Assignment 4, Fall 2015
CS 4630, Defense Against the Dark Arts
Buffer Overflow Attack
(Arcus Injectus Mutatio Nota Curse)

Purpose
In this assignment, you will learn how buffer overflow vulnerabilities can be exploited.

Due Date
This assignment is due on Wednesday, 10/07/2015, at 10:00am.

Prerequisites to Review
1. Review the “Stack Buffer Overflow” slides.
2. Read the article “Detection and Prevention of Stack Buffer Overflow Attacks”.
3. Download the code *dumbledore* from the Collab and examine its operation using *objdump* and *gdb*.

Assignment Details
1. This assignment **must** be completed using the Ubuntu 12.04 LTS OS you installed on your VM. This environment is where we will test your submitted code. It is possible, due to the sensitivity of vulnerabilities to the operating environment, that exploit code developed for one environment will work not correctly in a slightly different environment.

2. The class Collab site has the file that you should download. You **must** attack the supplied version of *dumbledore*.

3. Ubuntu has ASLR (Address Space Layout Randomization) turned on as a defense. To simplify your task, we will turn it off. The command

   ```
   $ setarch i386 -RL /bin/bash
   ```

   will disable ASLR for that shell. It does not affect any other shells.

4. Examine *dumbledore*. It contains an obvious buffer overrun vulnerability in the readString function

5. Create a file named *grade.txt* that contains your name and run *dumbledore*.

   ```
   $ ./dumbledore <grade.txt
   ```

   ```
   Thank you, Wei Wang.
   I recommend that you get a grade of D on this assignment.
   ```

6. Your ultimate goal is to attack this program by exploiting the buffer overrun vulnerability so that the program execution is as follows.

   ```
   $ ./dumbledore <gradeA.txt

   Thank you, Wei Wang.
   I recommend that you get a grade of A on this assignment.
   ```

   If you can demonstrate your attack and make it do the above (with your name in place of mine), you will receive a grade of “A” on this assignment.

7. Please read the following requirements completely before beginning the assignment.

Requirements

1. Your first step should be to analyze the *dumbledore* binary using *gdb* and *objdump*.

2. Next you should analyze *readString* ’s stack frame. To do this, you need to set a breakpoint in *readString* .

3. Your next task is to get the program to crash. Write a C program named *attack-crash.c* that produces a *data.txt* file, as simple as possible, that causes grader to generate a segmentation fault. Make sure you understand why the program is crashing. To get you going, here is a program that creates a legal input.

   ```
   #include <stdio.h>
   #include <string.h>
   char attackString[] = "Bill Smith\n";

   int main() {
    int i;
    char *p = attackString;

    for (i = 0; i < sizeof(attackString); i++) {
    putchar(*p);
    p++;
    }
    return 1;
   }
   ```

   Here is a sample run.

   ```
   $ ./attack-gradeC >data.txt
   $ ./dumbledore <data.txt
   Thank you, Bill Smith.
   I recommend that you get a grade of D on this assignment.
   ```

4. The final task is to get the program to give you a grade of “A”. Write a program named attack-gradeA.c , that produces an input file (called grade.txt), as simple as possible, that causes the grader program to print your name and recommend a grade of “B” or “A”. You can see by reading the program that if your name is “Wizard in Training”, this task is easy. However, your name is probably not “Wizard in Training”, so you will need to figure out how to exploit the

vulnerability to get a grade of “A”.

Miscellaneous Notes

1. Submit your generated input file via the Collab. It must be named *grade.txt*.

2. You can also use *perl* to generate the attack input file, like we did in class. *Python* also works.

3. This assignment is pledged.

4. Recall the command *objdump -d*. This command will disassemble a file. This command, combined with *GCC*, is useful for seeing the binary encoding of instructions.

5. Review the commands for assembly debugging in GDB, they will help you a lot.

6. I found it useful to draw a picture of the stack when the program was executing inside function readString.

7. This assignment was adopted from one given at Princeton in COS 217 by Andrew Appel and one previously given in CS 4630 Spring 2010.

Items to Submit and Due Date

1. Upload your *grade.txt* file. By doing very little on the assignment, you can receive a grade of “D”. (The actual score recorded for grade computation purposes will be a 60.)