-We ssh into smith for easier access of the files.

-Within agent-smith.c, we noticed that it allocated 128 bytes for “msg” and it tried to limit the value of “size” to be less than 128, through the if condition.

-size\_t n = fread(&size, 1, 1, file); This code allows us to read the first byte of data from the file into the address of “size”. Therefore, we can make use of this code to assign a value for “size”.

-n = fread(msg, 1, size, file); This code reads 1 byte of data at a time from the file into the memory allocated for “msg”. “size” determines the number of bytes to read from the file into “msg”.

-As long as the value of “size” is larger than 128, we will be able to cause a stack overflow. However, the ‘if’ condition is preventing us from directly entering a number that is larger than 128 for “size”.

-But we noticed that “size” is a signed data type while fread() takes in a unsigned data type. Therefore, we can input a negative value for “size” and it will still pass the ‘if’ condition while converting to a large number during fread(), allowing us to exploit stack overflow.

-From the gdb, we set a breakpoint at the method “display”.

-After that, we ran the program and used “print &msg” to get the address of “msg”, which is 0xbffffab8.

-We also used “info register ebp” to get the address of ebp, which is 0xbffffb48.

-From the 2 addresses, we know that the ebp is 144 bytes above “msg”.

-The rip is right above ebp, and we need rip to point to the address right above itself, where we insert the shell code. (8 bytes above the ebp is the address that rip should store)

-We also have to remember that size\_t n = fread(&size, 1, 1, file); reads away 1 byte of data to initialize the value of “size”.

-Therefore, we input “\xff” as the first byte to initialize “size” to a negative number (becomes a large positive number when it is unsigned), then add in 148 bytes of junk, and add in the address for rip, then the shell code.

-Input becomes: (\xff to initialize the value of size) + (148 bytes of junk, which we used 148 “a”) + (return address for rip, which is ebp address + 8 bytes) + (shell code)

-This causes the rip to point to the shell code we inserted and spawns a shell script