Reversing VB Crackme 3.0 Eternal Bliss
by Rhytm [Dread]

(This short tutorial was written in Wordpad)

Collecting Information

Hello all !! Well I guess you're reading this to learn how to reverse Eternal Bliss' third Vb crackme. Let's start with collecting some info. First read the .txt file:

Find the correct hardcoded code in this CrackMe.

What you won't see:
1) The correct code using Hexeditor
2) The correct code in Softice
3) The correct code in SmartCheck

ok, nice to know, now get ready for the real work :)
Load the target in Smartcheck, enter a dummy serial, notice that the register button will be enabled after typing a minumum of six characters.
That's why you'll see the Text1_Change function in the smartcheck code.
Press the register button, and look at the Smartcheck code again :P
Maybe it's nice to explain the Mid function in detail, you don't need much info about this function in this crackme, but it can come out handy:

```
Dim MyString
MyString = "The dog jumps"    ' Initialize string.
Mid(MyString, 5, 3) = "fox"     ' MyString = "The fox jumps".
Mid(MyString, 5) = "cow"     ' MyString = "The cow jumps".
Mid(MyString, 5) = "cow jumped over"     ' MyString = "The cow jumpe".
Mid(MyString, 5, 3) = "duck"     ' MyString = "The duc jumpe".
```

Did it help ?? I hope so :)
We see that every number of our serial is converted to the ascii value..
After this the program suddenly tells us we've entered the wrong serial :(

Analyzing the Info and Reversing the Target

Lets switch to advanced mode, by selecting "Show All Event".
What do see here ?? For every number you entered you see this row of functions:

Text1.text	; Get the text from the messagebox	
__vbal4Var	**;** String to long	
Mid	; Get the x-th number from your serial	
__vbaStrVarVal	**; ALWAYS HERE**	
Asc	; Get the ascii value of this number	
__vbaFreeStr	; Free the memory that was used by a string	**ALWAYS HERE**
__vbaFreeObj	; Free some more memory	**ALWAYS HERE**

SysFreeString ; ALWAYS HERE
SysFreeString ; ALWAYS HERE
__vbaVarForNext ; ALWAYS HERE
Textbox::Addref ; ALWAYS HERE
__vbaObjSet ; ALWAYS HERE

Just load Eternal Bliss' 2nd crackme to see this structure :)
First in needed some more info, especially about the address where a check is done... So lets dissassemble our target in W32Dasm, do some stepping in SoftICE and isolate this loop:

```
:00402C47 8B4508                mov eax, dword ptr [ebp+08]
:00402C4A 8B00                          mov eax, dword ptr [eax]
:00402C4C FF7508                push [ebp+08]
:00402C4F FF9000030000          call dword ptr [eax+00000300]
:00402C55 50                            push eax
:00402C56 8D4594                lea eax, dword ptr [ebp-6C]
:00402C59 50                            push eax
:00402C5A E809E6FFFF            Call 00401268                   <-- vbaObjSet
:00402C5F 8985C8FEFFFF          mov dword ptr [ebp+FFFFFEC8], eax
:00402C65 8D45AC                lea eax, dword ptr [ebp-54]
:00402C68 50                            push eax
:00402C69 8B85C8FEFFFF          mov eax, dword ptr [ebp+FFFFFEC8]
:00402C6F 8B00                          mov eax, dword ptr [eax]
:00402C71 FFB5C8FEFFFF          push dword ptr [ebp+FFFFFEC8]
:00402C77 FF90A0000000          call dword ptr [eax+000000A0]
:00402C7D DBE2                  fclex
:00402C7F 8985C4FEFFFF          mov dword ptr [ebp+FFFFFEC4], eax
:00402C85 83BDC4FEFFFF00        cmp dword ptr [ebp+FFFFFEC4], 00000000
:00402C8C 7D23                  jge 00402CB1
:00402C8E 68A0000000            push 000000A0
:00402C93 6898234000            push 00402398
:00402C98 FFB5C8FEFFFF          push dword ptr [ebp+FFFFFEC8]
:00402C9E FFB5C4FEFFFF          push dword ptr [ebp+FFFFFEC4]
:00402CA4 E8B9E5FFFF            Call 00401262                   <-- vbaHresultCheckObj
:00402CA9 898564FEFFFF          mov dword ptr [ebp+FFFFFE64], eax
:00402CAF EB07                  jmp 00402CB8
:00402CB1 83A564FEFFFF00        and dword ptr [ebp+FFFFFE64], 00000000
:00402CB8 C78574FFFFFF01000000   mov dword ptr [ebp+FFFFFF74], 00000001
:00402CC2 C7856CFFFFFF02000000  mov dword ptr [ebp+FFFFFF6C], 00000002
:00402CCC 8B45AC                mov eax, dword ptr [ebp-54]
:00402CCF 898578FEFFFF          mov dword ptr [ebp+FFFFFE78], eax
:00402CD5 8365AC00              and dword ptr [ebp-54], 00000000
:00402CD9 8B8578FEFFFF          mov eax, dword ptr [ebp+FFFFFE78]
:00402CDF 894584                mov dword ptr [ebp-7C], eax
:00402CE2 C7857CFFFFFF08000000  mov dword ptr [ebp+FFFFFF7C], 00000008
:00402CEC 8D856CFFFFFF          lea eax, dword ptr [ebp+FFFFFF6C]
:00402CF2 50                            push eax
:00402CF3 8D45D0                lea eax, dword ptr [ebp-30]
:00402CF6 50                            push eax
:00402CF7 E842E5FFFF            Call 0040123E                   <-- vbaI4Var
:00402CFC 50                            push eax
```

```
:00402CFD 8D857CFFFFFF          lea eax, dword ptr [ebp+FFFFFF7C]
:00402D03 50                            push eax
:00402D04 8D855CFFFFFF          lea eax, dword ptr [ebp+FFFFFF5C]
:00402D0A 50                            push eax
:00402D0B E834E5FFFF            Call 00401244                     <-- rtcMidCharVar
:00402D10 8D855CFFFFFF          lea eax, dword ptr [ebp+FFFFFF5C]
:00402D16 50                            push eax
:00402D17 8D45A8                lea eax, dword ptr [ebp-58]
:00402D1A 50                            push eax
:00402D1B E82AE5FFFF            Call 0040124A                     <-- vbaStrVarVal
:00402D20 50                            push eax
:00402D21 E82AE5FFFF            Call 00401250                     <-- rtcAnsiValueBstr
:00402D26 0FBFC0                movsx eax, ax
:00402D29 8B4DC0                mov ecx, dword ptr [ebp-40]
:00402D2C 03C8                  add ecx, eax
:00402D2E 0F80F90A0000          jo 0040382D
:00402D34 894DC0                mov dword ptr [ebp-40], ecx
:00402D37 8D4DA8                lea ecx, dword ptr [ebp-58]
:00402D3A E81DE5FFFF            Call 0040125C                     <-- vbaFreeStr
:00402D3F 8D4D94                lea ecx, dword ptr [ebp-6C]
:00402D42 E80FE5FFFF            Call 00401256                     <-- vbaFreeObj
:00402D47 8D855CFFFFFF          lea eax, dword ptr [ebp+FFFFFF5C]
:00402D4D 50                            push eax
:00402D4E 8D856CFFFFFF          lea eax, dword ptr [ebp+FFFFFF6C]
:00402D54 50                            push eax
:00402D55 8D857CFFFFFF          lea eax, dword ptr [ebp+FFFFFF7C]
:00402D5B 50                            push eax
:00402D5C 6A03                  push 00000003
:00402D5E E8D5E4FFFF            Call 00401238                     <-- vbaFreeVarList
:00402D63 83C410                add esp, 00000010
:00402D66 8D8590FEFFFF          lea eax, dword ptr [ebp+FFFFFE90]
:00402D6C 50                            push eax
:00402D6D 8D85A0FEFFFF          lea eax, dword ptr [ebp+FFFFFEA0]
:00402D73 50                            push eax
:00402D74 8D45D0                lea eax, dword ptr [ebp-30]
:00402D77 50                            push eax
:00402D78 E8B5E4FFFF            Call 00401232                     <-- vbaVarForNext
:00402D7D 89857CFEFFFF          mov dword ptr [ebp+FFFFFE7C], eax
:00402D83 83BD7CFEFFFF00        cmp dword ptr [ebp+FFFFFE7C], 00000000
:00402D8A 0F85B7FEFFFF          jne 00402C47
```

So this is the whole loop we see in smartcheck, just look at the structure of the VB calls..
The jump to 402C47 won't be taken anymore when every number has walked through the loop..
After that we see this :)

```
:00402D8A 0F85B7FEFFFF          jne 00402C47
:00402D90 8B45E0                mov eax, dword ptr [ebp-20]
:00402D93 0345BC                add eax, dword ptr [ebp-44]
:00402D96 0F80910A0000          jo 0040382D                       <-- jo stands for jump overflow
:00402D9C 0345B0                add eax, dword ptr [ebp-50]
:00402D9F 0F80880A0000          jo 0040382D
```

```
:00402DA5 0345BC                add eax, dword ptr [ebp-44]
:00402DA8 0F807F0A0000          jo 0040382D
:00402DAE 0345CC                add eax, dword ptr [ebp-34]
:00402DB1 0F80760A0000          jo 0040382D
:00402DB7 0345C4                add eax, dword ptr [ebp-3C]
:00402DBA 0F806D0A0000          jo 0040382D
:00402DC0 0345BC                add eax, dword ptr [ebp-44]
:00402DC3 0F80640A0000          jo 0040382D
:00402DC9 8945C8                mov dword ptr [ebp-38], eax
:00402DCC 8B45C0                mov eax, dword ptr [ebp-40]
:00402DCF 3B45C8                cmp eax, dword ptr [ebp-38]
:00402DD2 0F85B9060000          jne 00403491            <-- Important Compare
```

Well just look at the values behind ebp-xx, add them together (Total: 2DC).
You'll notice that the characters form the word "Reverse"
This value is compared with the total of the the ascii values of your serial..
Enter the serial and you've reversed this program !!!!
Now I tried to enter a code containing my nickname and some other characters that form a total of 2DC. And it didn't work :((
So there is more work to do for us :)
Open smartcheck, load the program enter the correct code and take a look at the new information:

MID(Variant:String:"Reverse",long:2,Variant:Integer:1)
MID(Variant:String:"Reverse",long:4,Variant:Integer:1)
MID(Variant:String:"Reverse",long:7,Variant:Integer:1)

Thus we see that the program checks is the characters at position 2, 4 and 7 are the same as the ones in the word reverse. Knowing this we can find ourselves some other good serials.
Examples:
Reserve,Veserre,Peveste etc..
You could have done this at the SoftICE way too, two of the checks would be at addresses: 403115 and 403147

I like to thank Eternal Bliss for his great page on the web. Where Fravia+ gives all the theoretical information Eternal Bliss gives us the **BEST** page on the web with lots of bits and bytes to practice.

Also I like to great all the guys from #cracking4newbies #dread #faith2000 and #win32asm :))

I want to tell all the people reading this essay to **START REVERSING CRACKME'S THEMSELVES** since loads of crackmes on Eternal Bliss' site haven't been reversed yet :))
Feel free to send all your questions & comments to Rhytm@newmail.net

Bye !!!!
Rhytm, Hope to see you soon in #cracking4newbies