CyberGuard Solutions

Penetration Testing Report

CLIENT-CONFIDENTIAL

Issue date: 04 December 2023

Confidentiality Statement

This report has been prepared for the sole and exclusive use of Harrow AS (henceforth referred to as Harrow). Therefore, it may not be made available to anyone other than authorized persons within the organization or relied upon by any third party. No part of this work may be reproduced or transmitted in any form by any means, electronic or mechanical, including photocopying and recording, or by information storage or retrieval system except as permitted, in writing by CyberGuard Solutions AS or Harrow.

Distribution List

Name & Role	Act	Review	Privy to Information
Jacob Johnson Chief Security Officer	✓	✓	
Anita Erasmus Senior Risk Manager		✓	
Joshua Vieira Head: IT			✓
Felisha Stokkeland Head: Internal Audit			✓

Version Control

Date	Version	Description	Approved by
30-Oct-2023	0.1	Document created	Lead penetration tester
30-Oct-2023	0.2	Added CVSS scores to the found vulnerabilities	Lead penetration tester
03-Nov-2023	0.3	Added documentation to Methodology	Lead penetration tester
04-Nov-2023	0.4	General review	Lead penetration tester
04-Nov-2023	1.0	Document published	Lead penetration tester

Table of Contents

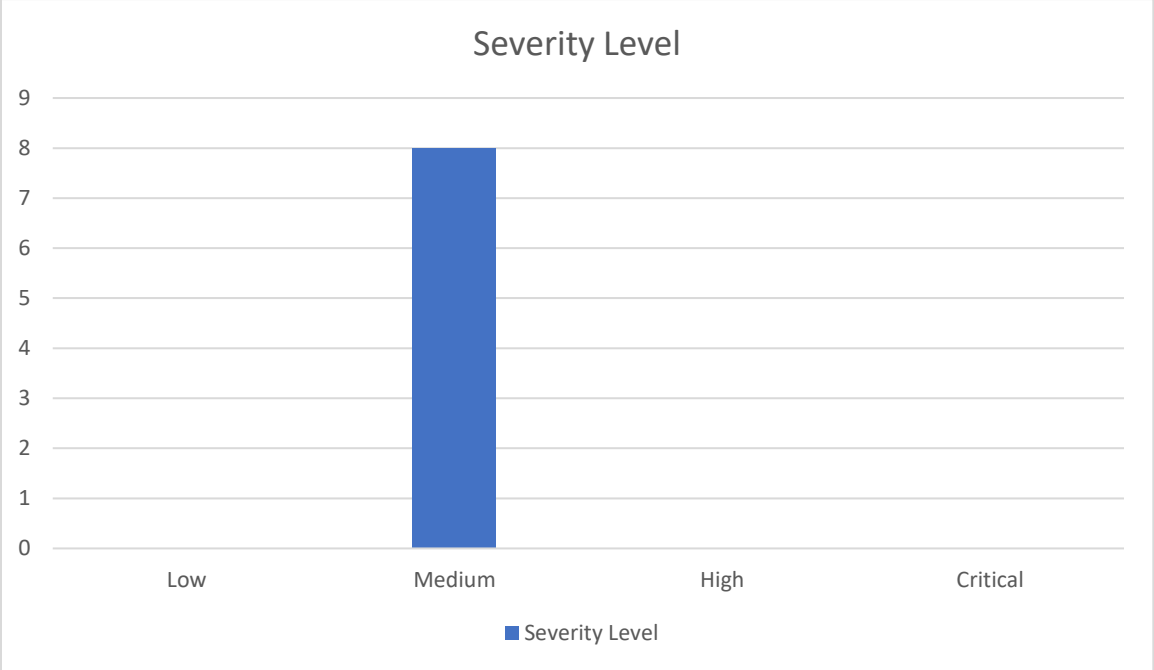
Executive Summary

This rapport details the findings from the internal penetration test performed on 30. November 2023 between hours 10:00 and 14:00 by CyberGuard Solutions AS at the request of the CSO at Harrow, Jacob Johnson, as outlined in the engagement letter provided. The letter confirmed the agreed terms for the test along with the authorization to act within confines of the designated scope.

The goal of the security test was to identify any potential vulnerabilities in the newly acquired web property, which as commented upon in the engagement letter was recognized as “likely a serious security risk to our organization”. As such, the test was to make an assessment of the web application and any utilized we application framework to broadly target the structural integrity of the web property. The test did not assess the host operating system, HTTP(S) configuration, external systems, social engineering or phishing attacks, as well as the following previously mapped vulnerabilities: clickjacking, cross-site request forgery, cookie attributes, logout functionality, brute force against the login page, ID enumeration, and open redirection.

The penetration test identified 8 vulnerabilities during the four hours of the internal investigation, which could cause pressing issues in the future if not properly addressed.
The vulnerabilities are presented in Figure 1 showing the potential impact each have on the business, totaling four separate severity levels. The level of severity of the found vulnerabilities in the web application is based on the Common Vulnerability Scoring System (CVSS), and the resulting scores are located in appendix 1, CVSS Results, at the end of this rapport.

Figure 1: Severity of the vulnerabilities in accordance with CVSS scores.



All of the vulnerabilities are deemed to be of medium severity, and is mainly due to each vulnerability having similar prerequisites regarding their exploitation. No high or critically rated vulnerabilities were found during the penetration test. However, Harrow should perform their own internal analysis regarding the presented vulnerabilities and come to their own conclusion determining the vulnerabilities’ severity and corresponding risk acceptance. This is especially true as the mapped vulnerabilities share severity levels, and it is emphasized that ensuring the future prevention and detection of cyber-attacks is dependent on the assessment of the

management. As this penetration test was to assist Harrow with the assessment of their newly acquired web property in preparation of a future rebranding, the impact and potential costs of the presented vulnerabilities must be considered in regards to Harrow's goals as a company, their risk tolerance, and future operational plans. It is the duty of the management level at Harrow to remain due diligent in the monitoring of the web application, and to keep their security measures up-to-date after this investigation has been concluded.

Many of the potential points of entry into the web application for an attacker are protected by effective input sanitation and validation, and were in adherence with best practices and standards for protecting against cyber-attacks.

Many thanks to the personnel and management of Harrow for their time and cooperation before and throughout the penetration testing engagement.

Methodology

This section aims to give a good understanding as to how the rapport's findings were discovered, tested, and documented.Figure 2 below shows a mind map explaining the merhodology behind the penetration test. It consists of a preparation phase where written authorization is presented, and the scope of the testis made clear. During the test, the reconnaissance, mapping, discovery and exploitation phases were utilized from a methodology standpoint.

Reconnaissance refers to gathering information about the target system or network using resources or tools for scanning the application's potential vulnerabilities. The objective in the mapping phase is to create an understanding of the target system and network, and to identify the relationships between them. The discovery phase refers to the identification of entry points for attackers, and the vulnerabilities that are potentially exploitable. The exploitation phase is a phase where potential vulnerabilities are actively being identified and exploited in order to show a proof-of-concept demonstration of the vulnerability. After the vulnerabilities have been found, or the deadline has been met, the results of the test is compiled and analyzed in the results phase.

In practice, the information usually regarding the reconnaissance and mapping phases had already been decided upon given the information in the engagement letter. The first step during the test was to identify all of the relevant input fields found on the web application in accordance with the stipulated scope presented in the engagement letter, meaning the discovery phase of Figure 2. These fields are used by attackers in order to bypass security requirements, or break mismanaged code which can yield sensitive information. There are many possible points of entry into web applications with lacking security implementations, and it is important to find as many of these ways into the underlying systems as possible for the mapping process to create a complete view of potential security flaws. Other than input fields, there were also discovered vulnerabilities in the URL of certain page requests that could be tampered with. There were also logic flaws regarding access control and the ownership of functions on the web application, such as discount codes working on days they were not supposed to be redeemable.

After having found flawed security measures within input fields or HTTP requests, it must be discerned whether or not they have implementation flaws which an attacker may be able to manipulate, leading to the exploitation phase in Figure 2. An essential tool for this is called Burp Suite, and allows for the interception of requests made to the server, which improves clarity regarding the information being submitted or changed, sent from the user. Studying the layout of each web page's source code is also vital, as it provides much information about how the user input is being validated and used to interact with the web page. Both of these means are utilized for each mapped entry point, and for some of the input fields, studying the layout alone would be enough to ascertain lacking input validation, leading to vulnerabilities on the web application. However, it is important to thoroughly check for input payloads that are able to bypass the security implementations of the web application.

After having deemed the mapped input fields as potential security vulnerabilities, they would have to be extensively tested by using specific payloads constructed to bypass user validated control mechanisms. Some of these mechanisms include:

- Encoding characters used in JavaScript syntax.
- Blacklisting certain keywords and commands which are used to create scripts.

Even though some input fields might have proper user validation, that is certainly not to be taken for granted, and as such every field must therefore be tested for relevant attack methods. Cross site scripting related vulnerabilities were discovered by using the following inputs:

- <script>test()</script>
- <img src=x onerror=test()>

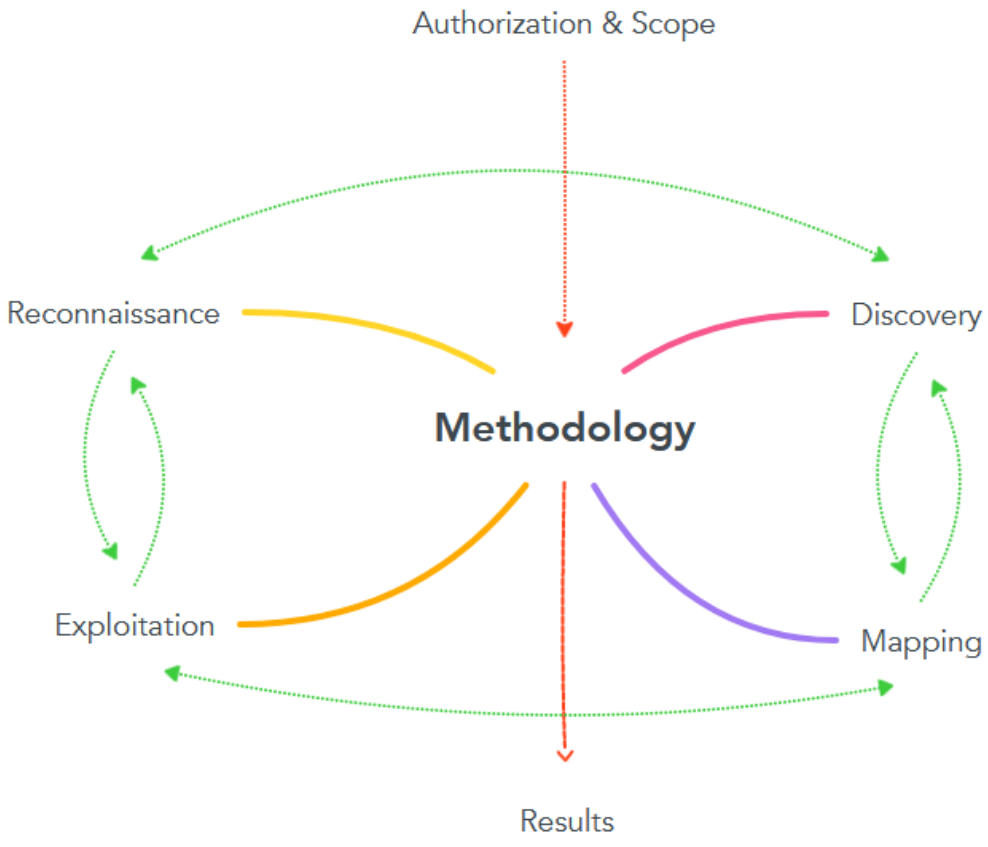
These create a popup alert window on the screen indicating that they have bypassed user input validation, and are not interfered with by the implemented security mechanisms as previously mentioned.

SQL injection related vulnerabilities were found using the following inputs:

- ‘
- ‘ or ‘1’=’1

These tests indicate whether or not the input field is vulnerable to returning information not owned by the user, and depending on the severity of lacking security implementations, have the possibility to show to the user far more entries from a database than intended. After having discovered vulnerabilities on the web application in the exploitation phase, these were then recorded and documented in the results phase of Figure 2, which would be used to create this rapport following up on the engagement.

Figure 2: Mind map of the penetration testing process



Detailed Findings

This section of the report will analyze one of the exposed vulnerabilities in greater detail as to provide a clear and concise example of one of the types of vulnerabilities found on the web application during the penetration test. The vulnerability being addressed in this section is referred to as “Stored XXS Vulnerability on the Guestbook page” in appendix 1 of this report.

Lacking Security Measures on the Guestbook Page Against Cross-site Scripting Attacks

Severity Rating

CVSS v4.0 Score: 5.3 / Medium

CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:L/VI:L/VA:N/SC:L/SI:L/SA:L

Description

XSS is an abbreviation for “Cross Site Scripting” and is an attack against web applications where a malicious actor aims to bypass the same origin policy which is implemented in order to limit the options users have when attempting to interact with web applications (PortSwigger, 2023). The policy states that scripts executed in one web page is allowed to fetch data from another web page, as long as those two web pages share a common origin. This in turn is to stop malicious code from one web page from impacting and gathering data from another web page.

In a stored XXS attack, the attacker uploads malicious code onto the website, into a database which is accessible to a specific page on the site. This injected code is then read and reiterated every time a user accesses the specific page. This often happens in comment section fields, as the information from the database is shown to the user whenever they read the comments on the page, leaving no need for user input. The goal of an XXS attack is to get the malicious code to return to the user, which in turn can allow the attacker to gain access to information regarding the session ID or cookie header when the victim sends a page request to a compromised web page (Germán E. Rodríguez, 2020). Doing so allows the attacker to potentially hijack the session if the cookies lack security attributes, allowing them to impersonate the victim or steal login credentials. The impact is greatly increased in stored XXS attacks, as the harmful code is located on the server side, affecting all visiting users. In a real attack, the issued code will not be loudly proclaimed as seen in Evidence 3, but silently executed unknowing to the affected victims.

During the penetration test, it was possible to upload a script onto the Guestbook page on the Harrow website, and this script would run every time the page was reloaded. The Guestbook page included a comments section with two fields for user input, one for the name and one for the comment itself, which indicated that there was a possibility for a potential stored XSS attack. Refer to

Evidence 1 for the page location and user input requirements. The script used in order to verify this vulnerability was <img src=x onerror=test(XSS)>>", as seen in Evidence 2. This JavaScript imbeds a small, incomplete image in the comment field. As it is incomplete, the "onerror" part allows an alert to be raised if the code is lacking in parameters. This is authenticated by an alert box popping up on the user's screen, indicating that the code is being automatically run without validating the user input. Refer to Evidence 3 for this observation.

This was not possible to properly embed into the "Comment:" field, as there were security implementations which alter the code attempted to be posted by an attacker. Such alterations include encoding the "<" and ">" signs, making them be read as "<" and ">" respectively. This hampers the attacker's ability to interact with the web application as they please, and can hinder them in successfully uploading malicious scripts. However, such encoding implementations were not present in the "Name:" field, and the example code was able to be executed according to the attacker's wishes.

Exploitation Method

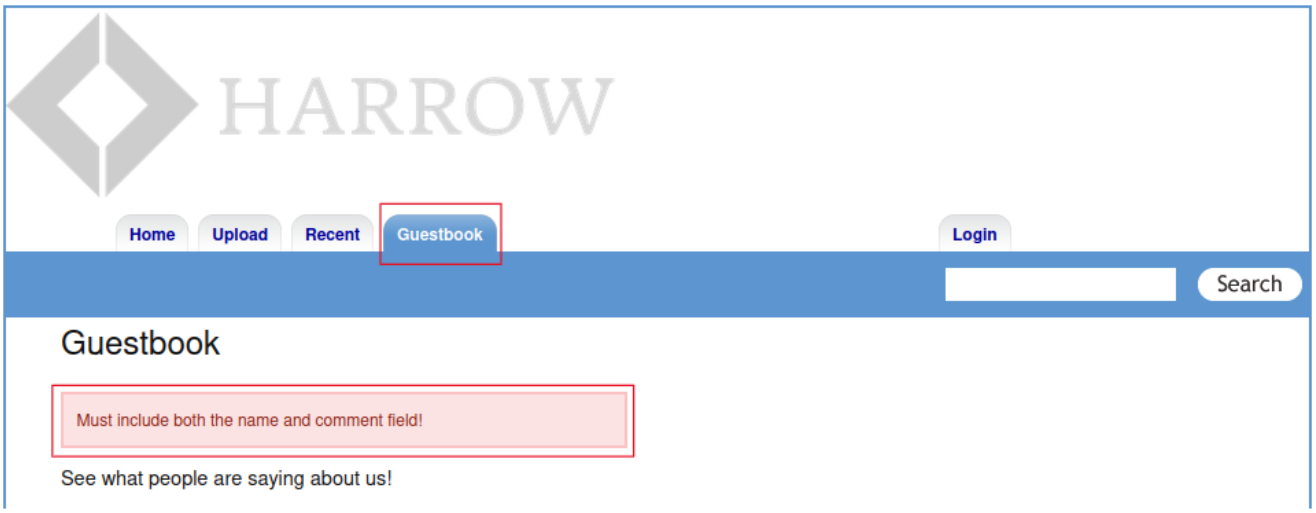
1. Go to the "Guestbook" page on the Harrow web application. There is no requirement to be logged in as a user, as unregistered guests also are able to visit the Guestbook page.
2. Locate the "Name:" input field.
3. Enter the following code into the input field (disregard the double quotes) – "<img src=x onerror=test()>".
4. As both the Name and Comments fields must have entries in order to leave a comment, simply type something arbitrary in the "Comments:" field, as the contents do not matter.
5. Observe an alert popup message appearing on the screen. This message is now imbedded in the database and loaded when the page is visited again in the future by any user.

Potential Impact

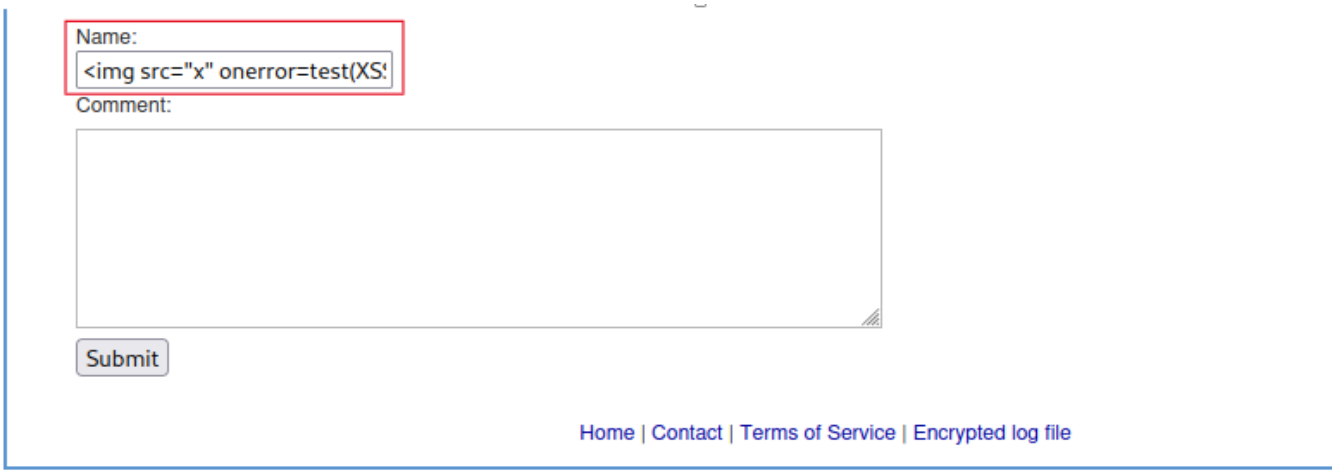
If the injected scripts capture and transmit sensitive user information, which can be easily done on periods of high user traffic on the web application, it could violate GDPR privacy regulations. This could result in legal consequences, regulatory fines, and damage to the organization's compliance status.

Evidence

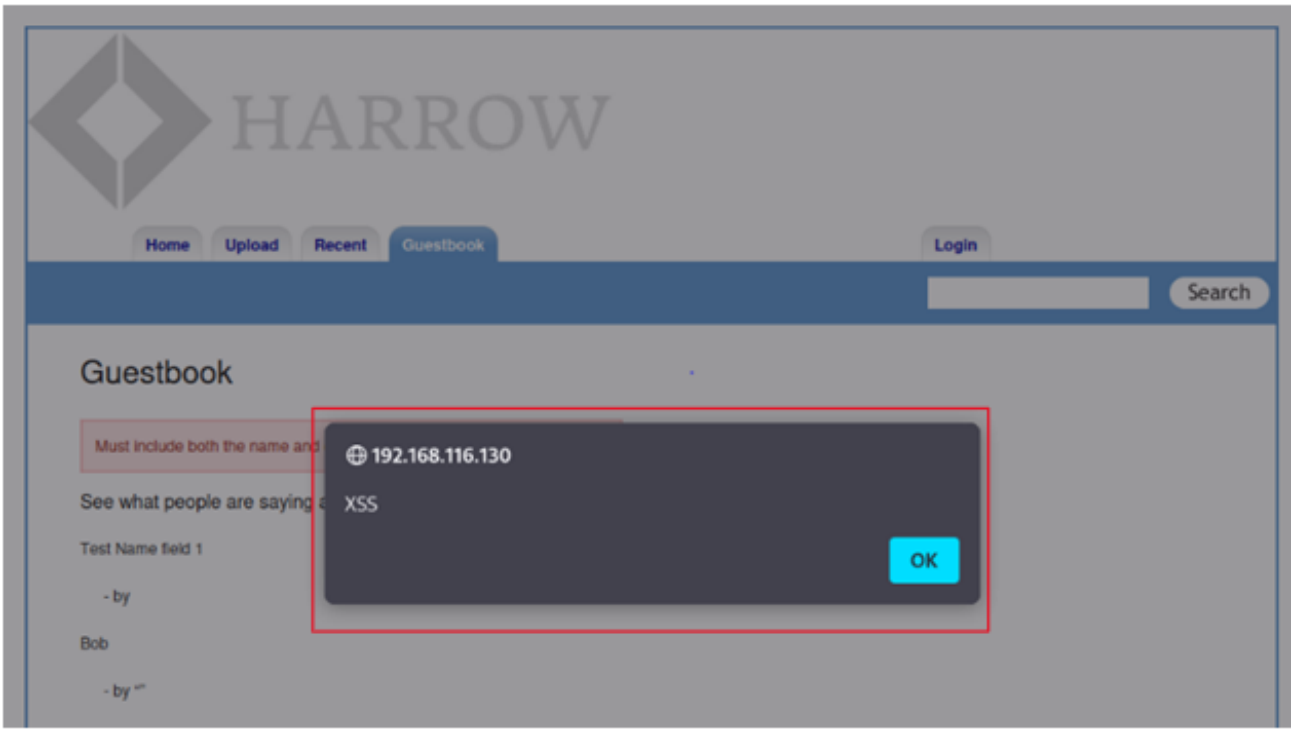
Evidence 1: shows the guestbook page location and input requirements needed for a successful XSS code manipulation to be possible.



Evidence 2: shows the input field entry, being "<img src="x" onerror=test(XSS)>" in the "Name:" field.



Evidence 3: shows an observable XSS vulnerability appearing on screen through an alert popup indicating that the script was successfully executed in the underlying code.



Remediation

A vital method of combatting XXS attacks is proper user validation. As was pointed out earlier, such validation methods had already been implemented in certain parts of the web application, hindering users in manipulating the web site's underlying code in unpredictable ways. All of the possible user input fields should therefore include proper input sanitation and validation, limiting the options attackers have when mapping the web application for entry points. Such input sanitation methods include user input encoding, and removal of keywords used in the creation of JavaScript code within the input field. The sanitation must be rigorous and should take into account the ways attackers could bypass such restrictions, as for example with the removal of instances of "<script>" in the input field, an attacker could enter "<scr<script>ipt>". If this is not taken into consideration, the <script> would remain valid after the innermost part in "<scr<script>ipt>" would be removed. Developers and stakeholders should be educated on the risks associated with XSS vulnerabilities, and the importance of proactive security measures like character encoding and blacklisting of words.

Appendices

1. CVSS Results

Vulnerability Name	CVSS Score & Vector
XSS vulnerability in the "What is your favorite color?" field.	CVSS v4.0 Score: 5.3 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N
SQL injection in the picture search box.	CVSS v4.0 Score: 6.9 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N
SQL injection on the login page.	CVSS v4.0 Score: 6.9 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N
Successful use of supposedly unavailable coupon code.	CVSS v4.0 Score: 6.9 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
HTML Get request manipulation from the "Check out a sample user!" link.	CVSS v4.0 Score: 6.9 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
XSS vulnerability in the "Name:" field on the Guestbook page.	CVSS v4.0 Score: 5.3 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:L/VI:L/VA:N/SC:L/SI:L/SA:L
XSS vulnerability in the "Comments" field of individual pictures.	CVSS v4.0 Score: 5.3 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:L/VI:L/VA:N/SC:L/SI:L/SA:L
Access to a high-resolution picture in the cart before purchasing it.	CVSS v4.0 Score: 6.9 / Medium CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

References

Germán E. Rodríguez, J. G. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. *Computer Networks*, 1-3. Retrieved from https://www.sciencedirect.com/science/article/pii/S1389128619311247

PortSwigger. (2023). *Cross-site scripting: PortSwigger*. Retrieved from portswigger.net: https://portswigger.net/web-security/cross-site-scripting