Matt Ritchie

Special Topics – Cyber Security

Buffer Overflow Vulnerability Lab

2-12-18

In the stack.c file, with the method

int bof(char \*str){

char buffer[**12**];

/\* The following statement has a buffer overflow problem \*/

strcpy(buffer, str);

return 1;

}

If I run this in this default state as mentioned/corrected in the words alter on, the buffer overflows just nicely. When I changed it to char buffer[**48**]; however, the buffer did not break, so the attack code didn’t get a chance to execute. Custom code not written at this point.

Task 1: this is the code I stuck in at the /\* You need to fill the buffer with appropriate contents here \*/ line

//----------------------start buffer filler----------------------

char \*cptr;

long \*lptr,temp;

int shellSize = sizeof(shellcode);

int buffSize = sizeof(buffer);

cptr=buffer;

lptr = (long \*) cptr;

temp = get\_sp()+offset;

for (int i = 0; i < 300; i+4) {

\*(lptr++) = temp;

}//fills in 300 units of crap

for(int i = 500; i < shellSize + 500; i++) {

buffer[i] = shellcode[i-500];

}//injects shellcode into buffer array

buffer[buffSize - 1] = '\0';

//-----------------------end buffer filler-----------------------

Task 2:

Yeah it didn’t work, because I’m not good with this kind of stuff I probably wrote a bunch of junk. Set up the loop, left it run for an hour and no activation, so I don’t think it’s a good badfile.

Task 3:

No errors thrown, but the injected bad code didn’t run either, so I figure the stack guard worked and I didn’t overflow the buffer.

Task 4:

I could not get a shell. Could be because of bad code, could be because once injected onto the stack, the badfile code was not permitted to execute on nonexecutable stacks. I tried the same trick as in task 2, but there was no result in 15min.