

Ventrilo 2.2 server port hacking

Since Ventrilo version 2.2 its not possible to change the port for the server without a license. So we need to cack it ;>

What u need is a debugger (Ollydug) and a Hexeditor (Hexworkshop 32)

If we change the variable „Port“ in ventrilo_srv.ini and then start the server it exits immediately. So we change the variable for example to 1234 and then load ventrilo_srv.exe into ollydebug.

Now we can search for all „text strings“ Ventrilo has stored in the executable.

Address	Disassembly	Comment
A85B	PUSH EBP	
A85C	. 8BEC	Backup
A85E	. 6A FF	Copy
A860	. 68 C0	Copy
A865	. 68 34	Copy
A86A	. 64:A0	Binary
A870	. 50	Assemble
A871	. 64:80	Space
A878	. 83EC	,ESP
A87B	. 53	Label
A87C	. 56	:
A87D	. 57	Comment
A87E	. 8965	;
A87F	. FF15	Breakpoint
A887	. 33D2	Hit trace
A889	. 8AD4	Hit trace
A88B	. 8915	Run trace
A891	. 8BC8	0194], EDX
A893	. 81E1	
A897	. 8900	Go to
A898	. C1E1	0190], ECX
A8A2	. 030A	
A8A4	. 8900	Follow in Dump
A8AA	. C1E8	018C], ECX
A8AD	. A3 80	View call tree
A8B2	. 6A 01	Ctrl+K
A8B4	. E8 10	0188], EAX
A8B9	. 59	
A8BC	. 85C0	Search for
A8C0	. 75 00	Find references to
A8C5	. 6A 10	Name (label) in current module
A8C6	. E8 A0	Ctrl+N
A8C8	. 85C0	View
A8CF	. 75 00	Name in all modules
A8D1	. E8 90	Command
A8D6	. 59	Ctrl+F
A8D7	. 8365	Copy to executable
A8D8	. E8 B0	Sequence of commands
A8E0	. FF15	Ctrl+S
A8E8	. A3 B8	Analysis
A8F0	. A3 70	Constant
A8F5	. E8 25	Binary string
A8FA	. E8 67	Ctrl+B
A8FF	. E8 A1	Next
		Ctrl+L
		All intermodular calls
		All commands
		All sequences
		All constants
		All switches
		All referenced text strings
		User-defined label
		User-defined comment

After a deeper look into the stringlist we found this:

Address	Disassembly	Comment
00410459	PUSH ventrilo.0042BAF0	ASCII "Server"
0041046C	PUSH ventrilo.0042BA9C	ASCII "Port"
00410471	PUSH ventrilo.0042BAF0	ASCII "Server"
0041048B	PUSH ventrilo.0042BA5C	ASCII "Only port 3784 is allowed on the public version of the server."
004104CF	PUSH ventrilo.0042BA50	ASCII "Duplicates"
004104D4	PUSH ventrilo.0042BAF0	ASCII "Server"
004104EC	PUSH ventrilo.0042BA44	ASCII "SendBuffer"
004104F1	PUSH ventrilo.0042BAF0	ASCII "Server"
0041050B	PUSH ventrilo.0042BA28	ASCII "SendBuffer must be >= 4096."
00410551	PUSH ventrilo.0042BA10	ASCII "RecvBuffer"

Now we switch into the memory where the string is pushed on the stack:

Address	Disassembly	Comment
00410440	. E8 DE26FFFF	CALL ventrilo.00402B30
00410452	. 6A 00	PUSH 0
00410454	. 68 A4BA4200	PUSH ventrilo.0042BAA4
00410459	. 68 F0BA4200	PUSH ventrilo.0042BAF0
0041045E	. 8D4C24 2C	LEA ECX, DWORD PTR SS:[ESP+2C]
00410462	. E8 A99EFFFF	CALL ventrilo.0040A310
00410467	. 68 C0E00000	PUSH 0EC8
0041046C	. 68 9CBA4200	PUSH ventrilo.0042BA9C
00410471	. 68 F0BA4200	PUSH ventrilo.0042BAF0
00410476	. 8D4C24 2C	LEA ECX, DWORD PTR SS:[ESP+2C]
0041047A	. A3 F0004300	MOV DWORD PTR DS:[4300F8], EAX
0041047F	. E8 8C9EFFFF	CALL ventrilo.0040A310
00410484	. 3D C0E00000	CMP EAX, 0EC8
00410489	. 74 42	JE SHORT ventrilo.004104CD
0041048B	. 68 5CBA4200	PUSH ventrilo.0042BA5C
00410490	. E8 CB320000	CALL ventrilo.00413760
00410495	. 83C4 04	ADD ESP, 4
00410498	. 8D4C24 20	LEA ECX, DWORD PTR SS:[ESP+20]
0041049C	. C64424 E8 00	MOV BYTE PTR SS:[ESP+58], 0
004104A1	. E8 7A97FFFF	CALL ventrilo.00409C20
004104A6	. 6A 01	PUSH 1
004104A8	. 8D4C24 34	LEA ECX, DWORD PTR SS:[ESP+34]
004104AC	. 896C24 5C	MOV DWORD PTR SS:[ESP+5C], EBP
004104B0	. E8 7B26FFFF	CALL ventrilo.00402B30
004104B5	. 5F	POP EDI
004104B6	. 5C	POP EBT

We have to interpret the following assembler instructions:
PUSH EAX 0EC8
PUSH „Port“
PUSH „Server“
LEAxxx
MOV xxx
CALL xxx
CMP EAX, 0EC8
JMP xxx

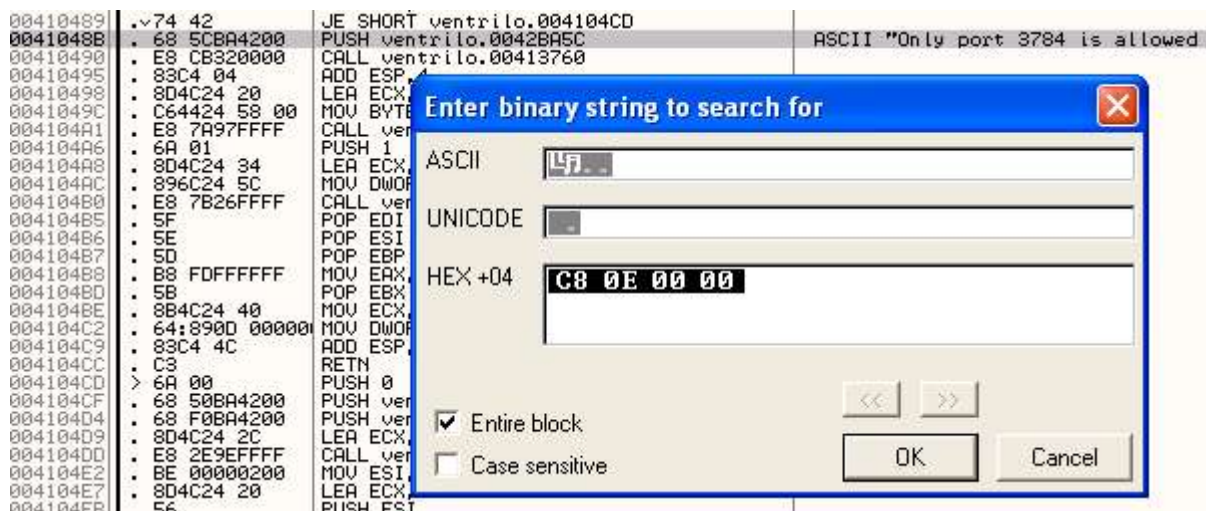
If we convert 0EC8 from hexadecimal into decimal we get the value 3784, this is the standard port ventrilo is using.

- PUSH standard port <- default port
- PUSH string „Port“ <- „Port“ in ini
- PUSH string „Server“ <- „[Server]“ in ini
- ...
- CALL something <- read from ini ???
- CMP EAX, 0EC8 <- is the value which is read from ini the standard port
- JE xxx <- if so go on
- PUSH string „Only...“ <- if not show message that only port 3784 is allowed

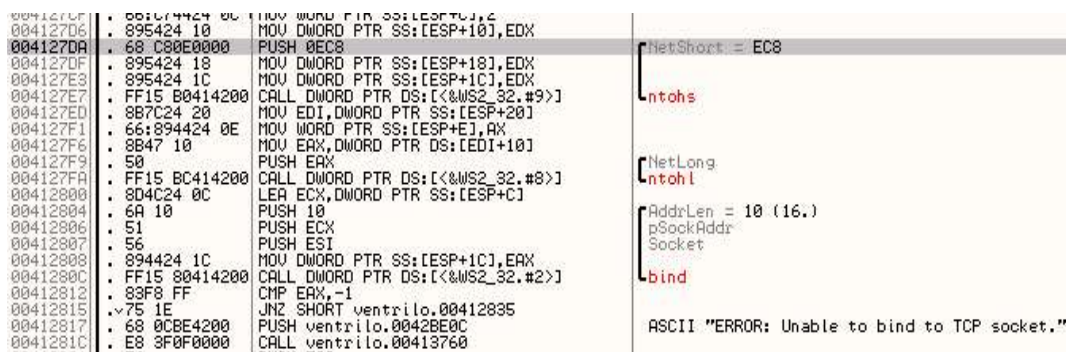
So we must change JE to JMP and our problem is solved?

No, because the return value (EAX = port) is never stored into a variable, so if ventrilo is now creating a new connection it does not use the value which was read from ini file.

So we must search for the standard port in the executable. The Hexvalue is 0x0EC8, in the executable it must be stored as a dword C8 0E 00 00.



We get this result:



Ventrilo wants to create a new connection with the port 3784. But we want to use our port which was read from the ini. So we must push it on the stack instead of 0EC8. Pushing a variable on the stack needs more than 6 bytes, but we can only overwrite the 6 bytes for the PUSH 0EC8 instruction (68 C8 0E 00 00).

We could replace it with a distance jump but first we need some unused memory... But hey, we can use the memory which is never executed anymore and is stored behind the procedure which is reading the port from the ini.

00410467	. 68 C0E0000	PUSH 0EC8	
0041046C	. 68 9CBA4200	PUSH ventrilo.0042BA9C	ASCII "Port"
00410471	. 68 F0BA4200	PUSH ventrilo.0042BAF0	ASCII "Server"
00410476	. 8D4C24 2C	LEA ECX,DWORD PTR SS:[ESP+2C]	
0041047A	. A3 F8004300	MOV DWORD PTR DS:[4300F8],EAX	
0041047F	. E8 8C9EFFFF	CALL ventrilo.0040A310	
00410489	> 30 C9E0000	CMPL EAX,0EC8	
0041048B	> 74 42	JE SHORT ventrilo.004104CD	
00410488	. 68 5CBA4200	PUSH ventrilo.0042BA5C	ASCII "Only port"
00410490	. E8 CB320000	CALL ventrilo.00413760	
00410495	. 83C4 04	ADD ESP,4	
00410498	. 8D4C24 20	LEA ECX,DWORD PTR SS:[ESP+20]	
0041049C	. C64424 58 00	MOV BYTE PTR SS:[ESP+58],0	
004104A1	. E8 7A97FFFF	CALL ventrilo.00409C20	
004104A8	. 8D4C24 34	LEA ECX,DWORD PTR SS:[ESP+34]	
004104AC	. 896C24 5C	MOV DWORD PTR SS:[ESP+5C],EBP	
004104B0	. E8 7B26FFFF	CALL ventrilo.00402B30	
004104B5	. 5F	POP EDI	
004104B6	. 5E	POP ESI	
004104B7	. 5D	POP EBP	
004104B8	. E8 F0FFFFFF	CALL ventrilo.00409C20	
004104BD	. 5B	POP EBX	
004104BE	. 8B4C24 40	MOV ECX,DWORD PTR SS:[ESP+40]	
004104C2	. 64:890D 00000	MOV DWORD PTR FS:[0],ECX	
004104C9	. 83C4 4C	ADD ESP,4C	
004104CD	. C3	RETN	
004104CD	> 6A 00	PUSH 0	
004104CF	. 68 50BA4200	PUSH ventrilo.0042BA50	ASCII "Duplicates"
004104D4	. 68 F0BA4200	PUSH ventrilo.0042BAF0	ASCII "Server"

unused

First we save EAX into a variable and then we change the JE to JMP

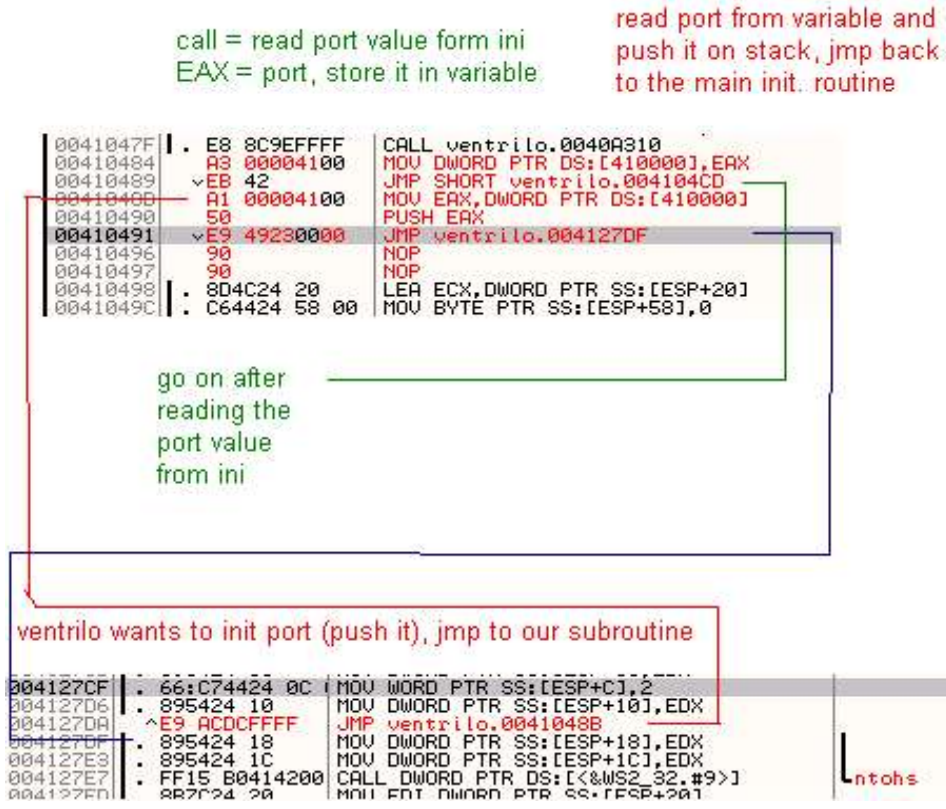
0041046C	. 68 9CBA4200	PUSH ventrilo.0042BA9C	ASCII "Port"
00410471	. 68 F0BA4200	PUSH ventrilo.0042BAF0	ASCII "Server"
00410476	. 8D4C24 2C	LEA ECX,DWORD PTR SS:[ESP+2C]	
0041047A	. A3 F8004300	MOV DWORD PTR DS:[4300F8],EAX	
0041047F	. E8 8C9EFFFF	CALL ventrilo.0040A310	
00410484	A3 00004100	MOV DWORD PTR DS:[410000],EAX	
00410489	> EB 42	JMP SHORT ventrilo.004104CD	
0041048B	. 68 5CBA4200	PUSH ventrilo.0042BA5C	ASCII "Only port"
00410490	. E8 CB320000	CALL ventrilo.00413760	
00410495	. 83C4 04	ADD ESP,4	
00410498	. 8D4C24 20	LEA ECX,DWORD PTR SS:[ESP+20]	

Note: We must change MOV DWORD PTR DS:[410000], EAX later to a valid address.

Now we can use the address from 0x0041048B to 0x004104CC to store our assembler instructions to push the variable on the stack and jumping behind the PUSH 0EC8 instruction where ventrilo initializes the port. After that we must change the PUSH 0EC8 instruction to a jump into our new instruction which pushes the variable to the stack.

004127CF	. 66:C74424 0C	MOV WORD PTR SS:[ESP+C],2	
004127D6	. 895424 10	MOV DWORD PTR SS:[ESP+10],EDX	
004127DA	> E9 ACDCFFFF	JMP ventrilo.0041048B	
004127DF	. 895424 18	MOV DWORD PTR SS:[ESP+18],EDX	
004127E3	. 895424 1C	MOV DWORD PTR SS:[ESP+1C],EDX	
004127E7	. FF15 B0414200	CALL DWORD PTR DS:[<&WS2_32.#9>]	ntohs
004127FD	. 8B7C24 20	MOV EDI,DWORD PTR SS:[ESP+20]	

Maybe a little bit hard to understand but this picture should show what's the idea behind this method:



The last what we need to do is changing the variable address. Normal variables are stored in the data section of the executable:

00400000	00001000	ventrilo		PE header	Imag	R	RWE
00401000	00023000	ventrilo	.text	code	Imag	R	RWE
00424000	00003000	ventrilo	.rdata	imports	Imag	R	RWE
00427000	00008000	ventrilo	.data	data	Imag	R	RWE
71A00000	00001000	WS2HELP		PE header	Imag	R	RWE

We only need to search some unused bytes there:

00431600	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431610	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431620	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431630	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431640	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431650	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431660	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431670	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431680	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00431690	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004316A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004316B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004316C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00

and can change the variable:

00410477	E8	0C7EFFFF	CALL	ventrilo.00400310
00410484	A3	60164300	MOV	DWORD PTR DS:[431660],EAX
00410489	EB	42	JMP	SHORT ventrilo.004104CD
0041048B	A1	60164300	MOV	EAX,DWORD PTR DS:[431660]
00410490	50		PUSH	EAX
00410491	E9	49230000	JMP	ventrilo.004127DF
00410496	90		NOP	
00410497	90		NOP	
00410498	90		NOP	

Now we must patch the changes with a Hexeditor directly into the executable and then we can use our new hacked ventrilo version :)

- uall