Security For Databases and Throughout Computer Science Curriculum

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Franceska Padilla Coo

Spring, 2024

Technical Project Team Members

Noah Cook

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Nada Basit, Department of Computer Science

Introduction

The goal of this capstone project was to generate three more exercises that gave additional practice in database and sql security. This capstone came about to address two issues 1) the need for practical and hands-on exercises to teach computer science and 2), the need to incorporate security practices earlier on in the computer science course of learning. For the first reason, one of the most effective ways students learn computer science logic, techniques, and understanding, is to see a practical application of their lessons. This capstone used a pre-existing teaching application and built on top of it, allowing students to scan code and familiarize themselves with the syntax needed to execute three different sql injection protection methods. Second, the need to incorporate security practices earlier on, is incorporated in this assignment because these three security methods meant to ensure proper database addition and obtainment of information are within a teaching application meant for an introductory course to databases. This means that, by introducing these practices earlier on, students and future computer scientists have familiarized themselves with certain security solutions and can incorporate them into their usual procedures, stressing the importance of security throughout the application process. Using an additional prepared statement exercise, a syntactical exercise using double quotes, and a regular expression (regex) exercise, this capstone project aimed at providing students with interactive introductory practice problems, stepping through the order of operations and syntax for a robust set of sql injection prevention tactics that they can continue using in more practical environments.

Research Methodology

To establish the best way to execute this capstone project, I received aid from Professor Basit about effective teaching methods. I learned about the importance of immediate exercises and practical implementation of computer science skills. As a relatively new topic within education, approaches have mirrored those within mathematics and science methodology. This means that approaches include exercises incorporating the skills outlined in the subject's textbook, such as snippets of code, functions, or entire programs. Professor Basit also provided specific advice on how to best gear the exercises towards the students. Because of the more complex nature of some of the security implementations, we pre-wrote a majority of the actual security code in the exercises. To make up for this transparent information and ensure that students still learn the step-by-step process to enact these security implementations, brief passages walking the students through the implementations and their components sit to the left of the exercises. Therefore, the resulting additions included three exercises with explanations on the left of the row and the actual exercises and answer submissions on the right. The exercises still employ a sense of interactivity by, instead of requiring the students to write the entire code from scratch, fill in the blanks provided and check their work against the answers shown after submission.

As for the actual generation of these exercises and their topics, two main sources were used to generate both the exercises and their explanations. The first new exercise using prepared statements walks the students through the makeup of the prepared statements. By separating the two variables of $username and $password, the student can see what the prepared statement class handles and how it binds the user input and variables with the sql query to be sent over the network. The escaping inputs exercise was generated after viewing the Hacksplaining article, which also outlines several ways people can securely execute their sql statements and tap into their databases. This approach was chosen because of its simplicity to understand the syntactical

tactic of escaping inputs. Hacksplaining also included regex as a type of input verification, and this was the last exercise created for students to practice. With two types of input verification and then also an imported security package, the total of four exercises allow users to handle both technical checks and checks against human error.

Initial State

Before adding these three exercises to the site, the sql injection prepared statements page consisted of only one prepared statements exercise. It included two given inputs for the users to try which tested for a regular username output versus a sql injection username output using a single end quote to finish the username input off and additional malicious code. Because of transition periods and initial access constraints, this was the one page added to and expanded.

Capstone

The new exercises added consisted of three new exercises: a new prepared statement exercise, an escaped input exercise, and a regex check exercise. The first prepared statement exercise differentiates itself from the initial exercise included in the page. In this exercise, the $username and $password user inputs are separated within the prepared statement steps. There are also explanations for the entire process. The user has to fill in four blanks within the prepared statement's prepare, bind, and execute process, implemented by creating an html form. The user's input is then checked against string literals incorporated in php code. By providing an explanation and encouraging users to fill in the blanks of pre-written code, the student gets to parse the entire structure and see where things are connected. It requires familiarity with the structure of the prepared statement. This wraps up the first instance of reinforcing how to implement security measures for user input database interaction.

Second Exercise

The code is the following:

```
$username = $_POST['username']
$password = $_POST['password']
$stmt = $mysqli->prepare('SELECT username FROM users_le
if ($stmt) {
$stmt->bind_param(':userN', [BLANK3])
$stmt->bind_param(':passW', [BLANK4])
$stmt->execute()
$result = $stmt->get_result()
}
```

Type in the answer here:

BLANK1

:userN

BLANK2

:passW

BLANK3

$username

BLANK4

$password

Search

Figure 1: Answer Key for First Exercise

The escaped input exercise bases itself in a more syntactical fashion. It encourages students and future developers to use double quotes when writing their prepared statements and sql codes that include user input. Using double quotes allows the developer to account for any single quote ending inputs and therefore prevent escaped input. Although one of the weaker forms of security, it was still good to instill this practice from an early career.

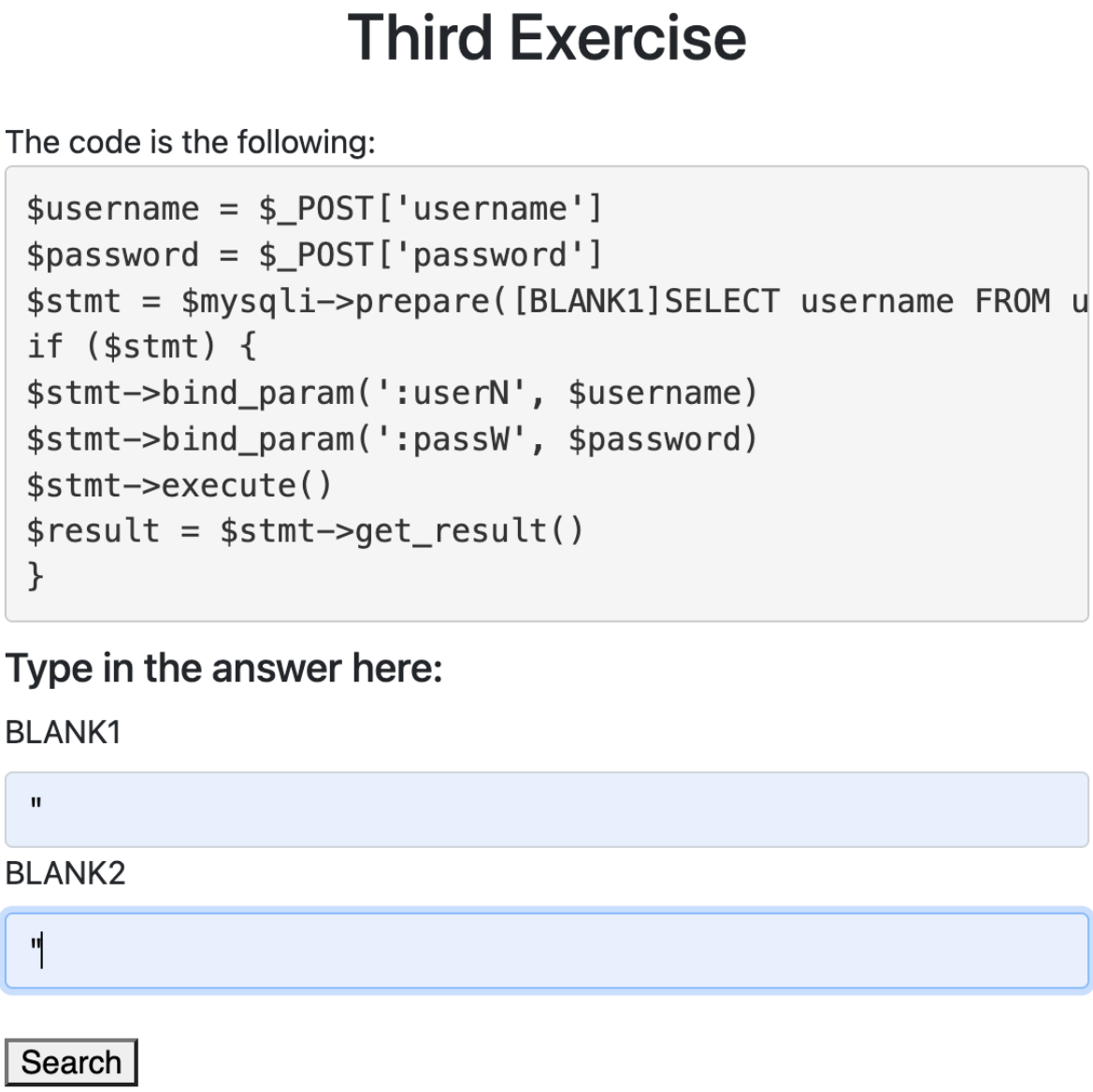

Figure 2: Answer Key for Third Exercise

The last exercise consisting of a regex helps tie the bridge between application level security and general programming practices that should always be incorporated. This is because the user input is checked for an alphanumeric regex, which can help prevent malicious user input. By using a regex, it reinforces that not only should developers restrict what type of characters make up a username and what type of characters can represent the username from the database level, but it also reminds developers of the ways they can check this user input through basic php and html. The more security thought throughout the entire development process can allow as many checks as possible to prevent security threats.

Figure 3: Answer Key for Fourth Exercise

Conclusion and Discussion

This capstone can be continuously developed to provide positive reinforcement and security explanations depending on the username input. At the moment it checks for a literal comparison of the user input, but it can be further developed to account for capitalization differences, completely wrong answers, and provide hints if prompted. This will allow the learning process to be encouraging, and the information learned can be followed through and reinforced.

Overall, this capstone project aimed to provide three new exercises to expand on possible database application security for the standard developer. To ensure that these skills are taught early and therefore incorporated and encouraged early, this capstone project uses a pre-existing application for students in introductory databases. Additionally, the interface is interactive and

calls for user participation through a hands-on activity with explanations, found to be effective among computer science students. Through this, future developers can learn how to incorporate database security early into their career and the many ways to prepare against any threats.

WORKS CITED

1. “Protecting Against SQL Injection,” Hacksplaining. https://www.hacksplaining.com/prevention/sql-injection.
2. “How to prevent SQL Injection Vulnerabilities: How Prepared Statements Work,” Security Journey. 11 February 2020, https://www.securityjourney.com/post/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work#:~:text=A%20prepared%20statement%20is%20a,safely%2C%20preventing%20SQL%20Injection%20vulnerabilities.