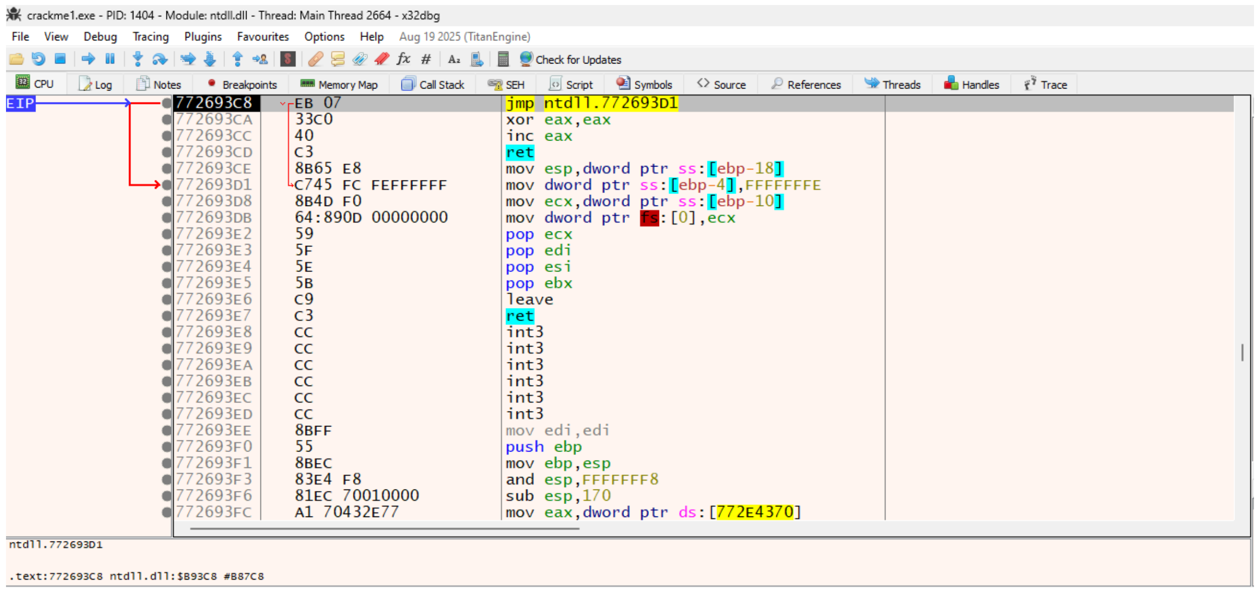
Steps to Patch CrackMe1 Executable

Objective: Learn to bypass password validation in a simple executable by identifying and patching conditional jump instructions using a debugger.

- **Open your debugger** (x32dbg or OllyDbg)
- **Load the sample:**
  - File → Open (or drag and drop CrackMe1.exe)
- **Observe the initial state:**
  - Notice the title bar shows **"Module: ntdll.dll"** instead of the actual program
  - This is because the debugger pauses at the system loader, not the program's entry point

Why this matters: The system loader (ntdll.dll) is Windows' internal code that prepares programs to run. We need to let this complete to reach the actual program code.


crackme1.exe - PID: 1404 - Module: ntdll.dll - Thread: Main Thread 2664 - x32dbg
File View Debug Tracing Plugins Favourites Options Help Aug 19 2025 (TitanEngine)
Check for Updates
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace

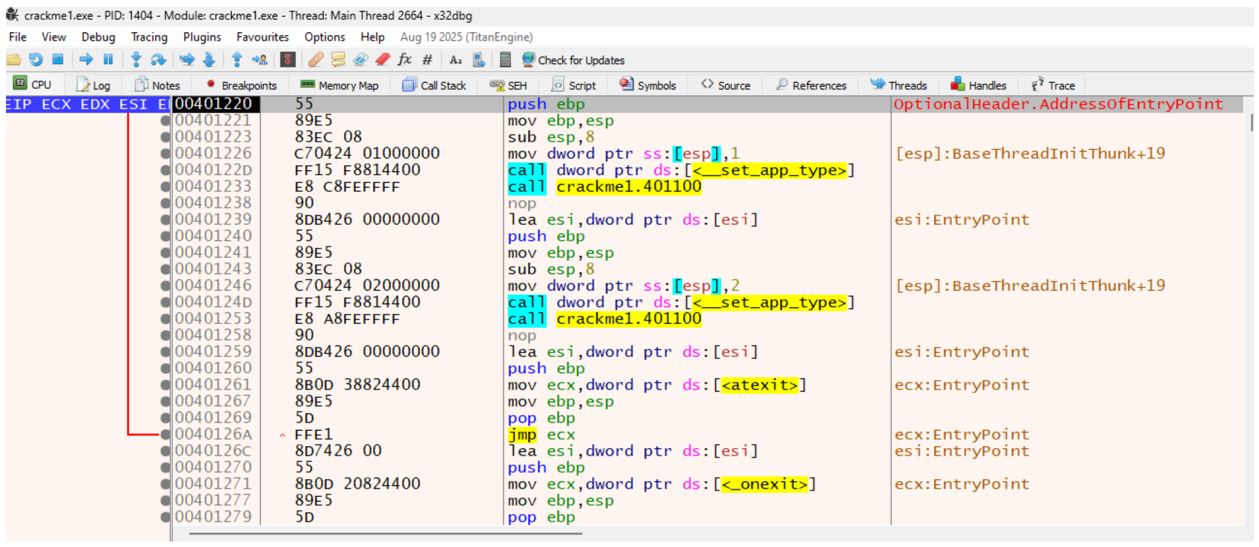
```
EIP  772693C8  EB 07              jmp ntdll.772693D1
     772693CA  33C0               xor eax,eax
     772693CC  40                 inc eax
     772693CD  C3                 ret
     772693CE  8B65 E8            mov esp,dword ptr ss:[ebp-18]
     772693D1  C745 FC FEFFFFFF   mov dword ptr ss:[ebp-4],FFFFFFFE
     772693D8  8B4D F0            mov ecx,dword ptr ss:[ebp-10]
     772693DB  64:890D 00000000   mov dword ptr fs:[0],ecx
     772693E2  59                 pop ecx
     772693E3  5F                 pop edi
     772693E4  5E                 pop esi
     772693E5  5B                 pop ebx
     772693E6  C9                 leave
     772693E7  C3                 ret
     772693E8  CC                 int3
     772693E9  CC                 int3
     772693EA  CC                 int3
     772693EB  CC                 int3
     772693EC  CC                 int3
     772693ED  CC                 int3
     772693EE  8BFF               mov edi,edi
     772693F0  55                 push ebp
     772693F1  8BEC               mov ebp,esp
     772693F3  83E4 F8            and esp,FFFFFFF8
     772693F6  81EC 70010000      sub esp,170
     772693FC  A1 70432E77        mov eax,dword ptr ds:[772E4370]

ntdll.772693D1

.text:772693C8 ntdll.dll:$B93C8 #B87C8
```

Step 2: Navigate to the Program Entry Point

Instructions:

1. **Run to the program's actual entry point:**
   - Press **F9** (or click the blue **Run** button ▶)
   - The debugger will execute through system initialization and pause at the program's first instruction
2. **Verify you're at the correct location:**
   - The title bar should now show **"Module: CrackMe1.exe"**
   - You should see the program's actual code, not system libraries

What just happened: You allowed Windows to finish loading the program, and now you're at the Address of Entry Point (AEP) where the actual program code begins.



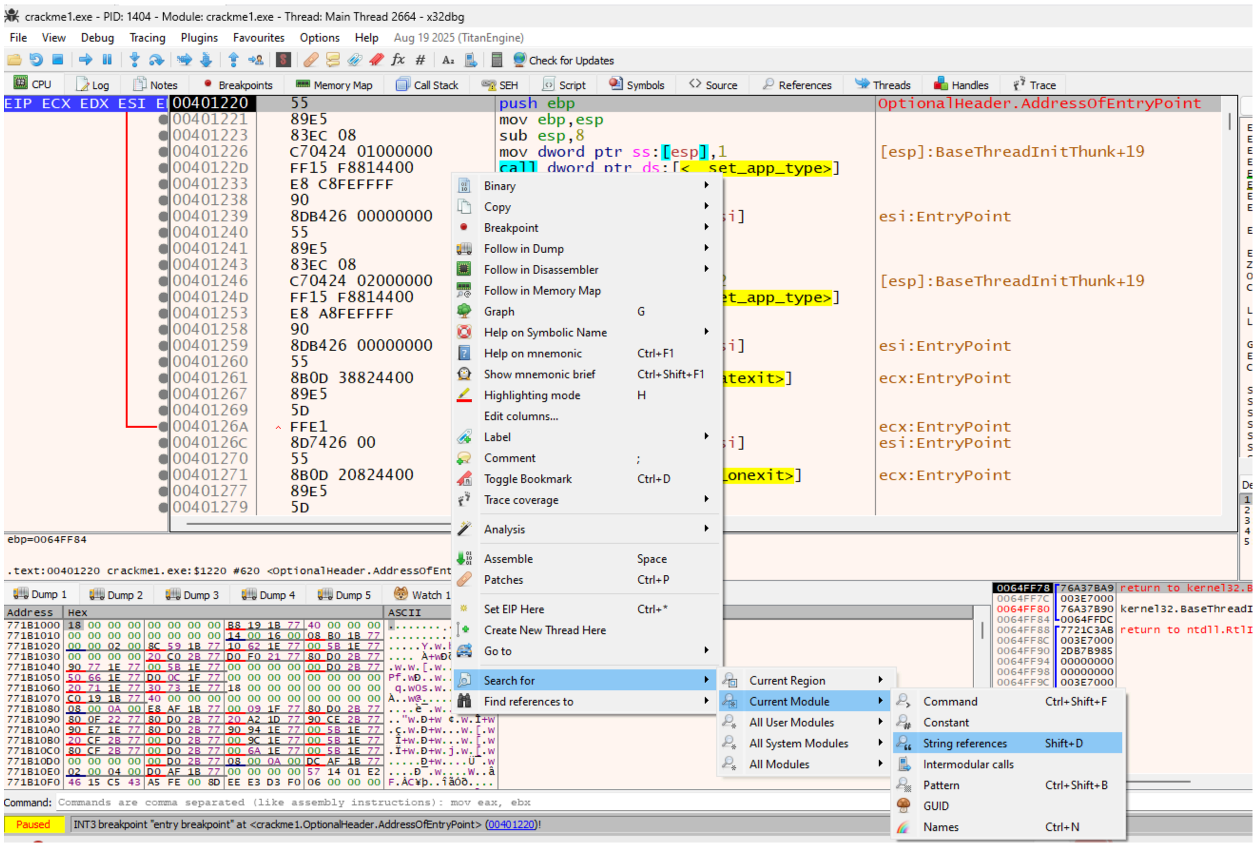
```
00401220    55                  push ebp                                      OptionalHeader.AddressOfEntryPoint
00401221    89E5                mov ebp,esp
00401223    83EC 08             sub esp,8
00401226    C70424 01000000     mov dword ptr ss:[esp],1                      [esp]:BaseThreadInitThunk+19
0040122D    FF15 F8814400       call dword ptr ds:[<__set_app_type>]
00401233    E8 C8FEFFFF         call crackme1.401100
00401238    90                  nop
00401239    8DB426 00000000     lea esi,dword ptr ds:[esi]                    esi:EntryPoint
00401240    55                  push ebp
00401241    89E5                mov ebp,esp
00401243    83EC 08             sub esp,8
00401246    C70424 02000000     mov dword ptr ss:[esp],2                      [esp]:BaseThreadInitThunk+19
0040124D    FF15 F8814400       call dword ptr ds:[<__set_app_type>]
00401253    E8 A8FEFFFF         call crackme1.401100
00401258    90                  nop
00401259    8DB426 00000000     lea esi,dword ptr ds:[esi]                    esi:EntryPoint
00401260    55                  push ebp
00401261    8B0D 38824400       mov ecx,dword ptr ds:[<atexit>]               ecx:EntryPoint
00401267    89E5                mov ebp,esp
00401269    5D                  pop ebp
0040126A  ^ FFE1                jmp ecx                                       ecx:EntryPoint
0040126C    8D7426 00           lea esi,dword ptr ds:[esi]                    esi:EntryPoint
00401270    55                  push ebp
00401271    8B0D 20824400       mov ecx,dword ptr ds:[<_onexit>]              ecx:EntryPoint
00401277    89E5                mov ebp,esp
00401279    5D                  pop ebp
```

Step 3: Search for Interesting Strings

Instructions:

1. **Open the string reference search:**
   - **Right-click** anywhere in the CPU window
   - Select **"Search for"** → **"Current Module"** → **"String references"**
2. **Analyze the strings list:**
   - Look for strings that suggest password validation:
     - "Correct password"
     - "Wrong password"
     - "Access granted"
     - "Access denied"
     - Any other success/failure messages
3. **Select a promising string:**
   - **Double-click** on a string that seems related to the password check
   - This will take you to the code location where that string is referenced

Why strings matter: Password validation routines typically display messages to the user. Finding these messages leads us directly to the validation logic we want to bypass.



```
EIP ECX EDX ESI E 00401220  55                 push ebp                          OptionalHeader.AddressOfEntryPoint
                  00401221  89E5               mov ebp,esp
                  00401223  83EC 08            sub esp,8
                  00401226  C70424 01000000    mov dword ptr ss:[esp],1          [esp]:BaseThreadInitThunk+19
                  0040122D  FF15 F8814400      call dword ptr ds:[<__set_app_type>]
                  00401233  E8 C8FEFFFF                                    i]
                  00401238  90
                  00401239  8DB426 00000000                                      esi:EntryPoint
                  00401240  55
                  00401241  89E5
                  00401243  83EC 08
                  00401246  C70424 02000000                                      [esp]:BaseThreadInitThunk+19
                  0040124D  FF15 F8814400                             t_app_type>]
                  00401253  E8 A8FEFFFF
                  00401258  90
                  00401259  8DB426 00000000                           i]         esi:EntryPoint
                  00401260  55
                  00401261  8B0D 38824400                             atexit>]   ecx:EntryPoint
                  00401267  89E5
                  00401269  5D
                  0040126A  FFE1                                                 ecx:EntryPoint
                  0040126C  8D7426 00                                 i]         esi:EntryPoint
                  00401270  55
                  00401271  8B0D 20824400                             onexit>]   ecx:EntryPoint
                  00401277  89E5
                  00401279  5D
ebp=0064FF84

.text:00401220 crackme1.exe:$1220 #620 <OptionalHeader.AddressOfEnt
```

```
Address   Hex                                              ASCII
771B1000  18 00 00 00 00 00 00 00 B8 19 1B 77 40 00 00 00  .........
771B1010  00 00 00 00 00 00 00 00 14 00 16 00 08 B0 1B 77  .........
771B1020  00 00 02 00 8C 59 1B 77 10 62 1E 77 00 5B 1E 77  .....Y.w.
771B1030  00 00 00 00 20 C0 2B 77 D0 F0 21 77 80 D0 2B 77  .... À+wĐð
771B1040  90 77 1E 77 00 5B 1E 77 00 00 00 00 00 D0 2B 77  .w.w.[.w..
771B1050  50 66 1E 77 D0 0C 1F 77 00 00 00 00 00 00 00 00  Pf.wĐ..w..
771B1060  20 71 1E 77 30 73 1E 77 18 00 00 00 00 00 00 00   q.w0s.w..
771B1070  C0 19 1B 77 40 00 00 00 00 00 00 00 00 00 00 00  À..w@.....
771B1080  08 00 0A 00 E8 AF 1B 77 00 09 1F 77 80 D0 2B 77  ....è¯.w..
771B1090  80 0F 22 77 80 D0 2B 77 20 A2 1D 77 90 CE 2B 77  .."w.Đ+w ¢.w.Î+w
771B10A0  90 E7 1E 77 80 D0 2B 77 90 94 1E 77 00 5B 1E 77  .ç.w.Đ+w...w.[.w
771B10B0  20 CF 2B 77 00 D0 2B 77 00 9C 1E 77 00 5B 1E 77   Ï+w.Đ+w...w.[.w
771B10C0  80 CF 2B 77 00 D0 2B 77 00 6A 1E 77 00 5B 1E 77  .Ï+w.Đ+w.j.w.[.w
771B10D0  00 00 00 00 00 D0 2B 77 08 00 0A 00 DC AF 1B 77  .....Đ+w....Ü¯.w
771B10E0  02 00 04 00 D0 AF 1B 77 00 00 00 00 57 14 01 E2  ....Đ¯.w....W..â
771B10F0  46 15 C5 43 A5 FE 00 8D EE E3 D3 F0 06 00 00 00  F.ÅC¥þ..îãÓð....
```

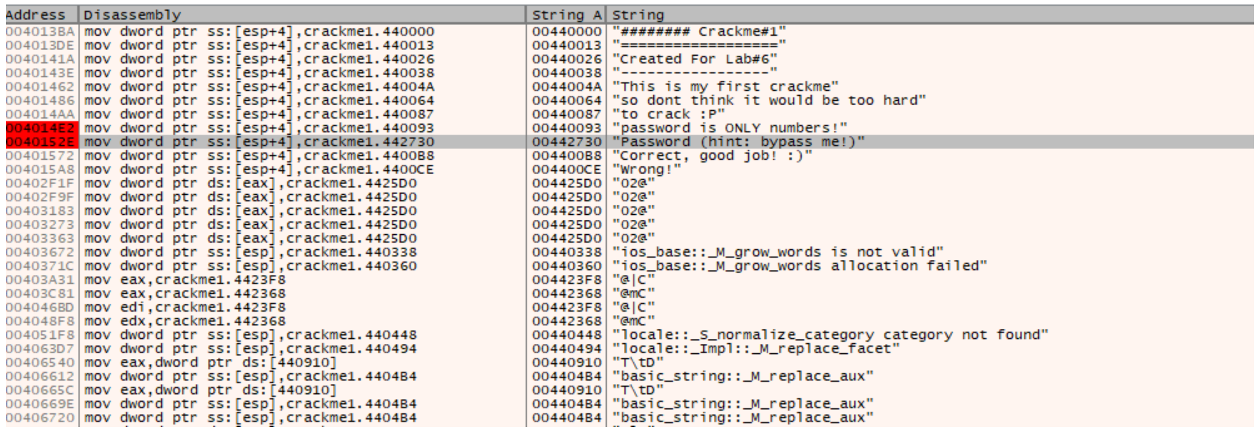
```
0064FF78  76A37BA9  return to kernel32.B
0064FF7C  003E7000
0064FF80  76A37B90  kernel32.BaseThreadI
0064FF84  0064FFDC
0064FF88  7721C3AB  return to ntdll.RtlI
0064FF8C  003E7000
0064FF90  2DB7B985
0064FF94  00000000
0064FF98  00000000
0064FF9C  003E7000
```

```
Command: Commands are comma separated (like assembly instructions): mov eax, ebx
Paused  INT3 breakpoint "entry breakpoint" at <crackme1.OptionalHeader.AddressOfEntryPoint> (00401220)!
```

Once you double-click a string, you'll be taken to the **CPU disassembly view** showing the code that uses that string.

What to look for:

- The code surrounding the string reference
- Comparison instructions (`CMP`, `TEST`)
- Conditional jump instructions (`JE`, `JNE`, `JZ`, `JNZ`)



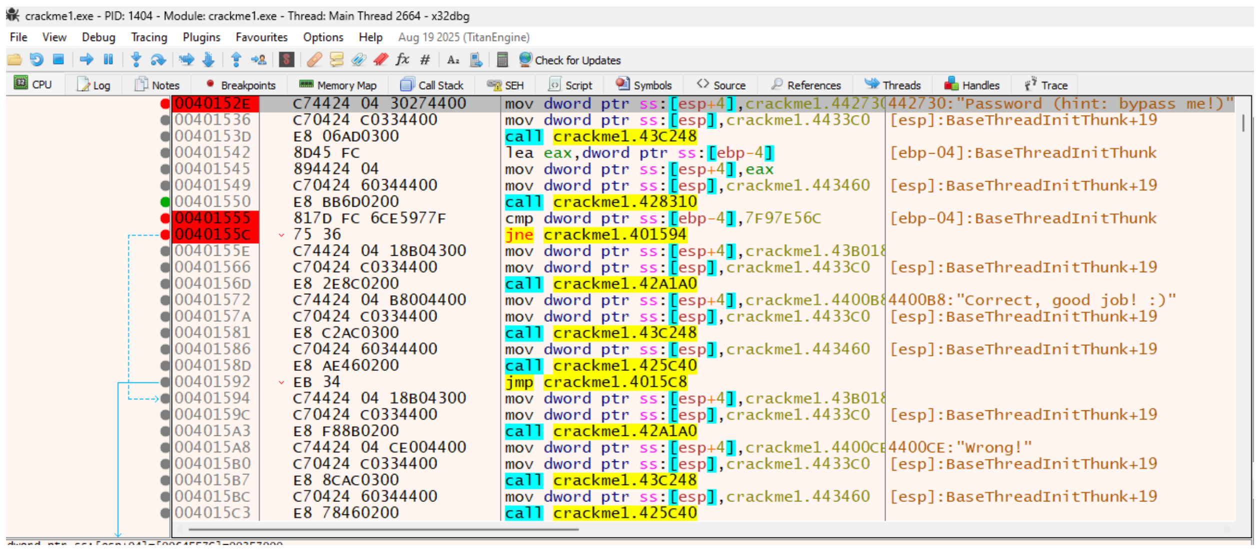
```
Address  Disassembly                                   String A String
004013BA mov dword ptr ss:[esp+4],crackme1.440000      00440000 "######## Crackme#1"
004013DE mov dword ptr ss:[esp+4],crackme1.440013      00440013 "=================="
0040141A mov dword ptr ss:[esp+4],crackme1.440026      00440026 "Created For Lab#6"
0040143E mov dword ptr ss:[esp+4],crackme1.440038      00440038 "-----------------"
00401462 mov dword ptr ss:[esp+4],crackme1.44004A      0044004A "This is my first crackme"
00401486 mov dword ptr ss:[esp+4],crackme1.440064      00440064 "so dont think it would be too hard"
004014AA mov dword ptr ss:[esp+4],crackme1.440087      00440087 "to crack :P"
004014E2 mov dword ptr ss:[esp+4],crackme1.440093      00440093 "password is ONLY numbers!"
0040152E mov dword ptr ss:[esp+4],crackme1.442730      00442730 "Password (hint: bypass me!)"
00401572 mov dword ptr ss:[esp+4],crackme1.4400B8      004400B8 "Correct, good job! :)"
004015A8 mov dword ptr ss:[esp+4],crackme1.4400CE      004400CE "Wrong!"
00402F1F mov dword ptr ds:[eax],crackme1.4425D0        004425D0 "02@"
00402F9F mov dword ptr ds:[eax],crackme1.4425D0        004425D0 "02@"
00403183 mov dword ptr ds:[eax],crackme1.4425D0        004425D0 "02@"
00403273 mov dword ptr ds:[eax],crackme1.4425D0        004425D0 "02@"
00403363 mov dword ptr ds:[eax],crackme1.4425D0        004425D0 "02@"
00403672 mov dword ptr ss:[esp],crackme1.440338        00440338 "ios_base::_M_grow_words is not valid"
0040371C mov dword ptr ss:[esp],crackme1.440360        00440360 "ios_base::_M_grow_words allocation failed"
00403A31 mov eax,crackme1.4423F8                       004423F8 "@|C"
00403C81 mov eax,crackme1.442368                       00442368 "@mC"
004046BD mov edi,crackme1.4423F8                       004423F8 "@|C"
004048F8 mov edx,crackme1.442368                       00442368 "@mC"
004051F8 mov dword ptr ss:[esp],crackme1.440448        00440448 "locale::_S_normalize_category category not found"
004063D7 mov dword ptr ss:[esp],crackme1.440494        00440494 "locale::_Impl::_M_replace_facet"
00406540 mov eax,dword ptr ds:[440910]                 00440910 "T\tD"
00406612 mov dword ptr ss:[esp],crackme1.4404B4        004404B4 "basic_string::_M_replace_aux"
0040665C mov eax,dword ptr ds:[440910]                 00440910 "T\tD"
0040669E mov dword ptr ss:[esp],crackme1.4404B4        004404B4 "basic_string::_M_replace_aux"
00406720 mov dword ptr ss:[esp],crackme1.4404B4        004404B4 "basic_string::_M_replace_aux"
```

Step 5: Set a Breakpoint

Instructions:

1. **Identify the key instruction:**
   - Find the conditional jump (`JNE`, `JZ`, etc.) that determines success or failure
   - This is typically right after a comparison instruction
2. **Set a breakpoint:**
   - **Click on the instruction** you want to break at
   - Press **F2** (or click in the left margin)
   - The instruction line should be highlighted (usually red or a different color)

Why set a breakpoint: This allows you to pause execution at the critical decision point so you can observe the program's behavior before patching.



```
0040152E  C74424 04 30274400  mov dword ptr ss:[esp+4],crackme1.44273( 442730:"Password (hint: bypass me!)"
00401536  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0   [esp]:BaseThreadInitThunk+19
0040153D  E8 06AD0300         call crackme1.43C248
00401542  8D45 FC             lea eax,dword ptr ss:[ebp-4]             [ebp-04]:BaseThreadInitThunk
00401545  894424 04           mov dword ptr ss:[esp+4],eax
00401549  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460   [esp]:BaseThreadInitThunk+19
00401550  E8 BB6D0200         call crackme1.428310
00401555  817D FC 6CE5977F    cmp dword ptr ss:[ebp-4],7F97E56C        [ebp-04]:BaseThreadInitThunk
0040155C  75 36               jne crackme1.401594
0040155E  C74424 04 18B04300  mov dword ptr ss:[esp+4],crackme1.43B01
00401566  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0   [esp]:BaseThreadInitThunk+19
0040156D  E8 2E8C0200         call crackme1.42A1A0
00401572  C74424 04 B8004400  mov dword ptr ss:[esp+4],crackme1.4400B 4400B8:"Correct, good job! :)"
0040157A  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0   [esp]:BaseThreadInitThunk+19
00401581  E8 C2AC0300         call crackme1.43C248
00401586  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460   [esp]:BaseThreadInitThunk+19
0040158D  E8 AE460200         call crackme1.425C40
00401592  EB 34               jmp crackme1.4015C8
00401594  C74424 04 18B04300  mov dword ptr ss:[esp+4],crackme1.43B01
0040159C  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0   [esp]:BaseThreadInitThunk+19
004015A3  E8 F88B0200         call crackme1.42A1A0
004015A8  C74424 04 CE004400  mov dword ptr ss:[esp+4],crackme1.4400C 4400CE:"Wrong!"
004015B0  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0   [esp]:BaseThreadInitThunk+19
004015B7  E8 8CAC0300         call crackme1.43C248
004015BC  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460   [esp]:BaseThreadInitThunk+19
004015C3  E8 78460200         call crackme1.425C40
```

Step 6: Execute Line by Line

Instructions:

1. **Run to the breakpoint:**
   - Press **F9** to run the program
   - The program window will appear asking for a password
   - **Enter any password** (even a wrong one like "test123")
   - Execution will pause at your breakpoint
2. **Step through the code:**
   - Press **F8** (Step Over) or click the **"Step Over"** button
   - Watch the instruction pointer move line by line
   - Observe which path the code takes after the jump

What to observe:

- Before the jump: Check the **flags register** (especially ZF - Zero Flag)
- The jump instruction (e.g., `JNE 00401234`)
- Whether the jump is taken (wrong password path) or not taken (correct password path)

Understanding the Jump Instruction:

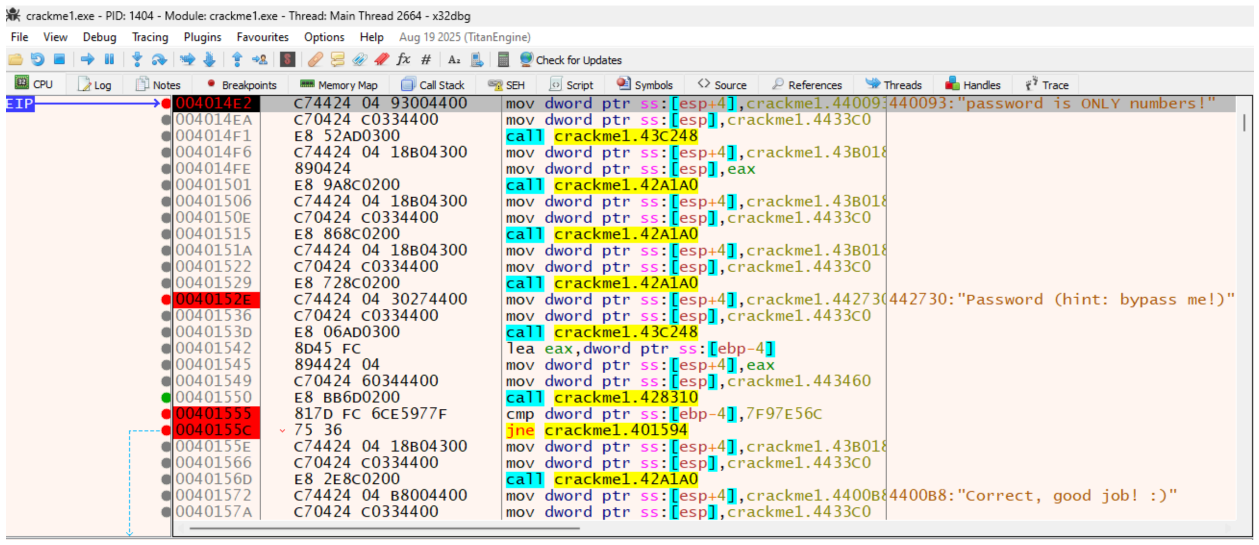
Common jump instructions:

- `JE` / `JZ` - Jump if Equal / Jump if Zero

- JNE / JNZ - Jump if Not Equal / Jump if Not Zero
- JA - Jump if Above
- JB - Jump if Below

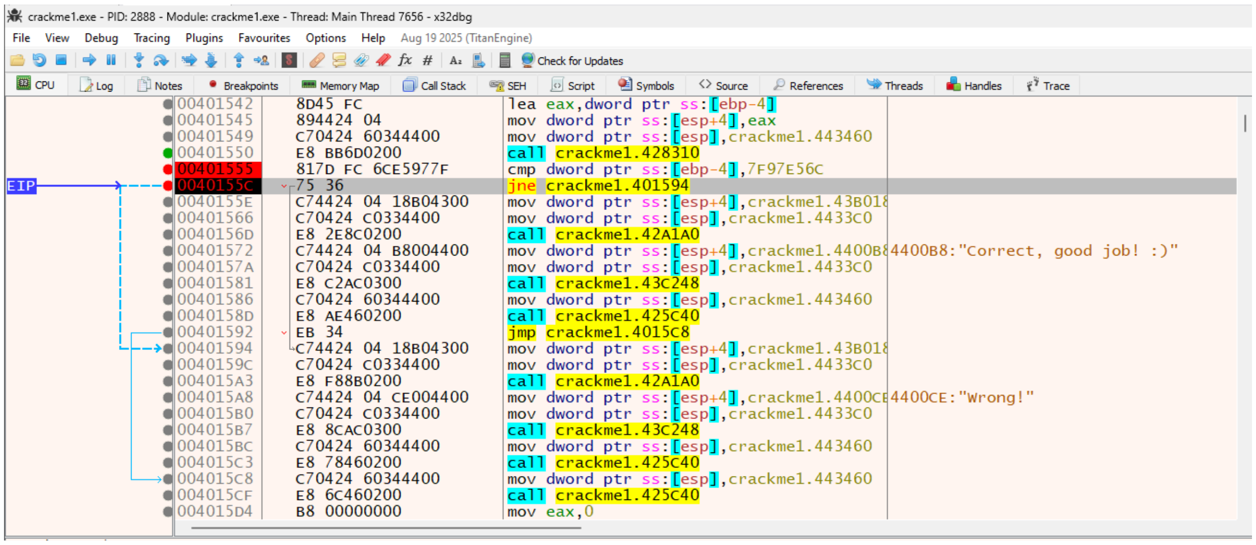
In this case, you'll likely see **JNE** (Jump if Not Equal), which means:

- **If the password is wrong** → jump to error message
- **If the password is correct** → continue to success message


crackme1.exe - PID: 1404 - Module: crackme1.exe - Thread: Main Thread 2664 - x32dbg
File View Debug Tracing Plugins Favourites Options Help Aug 19 2025 (TitanEngine)
Check for Updates
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
EIP

```
004014E2   C74424 04 93004400   mov dword ptr ss:[esp+4],crackme1.44009 440093:"password is ONLY numbers!"
004014EA   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
004014F1   E8 52AD0300          call crackme1.43C248
004014F6   C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B01
004014FE   890424               mov dword ptr ss:[esp],eax
00401501   E8 9A8C0200          call crackme1.42A1A0
00401506   C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B01
0040150E   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
00401515   E8 868C0200          call crackme1.42A1A0
0040151A   C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B01
00401522   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
00401529   E8 728C0200          call crackme1.42A1A0
0040152E   C74424 04 30274400   mov dword ptr ss:[esp+4],crackme1.44273 442730:"Password (hint: bypass me!)"
00401536   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
0040153D   E8 06AD0300          call crackme1.43C248
00401542   8D45 FC              lea eax,dword ptr ss:[ebp-4]
00401545   894424 04            mov dword ptr ss:[esp+4],eax
00401549   C70424 60344400      mov dword ptr ss:[esp],crackme1.443460
00401550   E8 BB6D0200          call crackme1.428310
00401555   817D FC 6CE5977F     cmp dword ptr ss:[ebp-4],7F97E56C
0040155C   75 36                jne crackme1.401594
0040155E   C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B01
00401566   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
0040156D   E8 2E8C0200          call crackme1.42A1A0
00401572   C74424 04 B8004400   mov dword ptr ss:[esp+4],crackme1.4400B 4400B8:"Correct, good job! :)"
0040157A   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
```

We can see the JNE opcode, which will make the jump based on whether the entered password is true or not. This will be determined based on the comparison instruction in the line before the jump


crackme1.exe - PID: 2888 - Module: crackme1.exe - Thread: Main Thread 7656 - x32dbg
File View Debug Tracing Plugins Favourites Options Help Aug 19 2025 (TitanEngine)
Check for Updates
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
EIP

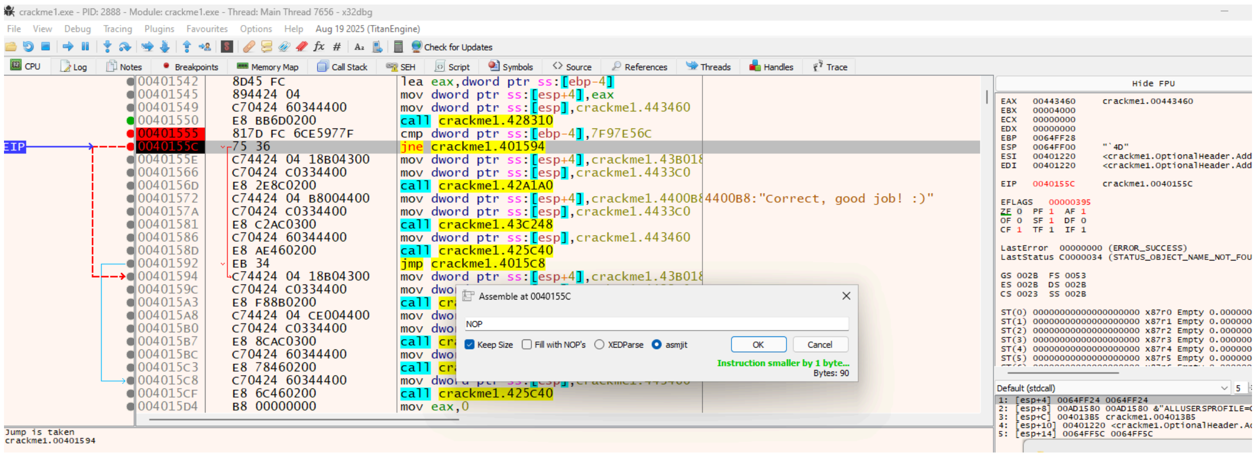
```
00401542  8D45 FC             lea eax,dword ptr ss:[ebp-4]
00401545  894424 04           mov dword ptr ss:[esp+4],eax
00401549  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460
00401550  E8 BB6D0200         call crackme1.428310
00401555  817D FC 6CE5977F    cmp dword ptr ss:[ebp-4],7F97E56C
0040155C  75 36               jne crackme1.401594
0040155E  C74424 04 18B04300  mov dword ptr ss:[esp+4],crackme1.43B018
00401566  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0
0040156D  E8 2E8C0200         call crackme1.42A1A0
00401572  C74424 04 B8004400  mov dword ptr ss:[esp+4],crackme1.4400B84400B8:"Correct, good job! :)"
0040157A  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0
00401581  E8 C2AC0300         call crackme1.43C248
00401586  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460
0040158D  E8 AE460200         call crackme1.425C40
00401592  EB 34               jmp crackme1.4015C8
00401594  C74424 04 18B04300  mov dword ptr ss:[esp+4],crackme1.43B018
0040159C  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0
004015A3  E8 F88B0200         call crackme1.42A1A0
004015A8  C74424 04 CE004400  mov dword ptr ss:[esp+4],crackme1.4400CE4400CE:"Wrong!"
004015B0  C70424 C0334400     mov dword ptr ss:[esp],crackme1.4433C0
004015B7  E8 8CAC0300         call crackme1.43C248
004015BC  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460
004015C3  E8 78460200         call crackme1.425C40
004015C8  C70424 60344400     mov dword ptr ss:[esp],crackme1.443460
004015CF  E8 6C460200         call crackme1.425C40
004015D4  B8 00000000         mov eax,0
```

Step 7: Patch the Jump Instruction

Instructions:

1. **Position at the jump instruction:**
   - Click on the line containing `JNE` (or similar conditional jump)
2. **Open the assembly editor:**
   - Press **Space bar**
   - An **"Assemble at offset"** window will appear
3. **Change the instruction:**
   - **Original:** `JNE xyz` (Jump if Not Equal)
   - **Type:** `NOP` (No Operation)
   - **Check the box: "Keep Size"** (very important!)
   - Click **OK**

Why "Keep Size"? The `JNE` instruction is 2 bytes (opcode `75 XX`). When you replace it with `NOP`, the debugger will use two `NOP` instructions (`90 90`) to maintain the same size. This prevents breaking the program's structure.



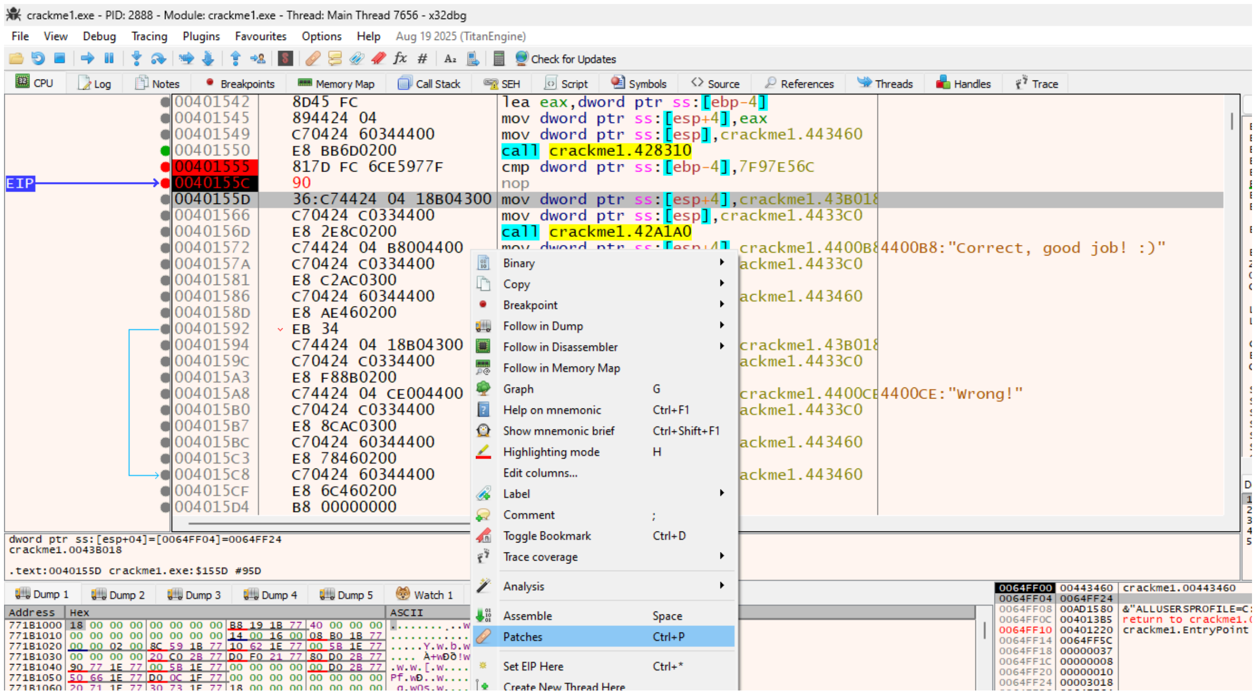
```
crackme1.exe - PID: 2888 - Module: crackme1.exe - Thread: Main Thread 7656 - x32dbg

00401542   8D45 FC              lea eax,dword ptr ss:[ebp-4]
00401545   894424 04            mov dword ptr ss:[esp+4],eax
00401549   C70424 60344400      mov dword ptr ss:[esp],crackme1.443460
00401550   E8 BB6D0200          call crackme1.428310
00401555   817D FC 6CE5977F     cmp dword ptr ss:[ebp-4],7F97E56C
0040155C   75 36                jne crackme1.401594
0040155E   C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B018
00401566   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
0040156D   E8 2E8C0200          call crackme1.42A1A0
00401572   C74424 04 B8004400   mov dword ptr ss:[esp+4],crackme1.4400B84400B8:"Correct, good job! :)"
0040157A   C70424 C0334400      mov dword ptr ss:[esp],crackme1.4433C0
00401581   E8 C2AC0300          call crackme1.43C248
00401586   C70424 60344400      mov dword ptr ss:[esp],crackme1.443460
0040158D   E8 AE460200          call crackme1.425C40
00401592   EB 34                jmp crackme1.4015C8
00401594   C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B018
0040159C   C70424 C0334400      mov dwo
004015A3   E8 F88B0200          call cr
004015A8   C74424 04 CE004400   mov dwo
004015B0   C70424 C0334400      mov dwo
004015B7   E8 8CAC0300          call cr
004015BC   C70424 60344400      mov dwo
004015C3   E8 78460200          call cr
004015C8   C70424 60344400      mov dwo
004015CF   E8 6C460200          call crackme1.425C40
004015D4   B8 00000000          mov eax,0

Jump is taken
crackme1.00401594
```

```
Assemble at 0040155C
NOP
[✓ Keep Size]  [ ] Fill with NOP's  ( ) XEDParse  (• asmjit)   OK   Cancel
Instruction smaller by 1 byte...
Bytes: 90
```

```
EAX  00443460   crackme1.00443460
EBX  00004000
ECX  00000000
EDX  00000000
EBP  0064FF28
ESP  0064FF00   "`4D"
ESI  00401220   <crackme1.OptionalHeader.Add
EDI  00401220   <crackme1.OptionalHeader.Add

EIP  0040155C   crackme1.0040155C

EFLAGS  00000395
ZF 0  PF 1  AF 1
OF 0  SF 1  DF 0
CF 1  TF 1  IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOU

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 0000000000000000000 x87r0 Empty 0.000000
ST(1) 0000000000000000000 x87r1 Empty 0.000000
ST(2) 0000000000000000000 x87r2 Empty 0.000000
ST(3) 0000000000000000000 x87r3 Empty 0.000000
ST(4) 0000000000000000000 x87r4 Empty 0.000000
ST(5) 0000000000000000000 x87r5 Empty 0.000000

Default (stdcall)
1: [esp+4] 0064FF24 0064FF24
2: [esp+8] 00AD1580 00AD1580 &"ALLUSERSPROFILE=C
3: [esp+C] 004013B5 crackme1.004013B5
4: [esp+10] 00401220 <crackme1.OptionalHeader.Ad
5: [esp+14] 0064FF5C 0064FF5C
```

You can see now that the instruction changed to 90, which is NOP (Do nothing)

Right-click and select the Patches option



```
crackme1.exe - PID: 2888 - Module: crackme1.exe - Thread: Main Thread 7656 - x32dbg

00401542   8D45 FC                 lea eax,dword ptr ss:[ebp-4]
00401545   894424 04               mov dword ptr ss:[esp+4],eax
00401549   C70424 60344400         mov dword ptr ss:[esp],crackme1.443460
00401550   E8 BB6D0200             call crackme1.428310
00401555   817D FC 6CE5977F        cmp dword ptr ss:[ebp-4],7F97E56C
0040155C   90                      nop
0040155D   36:C74424 04 18B04300   mov dword ptr ss:[esp+4],crackme1.43B018
00401566   C70424 C0334400         mov dword ptr ss:[esp],crackme1.4433C0
0040156D   E8 2E8C0200             call crackme1.42A1A0
00401572   C74424 04 B8004400      crackme1.4400B84400B8:"Correct, good job! :)"
0040157A   C70424 C0334400         ackme1.4433C0
00401581   E8 C2AC0300
00401586   C70424 60344400         ackme1.443460
0040158D   E8 AE460200
00401592   EB 34
00401594   C74424 04 18B04300      crackme1.43B018
0040159C   C70424 C0334400         ackme1.4433C0
004015A3   E8 F88B0200
004015A8   C74424 04 CE004400      crackme1.4400CE4400CE:"Wrong!"
004015B0   C70424 C0334400         ackme1.4433C0
004015B7   E8 8CAC0300
004015BC   C70424 60344400         ackme1.443460
004015C3   E8 78460200
004015C8   C70424 60344400         ackme1.443460
004015CF   E8 6C460200
004015D4   B8 00000000

dword ptr ss:[esp+04]=[0064FF04]=0064FF24
crackme1.0043B018

.text:0040155D crackme1.exe:$155D #95D
```

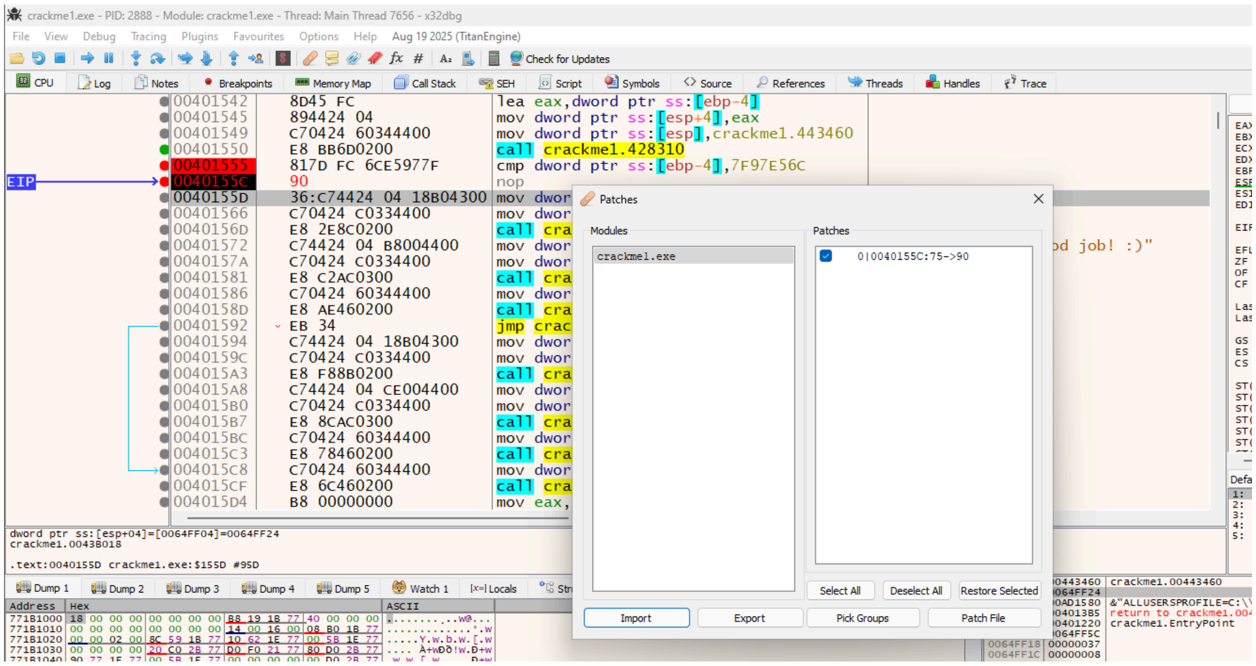
```
Address   Hex                                                ASCII
771B1000  18 00 00 00 00 00 00 00 B8 19 1B 77 40 00 00 00   ..........w
771B1010  00 00 00 00 00 00 00 00 14 00 16 00 08 B0 1B 77   ...........
771B1020  00 00 02 00 8C 59 1B 77 10 62 1E 77 00 5B 1E 77   .....Y.w.b.w
771B1030  00 00 00 00 20 C0 2B 77 D0 F0 21 77 80 D0 2B 77   .... À+wÐð!w
771B1040  90 77 1E 77 00 5B 1E 77 00 00 00 00 00 D0 2B 77   .w.w.[.w....
771B1050  50 66 1E 77 D0 0C 1F 77 00 00 00 00 00 00 00 00   Pf.wÐ..w....

0064FF00  00443460  crackme1.00443460
0064FF04  0064FF24
0064FF08  00AD1580  &"ALLUSERSPROFILE=C:
0064FF0C  004013B5  return to crackme1.
0064FF10  00401220  crackme1.EntryPoint
0064FF14  0064FF5C
0064FF18  00000037
0064FF1C  00000008
0064FF20  00000010
0064FF24  00003018
```

Now the program will **never jump** to the "wrong password" code, regardless of what you enter!

Step 8: Review Your Patches

Instructions:

1. **Open the patches window:**
   - **Right-click** anywhere in the CPU window
   - Select **"Patches"**
2. **Verify the patch:**
   - You should see your modification listed
3. **Select patches to apply:**
   - If you made multiple patches, click **"Select All"**
   - If only one patch, it should already be selected


crackme1.exe - PID: 2888 - Module: crackme1.exe - Thread: Main Thread 7656 - x32dbg
File View Debug Tracing Plugins Favourites Options Help Aug 19 2025 (TitanEngine)
Check for Updates
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
EIP
Patches
Modules
Patches
Select All
Deselect All
Restore Selected
Import
Export
Pick Groups
Patch File
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals

```
00401542   8D45 FC                  lea eax,dword ptr ss:[ebp-4]
00401545   894424 04                mov dword ptr ss:[esp+4],eax
00401549   C70424 60344400          mov dword ptr ss:[esp],crackme1.443460
00401550   E8 BB6D0200              call crackme1.428310
00401555   817D FC 6CE5977F         cmp dword ptr ss:[ebp-4],7F97E56C
0040155C   90                       nop
0040155D   36:C74424 04 18B04300    mov dwor
00401566   C70424 C0334400          mov dwor
0040156D   E8 2E8C0200              call cra
00401572   C74424 04 B8004400       mov dwor
0040157A   C70424 C0334400          mov dwor
00401581   E8 C2AC0300              call cra
00401586   C70424 60344400          mov dwor
0040158D   E8 AE460200              call cra
00401592   EB 34                    jmp crac
00401594   C74424 04 18B04300       mov dwor
0040159C   C70424 C0334400          mov dwor
004015A3   E8 F88B0200              call cra
004015A8   C74424 04 CE004400       mov dwor
004015B0   C70424 C0334400          mov dwor
004015B7   E8 8CAC0300              call cra
004015BC   C70424 60344400          mov dwor
004015C3   E8 78460200              call cra
004015C8   C70424 60344400          mov dwor
004015CF   E8 6C460200              call cra
004015D4   B8 00000000              mov eax,

dword ptr ss:[esp+04]=[0064FF04]=0064FF24
crackme1.0043B018

.text:0040155D crackme1.exe:$155D #95D
```

Modules: `crackme1.exe`

Patches: ✓ `0|0040155C:75->90`

```
od job! :)"

Address  Hex                                              ASCII
771B1000 18 00 00 00 00 00 00 00 B8 19 1B 77 40 00 00 00 ...........w@...
771B1010 00 00 00 00 00 00 00 00 14 00 16 00 08 B0 1B 77 ..............°.w
771B1020 00 00 02 00 8C 59 1B 77 10 62 1E 77 00 5B 1E 77 .....Y.w.b.w.[.w
771B1030 00 00 00 00 20 C0 2B 77 D0 F0 21 77 80 D0 2B 77 .... À+wÐð!w.Ð+w

0443460  crackme1.00443460
064FF24
0AD1580  &"ALLUSERSPROFILE=C:\
04013B5  return to crackme1.00
0401220  crackme1.EntryPoint
064FF5C
0064FF18 00000037
0064FF1C 00000008
```

Step 9: Save the Patched File

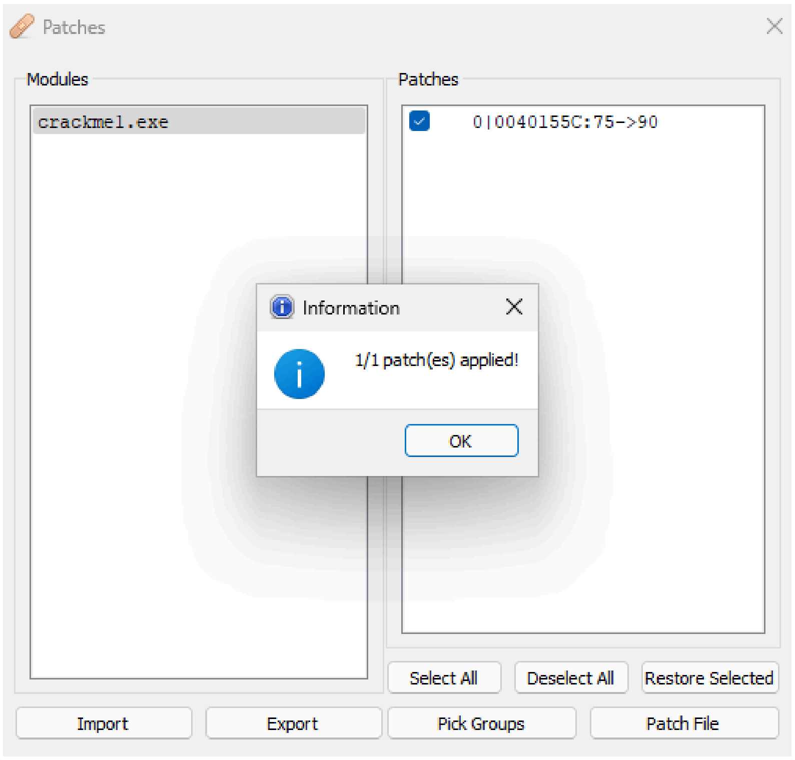
Instructions:

1. **Apply patches to file:**
   - Click the **"Patch File"** button in the Patches window
2. **Choose save location:**
   - A "Save As" dialog will appear
   - **Recommended naming:** Add a suffix to indicate it's patched

        - Original: `CrackMe1.exe`
        - Patched: `CrackMe1_patched.exe` or `CrackMe1_p.exe`
3. **Confirm success:**
    - You should see a message: **"Patching was successful"** or similar
    - Click **OK**

Important: The original file remains unchanged. You now have two versions:

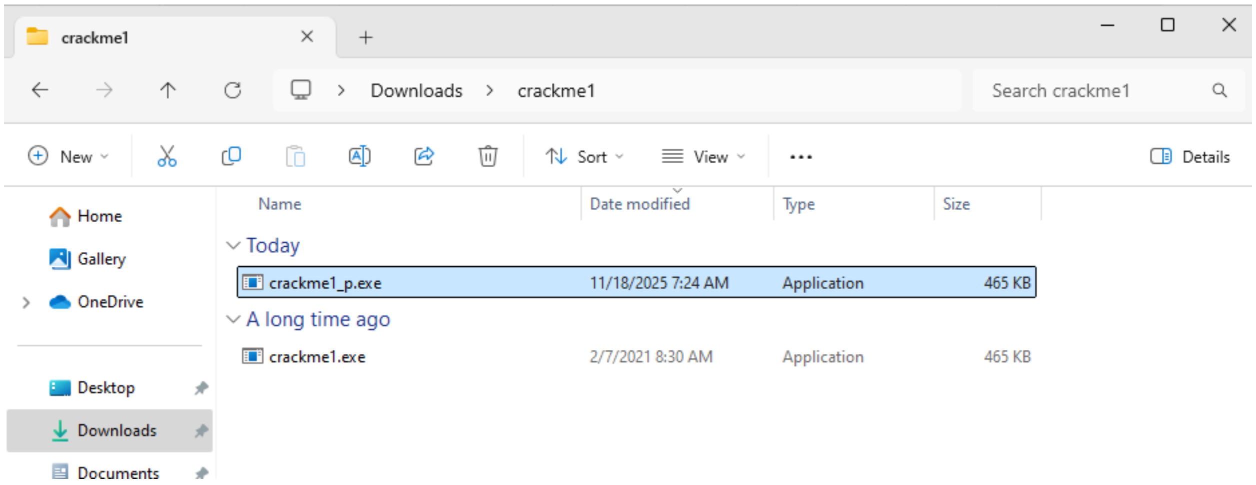
- ✅ `CrackMe1.exe` - Original (still checks password)
- ✅ `CrackMe1_patched.exe` - Modified (bypasses password check)



```
0|0040155C:75->90
```

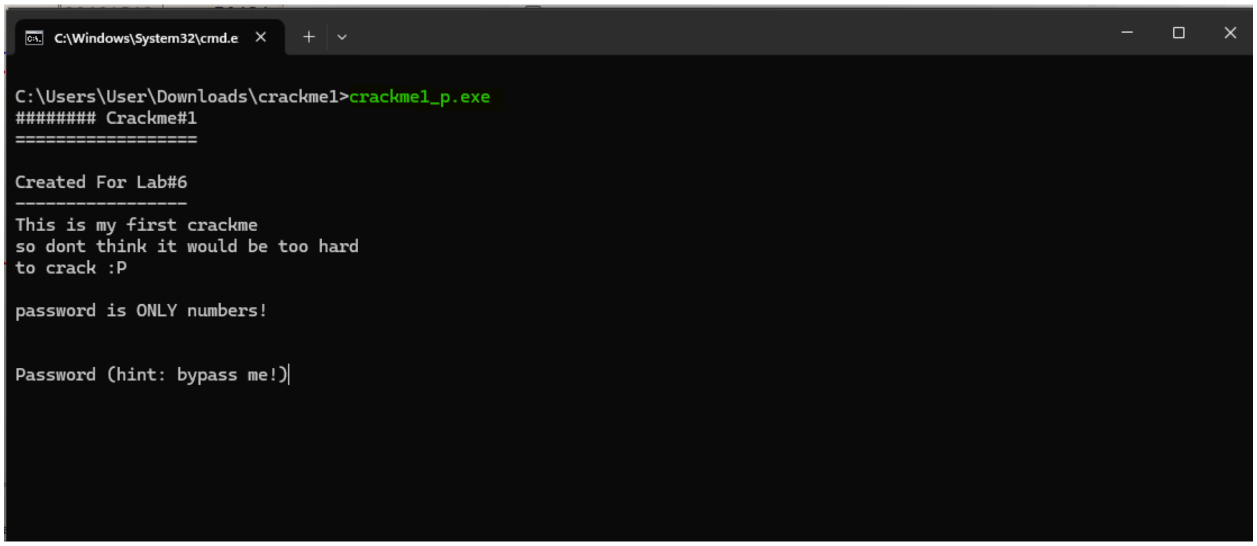
1/1 patch(es) applied!

I saved my file as crackme1_p.exe



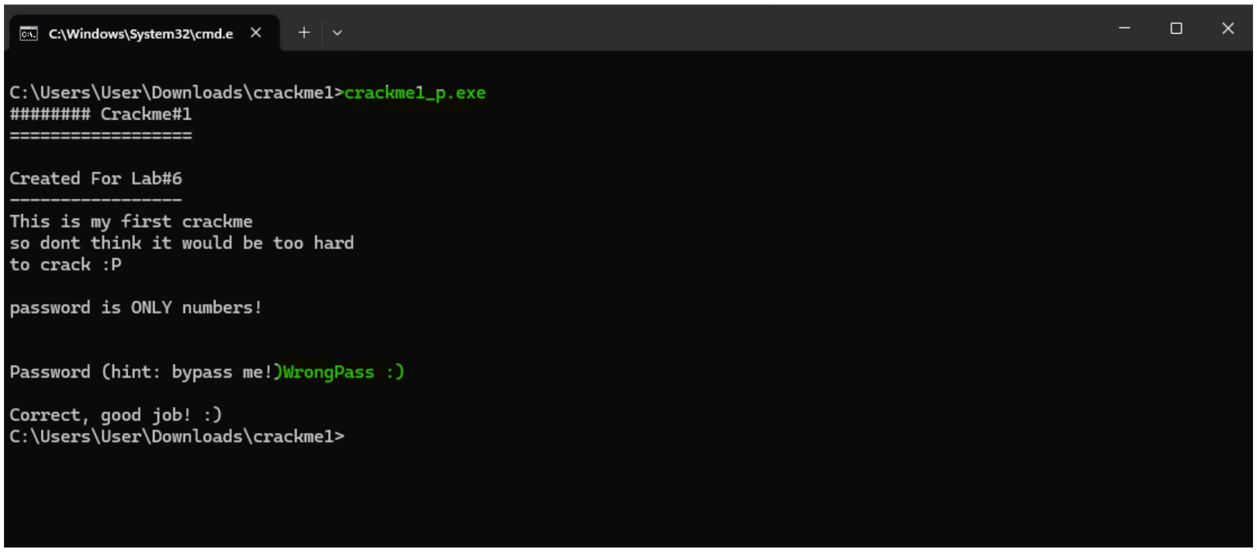
Name	Date modified	Type	Size
Today			
crackme1_p.exe	11/18/2025 7:24 AM	Application	465 KB
A long time ago			
crackme1.exe	2/7/2021 8:30 AM	Application	465 KB

Open a command line and run the patched file



```
C:\Users\User\Downloads\crackme1>crackme1_p.exe
######## Crackme#1
==================

Created For Lab#6
-----------------
This is my first crackme
so dont think it would be too hard
to crack :P

password is ONLY numbers!


Password (hint: bypass me!)
```

Enter any password, even a wrong one, and it should work!



```
C:\Users\User\Downloads\crackme1>crackme1_p.exe
######## Crackme#1
==================

Created For Lab#6
-----------------
This is my first crackme
so dont think it would be too hard
to crack :P

password is ONLY numbers!


Password (hint: bypass me!)WrongPass :)

Correct, good job! :)
C:\Users\User\Downloads\crackme1>
```

Lab Questions & Analysis

Question 1: Understanding the Patch

Q: Why did replacing `JNE` with `NOP` bypass the password check?

A: The `JNE` instruction jumps to the "wrong password" code path when the comparison fails. By replacing it with `NOP` (No Operation), the program never takes that jump and always continues to the "correct password" code path.

Question 2: Alternative Patches

Q: What other ways could you patch this program to achieve the same result?

Possible answers:

- Change `JNE` to `JE` (invert the condition)
- Change `JNE` to `JMP` to always jump to the success path
- Modify the comparison instruction to force a successful comparison
- Patch the string to change "Wrong" to "Correct"

Question 3: Detection and Prevention

Q: How might a programmer prevent this type of patching?

Possible answers:

- Code integrity checks (checksum/hash verification)
- Anti-debugging techniques
- Code obfuscation
- Encrypted/packed executables
- Server-side validation
- Multiple validation checks throughout the code