
Problem Set 4

This problem set is due *via email*, to `6857-hw@mit.edu` on *Monday, April 14* by the beginning of class.

You are to work on this problem set in groups of three or four people. Problems turned in by individuals, pairs, pentuples, etc. will not be accepted. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration. If you do not have a group, let us know.

Homework must be submitted electronically! Each problem answer must appear on a separate page. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for L^AT_EX and Microsoft Word on the course website (see the *Resources* page).

Grading and Late Policy: Each problem is worth 3 points. Late homework will not be accepted without prior approval.

With the authors' permission, we will distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this on your homework.

Problem 4-1. Buffer overflow

The TAs are sick of sending e-mail to every homework group, so they're testing out a new system for entering and tracking problem set grades. Your group should have received, by e-mail, a username and password for our shiny new grades server, located at `6857.csail.mit.edu`. In your home directory, there is a directory named `grades`. The idea is that the TAs will drop text files containing your grades into your `grades` directory. Then, you would SSH into the server and look in the `grades` directory to find your grades.

Naturally, you don't have permission to write to your `grades` directory. You didn't think the TAs were *that* careless, did you?

Actually, it turns out the TAs are a little careless. You will find, in your home directory, a program named `hello`, which for some reason *does* have permission to write to your `grades` directory. (Alas, so much for the principle of least privilege.) The source code of `hello` follows:

```
#include<stdio.h>

int main() {
    int leet=0;
    char buf[80];

    puts("Hello, What's your name?");
    gets(buf);

    printf("Hello, %s!\n", buf);

    if(leet == 31337) {
        FILE *fp;
        fp = fopen("grades/my-grades.txt", "w");
        fprintf(fp, "Sweet! I got some points!\n");
        close(fp);
    }

    return 0;
}
```

Actually, it turns out the TAs are really, incredibly careless (as you learned from the two-time pad problem in Problem Set 1). You point out to the TAs that `gets` is inherently unsafe. The TAs mutter something¹ in response, and then walk away.

Because the TAs won't listen to you, you decide to resort to more drastic measures to get their attention. Giving yourself a few hundred bonus points ought to do the trick...

- (a) In fact, `gets` is so incredibly unsafe that `gcc` will throw a warning if you try to use it. Explain why the `gets` function is unsafe.
- (b) Execute a buffer overflow attack to overwrite the local variable `leet` with the value `31337`, thus snagging yourself some sweet points. Include in your solution the string you used in your attack.
- (c) Explain how you would craft an input string that would execute a shell, thus allowing you to write whatever you want into your `grades` directory. (*Note: you don't have to actually execute the attack for this part.*)
- (d) Suggest several ways that the code author or the compiler could prevent the attack in part (c).
- (e) **Optional:** For a bonus point, execute the attack in part (c) and leave us an interesting note in your `grades` directory. Include your code.

The server is an x86 machine, and the programs were compiled with `gcc 4.1`. If you, for some reason, want to know about other specs of the machine, ask the TAs.

For a nice overview of buffer overflow attacks, read "Smashing the Stack for Fun and Profit", by Aleph One, available on the course website.

Problem 4-2. Investigating Malware

Research two viruses/worms: the one corresponding to your group number (same number as in the previous problem) from the table below, as well as one more of your choice (from the table or otherwise, so long as it is interesting and/or famous). A good place to start looking is the CERT advisory database, located at www.cert.org. We recommend researching several of these, but you need only submit writeups for two.

Group number	Malware Name
1 or 10	Nimda
2 or 11	Blaster
3 or 12	Sobig.F
4 or 13	Code Red
5	Sircam
6	Bagle
7	Trojan Horse Sendmail Distribution
8	MyDoom
9	Santy

For each of your two viruses/worms, answer the following questions in technical detail:

- What is the payload, if any?
- What is/are the specific mechanism(s), if any, by which the malware reproduces itself?
- What systems are vulnerable to infection?
- What kind of damage does the malware cause to computers and networks?
- What preventative measures can one take to avoid being infected?
- If one becomes infected, what measures should one take to recover from the compromise?

¹<http://www.crypto.com/bingo/pr>