**Reversing VB Crackme 4.0 Eternal Bliss**
**by Rhytm [Dread]**

(This short tutorial was written in Wordpad)

Collecting Information

Heeeellllloooooo !!! (That'll wake everyone).. Let's start reversing Eternal Bliss' 3rd crackme. I hope you've tried his two other crackme's before starting to reverse this one ***grin***
As always first take a look at Eternal Bliss' .txt file:

*What to do with this CrackMe:*
*Find out how the correct code is generated. No KeyGen is asked from you. Just need to send me an email saying how you manage to find out how the correct code is generated.*

*Since this is more of a practice and taste of VB cracking, I'll give you a few hints:*
*1) If you want to crack it fast, use SmartCheck first and look for breakpoints for Softice.*
*2) Before using Softice, look at what SmartCheck has to show you.*

Hmmm.. No Keygen is asked ?? So one is possible !! :)
ok, fire up Smartcheck (I just luvvvv this tool) and load the crackme, enter a dummy serial, press the register button and take a look at smartcheck code..

Notice that your systemdate will be important during this crackme !!
There's some fooling around with the systemdate, after this the most right number of the first piece of your serial is grabbed and the program has the usual unexpected end...

Analysing the Info and Reverse the Target

Well lets take a look at All Events in Smartcheck.. Notice the __vbaVarTstEq right after the "right" function 8). Lets take a look at that one in SoftICE :P
Fie it up, set your breakpoints, trace a bit here, trace a bit there..
The messagebox function is a nice way to get a hold on the program. The function name is: rtcMsgBox. There are three calls to the messagebox: the two we're interested in are the ones located at 40328F and 403322.
403322 = Badguy
40328F = Goodguy
4030BC = You've entered nothing
There is one conditional jump that'll make is go to either the wrong or the right messagebox, it's located at 40321A...
Here's a little snippet to make things clear:

---

```
:004031E6(U)

:004031ED 8B4508                mov eax, dword ptr [ebp+08]
:004031F0 8B4D08                mov ecx, dword ptr [ebp+08]
```

```
:004031F3 8B8080000000          mov eax, dword ptr [eax+00000080]
:004031F9 2B818C000000          sub eax, dword ptr [ecx+0000008C]      <-- Eax can't be
zero here!
:004031FF 0F80EC010000          jo 004033F1
:00403205 85C0                  test eax, eax                          <-- If the result is one
:00403207 750A                  jne 00403213                           proceed and set the flag
to
:00403209 8B4508                mov eax, dword ptr [ebp+08]            1
:0040320C C7403401000000        mov [eax+34], 00000001

:00403207(C)

:00403213 8B4508                mov eax, dword ptr [ebp+08]
:00403216 83783401              cmp dword ptr [eax+34], 00000001       <-- Then the
compare will
:0040321A 0F8593000000          jne 004032B3                           be ok. Goto goodguy
:00403220 C7459404000280        mov [ebp-6C], 80020004
:00403227 C7458C0A000000        mov [ebp-74], 0000000A
:0040322E C745A404000280        mov [ebp-5C], 80020004
:00403235 C7459C0A000000        mov [ebp-64], 0000000A
```

---

The problem is the program still doesn't reach 403205, so lets backtrace some more :)

---

```
:00403198 0F8053020000          jo 004033F1
:0040319E 85C0                  test eax, eax
:004031A0 7546                  jne 004031E8                           <-- Jumps to the Badguy
:004031A2 8B4508                mov eax, dword ptr [ebp+08]
:004031A5 8B00                  mov eax, dword ptr [eax]
:004031A7 FF7508                push [ebp+08]
:004031AA FF9000070000          call dword ptr [eax+00000700]
:004031B0 898548FFFFFF          mov dword ptr [ebp+FFFFFF48], eax
:004031B6 83BD48FFFFFF00        cmp dword ptr [ebp+FFFFFF48], 00000000
:004031BD 7D20                  jge 004031DF                           <-- Dunno
:004031BF 6800070000            push 00000700
:004031C4 68BC244000            push 004024BC
:004031C9 FF7508                push [ebp+08]
:004031CC FFB548FFFFFF          push dword ptr [ebp+FFFFFF48]
:004031D2 E8FFE0FFFF            Call 004012D6                          <-- vbaHresultCheckObj
:004031D7 898508FFFFFF          mov dword ptr [ebp+FFFFFF08], eax
:004031DD EB07                  jmp 004031E6

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004031BD(C)

:004031DF 83A508FFFFFF00        and dword ptr [ebp+FFFFFF08], 00000000

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
:004031DD(U)

:004031E6 EB05                  jmp 004031ED                           <-- Good
```

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
:004031A0(C)

:004031E8 E9C6000000                jmp 004032B3                    <-- Bad
```

---

As we can see we need to take the jump at 4031E6 and not the one at 4031E8. To reach this jump we need to pass the test at **40319E** and maybe we have to deal with the jump at 4031BD later..
So lets enter a dummy serial again, fire up SoftICE, set some breakpoints and press the register button :P
Hmmm... Again no luck here.. Lets backtrace just a little more :)))

---

```
:00403152 83BD48FFFFFF00            cmp dword ptr [ebp+FFFFFF48], 00000000
:00403159 7D20                      jge 0040317B            <-- Don't take this jump !!!!
:0040315B 68FC060000                push 000006FC
:00403160 68BC244000                push 004024BC
:00403165 FF7508                    push [ebp+08]
:00403168 FFB548FFFFFF              push dword ptr [ebp+FFFFFF48]
:0040316E E863E1FFFF                Call 004012D6                   <-- __vbaHresultCheckObj
:00403173 89850CFFFFFF              mov dword ptr [ebp+FFFFFF0C], eax
:00403179 EB07                          jmp 00403182

:00403159(C)

:0040317B 83A50CFFFFFF00            and dword ptr [ebp+FFFFFF0C], 00000000

:00403179(U)

:00403182 EB05                      jmp 00403189            <-- Goodguy

:0040313C(C)

:00403184 E92A010000                jmp 004032B3            <-- Badguy
```

---

ok, so we need to take the jump at 403179, and **NOT** the one located at 403159. Lets enable our breakpoints in SoftICE again and set a new breakpoint on segment:403152..
AAAAARRRRGGGGGGGGHHHHHH, no hit again :)) Looks like more backtracing is needed to get to the core of this crackme 8)

---

```
:00403111 E8C0E1FFFF                Call 004012D6
:00403116 898510FFFFFF              mov dword ptr [ebp+FFFFFF10], eax
:0040311C EB07                          jmp 00403125

:004030FC(C)
```

```
:0040311E 83A510FFFFFF00          and dword ptr [ebp+FFFFFF10], 00000000

:0040311C(U)

:00403125 8B4508                  mov eax, dword ptr [ebp+08]
:00403128 8B4D08                  mov ecx, dword ptr [ebp+08]      <-- Important
:0040312B 8B4078                  mov eax, dword ptr [eax+78]
:0040312E 2B8184000000            sub eax, dword ptr [ecx+00000084]<-- The result has to be 0
:00403134 0F80B7020000            jo 004033F1
:0040313A 85C0                    test eax, eax
:0040313C 7546                    jne 00403184                     <-- Noooooooooo !!!!!
:0040313E 8B4508                  mov eax, dword ptr [ebp+08]
:00403141 8B00                          mov eax, dword ptr [eax]
```

---

Now we're getting hot !! We **DO** reach the check at 40313C. Now lets find out two things:

1. What's behind ecx+84
2. How to be sure that eax = ecx+84

Lets do another round of SoftICE and check the code starting from 40312B:
I got this info:
[eax + 78] = "3" and [ecx + 84] = "6F"

**(Notice that the date I'm working with is 17 March 1999)**

Two things popped into my mind, the first thing was that the month is march (3) and that the total of the numbers I entered in the first box was 3. The first thing didn't make sense since it couldn't be the 6Fth month.
So the 3 is a constant value can be your month. Lets be sure, change your month and try again. Aha !! The value has changed to 4 :) (And all the shareware on my computer stopped working.. lol..) So we have to be sure that our value is the same as the month..
Since the decimal value of 0x3 is 3 we have to enter 3 in the first box..
Set your breakpoints again and go to the call, you'll pass the first test and reach a second one.. Your value has to be 0x154 = 340 decimal.. Re-fill the boxes with the new number and try again, you'll pass the second test !! Now you reach the third test (You should know the story by now)... Start all over again enter all numbers press the register button and we did it !!!!!

Now we should find out how the second and third number are generated using the month and year.. Lets fire up smartcheck, enter a correct serial and take a look at the code..
We like to know more about the text entered in the second box, that's why i set my breakpoint at 403B2F (the __vbaVarCat function (adds two numbers as a string))..

Keep stepping for a little while, till you see a familiar piece of code :)))
It has the same structure as the comparepieces we've seen before.

```
:00403B75 8B4508                  mov eax, dword ptr [ebp+08]
:00403B78 8B4D08                  mov ecx, dword ptr [ebp+08]
:00403B7B 8B4078                  mov eax, dword ptr [eax+78]
```

```
:00403B7E 03417C                add eax, dword ptr [ecx+7C]
:00403B81 0F80C9010000          jo 00403D50
:00403B87 8B4D08                mov ecx, dword ptr [ebp+08]
:00403B8A 8B497C                mov ecx, dword ptr [ecx+7C]
:00403B8D 0FAFC8                imul ecx, eax
```

Study this routine, and soon you'll found out that:

(0x3  + 0x11) x 0x11 = 0x154  <-- Serial we've entered in the second box
(3 + 17) x 17 = 340
(Month + Day) * Day = 340

Cool !! :) Lets try do the same thing for the third part.. So lets take the __vbaVarCat function that combines the 9 and 9 of the year together.. The address is 403EB3..
After some stepping you'll get to the same routine as before.
Study this routine, and soon you'll found out that:

(0x154 + 0x63) x 0x63 = 0xA9C5
(340 + 99) x 99 = 43461
(Result of second part + year) x year = 43461

Jippieee !!!!
we've solved the crackme :)))))) You see it...
Lets write a keygen for it.. :)
I'm using Java, just use your own language !!

---

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