lendigs - C++ idk — Reverse Engineering Write-up

Challenge link: https://crackmes.one/crackme/68ff68d62d267f28f69b78e3
Author: *lendigs*
Write-up by: *SenorGPT*
Tools used: *CFF Explorer*, *x64dbg*

Platform	Difficulty	Quality	Arch	Language
Windows	1.8	3.2	x86-64	C/C++

Cover Snapshot

> **Status:** Complete
> **Goal:** Document a clean path from initial recon → locating key-check logic → validation/reversal strategy

---

---

1. Executive Summary

This *crackme* is a straightforward but well-structured console challenge that validates a user-supplied password. The binary contains no packing, obfuscation, or anti-debug measures, allowing direct observation of stack initialization, immediate constant loading, C++ std::string operations, and multiple helper functions leading to the extraction of a hard-coded password string.

During analysis, a suspicious-looking constant array placed on the stack (`a#l67'gdb`) initially appeared to be the key, but further tracing revealed a deeper string transformation path. Following the execution flow of the password-handling functions ultimately exposed the correct computed value.

▶ Click to reveal password

Stepping through the internal comparison function confirmed this string as the exact value checked against user input. The binary accepted this value, completing the challenge successfully.

---

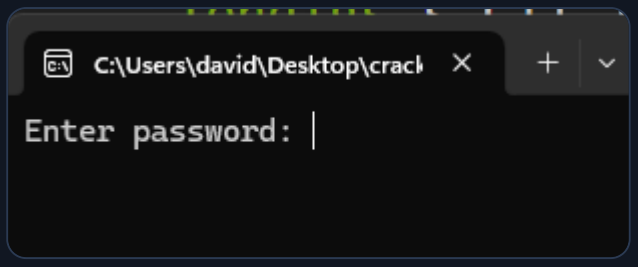
2. Target Overview

2.1 UI / Behaviour

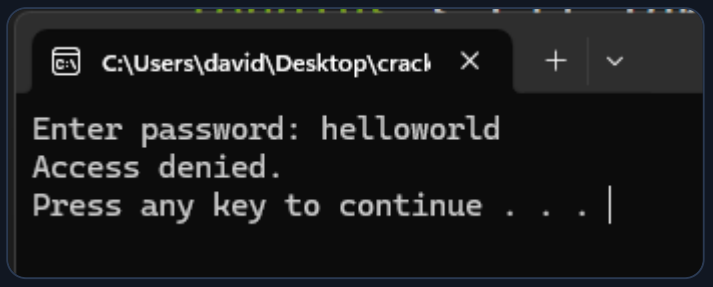
- Inputs: **Accepts user input for a password.**
- Outputs: *"Enter password: ", "Access denied.", "Access granted."*

2.2 Screens

Start-up



Failure case



---

3. Tooling & Environment

- OS: *Windows 11*
- Debugger: *x64dbg*
- Static tools: *CFF Explorer*

---

4. Static Recon

4.1 File & Headers

crackmes.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.textbss	00010000	00001000	00000000	00000000	00000000	00000000	0000	0000	E00000A0
.text	000105AB	00011000	00010600	00000400	00000000	00000000	0000	0000	60000020
.rdata	00004D3C	00022000	00004E00	00010A00	00000000	00000000	0000	0000	40000040
.data	00000CD0	00027000	00000600	00015800	00000000	00000000	0000	0000	C0000040
.pdata	0000297C	00028000	00002A00	00015E00	00000000	00000000	0000	0000	40000040
.idata	00001BB1	0002B000	00001C00	00018800	00000000	00000000	0000	0000	40000040
.msvcjmc	00000190	0002D000	00000200	0001A400	00000000	00000000	0000	0000	C0000040
.00cfg	00000175	0002E000	00000200	0001A600	00000000	00000000	0000	0000	40000040
.rsrc	0000043C	0002F000	00000600	0001A800	00000000	00000000	0000	0000	40000040
.reloc	00000353	00030000	00000400	0001AE00	00000000	00000000	0000	0000	42000040

The *PE* section table looks like a standard 64-bit *MSVC* debug build with no signs of packing or obfuscation. Code lives in `.textbss` / `.text` (readable + executable), constants and string literals in `.rdata`, and writable globals in `.data`. `.pdata` and `.reloc` provide normal *x64* exception/unwind and relocation info, while `.idata` holds the import table for the *CRT* and *Windows APIs*. Extra sections like `.msvcjmc` and `.00cfg` come from Visual Studio's debug/runtime features and *Control Flow Guard* configuration. Overall the layout is clean, contiguous, and exactly what you'd expect from an uncomplicated console *crackme* compiled in a debug configuration.

4.2 Imports / Exports

crackmes.exe

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
MSVCP140D.dll	32	0002B7B0	00000000	00000000	0002C2CA	0002B150
VCRUNTIME140D.dll	13	0002B938	00000000	00000000	0002C43E	0002B2D8
VCRUNTIME140_1D....	1	0002BA08	00000000	00000000	0002C450	0002B3A8
ucrtbased.dll	41	0002BA68	00000000	00000000	0002C72E	0002B408
KERNEL32.dll	26	0002B660	00000000	00000000	0002C954	0002B000

4.2.1 MSVCP140D.dll

OFTs	FTs (IAT)	Hint	Name
Qword	Qword	Word	szAnsi
000000000002C212	000000000002C212	01F6	?_Id_cnt@id@locale@std@@0HA
000000000002C230	000000000002C230	03D7	?id@?$ctype@D@std@@2V0locale@2@A
000000000002C1D0	000000000002C1D0	0221	?_Ipfx@?$basic_istream@DU?$char_traits@D@std@@@std@@QEAA_N_N@Z
000000000002C28E	000000000002C28E	02BC	?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A
000000000002C2D8	000000000002C2D8	0293	?_Xbad_alloc@std@@YAXXZ
000000000002C18C	000000000002C18C	0370	?flush@?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV12@XZ
000000000002C148	000000000002C148	0469	?put@?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV12@D@Z
000000000002C0F4	000000000002C0F4	010C	??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z
000000000002C0B6	000000000002C0B6	024C	?_Osfx@?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAXXZ
000000000002C07A	000000000002C07A	053C	?widen@?$basic_ios@DU?$char_traits@D@std@@@std@@QEBADD@Z
000000000002C040	000000000002C040	0369	?fill@?$basic_ios@DU?$char_traits@D@std@@@std@@QEBADXZ
000000000002BFD6	000000000002BFD6	048F	?rdbuf@?$basic_ios@DU?$char_traits@D@std@@@std@@QEBAPEAV?$basic_streambuf@DU?$char_traits@D@std@@@2@XZ
000000000002BF70	000000000002BF70	0512	?tie@?$basic_ios@DU?$char_traits@D@std@@@std@@QEBAPEAV?$basic_ostream@DU?$char_traits@D@std@@@2@XZ
000000000002BF30	000000000002BF30	04CD	?setstate@?$basic_ios@DU?$char_traits@D@std@@@std@@QEAAXH_N@Z
000000000002BEE8	000000000002BEE8	04E9	?sputn@?$basic_streambuf@DU?$char_traits@D@std@@@std@@QEAA_JPEBD_J@Z
000000000002BEA6	000000000002BEA6	04E6	?sputc@?$basic_streambuf@DU?$char_traits@D@std@@@std@@QEAAHD@Z
000000000002BE64	000000000002BE64	04E0	?snextc@?$basic_streambuf@DU?$char_traits@D@std@@@std@@QEAAHXZ
000000000002BE24	000000000002BE24	04D7	?sgetc@?$basic_streambuf@DU?$char_traits@D@std@@@std@@QEAAHXZ
000000000002BDF8	000000000002BDF8	03CB	?getloc@ios_base@std@@QEBA?AVlocale@2@XZ
000000000002BDD6	000000000002BDD6	0545	?width@ios_base@std@@QEAA_J_J@Z
000000000002BDB6	000000000002BDB6	0546	?width@ios_base@std@@QEBA_JXZ
000000000002BD96	000000000002BD96	036F	?flags@ios_base@std@@QEBAHXZ
000000000002BD76	000000000002BD76	03CD	?good@ios_base@std@@QEBA_NXZ
000000000002BD38	000000000002BD38	01BE	?_Getcat@?$ctype@D@std@@SA_KPEAPEBVfacet@locale@2@PEBV42@@Z
000000000002BD18	000000000002BD18	0417	?is@?$ctype@D@std@@QEBA_NFD@Z
000000000002BCE4	000000000002BCE4	01DD	?_Getgloballocale@locale@std@@CAPEAV_Locimp@12@XZ
000000000002BCC2	000000000002BCC2	0526	?uncaught_exception@std@@YA_NXZ
000000000002BCA0	000000000002BCA0	0297	?_Xout_of_range@std@@YAXPEBD@Z
000000000002BC7E	000000000002BC7E	0296	?_Xlength_error@std@@YAXPEBD@Z
000000000002BC64	000000000002BC64	00A5	??1_Lockit@std@@QEAA@XZ
000000000002C254	000000000002C254	02AB	?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A
000000000002BC48	000000000002BC48	006D	??0_Lockit@std@@QEAA@H@Z

This import table is full of demangled *C++* standard-library symbols (`std::basic_istream`, `std::basic_ostream`, streambuf/ios_base, locale and exception helpers, etc), which confirms this *crackme* is a *C++* console program built with the *MSVC* debug runtime. The heavy use of iostream and locale/stream machinery lines up with what we see dynamically: the program uses `std::cout` / `std::cin`-style printing and input to prompt for the password, then relies on standard *C++* string and stream operations inside the validation logic rather than raw *WinAPI* calls.

4.2.2 VCRUNTIME140D.dll

OFTs	FTs (IAT)	Hint	Name
Qword	Qword	Word	szAnsi
000000000002C306	000000000002C306	0021	__std_exception_copy
000000000002C31E	000000000002C31E	0022	__std_exception_destroy
000000000002C338	000000000002C338	0001	_CxxThrowException
000000000002C364	000000000002C364	0008	__C_specific_handler
000000000002C37C	000000000002C37C	0009	__C_specific_handler_noexcept
000000000002C39C	000000000002C39C	0025	__std_type_info_destroy_list
000000000002C3BC	000000000002C3BC	001B	__current_exception
000000000002C3D2	000000000002C3D2	001C	__current_exception_context
000000000002C3F0	000000000002C3F0	002E	__vcrt_GetModuleFileNameW
000000000002C40C	000000000002C40C	002F	__vcrt_GetModuleHandleW
000000000002C426	000000000002C426	0031	__vcrt_LoadLibraryExW
000000000002C2F2	000000000002C2F2	003B	memcmp
000000000002C2FC	000000000002C2FC	003C	memcpy

4.2.3 VCRUNTIME140D_1D.dll

OFTs	FTs (IAT)	Hint	Name
Qword	Qword	Word	szAnsi
000000000002C34E	000000000002C34E	0000	__CxxFrameHandler4

4.2.4 ucrtbased.dll

OFTs	FTs (IAT)	Hint	Name
Qword	Qword	Word	szAnsi
000000000002C62C	000000000002C62C	004D	__p__commode
000000000002C63C	000000000002C63C	02C2	_seh_filter_dll
000000000002C64E	000000000002C64E	0172	_initialize_onexit_table
000000000002C66A	000000000002C66A	02B5	_register_onexit_function
000000000002C686	000000000002C686	00E5	_execute_onexit_table
000000000002C69E	000000000002C69E	00C2	_crt_atexit
000000000002C6AC	000000000002C6AC	00C1	_crt_at_quick_exit
000000000002C6C2	000000000002C6C2	052C	strcpy_s
000000000002C6CE	000000000002C6CE	0528	strcat_s
000000000002C6DA	000000000002C6DA	0068	__stdio_common_vsprintf_s
000000000002C6F6	000000000002C6F6	054B	terminate
000000000002C606	000000000002C606	00B5	_configthreadlocale
000000000002C712	000000000002C712	03B8	_wsplitpath_s
000000000002C722	000000000002C722	0564	wcscpy_s
000000000002C5C4	000000000002C5C4	00A4	_cexit
000000000002C4CA	000000000002C4CA	0005	_CrtDbgReportW
000000000002C4C0	000000000002C4C0	04D8	malloc
000000000002C4B4	000000000002C4B4	00A1	_callnewh
000000000002C4A6	000000000002C4A6	0205	_malloc_dbg
000000000002C49A	000000000002C49A	011D	_free_dbg
000000000002C490	000000000002C490	0531	strlen
000000000002C486	000000000002C486	0546	system
000000000002C476	000000000002C476	0004	_CrtDbgReport
000000000002C464	000000000002C464	0179	_invoke_watson
000000000002C61C	000000000002C61C	02CE	_set_new_mode
000000000002C5CE	000000000002C5CE	009F	_c_exit
000000000002C5B6	000000000002C5B6	004A	__p___argv
000000000002C5A8	000000000002C5A8	0049	__p___argc
000000000002C59A	000000000002C59A	02CB	_set_fmode
000000000002C592	000000000002C592	00EA	_exit
000000000002C58A	000000000002C58A	0450	exit
000000000002C57C	000000000002C57C	0175	_initterm_e
000000000002C570	000000000002C570	0174	_initterm
000000000002C54E	000000000002C54E	013D	_get_initial_narrow_environment

000000000002C5D8	000000000002C5D8	02B6	_register_thread_local_exe_atexit_ca...
000000000002C52C	000000000002C52C	0171	_initialize_narrow_environment
000000000002C512	000000000002C512	00B6	_configure_narrow_argv
000000000002C4FE	000000000002C4FE	005B	__setusermatherr
000000000002C702	000000000002C702	039C	_wmakepath_s
000000000002C4DC	000000000002C4DC	02C3	_seh_filter_exe
000000000002C4EE	000000000002C4EE	02C6	_set_app_type

4.2.5 KERNEL32.dll

OFTs	FTs (IAT)	Hint	Name
Qword	Qword	Word	szAnsi
000000000002C73C	000000000002C73C	0245	GetCurrentThreadId
000000000002C752	000000000002C752	03AF	IsDebuggerPresent
000000000002C766	000000000002C766	0497	RaiseException
000000000002C778	000000000002C778	0422	MultiByteToWideChar
000000000002C78E	000000000002C78E	064B	WideCharToMultiByte
000000000002C7A4	000000000002C7A4	0508	RtlCaptureContext
000000000002C7B8	000000000002C7B8	0510	RtlLookupFunctionEntry
000000000002C7D2	000000000002C7D2	0517	RtlVirtualUnwind
000000000002C7E6	000000000002C7E6	05FA	UnhandledExceptionFilter
000000000002C802	000000000002C802	05B7	SetUnhandledExceptionFilter
000000000002C820	000000000002C820	0240	GetCurrentProcess
000000000002C834	000000000002C834	05D7	TerminateProcess
000000000002C848	000000000002C848	03B7	IsProcessorFeaturePresent
000000000002C864	000000000002C864	0480	QueryPerformanceCounter
000000000002C87E	000000000002C87E	0241	GetCurrentProcessId
000000000002C894	000000000002C894	0319	GetSystemTimeAsFileTime
000000000002C8AE	000000000002C8AE	0399	InitializeSListHead
000000000002C8C4	000000000002C8C4	0300	GetStartupInfoW
000000000002C8D6	000000000002C8D6	02A4	GetModuleHandleW
000000000002C8EA	000000000002C8EA	028C	GetLastError
000000000002C8FA	000000000002C8FA	037B	HeapAlloc
000000000002C906	000000000002C906	037F	HeapFree
000000000002C912	000000000002C912	02E3	GetProcessHeap
000000000002C924	000000000002C924	061B	VirtualQuery
000000000002C934	000000000002C934	01D3	FreeLibrary
000000000002C942	000000000002C942	02DC	GetProcAddress

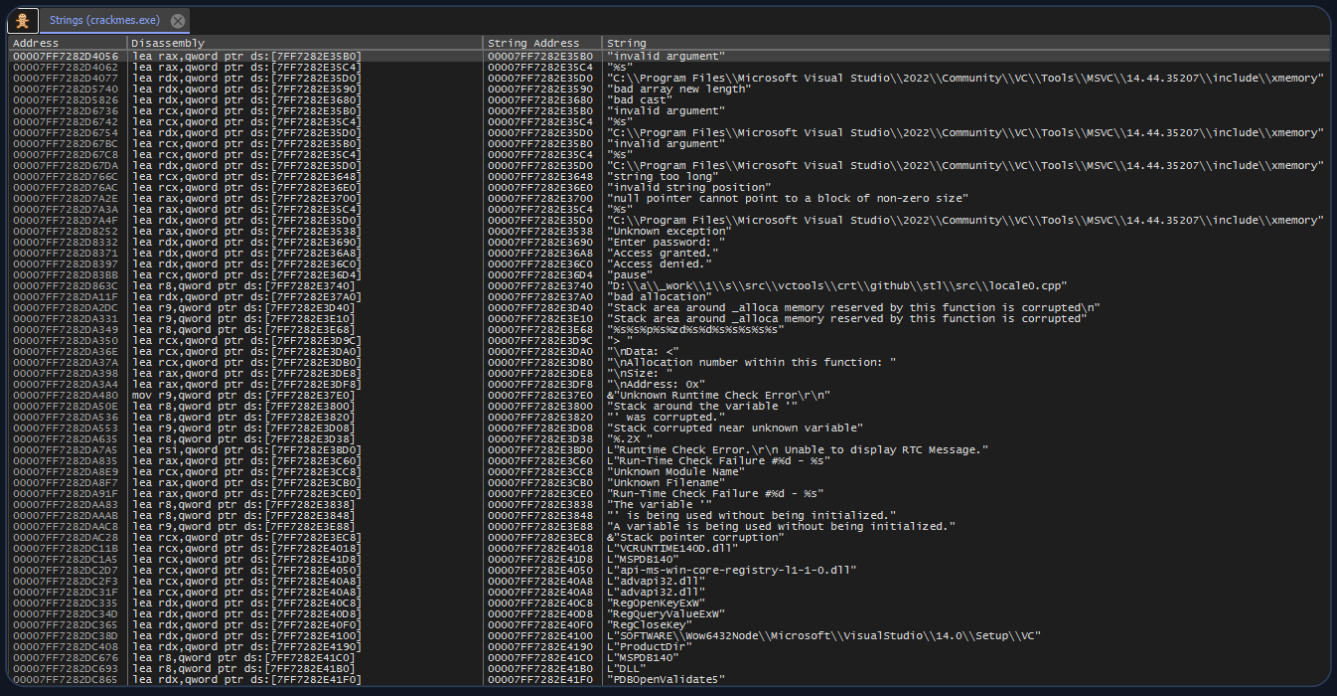
5. Dynamic Analysis

5.1 Baseline Run

Starting the program in *x64dbg* yields no immediate or obvious signs of any anti-debugging logic.

5.2 String Driven-Entry

Searching for string references within the target *Portable Executable* (*PE*) yields the following results.


Strings (crackmes.exe)

Address	Disassembly	String Address	String
00007FF7282D4056	lea rax,qword ptr ds:[7FF7282E35B0]	00007FF7282E35B0	"invalid argument"
00007FF7282D4062	lea rax,qword ptr ds:[7FF7282E35C4]	00007FF7282E35C4	"%s"
00007FF7282D4077	lea rdx,qword ptr ds:[7FF7282E35D0]	00007FF7282E35D0	"C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.44.35207\\include\\xmemory"
00007FF7282D5740	lea rdx,qword ptr ds:[7FF7282E3590]	00007FF7282E3590	"bad array new length"
00007FF7282D5826	lea rdx,qword ptr ds:[7FF7282E3680]	00007FF7282E3680	"bad cast"
00007FF7282D6736	lea rcx,qword ptr ds:[7FF7282E35B0]	00007FF7282E35B0	"invalid argument"
00007FF7282D6742	lea rcx,qword ptr ds:[7FF7282E35C4]	00007FF7282E35C4	"%s"
00007FF7282D6754	lea rdx,qword ptr ds:[7FF7282E35D0]	00007FF7282E35D0	"C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.44.35207\\include\\xmemory"
00007FF7282D678C	lea rcx,qword ptr ds:[7FF7282E35B0]	00007FF7282E35B0	"invalid argument"
00007FF7282D67C8	lea rcx,qword ptr ds:[7FF7282E35C4]	00007FF7282E35C4	"%s"
00007FF7282D67DA	lea rdx,qword ptr ds:[7FF7282E35D0]	00007FF7282E35D0	"C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.44.35207\\include\\xmemory"
00007FF7282D766C	lea rcx,qword ptr ds:[7FF7282E3648]	00007FF7282E3648	"string too long"
00007FF7282D76AC	lea rcx,qword ptr ds:[7FF7282E36E0]	00007FF7282E36E0	"invalid string position"
00007FF7282D7A2E	lea rax,qword ptr ds:[7FF7282E3700]	00007FF7282E3700	"null pointer cannot point to a block of non-zero size"
00007FF7282D7A3A	lea rax,qword ptr ds:[7FF7282E35C4]	00007FF7282E35C4	"%s"
00007FF7282D7A4F	lea rdx,qword ptr ds:[7FF7282E35D0]	00007FF7282E35D0	"C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.44.35207\\include\\xmemory"
00007FF7282D8252	lea rax,qword ptr ds:[7FF7282E3538]	00007FF7282E3538	"Unknown exception"
00007FF7282D8332	lea rdx,qword ptr ds:[7FF7282E3690]	00007FF7282E3690	"Enter password: "
00007FF7282D8371	lea rdx,qword ptr ds:[7FF7282E36A8]	00007FF7282E36A8	"Access granted."
00007FF7282D8397	lea rdx,qword ptr ds:[7FF7282E36C0]	00007FF7282E36C0	"Access denied."
00007FF7282D83BB	lea rcx,qword ptr ds:[7FF7282E36D4]	00007FF7282E36D4	"pause"
00007FF7282D863C	lea r8,qword ptr ds:[7FF7282E3740]	00007FF7282E3740	"D:\\a_work\\1\\s\\src\\vctools\\crt\\github\\stl\\src\\locale0.cpp"
00007FF7282DA11F	lea rdx,qword ptr ds:[7FF7282E37A0]	00007FF7282E37A0	"bad allocation"
00007FF7282DA2DC	lea r9,qword ptr ds:[7FF7282E3D40]	00007FF7282E3D40	"Stack area around _alloca memory reserved by this function is corrupted\n"
00007FF7282DA331	lea r9,qword ptr ds:[7FF7282E3E10]	00007FF7282E3E10	"Stack area around _alloca memory reserved by this function is corrupted"
00007FF7282DA349	lea r8,qword ptr ds:[7FF7282E3E68]	00007FF7282E3E68	"%s%s%p%s%zd%s%d%s%s%s%s%s"
00007FF7282DA350	lea rcx,qword ptr ds:[7FF7282E3D9C]	00007FF7282E3D9C	"> "
00007FF7282DA36E	lea rcx,qword ptr ds:[7FF7282E3DA0]	00007FF7282E3DA0	"\nData: <"
00007FF7282DA37A	lea rcx,qword ptr ds:[7FF7282E3DB0]	00007FF7282E3DB0	"\nAllocation number within this function: "
00007FF7282DA398	lea rax,qword ptr ds:[7FF7282E3DE8]	00007FF7282E3DE8	"\nSize: "
00007FF7282DA3A4	lea rax,qword ptr ds:[7FF7282E3DF8]	00007FF7282E3DF8	"\nAddress: 0x"
00007FF7282DA480	mov r9,qword ptr ds:[7FF7282E37E0]	00007FF7282E37E0	&"Unknown Runtime Check Error\r\n"
00007FF7282DA50E	lea r8,qword ptr ds:[7FF7282E3800]	00007FF7282E3800	"Stack around the variable '"
00007FF7282DA536	lea r8,qword ptr ds:[7FF7282E3820]	00007FF7282E3820	"' was corrupted."
00007FF7282DA553	lea r9,qword ptr ds:[7FF7282E3D08]	00007FF7282E3D08	"Stack corrupted near unknown variable"
00007FF7282DA635	lea r8,qword ptr ds:[7FF7282E3D38]	00007FF7282E3D38	"%.2X "
00007FF7282DA7A5	lea rsi,qword ptr ds:[7FF7282E3BD0]	00007FF7282E3BD0	L"Runtime Check Error.\r\n Unable to display RTC Message."
00007FF7282DA835	lea rax,qword ptr ds:[7FF7282E3C60]	00007FF7282E3C60	L"Run-Time Check Failure #%d - %s"
00007FF7282DA8E9	lea rcx,qword ptr ds:[7FF7282E3CC8]	00007FF7282E3CC8	"Unknown Module Name"
00007FF7282DA8F7	lea rax,qword ptr ds:[7FF7282E3CB0]	00007FF7282E3CB0	"Unknown Filename"
00007FF7282DA91F	lea rax,qword ptr ds:[7FF7282E3CE0]	00007FF7282E3CE0	"Run-Time Check Failure #%d - %s"
00007FF7282DAA83	lea r8,qword ptr ds:[7FF7282E3838]	00007FF7282E3838	"The variable '"
00007FF7282DAAA8	lea r8,qword ptr ds:[7FF7282E3848]	00007FF7282E3848	"' is being used without being initialized."
00007FF7282DAAC8	lea r9,qword ptr ds:[7FF7282E3E88]	00007FF7282E3E88	"A variable is being used without being initialized."
00007FF7282DAC28	lea rcx,qword ptr ds:[7FF7282E3EC8]	00007FF7282E3EC8	&"Stack pointer corruption"
00007FF7282DC118	lea rcx,qword ptr ds:[7FF7282E4018]	00007FF7282E4018	L"VCRUNTIME140D.dll"
00007FF7282DC1A5	lea rcx,qword ptr ds:[7FF7282E41D8]	00007FF7282E41D8	L"MSPDB140"
00007FF7282DC2D7	lea rcx,qword ptr ds:[7FF7282E4050]	00007FF7282E4050	L"api-ms-win-core-registry-l1-1-0.dll"
00007FF7282DC2F3	lea rcx,qword ptr ds:[7FF7282E40A8]	00007FF7282E40A8	L"advapi32.dll"
00007FF7282DC31F	lea rcx,qword ptr ds:[7FF7282E40A8]	00007FF7282E40A8	L"advapi32.dll"
00007FF7282DC335	lea rdx,qword ptr ds:[7FF7282E40C8]	00007FF7282E40C8	"RegOpenKeyExW"
00007FF7282DC34D	lea rdx,qword ptr ds:[7FF7282E40D8]	00007FF7282E40D8	"RegQueryValueExW"
00007FF7282DC365	lea rdx,qword ptr ds:[7FF7282E40F0]	00007FF7282E40F0	"RegCloseKey"
00007FF7282DC38D	lea rdx,qword ptr ds:[7FF7282E4100]	00007FF7282E4100	L"SOFTWARE\\Wow6432Node\\Microsoft\\VisualStudio\\14.0\\Setup\\VC"
00007FF7282DC408	lea rdx,qword ptr ds:[7FF7282E4190]	00007FF7282E4190	L"ProductDir"
00007FF7282DC676	lea r8,qword ptr ds:[7FF7282E41C0]	00007FF7282E41C0	L"MSPDB140"
00007FF7282DC693	lea r8,qword ptr ds:[7FF7282E41B0]	00007FF7282E41B0	L"DLL"
00007FF7282DC865	lea rdx,qword ptr ds:[7FF7282E41F0]	00007FF7282E41F0	"PDBOpenValidate5"

I make note of the following interesting references that I seen output onto the console from the target overview, "Enter password: ", "*Access denied.*", and based on assumption "*Access granted.*" - which we haven't seen yet.

Address	Disassembly	String Address	String
00007FF7282D8332	lea rdx,qword ptr ds:[7FF7282E3690]	00007FF7282E3690	"Enter password: "

Address	Disassembly	String Address	String
00007FF7282D8371	lea rdx,qword ptr ds:[7FF7282E36A8]	00007FF7282E36A8	"Access granted."
00007FF7282D8397	lea rdx,qword ptr ds:[7FF7282E36C0]	00007FF7282E36C0	"Access denied."

Double clicking on the string reference for "*Enter password:* " brings me into the disassembly view where I start to poke and prod around. I land on what seems to be the `main` function.

```
00007FF7282D82A0 | 40:55                    | push rbp
     |
00007FF7282D82A2 | 57                       | push rdi
     |
00007FF7282D82A3 | 48:81EC F8010000         | sub rsp,1F8
    |
00007FF7282D82AA | 48:8D6C24 20             | lea rbp,qword ptr ss:[rsp+20]
    |
00007FF7282D82AF | 48:8D7C24 20             | lea rdi,qword ptr ss:[rsp+20]
    |
00007FF7282D82B4 | B9 46000000              | mov ecx,46
     | 46:'F'
00007FF7282D82B9 | B8 CCCCCCCC              | mov eax,CCCCCCCC
     |
00007FF7282D82BE | F3:AB                    | rep stosd
    |
00007FF7282D82C0 | 48:8B05 79ED0000         | mov rax,qword ptr ds:[7FF7282E7040]
    | rax:EntryPoint
00007FF7282D82C7 | 48:33C5                  | xor rax,rbp
    | rax:EntryPoint
00007FF7282D82CA | 48:8985 C8010000         | mov qword ptr ss:[rbp+1C8],rax
     | rax:EntryPoint
00007FF7282D82D1 | 48:8D0D 9F4D0100         | lea rcx,qword ptr ds:[7FF7282ED077]
    |
00007FF7282D82D8 | E8 FE93FFFF              | call crackmes.7FF7282D16DB
     |
00007FF7282D82DD | 90                       | nop
    |
00007FF7282D82DE | C645 08 61               | mov byte ptr ss:[rbp+8],61
     | 61:'a'
00007FF7282D82E2 | C645 09 23               | mov byte ptr ss:[rbp+9],23
     | 23:'#'
00007FF7282D82E6 | C645 0A 6C               | mov byte ptr ss:[rbp+A],6C
     | 6C:'l'
00007FF7282D82EA | C645 0B 36               | mov byte ptr ss:[rbp+B],36
     | 36:'6'
00007FF7282D82EE | C645 0C 37               | mov byte ptr ss:[rbp+C],37
     | 37:'7'
00007FF7282D82F2 | C645 0D 27               | mov byte ptr ss:[rbp+D],27
     | 27:'''
00007FF7282D82F6 | C645 0E 67               | mov byte ptr ss:[rbp+E],67
     | 67:'g'
00007FF7282D82FA | C645 0F 64               | mov byte ptr ss:[rbp+F],64
     | 64:'d'
00007FF7282D82FE | C645 10 62               | mov byte ptr ss:[rbp+10],62
    | 62:'b'
```

```
00007FF7282D8302 | 48:C745 38 09000000      | mov qword ptr ss:[rbp+38],9
    | 09:'\t'
00007FF7282D830A | C645 54 55               | mov byte ptr ss:[rbp+54],55
    | 55:'U'
00007FF7282D830E | 41:B1 55                 | mov r9b,55
     | 55:'U'
00007FF7282D8311 | 41:B8 09000000           | mov r8d,9
    | 09:'\t'
00007FF7282D8317 | 48:8D55 08               | lea rdx,qword ptr ss:[rbp+8]
     | rdx:EntryPoint
00007FF7282D831B | 48:8D4D 78               | lea rcx,qword ptr ss:[rbp+78]
    |
00007FF7282D831F | E8 EC90FFFF              | call crackmes.7FF7282D1410
     |
00007FF7282D8324 | 90                       | nop
    |
00007FF7282D8325 | 48:8D8D B8000000         | lea rcx,qword ptr ss:[rbp+B8]
    |
00007FF7282D832C | E8 9B93FFFF              | call crackmes.7FF7282D16CC
     |
00007FF7282D8331 | 90                       | nop
    |
00007FF7282D8332 | 48:8D15 57B30000         | lea rdx,qword ptr ds:[7FF7282E3690]
    | rdx:EntryPoint, 00007FF7282E3690:"Enter password: "
00007FF7282D8339 | 48:8B0D 282E0100         | mov rcx,qword ptr ds:[<class
std::basic |
00007FF7282D8340 | E8 7E8DFFFF              | call crackmes.7FF7282D10C3
     |
00007FF7282D8345 | 90                       | nop
    |
00007FF7282D8346 | 48:8D95 B8000000         | lea rdx,qword ptr ss:[rbp+B8]
    | rdx:EntryPoint
00007FF7282D834D | 48:8B0D EC2E0100         | mov rcx,qword ptr ds:[<class
std::basic |
00007FF7282D8354 | E8 D48CFFFF              | call crackmes.7FF7282D102D
     |
00007FF7282D8359 | 90                       | nop
    |
00007FF7282D835A | 48:8D55 78               | lea rdx,qword ptr ss:[rbp+78]
    | rdx:EntryPoint
00007FF7282D835E | 48:8D8D B8000000         | lea rcx,qword ptr ss:[rbp+B8]
    |
00007FF7282D8365 | E8 678EFFFF              | call crackmes.7FF7282D11D1
     |
00007FF7282D836A | 0FB6C0                   | movzx eax,al
     |
```

```
00007FF7282D836D | 85C0                    | test eax,eax
     |
00007FF7282D836F | 74 26                   | je crackmes.7FF7282D8397
     |
00007FF7282D8371 | 48:8D15 30B30000        | lea rdx,qword ptr ds:[7FF7282E36A8]
    | rdx:EntryPoint, 00007FF7282E36A8:"Access granted."
00007FF7282D8378 | 48:8B0D E92D0100        | mov rcx,qword ptr ds:[<class
std::basic |
00007FF7282D837F | E8 3F8DFFFF             | call crackmes.7FF7282D10C3
     |
00007FF7282D8384 | 48:8D15 C58CFFFF        | lea rdx,qword ptr ds:[7FF7282D1050]
    | rdx:EntryPoint
00007FF7282D838B | 48:8BC8                 | mov rcx,rax
    | rax:EntryPoint
00007FF7282D838E | FF15 F42D0100           | call qword ptr ds:[<public: class
std:: |
00007FF7282D8394 | 90                      | nop
    |
00007FF7282D8395 | EB 24                   | jmp crackmes.7FF7282D83BB
    |
00007FF7282D8397 | 48:8D15 22B30000        | lea rdx,qword ptr ds:[7FF7282E36C0]
    | rdx:EntryPoint, 00007FF7282E36C0:"Access denied."
00007FF7282D839E | 48:8B0D C32D0100        | mov rcx,qword ptr ds:[<class
std::basic |
00007FF7282D83A5 | E8 198DFFFF             | call crackmes.7FF7282D10C3
     |
00007FF7282D83AA | 48:8D15 9F8CFFFF        | lea rdx,qword ptr ds:[7FF7282D1050]
    | rdx:EntryPoint
00007FF7282D83B1 | 48:8BC8                 | mov rcx,rax
    | rax:EntryPoint
00007FF7282D83B4 | FF15 CE2D0100           | call qword ptr ds:[<public: class
std:: |
00007FF7282D83BA | 90                      | nop
    |
00007FF7282D83BB | 48:8D0D 12B30000        | lea rcx,qword ptr ds:[7FF7282E36D4]
    | 00007FF7282E36D4:"pause"
00007FF7282D83C2 | FF15 E8300100           | call qword ptr ds:[<system>]
     |
00007FF7282D83C8 | 90                      | nop
    |
00007FF7282D83C9 | C785 B4010000 00000000  | mov dword ptr ss:[rbp+1B4],0
     |
00007FF7282D83D3 | 48:8D8D B8000000        | lea rcx,qword ptr ss:[rbp+B8]
    |
00007FF7282D83DA | E8 618DFFFF             | call crackmes.7FF7282D1140
     |
```

```
00007FF7282D83DF | 90                          | nop
    |
00007FF7282D83E0 | 48:8D4D 78                  | lea rcx,qword ptr ss:[rbp+78]
    |
00007FF7282D83E4 | E8 578DFFFF                 | call crackmes.7FF7282D1140
     |
00007FF7282D83E9 | 8B85 B4010000               | mov eax,dword ptr ss:[rbp+1B4]
     |
00007FF7282D83EF | 8BF8                        | mov edi,eax
    |
00007FF7282D83F1 | 48:8D4D E0                  | lea rcx,qword ptr ss:[rbp-20]
    |
00007FF7282D83F5 | 48:8D15 A4AD0000            | lea rdx,qword ptr ds:[7FF7282E31A0]
    | rdx:EntryPoint
00007FF7282D83FC | E8 C291FFFF                 | call crackmes.7FF7282D15C3
     |
00007FF7282D8401 | 8BC7                        | mov eax,edi
    |
00007FF7282D8403 | 48:8B8D C8010000            | mov rcx,qword ptr ss:[rbp+1C8]
     |
00007FF7282D840A | 48:33CD                     | xor rcx,rbp
    |
00007FF7282D840D | E8 368FFFFF                 | call crackmes.7FF7282D1348
     |
00007FF7282D8412 | 48:8DA5 D8010000            | lea rsp,qword ptr ss:[rbp+1D8]
     |
00007FF7282D8419 | 5F                          | pop rdi
    |
00007FF7282D841A | 5D                          | pop rbp
    |
00007FF7282D841B | C3                          | ret
    |
```

---

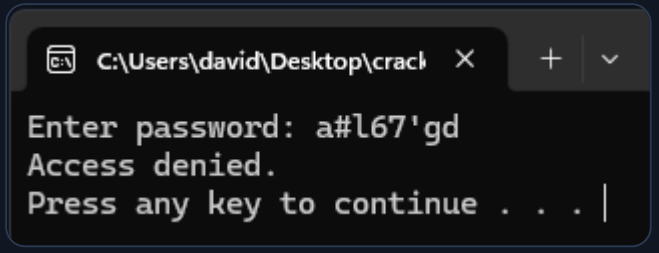
6. Validation Path

Right off the bat I notice a constant being loaded onto the stack.

```
00007FF7282D82DE | C645 08 61              | mov byte ptr ss:[rbp+8],61
     | 61:'a'
00007FF7282D82E2 | C645 09 23              | mov byte ptr ss:[rbp+9],23
     | 23:'#'
00007FF7282D82E6 | C645 0A 6C              | mov byte ptr ss:[rbp+A],6C
     | 6C:'l'
00007FF7282D82EA | C645 0B 36              | mov byte ptr ss:[rbp+B],36
     | 36:'6'
00007FF7282D82EE | C645 0C 37              | mov byte ptr ss:[rbp+C],37
     | 37:'7'
00007FF7282D82F2 | C645 0D 27              | mov byte ptr ss:[rbp+D],27
     | 27:'''
00007FF7282D82F6 | C645 0E 67              | mov byte ptr ss:[rbp+E],67
     | 67:'g'
00007FF7282D82FA | C645 0F 64              | mov byte ptr ss:[rbp+F],64
     | 64:'d'
00007FF7282D82FE | C645 10 62              | mov byte ptr ss:[rbp+10],62
    | 62:'b'
00007FF7282D8302 | 48:C745 38 09000000     | mov qword ptr ss:[rbp+38],9
    | 09:'\t'
00007FF7282D830A | C645 54 55              | mov byte ptr ss:[rbp+54],55
    | 55:'U'
00007FF7282D830E | 41:B1 55                | mov r9b,55
     | 55:'U'
00007FF7282D8311 | 41:B8 09000000          | mov r8d,9
    | 09:'\t'
```

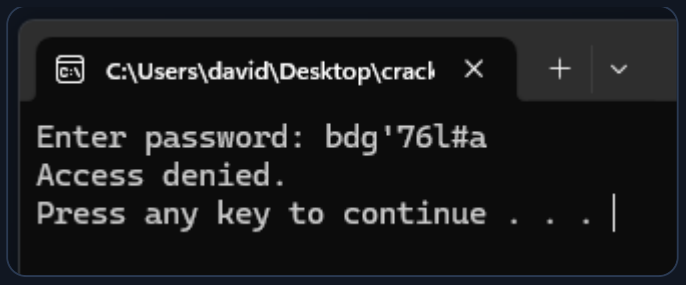
Specifically focusing on the offsets `+0x8` - `+0xF`, which seems to load the bytes `61 23 6C 36 37 27 67 64 62` == `a # l 6 7 ' g d b`, `a#l67'gdb` without the spaces added for readability.

Due to how strange and suspicious looking this sequence of bytes looks, I decide to plug it in as the password and see if it could be the flag.



It's not going to be that easy it seems!

What about, backwards - `bdg'76l#a`?



Womp womp womp!
With the horsing around out of the way, I start further analysing the `main` function.

Here we can see the logic that outputs "*Enter password:* " to the console.

```
00007FF7282D8332 | 48:8D15 57B30000          | lea rdx,qword ptr ds:[7FF7282E3690]
    | 00007FF7282E3690:"Enter password: "
00007FF7282D8339 | 48:8B0D 282E0100          | mov rcx,qword ptr ds:[<class
std::basic |
00007FF7282D8340 | E8 7E8DFFFF               | call crackmes.7FF7282D10C3
     |
```

Followed by a call that retrieves the user input from the console.

```
00007FF7282D8346 | 48:8D95 B8000000          | lea rdx,qword ptr ss:[rbp+B8]
    |
00007FF7282D834D | 48:8B0D EC2E0100          | mov rcx,qword ptr ds:[<class
std::basic |
00007FF7282D8354 | E8 D48CFFFF               | call crackmes.7FF7282D102D
     | get input from user
```

Which is then proceeded by what I think is the comparison function.

```
00007FF7282D835A | 48:8D55 78                | lea rdx,qword ptr ss:[rbp+78]
    |
00007FF7282D835E | 48:8D8D B8000000          | lea rcx,qword ptr ss:[rbp+B8]
    |
00007FF7282D8365 | E8 678EFFFF               | call crackmes.7FF7282D11D1
     |
```

Stepping into the comparison function reveals the following assembly.

```
00007FF7282D3D70 | 48:895424 10              | mov qword ptr ss:[rsp+10],rdx
    |
```

```
00007FF7282D3D75 | 48:894C24 08              | mov qword ptr ss:[rsp+8],rcx
     |
00007FF7282D3D7A | 55                        | push rbp
     |
00007FF7282D3D7B | 57                        | push rdi
     |
00007FF7282D3D7C | 48:81EC E8000000          | sub rsp,E8
     |
00007FF7282D3D83 | 48:8D6C24 20              | lea rbp,qword ptr ss:[rsp+20]
    |
00007FF7282D3D88 | 48:8D0D C9920100          | lea rcx,qword ptr ds:[7FF7282ED058]
    |
00007FF7282D3D8F | E8 47D9FFFF               | call crackmes.7FF7282D16DB
     |
00007FF7282D3D94 | 90                        | nop
    |
00007FF7282D3D95 | 48:8B95 E8000000          | mov rdx,qword ptr ss:[rbp+E8]
    |
00007FF7282D3D9C | 48:8B8D E0000000          | mov rcx,qword ptr ss:[rbp+E0]
    |
00007FF7282D3DA3 | E8 07D8FFFF               | call crackmes.7FF7282D15AF
     |
00007FF7282D3DA8 | 48:8DA5 C8000000          | lea rsp,qword ptr ss:[rbp+C8]
    |
00007FF7282D3DAF | 5F                        | pop rdi
    |
00007FF7282D3DB0 | 5D                        | pop rbp
    |
00007FF7282D3DB1 | C3                        | ret
    |
```

Which appears to make a few more function calls of importance, `call crackmes.7FF7282D15AF`.

```
00007FF7282D6DF0 | 48:895424 10              | mov qword ptr ss:[rsp+10],rdx
    |
00007FF7282D6DF5 | 48:894C24 08              | mov qword ptr ss:[rsp+8],rcx
     |
00007FF7282D6DFA | 55                        | push rbp
     |
00007FF7282D6DFB | 57                        | push rdi
     |
00007FF7282D6DFC | 48:81EC F8000000          | sub rsp,F8
     |
```

```
00007FF7282D6E03 | 48:8D6C24 20             | lea rbp,qword ptr ss:[rsp+20]
    |
00007FF7282D6E08 | 48:8D0D 49620100         | lea rcx,qword ptr ds:[7FF7282ED058]
    |
00007FF7282D6E0F | E8 C7A8FFFF              | call crackmes.7FF7282D16DB
     |
00007FF7282D6E14 | 90                       | nop
    |
00007FF7282D6E15 | 48:8B85 F8000000         | mov rax,qword ptr ss:[rbp+F8]
    |
00007FF7282D6E1C | 48:8BC8                  | mov rcx,rax
    |
00007FF7282D6E1F | E8 F7A4FFFF              | call crackmes.7FF7282D131B
     |
00007FF7282D6E24 | 48:8985 C0000000         | mov qword ptr ss:[rbp+C0],rax
    |
00007FF7282D6E2B | 48:8B8D F0000000         | mov rcx,qword ptr ss:[rbp+F0]
    |
00007FF7282D6E32 | E8 E4A4FFFF              | call crackmes.7FF7282D131B
     |
00007FF7282D6E37 | 48:8B8D F8000000         | mov rcx,qword ptr ss:[rbp+F8]
    |
00007FF7282D6E3E | 4C:8B49 18               | mov r9,qword ptr ds:[rcx+18]
     |
00007FF7282D6E42 | 48:8B8D C0000000         | mov rcx,qword ptr ss:[rbp+C0]
    |
00007FF7282D6E49 | 4C:8BC1                  | mov r8,rcx
     |
00007FF7282D6E4C | 48:8B8D F0000000         | mov rcx,qword ptr ss:[rbp+F0]
    |
00007FF7282D6E53 | 48:8B51 18               | mov rdx,qword ptr ds:[rcx+18]
    |
00007FF7282D6E57 | 48:8BC8                  | mov rcx,rax
    | rax:"4v9cbr217"
00007FF7282D6E5A | E8 C8A7FFFF              | call crackmes.7FF7282D1627
     |
00007FF7282D6E5F | 48:8DA5 D8000000         | lea rsp,qword ptr ss:[rbp+D8]
    |
00007FF7282D6E66 | 5F                       | pop rdi
    |
00007FF7282D6E67 | 5D                       | pop rbp
    |
00007FF7282D6E68 | C3                       | ret
    |
```

It seems that `RAX` is `4v9cbr217` after `call crackmes.7FF7282D131B`. I decide to step into that function.

```
00007FF7282D71F0 | 48:894C24 08              | mov qword ptr ss:[rsp+8],rcx
     |
00007FF7282D71F5 | 55                        | push rbp
     |
00007FF7282D71F6 | 57                        | push rdi
     |
00007FF7282D71F7 | 48:81EC 08010000          | sub rsp,108
    |
00007FF7282D71FE | 48:8D6C24 20              | lea rbp,qword ptr ss:[rsp+20]
    |
00007FF7282D7203 | 48:8D0D 4E5E0100          | lea rcx,qword ptr ds:[7FF7282ED058]
    |
00007FF7282D720A | E8 CCA4FFFF               | call crackmes.7FF7282D16DB
     |
00007FF7282D720F | 90                        | nop
    |
00007FF7282D7210 | 48:8B85 00010000          | mov rax,qword ptr ss:[rbp+100]
     |
00007FF7282D7217 | 48:83C0 08                | add rax,8
    | rax:"4v9cbr217"
00007FF7282D721B | 48:8945 08                | mov qword ptr ss:[rbp+8],rax
     | [rbp+08]:"4v9cbr217"
00007FF7282D721F | 48:8B8D 00010000          | mov rcx,qword ptr ss:[rbp+100]
     |
00007FF7282D7226 | E8 51A4FFFF               | call crackmes.7FF7282D167C
     |
00007FF7282D722B | 0FB6C0                    | movzx eax,al
     |
00007FF7282D722E | 85C0                      | test eax,eax
     |
00007FF7282D7230 | 74 14                     | je crackmes.7FF7282D7246
     |
00007FF7282D7232 | 48:8B85 00010000          | mov rax,qword ptr ss:[rbp+100]
     |
00007FF7282D7239 | 48:8B48 08                | mov rcx,qword ptr ds:[rax+8]
     |
00007FF7282D723D | E8 A8A4FFFF               | call crackmes.7FF7282D16EA
     |
00007FF7282D7242 | 48:8945 08                | mov qword ptr ss:[rbp+8],rax
     | [rbp+08]:"4v9cbr217"
00007FF7282D7246 | 48:8B45 08                | mov rax,qword ptr ss:[rbp+8]
     | [rbp+08]:"4v9cbr217"
```

```
00007FF7282D724A | 48:8DA5 E8000000          | lea rsp,qword ptr ss:[rbp+E8]
    |
00007FF7282D7251 | 5F                        | pop rdi
    |
00007FF7282D7252 | 5D                        | pop rbp
    |
00007FF7282D7253 | C3                        | ret
    |
```

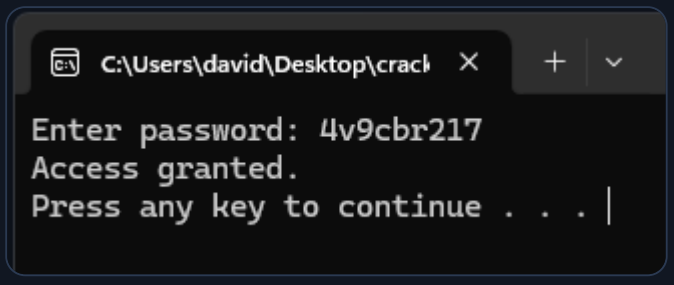
Which seems to load the bytes `34 76 39 63 62 72 32 31 37` - `4v9cbr217` - and return it, which is looking awfully a lot like the flag.
Stepping back out of the above function, I notice my input `helloworld` being loaded in from `ss:[RBP+F8]` into `RCX` and `4v9cbr217` being loaded into `R8` before the following *call* instruction which seems to do the actual comparison.

```
00007FF7282D6E5A | E8 C8A7FFFF               | call crackmes.7FF7282D1627
     |
```

---

7. Testing the New Flag

With the dynamic analysis done, I fire up the *crackme* and enter `4v9cbr217` as the flag - password.

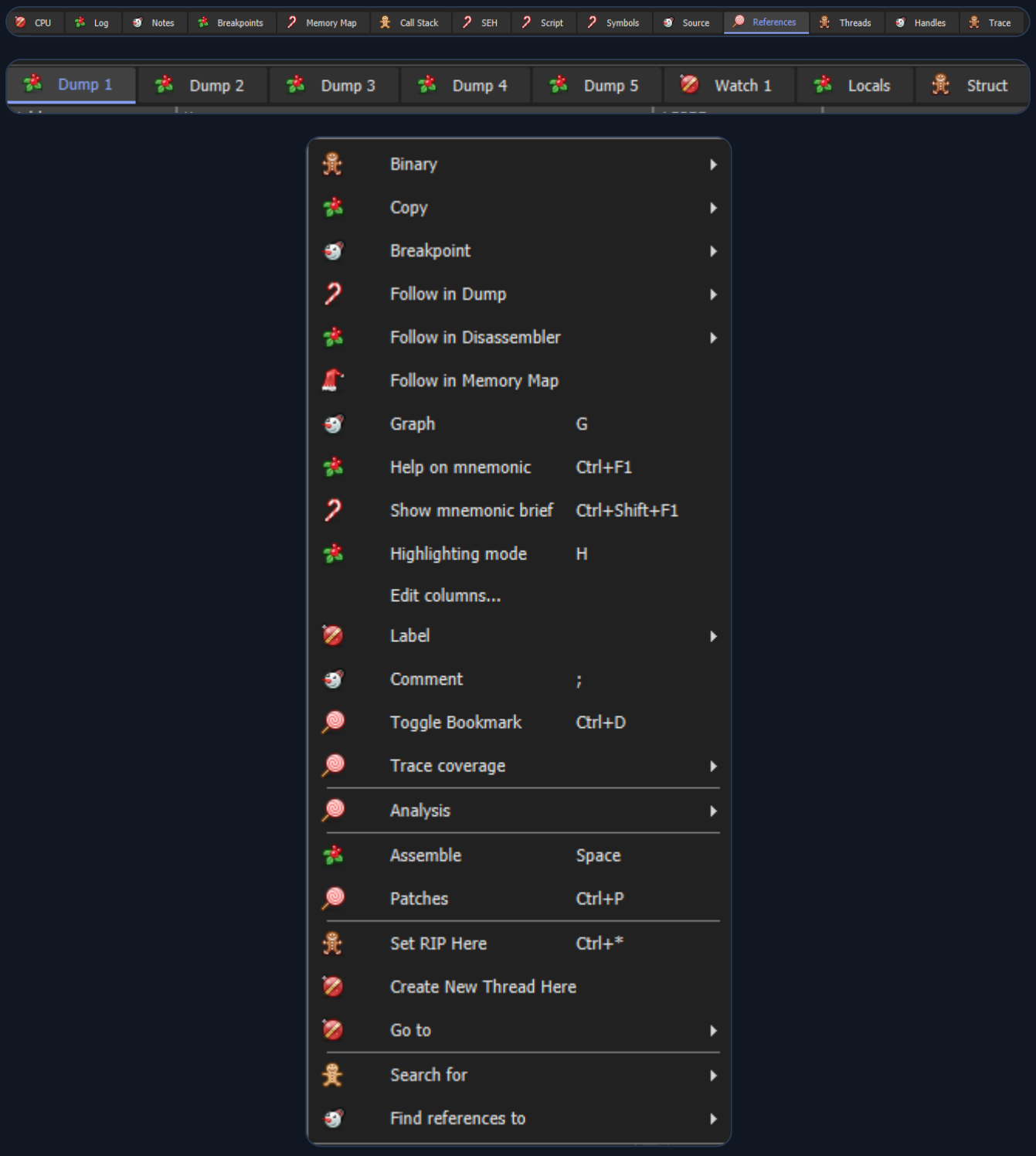


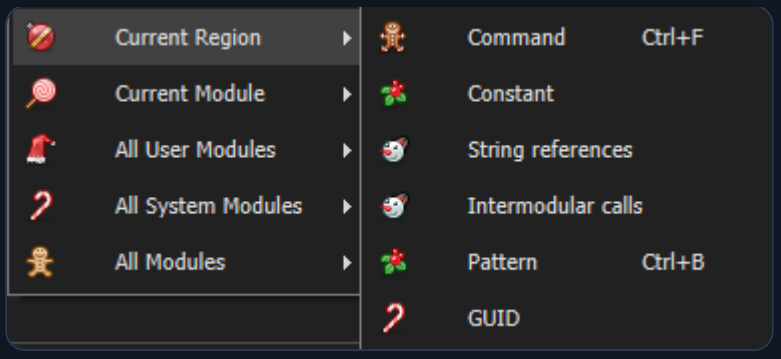
Amazing! Third time's the charm.

---

8. *x64dbg* Festive Icon Set

Side note, I just launched *x64dbg* on Christmas Eve and it has a whole other icon set for the holiday!





Neato!

---

9. Conclusion

In the end, this crackme was a straightforward C++ console app hiding behind a lot of standard library noise. The section layout and imports made it clear early on that there was no packing or fancy obfuscation going on, just a debug build using *MSVCP* and the usual iostream and string machinery. Once I followed the "*Enter password: *" prompt into `main` and traced the calls that shuffled `std::string` objects around, the real logic basically revealed itself. Stepping through the comparison path in *x64dbg* showed the program constructing the secret string `4v9cbr217` and feeding it into the final check against my input. Typing that in gives a clean "*Access granted.*".
The main takeaway here is that even when the *C++* runtime makes the call graph look busy, sticking to the basics (strings, calls, and comparisons) gets you to the flag without needing any heavy tricks.

The final solution is the recovered password:

```
4v9cbr217
```