COSC 366: Introduction to Cybersecurity: Assignment 1

Ground Rules. Work must be typeset (not handwritten and scanned) and submitted by 23:59:59 of the due date. You do not need to provide code or visual diagrams for any question, but are welcome to if it helps you explain your answers.

1. Security, Security Everywhere.

Consider the following three scenarios:

- a website that allows consumers to order products with their credit/debit card (e.g. Amazon).
- a social media website that allows anyone to sign-up and post content. **Hint** (for one possible security issue): think about what kinds of automated things could be done with the sign-up system and access to the network post-sign-up.
- an internet-connected thermostat that allows electric utility operators to adjust temperatures to regulate power supply based on demand (e.g. when demand is high, operators turn off heating units (example: https://www.wvlt.tv/content/news/KUB-offers-free-smart-thermostat-and-installation--566989511.html)).

Answer the following questions for each of these scenarios (short answers):

A. Who might want to attack the system?
B. What types of harm might they want to cause?
C. What kinds of vulnerabilities might they exploit to cause harm?

2. **Threat Models and Risk Assessment.** Suppose the course instructor has created a database of all information for this course: homeworks, exams, handouts, and grades. Create a detailed threat model for this database: what should the security goals be? What are reasonable attacks, and who are the potential attackers? What threats should we explicitly exclude from consideration?

Now assume that the database is stored on the instructor's personal laptop, with no network card and no floppy disk drive. Propose at least two security mechanisms that would help counter your threat model (e.g. file or disk encryption, a laptop lock, a safe to store the laptop, a kevlar laptop sleeve, relocation to Fort Knox . . .), and analyze the net risk reduction of both. You should justify your estimates for the various incidence rates and costs. While we do want to see numbers for this part, don't worry about figuring out exact costs or risk reductions, guess at some reasonable numbers but don't spend very long at this part of the assignment.

3. Finding vulnerabilities. Here are a few code excerpts. For this exercise, you'll find the vulnerability in each and describe how to exploit it.

1) Below is a short POST-method CGI script - it reads a line of the form “field-name=value” from standard input, and then executes the last command (in the line $result = ‘last ...‘) to see if the user name “value” has logged in recently. Describe how to construct an input that executes an arbitrary command with the privileges of the script. Explain how your input will cause the program to execute your command, and suggest how the code could be changed to avoid the problem.

```
#!/usr/bin/perl

print "content-type: text/html\r\n\r\n<HTML><BODY>\n";

($field_name, $username_to_look_for) = split(/=/, <>);

chomp $username_to_look_for;

$result = ‘last -1000 | grep $username_to_look_for‘;

if ($result) {

    print "$username_to_look_for has logged in recently.\n";

} else {

    print "$username_to_look_for has NOT logged in recently.\n";

}

print "</BODY></HTML>\n";
```

b) This is a short (and poorly written) C function that deletes the last byte from any file that is not the extremely important file /what/ever. Describe how to exploit a race condition to make the function delete the last byte of /what/ever, assuming the program has read and write access to the file (/what/ever) but the user does not. Your description should list what file the fixed string pathname refers to at each important point in the exploit, and why it will work. (You can read about the various system calls used below at https://man7.org/linux/man-pages/dir_section_2.html)

```
void silly_function(char *pathname) {
    struct stat f, we;
    int rfd, wfd;
    char *buf;
    stat(pathname, &f);
    stat("/what/ever", &we);
    if (f.st_dev == we.st_dev && f.st_ino == we.st_ino) {
        return;
    }
    rfd = open(pathname, O_RDONLY);
    buf = malloc(f.st_size - 1);
    read(rfd, buf, f.st_size - 1);
    close(rfd);
    stat(pathname, &f);
    if (f.st_dev == we.st_dev && f.st_ino == we.st_ino) {
        return;
    }
    wfd = open(pathname, O_WRONLY | O_TRUNC);
    write(wfd, buf, f.st_size - 1);
    close(wfd);
    free(buf);
}
```