Secure Programming: Formative Assignment

Christophe Petit
University of Birmingham

October 18, 2019

The purpose of this assignment is to provide a sample of potential exam questions, to help you gauge my expectations there[1]*. The assignment does not count for the final mark of the module. The assignment is due on November 7. Collective feedback will be provided during the lecture (probably at the first lecture hour on Monday of Week 7). Individual feedback will also be provided upon request.*

Question 1

Consider the following PHP code, which is part of a library website

```
$author = $_GET['author'];
$query = "SELECT (*) FROM books WHERE author  = '$author'";
$result = mysqli_query($query);
```

1. Explain what makes this code vulnerable, and give one potential consequence of this vulnerability.
2. Describe two techniques that can be used to protect against this type of attack.
3. Write a secure version of the code above.

Question 2

1. Explain what a HttpOnly cookie is.
2. Can HTTPonly cookies protect against Cross-Site Scripting attacks? justify your answer.

[1] Note that previous years' exams are also available in the Sloaman lounge

Question 3

1. Explain what the effective UID, real UID and saved UID are
2. Suppose the C code of Figure 1, compiled by root, is also executed as root. Describe the effect of this program. Justify your answer.

Question 4

What is the meaning of "defense-in-depth"? Give an example in the context of secure programming.

Question 5

Suppose an attacker manages to steal your Canvas session, or to exploit it in a cross-site request forgery attack. List all services and all sensitive information that they will have access to. Among all potential consequences, what will affect you most? What will affect the university most?

Figure 1: C code for Question 3

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

void testFile(unsigned int attempt) {
        char * filename = "/etc/shadow";
        FILE * f;
        f = fopen(filename, "r");
        if (f == NULL) {
                printf("Opening attempt %u failed!\n",attempt);
        }
        else {
                fclose(f);
                printf("Opening attempt %u succeeded!\n",attempt);
        }
}


int main(int argc, char ** argv) {
        int status;
        testFile(1);

        status = seteuid(500);
        if (status < 0) {
                fprintf(stderr, "setuid failed!\n");
                return -1;
        }
        testFile(2);

        status = seteuid(0);
        if (status < 0) {
                fprintf(stderr, "setuid failed!\n");
                return -1;
        }
        testFile(3);

        status = setuid(500);
        if (status < 0) {
                fprintf(stderr, "setuid failed!\n");
                return -1;
        }
        testFile(4);

        status = setuid(0);
        if (status < 0) {
                fprintf(stderr, "setuid failed!\n");
                return -1;
        }
        testFile(5);
}
```