Intro- what is hacking? Vulnerability

First to understand hacking, we can use a simple example that was presented in the lecture. A simple driving program that allows user to drive on the road and then when it reach a split in the road, the program will be expecting the user to input a direction. The developer of the program may design the program to expect a simple left or right as input, however, what happen when the user enter straight? This may have a few consequences 1. Program crashes or 2. Program behaves in a way that allow user to gain control and go around the intended program. This is a vulnerability in the program and an exploit that hackers can use to gain access to the program that the developer did not consider. So in short, by thinking out side of the box, hackers can find vulnerabilities in a program and exploit that program for their purpose of either redirecting users, placing bugs, steal information…etc.

Registers and Memory

In a computer system the CPU uses assembly language that perform simple operations and since the CPU executes instructions so quick that they need to have a location to store values and be able to perform operations on them quickly and thus, the memory registers are created to overcome this. There are 8 registers in a computer system and they are <b> EAX, ECX, EDX, EBX, ESP, EBP, ESI </b> and <b> EDI. </b> These registers have different purpose for example the ESP and EBP registers are called stack registers, they are used to point to the top and the bottom of the current function stack. (A stack is referred to stacking the value or address being stored on top of each other.) The ECX is usually used for counting or counters. The ESI and EDI are called index registers and they are used to store pointers that holds the location of the source and destination of a string operation. The EAX us a general purpose register and it the mostly used to store current return value of a function.

WinDbg- windows debugging application used to pause a program and see its step by step action.

In the lecture, WinDbg is introduced as the main tool that is used for debugging applications, it is a very useful tool for you to pause a program and see its step by step action, see all the memory address and what values are in those memory address.   
  
We were given 3 lessons to try out with WinDbg.   
In lesson 1, we set a breakpoint and allow the debugger to run and let breakpoint to be triggered and we then learn to look at the location of the address, we learn to look at the stack, how much space is allocated for the values on the stack and we learned how to look at the value that is on the stack.  
In lesson 2, we learned how the stack works and to find vulnerability and exploit it. First we learned how to find the address of the instruction in the calling function and find out how many arguments are passed to the function and learned how function declaration and call looks like. Next we learned what registers contain attacker-controlled data and which point to attacker-controlled data. We then find out what is the address that contain vulnerability and the size of the buffer. After finding all those information we can then trigger the vulnerability to see if we have control over the eip register and then we can replace the address to go to the address where our shellcode is and execute our exploit.  
In lesson 3, we will explore the vulnerability in the heap. First we need to find the size of the heap and then we find where is the heap block pointed to and then we can find the address. Next we can find the size of the freed element and then we filled it with string of the same size and then we fill the heap so we can have an estimate of the location of our shellcode. Then we can pass in the address that will then run our exploits.

Conclusion

In conclusion, I learned more detail of how the computer memory works and by exploring those memory we can use it to find vulnerability in the system and therefore, allowing exploits to be executed. Before this week’s lecture, I never used WinDBG and now I feel like I have learned a new tool that teach me how to find vulnerability in a program. The lab lessons taught me so much about how to look at the memory and addresses and then getting access to those registers to store or point to the address we want so it will execute the exploits. I think overall this lecture taught me the importance of keeping your system updated so those vulnerabilities can fixed.

UPDATE