Cross-Site Scripting (XSS) Workout Instructions

Introduction

Welcome to your team's *Cross-Site Scripting* workout, where you will learn how to perform multiple types of cross-site scripting (XSS) attacks on a web server.

Accessing the Workout

From your student landing page, click the *Enter Workout* button to automatically access UA Little Rock: Classified's cross-site scripting workout.

Your Mission

In this workout, you have three different types of XSS attacks to explore: DOM-based reflected and stored cross-site scripting. You may choose to work on any type first.

DOM-Based XSS

DOM stands for the document object model and is created when a web page is loaded. With the DOM model, JavaScript is able to create dynamic HTML. For example, this can be changing, creating, or removing HTML attributes, events, and elements. In a DOM-Based XSS attack, the source and execution of the attack are in the DOM and never leaves the browser. Since the URL of the web page is part of the DOM, the source of your attack will be in the URL. In this section, there is a bad request that doesn’t leave the browser. Perhaps you can use bad_request and JavaScript to create a DOM-Based XSS attack.

- Use JavaScript to update the title of the HTML page to “Classified DOMED”
- Use JavaScript alert to prompt the message, “Successful DOM-Based Attack”

Reflected XSS

In a reflected XSS attack, user input is immediately reflected back to the web application without it rendering safe by the browser. The web application can return the user input as part of a request. Either some or all of the requests can be JavaScript that executes an attack. In this section, there is a text field that can be exploited by inputting JavaScript into the field.

- Using JavaScript in the text field, change the CSS for ‘Hello, Stranger’ to your favorite color
- Using JavaScript in the text field, navigate to another site like https://www.google.com/

Stored XSS

Stored XSS occurs when the user input of a request that is not rendered safe by the browser is stored on the server, such as in a database. Everytime the user requests the stored input, they retrieve any injected script that was stored. This is persistent and will constantly occur until it is either removed from the database or rendered safe by the browser. In this section, there is a feedback section that is vulnerable to stored XSS attacks. Use JavaScript to inject code into the database. **Remember, this attack is persistent and will stay in the database until it is removed.**

- Add your own feedback to the database
- Use JavaScript alert in the input field to prompt a calculation for (23*5-4+3+57/27-42)
- Reload the page by navigating to another tab or refreshing

Extra scripts to explore

- `<script>window.location.replace("http://tiny.cc/safe-url");</script>`