Stack Overflow Exercise

In this exercise you will commit the most hideous of all security attacks. You will force a simple program to run your own arbitrary program. This requires creating a data file long enough to overflow an array and store machine language of your exploit program on the stack. For this exercise you will create a file with Intel machine language to print your name.

- Load the Stack Overflow Virtual Machine with VMPlayer.
- Using Notepad, edit the simple text file `C:\temp\attackdata.txt` Put your name in the file starting at the 31st character. You can put anything in the first 30 bytes. Save the file and exit.
- In Visual Studio, build the project (press F7). Note that the Visual Studio options have been set to not include the default protection against buffer overflow.
- Put a breakpoint at the "`i = 0;`" statement around line 15.
- Start debugging the program by clicking the green arrow or pressing F5.
- When the debugger is in the `doit` function, hover the mouse over the following names and record the addresses:

main 0X______________________________

str 0X______________________________

When in the `doit` function, the stack of the program contains the following information:

str array	frame pointer	return addr
0	4	8

- Open the memory window under Debug → Windows → Memory → Memory1. Look at the memory of the stack by entering the address of `str` in the address box. The `str` array holds four characters. Following this variable in memory will be the four byte frame pointer. This will be an address on the stack. Numerically, it should not be very different from the address of `str`. Remember that the Intel processor is a Little Endian machine and the addresses are in reverse byte order. After the frame pointer will be the return address. Since the doit function was called from the main function, the return address should not be much larger than the start address of the main function. If your memory looks like this, you will know that the 8th to 11th bytes from the start of your input file will overwrite the return address. Record the address of the frame pointer and return address.

frame pointer 0X______________________________ *(remember to reverse the bytes)*

return address 0X______________________________ *(remember to reverse the bytes)*

- The following was taken from the assembler listing file (generated by previously turning on the assembler listing option) showing the machine language of the program. In the main function the call to printf was compiled to:

```
          printf("Normal end of the program\n");

00026     68 00 00 00 00     push  OFFSET "Normal end of the program\n"
0002b     ff 15 00 00 00 00  call  printf
00031     83 c4 04           add   esp, 4
```

Bytes `0X27` to `0X2a` and `0X2d` to `0X30` are zero in the listing because the compiler did not know the address of the character string or the printf function at compile time. These bytes will contain the proper addresses when viewed in the program's memory. Our exploit is to print our name on the screen, so we will need to use the printf function.

- Add 0X2b to the address of the main method that you determined earlier. (Use hexadecimal addition.) Enter this address in the memory address window. You should see the machine language for the call instruction starting with "`ff 15`". The four byte address of the printf function appears two bytes after the address you calculated above (after the `ff` and `15`). The address was unknown at compile time, but appears in the debugger memory view. Record the address of the printf function.

  printf 0X__________________________ *(Remember to reverse the bytes.)*

You can create a data file that will overwrite the stack information starting at the str array. The format of your data file is given below. The numbers give the distance from the beginning of the file in hexadecimal.

str array	frame pointer	address of exploit program	instruction to push address of name on the stack	instruction to call printf	mov ecx, return address	jmp ecx	Your name
0	4	8	C	11	17	1D	1F

The system will notice if the frame pointer is changed, so your data file should contain the same value that was originally there. The return address is overwritten with the address of the machine language you are entering, which are the next bytes you are loading onto the stack. The first machine language instruction pushes the address of your name of the stack. This will be a parameter to the printf method. The next machine language instruction calls the printf method. After printing your name, the program jumps back to where the doit method was originally called.

- Using the Frhed hexadecimal editor, edit your input file to include the following. Frhed gives the offset of the cursor position in the lower left corner. Remember that the Intel processor is a Little Endian machine and the addresses are in reverse byte order.

offset	**Value**
0	4 bytes of anything as data for str.
4	Enter the frame pointer. *Remember to reverse the bytes so the least significant is first.*
8	Copy the address of the str variable ____________________ Add 0X0C ____________________ This is the start address of your exploit program on the stack. Enter this in the file in reverse byte order.
C	Enter “`68`”, the opcode of the Intel `push` instruction
D	Copy the address of the str variable ____________________ Add 0X1F ____________________ This is the address on the stack of the text containing your name. Enter this in the file in reverse byte order.
11	Enter “`ff 15`” the opcode of the Intel call instruction
13	Enter the address of the printf function. *Remember to reverse the bytes.*
17	Enter “8d 0d” the opcode of the Intel `LEA exc` (Load Effective Address) instruction
19	Enter the return address. *Remember to reverse the bytes.*
1D	Enter “ff e1” the opcode of the Intel `jmp ecx` instruction
1F	Text of your name. At the end of your name, enter 0X00 to terminate the string.

- Set a breakpoint on the return statement in the main method to keep the output window from disappearing when the program terminates. Run the program in Visual C++ and see if it prints your message. You name should appear in the console output of the program.