Lab 2

Exercise A - Format String Error

Input “%x” separated by a “-” 10 times to print the contents of the stack in hexadecimal. (the last hex will be the answer)

```
echo $((0xhex_num)) # e.g. echo $((0xffffe1f8))
```

This command can then be used to convert the hex into decimal which can then be used as input to the program.

Exercise B - Stack Buffer Overflow

1. Follow the usual steps to attach **GDB** to the program.
2. Use the command `disassem admin` to get the address of the first instruction executed during the `admin()` function.
3. After this use the command `disassem foo` and put a breakpoint at the `retq` instruction using `b *foo+<retq_number>`
    a. We do this because we need to know what part of input overwrites the `$rbp` (frame base pointer) address. This needs to be done because the return address is pushed right before `$rbp` is pushed on the stack. (function prologue)
    b. The breakpoint is placed at `retq` because it is preceded by `leaveq` which pops the value of `$rbp` placed during function prologue back into it (in our case which has been overwritten)
    c. After continuing till the breakpoint, we can use `info reg` to print values of all registers. This will show the part of our input that overwrites the `$rbp`.
        i. Use the input “AAAABBBBCCCCDDDDEEEEFFFFGGGHHHHIIIIJJJJKKKK”
        ii. This makes it easier to detect which part of the input overwrites the `$rbp` register.

iii. We need to replace the segment of our input (step 3.c.i) after the one present in `$rbp` with the address of the `admin()` function instruction address. This new address will be placed in `$rip` (instruction pointer) hence executing our desired function. This is the part where we overwrite the return address with a new address.

```
# input = AAAABBBBCCCCDDDDEEEEFFFFGGGHHHHIIIIJJJJKKKK
# e.q. $rbp = 0x4343434342424242 - intel uses little-endian,
# This means all Bs (0x42 - ASCII of B in hex) and Cs (0x43) overwrite
# the $rbp register.
# So we need to keep AAAABBBBCCCC and replace the next segment the admin()
# address
```

d. Use the steps present in the pseudocode in bof-admin.c to encode the value of the `admin()` function instruction address into the input.

e. Run the program with the encoded input.

Information about function prologue and epilogue.