FLdPr	;[SR]=[LOCAL_0098]
*****Reference To:[propget]TextBox.Text			
:0040E3A6	0DA0000300	VCallHresult	;Call ptr_0040342C

来看下VB Explorer中显示的操作码，后面的注释提示这是在调用一个函数，0D后面的是操作码的参数，接着esi自增1，指向操作码的参数



吾爱破解 - CyberBlade.2.exe - [LCG - 主线程, 模块 - MSVBVM50]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+]  
快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释
741BE3E1	8945 B4	mov dword ptr ss:[ebp-0x4C],eax	
741BE3E4	33C0	xor eax,eax	
741BE3E6	8A06	mov al,byte ptr ds:[esi]	
741BE3E8	46	inc esi	CyberBla.0040E381
741BE3E9	FF2485 94ED1B7	jmp dword ptr ds:[eax*4+0x741BED94]	MSVBVM50.741BE829
741BE3F0	8B7D B4	mov edi,dword ptr ss:[ebp-0x4C]	
741BE3F3	0FB706	movzx eax,word ptr ds:[esi]	
741BE3F6	83C6 02	add esi,0x2	
741BE3F9	8B0438	mov eax,dword ptr ds:[eax+edi]	
741BE3FC	EB DB	jmp short MSVBVM50.741BE3D9	
741BE3FE	0FBF3E	movsx edi,word ptr ds:[esi]	
741BE401	0FB746 02	movzx eax,word ptr ds:[esi+0x2]	
741BE405	83C6 04	add esi,0x4	

ds:[741BEDC8]=741BE829 (MSVBVM50.741BE829)

地址	HEX 数据	ASCII	地址	数值	注释
0040E380	0D A0 00 03	00 6C 64 FF 4A FD 69 34 FF 2F 64 FF	0019F04C	0019F134	
0040E390	1A 68 FF FE	68 B4 FE 89 01 04 64 FF 21 0F 00 03	0019F050	0019F0A4	
0040E3A0	19 68 FF 08	68 FF 0D A0 00 03 00 04 6C FF 28 24	0019F054	0019F0E4	

接着通过一个jmp跳转去执行操作码，0x741BED94是地址跳转表的首地址，eax保存下一条指令的操作码，由于每一个跳转地址是一个DWORD，所以用eax乘以4的值加上跳转表的基地址来索引下一条伪指令的解释单元，我们跟随这个jmp

暂停

首先把[ebp-0x4C]赋值给eax，然后将eax压栈。我们需要知道eax的含义。数据窗口跟随之后，发现是一个指针，再次选中前四个字节，数据窗口跟随DWORD，然后将数据显示方式切换为长型->地址



741BE847	8A46 04	[mov al,byte ptr ds:[esi+0x4]
ds:[0040E381]=00A0		
edi=0019F0E4		
地址	数值	注释
022330B8	740E5A95	MSVBVM50. 740E5A95
022330BC	740DE2A2	MSVBVM50. 740DE2A2
022330C0	740DE258	MSVBVM50. 740DE258
022330C4	7416684A	MSVBVM50. 7416684A
022330C8	74120D55	MSVBVM50. 74120D55
022330CC	740E4380	MSVBVM50. 740E4380
022330D0	740ECDAB	MSVBVM50. 740ECDAB
022330D4	74112647	MSVBVM50. 74112647
022330D8	740F376F	MSVBVM50. 740F376F
022330DC	740ED409	MSVBVM50. 740ED409
022330E0	74101E55	MSVBVM50. 74101E55
022330E4	74183B20	MSVBVM50. 74183B20
022330E8	740E94A4	MSVBVM50. 740E94A4

这其实是一个函数的跳转表，再接着把esi的内容A0赋值给edi，而esi始终执行的是操作码

地址	HEX 数据	反汇编	注释
741BE829	8B45 B4	mov eax,dword ptr ss:[ebp-0x4C]	
741BE82C	50	push eax	
741BE82D	0FB73E	movzx edi,word ptr ds:[esi]	
741BE830	8B00	mov eax,dword ptr ds:[eax]	
741BE832	03C7	add eax,edi	
741BE834	FF10	call dword ptr ds:[eax]	
741BE836	8B55 BC	mov edx,dword ptr ss:[ebp-0x44]	
741BE839	66:F742 76 0200	test word ptr ds:[edx+0x76],0x2	
741BE83F	75 13	jnz short MSVBVM50. 741BE854	
741BE841	0BC0	or eax,eax	
741BE843	78 19	js short MSVBVM50. 741BE85E	
741BE845	33C0	xor eax,eax	
741BE847	8A46 04	mov al,byte ptr ds:[esi+0x4]	

ds:[0223E514]=022330B8	eax=0223E514
------------------------	--------------

地址	数值	注释	地址	数值	注释
0040E381	000300A0		0019F048	0223E514	
0040E385	4AFF646C		0019F04C	0019F134	
0040E389	FF3469FD		0019F050	0019F0A4	
0040E38D	1AFF642F		0019F054	0019F0E4	
0040E391	68FFFE68		0019F058	00000000	

0040E380	0DA0000300	VCallHresult	;Call ptr_0040342C
0040E385	6C64FF	ILDrf	;Push DWORD [LOCAL_009C]
0040E388	4A	FnLenStr	;vbaLenBstr
0040E389	FD6934FF	CVarI4	;
0040E38D	2F64FF	FFreeIStr	;SysFreeString [LOCAL_009C]; [LOCAL_009C]=0
0040E390	1A68FF	FFreeIAd	;Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_0098]]
0040E393	FE68B4FE8901	ForVar	;
0040E399	0464FF	FLdrfVar	;Push LOCAL_009C
0040E39C	21	FLdrfThis	;[SR]=[stack2]
0040E39D	0F0003	VCallAd	;Return the control index 02
0040E3A0	1968FF	FStAdFunc	;
0040E3A3	0868FF	FLdrPr	;[SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text			
0040E3A6	0DA0000300	VCallHresult	;Call ptr_0040342C
0040E3AB	046CFF	FLdrfVar	;Push LOCAL_0094
0040E3AE	00000000	FLdrfThis	;Push [LOCAL_0098]

可以看到这一步实际上是在取操作码的参数A0了

吾爱破解 - CyberBlade.2.exe - [LCG - 主线程, 模块 - MSVBVM50]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BE829	8B45 B4	mov eax,dword ptr ss:[ebp-0x4C]		EAX 022330B8
741BE82C	50	push eax		ECX 00000000
741BE82D	0FB73E	movzx edi,word ptr ds:[esi]		EDX 022330B8
741BE830	8B00	mov eax,dword ptr ds:[eax]	MSVBVM50.740E5A95	EBX FFFFFFFF
741BE832	03C7	add eax,edi		ESP 0019F048
741BE834	FF10	call dword ptr ds:[eax]	MSVBVM50.740E5A95	EBP 0019F1D0
741BE836	8B55 BC	mov edx,dword ptr ss:[ebp-0x44]		ESI 0040E381
741BE839	66:F742 76 0200	test word ptr ds:[edx+0x76],0x2		EDI 000000A0
741BE83F	75 13	jnz short MSVBVM50.741BE854		EIP 741BE832
741BE841	0BC0	or eax,eax		C 0 ES 002B
741BE843	78 19	js short MSVBVM50.741BE85E		P 1 CS 0023
741BE845	33C0	xor eax,eax		A 0 SS 002B
741BE847	8A46 04	mov al,byte ptr ds:[esi+0x4]		Z 0 DS 002B
edi=000000A0 eax=022330B8				

地址	数值	注释	地址	数值	注释
022330B8	740E5A95	MSVBVM50.740E5A95	0019F048	0223E514	
022330BC	740DE2A2	MSVBVM50.740DE2A2	0019F04C	0019F134	
022330C0	740DE258	MSVBVM50.740DE258	0019F050	0019F0A4	
022330C4	7416684A	MSVBVM50.7416684A	0019F054	0019F0E4	
022330C8	74120D55	MSVBVM50.74120D55	0019F058	00000000	
022330CC	740E4380	MSVBVM50.740E4380	0019F05C	00000000	
022330D0	740ECDAB	MSVBVM50.740ECDAB	0019F060	00000000	
022330D4	74112647	MSVBVM50.74112647	0019F064	00000000	
022330D8	740F376F	MSVBVM50.740F376F	0019F068	00000000	

接着取出eax的内容，然后将eax加上edi，eax实际是跳转表的首地址，那么参数A0，就是跳转表的偏移

吾爱破解 - CyberBlade.2.exe - [LCG - m主线程, 模块 - MSVBVM50]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BE829	8B45 B4	mov eax,dword ptr ss:[ebp-0x4C]		EAX 02233158
741BE82C	50	push eax		ECX 00000000
741BE82D	0FB73E	movzx edi,word ptr ds:[esi]		EDX 022330B8
741BE830	8B00	mov eax,dword ptr ds:[eax]	MSVBVM50.7411A5B6	EBX FFFFFFFF
741BE832	03C7	add eax,edi		ESP 0019F048
741BE834	FF10	call dword ptr ds:[eax]	MSVBVM50.7411A5B6	EBP 0019F1D0
741BE836	8B55 BC	mov edx,dword ptr ss:[ebp-0x44]		ESI 0040E381 Cybe
741BE839	66:F742 76 0200	test word ptr ds:[edx+0x76],0x2		EDI 000000A0
741BE83F	75 13	jnz short MSVBVM50.741BE854		EIP 741BE834 MSVB
741BE841	0BC0	or eax,eax		C 0 ES 002B 32位
741BE843	78 19	js short MSVBVM50.741BE85E		P 0 CS 0023 32位
741BE845	33C0	xor eax,eax		A 0 SS 002B 32位
741BE847	8A46 04	mov al,byte ptr ds:[esi+0x4]		Z 0 DS 002B 32位
ds:[02233158]=7411A5B6 (MSVBVM50.7411A5B6)				

地址	数值	注释	地址	数值	注释
0019F134	00000000		0019F048	0223E514	
0019F138	0223E514		0019F04C	0019F134	
0019F13C	00000000		0019F050	0019F0A4	
0019F140	00000000		0019F054	0019F0E4	
0019F144	00000000		0019F058	00000000	
0019F148	00000000		0019F05C	00000000	
0019F14C	0019F1DC		0019F060	00000000	
0019F150	0019F2B8		0019F064	00000000	

接着call eax，我们不需要跟进这个函数，只需要关注栈中的第二个参数0x19F134即可。直接步过这个函数

741BE830	8B00	mov eax,dword ptr ds:[eax]			EBX 1
741BE832	03C7	add eax,edi			ESP (
741BE834	FF10	call dword ptr ds:[eax]			EBP (
741BE836	8B55 BC	mov edx,dword ptr ss:[ebp-0x44]			ESI (
741BE839	66:F742 76 020	test word ptr ds:[edx+0x76],0x2			EDI (
741BE83F	75 13	jnz short MSVBVM50.741BE854			EIP 7
741BE841	0BC0	or eax,eax			
741BE843	78 19	js short MSVBVM50.741BE85E			C 0
741BE845	33C0	xor eax,eax			P 1
741BE847	8A46 04	mov al,byte ptr ds:[esi+0x4]			A 0
堆栈 ss:[0019F18C]=006B58D8					
edx=02230000					
地址	数值	注释	地址	数值	注释
0019F134	006BA36C	UNICODE "GuiShou"	0019F050	0019F0A4	
0019F138	0223E514		0019F054	0019F0E4	
0019F13C	00000000		0019F058	00000000	
0019F140	00000000		0019F05C	00000000	
0019F144	00000000		0019F060	00000000	
0019F148	00000000		0019F064	00000000	
0019F14C	0019F1DC		0019F068	00000000	
0019F150	0019F2B8		0019F06C	00000000	

可以看到栈中的参数显示出了我们刚才输入的用户名，再接着单步到xor eax,eax的地址

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 暂停										快速菜单 Tools BreakPoint->																																																	
地址										HEX 数据										反汇编										注释										寄存器 (FPU)																			
741BE830										8B00										mov eax,dword ptr ds:[eax]																				EAX 00000000																			
741BE832										03C7										add eax,edi																				ECX 02230000																			
741BE834										FF10										call dword ptr ds:[eax]																				EDX 006B58D8																			
741BE836										8B55 BC										mov edx,dword ptr ss:[ebp-0x44]																				EBX FFFFFFFF																			
741BE839										66:F742 76 0200										test word ptr ds:[edx+0x76],0x2																				ESP 0019F050																			
741BE83F										75 13										jnz short MSVBVM50.741BE854																				EBP 0019F1D0																			
741BE841										0BC0										or eax,eax																				ESI 0040E381 CyberBl																			
741BE843										78 19										js short MSVBVM50.741BE85E																				EDI 000000A0																			
741BE845										33C0										xor eax,eax																				EIP 741BE847 MSVBVM50.741BD39E																			
741BE847										8A46 04										mov al,byte ptr ds:[esi+0x4]																				C 0 ES 002B 32位 0																			
741BE84A										83C6 05										add esi,0x5																				P 1 CS 0023 32位 0																			
741BE84D										FF2485 94ED1B74										jmp dword ptr ds:[eax*4+0x741BED94]										MSVBVM50.741BD39E										A 0 SS 002B 32位 0																			
741BE854										B8 689C0000										mov eax,0x9C68																				Z 1 DS 002B 32位 0																			
ds:[0040E385]=6C ('1')																																																											
al=00																																																											
地址										数值										注释										地址										数值										注释									
006B594E										00000000																				0019F050										0019F0A4																			
006B5952										00000000																				0019F054										0019F0E4																			
006B5956										00000000																				0019F058										00000000																			
006B595A										00000000																				0019F05C										00000000																			

这里把eax的值清零了。也就是说第一条伪指令已经执行完成了。

这个就是P-Code虚拟机的解释过程，其中esi始终指向要解释的伪指令，eax保存的是将要解释的伪指令。

0040E37D	0868FF	FLdPr	: [SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text			
0040E380	0DA0000300	VCallHresult	: Call ptr_0040342C
0040E385	6C64FF	ILdRf	: Push DWORD [LOCAL_009C]
0040E388	4A	FnLenStr	: vbaLenBstr
0040E389	FD6934FF	CVarI4	:
0040E38D	2F64FF	FFreeIStr	: SysFreeString [LOCAL_009C]; [LOCAL_009C]=0
0040E390	1A68FF	FFreeIAd	: Push [LOCAL_0098]; Call [[LOCAL_0098]+8]; [[LOCAL_0098]]
0040E393	FE68B4FE8901	ForVar	:
0040E399	0464FF	FLdRfVar	: Push LOCAL_009C
0040E39C	21	FLdPrThis	: [SR]=[stack2]
0040E39D	0F0003	VCallAd	: Return the control index 02
0040E3A0	1968FF	FStAdFunc	:
0040E3A3	0868FF	FLdPr	: [SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text			
0040E3A6	0DA0000300	VCallHresult	: Call ptr_0040342C
0040E3AB	046CFF	FLdRfVar	: Push LOCAL_0094
0040E3AE	2824FF0100	LiIVarI2	: PushVarInteger 0001
0040E3B3	0404FF	FLdRfVar	: Push LOCAL_012C

接着看下一条伪指令，6C是操作码，64FF是参数。后面的注释告诉我们这条伪指令是在将某个DWORD值入栈。继续跟踪

暂停										l e m t w h c p k b r ... s										CyberBla. 0040													
地址		HEX 数据		反汇编		注释		寄存器 (FPU)																									
741BE834		FF10		call dword ptr ds:[eax]				EAX 0000006C																									
741BE836		8B55 BC		mov edx,dword ptr ss:[ebp-0x44]				ECX 02230000																									
741BE839		66:F742 76 0200		test word ptr ds:[edx+0x76],0x2				EDX 006B58D8																									
741BE83F		75 13		jnz short MSVBVM50.741BE854				EBX FFFFFFFF																									
741BE841		0BC0		or eax,eax				ESP 0019F050																									
741BE843		78 19		js short MSVBVM50.741BE85E				EBP 0019F1D0																									
741BE845		33C0		xor eax,eax				ESI 0040E381																									
741BE847		8A46 04		mov al,byte ptr ds:[esi+0x4]				EDI 000000A0																									
741BE84A		83C6 05		add esi,0x5				EIP 741BE84A																									
741BE84D		FF2485 94ED1B74		jmp dword ptr ds:[eax*4+0x741BED94]		MSVBVM50.741BD947		MSVBVM50.741B																									
741BE854		B8 689C0000		mov eax,0x9C68				C 0 ES 002B 32位 0(FFFFFFF)																									
741BE859		E9 3AEBFFFF		jmp MSVBVM50.741BD398				P 1 CS 0023 32位 0(FFFFFFF)																									
741BE85E		66:3D 689C		cmp ax,0x9C68				A 0 SS 002B 32位 0(FFFFFFF)																									
ds:[0040E385]=6C ('1')																																	
al=6C ('1')																																	
地址		HEX 数据		ASCII		地址		数值		注释																							
0040E381		A0 00 03 00		6C 64 FF 4A		FD 69 34 FF		2F 64 FF 1A		?[]1d J端4 /		0019F050		0019F0A4																			
0040E391		68 FF FE 68		B4 FE 89 01		04 64 FF 21		0F 00 03 19		h 柄逮?[]d ! /		0019F054		0019F0E4																			

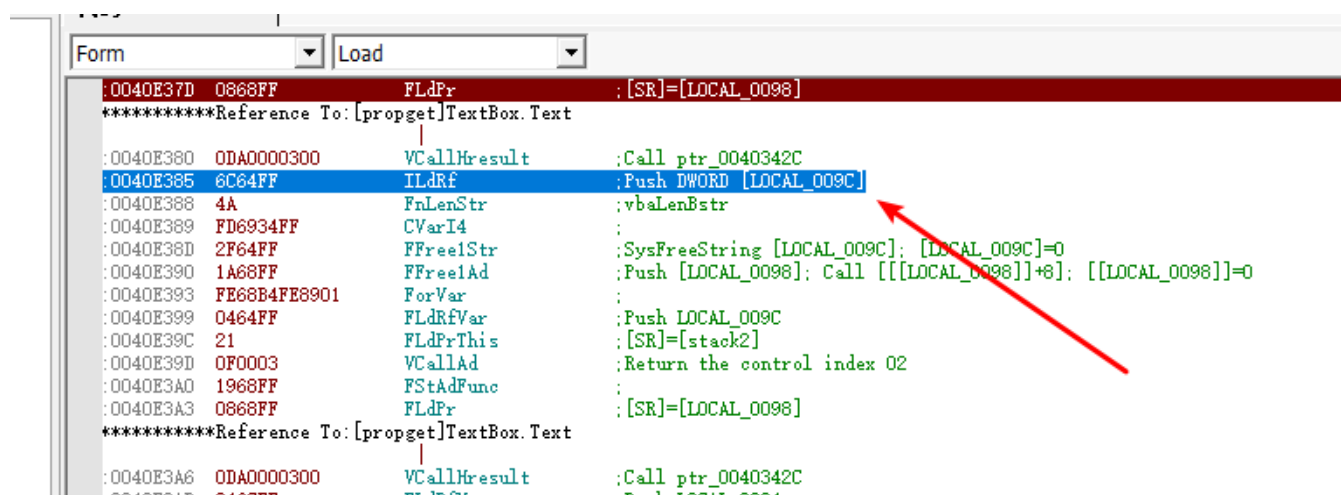
首先取出操作码6C，然后esi+5执行向伪指令参数，接着跳转执行这条伪指令，直接跟进jmp

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BD947	0FBF06	movsx eax,word ptr ds:[esi]		EAX FFFFFFF64
741BD94A	FF3428	push dword ptr ds:[eax+ebp]		ECX 02230000
741BD94D	33C0	xor eax,eax		EDX 006B58D8
741BD94F	8A46 02	mov al,byte ptr ds:[esi+0x2]		EBX FFFFFFFF
741BD952	83C6 03	add esi,0x3		ESP 0019F050
741BD955	FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]		EBP 0019F1D0
741BD95C	0FBF06	movsx eax,word ptr ds:[esi]		ESI 0040E386 CyberBla.004
741BD95F	FF3428	push dword ptr ds:[eax+ebp]		EDI 000000A0
741BD962	C70428 00000000	mov dword ptr ds:[eax+ebp],0x0		EIP 741BD94A MSVBVM50.741
741BD969	33C0	xor eax,eax		C 0 ES 002B 32位 0(FFFFF)
741BD96B	8A46 02	mov al,byte ptr ds:[esi+0x2]		P 0 CS 0023 32位 0(FFFFF)
741BD96E	83C6 03	add esi,0x3		A 0 SS 002B 32位 0(FFFFF)
741BD971	FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]		Z 0 DS 002B 32位 0(FFFFF)
堆栈 ds:[0019F134]=006B836C, (Unicode "Guishou")				

首先取出参数FF64放到eax中，FF64是一个负数，



也就是十进制的9C，这也是为什么VBExplorer里会显示LOCAL\_009C的原因，这个9C代表[ebp-9C]，是个局部变量





CyberBlade-2.exe - [LCG - 主线程, 模块: MSVBVM50]											
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->											
暂停											
地址 HEX 数据 反汇编 注释 寄存器 (FPU)											
741BD947		0FBF06		movsx eax,word ptr ds:[esi]				EAX FFFFFFF64			
741BD94A		FF3428		push dword ptr ds:[eax+ebp]				ECX 02230000			
741BD94D		33C0		xor eax,eax				EDX 006B58D8			
741BD94F		8A46 02		mov al,byte ptr ds:[esi+0x2]				EBX FFFFFFFF			
741BD952		83C6 03		add esi,0x3				ESP 0019F04C			
741BD955		- FF2485 94ED1B74		jmp dword ptr ds:[eax*4+0x741BED94]				EBP 0019F1D0			
741BD95C		0FBF06		movsx eax,word ptr ds:[esi]				ESI 0040E386 Cyb			
741BD95F		FF3428		push dword ptr ds:[eax+ebp]				EDI 000000A0			
741BD962		C70428 00000000		mov dword ptr ds:[eax+ebp],0x0				EIP 741BD94D MSV			
741BD969		33C0		xor eax,eax				C 0 ES 002B 32位			
741BD96B		8A46 02		mov al,byte ptr ds:[esi+0x2]				P 0 CS 0023 32位			
741BD96E		83C6 03		add esi,0x3				A 0 SS 002B 32位			
741BD971		- FF2485 94ED1B74		jmp dword ptr ds:[eax*4+0x741BED94]				Z 0 DS 002B 32位			
堆栈 ds:[0019F134]=006BA36C, (UNICODE "GuiShou")											
S 0 ES 0053 32位											
地址 HEX 数据 ASCII 地址 数值 注释											
0040E381		A0 00 03 00		6C 64 FF 4A		FD 69 34 FF		2F 64 FF 1A		?□.ld J齣4 /	
0040E391		68 FF FE 68		B4 FE 89 01		04 64 FF 21		0F 00 03 19		h 树逮?□d !□	
0040E3A1		68 FF 08 68		FF 0D A0 00		03 00 04 6C		FF 28 24 FF		h □h .?□.□	
0040E3B1		01 00 04 D4		FE FC 22 3E		64 FF 46 44		FF 04 04 FF		□.□軋?>d FD	
0040E3C1		0A 0A 00 10		00 04 04 FF		FD FE B0 FE		0B 0B 00 04		...□.□□ 录	
0019F04C		006BA36C								UNICODE "GuiShou"	
0019F050		0019F0A4									
0019F054		0019F0E4									
0019F058		00000000									
0019F05C		00000000									

接着将[eax+ebp]入栈，压栈的是刚刚输入的用户名[eax]的值是-98，这里其实是将[ebp-98]局部变量压入栈，然后eax清零，表示这条伪指令结束。

继续看下一条，伪代码解释的求长度

代码 - Crackmetrm
Form Load
0040E37D 0868FF FLDPr :[SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text
0040E380 0DA0000300 VCallHresult :Call ptr_0040342C
0040E385 6C64FF ILdRf :Push DWORD [LOCAL_009C]
0040E388 4A FnLenStr :vbaLenBstr
0040E389 FD6934FF CVarI4 :
0040E38D 2F64FF FFreeLStr :SysFreeString [LOCAL_009C]; [LOCAL_009C]=0
0040E390 1A68FF FFreeLAd :Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_0
0040E393 FE68B4FE8901 ForVar :
0040E399 0464FF FLdRfVar :Push LOCAL_009C
0040E39C 21 FLdPrThis :[SR]=[stack2]
0040E3A0 0868FF FLDPr :[SR]=[LOCAL_0098]

首先取出伪指令，然后直接跟进jmp

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BD947	0FBF06	movsx eax,word ptr ds:[esi]		EAX 0000004A
741BD94A	FF3428	push dword ptr ds:[eax+ebp]		ECX 02230000
741BD94D	33C0	xor eax,eax		EDX 006B58D8
741BD94F	8A46 02	mov al,byte ptr ds:[esi+0x2]		EBX FFFFFFFF
741BD952	83C6 03	add esi,0x3		ESP 0019F04C
741BD955	FF2485 94ED1B7	jmp dword ptr ds:[eax*4+0x741BED94]	MSVBVM50.741BEC63	EBP 0019F1D0
741BD95C	0FBF06	movsx eax,word ptr ds:[esi]		ESI 0040E386 CyberB
741BD95F	FF3428	push dword ptr ds:[eax+ebp]		EDI 000000A0
741BD962	C70428 00000000	mov dword ptr ds:[eax+ebp],0x0		EIP 741BD952 MSVBVM
741BD969	33C0	xor eax,eax		C 0 ES 002B 32位 0
741BD96B	8A46 02	mov al,byte ptr ds:[esi+0x2]		P 1 CS 0023 32位 0
741BD96E	83C6 03	add esi,0x3		A 0 SS 002B 32位 0
741BD971	FF2485 94ED1B7	jmp dword ptr ds:[eax*4+0x741BED94]	MSVBVM50.741BEC63	Z 1 DS 002B 32位 0

× (F) 主窗 (V) 窗口									
--	--	--	--	--	--	--	--	--	--

这里调用vbaLenBster，参数是之前输入的用户名，接着将用户名长度入栈后，eax清零，接着看下一条FD6934EF

代码 - Crackmefrm			
Form	Load		
rm			
:0040E37D 0868FF	FLdPr	: [SR]=[LOCAL_0098]	
*****Reference To: [propget]TextBox.Text			
:0040E380 0DA0000300	VCallHresult	:Call ptr_0040342C	
:0040E385 6C64FF	ILdRf	:Push DWORD [LOCAL_009C]	
:0040E388 4A	FnLenStr	:vbaLenBstr	
:0040E389 FD6934FF	CVarI4		
:0040E38D 2F64FF	FFreeStr	:SysFreeString [LOCAL_009C]; [LOCAL_009C]=0	
:0040E390 1A68FF	FFreeAd	:Push [LOCAL_0098]; Call [[[LOCAL_0098]]+8]; [[LOCAL_0098]]=0	
:0040E393 FE68B4FE8901	ForVar		
:0040E399 0464FF	FLdRfVar	:Push LOCAL_009C	
:0040E39C 21	FLdPrThis	[SR]=[stack2]	
:0040E39D 0F0003	VCallAd	:Return the control index 02	
:0040E3A0 1968FF	FStAdFunc		
:0040E3A3 0868FF	FLdPr	: [SR]=[LOCAL_0098]	
*****Reference To: [propget]TextBox.Text			
:0040E3A6 0DA0000300	VCallHresult	:Call ptr_0040342C	
:0040E3AB 046CFF	FLdRfVar	:Push LOCAL_0094	
:0040E3AE 2824FF0100	LiTVarI2	:PushVarInteger 0001	

这条伪指令后面并没有给出解释，但是没有关系，我们可以根据伪指令的执行过程猜测指令含义，这里其实是一个双操作码的指令，

CyberBlade.2.exe - [LCG - 主线程, 模块 - MSVBVM50]				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BEC63	E8 18D1F0FF	call MSVBVM50. __vbaLenBstr		EAX 000000FD
741BEC68	50	push eax		ECX 2230000
741BEC69	33C0	xor eax, eax		EDX 006B58D8
741BEC6B	8A06	mov al, byte ptr ds:[esi]		EBX FFFFFFFF
741BEC6D	46	inc esi	CyberBla. 0040E38A	ESP 0019F04C
741BEC6E	FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]	MSVBVM50. 741BEC9F	EBP 0019F1D0
741BEC75	E8 1AD1F0FF	call MSVBVM50. __vbaInStr		ESI 0040E38A CyberBla
741BEC7A	50	push eax		EDI 000000A0
741BEC7B	33C0	xor eax, eax		EIP 741BEC6E MSVBVM50
741BEC7D	8A06	mov al, byte ptr ds:[esi]		C 0 ES 002B 32位 0 (F
741BEC7F	46	inc esi	CyberBla. 0040E38A	P 0 CS 0023 32位 0 (F
741BEC80	FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]	MSVBVM50. 741BEC9F	A 0 SS 002B 32位 0 (F
741BEC87	33C0	xor eax, eax		Z 0 DS 002B 32位 0 (F
ds:[741BF188]=741BEC9F (MSVBVM50.741BEC9F)				S 0 ES 0053 32位 220

首先取出操作码FD，直接跟进jmp，什么都没有做，直接将eax清零。之后再次取出操作码69，继续跟进jmp



言爱破解 - CyberBlade.2.exe - [LCG - 主线程, 模块 - MSVBVM50]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BEC9F	33C0	xor eax, eax		EAX 00000069
741BECA1	8A06	mov al, byte ptr ds:[esi]		ECX 02230000
741BECA3	46	inc esi	CyberBla. 0040E38A	EDX 006B58D8
741BECA4	FF2485 94F91B7	jmp dword ptr ds:[eax*4+0x741BF994]	MSVBVM50. 741BD7D7	EBX FFFFFFFF
741BECAB	33C0	xor eax, eax		ESP 0019F04C
741BECAD	8A06	mov al, byte ptr ds:[esi]		EBP 0019F1D0

这里将bx赋值为0x3，然后跳转，继续跟进

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BD7D7	66:BB 0300	mov bx, 0x3		EAX 00000069
741BD7DB	EB D8	jmp short MSVBVM50. 741BD7B5		ECX 02230000
741BD7DD	0FBF06	movsx eax, word ptr ds:[esi]		EDX 006B58D8
741BD7E0	03C5	add eax, ebx		EBX FFFF0003
741BD7E2	D958 08	fstp dword ptr ds:[eax+0x8]		ESP 0019F04C
741BD7E5	66:C700 0400	mov word ptr ds:[eax], 0x4		EBP 0019F1D0
741BD7E7	50	push eax		ESI 0040E38B CyberBla

接着取出参数FF34，FF34也是个负数，然后再将FF34加上ebp，代表这是一个局部变量

言爱破解 - CyberBlade.2.exe - [LCG - 主线程, 模块 - MSVBVM50]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BD7B5	59	pop ecx	0019F0A4	EAX FFFFFFF34
741BD7B6	0FBF06	movsx eax, word ptr ds:[esi]		ECX 00000007
741BD7B9	03C5	add eax, ebp		EDX 006B58D8
741BD7BB	8948 08	mov dword ptr ds:[eax+0x8], ecx		EBX FFFF0003
741BD7BE	66:8918	mov word ptr ds:[eax], bx		ESP 0019F050
741BD7C1	50	push eax		EBP 0019F1D0
741BD7C2	33C0	xor eax, eax		ESI 0040E38B CyberBla
741BD7C4	8A46 02	mov al, byte ptr ds:[esi+0x2]		EDI 000000A0

接下来将ecx赋值给[eax+0x8]的地址处，我们数据窗口跟随eax，然后将bx赋值给eax，赋值完成后eax值如下：

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
741BD7B5	59	pop ecx	0019F104	EAX 0019F104
741BD7B6	0FBF06	movsx eax, word ptr ds:[esi]		ECX 00000007
741BD7B9	03C5	add eax, ebp		EDX 006B58D8
741BD7BB	8948 08	mov dword ptr ds:[eax+0x8], ecx		EBX FFFF0003
741BD7BE	66:8918	mov word ptr ds:[eax], bx		ESP 0019F04C
741BD7C1	50	push eax		EBP 0019F1D0
741BD7C2	33C0	xor eax, eax		ESI 0040E38B CyberBla. 0040E3
741BD7C4	8A46 02	mov al, byte ptr ds:[esi+0x2]		EDI 000000A0
741BD7C7	83C6 03	add esi, 0x3		EIP 741BD7C2 MSVBVM50. 741BD7
741BD7CA	FF2485 94ED1B7	jmp dword ptr ds:[eax*4+0x741BED94]		C 1 ES 002B 32位 0(FFFFFFFF)
741BD7D1	66:BB 0800	mov bx, 0x8		P 0 CS 0023 32位 0(FFFFFFFF)
741BD7D5	EB DE	jmp short MSVBVM50. 741BD7B5		Z 0 DS 002B 32位 0(FFFFFFFF)
741BD7D7	66:BB 0300	mov bx, 0x3		S 0 ES 0053 32位 220000(FEE)

地址	HEX 数据	ASCII	地址	数值	注释
0019F104	03 00 00 00 00 00 00 00	07 00 00 00 00 00 00 00	0019F04C	0019F104	
0019F114	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	0019F050	0019F0A4	
0019F124	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	0019F054	0019F0E4	
0019F134	6C A3 6B 00 14 E5 23 02	00 00 00 00 00 00 00 00	0019F058	00000000	
0019F144	00 00 00 00 00 00 00 00	DC F1 19 00 B8 F2 19 00	0019F05C	00000000	
0019F154	CC E6 23 02 15 20 00 00	AC 93 DA 00 CA 3C 01 04	0019F060	00000000	

还记得VB的变量类型吗？前两个字节是变量类型，03代表是Long，中间是6个字节的保留位，首地址+8的位置才是真正的数值。

这个07是之前通过vbaLenBstr获取到的用户名长度，在这里转成了变量，并将变量首地址压栈。

Form	Load	
0040E37D 0868FF	FLdPr	: [SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text		
0040E380 0DA0000300	VCallHresult	: Call ptr_0040342C
0040E385 6C64FF	ILdRf	: Push DWORD [LOCAL_009C]
0040E388 4A	FnLenStr	: vbaLenBstr
0040E389 FD6934FF	CVarI4	:
0040E38D 2F64FF	FFree1Str	: SysFreeString [LOCAL_009C]; [LOCAL_009C]=0
0040E390 1A68FF	FFree1Ad	: Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_0098]]=0
0040E393 FE68B4FE8901	ForVar	:
0040E399 0464FF	FLdRfVar	: Push LOCAL_009C
0040E39C 21	FLdPrThis	: [SR]=[stack2]
0040E39D 0F0003	VCallAd	: Return the control index 02
0040E3A0 1968FF	FStAdFunc	:

现在我们就通过实际的跟踪结果来得出这条指令的含义了。就是将int值转成变量类型。由于整个跟踪过程实在是复杂，我这里只贴出算法的关键部分

0040E3A0 1968FF	FStAdFunc	:
0040E3A3 0868FF	FLdPr	: [SR]=[LOCAL_0098]
*****Reference To: [propget]TextBox.Text		
0040E3A6 0DA0000300	VCallHresult	: Call ptr_0040342C
0040E3AB 046CFF	FLdRfVar	: Push LOCAL_0094
0040E3AE 2824FF0108	LItrVarI2	: PushVarInteger 0001
0040E3B3 04D4FE	FLdRfVar	: Push LOCAL_012C
0040E3B6 FC22	CI4Var	: vbaI4Var
0040E3B8 3E64FF	FLdZeroAd	: Push DWORD [LOCAL_009C]; [LOCAL_009C]=0
0040E3BB 4644FF	CVarStr	:
0040E3BE 0404FF	FLdRfVar	: Push LOCAL_00FC
*****Reference To: msvbvm50.rtcMidCharVar		
0040E3C1 0DA0001000	ImpAdCallFPR4	: Call ptr_00401006; check stack 0010; Push EAX
0040E3C6 0404FF	FLdRfVar	: Push LOCAL_00FC
0040E3C9 FDFE0FE	CStrVarVal	:
*****Reference To: msvbvm50.rtcAnsiValueBstr		
0040E3CD 0B0B000400	ImpAdCallI2	: Call ptr_0040100C; check stack 0004; Push EAX
0040E3D2 4434FF	CVarI2	:
0040E3D5 FBFE4FE	ConcatVar	:
0040E3D9 FCF6CFF	FStVar	:
0040E3DD 2FB0FE	FFree1Str	: SysFreeString [LOCAL_0150]; [LOCAL_0150]=0
0040E3E0 1A68FF	FFree1Ad	: Push [LOCAL_0098]; Call [[LOCAL_0098]]+8; [[LOCAL_0098]]=0
0040E3E3 36060044FF24FF04	FFreeVar	: Free 0006/2 variants
0040E3EC 04D4FE	FLdRfVar	: Push LOCAL_012C

CyberBlade.2.exe - [LCG - m主线程, 模块 - MSVBVM50]		
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+]		
暂停 停止 单步 断点 窗口 地址 数据 反汇编 注释 寄存器 (FPU)		
地址	HEX 数据	反汇编
741BE7A6	0FB70E	movzx ecx, word ptr ds:[esi]
741BE7A9	8B55 AC	mov edx, dword ptr ss:[ebp-0x54]
741BE7AC	8B048A	mov eax, dword ptr ds:[edx+ecx*4]
741BE7AF	0BC0	or eax, eax
741BE7B1	0F84 78790000	je MSVBVM50.741C612F
741BE7B7	0FB77E 02	movzx edi, word ptr ds:[esi+0x2]
741BE7BB	83C6 04	add esi, 0x4
741BE7BE	03FC	add edi, esp
741BE7C0	FFD0	call eax
741BE7C2	3BFC	cmp edi, esp
741BE7C4	0F85 5B790000	jnz MSVBVM50.741C6125
741BE7CA	33C0	xor eax, eax
741BE7CC	8A06	mov al, byte ptr ds:[esi]
eax=00401006 (<jmp.&MSVBVM50.rtcMidCharVar_632>)		
地址	数值	注释
0019F114	00000008	
0019F118	00000000	
0019F11C	00557F34	UNICODE "GuiShou"
0019F120	00000000	
0019F124	00000000	
0019F128	00000000	
地址	数值	注释
0019F044	0019F0D4	
0019F048	0019F114	
0019F04C	00000001	
0019F050	0019F0F4	
0019F054	0019F13C	
0019F058	00000000	

在40E3C1处截取用户名的第一个字符串

地址			HEX 数据	反汇编	注释	寄存器 (FPU)
741BE7A6	0FB70E			movzx ecx,word ptr ds:[esi]		EAX 0019F0D4
741BE7A9	8B55 AC			mov edx,dword ptr ss:[ebp-0x54]	CyberBla. 00402AEC	ECX 00000000
741BE7AC	8B048A			mov eax,dword ptr ds:[edx+ecx*4]		EDX 00000002
741BE7AF	0BC0			or eax,eax		EBX FFFF0008
741BE7B1	0F84 78790000			je MSVBVM50.741C612F		ESP 0019F054
741BE7B7	0FB77E 02			movzx edi,word ptr ds:[esi+0x2]		EBP 0019F1D0
741BE7BB	83C6 04			add esi,0x4		ESI 0040E3C6 CyberBla. 004
741BE7BE	03FC			add edi,esp		EDI 0019F054
741BE7C0	FFD0			call eax		EIP 741BE7C2 MSVBVM50.741
741BE7C2	3BFC			cmp edi,esp		C 1 ES 002B 32位 0 (FFFFI
741BE7C4	0F85 5B790000			jnz MSVBVM50.741C6125		P 0 CS 0023 32位 0 (FFFFI
741BE7CA	33C0			xor eax,eax		A 1 SS 002B 32位 0 (FFFFI
741BE7CC	8A06			mov al,byte ptr ds:[esi]		Z 0 DS 002B 32位 0 (FFFFI
esp=0019F054 edi=0019F054						S 1 ES 0053 32位 338000
地址	数值	注释	地址	数值	注释	
0019F0D4	00000008		0019F054	0019F13C		
0019F0D8	0040E3A7	CyberBla. 0040E3A7	0019F058	00000000		
0019F0DC	00590BFC	UNICODE "G"	0019F05C	00000000		
0019F0E0	0019F1D0		0019F060	00000000		
0019F0E4	00000002		0019F064	00000000		
0019F0E8	00000000		0019F068	00000000		

接着在40E3CD处将截取的用户名每一位转成ASCII值

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FFreeVar:Free 0006/2 variants  
0040E3EC04D4FEFLDRfVar:Push LOCAL\_012C  
0040E3EFFE7EB4FE2D01NextStepVar:  
0040E3F5046CFFFLDRfVar:Push LOCAL\_0094  
0040E3F8FBEB44FFFnLenVar:vbaLenVar  
0040E3FC2854FF0900LitVarI2:PushVarInteger 0009  
0040E4015DHardType:  
0040E402FB74GtVarBool:Push (Pop1 >= Pop2)  
0040E4041CB901BranchF:If Pop=0 then ESI=0040E425  
0040E407046CFFFLDRfVar:Push LOCAL\_0094  
0040E40AFEC454FF50455254LitVarR8:  
0040E416FBBC44FFDivVar:  
0040E41AFBE124FFFnFixVar:  
0040E41EFCF66CFFFSrVar:

0040E3BE0404FFFLDRfVar:Push LOCAL\_00FC  
\*\*\*\*\*Reference To:msvbvm50.rtcMidCharVar  
0040E3C10A0A001000ImpAdCallFPR4:Call ptr\_00401006; check stack 0010; Push EAX  
0040E3C60404FFFLDRfVar:Push LOCAL\_00FC  
0040E3C9FDFF0FECSrVarVal:  
\*\*\*\*\*Reference To:msvbvm50.rtcAnsiValueBstr  
0040E3CD0B0B000400ImpAdCallI2:Call ptr\_0040100C; check stack 0004; Push EAX  
0040E3D24434FFCVarI2:  
0040E3D5FBFE4FEConcatVar:  
0040E3D9FCF66CFFFSrVar:  
0040E3DD2FB0FEFFree1Str:SysFreeString [LOCAL\_0150]; [LOCAL\_0150]=0  
0040E3E01A68FFFFree1Ad:Push [LOCAL\_0098]; Call [[LOCAL\_0098]]+8; [[LOCAL\_0098]]=0  
0040E3E336060044FF24FF04FF



地址	HEX 数据	反汇编	注释
741BEC54	33C0	xor eax, eax	
741BEC56	8A46 02	mov al, byte ptr ds:[esi+0x2]	
741BEC59	83C6 03	add esi, 0x3	
741BEC5C	- FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]	
741BEC63	E8 18D1F0FF	call MSVBVM50. __vbaLenBstr	
741BEC68	50	push eax	
741BEC69	33C0	xor eax, eax	
741BEC6B	8A06	mov al, byte ptr ds:[esi]	
741BEC6D	46	inc esi	CyberBla. 0040E3D7
741BEC6E	- FF2485 94ED1B74	jmp dword ptr ds:[eax*4+0x741BED94]	
741BEC75	E8 1AD1F0FF	call MSVBVM50. __vbaInStr	
741BEC7A	50	push eax	
741BEC7B	33C0	xor eax, eax	
eax=0019F0B4			
地址	数值	注释	地址
0019F0B4	00000008		0019F054
0019F0B8	00000000		0019F058
0019F0BC	00533CDC	UNICODE "7111710583104111117"	0019F05C
0019F0C0	00000000		0019F060
0019F0C4	00000000		0019F064
0019F0C8	00000000		0019F068

即拼接用户名的ASCII十进制字符串，第二部分的算法就完成

```

loc_40E393: jmp 41
loc_40E393: For var_14C = 1 To CVar(Len(Me.txtname.Text)): var_12C = var_14C 'Variant
loc_40E3C1:   var_FC = Mid(CVar(Me.txtname.Text), CLng(var_12C), 1)
loc_40E3D5:   var_11C = var_94 & CVar(Asc(CStr(var_FC)))
loc_40E3D9:   var_94 = var_11C 'Variant
loc_40E3EF: Next var_14C 'Variant

```

在VB伪代码中,var\_94就是最后拼接的结果

## 第三部分 除以圆周率

接下来是第三部分，直接来看VB Decompiler中的伪代码

```

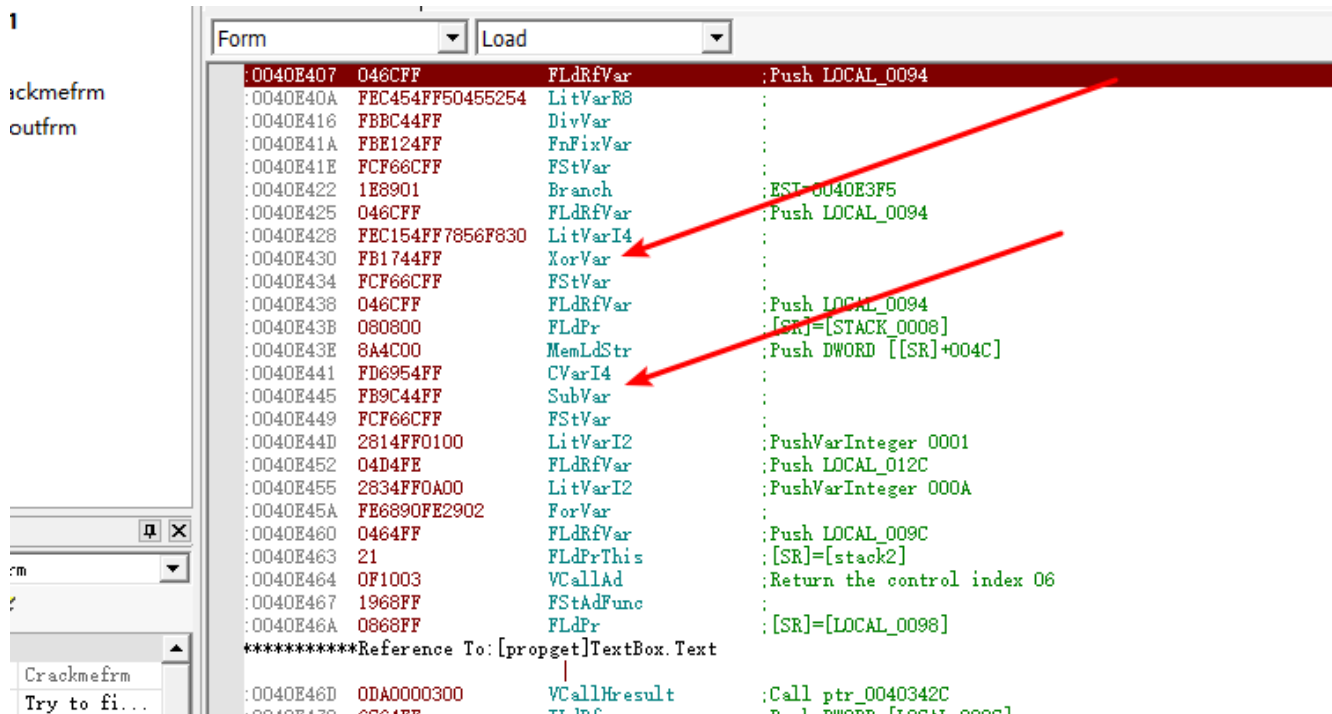
loc_40E3F5: ' Referenced from: 40E422
loc_40E404: If (Len(var_94) > 9) Then
loc_40E41E:   var_94 = Fix((var_94 / 3.141592654)) 'Variant
loc_40E422:   GoTo loc_40E3F5
loc_40E425: End If
loc_40E449: var_94 = (var_94 Xor &H30F85678 - CVar(global_76)) 'Variant

```

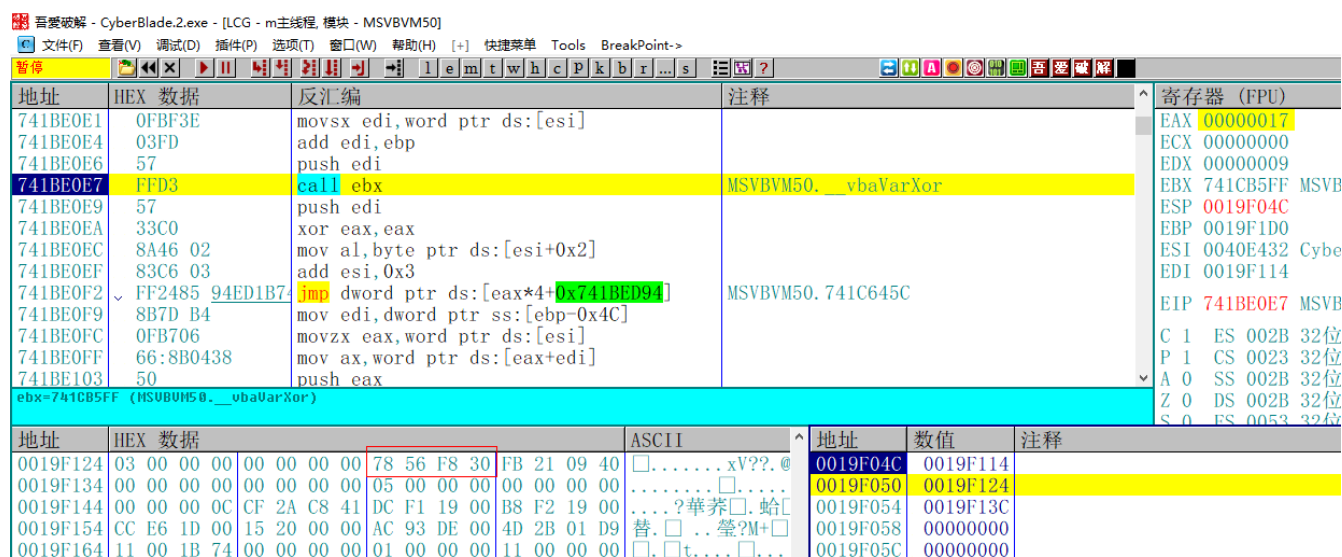
这一部分的逻辑也很清晰，如果用户名拼接的字符串长度大于9的话，就将这个结果转为浮点数除以圆周率，一直除到结果的长度小于9。这个部分我也用OD详细跟过每一条伪指令，确实和静态反汇编的逻辑是一样的

但是在loc\_40E449的位置，将var\_94和一个值进行了异或，并且还减去了另外一个值。这两个数值我们无从得知，只能跟踪OD





根据伪指令的助记符XorVar和SubVar快速定位到这两个地址，下内存访问断点，很快就能找到这两个值



可以看到这里实际上是将0x30F85678和用户名的结果进行异或

文件(F)查看(V)调试(D)插件(P)选项(O)窗口(W)帮助(H)快捷菜单ToolsBreakPoint->

暂停

然后减去0xD8B3，找到了这两个数，第三部分也就结束了

## 第四部分 干扰代码

```

loc_40E45A: For var_170 = 1 To 10: var_12C = var_170 'Variant
loc_40E489: If (Me.txtkey.Text = global_52(CLng(var_12C))) Then
loc_40E48C: End If
loc_40E48F: Next var_170 'Variant

```

这个是最有意思的，你会发现代码初始化了10次循环，循环将一个变量和Key值进行比较，但问题在于Then分支没有任何代码。这也就是说不管这个循环中的比较成立与否 都对我们没有任何影响

## 第五部分 关键比较

```

loc_40E4DE: If ((CVar(Me.txtkey.Text) - var_94) = CVar(Len(Me.txtname.Text))) Then
loc_40E502: MsgBox("Wow, you have found a correct key!", &H40, "Correct key", var_FC, var_11C)
loc_40E533: MsgBox("Mail me, how you got it: CyberBlade@gmx.net ", &H40, "Correct key!", var_FC, var_11C)
loc_40E550: Me.Command2.Caption = "Exit"
loc_40E55B: Else
loc_40E567: global_80 = (global_80 + 1)
loc_40E570: var_176 = global_80
loc_40E579: If (var_176 = 6) Then
loc_40E5B3: If (MsgBox("==Do you need a hint ?==", &H24, "I can't stand it anymore", var_FC, var_11C) = 7) Then
loc_40E5B6: Exit Sub
loc_40E5BA: Else
loc_40E5DB: MsgBox("Forget it.", &H40, "he, he...", var_FC, var_11C)
loc_40E5F0: global_80 = 0
loc_40E5F3: End If
loc_40E5F6: Else
loc_40E5FC: If (var_176 > 3) Then
loc_40E620: MsgBox("Have you ever been trying to be successful in cracking my password ?", &H20, "Failed", var_FC, var_11C)
loc_40E633: Else
loc_40E639: If (var_176 <= 3) Then
loc_40E65D: MsgBox("Sorry, wrong key.", &H40, "Failed", var_FC, var_11C)

```

最后一部分，比较序列号减去var\_94是否等于用户名长度，也就是说用户名计算的结果再加上用户名的长度就是真正的序列号。最后对这个程序的校验过程做一个总结

### 总结:

1. 检测用户名和序列号是否为空，序列号长度是否小于5
2. 将用户名各个字符的ASCII码的十进制转换成字符串连接起来得到一个数result
3. 判断result长度是否大于9 如果大于则将result转为浮点数之后除以圆周率 直到result的长度小于或等于9



4. 将result和0x30F85678进行异或并减去0xD8B3，再加上用户名长度就是正确的序列号

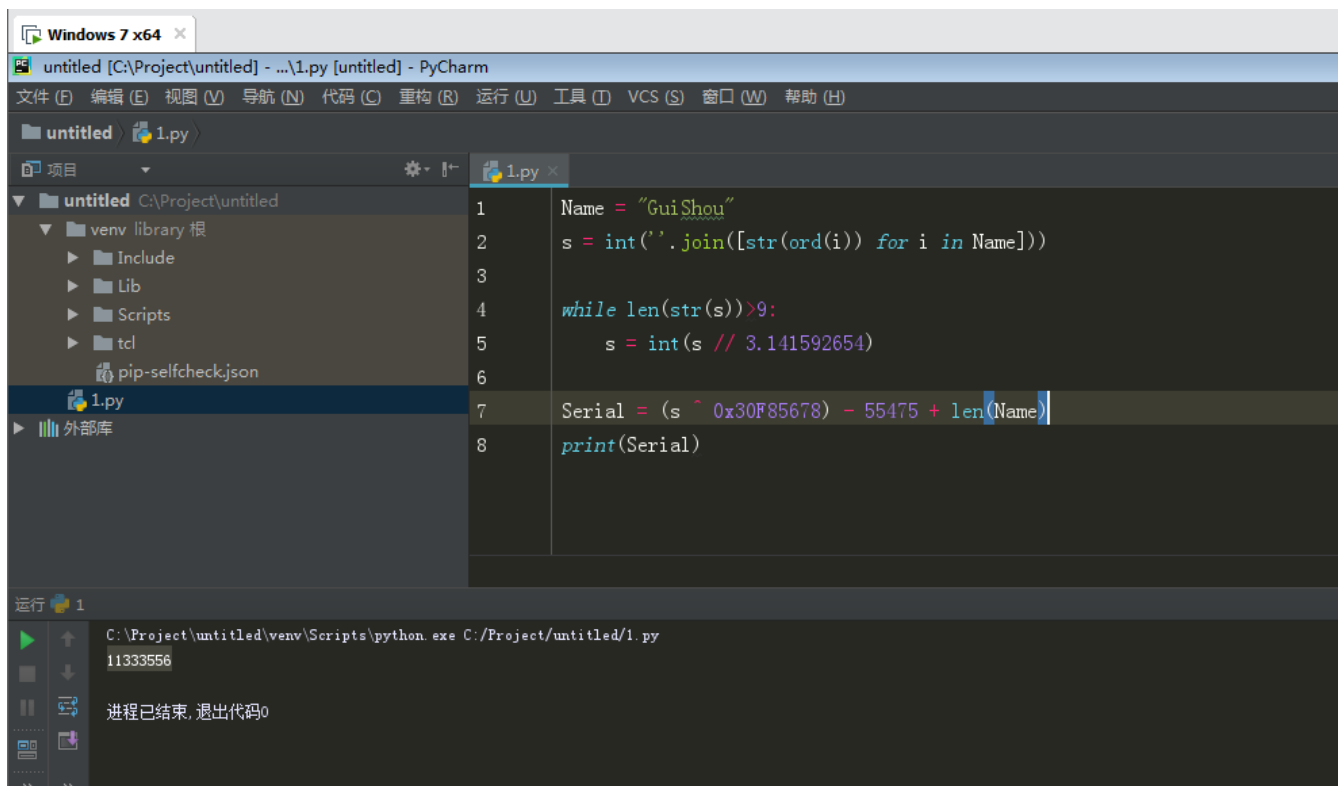
## 写出注册机

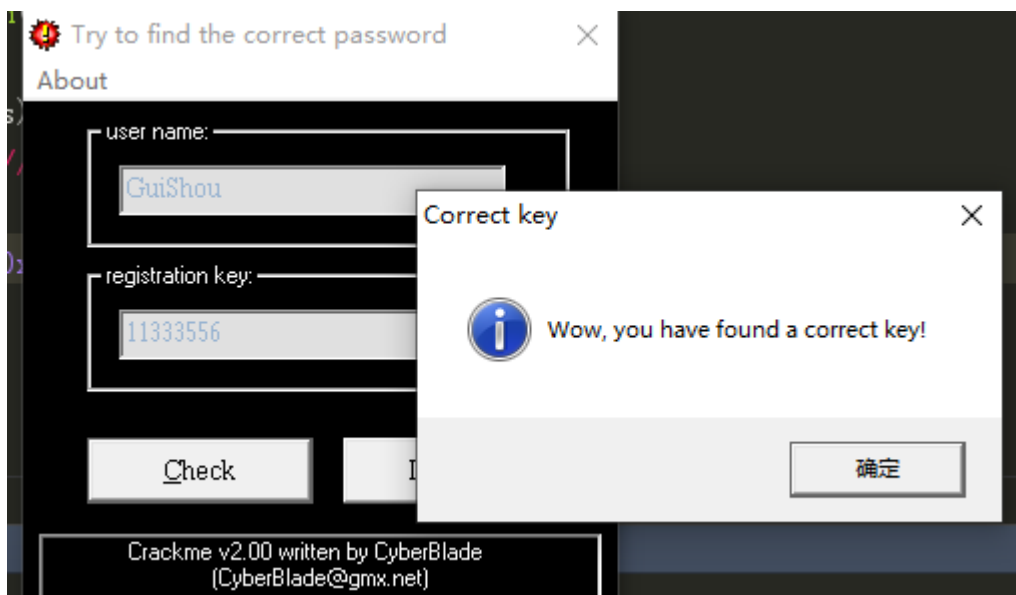
接着我们根据已经分析的算法写出这个程序的注册机，这个注册机用C++写还是太费劲了 直接用python快

```
Name = "GuiShou"
s = int(''.join([str(ord(i)) for i in Name]))

while len(str(s))>9:
    s = int(s // 3.141592654)

serial = (s ^ 0x30F85678) - 55475 + len(Name)
print(serial)
```





## 总结

P-Code的程序并非如传言一样不可战胜，只要你有足够的耐心，配合VB Decompiler+VBExplorer+OD的黄金组合，剩下的就是纯体力活了。

P-Code类的程序用OD跟踪伪指令虽然能看到每一处实现细节，但是毕竟还是太费力了。这个时候如果能总结出一套相对比较完整的P-Code的伪指令及每个参数的具体含义再配合WKTVBDebugger，调试P-Code就显得游刃有余了。如果有大佬总结出来了还请发我一份 哈哈。

## 附上分析过程

最后附上分析过程和相关文件

:0040E380	0DA0000300	vCallHresult	;Call ptr_0040342C
获取输入的用户名			
:0040E385	6C64FF	ILdrf	;Push DWORD [LOCAL_009C]
将用户名压入堆栈			
:0040E388	4A	FnLenStr	;vbaLenBstr
求用户名长度			
:0040E389	FD6934FF	CVarI4	;
将用户名长度转为变量			
:0040E38D	2F64FF	FFree1Str	;SysFreeString
[LOCAL_009C]; [LOCAL_009C]=0 释放用户名内存			
:0040E390	1A68FF	FFree1Ad	;Push [LOCAL_0098]; Call
[[[LOCAL_0098]]+8]; [[LOCAL_0098]]=0 释放局部变量			
:0040E393	FE68B4FE8901	ForVar	;
初始化循环次数 开始循环			
:0040E399	0464FF	FLdrfVar	;Push LOCAL_009C
将局部变量0压入堆栈			

```

:0040E39C  21          FLdPrThis      ;[SR]=[stack2]
:0040E39D  0F0003      VCallAd        ;Return the control index
02
:0040E3A0  1968FF      FStAdFunc      ;
:0040E3A3  0868FF      FLdPr          ;[SR]=[LOCAL_0098]
*****Reference To:[propget]TextBox.Text
|
:0040E3A6  0DA0000300  VCallHresult   ;Call ptr_0040342C
  获取输入的用户名
:0040E3AB  046CFF      FLdRfVar       ;Push LOCAL_0094
  将局部变量0压入堆栈
:0040E3AE  2824FF0100  LitVarI2       ;PushVarInteger 0001
  将局部变量1压入堆栈
:0040E3B3  04D4FE      FLdRfVar       ;Push LOCAL_012C
  将局部变量0压入堆栈
:0040E3B6  FC22       CI4Var        ;vbaI4Var
  将变量1转为数字
:0040E3B8  3E64FF      FLdZeroAd      ;Push DWORD [LOCAL_009C];
[LOCAL_009C]=0  将用户名压入堆栈
:0040E3BB  4644FF      CVarStr        ;
  将用户名字符串转为变量
:0040E3BE  0404FF      FLdRfVar       ;Push LOCAL_00FC
  将局部变量0压入堆栈
*****Reference To->msvbvm50.rtcMidCharVar
|
:0040E3C1  0A0A001000  ImpAdCallFPR4  ;Call ptr_00401006; check
stack 0010; Push EAX  截取用户名的第一个字符
:0040E3C6  0404FF      FLdRfVar       ;Push LOCAL_00FC
  将用户名的第一个字符压入堆栈
:0040E3C9  FDFEB0FE   CStrVarVal     ;
  将用户名的第一个字符从变量转为字符串
*****Reference To->msvbvm50.rtcAnsiValueBstr
|
:0040E3CD  0B0B000400  ImpAdCallI2    ;Call ptr_0040100C; check
stack 0004; Push EAX  将用户名的第一个字符转为ASCII值
:0040E3D2  4434FF      CVarI2         ;          将用户名
的第一个字符的ASCII值转为变量
:0040E3D5  FBFE4FE     ConcatVar      ;          将ASCII
值的十进制进行字符串拼接
:0040E3D9  FCF66CFF   FStVar        ;          将拼接的
字符串转为变量
:0040E3DD  2FB0FE     FFree1Str      ;SysFreeString
[LOCAL_0150]; [LOCAL_0150]=0  释放字符串
:0040E3E0  1A68FF      FFree1Ad       ;Push [LOCAL_0098]; call
[[[LOCAL_0098]]+8]; [[LOCAL_0098]]=0

```

:0040E3E3	36060044FF24FF04	FFreeVar	;Free 0006/2 variants
释放变量			
:0040E3EC	04D4FE	FLdRfVar	;Push LOCAL_012C
:0040E3EF	FE7EB4FE2D01	NextStepVar	;
开始下一轮循环			
:0040E3F5	046CFF	FLdRfVar	;Push LOCAL_0094
将用户名拼接的字符串压入堆栈——7111710583104111117(0x13)			
:0040E3F8	FBEB44FF	FnLenVar	;vbaLenVar
求用户名拼接的字符串长度			
:0040E3FC	2854FF0900	LitVarI2	;PushVarInteger 0009
将整形变量9压入堆栈			
:0040E401	5D	HardType	;
修改变量9的类型			
:0040E402	FB74	GtVarBool	;Push (Pop1 >= Pop2)
比较长度是否大于9			
:0040E404	1CB901	BranchF	;If Pop=0 then ESI=0040E425
如果不大于9则跳转到0040E425			
:0040E407	046CFF	FLdRfVar	;Push LOCAL_0094
将用户名拼接的字符串压入堆栈			
:0040E40A	FEC454FF50455254	LitVarR8	;
将参数一(圆周率)的类型修改为浮点数			
:0040E416	FBBC44FF	DivVar	;
将用户名拼接的字符串除以圆周率			
:0040E41A	FBE124FF	FnFixVar	;
相当于字符串拷贝			
:0040E41E	FCF66CFF	FStVar	;
将用户名拼接的字符串转为浮点数			
:0040E422	1E8901	Branch	;ESI=0040E3F5
如果长度大于9则跳转至0040E3F5			
:0040E425	046CFF	FLdRfVar	;Push LOCAL_0094
将浮点数结果压入堆栈			
:0040E428	FEC154FF7856F830	LitVarI4	;
将浮点数转为整形			
:0040E430	FB1744FF	XorVar	;
将结果和30F85678进行异或			
:0040E434	FCF66CFF	FStVar	;
保存结果			
:0040E438	046CFF	FLdRfVar	;Push LOCAL_0094
将异或后的结果压栈			
:0040E43B	080800	FLdPr	;[SR]=[STACK_0008]
:0040E43E	8A4C00	MemLdStr	;Push DWORD [[SR]+004C]
将0xDBD3压栈			
:0040E441	FD6954FF	CVarI4	;
将0xDBD3转为变量			
:0040E445	FB9C44FF	SubVar	;
用异或后的结果减去0xDBD3			

:0040E449 FCF66CFF FStVar ;  
保存结果

需要分析记录和相关文件可以到我的Github下载: <https://github.com/TonyChen56/160-Crackme>