Homework 5

CS161 Computer Security, Spring 2008

Assigned 4/23/08

Due 5/05/08

1. **Worm Propagation**

   In lecture, we talked about ways of increasing the propagation rate of worms. In this problem, we'll examine the effects of decreasing the propagation rate of worms.

   Recall that $i(t)$ is the proportion of machines in a network that are infected by a worm at time t, β is the contact rate, and T is a constant of integration that fixes the time position of the incident. We'll use a Random Spread (a.k.a Susceptible-Infected) model for worm propagation and assume a network of tens of millions of susceptible machines.

   Please limit each answer to 1-3 sentences. You may also include graphs or tables if you like, though they are not necessary.

   (a) (4 points) If β is 3.5 and T is 15, at what time are 50 percent of the machines infected? At what time will 99% of all machines be infected? Hint: An easy way to work through this problem is to use Mathematica, Excel, or OpenOffice to generate or graph your results.

   (b) (4 points) If we are able to reduce the initial infection rate to 0.5, what is the 50 percent infection time? What is the time for 99% of all machines to be infected?

   (c) (4 points) Sometimes a worm is initially distributed to a **hitlist**, a set of hosts known to be vulnerable. Once the hosts on the hitlist are infected, these hosts scan randomly to continue spreading. Consider a hitlist that makes up one percent of all vulnerable hosts. Modify the formula for i(t) to take into account the speedup gained by the hitlist.

2. **Honeypots and Tarpits**

   A tarpit is a honeypot that consumes as many of the adversary's resources as possible. For each of the following honeypots, describe a way we could turn the honeypot into a tarpit. Do not use the same answer for more than one part of this problem. Keep your answers short, no more than two or three sentences.

   (a) (3 points) An FTP server with no password. Located on the FTP server is a file with a tempting name, such as corporate_secrets.txt

   (b) (3 points) A host with many open ports.

   (c) (3 points) An open directory on a web server. The directory has a tempting name, such as /highly_proprietary_software/source_code.

   (d) (3 points) A compromised SSH server containing industrial control software to accompany already-stolen steel mill blueprints.

3. **Taint Analysis**

   In this question, you are to perform a taint analysis on the following code.

```
float area_circle(float radius)
{
        float pi=3.14;
        return pi*radius*radius;
}

float area_square(float width)
{
        return width*width;
}

int main(char** argv, int argc)
{
         float radius_1 = 7.2;
         float area_1 = area_circle(radius_1);
         float radius_2 = read_float_from_keyboard();
         float area_2 = area_circle(radius_2);
         float summed_area = 0;
```

```
        summed_area += area_1;
        summed_area += area_2;

        int n = read_int_from_keyboard();
        int fibonacci = 1;
        int i;
        for(i = n; i != 0; i--)
           {
           fibonacci *= i;
           }

        float (*pt2Area)(float);

        printf("Enter 'c' for circles, enter 's' for squares: ");
        char which_area = read_char_from_keyboard();

        if (which_area == 'c')
            pt2Area = area_circle;
        if (which_area == 's')
            pt2Area = area_square;

        char buf[42];
        printf("Enter a size: \n");
        gets(buf);

        float size = string_to_float(buf);
        float area_3 = pt2Area(size);
        printf("Area is: %f \n", area_3);
        printf("The ratio of area 3 to area 2 is: %f \n", area_3/area_2);

        return 0;
}
```

(a) (6 points)
To the right each line of code, write the names of any variables that became tainted due to that line of code executing.

(b) (6 points)
List any security vulnerabilities that exist because of tainted variables. For each vulnerability, give an example of an unsafe in-

put that would exploit the vulnerability, and explain what happens when this input is given.

4. **Symbolic Execution**

Consider the following code.

```
void f(void)
{
int step = 0; int user_increment = 0;
int start = 1;
printf("Would you like to count by 1? (y/n)\n");
char choice = read_char_from_keyboard();

printf("Start counting at: \n");
start = read_char_from_keyboard();

if (choice == 'n')
   {
   user_increment = read_int_from_keyboard();
   if (user_increment > (100 - start))
       printf("WARNING: You may not experience many iterations.\n");
}
if (choice == 'y')
   {
   step = 1;
   if (user_increment > 100)
       printf("WARNING: You will not experience many iterations.\n");
}

step += user_increment;

/* Place assertion here */

printf("Counting to 100, incrementing by %d. \n", step);
for(int i = start; i <= 100; i+=step)
    printf("Currently we are at %d. \n", i);
```

(a) (1 point) Write the assertion that must be true for correct execution.

(b) (11 points) Identify each path this code might take (up to the assertion). For each path, give the path predicate. Determine whether or not each path is feasible. For each feasible path, give an example of input that would cause this path to be executed. For each feasible path, write a symbolic expression that must be satisfied in order for the assertion to **fail**. Determine whether each symbolic expression is satisfiable. For each satisfiable expression, give an example of input that causes the assertion to fail.