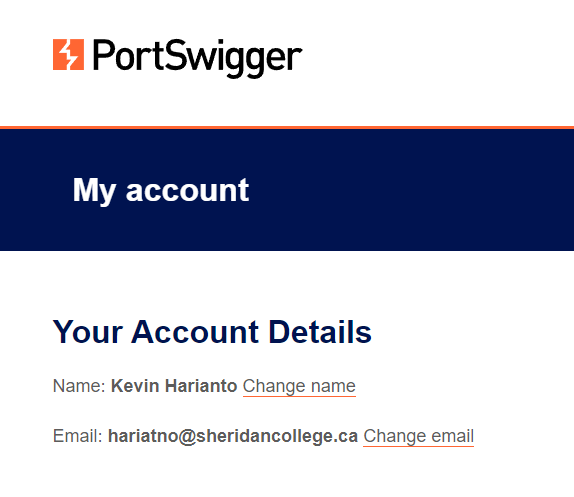
Assignment 4 Secure Software By Kevin Harianto

portswigger labs:



^Created an account

Lab 1: Graphical user interface, text, application, email

Description automatically generated

A picture containing application

Description automatically generated

Table

Description automatically generated with medium confidence

Entered Random

Alphanumerical Characters in search bar

Graphical user interface, text, application, Teams

Description automatically generated

^inspected the page

Graphical user interface, text, application, email

Description automatically generated

Noticed that it is also in the img src attribute

Searching for the characters entered.

^In the Inspection area Searched for the random characters entered.

NOTE: Now we will be breaking out of the img src with the help of double qoutes.

Noticed that the img src can be avoided with double quotes at the beginning right after the random characters are entered.

A picture containing application

Description automatically generated

Graphical user interface, application, website

Description automatically generated

^Solved first lab

Lab 2: Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Throwing an error to execute the alert

Icon

Description automatically generated

Graphical user interface, application

Description automatically generated

^Second Lab solved

Lab 3:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Going to the submit feedback page

Graphical user interface, text, application

Description automatically generated



^Changing the return path to random strings

Graphical user interface, application

Description automatically generated

^Inspecting source

Graphical user interface, text, application, email

Description automatically generated

Noticed that the website has made a reference on the random string.



^Changing the returnPath to showcase the cookie document.

Graphical user interface, text

Description automatically generated

Pressing the back button to execute the return path

A picture containing bar chart

Description automatically generated

^Lab 3 Is Solved.

Lab 4:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

^inside lab 4.

Going to the exploit server

Graphical user interface, text, application, email, website

Description automatically generated

^Inside the exploit server.

Graphical user interface, application

Description automatically generated

^Inserting the malicious iframe provided.

Graphical user interface

Description automatically generated

Storing the exploit

Viewing the exploit

Graphical user interface, text, application

Description automatically generated

^output of the stored exploit.

Text

Description automatically generated

^Solved Lab 4 by successfully delivering the exploit by the button previously shown.

Lab 5:

Graphical user interface, text, application

Description automatically generated

^In this case I am executing an SQL Injection lab as previously explained in the assignment description is that I can do any lab provided from the link. I am assuming that SQL injection labs are good.

A picture containing shape

Description automatically generated

^In the login account to bypass.

A picture containing bar chart

Description automatically generated

SQL Injection area

^Using SQL Injection to utilize single quote to bypass input validation

Graphical user interface, website

Description automatically generated

^Solving the lab 5.

List of all labs solved

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Labs that were shown to be solved.

1. Real-world example of a Cross-Site Scripting attack.

References: <https://www.wired.com/story/british-airways-hack-details/>, <https://brightsec.com/blog/xss-attack>.

Summary:

2018 British Airways attack by the hacker group Magecart, a hacker group known for their web-based credit card skimming attacks, meaning that most of their serious attacks are financially driven.

There was a data breach which made an impact on the activities of 380 000 booking transactions from customers between August 21 and September 5 in 2018. The hacker group managed to obtain names, addresses, email addresses as well as payment card details. Threat detection firm RiskIQ has provided insight into how it was accomplished. The hacker group basically implemented the Cross-site scripting attack in which the threat firm stated how the attackers identified a poorly secured web page component which allowed them to inject their own code and to alter the site’s behaviour. The poorly secured component was found to be a JavaScript component.

The JavaScript component was directly connected to the British Airways baggage claim information page. The malicious code was installed in order to obtain the data that the customers entered into the payment form -which allowed for the stealing of payment details- and set it to the attackers server when the user submitted the information. The attacks additionally paid for the Secure Socket Layer certificate to establish a sense of legitimacy.

Furthermore, the malicious JavaScript component injected by the attackers also impacted the mobile application as well, this meant that the attackers also accommodated touchscreen inputs as well. This makes the attack effective and simple as well because it was designed specifically to exploit the JavaScript weakness as well as the data flow issue of the British Airways Website.

Summary Diagram: NOTE: Explanations are labelled next page. I was not able to get more information on this due to the confidentiality of the attack, and that the RiskIQ consultation service keeps data on the in-depth context of the attacks caught a secret.

Web Server

Attackers Looking at the web site noticed that there was a JavaScript component that was vulnerable

2.

Victim information (User)

Attackers Site (where Rerouted Information is stored)

Legitimate (Victim) Site

1.

3.

4.

Information is entered

Through SSL Certification

1. The Legitimate website page JavaScript component has been modified in a way such that whenever the user has submitted the information, it gets sent to the attackers server for use instead of the database server.
2. The injected script is then stored permanently on the web server, completely altering how the website worked, for both mobile and pc users.
3. The user enters sensitive information into the Legitimate Web page such as their names, addresses, email addresses and payment card details.
4. The sensitive information is then sent to the attackers server encrypted with SSL to go under the radar and to act authentic.