User Instructions

Website Security Research

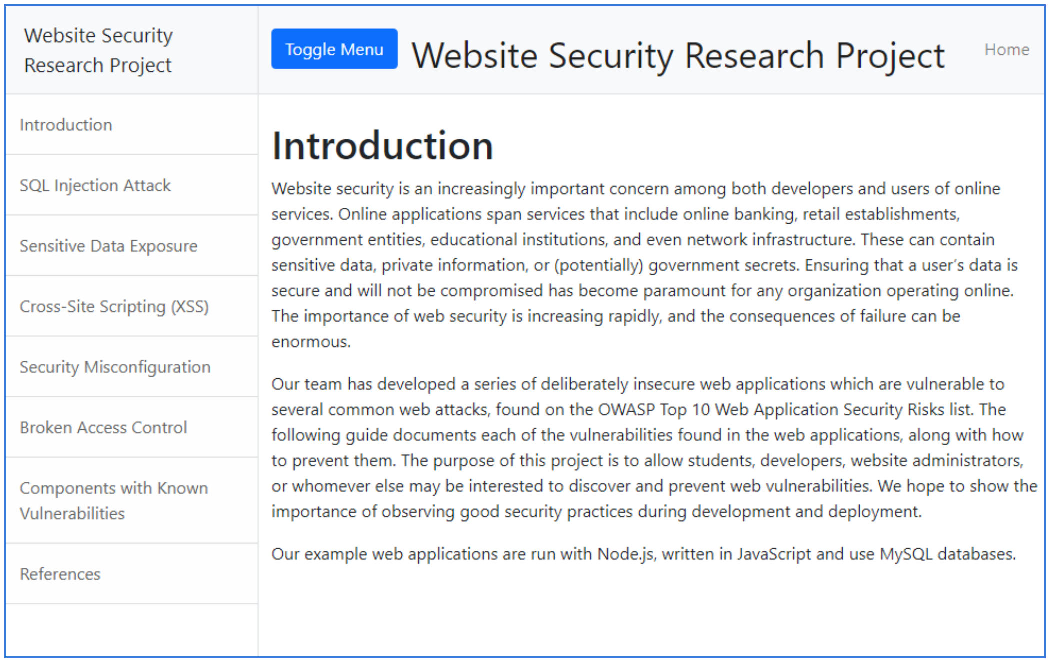
June 2nd 2021

Team Members

Patrick Dougan
Reed Hardin
Isfandyar Rabbani
Michael Volz

1. First, you will want to navigate to our main index website, which is located at: http://ec2-34-210-43-2.us-west-2.compute.amazonaws.com:3000/

2. You will see an introduction to our project, along with a side menu which links to each exploit we implemented:


Website Security Research Project
Introduction
SQL Injection Attack
Sensitive Data Exposure
Cross-Site Scripting (XSS)
Security Misconfiguration
Broken Access Control
Components with Known Vulnerabilities
References
Toggle Menu
Website Security Research Project
Home

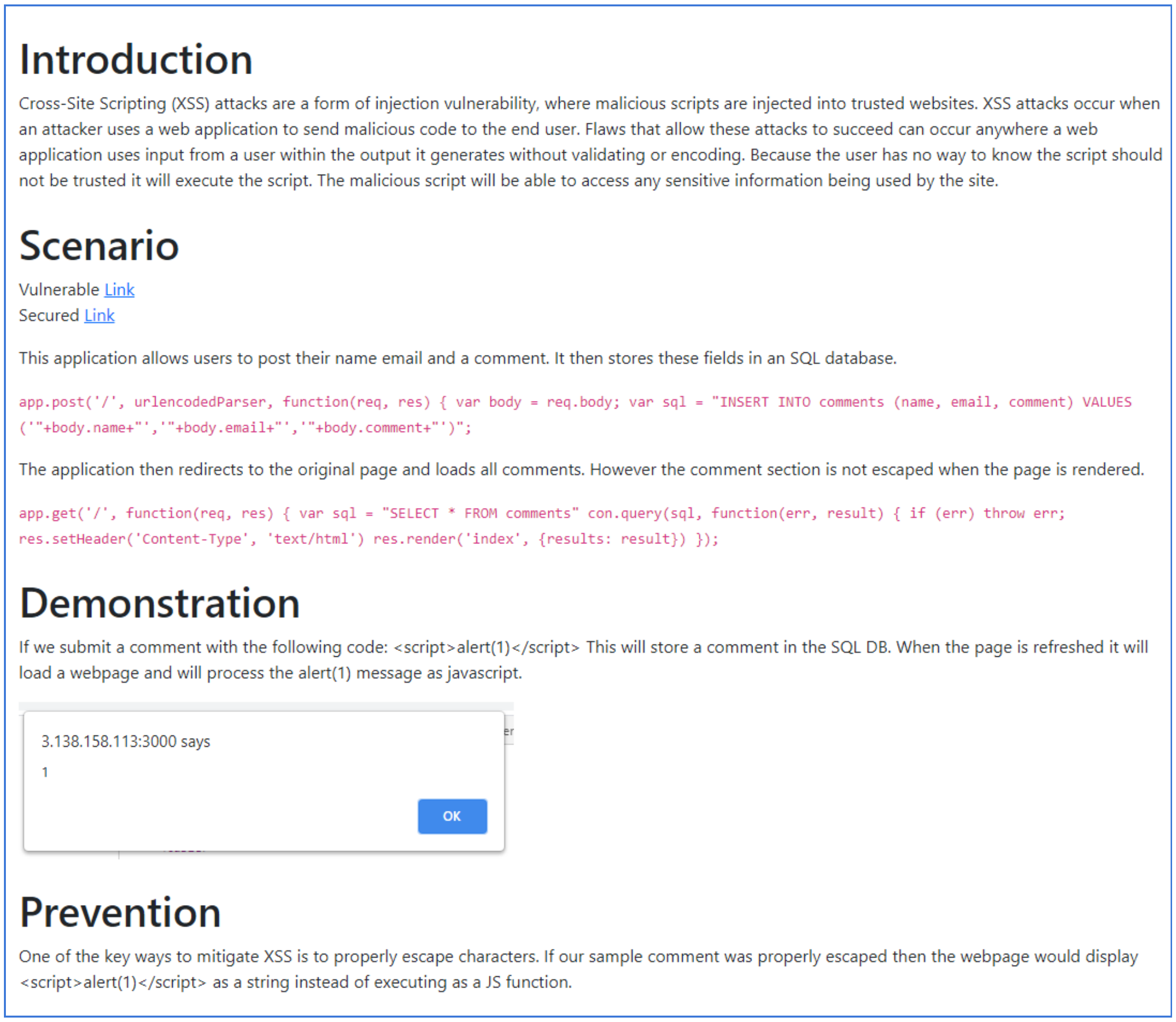
Introduction

Website security is an increasingly important concern among both developers and users of online services. Online applications span services that include online banking, retail establishments, government entities, educational institutions, and even network infrastructure. These can contain sensitive data, private information, or (potentially) government secrets. Ensuring that a user's data is secure and will not be compromised has become paramount for any organization operating online. The importance of web security is increasing rapidly, and the consequences of failure can be enormous.

Our team has developed a series of deliberately insecure web applications which are vulnerable to several common web attacks, found on the OWASP Top 10 Web Application Security Risks list. The following guide documents each of the vulnerabilities found in the web applications, along with how to prevent them. The purpose of this project is to allow students, developers, website administrators, or whomever else may be interested to discover and prevent web vulnerabilities. We hope to show the importance of observing good security practices during development and deployment.

Our example web applications are run with Node.js, written in JavaScript and use MySQL databases.

3. You can click through and explore each of the attacks as you please. Each attack has a demonstration section which links to an external, deliberately vulnerable web application to showcase the exploit first hand. Some attacks also have a fixed version of the web application where the exploit is no longer possible.



Introduction

Cross-Site Scripting (XSS) attacks are a form of injection vulnerability, where malicious scripts are injected into trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code to the end user. Flaws that allow these attacks to succeed can occur anywhere a web application uses input from a user within the output it generates without validating or encoding. Because the user has no way to know the script should not be trusted it will execute the script. The malicious script will be able to access any sensitive information being used by the site.

Scenario

Vulnerable Link
Secured Link

This application allows users to post their name email and a comment. It then stores these fields in an SQL database.

```
app.post('/', urlencodedParser, function(req, res) { var body = req.body; var sql = "INSERT INTO comments (name, email, comment) VALUES
('"+body.name+"','"+body.email+"','"+body.comment+"')";
```

The application then redirects to the original page and loads all comments. However the comment section is not escaped when the page is rendered.

```
app.get('/', function(req, res) { var sql = "SELECT * FROM comments" con.query(sql, function(err, result) { if (err) throw err;
res.setHeader('Content-Type', 'text/html') res.render('index', {results: result}) });
```

Demonstration

If we submit a comment with the following code: <script>alert(1)</script> This will store a comment in the SQL DB. When the page is refreshed it will load a webpage and will process the alert(1) message as javascript.

3.138.158.113:3000 says

1

Prevention

One of the key ways to mitigate XSS is to properly escape characters. If our sample comment was properly escaped then the webpage would display <script>alert(1)</script> as a string instead of executing as a JS function.

4. You may also navigate to our GitHub repo to check out the source code used to make all of the web applications and the index application at the following URL: https://github.com/PatrickDougan/Website-Security-Research-Project

5. Test, test, test! Send a request to the server using postman or interact using the UI for the feature in question, and really find out whether the weak and strong versions of the website are vulnerable to the attack.