Sri Lanka Institute of Information Technology

Bug Bounty Report - 02

Module: IE2062

Web Security

Year 2, Semester 2

Aazaf Ritha. J – IT23151710

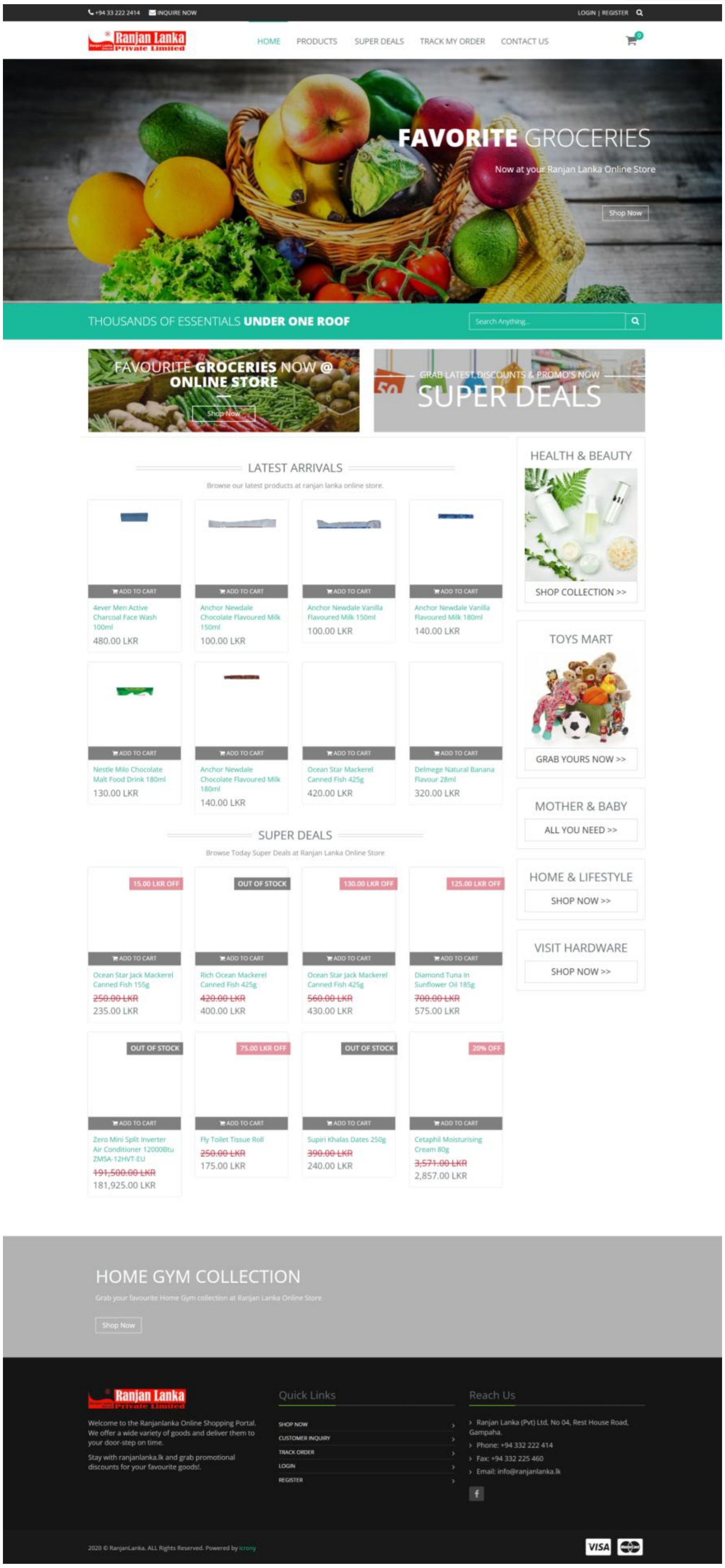
B.Sc. (Hons) in Information Technology

Specialized in Cyber Security

Table of Contents

Introduction

This report presents the findings of security vulnerabilities identified on the website www.ranjanlanka.lk as part of a Bug Bounty exercise for the Web Security assignment. During testing, two high-risk vulnerabilities were discovered: **Personally Identifiable Information (PII) Exposure** and **Cross-Site Scripting (XSS)**. The aim of this report is to describe these vulnerabilities in detail, assess their potential impact, demonstrate proof of concept, and propose effective mitigation strategies to enhance the website's security.



+94 33 222 2414 INQUIRE NOW

LOGIN | REGISTER

Ranjan Lanka Private Limited

HOME PRODUCTS SUPER DEALS TRACK MY ORDER CONTACT US

FAVORITE GROCERIES

Now at your Ranjan Lanka Online Store

Shop Now

THOUSANDS OF ESSENTIALS **UNDER ONE ROOF**

Search Anything...

FAVOURITE **GROCERIES** NOW @ **ONLINE STORE**

Shop Now

GRAB LATEST DISCOUNTS & PROMO'S NOW

SUPER DEALS

LATEST ARRIVALS

Browse our latest products at ranjan lanka online store.

ADD TO CART	4ever Men Active Charcoal Face Wash 100ml	480.00 LKR
ADD TO CART	Anchor Newdale Chocolate Flavoured Milk 150ml	100.00 LKR
ADD TO CART	Anchor Newdale Vanilla Flavoured Milk 150ml	100.00 LKR
ADD TO CART	Anchor Newdale Vanilla Flavoured Milk 180ml	140.00 LKR
ADD TO CART	Nestle Milo Chocolate Malt Food Drink 180ml	130.00 LKR
ADD TO CART	Anchor Newdale Chocolate Flavoured Milk 180ml	140.00 LKR
ADD TO CART	Ocean Star Mackerel Canned Fish 425g	420.00 LKR
ADD TO CART	Delmege Natural Banana Flavour 28ml	320.00 LKR

SUPER DEALS

Browse Today Super Deals at Ranjan Lanka Online Store

15.00 LKR OFF	ADD TO CART	Ocean Star Jack Mackerel Canned Fish 155g	~~250.00 LKR~~	235.00 LKR
OUT OF STOCK	ADD TO CART	Rich Ocean Mackerel Canned Fish 425g	~~420.00 LKR~~	400.00 LKR
130.00 LKR OFF	ADD TO CART	Ocean Star Jack Mackerel Canned Fish 425g	~~560.00 LKR~~	430.00 LKR
125.00 LKR OFF	ADD TO CART	Diamond Tuna In Sunflower Oil 185g	~~700.00 LKR~~	575.00 LKR
OUT OF STOCK	ADD TO CART	Zero Mini Split Inverter Air Conditioner 12000Btu ZMSA-12HVT-EU	~~191,500.00 LKR~~	181,925.00 LKR
75.00 LKR OFF	ADD TO CART	Fly Toilet Tissue Roll	~~250.00 LKR~~	175.00 LKR
OUT OF STOCK	ADD TO CART	Supiri Khalas Dates 250g	~~390.00 LKR~~	240.00 LKR
20% OFF	ADD TO CART	Cetaphil Moisturising Cream 80g	~~3,571.00 LKR~~	2,857.00 LKR

HEALTH & BEAUTY

SHOP COLLECTION >>

TOYS MART

GRAB YOURS NOW >>

MOTHER & BABY

ALL YOU NEED >>

HOME & LIFESTYLE

SHOP NOW >>

VISIT HARDWARE

SHOP NOW >>

HOME GYM COLLECTION

Grab your favourite Home Gym collection at Ranjan Lanka Online Store

Shop Now

Ranjan Lanka Private Limited

Welcome to the Ranjanlanka Online Shopping Portal. We offer a wide variety of goods and deliver them to your door-step on time.

Stay with ranjanlanka.lk and grab promotional discounts for your favourite goods!.

Quick Links

- SHOP NOW
- CUSTOMER INQUIRY
- TRACK ORDER
- LOGIN
- REGISTER

Reach Us

- Ranjan Lanka (Pvt) Ltd, No 04, Rest House Road, Gampaha.
- Phone: +94 332 222 414
- Fax: +94 332 225 460
- Email: info@ranjanlanka.lk

2020 © RanjanLanka. ALL Rights Reserved. Powered by Icrony

VISA

Vulnerability - 01

Title – PII Exposure

Risk Level- High

Domain – www.ranjanlanka.lk

Description

A **PII Exposure Vulnerability** occurs when a web application, system, or service inadvertently exposes sensitive **Personally Identifiable Information (PII)** of its users or employees, often due to inadequate security measures. PII can include data such as names, addresses, phone numbers, email addresses, social security numbers, bank account details, medical records, or any other information that can uniquely identify an individual. Exposure of PII can occur in various ways, including unencrypted storage, insufficient access controls, insecure data transmissions, logging sensitive information, or mishandling of data. When PII is exposed, it can lead to identity theft, financial fraud, social engineering attacks, and other malicious activities that harm users and compromise the reputation of the organization. This type of vulnerability is directly related to several categories in the **OWASP Top 10**, including **A01:2021 – Broken Access Control**, **A02:2021 – Cryptographic Failures**, and **A09:2021 – Security Logging and Monitoring Failures**. These categories emphasize how improper handling of authentication, encryption, and monitoring mechanisms can result in the unintentional disclosure of sensitive personal information.

Affected Component

Endpoint Affected – Product category (Purchased Items)

Vulnerable Data –	Credit Card Type
	Bank Identification Number
	Brand
	Category
	Issuer

Web Application or Server - www.ranjanlanka.lk

Impact Assessment

Privacy Violations: Exposure of personal information leads to a direct violation of user privacy, causing loss of trust in the application or service.

Identity Theft: Attackers can use exposed PII to impersonate victims and commit fraudulent activities like opening accounts, applying for loans, or accessing sensitive systems.

Financial Loss: If financial data like bank account details or credit card numbers are exposed, users may suffer financial loss due to fraudulent transactions.

Reputation Damage: PII exposure can severely damage an organization's reputation, particularly if it is subject to a public breach. This can lead to a loss of customers, legal actions, and damage to brand integrity.

Legal and Regulatory Consequences: Organizations are legally required (e.g., under GDPR, CCPA, HIPAA) to protect PII. A breach could result in heavy fines, penalties, and lawsuits.

Social Engineering Attacks: Exposed PII can enable attackers to perform targeted phishing or social engineering attacks against users.

Steps to Reproduce

Step 01: Identify PII in the Application:

Review the application to identify areas where PII might be collected, such as login forms, registration forms, profile settings, payment processing pages, and databases.

PII typically includes personal details like name, email, phone number, address, social security number, and bank account information.

Step 02: Check for Unencrypted Storage of PII:

Inspect how PII is stored in the backend systems (e.g., databases, files).

If PII is stored in plain text or weakly encrypted, this could be an exposure risk.

Use database query tools to manually retrieve records with PII or inspect API responses.

Step 03: Review Data Transmission:

Analyze how data containing PII is transmitted. Ensure that HTTPS (SSL/TLS encryption) is used to encrypt sensitive information during transmission.

You can use Burp Suite or Wireshark to monitor traffic between the client and server to check if sensitive data is being transmitted over an unencrypted HTTP connection.

Step 04: Check Access Controls and Authentication Mechanisms:

Review the access control mechanisms to ensure that only authorized users can access PII.

Check if there are insufficient authentication checks that allow unauthorized access to PII. For example, **tests** for weak passwords, session hijacking vulnerabilities, or missing multi-factor authentication (MFA).

Step 05: Examine Logs for PII Exposure:

Review application logs for instances where sensitive information (e.g., full names, emails, passwords, etc.) is written to logs.

Ensure that PII is not inadvertently logged by inspecting log files or using security scanning tools to identify logging issues.

Step 06: Test for Insecure Data Access (SQL Injection or Direct Database Access):

Use tools like Burp Suite to test if PII can be extracted via injection attacks, like SQL injection. If the application fails to properly sanitize user input, attackers could potentially retrieve sensitive data directly from the database.

Step 07: Check Backup and Recovery Systems:

Review how backups containing PII are handled. If backups are unencrypted or stored in insecure locations, they might be exposed during data recovery processes.

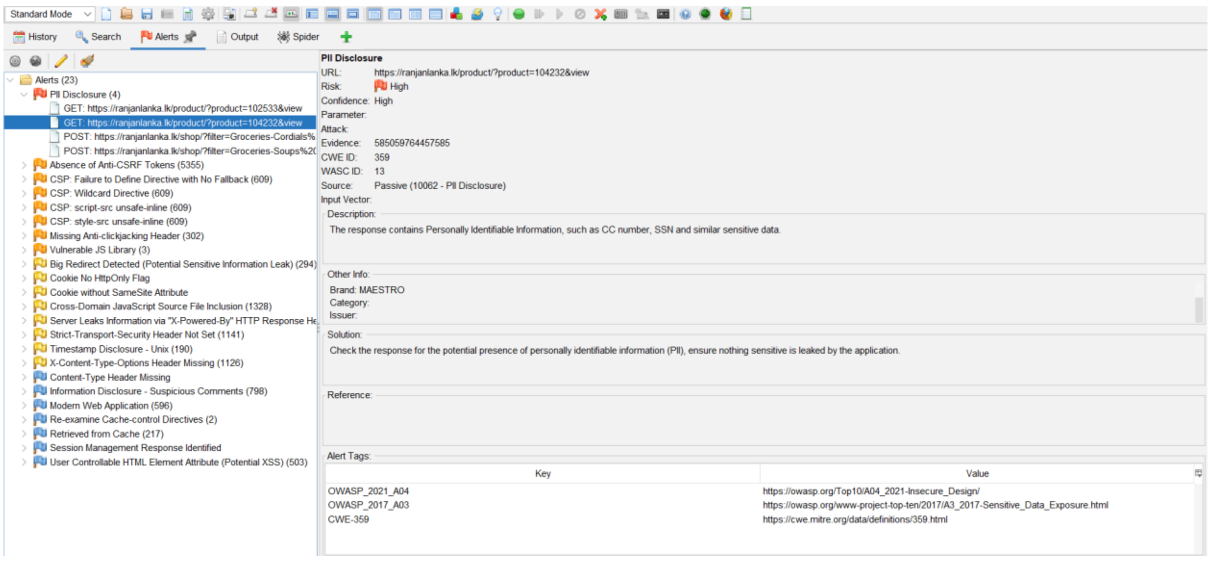
Step 08: Verify Public Access:

Test for publicly accessible URLs or API endpoints that might expose PII, such as through a search or profile page.

Use a tool like Burp Suite or OWASP ZAP to perform vulnerability scans and identify insecure endpoints that could expose PII.

Proof of Concept (Screenshots or video link)

OWASP ZAP Scan



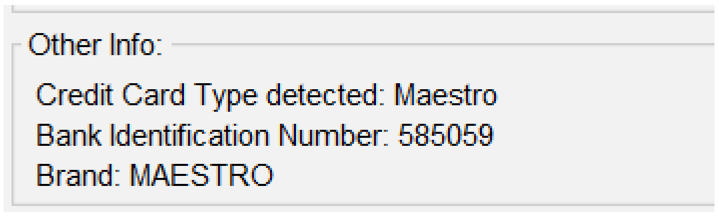
```
Alerts (23)
  PII Disclosure (4)
    GET: https://ranjanlanka.lk/product/?product=102533&view
    GET: https://ranjanlanka.lk/product/?product=104232&view
    POST: https://ranjanlanka.lk/shop/?filter=Groceries-Cordials%
    POST: https://ranjanlanka.lk/shop/?filter=Groceries-Soups%20
  Absence of Anti-CSRF Tokens (5355)
  CSP: Failure to Define Directive with No Fallback (609)
  CSP: Wildcard Directive (609)
  CSP: script-src unsafe-inline (609)
  CSP: style-src unsafe-inline (609)
  Missing Anti-clickjacking Header (302)
  Vulnerable JS Library (3)
  Big Redirect Detected (Potential Sensitive Information Leak) (294)
  Cookie No HttpOnly Flag
  Cookie without SameSite Attribute
  Cross-Domain JavaScript Source File Inclusion (1328)
  Server Leaks Information via "X-Powered-By" HTTP Response He
  Strict-Transport-Security Header Not Set (1141)
  Timestamp Disclosure - Unix (190)
  X-Content-Type-Options Header Missing (1126)
  Content-Type Header Missing
  Information Disclosure - Suspicious Comments (798)
  Modern Web Application (596)
  Re-examine Cache-control Directives (2)
  Retrieved from Cache (217)
  Session Management Response Identified
  User Controllable HTML Element Attribute (Potential XSS) (503)
```

```
PII Disclosure
URL:         https://ranjanlanka.lk/product/?product=104232&view
Risk:        High
Confidence:  High
Parameter:
Attack:
Evidence:    585059764457585
CWE ID:      359
WASC ID:     13
Source:      Passive (10062 - PII Disclosure)
Input Vector:
Description:
  The response contains Personally Identifiable Information, such as CC number, SSN and similar sensitive data.
Other Info:
  Brand: MAESTRO
  Category:
  Issuer:
Solution:
  Check the response for the potential presence of personally identifiable information (PII), ensure nothing sensitive is leaked by the application.
Reference:
Alert Tags:
```

Key	Value
OWASP_2021_A04	https://owasp.org/Top10/A04_2021-Insecure_Design/
OWASP_2017_A03	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html
CWE-359	https://cwe.mitre.org/data/definitions/359.html

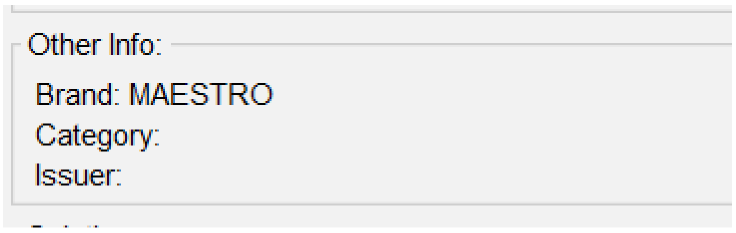
Credit Card Type, Brand, Category, Issuer, Bank Identification Number are exposed in the URL as query parameters, which can be intercepted by anyone with access to network traffic.

Example of Exposed Data



```
Other Info:
  Credit Card Type detected: Maestro
  Bank Identification Number: 585059
  Brand: MAESTRO
```

Figure 1 OWASP ZAP Scan Results



```
Other Info:
  Brand: MAESTRO
  Category:
  Issuer:
```

Figure 2 OWASP ZAP Scan Results 2

Proposed Mitigation or Fix

To address the exposure of PII, the following mitigation strategies should be implemented

Encrypt PII: Use strong encryption (e.g., AES-256) for storing PII both at rest and in transit. Ensure that sensitive data such as passwords, social security numbers, and credit card details are never stored in plaintext.

- Encrypt sensitive data using modern cryptographic standards and store keys in a secure key management system.

Use HTTPS for Data Transmission: Ensure that all data containing PII is transmitted over HTTPS, which encrypts the data in transit. Use SSL/TLS certificates and configure your web server to enforce HTTPS.

- Consider implementing HTTP Strict Transport Security (HSTS) to enforce HTTPS connections.

Implement Proper Access Controls: Restrict access to PII based on user roles and permissions. Ensure that only authorized users (e.g., admins or the data owner) can access sensitive information.

- Use least privilege principles to limit access to sensitive data, and ensure that the application performs proper role-based access control (RBAC).

Mask or Redact PII in Logs: Ensure that logs do not contain any sensitive PII. If logging is necessary, redact or mask sensitive information such as credit card numbers or personal addresses.

- Use a logging solution that filters or obfuscates PII to prevent accidental exposure.

Use Secure Backup Practices: Encrypt all backup files containing PII and store them in secure, access-controlled locations. Implement regular checks to ensure backup files are protected.

Regulatory Compliance: Ensure compliance with privacy regulations such as GDPR, HIPAA, or CCPA, which mandate strong security and privacy practices for handling PII.

- Conduct regular audits to ensure your PII protection policies are effective and up to date.

Vulnerability - 02

Title – Reflected Cross-Site Scripting (XSS)

Risk Level – High

Domain – www.ranjanlanka.lk

Description

Reflected Cross-Site Scripting (XSS) occurs when user-supplied input is included in the output page without proper sanitization or encoding, allowing attackers to inject and execute malicious scripts in the context of the victim's browser. These scripts can lead to serious consequences such as **session hijacking**, **website defacement**, **theft of sensitive data**, or **redirection to malicious sites**. This vulnerability was identified in a search input field where the input provided by the user is immediately reflected in the HTML response without appropriate output encoding or input validation. As a result, an attacker could craft a malicious URL containing a script, which when visited by a victim, would execute in their browser and compromise their session or personal data. Reflected XSS is categorized under **A03:2021 – Injection** in the **OWASP Top 10**, emphasizing the risk of untrusted data being interpreted as code by a web application.

Affected Component

Endpoint Affected – Search functionality (/search?q= parameter)

Vulnerable Parameter – q (query string)

Vulnerable Component – Input reflection in HTML output without sanitization

Web Application or Server – www.ranjanlanka.lk

Impact Assessment

Session Hijacking: Attackers can steal session cookies through malicious scripts, allowing them to impersonate users and access their accounts without authentication.

Credential Theft: Injected scripts can mimic login forms or intercept keystrokes to capture usernames and passwords.

Phishing Attacks: The attacker can redirect users to fake pages that appear to be part of the legitimate website, tricking them into revealing sensitive information.

Reputation Damage: If users encounter malicious behavior on the site (e.g., fake alerts, redirects), it may reduce trust in the platform and harm the organization's brand image.

Browser Exploitation: In certain cases, XSS can be used to exploit browser vulnerabilities, potentially allowing malware delivery or deeper system compromise.

Privilege Escalation: If executed in an administrative context (such as through blind XSS), the attacker may gain access to restricted functionalities.

Legal and Compliance Risks: Exploitation of XSS could lead to unauthorized access to personal data, triggering violations of privacy laws like GDPR or CCPA.

Steps to Reproduce

Step 01: Identify Input Reflection Points

Navigate through the application and look for user input fields or query parameters that reflect user-supplied data back into the webpage. In this case, the search functionality was selected for testing.

Step 02: Insert Malicious Payload into the URL

Modify the search URL to include a script payload. Example:

php-template

CopyEdit

https://www.ranjanlanka.lk/search?q=<script>alert(0)</script>

Step 03: Observe the Behavior of the Webpage

After visiting the modified URL, observe the browser behavior. The script is executed in the browser, confirming that the input is not properly sanitized or encoded.

Step 04: Confirm Script Execution

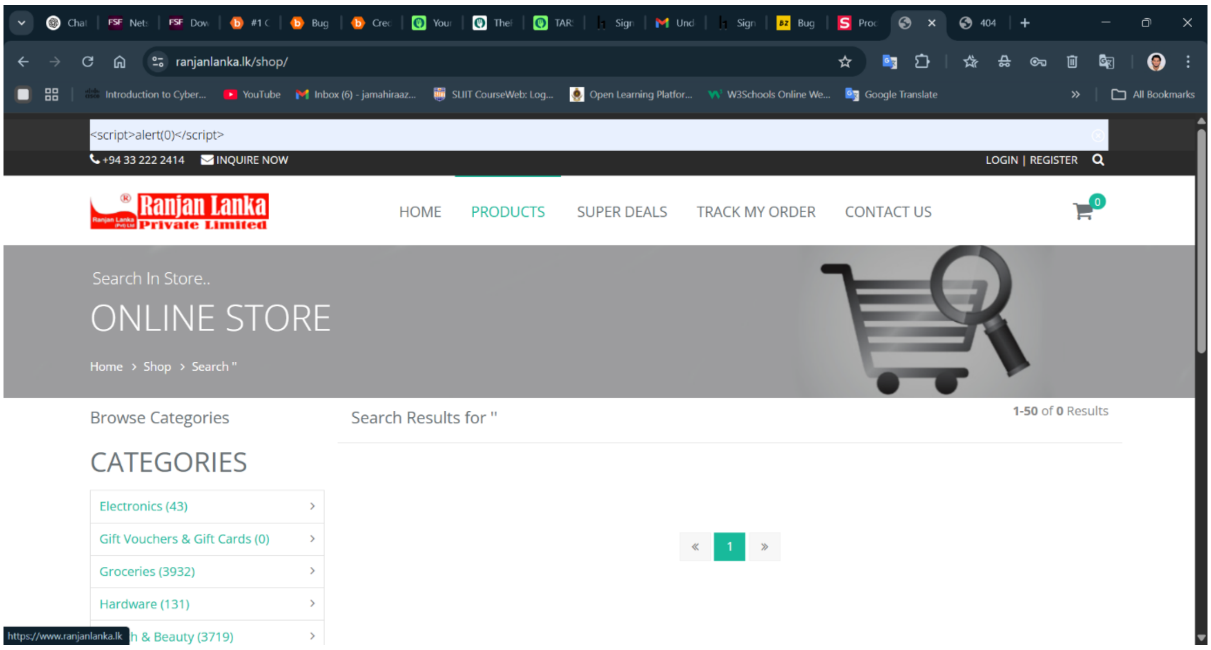
The browser displays a JavaScript alert box, indicating that the script was executed successfully in the context of the application. This confirms the vulnerability.

Step 05: Test with Security Tools

Use tools like OWASP ZAP, Burp Suite, and XSStrike to automate discovery and fuzz for more payloads. Observe reflected parameters and automated detection of XSS vectors.

Proof of Concept (Screenshots or video link)

The malicious script (<script>alert(0)</script>) is injected via the browser's address bar into the search field.



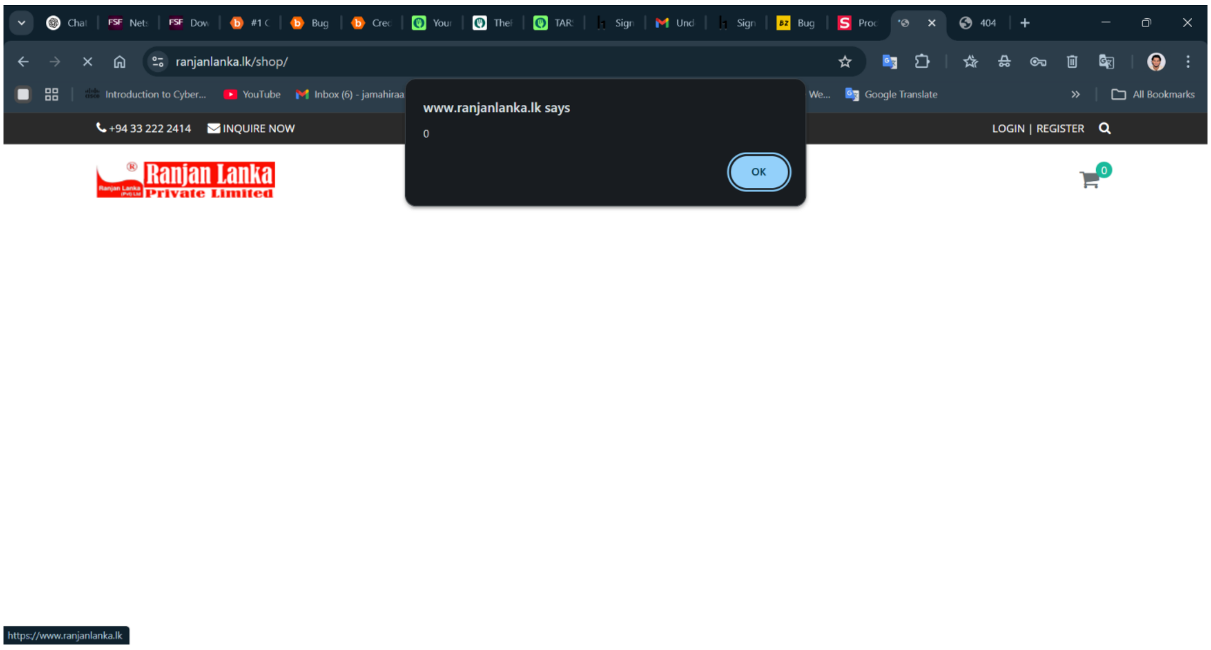
ranjanlanka.lk/shop/

<script>alert(0)</script>

Search Results for "

1-50 of 0 Results

The browser executes the payload and displays a JavaScript alert box (alert(0)), confirming that the input did not properly escape or encoded.



ranjanlanka.lk/shop/

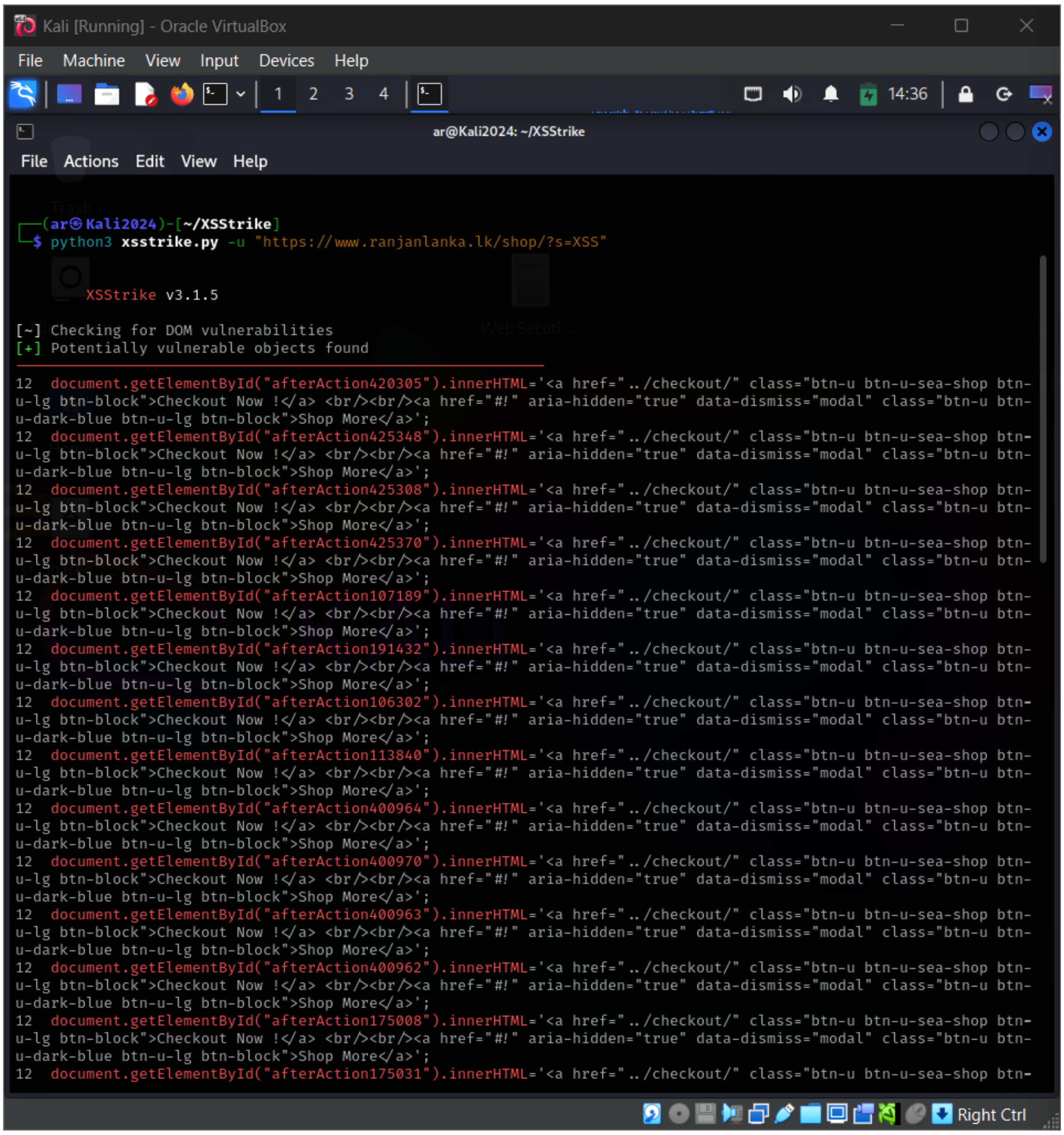
www.ranjanlanka.lk says

0

XSStrike Scan Results

The XSStrike tool was executed against the vulnerable endpoint using the following command:

python3 xsstrike.py -u "https://www.ranjanlanka.lk/shop/?s=XSS"


Kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
ar@Kali2024: ~/XSStrike
File Actions Edit View Help
Right Ctrl

```
┌──(ar㉿Kali2024)-[~/XSStrike]
└─$ python3 xsstrike.py -u "https://www.ranjanlanka.lk/shop/?s=XSS"


        XSStrike v3.1.5

[~] Checking for DOM vulnerabilities
[+] Potentially vulnerable objects found
------------------------------------------------------------
12  document.getElementById("afterAction420305").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction425348").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction425308").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction425370").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction107189").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction191432").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction106302").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction113840").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction400964").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction400970").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction400963").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction400962").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction175008").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-u-lg btn-block">Checkout Now !</a> <br/><br/><a href="#!" aria-hidden="true" data-dismiss="modal" class="btn-u btn-u-dark-blue btn-u-lg btn-block">Shop More</a>';
12  document.getElementById("afterAction175031").innerHTML='<a href="../checkout/" class="btn-u btn-u-sea-shop btn-
```

Proposed Mitigation or Fix

Output Encoding: Ensure all user-supplied input is properly encoded before being rendered in the HTML output. Use context-aware encoding:

- HTML entity encoding for data placed in HTML body.
- JavaScript escaping for data placed within scripts.
- URL encoding for data used in URLs.
- Attribute encoding for data inserted into HTML attributes.

Input Validation: Validate input on both the client and server sides. Implement allow-lists (positive validation) to accept only expected input formats (e.g., alphanumeric for search fields).

Use Security Libraries: Utilize libraries and frameworks that automatically handle encoding (e.g., React, Angular, or templating engines like Handlebars) to reduce the risk of XSS by design.

HTTP Headers: Implement security headers such as:

- Content-Security-Policy (CSP) to restrict sources of scripts.
- X-XSS-Protection (legacy support).
- X-Content-Type-Options: nosniff.

Avoid Unsafe JavaScript Functions: Do not use functions like eval(), innerHTML, or document.write() with untrusted data.

Conclusion

During the security assessment of www.ranjanlanka.lk, two high-risk vulnerabilities were identified: **Personally Identifiable Information (PII) Exposure** and **Reflected Cross-Site Scripting (XSS)**. These issues pose serious threats to user privacy, system security, and the organization's reputation.

The PII exposure can lead to identity theft, financial fraud, and violations of data protection laws. The XSS vulnerability allows attackers to execute malicious scripts, hijack sessions, and compromise user trust.

To mitigate these risks, immediate actions such as data encryption, secure input/output handling, and robust access control mechanisms are necessary. Regular security testing and adherence to secure coding practices are strongly recommended to maintain a secure web environment and ensure regulatory compliance.