

第十一章：用 NOOB 技术破解

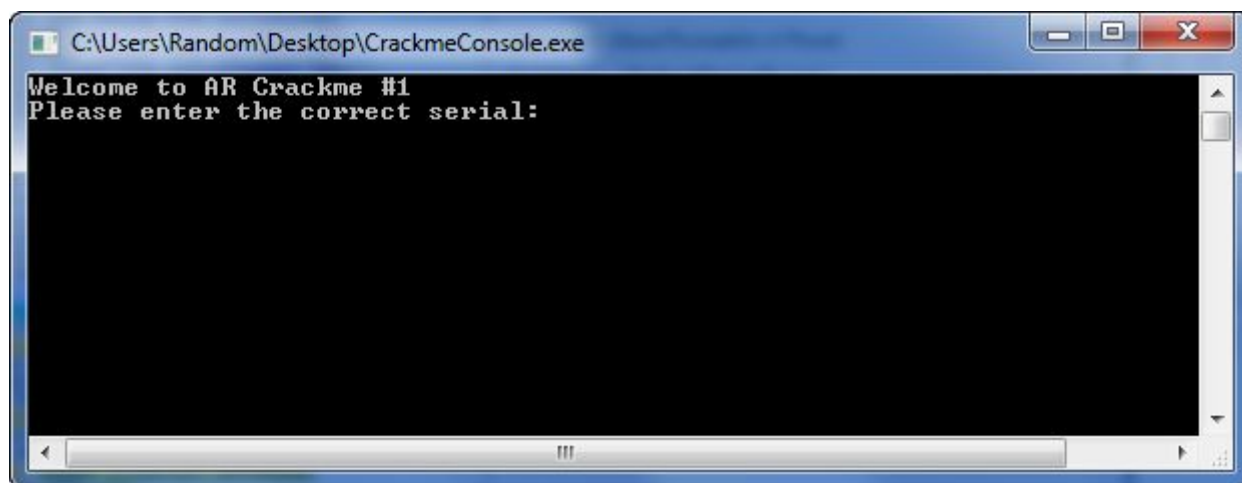
一、简介

本章我们将再次讨论补丁程序，不过比典型的单个“我们遇到的第一个补丁”要深入一点点。我们将从一个控制台程序开始，找到隐藏在其中的正确密码。教程的相关下载中有。除此之外，你只需要 OllyDbg。

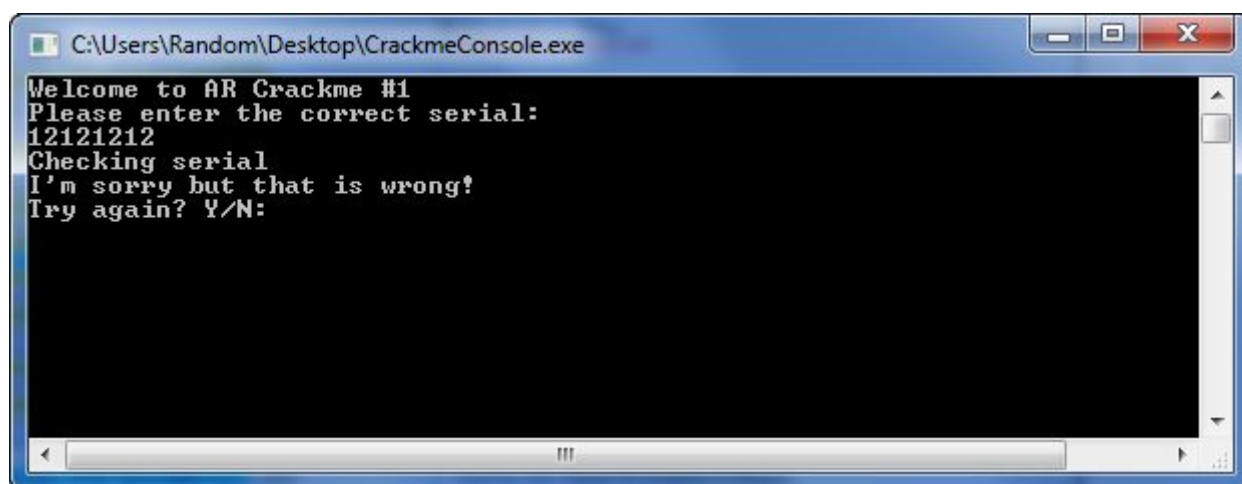
你可以在[教程](#)页下载相关文件及本文的 PDF 版。

那么，咱们开始吧....

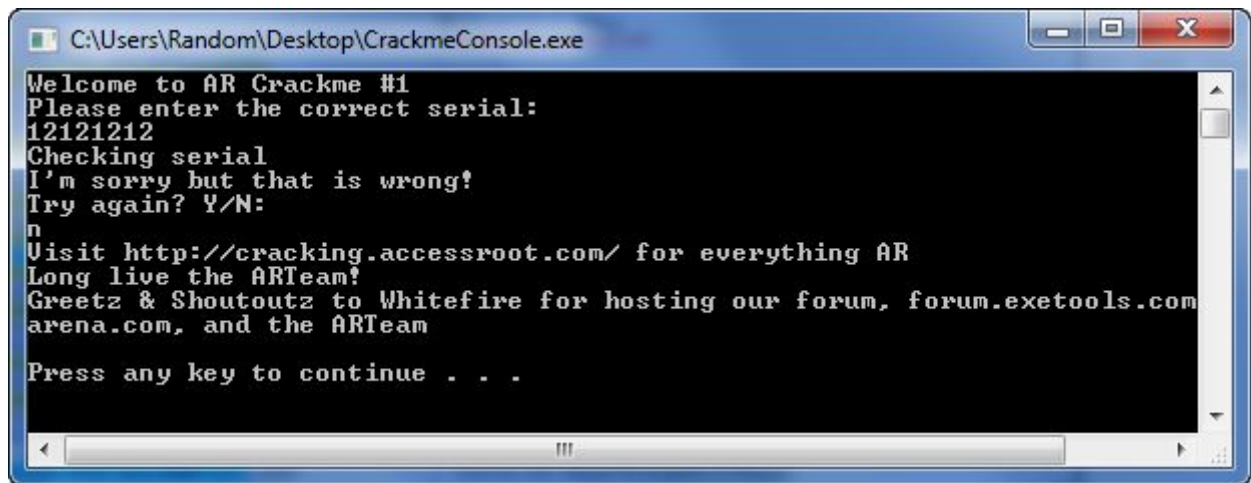
控制台程序是和其他 windows 下 32 位的程序一样。唯一的不同是它们不使用图形界面。除此之外，它们是一样的。此次的 crackme 叫 CrackmeConsole.exe。咱们来运行一下看看情况：



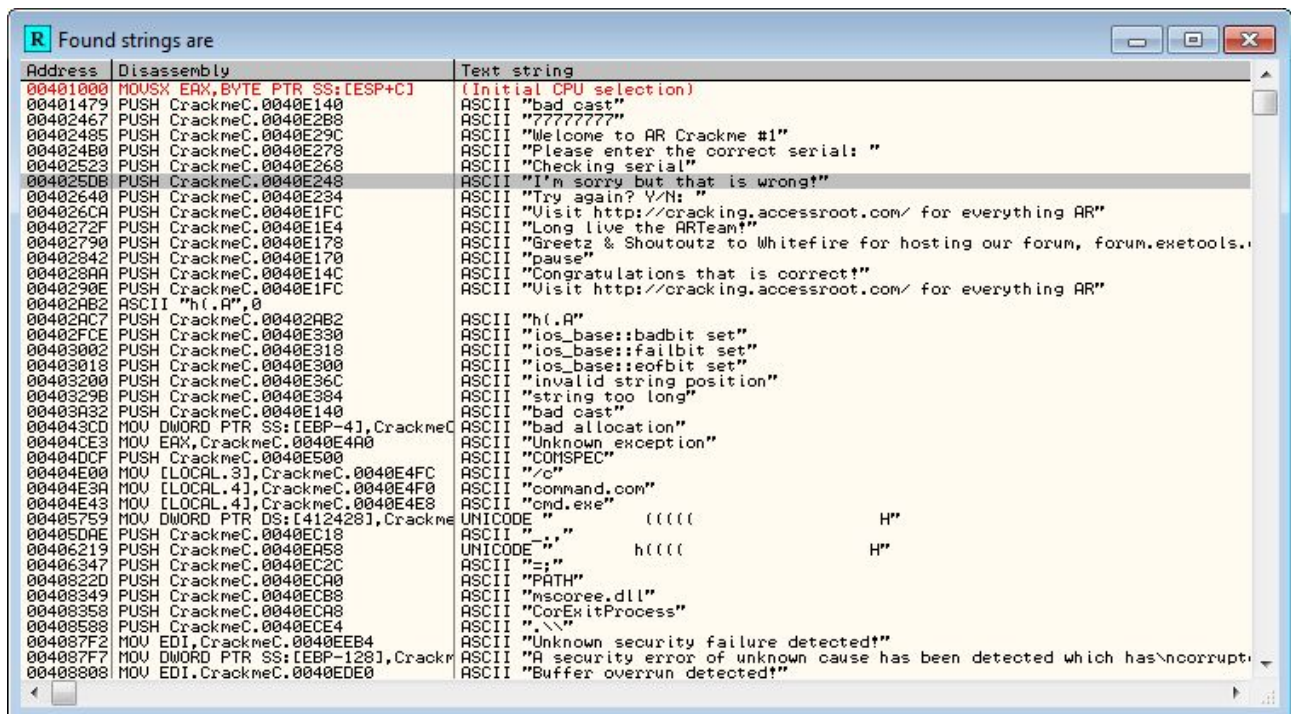
好，看起来挺简单得。咱们来随便输个密码：



真失败！按“N”结束程序吧：



好吧，我觉得至少我们有了足够的信息来开始研究它。GO，Olly 载入应用。开始，首先搜索字符串：



不是很难嘛！双击坏消息 “I m sorry, but that is wrong”，至少来到了正确的地方：

004025B0	74 05	JE SHORT CrackmeC.004025C4	
004025B1	1BC0	SBB EAX,EAX	kernel32.BaseThreadInitThunk
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025D8	
004025C8	3BD5	CMP EDI,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025D8	
004025CC	33C0	XOR EAX,EAX	kernel32.BaseThreadInitThunk
004025CE	3BD5	CMP EDI,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wr
004025E0	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	83C4 08	ADD ESP,8	
004025ED	8BF0	MOV ESI,EAX	kernel32.BaseThreadInitThunk
004025EF	6A 0A	PUSH 0A	
004025F1	8BCE	MOV ECX,ESI	
004025F3	E8 D8F8FFFF	CALL CrackmeC.00401ED0	
004025F8	8B0E	MOV ECX,DWORD PTR DS:[ESI]	
004025FA	8B51 04	MOV EDX,DWORD PTR DS:[ECX+4]	
004025FD	8A4C32 08	MOV CL,BYTE PTR DS:[EDX+ESI+8]	
00402601	8D0432	LEA EAX,DWORD PTR DS:[EDX+ESI]	
00402604	33FF	XOR EDI,EDI	
00402606	F6C1 06	TEST CL,6	
00402609	75 14	JNZ SHORT CrackmeC.0040261F	
0040260B	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	
0040260E	8B10	MOV EDX,DWORD PTR DS:[EAX]	
00402610	8BC8	MOV ECX,EAX	kernel32.BaseThreadInitThunk
00402612	FF52 2C	CALL DWORD PTR DS:[EDX+2C]	
00402615	83F8 FF	CMP EAX,-1	
00402618	75 05	JNZ SHORT CrackmeC.0040261F	
0040261A	BF 04000000	MOV EDI,4	
0040261F	8B06	MOV EAX,DWORD PTR DS:[ESI]	

好，咱们研究研究这个。我们看到一个从 4025C6 来的跳转，有红色箭头标出来了。我们也注意到，如果 4025D5 的 JE 指令没实现的话，我们也会得到坏消息。咱们来看看如果这个跳转实现的话会怎么样。点它：

004025CE	3BD5	CMP EDI,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wr
004025E0	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	83C4 08	ADD ESP,8	
004025ED	8BF0	MOV ESI,EAX	kernel32.BaseThreadInitThunk
004025EF	6A 0A	PUSH 0A	
004025F1	8BCE	MOV ECX,ESI	
004025F3	E8 D8F8FFFF	CALL CrackmeC.00401ED0	
004025F8	8B0E	MOV ECX,DWORD PTR DS:[ESI]	
004025FA	8B51 04	MOV EDX,DWORD PTR DS:[ECX+4]	
004025FD	8A4C32 08	MOV CL,BYTE PTR DS:[EDX+ESI+8]	
00402601	8D0432	LEA EAX,DWORD PTR DS:[EDX+ESI]	
00402604	33FF	XOR EDI,EDI	
00402606	F6C1 06	TEST CL,6	
00402609	75 14	JNZ SHORT CrackmeC.0040261F	
0040260B	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	
0040260E	8B10	MOV EDX,DWORD PTR DS:[EAX]	
00402610	8BC8	MOV ECX,EAX	kernel32.BaseThreadInitThunk
00402612	FF52 2C	CALL DWORD PTR DS:[EDX+2C]	
00402615	83F8 FF	CMP EAX,-1	
00402618	75 05	JNZ SHORT CrackmeC.0040261F	
0040261A	BF 04000000	MOV EDI,4	
0040261F	8B06	MOV EAX,DWORD PTR DS:[ESI]	
00402621	8B48 04	MOV ECX,DWORD PTR DS:[EAX+4]	
00402624	03CE	ADD ECX,ESI	
00402626	3BFB	CMP EDI,EBX	
00402628	74 16	JE SHORT CrackmeC.00402640	
0040262A	8B41 08	MOV EAX,DWORD PTR DS:[ECX+8]	
0040262D	8B51 28	MOV EDX,DWORD PTR DS:[ECX+28]	
00402630	0BC7	OR EAX,EDI	
00402632	3BFB	CMP EDI,EBX	

滚动到它指向的地方（在下面几页）：

00402881	72 00	JB SHORT CrackmeC.00402890	
00402883	8B5424 24	MOV EDX,DWORD PTR SS:[ESP+24]	CrackmeC.<ModuleEntryPoint>
00402887	52	PUSH EDX	
00402888	E8 C01E0000	CALL CrackmeC.00404740	
0040288D	83C4 04	ADD ESP,4	
00402890	8B4C24 40	MOV ECX,DWORD PTR SS:[ESP+40]	
00402894	64:8900 00000000	MOV DWORD PTR FS:[0],ECX	ntdll.77D2E115
0040289B	8B4C24 3C	MOV ECX,DWORD PTR SS:[ESP+3C]	kernel32.BaseThreadInitThunk
0040289F	33C0	XOR EAX,EAX	
004028A1	E8 C0260000	CALL CrackmeC.00404F66	
004028A6	83C4 4C	ADD ESP,4C	
004028A9	C3	RET	
004028AA	68 4CE14000	PUSH CrackmeC.0040E14C	ASCII "Congratulations that is correc
004028AF	68 A02F4100	PUSH CrackmeC.00412FA0	
004028B4	E8 D7F1FFFF	CALL CrackmeC.00401A90	
004028B9	83C4 08	ADD ESP,8	
004028BC	8BF0	MOV ESI,EAX	kernel32.BaseThreadInitThunk
004028BE	6A 0A	PUSH 0A	
004028C0	8BCE	MOV ECX,ESI	
004028C2	E8 09F6FFFF	CALL CrackmeC.00401ED0	
004028C7	8B06	MOV EAX,DWORD PTR DS:[ESI]	
004028C9	8B48 04	MOV ECX,DWORD PTR DS:[EAX+4]	
004028CC	8D431	LEA EAX,DWORD PTR DS:[ECX+ESI]	
004028CF	8A48 08	MOV CL,BYTE PTR DS:[EAX+8]	
004028D2	33FF	XOR EDI,EDI	
004028D4	F6C1 06	TEST CL,6	
004028D7	75 14	JNZ SHORT CrackmeC.004028ED	
004028D9	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	

这看起来就是我们想走的路😁。咱们回到上面看看周围地方：

00402571	8B41 08	MOV EAX,DWORD PTR DS:[ECX+8]	
00402574	8B51 28	MOV EDX,DWORD PTR DS:[ECX+28]	
00402577	0BC7	OR EAX,EDI	
00402579	3BD3	CMP EDX,EBX	
0040257B	75 03	JNZ SHORT CrackmeC.00402580	
0040257D	83C8 04	OR EAX,4	
00402580	53	PUSH EBX	
00402581	50	PUSH EAX	kernel32.BaseThreadInitThunk
00402582	E8 100A0000	CALL CrackmeC.00402F97	
00402587	837C24 2C 10	CMP DWORD PTR SS:[ESP+2C],10	
0040258C	8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	
00402590	73 04	JNB SHORT CrackmeC.00402596	
00402592	8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	
00402596	8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	
0040259A	3BD3	CMP EDX,EBX	
0040259C	8B6C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	
004025A0	74 26	JE SHORT CrackmeC.004025C8	
004025A2	3BD5	CMP EDX,EBP	
004025A4	8BCA	MOV ECX,EDX	CrackmeC.<ModuleEntryPoint>
004025A6	72 02	JB SHORT CrackmeC.004025AA	
004025A8	8BCD	MOV ECX,EBP	
004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	
004025B9	33C0	XOR EAX,EAX	kernel32.BaseThreadInitThunk
004025BB	F3A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:	
004025BD	74 05	JE SHORT CrackmeC.004025C4	kernel32.BaseThreadInitThunk
004025BF	1BC0	SBB EAX,EAX	
004025C1	83D8 FF	SBB EAX,-1	kernel32.BaseThreadInitThunk
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	
004025C8	3BD5	CMP EDX,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	kernel32.BaseThreadInitThunk
004025CE	3BD5	CMP EDX,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
004025DE	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	83C4 08	ADD ESP,8	
004025ED	8BF0	MOV ESI,EAX	kernel32.BaseThreadInitThunk
004025EF	6A 0A	PUSH 0A	
004025F1	8BCE	MOV ECX,ESI	
004025F3	E8 D8F8FFFF	CALL CrackmeC.00401ED0	
004025F8	8B0E	MOV ECX,DWORD PTR DS:[ESI]	
004025FA	8B51 04	MOV EDX,DWORD PTR DS:[ECX+4]	
004025FD	8A4C32 08	MOV CL,BYTE PTR DS:[EDX+ESI+8]	
00402601	8D432	LEA EAX,DWORD PTR DS:[EDX+ESI]	
00402604	33FF	XOR EDI,EDI	
00402606	F6C1 06	TEST CL,6	
00402609	75 14	JNZ SHORT CrackmeC.0040261F	
0040260B	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	
0040260E	8B10	MOV EDX,DWORD PTR DS:[EAX]	

4025D5是调到好消息那的，这就是我们想要实现的跳转。咱们点一下另一个跳转看看它将跳到哪去...。说不定前面也有个跳转可以跳到好消息那呢：

004025BF	1BC0	SBB EAX,EAX	kernel32.BaseThreadInitThunk
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	
004025C8	3BD5	CMP EDX,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	kernel32.BaseThreadInitThunk
004025CE	3BD5	CMP EDX,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
004025E0	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	83C4 08	ADD ESP,8	

这个是到坏消息的:

004025BB	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
004025BD	74 05	JE SHORT CrackmeC.004025C4	
004025BF	1BC0	SBB EAX,EAX	kernel32.BaseThreadInitThunk
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	
004025C8	3BD5	CMP EDX,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	kernel32.BaseThreadInitThunk
004025CE	3BD5	CMP EDX,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
004025E0	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	83C4 08	ADD ESP,8	

这个也是，如果你接着点那些跳转指令，你会发现 4025D5 是唯一一个跳到好消息的跳转。所以基本上，我们要阻止所有跳到坏消息的跳转实现，强制跳到好消息的跳转成功跳转。如果我们接着往上滚动，就会在 402582 找到第一个 call/compare (调用/比较) 指令:

00402579	3BD3	CMP EDX,EBX	
0040257B	75 03	JNZ SHORT CrackmeC.00402580	
0040257D	83C8 04	OR EAX,4	
00402580	53	PUSH EBX	
00402581	50	PUSH EAX	kernel32.BaseThreadInitThunk
00402582	E8 100A0000	CALL CrackmeC.00402F97	
00402587	837C24 2C 10	CMP DWORD PTR SS:[ESP+2C],10	
0040258C	8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	
00402590	73 04	JNB SHORT CrackmeC.00402596	
00402592	8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	
00402596	8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	
0040259A	3BD3	CMP EDX,EBX	
0040259C	8B6C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	
004025A0	74 26	JE SHORT CrackmeC.004025C8	
004025A2	3BD5	CMP EDX,EBP	
004025A4	8BCA	MOV ECX,EDX	CrackmeC.<ModuleEntryPoint>
004025A6	72 02	JB SHORT CrackmeC.004025AA	
004025A8	8BCD	MOV ECX,EBP	
004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	
004025B9	33C0	XOR EAX,EAX	kernel32.BaseThreadInitThunk
004025BB	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	

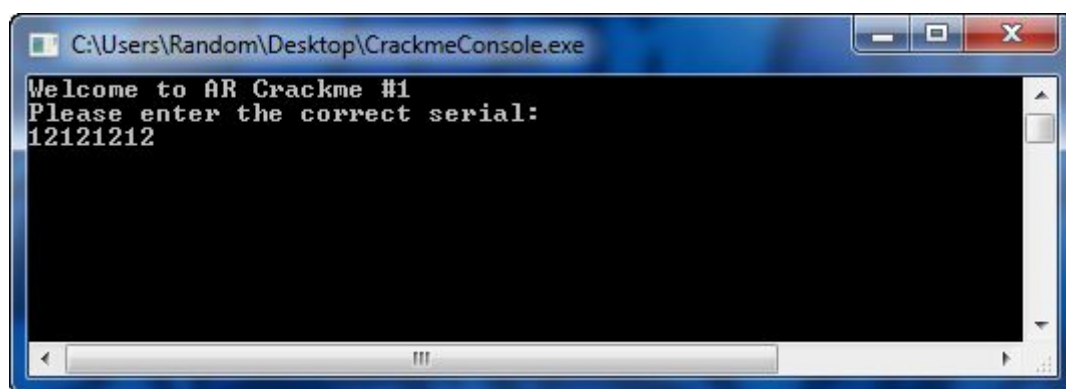
再往上滚动，就会发现有个跳转跳过了那个 CALL，但是仍然进行了比较:

0040255F	75 05	JNZ SHORT CrackmeC.004025BB	
00402561	BF 04000000	MOV EDI,4	
00402566	8B06	MOV EAX,DWORD PTR DS:[ESI]	
00402568	8B48 04	MOV ECX,DWORD PTR DS:[EAX+4]	
0040256B	03CE	ADD ECX,ESI	
0040256D	3BF8	CMP EDI,EBX	
0040256F	74 16	JE SHORT CrackmeC.00402587	
00402571	8B41 08	MOV EAX,DWORD PTR DS:[ECX+8]	
00402574	8B51 28	MOV EDX,DWORD PTR DS:[ECX+28]	
00402577	0BC7	OR EAX,EDI	
00402579	3BD3	CMP EDX,EBX	
0040257B	75 03	JNZ SHORT CrackmeC.00402580	
0040257D	83C8 04	OR EAX,4	
00402580	53	PUSH EBX	
00402581	50	PUSH EAX	kernel32.BaseThreadInitThunk
00402582	E8 100A0000	CALL CrackmeC.00402F97	
00402587	837C24 2C 10	CMP DWORD PTR SS:[ESP+2C],10	
0040258C	8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	
00402590	73 04	JNB SHORT CrackmeC.00402596	
00402592	8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	
00402596	8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	
0040259A	3BD3	CMP EDX,EBX	
0040259C	8B6C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	
004025A0	74 26	JE SHORT CrackmeC.004025C8	
004025A2	3BD5	CMP EDX,EBP	

这个行为不太正常，如果我们再往上滚动一点，就会发现另外一对 调用/比较 指令对。我在这两个 CALL 上都设置了 BP:

00402542	8B42 04	MOV EAX,DWORD PTR DS:[EDX+4]	
00402545	8A4C30 08	MOV CL,BYTE PTR DS:[EAX+ESI+8]	
00402549	03C6	ADD EAX,ESI	
0040254B	33FF	XOR EDI,EDI	
0040254D	F6C1 06	TEST CL,6	
00402550	75 14	JNZ SHORT CrackmeC.00402566	
00402552	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	
00402555	8B10	MOV EDX,DWORD PTR DS:[EAX]	
00402557	8BC8	MOV ECX,EAX	kernel32.BaseThreadInitThunk
00402559	FF52 2C	CALL DWORD PTR DS:[EDX+2C]	
0040255C	83F8 FF	CMP EAX,-1	
0040255F	75 05	JNZ SHORT CrackmeC.00402566	
00402561	BF 04000000	MOV EDI,4	
00402566	8B06	MOV EAX,DWORD PTR DS:[ESI]	
00402568	8B48 04	MOV ECX,DWORD PTR DS:[EAX+4]	
0040256B	03CE	ADD ECX,ESI	
0040256D	3BFB	CMP EDI,EBX	
0040256F	74 16	JE SHORT CrackmeC.00402587	
00402571	8B41 08	MOV EAX,DWORD PTR DS:[ECX+8]	
00402574	8B51 28	MOV EDX,DWORD PTR DS:[ECX+28]	
00402577	0BC7	OR EAX,EDI	
00402579	3BD3	CMP EDX,EBX	
0040257B	75 03	JNZ SHORT CrackmeC.00402580	
0040257D	83C8 04	OR EAX,4	
00402580	53	PUSH EBX	kernel32.BaseThreadInitThunk
00402581	50	PUSH EAX	
00402582	E8 100A0000	CALL CrackmeC.00402F97	
00402587	837C24 2C 10	CMP DWORD PTR SS:[ESP+2C],10	
0040258C	8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	
00402590	73 04	JNB SHORT CrackmeC.00402596	
00402592	8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	
00402596	8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	
0040259A	3BD3	CMP EDX,EBX	
0040259C	8B6C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	

好吧，咱们继续，在 Olly 中运行程序看看会发生什么。我将输入密码“12121212”：



Olly 断在了第一个 CALL：

00402545	8A4C30 08	MOV CL,BYTE PTR DS:[EAX+ESI+8]	
00402549	03C6	ADD EAX,ESI	CrackmeC.00412FA0
0040254B	33FF	XOR EDI,EDI	
0040254D	F6C1 06	TEST CL,6	
00402550	75 14	JNZ SHORT CrackmeC.00402566	
00402552	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	CrackmeC.0040E408
00402555	8B10	MOV EDX,DWORD PTR DS:[EAX]	CrackmeC.00412F40
00402557	8BC8	MOV ECX,EAX	CrackmeC.004037A1
00402559	FF52 2C	CALL DWORD PTR DS:[EDX+2C]	
0040255C	83F8 FF	CMP EAX,-1	
0040255F	75 05	JNZ SHORT CrackmeC.00402566	
00402561	BF 04000000	MOV EDI,4	
00402566	8B06	MOV EAX,DWORD PTR DS:[ESI]	CrackmeC.0040E470
00402568	8B48 04	MOV ECX,DWORD PTR DS:[EAX+4]	CrackmeC.00412FA0
0040256B	03CE	ADD ECX,ESI	
0040256D	3BFB	CMP EDI,EBX	
0040256F	74 16	JE SHORT CrackmeC.00402587	
00402571	8B41 08	MOV EAX,DWORD PTR DS:[ECX+8]	
00402574	8B51 28	MOV EDX,DWORD PTR DS:[ECX+28]	
00402577	0BC7	OR EAX,EDI	
00402579	3BD3	CMP EDX,EBX	
0040257B	75 03	JNZ SHORT CrackmeC.00402580	
0040257D	83C8 04	OR EAX,4	
00402580	53	PUSH EBX	
00402581	50	PUSH EAX	CrackmeC.00412F40
00402582	E8 100A0000	CALL CrackmeC.00402F97	
00402587	837C24 2C 10	CMP DWORD PTR SS:[ESP+2C],10	
0040258C	8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	
00402590	73 04	JNB SHORT CrackmeC.00402596	
00402592	8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	
00402596	8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	
0040259A	3BD3	CMP EDX,EBX	
0040259C	8B6C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	

单步调试，注意 42056F 处的跳转跳过了第二个 CALL。嗯，这倒给了我们一个提示，第二个跳转可能不是校验密码的，不过有可能是某种验证程序，如

果我们的密码不符合某种规则，比如太短或者太长？不管是啥，咱们接着单步运行就行了：

0040254D	F6C1 06	TEST CL,6	
00402550	75 14	JNZ SHORT CrackmeC.00402566	
00402552	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	CrackmeC.00404D25
00402555	8B10	MOV EDX,DWORD PTR DS:[EAX]	
00402557	8BC8	MOV ECX,EAX	CrackmeC.0040E470
00402559	FF52 2C	CALL DWORD PTR DS:[EDX+2C]	CrackmeC.004037A1
Jumps past our call			
00402563	8B48 04	MOV EAX,DWORD PTR DS:[ESI]	CrackmeC.0040E470
00402566	03CE	ADD ECX,ESI	
0040256D	3BFB	CMP EDI,EBX	CrackmeC.00412FA0
0040256F	74 16	JE SHORT CrackmeC.00402587	
00402571	8B41 08	MOV EAX,DWORD PTR DS:[ECX+8]	
00402574	8B51 28	MOV EDX,DWORD PTR DS:[ECX+28]	CrackmeC.00412F40
00402577	0BC7	OR EAX,EDI	
00402579	3BD3	CMP EDX,EBX	
0040257B	75 03	JNZ SHORT CrackmeC.00402580	
0040257D	83C8 04	OR EAX,4	
00402580	53	PUSH EBX	
00402581	50	PUSH EAX	CrackmeC.0040E470
00402583	E8 100A0000	CALL CrackmeC.00402F97	
00402587	837C24 2C 10	CMP DWORD PTR SS:[ESP+2C],10	
0040258C	8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	
00402590	73 04	JNB SHORT CrackmeC.00402596	
00402592	8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	
00402596	8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	
0040259A	3BD3	CMP EDX,EBX	
0040259C	8B7C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	

4025C6 这里，咱们看到了罪魁祸首了，就是它跳到了坏消息那：

004025A6	72 02	JB SHORT CrackmeC.004025AA	
004025A8	8BCD	MOV ECX,EBP	
004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	
004025B9	33C0	XOR EAX,EAX	
004025BB	F3A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
004025BD	74 05	JE SHORT CrackmeC.004025C4	
004025BF	1BC0	SBB EAX,EAX	
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	
004025C8	3BD5	CMP EDX,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	
004025CE	3BD5	CMP EDX,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	
004025DB	68 A02F4100	PUSH CrackmeC.00412FA0	ASCII "I'm sorry but that is wrong!"
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	83C4 08	ADD ESP,8	
004025ED	8BF0	MOV ESI,EAX	
004025EF	6A 0A	PUSH 0A	
004025F1	8BCE	MOV ECX,ESI	
004025F3	E8 D8F8FFFF	CALL CrackmeC.00401ED0	
004025F8	8B0E	MOV ECX,DWORD PTR DS:[ESI]	

咱们设置下 0 标志位，看看会怎么样：

```

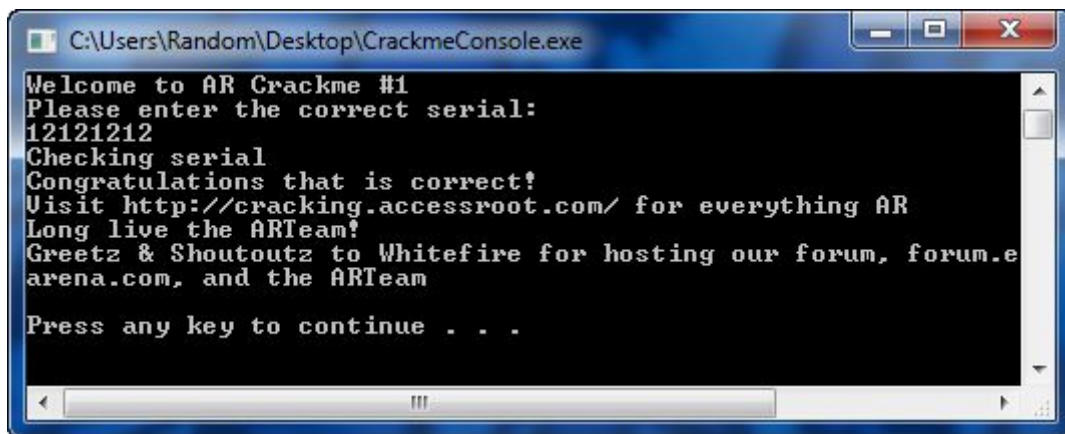
C 0  ES 0023
P 1  CS 001B
A 0  SS 0023
Z 1  DS 0023
S 1  FS 003B
T 0  GS 0000
O 0
O 0  LastErr

```

继续单步，终于和跳到好消息的跳转碰头了，注意它实现了：

00402500	33C0	XOR EAX,EAX	
00402501	3BD5	CMP EDX,EBP	
00402502	0F95C0	SETNE AL	
00402503	3BC3	CMP EAX,EBX	
00402504	0F84 CF020000	JE CrackmeC.004028AA	
00402505	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
00402506	68 A02F4100	PUSH CrackmeC.00412FA0	
00402507	E8 46F4FFFF	CALL CrackmeC.00401A90	
00402508	83C4 08	ADD ESP,8	
00402509	8BF0	MOV ESI,EAX	
0040250A	6A 0A	PUSH 0A	
0040250B	8BCE	MOV ECX,ESI	
0040250C	E8 D8F8FFFF	CALL CrackmeC.00401ED0	
0040250D	8B0E	MOV ECX,DWORD PTR DS:[ESI]	
0040250E	8B51 04	MOV EDX,DWORD PTR DS:[ECX+4]	
0040250F	8A4C32 08	MOV CL,BYTE PTR DS:[EDX+ESI+8]	
00402510	8D0432	LEA EDI,DWORD PTR DS:[EDX+ESI]	
00402511	33FF	XOR EAX,EAX	
00402512	F6C1 06	TEST EAX,EBP	
00402513	75 14	JNZ CrackmeC.0040261F	
00402514	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]	
00402515	8B10	MOV EDX,DWORD PTR DS:[EAX]	
00402516	8BC8	MOV ECX,EAX	
00402517	FF52 2C	CALL DWORD PTR DS:[EDX+2C]	
00402518	83F8 FF	CMP EAX,-1	
00402519	75 05	JNZ SHORT CrackmeC.0040261F	
0040251A	BF 04000000	MOV EDI,4	
0040251B	8B06	MOV EAX,DWORD PTR DS:[ESI]	
0040251C	8B48 04	MOV ECX,DWORD PTR DS:[EAX+4]	
0040251D	8B4F	MOV ECX,DWORD PTR DS:[EAX+7]	

继续运行程序，我们发现我们已经找到了第一个潜在的补丁：



现在，给我们刚才设置 0 标志位的那个跳转打上补丁，这可能有用也可能不起作用。这很难说。如果我们的密码太短会怎样？太长呢？是不同于我们输入的密码的（译者注：大概这个意思，我没弄明白作者啥意思。原文是 **A different password than the one entered**）。这个补丁不是一个非常好的补丁，因为我们真的不知道我们到底做了什么，我们只知道在这种情况下会起作用。

二、深入挖掘

咱们靠近点看看这段代码，用上一章我学到的级别，试试不那么 LAME 的方法。向上滚动到我们打过补丁的那个跳转，就是跳到坏消息的那个，咱们来试试看找出为什么我们没打补丁时它会跳转。注意，我已经在跳转那加了一个注释，这样后面比较容易记住（回想下，选中该行，按一下“；”来添加注释）：

004025B9	33C0	XOR EAX,EAX	CrackmeC.00412F40
004025BA	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:	
004025BB	74 05	JE SHORT CrackmeC.004025C4	
004025BC	1BC0	SBB EAX,EAX	CrackmeC.00412F40
004025BD	83D8 FF	SBB EAX,-1	
004025BE	3BC3	CMP EAX,EBX	
004025BF	75 13	JNZ SHORT CrackmeC.004025D8	### Jump to bad boy
004025C0	3BD5	CMP EDX,EBP	
004025C1	72 0F	JB SHORT CrackmeC.004025D8	
004025C2	33C0	XOR EAX,EAX	CrackmeC.00412F40
004025C3	3BD5	CMP EDX,EBP	
004025C4	0F95C0	SETNE AL	
004025C5	3BC3	CMP EAX,EBX	
004025C6	0F84 CF020000	JE CrackmeC.004028AA	
004025C7	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
004025C8	68 A02F4100	PUSH CrackmeC.00412FA0	
004025C9	E8 A6F4FFFF	CALL CrackmeC.00401A90	

我们通常在注释前加上“###”以示区别，这样的话在将来，当用其他的工具来向我们显示注释的时候，就更容易找到我自己得注释，因为它们比较突出。当然你也可以按自己喜欢的方式做。

现在，咱们就来看看跳转的上面，看能否找到是什么让它跳转的。我在下面已经标记出了跳转上面的第一个区块：

004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	
004025B9	33C0	XOR EAX,EAX	CrackmeC.00412F40
004025BB	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:	
004025BD	74 05	JE SHORT CrackmeC.004025C4	CrackmeC.00412F40
004025BF	1BC0	SBB EAX,EAX	
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	### Jump to bad boy
004025C8	3BD5	CMP EDI,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	CrackmeC.00412F40
004025CE	3BD5	CMP EDI,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
004025DB	68 A02F4100	PUSH CrackmeC.00412FA0	

我们能看到有几个 SBB 指令和一个比较指令。对于我们来说，这里的这段代码并不真正有什么意义，因为我们不知道它是干啥的，所以咱们网上看下一个区，看看咱们能不能开始对它有所了解：

004025A2	3BD5	CMP EDI,EBP	
004025A4	8BCA	MOV ECX,EDX	CrackmeC.0040E408
004025A6	72 02	JB SHORT CrackmeC.004025AA	
004025A8	8BCD	MOV ECX,EBP	
004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	CrackmeC.00412F40
004025B9	33C0	XOR EAX,EAX	
004025BB	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:	
004025BD	74 05	JE SHORT CrackmeC.004025C4	CrackmeC.00412F40
004025BF	1BC0	SBB EAX,-1	
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	### Jump to bad boy
004025C8	3BD5	CMP EDI,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	CrackmeC.00412F40
004025CE	3BD5	CMP EDI,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	ASCII "I'm sorry but that is wrong!"
004025DB	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	E8 A6F4FFFF	CALL CrackmeC.00401A90	

好，这里我们将会到达某个地方。可能你注意到第一个问题的是 REPE CMPS 指令。这是逆向工程的一个红色标志（译者注：原文是 This is a red flag in reverse engineering!，我不知道作者是啥意思，就直译了）！咱们查查 REPE 看看是啥意思：

Intel x86 Instructions

File

Edit

Bookmark

Options

Help

Contents

Index

Back

Print

REP/REPE/REPZ/REPNE/REPNZ—Repeat String Operation Prefix

See also

F2 AF

REPNE SCAS m32

Find EAX, starting at ES:[(E)DI]

Description

Repeats a string instruction the number of times specified in the count register ((E)CX) or until the indicated condition of the ZF flag is no longer met. The REP (repeat), REPE (repeat while equal), REPNE (repeat while not equal), REPZ (repeat while zero), and REPNZ (repeat while not zero) mnemonics are prefixes that can be added to one of the string instructions. The REP prefix can be added to the INS, OUTS, MOVS, LODS, and STOS instructions, and the REPE, REPNE, REPZ, and REPNZ prefixes can be added to the CMPS and SCAS instructions. (The REPZ and REPNZ prefixes are synonymous forms of the REPE and REPNE prefixes, respectively.) The behavior of the REP prefix is undefined when used with non-string instructions.

The REP prefixes apply only to one string instruction at a time. To repeat a block of instructions, use the LOOP instruction or another looping construct.

All of these repeat prefixes cause the associated instruction to be repeated until the count in register (E)CX is decremented to 0 (see the following table). (If the current address-size attribute is 32, register ECX is used as a counter, and if the address-size attribute is 16, the CX register is used.) The REPE, REPNE, REPZ, and REPNZ prefixes also check the state of the ZF flag after each iteration and terminate the repeat loop if the ZF flag is not in the specified state. When both termination conditions are tested, the cause of a repeat termination can be determined either by testing the (E)CX register with a JECXZ instruction or by testing the ZF flag with a JZ, JNZ, and JNE instruction.

Repeat Conditions

Repeat Prefix	Termination Condition 1	Termination Condition 2
REP	ECX=0	None
REPE/REPZ	ECX=0	ZF=0
REPNE/REPZ	ECX=0	ZF=1

When the REPE/REPZ and REPNE/REPZ prefixes are used, the ZF flag does not require initialization because both the CMPS and SCAS instructions affect the ZF flag according to the results of the comparisons they make.

A repeating string operation can be suspended by an exception or interrupt. When this happens, the state of the registers is preserved to allow the string operation to be resumed upon a return from the exception or interrupt handler. The source and destination registers point to the next string elements to be operated on, the EIP register points to the string instruction, and the ECX register has the value it held following the last successful iteration of the instruction. This mechanism allows long string operations to proceed without affecting the interrupt response time of the system.

When a fault occurs during the execution of a CMPS or SCAS instruction that is prefixed with REPE or REPNE, the EFLAGS value is restored to the state prior to the execution of the instruction. Since the SCAS and CMPS instructions do not use EFLAGS as an input, the processor can resume the instruction after the page fault handler.

Use the REP INS and REP OUTS instructions with caution. Not all I/O ports can handle the rate at which these instructions execute.

A REP STOS instruction is the fastest way to initialize a large block of memory.

这个不是非常的清楚，不过如果你对汇编语言稍有经验的话，就知道 **REPXX** 语句像循环一样重复直至 **ECX=0**。**REPXX** 后面的指令，这里是 **CMPS**，就是重复的内容。放在一块的话，这个语句就是“当 0 标志位保持不变时，重复比较两个内存地址的内存，每循环一次就增加一次地址大小”。简而言之，就是“比较两个字符串”。在逆向工程领域，任何时候我们比较两个字符串，红色标志都应该消失。应用程序不会经常这么做，校验序列号/密码/注册码只是多次比较中的一个。咱们在该区块的第一行也就是 **4025B5** 处设置一个 **BP**，并重启应用。输入咱们的密码，然后 **Oilly** 断了下来：

004025A6	72 02	JB SHORT CrackmeC.004025AA	
004025A8	8BCD	MOV ECX,EBP	
004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	
004025B9	33C0	XOR EAX,EAX	CrackmeC.0040E470
004025BB	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:	
004025BD	74 05	JE SHORT CrackmeC.004025C4	CrackmeC.0040E470
004025BF	1BC0	SBB EAX,EAX	
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	
004025C8	3BD5	CMP EDX,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	CrackmeC.0040E470
004025CC	33C0	XOR EAX,EAX	
004025CE	3BD5	CMP EDX,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	
004025DE	68 A02F4100	PUSH CrackmeC.00412FA0	ASCII "I'm sorry but that is wrong!"
004025E0	F8 06F4FFFF	CALL CrackmeC.00401090	

Stack address=0012FE88, (ASCII "12121212")
ESI=32313231

现在，注意第一条指令，LEA ESI, DWORD PTR SS: [ESP+34]，准备将一个栈中得有效地址载入到 ESI 中。SS: 表示堆栈，[ESP+34]表示的是栈中的位置，本例中是 ESP 所指向位置前面的第 34 字节。LEA 指令意思是取地址，而不是取内容。如果我们看那个中间区域（就是蓝色箭头指向的地方），可以发现 SS: [ESP+34] 等于地址 0012FE88，在这个地址存储的是我们的 ASCII 形式的密码。单步步过该行，可以看到 ESI 被设置成我们的密码（当前是在栈上）：

Registers (FPU)		
EAX	0040E470	CrackmeC.0040E470
ECX	00000008	
EDX	00000008	
EBX	00000000	
ESP	0012FE54	
EBP	00000008	
ESI	0012FE88	ASCII "12121212"
EDI	0012FE6C	ASCII "????????"
EIP	004025B9	CrackmeC.004025B9
C 1	ES 0023	32bit 0(FFFFFFFF)
P 1	CS 001B	32bit 0(FFFFFFFF)
D 0	SS 0023	32bit 0(FFFFFFFF)

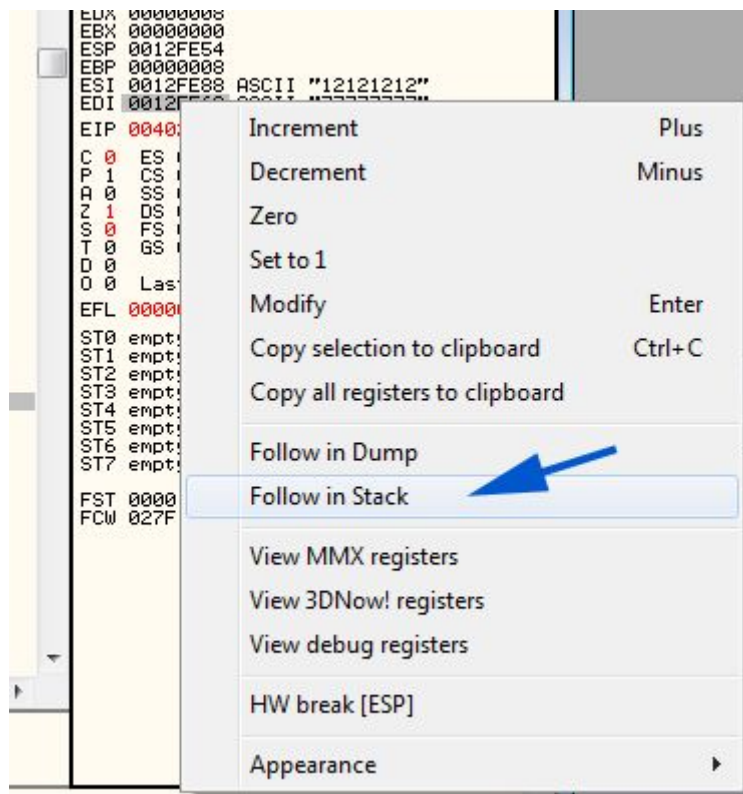
下一条指令将 EAX 设置为 0，然后就是 REPE 指令。本例中，是将存储在 ESI 中的地址的内容与存储在 EDI 中的地址中的内容进行比较：

004025A6	8BCD	MOV ECX,EBP	
004025AA	837C24 48 10	CMP DWORD PTR SS:[ESP+48],10	
004025AF	8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	
004025B3	73 04	JNB SHORT CrackmeC.004025B9	
004025B5	8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	
004025B9	33C0	XOR EAX,EAX	
004025BB	F3:A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:	
004025BD	74 05	JE SHORT CrackmeC.004025C4	
004025BF	1BC0	SBB EAX,EAX	
004025C1	83D8 FF	SBB EAX,-1	
004025C4	3BC3	CMP EAX,EBX	
004025C6	75 13	JNZ SHORT CrackmeC.004025DB	
004025C8	3BD5	CMP EDX,EBP	
004025CA	72 0F	JB SHORT CrackmeC.004025DB	
004025CC	33C0	XOR EAX,EAX	
004025CE	3BD5	CMP EDX,EBP	
004025D0	0F95C0	SETNE AL	
004025D3	3BC3	CMP EAX,EBX	
004025D5	0F84 CF020000	JE CrackmeC.004028AA	
004025D8	68 48E24000	PUSH CrackmeC.0040E248	
004025DE	68 A02F4100	PUSH CrackmeC.00412FA0	
004025E0	F8 06F4FFFF	CALL CrackmeC.00401090	

ECX=00000008 (decimal 8.)
DS:[ESI]=stack [0012FE88]=31 ('1')
ES:[EDI]=stack [0012FE6C]=37 ('7')

ECX 寄存器减一，比较就转到 ESI 和 EDI 的下一个内存位置，当 ECX=0 时循环结束。本例中，如果你往上看，会发现 ECX 被置为 8（就是我们密码的长度），所以该循环会遍历我们密码的 8 个数字，每一次将一个数字与 EDI 中相

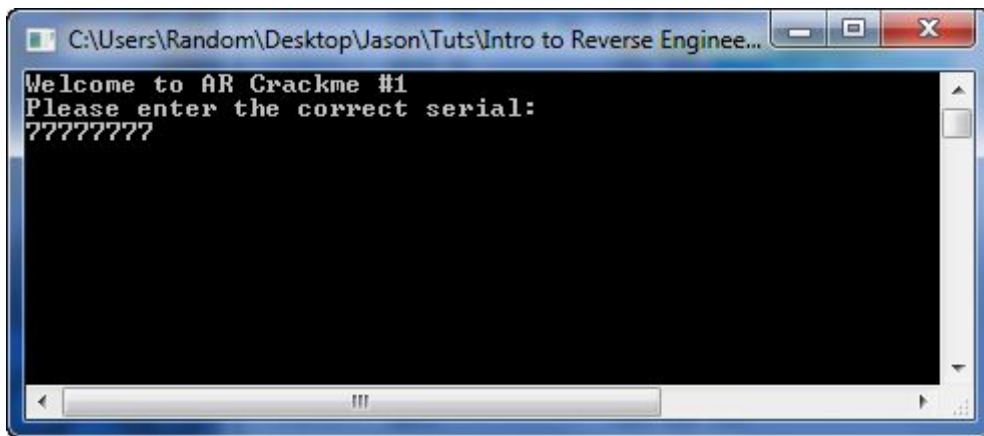
关的数字进行比较。不过，等等...，我们正和谁比较呢？如果我们再看看寄存器窗口，我们会发现 EDI 指向的是堆栈中的一个地址，其中存储着几个 ASCII 字符 7。咱们到堆栈中看看。点击挨着 EDI 的那个地址，在其上右键选择“Follow in stack (堆栈中跟随)”：



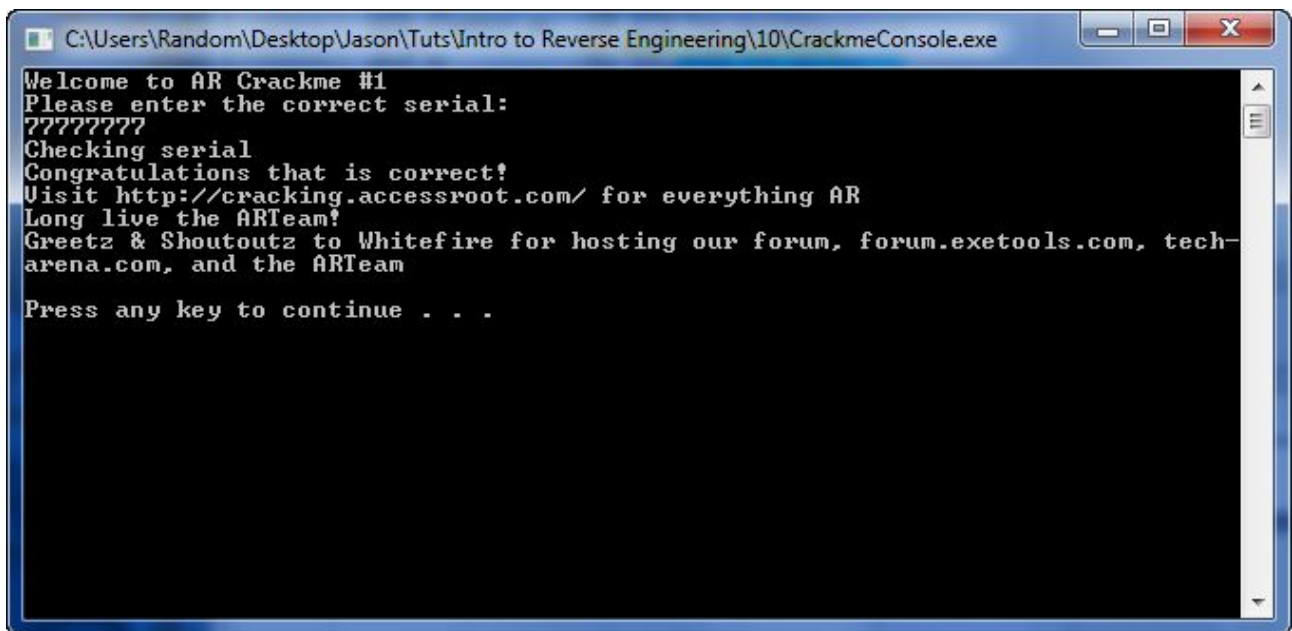
堆栈窗口立即就跳转到相关地址处，也就是 0012FE6C 处。在该地址（我们不能不注意到后面的也是一样）我们看到一串“37”。查查 ASCII 码表就知道 37 就是“7”，就是我们在寄存器窗口中看到的 EDI 寄存器中的内容：

0012FE6C	37373737	
0012FE70	37373737	
0012FE74	00412000	CrackmeC.00412000
0012FE78	0012FF88	
0012FE7C	00000008	
0012FE80	0000000F	
0012FE84	00000000	
0012FE88	32313231	
0012FE8C	32313231	
0012FE90	00406400	CrackmeC.00406400
0012FE94	003A0680	
0012FE98	00000008	
0012FE9C	0000000F	

好吧，不需要像外科医生那样就能够发现我们输入的密码正在和硬编码的全是“7”的字符串进行比较。堆栈中真切的只有 8 个“7”（很走运，我们输入的密码正好和硬编码密码的长度相同😄）。这八个“7”与我们输入的密码一个一个的进行比较。如果所有的 8 个都相等（也就是等于 7），我们就会执行下一个跳转。嗯...，我们输入的密码被拿来和 8 个“7”进行比较。给我的感觉就是密码可能就是八个“7”。咱们来重启应用试试看：



此处应该有掌声...。



我们拿到了😁。所以，在我们通常打补丁的地方的稍远处我们发现了密码，坦白的说这比给一个程序打补丁要好的多，因为我们不知道是真的打上了还是没有。相比 LAME 级别，这就是 NOOB 级别补丁的好处。

三、最后一件事

我只是想举个例子，是分析代码及对代码进行注释。不幸的是，在写教程时，你需要在相当深的层次上理解相关应用。下面是核心区块的图片，我在其中加了注释：

0040257D	. 83C8 04	OR EAX,4	
00402580	> 53	PUSH EBX	
00402581	. 50	PUSH EAX	
00402582	. E8 100A0000	CALL CrackmeC.00402F97	
00402587	> 837C24 2C 10	CMPI DWORD PTR SS:[ESP+2C],10	### EDI = HC password
0040258C	. 8B7C24 18	MOV EDI,DWORD PTR SS:[ESP+18]	### X
00402590	. 73 04	JNB SHORT CrackmeC.00402596	### EDI = HC password
00402592	. 8D7C24 18	LEA EDI,DWORD PTR SS:[ESP+18]	### EDX = Password length
00402596	> 8B5424 44	MOV EDX,DWORD PTR SS:[ESP+44]	### CMP length with zero
0040259A	. 3BD3	CMP EDX,EBX	### EBP = length of HC password
0040259C	. 8B6C24 28	MOV EBP,DWORD PTR SS:[ESP+28]	### Jump if password zero length
004025A0	. 74 26	JE SHORT CrackmeC.004025C8	### Is length < hard coded amount (8 - [esp+28])
004025A2	. 3BD5	CMP EDX,EBP	### ECX = length
004025A4	. 8BCA	MOV ECX,EDX	### X Jmp if our length is < hard coded length (8)
004025A6	. 72 02	JB SHORT CrackmeC.004025AA	### ECX = Length of HC password
004025A8	. 8BCD	MOV ECX,EBP	### CMP 0x0F with 0x10 ???
004025AA	> 837C24 48 10	CMPI DWORD PTR SS:[ESP+48],10	### MOV First digits of entered password into ESI
004025AF	. 8B7424 34	MOV ESI,DWORD PTR SS:[ESP+34]	### X
004025B3	. 73 04	JNB SHORT CrackmeC.004025B9	### ESI = entered password
004025B5	> 8D7424 34	LEA ESI,DWORD PTR SS:[ESP+34]	### EAX = 0
004025B9	. 33C0	XOR EAX,EAX	### CMP H.C. password with entered password
004025BB	. F3A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS	### JMP if they are the same
004025BD	. 74 05	JE SHORT CrackmeC.004025C4	### EAX = FFFFFFFF
004025BF	. 1BC0	SBB EAX,EAX	### No change to EAX
004025C1	. 83D8 FF	CMP EAX,-1	### EAX (FFFFFFFF) == EBX (0)
004025C4	> 3BC3	CMP EAX,EBX	### <> !!!!!
004025C6	> 75 13	JNZ SHORT CrackmeC.004025DB	### <> !!!!!
004025C8	. 3BD5	CMP EDX,EBP	
004025CA	. 72 0F	JB SHORT CrackmeC.004025DB	
004025CC	. 33C0	XOR EAX,EAX	
004025CE	. 3BD5	CMP EDX,EBP	
004025D0	. 0F95C0	SETNE AL	
004025D3	. 3BC3	CMP EAX,EBX	
004025D5	. 0F84 CF020000	JE CrackmeC.004028AA	
004025DB	> 68 48E24000	PUSH CrackmeC.0040E248	
004025DE	. 68 A02F4100	PUSH CrackmeC.00412FA0	
004025E5	. E8 A6F4FFFF	CALL CrackmeC.00401A90	
004025EA	. 83C4 08	ADD ESP,8	
004025ED	. 8BF0	MOV ESI,EAX	
004025EF	. 6A 0A	PUSH 0A	
004025F4	. 8BFC	MOV ECX,ESI	

如你所见，很多都是对应用程序工作方式的理解😄。