

whiter0se's Easy to crack

Author: whiter0se

Language: C/C++

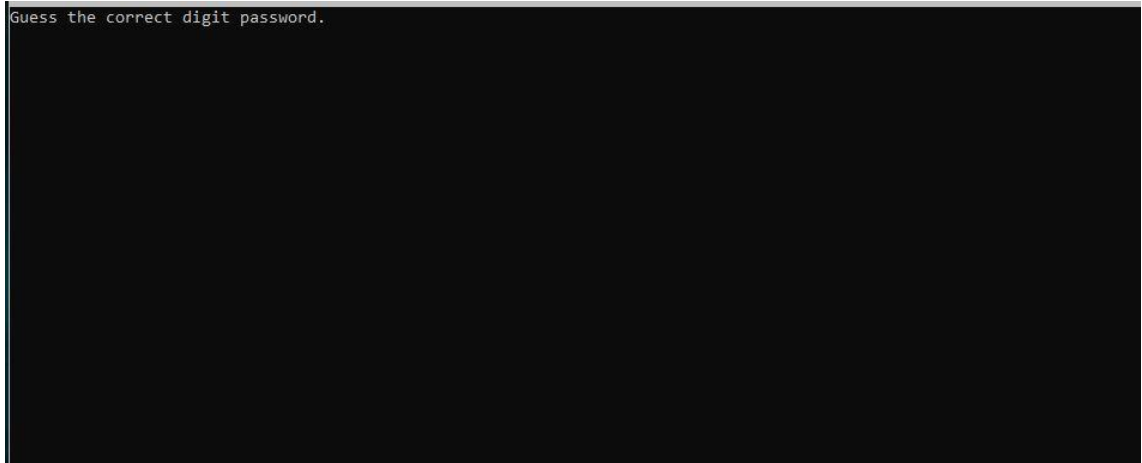
Platform: Windows

Difficulty: 1.0

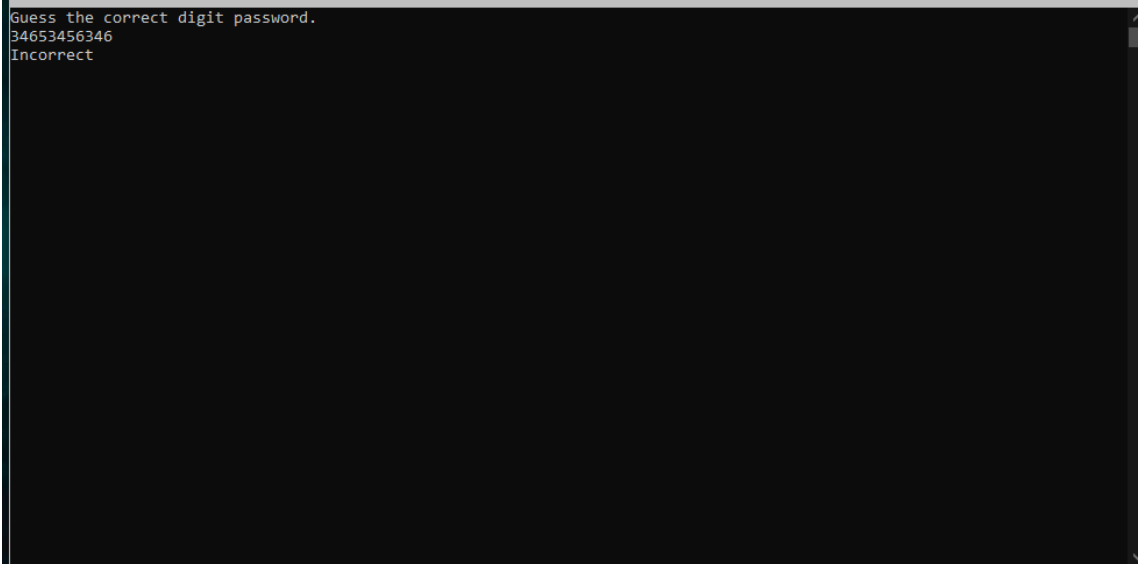
Quality: 2.6

Arch: x86-64

The program initiates with these echo telling us to guess the correct password.

A screenshot of a terminal window with a black background. The text "Guess the correct digit password." is displayed in white at the top left.

```
Guess the correct digit password.
```

A screenshot of a terminal window with a black background. The text "Guess the correct digit password." is displayed in white at the top left. Below it, the password "34653456346" has been entered, and the response "Incorrect" is shown on the next line.

```
Guess the correct digit password.  
34653456346  
Incorrect
```

After an initial attempt, I discovered that the password was incorrect. I then launched x64dbg, a tool that allows users to view, analyze, and even manipulate the machine code of '.exe' files.

After opening and attaching the file at x64dbg I thought of what would be the solution and what's the optimal way to gain access to it.

My Solution: Change the memory address of the jump given after trying the password. Jump immediatly to the correct password message memory address.

After going to all modules to search for strings I already found the 3 only strings that exist.

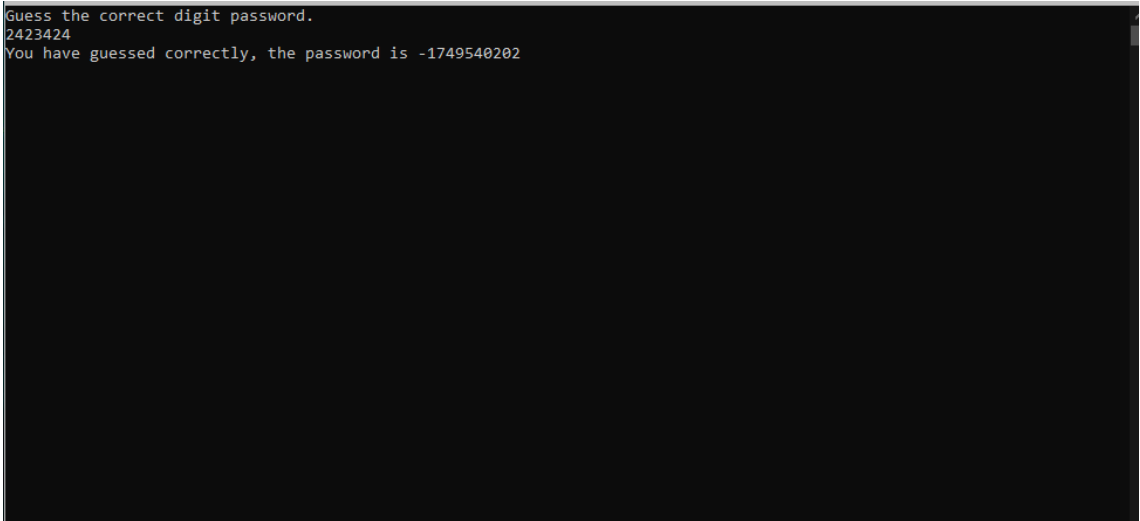
CPU										Log	Notes	Breakpoints	Memory Map	Call Stack	SEH	Script	Symbols	Source	References	Threads	Handles	Trace																	
All Modules (Strings)										All Modules (Strings)																													
Address										Disassembly										String Address										String									
00007FF7F3C710F3										lea rcx, qword ptr ds:[7FF7F3C72270]										00007FF7F3C72270										"Guess the correct digit password."									
00007FF7F3C71105										lea rcx, qword ptr ds:[7FF7F3C72294]										00007FF7F3C72294										"%d"									
00007FF7F3C71120										lea rcx, qword ptr ds:[7FF7F3C72298]										00007FF7F3C72298										"You have guessed correctly, the password is %d"									
00007FF7F3C7112E										lea rcx, qword ptr ds:[7FF7F3C722C8]										00007FF7F3C722C8										"Incorrect"									
00007FFA798B4400										lea rax, qword ptr ds:[7FA798B44000]										00007FFA798B4400										"Bad exception"									
00007FFA798A4255										lea rax, qword ptr ds:[7FA798B44000]										00007FFA798B4400										"Unknown exception"									
00007FFA798A4CEA										lea rdx, qword ptr ds:[7FA798B44000]										00007FFA798B4408										"Access violation - no RTTI data!"									
00007FFA798A4E2F										lea rdx, qword ptr ds:[7FA798B44000]										00007FFA798B4408										"Access violation - no RTTI data!"									
00007FFA798A4E5E										lea rdx, qword ptr ds:[7FA798B44560]										00007FFA798B4560										"Bad dynamic_cast!"									
00007FFA798A4E65										lea rdx, qword ptr ds:[7FA798B44560]										00007FFA798B4560										"Attempted a typeid of nullptr pointer!"									
00007FFA798A4F0B										lea rdx, qword ptr ds:[7FA798B44560]										00007FFA798B4560										"Bad read pointer - no RTTI data!"									
00007FFA798A5749										lea rdx, qword ptr ds:[7FA798B44560]										00007FFA798B44E8										"Access violation - no RTTI data!"									
00007FFA798A57FD										lea r9, qword ptr ds:[7FA798B44640]										00007FFA798B4640										"api-ms-"									
00007FFA798A580D										lea rdx, qword ptr ds:[7FA798B44640]										00007FFA798B4640										"FIsAlloc"									
00007FFA798A5844										lea r9, qword ptr ds:[7FA798B44658]										00007FFA798B4658										"FIsFree"									
00007FFA798A5857										lea rdx, qword ptr ds:[7FA798B44658]										00007FFA798B4658										"FIsFree"									
00007FFA798A588C										lea r9, qword ptr ds:[7FA798B44658]										00007FFA798B4658										"FIsGetValue"									
00007FFA798A589F										lea rdx, qword ptr ds:[7FA798B44658]										00007FFA798B4668										"FIsGetValue"									
00007FFA798A58D9										lea r9, qword ptr ds:[7FA798B44680]										00007FFA798B4680										"FIsSetValue"									
00007FFA798A58E2										lea rdx, qword ptr ds:[7FA798B44680]										00007FFA798B4680										"FIsSetValue"									
00007FFA798A5932										lea r9, qword ptr ds:[7FA798B44698]										00007FFA798B4698										"InitializeCriticalSectionEx"									
00007FFA798A5941										lea rdx, qword ptr ds:[7FA798B44698]										00007FFA798B4698										"InitializeCriticalSectionEx"									
00007FFA798A6322										lea rdx, qword ptr ds:[7FA798B446C0]										00007FFA798B46C0										"-based"									
00007FFA798A65A1										lea rax, qword ptr ds:[7FA798B55C00]										00007FFA798B55C0										";"									
00007FFA798A65DC										lea rcx, qword ptr ds:[7FA798B55C00]										00007FFA798B55D0										";"									

I quickly double clicked on the first string and found more about the program logic.

00007FF7F3C710E1	48:894424 28	mov qword ptr ss:[rsp+28],rax	
00007FF7F3C710F3	48:8000 76110000	lea rcx,qword ptr ds:[7FF7F3C72270]	00007FF7F3C72270:"Guess the correct digit password."
00007FF7F3C71101	FF15 01000000	call qword ptr ds:[puts+8]	
00007FF7F3C71105	48:805424 28	lea rdx,qword ptr ss:[rsp+20]	
00007FF7F3C71105	48:8000 88110000	lea rcx,qword ptr ds:[7FF7F3C72294]	00007FF7F3C72294:"%d"
00007FF7F3C7110C	E8 6FFFFF	call consoleapplication1.7FF7F3C71080	
00007FF7F3C71111	817C24 20 3EAB0700	cmp qword ptr ss:[rsp+20],7AB3E	
00007FF7F3C71119	75 13	jne consoleapplication1.7FF7F3C7112E	
00007FF7F3C7111B	BA 3EAB0700	mov edx,7AB3E	
00007FF7F3C71120	48:8000 71110000	lea rcx,qword ptr ds:[7FF7F3C72298]	
00007FF7F3C71127	E8 FFFFFF	call consoleapplication1.7FF7F3C71020	00007FF7F3C72298:"You have guessed correctly, the password is %d"
00007FF7F3C7112C	EB 0D	jmp consoleapplication1.7FF7F3C7113B	
00007FF7F3C7112E	48:8000 93110000	lea rcx,qword ptr ds:[7FF7F3C722C8]	00007FF7F3C722C8:"Incorrect"
00007FF7F3C71135	FF15 51000000	call qword ptr ds:[puts+8]	
00007FF7F3C71138	B9 D0070000	mov ecx,700	
00007FF7F3C71140	FF15 BA0E0000	call qword ptr ds:[<Sleep>]	
00007FF7F3C71146	33C0	xor eax,eax	
00007FF7F3C71148	48:8B84C4 28	mov rcx,qword ptr ss:[rsp+28]	
48:33CC		xor rcx,rcx	
00007FF7F3C71150	E8 18000000	call consoleapplication1.7FF7F3C71170	
00007FF7F3C71155	48:83C4 38	add rsp,38	
00007FF7F3C71159	C3	ret	
00007FF7F3C7115A	CC	int3	
00007FF7F3C7115B	CC	int3	

The program essentially asks for an integer input and checks if it matches the correct password. If the input is correct, it prints a success message; if not, the program executes a "jump not equal" (JNE) instruction to a memory address that prints an incorrect password message.

What I did was change the memory address of this JNE instruction to the memory address of the correct password message. I patched the file, and this is how it looks now.

A screenshot of a terminal window with a black background and white text. The text shows a prompt to guess a password, followed by the input '2423424', and a response indicating the password is correct and displaying a long hexadecimal string.

```
Guess the correct digit password.  
2423424  
You have guessed correctly, the password is -1749540202
```

Now everytime I input any password it will tell me the correct one and that I guessed it.