Attack Lab Bootcamp Handout

Tue, Jun 7, 2022

To download the activity, enter into a Shark machine:

```
$ wget https://www.cs.cmu.edu/~213/activities/attack-lab-activity.tar
$ tar xvf attack-lab-activity.tar
$ cd attack-lab-activity

$ gdb activity
```

Activity 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]. Work with your group to fill in the stack diagram, and discuss:

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 "`abcdefgh12345678\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?

Activity 2

We've upped the stakes! Can you figure out how to call `win(0x18213)` for two cookies?

1. Which lines of assembly correspond to win(0x15213) and win(0x18213)?
2. Which value will the retq instruction read off of the stack? Can it be overwritten?

Activity 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?

[1] Actual availability of cookies is neither guaranteed or implied. However, there are always plenty of stack cookies available for you to choose from!

Code for solve()

```
0x4006b5 <+0>:   sub    $0x38,%rsp
0x4006b9 <+4>:   movq   $0xb4,0x28(%rsp)
0x4006c2 <+13>:  movq   $0xaf,0x8(%rsp)
0x4006cb <+22>:  lea    0x10(%rsp),%rdi
0x4006d0 <+27>:  callq  0x40073f <Gets>

0x4006d5 <+32>:  mov    0x28(%rsp),%rdx
0x4006da <+37>:  movabs $0x3331323531,%rax
0x4006e4 <+47>:  cmp    %rax,%rdx
0x4006e7 <+50>:  jne    0x4006f3<solve+62>
0x4006e9 <+52>:  mov    $0x15213,%edi
0x4006ee <+57>:  callq  0x40064d <win>

0x4006f3 <+62>:  mov    0x8(%rsp),%rdx
0x4006f8 <+67>:  movabs $0x3331323831,%rax
0x400702 <+77>:  cmp    %rax,%rdx
0x400705 <+80>:  jne    0x400711<solve+92>
0x400707 <+82>:  mov    $0x18213,%edi
0x40070c <+87>:  callq  0x40064d <win>

0x400711 <+92>:  add    $0x38,%rsp
0x400715 <+96>:  retq
```

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

  Gets(buf);



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

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

}
```

Stack diagram:

	7	6	5	4	3	2	1	0	Notes
0x602058	00	00	00	00	00	40	07	83	Return Address
0x602050									
0x602048									
0x602040									
0x602038									
0x602030									
0x602028									
0x602020									