8 Cybersecurity (fms27)

Consider the following hacking game. You are given access, as user `player`, to a 32-bit x86 machine. The C program `challenge` (setuid `boss`) accepts user input and contains a buffer overflow vulnerability. The stack is executable, there are no stack canaries in the binary and ASLR is turned off. When the `challenge` program is running, the first byte of its vulnerable buffer (always word-aligned) may appear in memory anywhere within a 32768 byte region starting at `0xffff6000`. The vulnerable buffer is 100 bytes long and the C function that contains it has only two more local variables, both `char`. The `challenge` program truncates its input at 2048 bytes.

Your input to the `challenge` program, including a specific 50-byte payload, is built by the following Python function. You supply five parameters to `buildInput` to create an input and then the `challenge` program is run on the resulting input, but doing so costs you \$0.01 per byte of generated input plus \$50 per run. You may repeat this as needed. You win a large prize if you cause the payload to be executed under userid `boss`. You seek a winning strategy that maximises your profit.

```
def buildInput(length, payloadOffset, retOffset, retAddress, retCopies):
    payload = bytes(b"\x31...") # 50 bytes of evil x86 machine code
    input = bytearray([0x90] * length) # prefill with nop
    input[payloadOffset:payloadOffset+len(payload)] = payload
    for j in range(retCopies): # stack spraying
        input[retOffset+(j*4):retOffset+(j*4+1)] = \
            retAddress.to_bytes(4, byteorder='little')
    return input
```

(*a*) Explain whether you would favour more runs or longer inputs. [1 mark]

(*b*) Explain whether, in your input, you would put the nop sled and payload before or after the stack spraying region. [3 marks]

(*c*) Explain whether you would favour a longer nop sled or a longer stack spraying region. [3 marks]

(*d*) Outline, with justification, an efficient strategy for guaranteeing a win in this game, detailing among other things how many *runs* (invocations of the `challenge` program) it involves in the worst case. [5 marks]

(*e*) Give, with justification, the specific values of each of the five input parameters your strategy will pass to `buildInput()` during run i, with runs numbered from 0 onwards. The values may be constant or parametric in i. [8 marks]