CSC 2400: Buffer Overflow Attacks

Consider the following program that uses the unsafe **gets** function:

Executable bufattack

```
void Echo()
{
  char buf[4];
  gets(buf);
  puts(buf);
}

void Fire()
{
  printf("\nYou called Fire?!\n");
  exit(1);
}

int main()
{
  printf("\nType a string:\n");
  Echo();
  return 0;
}
```

For each of the following exercises, use **sendstring** to convert a hexadecimal string into a byte string that will serve as input to your program:

a) Provide an exploit string (as hexadecimal digits) that forces **Fire** to execute when **gets** terminates (see disassembled code on next page). Try it on **felix**.

b) Add two arguments to **Fire**, as shown below. Then provide an exploit string that forces **Fire(11, 22)** to execute when **gets** terminates.

```
void Fire()
{
  printf("\nYou called Fire(%d, %d)?!\n", x, y);
  exit(1);
}
```

Disassembled code for function **main**:

```
main:
        ...
        0x08048468 <main+24>:    call   0x8048340 <puts@plt>
        0x0804846d <main+29>:    call   0x8048414 <Echo>
        0x08048472 <main+34>:    mov    $0x0,%eax
        ...
```

Disassembled code for function **Echo**:

```
Echo:
        0x08048414 <Echo+0>:     push   %ebp
        0x08048415 <Echo+1>:     mov    %esp,%ebp
        ...
        0x0804841a <Echo+6>:     lea    -0x4(%ebp),%eax
        0x0804841d <Echo+9>:     mov    %eax,(%esp)
        0x08048420 <Echo+12>:    call   0x8048320 <gets@plt>
        ...
        0x08048430 <Echo+28>:    leave
        0x08048431 <Echo+29>:    ret
```

Disassembled code for function **Fire**:

```
Fire:
        0x08048432 <Fire+0>:     push   %ebp
        0x08048433 <Fire+1>:     mov    %esp,%ebp

        0x08048435 <Fire+3>:     sub    $0x8,%esp
        0x08048438 <Fire+6>:     movl   $0x8048544,(%esp)
        0x0804843f <Fire+13>:    call   0x8048340 <puts@plt>

        0x08048444 <Fire+18>:    movl   $0x1,(%esp)
        0x0804844b <Fire+25>:    call   0x8048350 <exit@plt>
```

Debugging Example:

```
        break *(Echo+12)          #prior to the gets call
        display /i $eip
        run
        x /3xw $ebp-4
        nexti                     #type in 1234abcd5678
        x /3xw $ebp-4             #note the change
                                  #identify your input

        # Build your exploit string, so that Fire overwrites RA(main).
        # Write your exploit string in a text file, say in.txt.
        # Use ./sendstring < in.txt > in.raw, to convert text to binary
             (executable sendstring is available in /tmp on felix).
        # Pay careful attention to the big endian format.
        # Rerun the code with your exploit string as input:

        run < in2.raw

        # If your exploit string is correct, Fire will get called.
```