Lab 4: Web Security

Submission policy. Part 1 is due on **Thursday, March 16, 2017** at **11:59PM** and Parts 2, 3 and the bonus are due on **Tuesday, March 28, 2017** at **11:59PM** via websubmit, following the submission checklist below. Late submissions will be penalized according to course policy. Your writeup MUST include the following information:

1. List of collaborators (on all parts of the project, not just the writeup)
2. List of references used (online material, course nodes, textbooks, wikipedia, etc.)
3. Number of late days used on this assignment
4. Total number of late days used thus far in the entire semester

If any of this information is missing, at least 20% of the points for the assignment will automatically be deducted from your assignment. See also discussion on plagiarism and the collaboration policy on the course syllabus.

Introduction

In this project, we provide an insecure website, and your job is to attack it by exploiting three common classes of vulnerabilities: cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection. You are also asked to exploit these problems with various flawed defenses in place. Understanding how these attacks work will help you better defend your own web applications.

Objectives

- Learn to spot common vulnerabilities in websites and to avoid them in your own projects.
- Understand the risks these problems pose and the weaknesses of naive defenses.
- Gain experience with web architecture and with HTML, JavaScript, and SQL programming.

Administration: This lab will be administered by Sean Smith.

Part 1. SQL Injection

Your first goal is to demonstrate SQL injection attacks that log you in as an arbitrary user without knowing the password. In order to protect other students' accounts, we've made a series of separate login forms for you to attack that aren't part of the main **Bungle!** site. For each of the following defenses, provide inputs to the target login form that successfully log you in as the user "`victim`" and a brief (2-3 sentences) explanation of why your attack works:

1.0 No defenses. [6 points]
Target: `http://cs558web.bu.edu/sqlinject0/`

1.1 Simple escaping. [6 points]
The server escapes single quotes (`'`) in the inputs by replacing them with two single quotes.

Target: `http://cs558web.bu.edu/sqlinject1/`

1.2 Escaping and hashing. [12 points]
The server uses the following PHP code, which escapes the username and applies the MD5 hash function to the password. (Hint: In MySQL when two binary values are compared, such as "$\backslash xd5S' =' \backslash xb2$" , the result is True.)

```
 if (isset($_POST['username']) and
isset($_POST['password'])) {
    $username = mysql_real_escape_string($_POST['username']);
    $password = md5($_POST['password'], true);
    $sql_s = "SELECT * FROM users WHERE username='$username' and pw='$password'";
    $rs = mysql_query($sql_s);
    if (mysql_num_rows($rs) > 0) {
        echo "Login successful!";
    } else {
        echo "Incorrect username or password";
    }
}
```

You will need to write a program to produce a working exploit. Please write a python script `sql_1-2.py` that prints out a single working password.

Target: `http://cs558web.bu.edu/sqlinject2/`

What to submit When you successfully log in as `victim`, the server will provide a URL-encoded version of your form inputs. Submit a single text file with the filename `sql.txt` containing these lines as well as the un-encoded form of the sql input you provided and a brief explanation of why it worked. For the last part, submit the single-file source code for the program you wrote as `sql_1-2.py` that prints the sql input, a brief description of how it works and the time it took to execute.