THE UNIVERSITY *of* NEW ORLEANS

DEPARTMENT OF COMPUTER SCIENCE

CSCI 2467, Spring 2020
Class Activity: More overflow practice

The program for this activity can be found in `/home/CSCI2467/labs/bonustarget/btarget`.

Level 1

The goal of this activity is to input a string that causes the program to call `win(0x15213)`, and thereby win a cookie.

1. Where is long before stored on the stack? What about long after?
2. How many bytes can Gets() copy before overwriting something?
3. If the user types "12345678\n", what will the resulting stack look like? (Fill in the stack diagram on the back.) What will the corresponding value read from %rdx be?
4. How can you use GDB to check if your buffer overflow worked as intended?

Level 2

For a little bit more of a challenge: Can you figure out how to call `win(0x18213)` for two cookies?

1. Which lines of assembly correspond to win(0x15213) and win(0x18213)?

Level 3

If you finished the other activities early, see if you can manage to call `win(0x18613)`!

1. Note the suspiciously named function `gadget1`. Does it obey calling conventions by preserving the stack pointer when it returns? What value will it place into %rdi?

```
4006b5: sub     rsp,0x38
4006b9: mov     QWORD PTR [rsp+0x28],0xb4
4006c0:
4006c2: mov     QWORD PTR [rsp+0x8],0xaf
4006c9:
4006cb: lea     rdi,[rsp+0x10]
4006d0: call    40073f <Gets>
4006d5: mov     rdx,QWORD PTR [rsp+0x28]
4006da: movabs  rax,0x3331323531
4006e1:
4006e4: cmp     rdx,rax
4006e7: jne     4006f3 <solve+0x3e>
4006e9: mov     edi,0x15213
4006ee: call    40064d <win>
4006f3: mov     rdx,QWORD PTR [rsp+0x8]
4006f8: movabs  rax,0x3331323831
4006ff:
400702: cmp     rdx,rax
400705: jne     400711 <solve+0x5c>
400707: mov     edi,0x18213
40070c: call    40064d <win>
400711: add     rsp,0x38
400715: ret
```

```c
void solve(void) {
    long before = 0xb4;
    char buf[16];
    long after = 0xaf;

    Gets(buf);

    if (before == 0x3331323531)
        win(0x15213);

    if (after == 0x3331323831)
        win(0x18213);


}
```

Table 1: Code

	7	6	5	4	3	2	1	0	Notes
0x602058	00	00	00	00	00	00	00	00	Return Addres
0x602050									
0x602048									
0x602040									
0x602038									
0x602030									
0x602028									
0x602020									

Table 2: Stack Diagram