

# Introduction to the Secure Development Training Program



# Why are we here?

- Introduce the Secure Development Training Program
- Discuss why it's important
- Understand the program objectives
- See how developers access the course and learn how they will be enrolled
- View an introductory video
- Peek into the HackEDU learning platform and the modules developers will complete
- Outline the steps to receive a HackEDU certificate and a 'thank-you' gift

# Security Awareness Strategy



# Background (why this is important)

Lack of secure coding practices in applications development and maintenance is a **key vulnerability**.



What can we do? → Secure Development Training Program

## Program Objectives: All Web & App Developers

**Organization Objective** – Minimize the attack vector caused by insecure applications and coding in order to reduce the number of incidents from vulnerabilities found in our applications.

**Learning Objective** – Developers will learn top vulnerabilities and how to guard against them using secure coding practices. By the end of the program, the developer will demonstrate vulnerability patching.

# Program Objectives

**Help developers code securely.**

# Content Focus

Program consists of an introductory course in Y-Train as well as 11 courses taken on the HackEDU platform. These are based on the OWASP Top 10 Vulnerabilities and include:

- SQL Injection
- Command Injection
- Broken Authentication and Session Management
- Sensitive Data Exposure
- XML External Entities
- Broken Access Control
- Security Misconfiguration
- Cross-Site Scripting
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging & Monitoring

# Overall flow of the program (communications diagram)

## Pre-Launch



## Program Flow





# Introduction Video

Participants will go to Y-Train and watch this introduction as a course.

Enroll:

1- go to [ytrain.byu.edu](https://ytrain.byu.edu)

2- Search for Introduction to Secure Development

# HackEDU Secure Development Modules

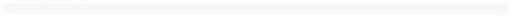
## Secure Development Training

This course will focus on the Open Web Application Security (OWASP) Top 10 vulnerabilities. These vulnerabilities are seen as the most critical security risks and they are plentiful in many web applications. This is a good starting point in web application security to understand how to exploit the OWASP Top 10 and how to protect against them.

### OWASP Top 10

SQL Injection LESSON

*Progress*



Command Injection LESSON

*Requires subscription*

Broken Authentication and Session Management LESSON

*Requires subscription*

Sensitive Data Exposure LESSON

*Requires subscription*

XML External Entities LESSON

*Requires subscription*

Broken Access Control LESSON

*Requires subscription*

Security Misconfiguration LESSON

*Requires subscription*

Cross-Site Scripting LESSON

*Requires subscription*

Insecure Deserialization LESSON

*Requires subscription*

Using Components with Known Vulnerabilities LESSON

*Requires subscription*

Insufficient Logging & Monitoring LESSON

*Requires subscription*

# SQL Injection

[Product Tour](#)

## Injection Introduction

Before diving into the hands-on portion of this lesson we will start with some background information on Injection, SQL Injection, and SQL Syntax. Then we will start the hands-on exercises with Reconnaissance.

An injection attack allows attackers to inject code into a program or query. Injection attacks come in many forms and we will explore both SQL Injection as well as Command Injection.

You will be using a browser and a web proxy. The proxy has the ability to stop all HTTP requests to the server so that they can be analyzed and modified before forwarding them to the server. There are many proxies that can be downloaded and used for security testing, but Burp Suite is one of the most heavily used. It comes bundled with Kali Linux and there is a free community version license available. Our proxy works the same as Burp Suite with simpler functionality for this intro lesson.

If you need a hint and see the Hint button, you can click it for a bit of additional information.

[Hint](#)

Proxy Status: LISTENING

Intercept Requests 

Popout Mode



Reset Sandbox

Target Application

<http://sandbox-hackedu.com/>

Go

[View Source](#)

## Social Media App

### App Sign In

[Sign in](#)

© Social



PROXY



HISTORY



CODE

OUTPUT &  
ERRORS

PATCHES



## HackEDU

[Product Tour](#)

## Injection Introduction

Before diving into the hands-on portion of this lesson, we will start with some background information on SQL Syntax. Then we will start the Reconnaissance.

An injection attack allows attacker to query. Injection attacks come in many forms, both SQL Injection as well as Command Injection.

You will be using a browser and a web proxy. The proxy has the ability to stop all HTTP requests to the server so that they can be analyzed and modified before forwarding them to the server. There are many proxies that can be downloaded and used for security testing, but Burp Suite is one of the most heavily used. It comes bundled with Kali Linux and there is a free community version license available. Our proxy works the same as Burp Suite with simpler functionality for this intro lesson.

If you need a hint and see the Hint button, you can click it for a bit of additional information.

[Hint](#)

## Browser Sandbox

This panel is the browser sandbox. It acts very similar to a normal browser. You can view the source code of the browser by clicking on "View Source". Many vulnerabilities can be found by viewing the web application front-end code. To get back to the browser click "View Browser".

[Skip Tour](#)[← Back](#)[Next →](#)

Target Application

http://sandbox-hackedu.com/ Go View Source

## Social Media App

# App Sign In

Username

Password

Sign in

© Social



HISTORY

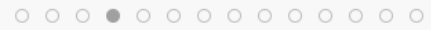


CODE

OUTPUT &  
ERRORS

PATCHES

# SQL Injection



## Reconnaissance

In this exercise, there is a social media app login screen in the browser sandbox. The goal is to try to get into Alice's account (username: alice). A SQL injection may be one way to do this. First, test if a SQL injection vulnerability exists by entering a special character in SQL for the password, such as a single quote and see what happens.

We are looking for a server error or something similar to test to see if there may be an issue with the site's security and the underlying software.

**Notice:** Don't worry if you see an internal server error.

Enter into the browser sandbox username: `alice` and the password:

```
alice'
```

(including the single quote). Keep in mind, if you have the Intercept Requests checkbox checked, you will have to click Submit Request in the Proxy (or just uncheck Intercept Requests for now).

What happened?

Proxy Status: LISTENING

Intercept Requests

Popout Mode ?

Reset Sandbox

Target Application

< > http://sandbox-hackedu.com/

Go

View Source

## Social Media App

### App Sign In

Username



Password



Sign in

© Social



PROXY



HISTORY



CODE



OUTPUT &  
ERRORS



PATCHES



# SQL Injection

Progress indicator: 12 circles, 11th is filled.

## Weaponization and Delivery

Now that you have the database name, why don't you try getting all of the passwords in the user table?

To do that, start iterating through the table names by using OFFSET and the following SELECT statement to find the user table:

```
SELECT table_name FROM information_schema.tables where table_schema
```

Once you have found the user table name, you need to find the column name since you need the SELECT statement to return a value and not a row. You can find the column names by iterating through the SELECT statement (substitute the table name you found with TABLENAME):

```
SELECT column_name FROM information_schema.COLUMNS WHERE table_sche
```

Once you have the table name and column name, you can then iterate through to find passwords.

[Back](#)[Next](#)

Proxy Status: LISTENING

Intercept Requests 

Popout Mode ?

Reset Sandbox

Target Application

<http://sandbox-hackedu.com/>

Go

View Source

## Social Media App

### App Sign In

© Social



PROXY



HISTORY



CODE



OUTPUT &amp; ERRORS



PATCHES



## Defense

How does one protect against SQL injection attacks?

Validate all input data ideally with a whitelist on the server side. In the example for searching for users the only input that should be accepted are usernames currently available. If it is not feasible, usernames should only be alphanumeric. Always enforce least privilege, giving the least privilege necessary to complete a task, and ensure that the mysql connection is at the minimum privilege to execute the query. These are general best practices.

For SQL injection specifically, use prepared statements (parameterized queries) for SQL queries. This separates both data and commands so that data won't be "executed". Stored procedures can also be used, but do not always protect against SQL injection. When used safely, stored procedures are similar to prepared statements in that there is separation of data and commands.

Examples in different languages are below.

Python:

```
c.execute("SELECT * FROM foo WHERE bar = %s AND baz = %s", (param1,
```

Java:

```
PreparedStatement stmt = connection.prepareStatement("SELECT * FROM  
stmt.setString(1, userid);  
stmt.setString(2, password);
```

Proxy Status: LISTENING Intercept

Target Application

< > http://sandbox-hackedu.com

Social Media

App Sign

© Social

PROXY

HISTORY 1

CODE EDITOR

CODE OUTPUT & ERRORS

PATCH HISTORY

Language: Java 8

```
1 package demo;  
2 import java.sql.*;  
3  
4 public class External{  
5     public static boolean login(String username, String password) throws Exception{  
6         try{  
7             Class.forName("com.mysql.jdbc.Driver");  
8             Connection con=DriverManager.getConnection("jdbc:mysql://mysql:3306/SocialMediaApp?useSSL=false","root","letme  
9  
10            Statement stmt=con.createStatement();  
11            ResultSet rs=stmt.executeQuery("SELECT * from tbl_user WHERE user_username ='" + username + "' AND user_passwo  
12  
13            if (rs.next()){  
14                con.close();  
15                return true;  
16            }  
17  
18            con.close();  
19            return false;  
20        } catch(Exception e){  
21            throw e;  
22        }  
23    }  
24  
25    public static boolean addPost(String post, String username) throws Exception{  
26        try{  
27            Class.forName("com.mysql.jdbc.Driver");  
28            Connection con=DriverManager.getConnection("jdbc:mysql://mysql:3306/SocialMediaApp?useSSL=false","root","letme  
29        } catch(Exception e){  
30            throw e;  
31        }  
32    }  
33 }
```

Patch Sandbox

### *i* Tips for getting output

For printing debug statements, do not use the browser console for output. This tab will act like your development terminal/console.

# What can we do to work together?

- Provide/review reports for completions and progress
- Dev Managers/CSRs can take the program and get certificate
- Encourage your program participants to schedule time each month to complete a module or two a month
- Discuss and promote in your meetings and relay any feedback or questions you receive in your teams or departments about the training
- Encourage use of the platform for practicing safe coding (go through the additional vulnerabilities after program completion)



# A Few Requests

- Review your initial department/team developer list and make corrections and additions
- Vote on the top three recognition gifts for those who complete the program
- Eat a brownie!

# Secure Development Training Program



Brian Anderson

Briank\_Anderson@byu.edu

801-422-0362

Be Wise, Be Alert.

This is in our power.

We can do this.

Secure Your code, Secure the Y.