**Lecture 4 Notes**

If you have a register that is 64 bits, you can divide it in half, for sample:

EAX 0 – 31 bits and RAX 32-64 bits.

When a call instruction is done, the next instruction after it in memory address is added to the RIP (Instruction Pointer) so it knows where to return to after the function completes

RBP = Bottom of the current stack frame (Base Pointer)

Whenever you see RBP-0xFFF it usually means its dealing with a local variable, like placing it on the new stack or stack frame.

x/gx = examine giant word

pwntools function: flat(offset: <var here>) \*\*Can also use more offsets are arguments\*\*

arguments for functions are stored in registers to set them up before a function call

shell pwn cyclic -n8 <size of pattern here>

info frame [ shows the current stack frame registers]

**Stack4**:

& 0xFFFFFFFF00000 the zeros will zero out any return address

Need to figure out what part of memory corresponds to the != 0x7FFFFFF000000

We want the return address to point to the stack

Don’t forget to use checksec to see what possible attacks could be done on a binary

RNX: Has RNX Segments means it has regions of memory that can be read from and executed from

Code injection attack is what’s needed for Stack 4

Memory must be designated as executable to be treated as code. The Instruction pointer can also have executable code.

Can manipulate a return address which could force the instruction pointer to become shellcode that can open a shell for us

Give it a string through the call to gets, needs to overrite the safe return address and supply a string sequence of instructions (assembly instructions) to open a shell

Want the return value to point to the start of the buffer

Could use system calls in our injected code. We need to open the flag file, read from it to a buffer, and then write to standard out, and then close the file.

Need to use open and sendfile system calls to solve the challenge

Need to hand code assembly in our string that we’re passing to the initial buffer

Need to set the argument registers in order to build the correct calling convention

Sys\_call does not need an address as an argument like a normal call, instead we give it a sys\_call number using the argument register