JPET STOREVAPT REPORT

short line

SANJAY S KUMAR

# 

**INDEX**

**I : Summery**

**II : Tools used**

**III : vulnerabilities lists.**

**Summery**

This document presents a comprehensive security assessment of the JPetStore Demo website. The primary purpose of this assessment is to thoroughly identify and analyze the existing vulnerabilities within the website and evaluate the associated security risks. Through this detailed evaluation, the document aims to provide a clear understanding of the current security posture of the JPetStore Demo website.

The assessment process involved a meticulous examination of the website to uncover various vulnerabilities. Each identified vulnerability is documented in detail within this report, highlighting the potential threats they pose.The document outlines specific mitigation strategies to address and remediate these vulnerabilities effectively. By implementing these mitigation measures, the security of the website can be significantly enhanced, reducing the risk of exploitation by malicious actors.

**Scope :** <https://petstore.octoperf.com/actions/Catalog.action>

**Tools used**

The tool used for almost every assessment is Burp suite community edition and chrome browser.Burp Suite is a popular and powerful tool used for web application security testing. It is widely utilized by security professionals, penetration testers, and ethical hackers to identify and exploit vulnerabilities in web applications.

**Vulnerabilities List**

| **SI NO** | **Vulnerability** | **Severity** |
| --- | --- | --- |
| **1** | **Weak password policy** | **8.2** |
| **2** | **Http parameter tampering** | **9.2** |
| **3** | **Inventory Management Issue** | **8.3** |
| **4** | **Design vulnerability** | **3.5** |
| **5** | **sensitive data exposure** | **8.9** |
| **6** | **Xss** | **8.2** |
| **7** | **improper validation** | **7.1** |
| **8** | **Session management error** | **6.7** |
| **9** | **Lack of rate limiting** | **8.9** |

# **1.Weak password policy**

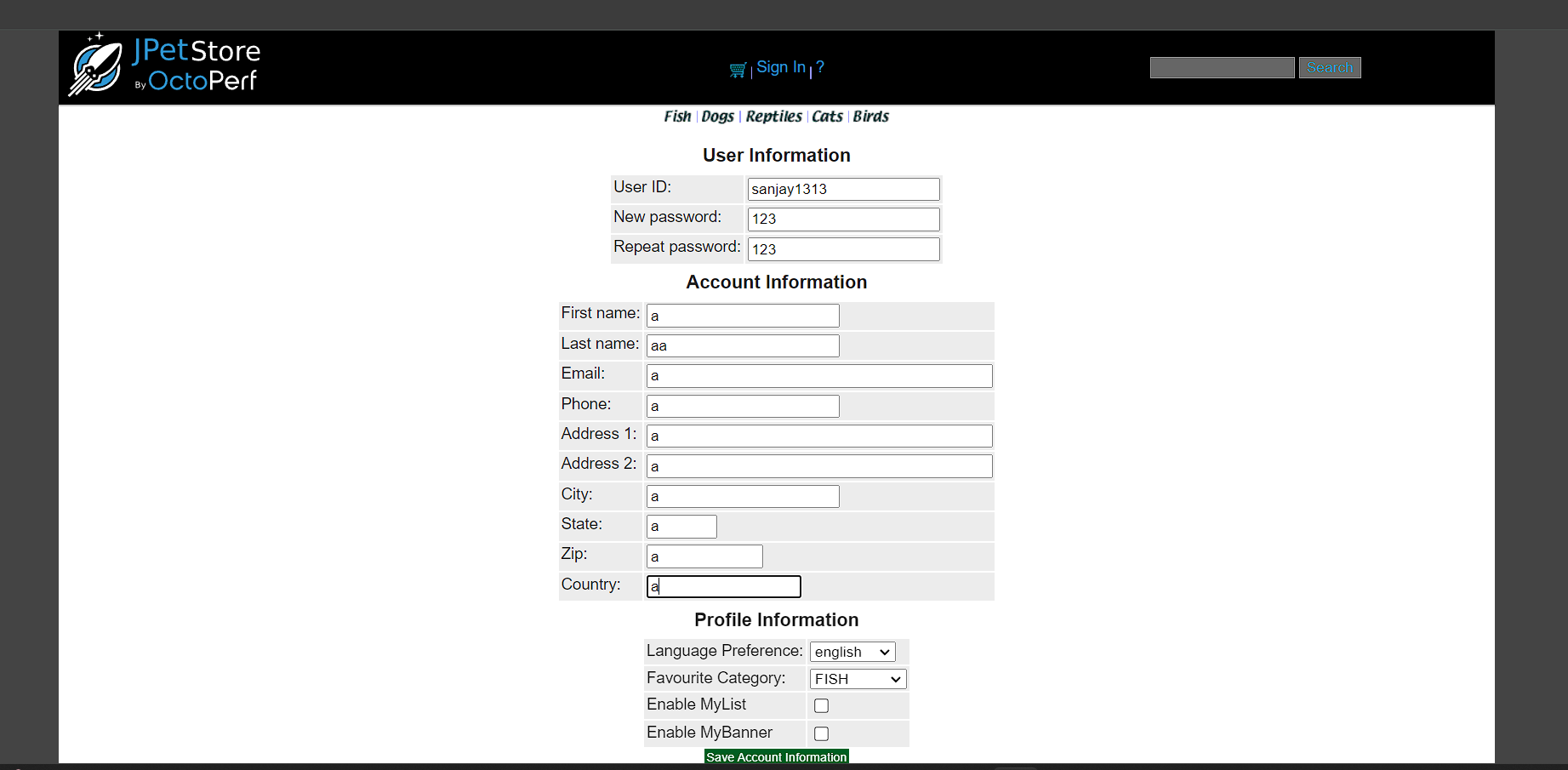
## **1.1 Description**

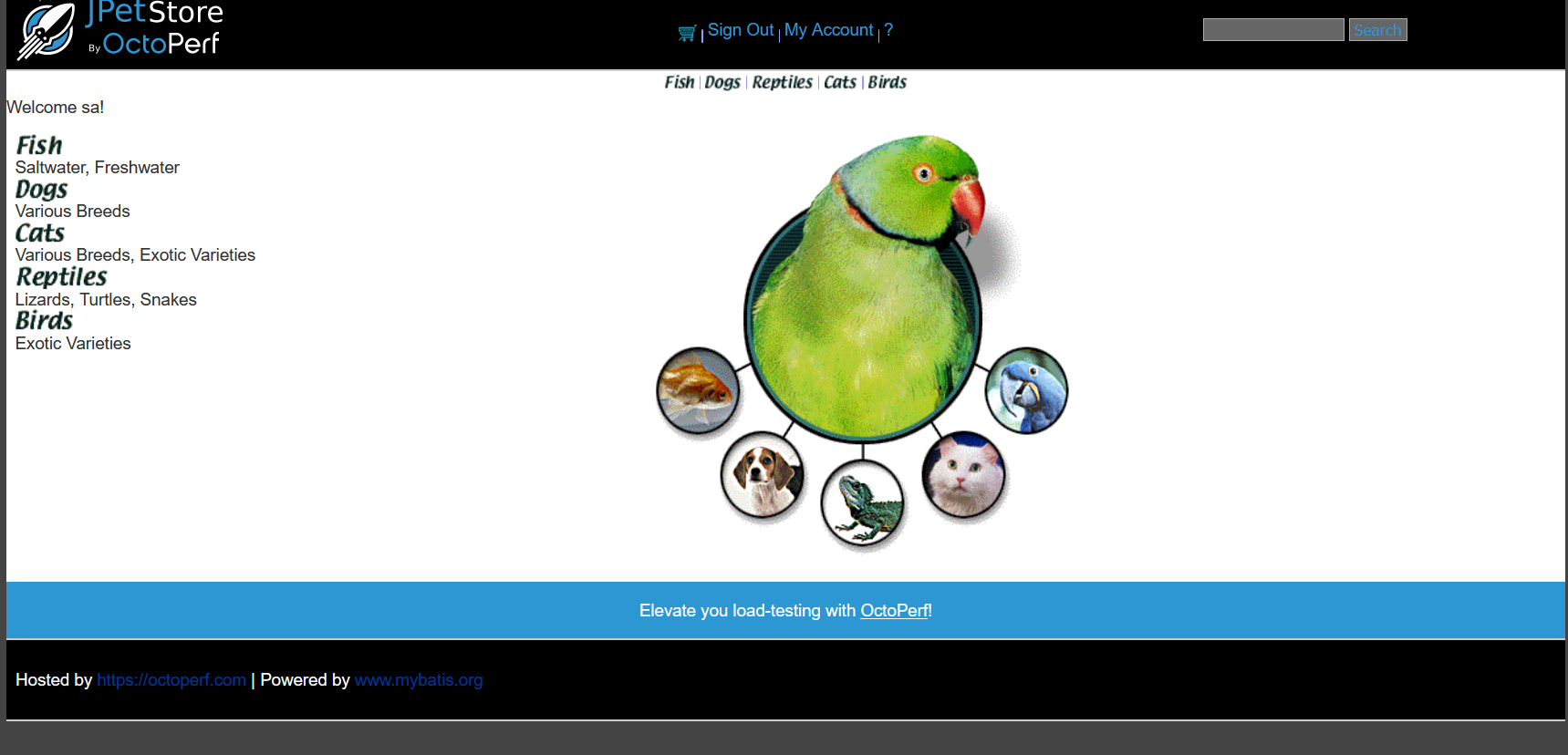
A critical security concern exists with the web application's registration form. The form currently lacks a password policy, meaning there are no enforced requirements for users to create strong passwords. This is problematic because weak passwords are easily guessed or cracked by hackers, putting user accounts and potentially sensitive data at risk. To improve security, the application should implement a password policy that mandates a minimum password length, along with complexity requirements that include a combination of uppercase and lowercase letters, numbers, and symbols.

## **1.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Account.action?newAccountForm=>

## **1.3 Proof of concept.**

**Step 1 :** go to the registration page then just fill the all box and set a weak password ,like 123 . we can see that the web application will not ask for a strong password.

**Step 2:** use the weak password we can login there is not password policy.

## **1.4 Mitigation**

**Minimum Length:** Set a minimum password length, ideally 12 characters or more. Longer passwords are exponentially harder to crack through brute-force attacks.

**Implement Multi-Factor Authentication (MFA):**MFA adds an extra layer of security beyond just a password. It requires users to provide a second verification factor.

# **2.HTTP parameter tampering**

## **2.1 Description**

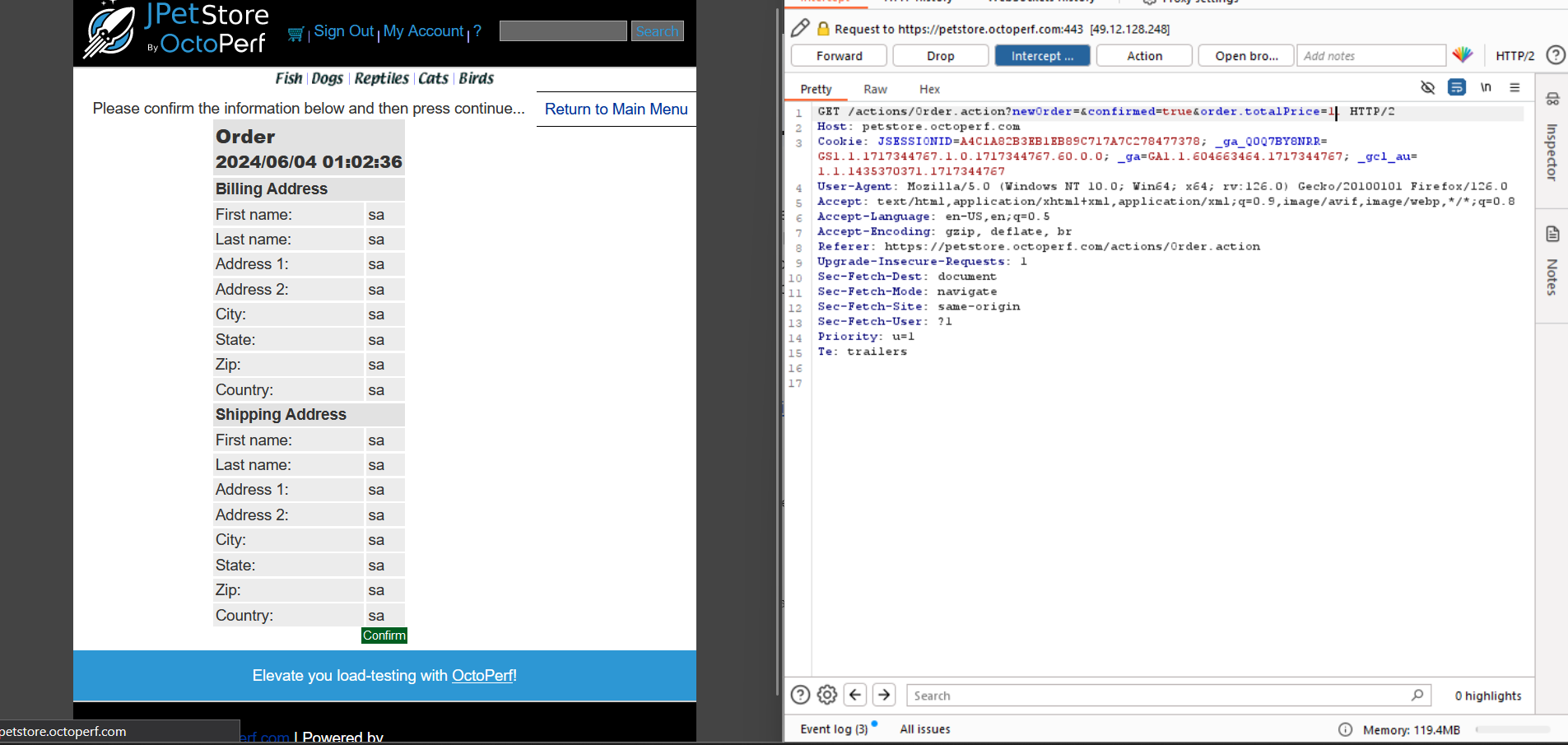
A serious security flaw exists within the website's checkout process. During the final confirmation page before payment, a vulnerability allows users to manipulate the 'totalPrice' field. This means someone could potentially alter the displayed total amount to a lower value. This poses a significant financial risk, as it enables unauthorized price changes that could lead to lost revenue for the website.

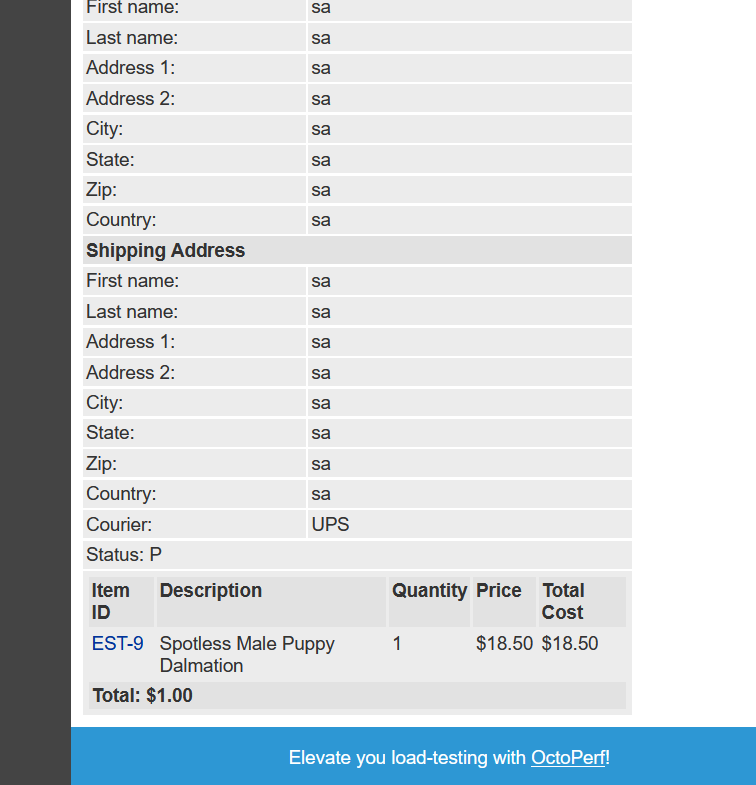
## **2.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Order.action>

## **2.3 Proof of concept.**

**Step1 :** add a product to the cart then place the order. At that time we can add a new parameter to the request ‘order.totalPrice=1”



**step 2:** we can see that the product final price is changed to 1.

## **2.4 Mitigation**

**Order Review:** Include a clear and detailed order review section on the confirmation page. This should display the breakdown of items, quantities, and individual prices, allowing users to verify the accuracy of the total price before submitting the order.

**Secure Data Transmission:**  Ensure data transmission between the user's browser and the server is encrypted using HTTPS.

# **3.Inventory Management Issue**

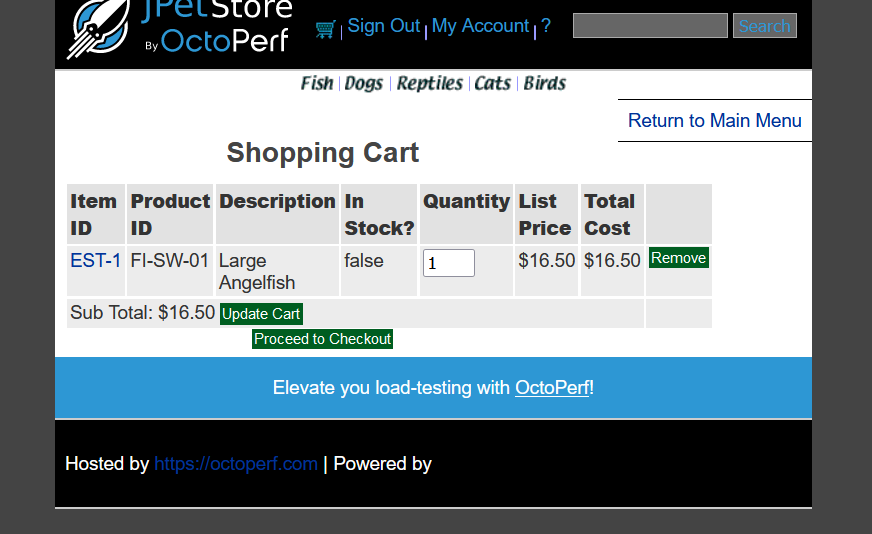
## **3.1 Description**

The web application contains a critical inventory management vulnerability. This security flaw enables users to bypass the intended stock checks and place orders for products that are currently unavailable. This can lead to order fulfillment issues, customer frustration, and potential lost revenue for the business. It's crucial to address this vulnerability by implementing robust inventory management practices and order processing controls.

## **3.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Order.action?newOrder=&confirmed=true>

## **3.3 Proof of concept.**

**Step1 :** first add an out of stock product.

**Step 2 :** Then proceed to the checkout ,place an order, that time we can see that we can buy the product if the product is out of stock .

## **3.4 Mitigation**

**Real-time Inventory Tracking:** Implement a system that accurately reflects real-time inventory levels.

**Disable Order Button for Out-of-Stock Items:** Automatically disable the "Add to Cart" or "Order Now" button for products that are out-of-stock.

# **4.Design vulnerability**

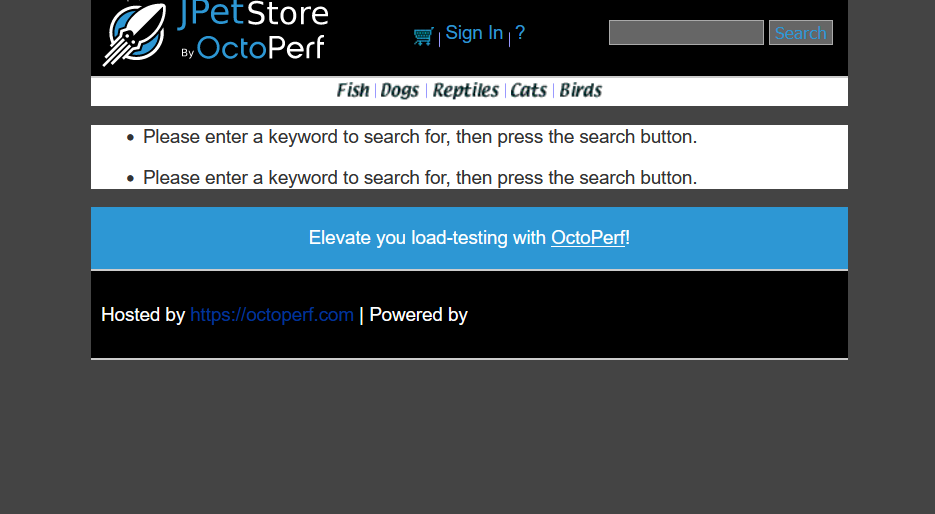
## **4.1 Description**

In this web app there is a vulnerability . there is a search ,if we click search button without any input we can see that the website display 2 alert “Please enter a keyword to search for, then press the search button“

## **4.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Catalog.action>

## **4.3 Proof of concept.**

The web application's search functionality has a minor usability issue. When a user clicks the search button without entering any keywords, two alert messages reminding them to enter a search term. While these alerts aim to be helpful, they can be disruptive to the user experience. Ideally, the search function should handle empty searches gracefully without requiring these unnecessary alerts.

## **4.4 Mitigation**

**Secure Design Patterns:** Utilize established secure design patterns that have been vetted by security experts. These patterns provide a blueprint for building secure applications and address common design flaws.

# **5.sensitive data exposure**

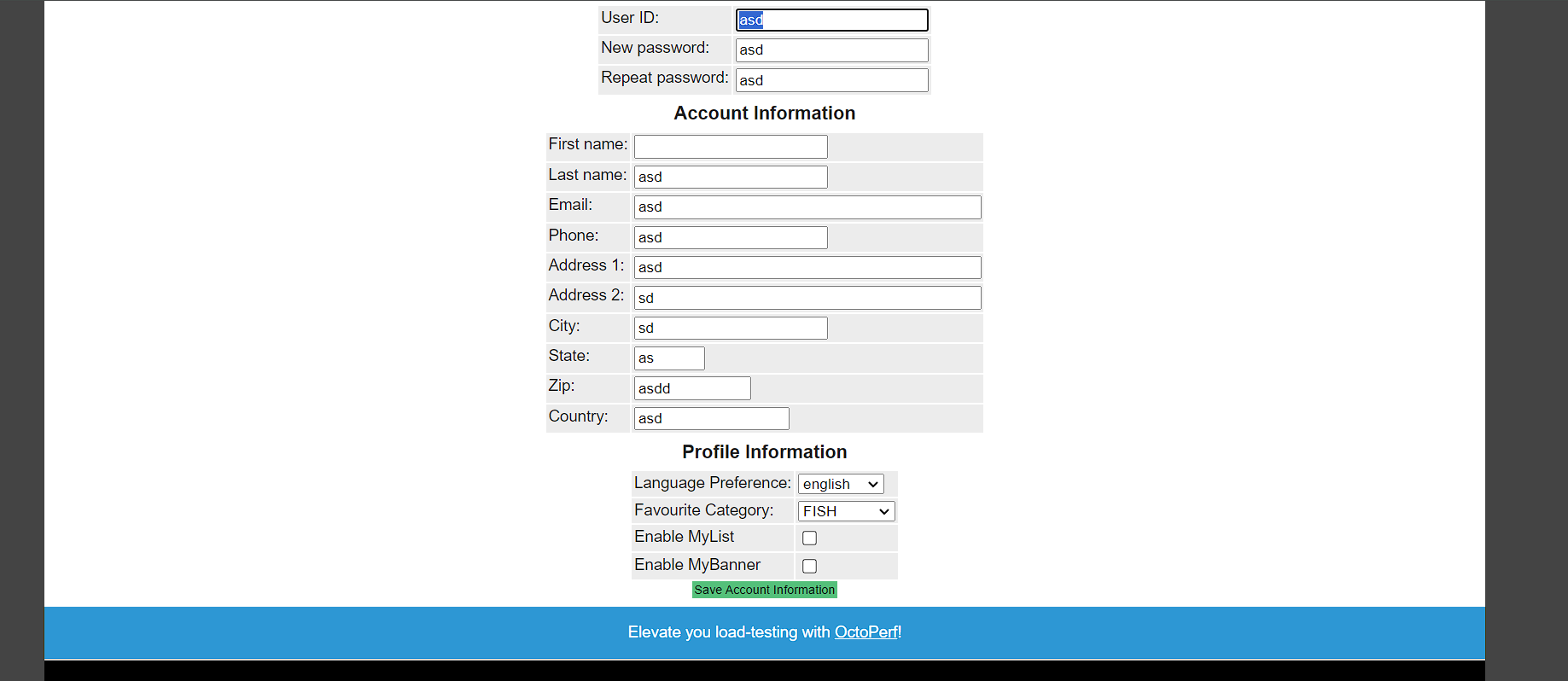
## **5.1 Description**

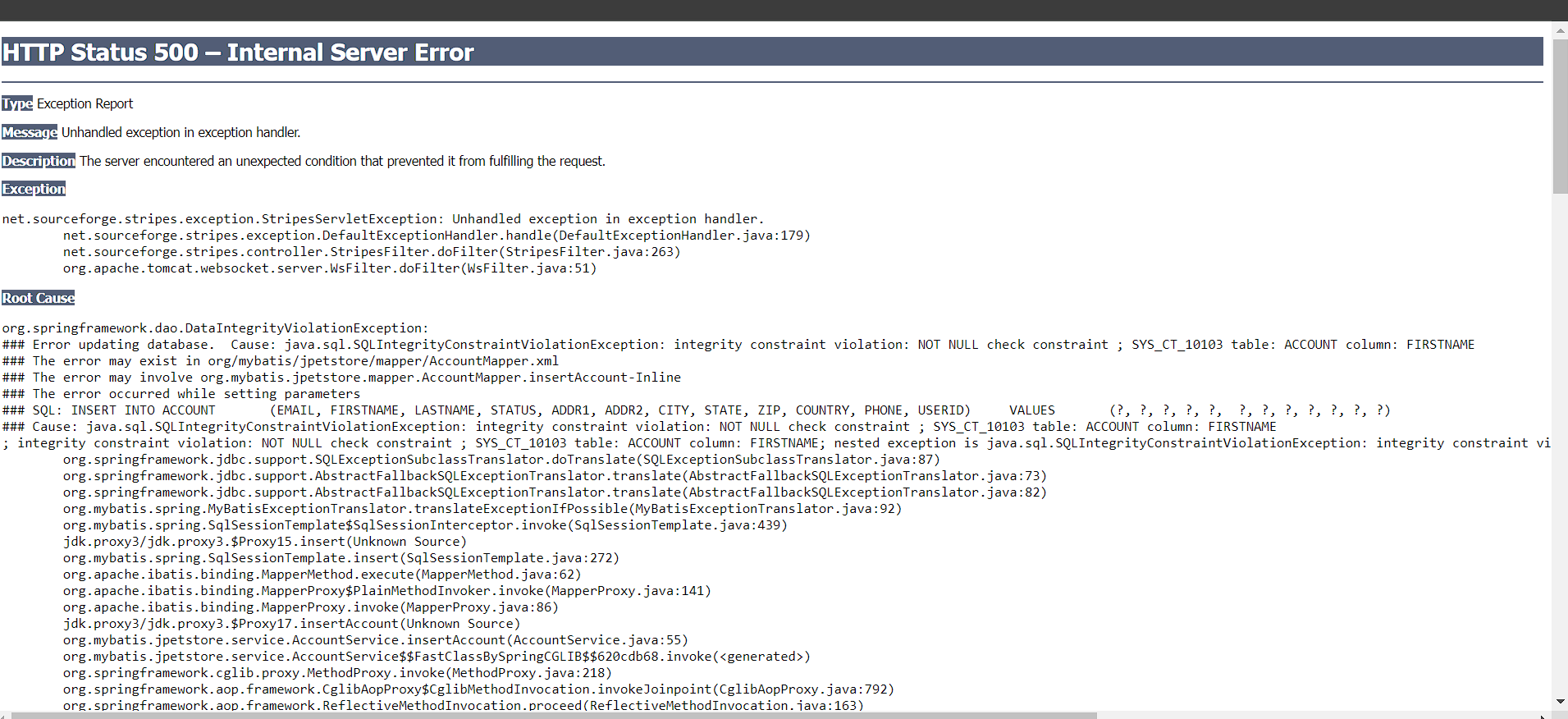
The web application's registration page has a critical security vulnerability. When a user attempts to register without completing any of the required fields, the error message displayed exposes sensitive information. This information includes technical terms like "Exception" and "Root Cause.

## **5.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Account.action>

## **5.3 Proof of concept.**

**Step 1:** Go to the registration page then fill all the input boxes to avoid one box .

**Step 2 :** Then press enter we get a status code error message .in that error we get the root cause.

## **5.4 Mitigation**

**Secure Error Handling:** Design the error handling mechanism to avoid exposing sensitive information in error messages.

**Input Validation:** Implement robust input validation on the client-side (user's browser) and server-side (web application server).

# **6.Cross-Site Scripting (xss)**

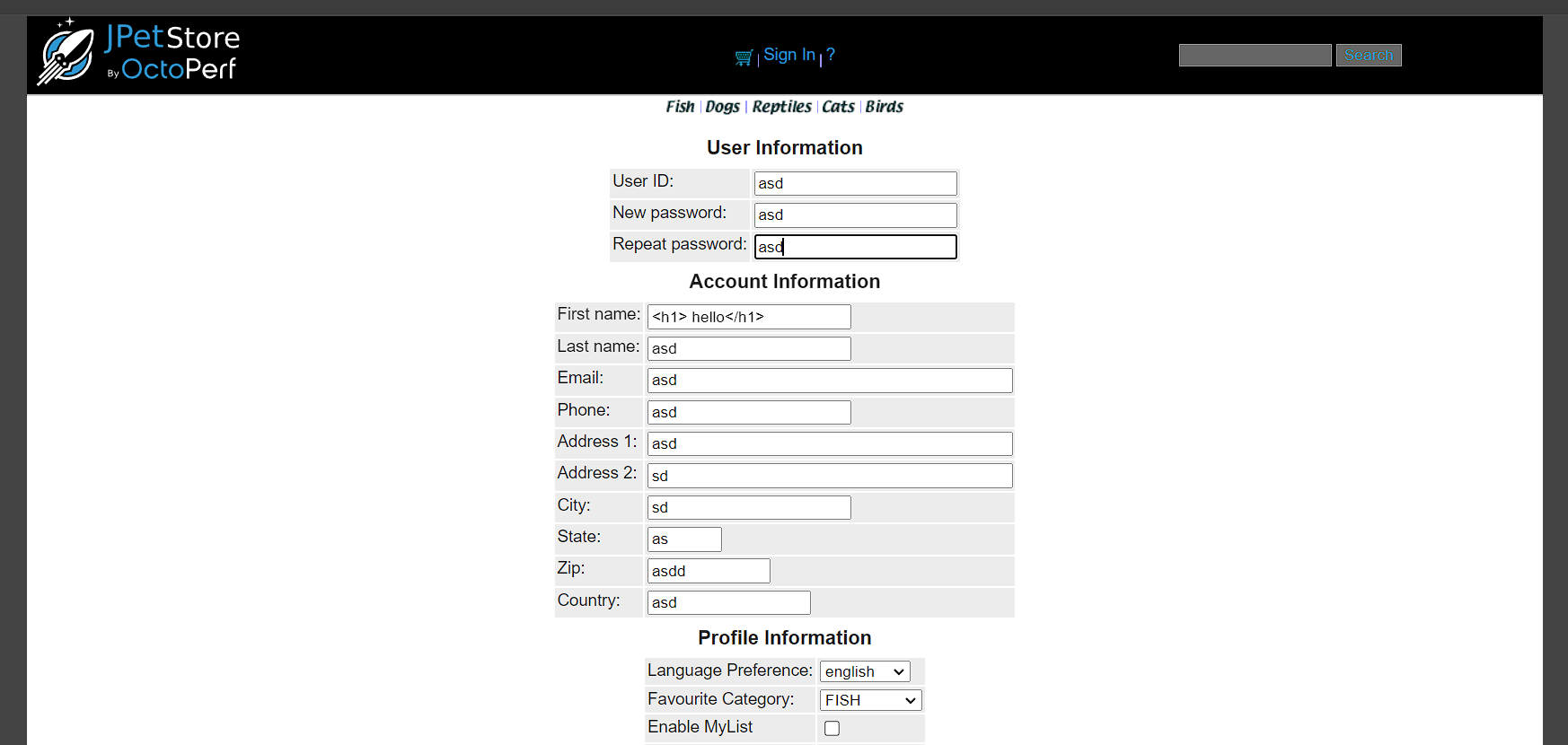
## **6.1 Description**

The web application's registration page contains a critical security issue known as a Cross-Site Scripting (XSS) vulnerability. This vulnerability allows malicious actors to inject scripts into the registration form. These scripts could then be executed in a user's browser when they visit the registration page.

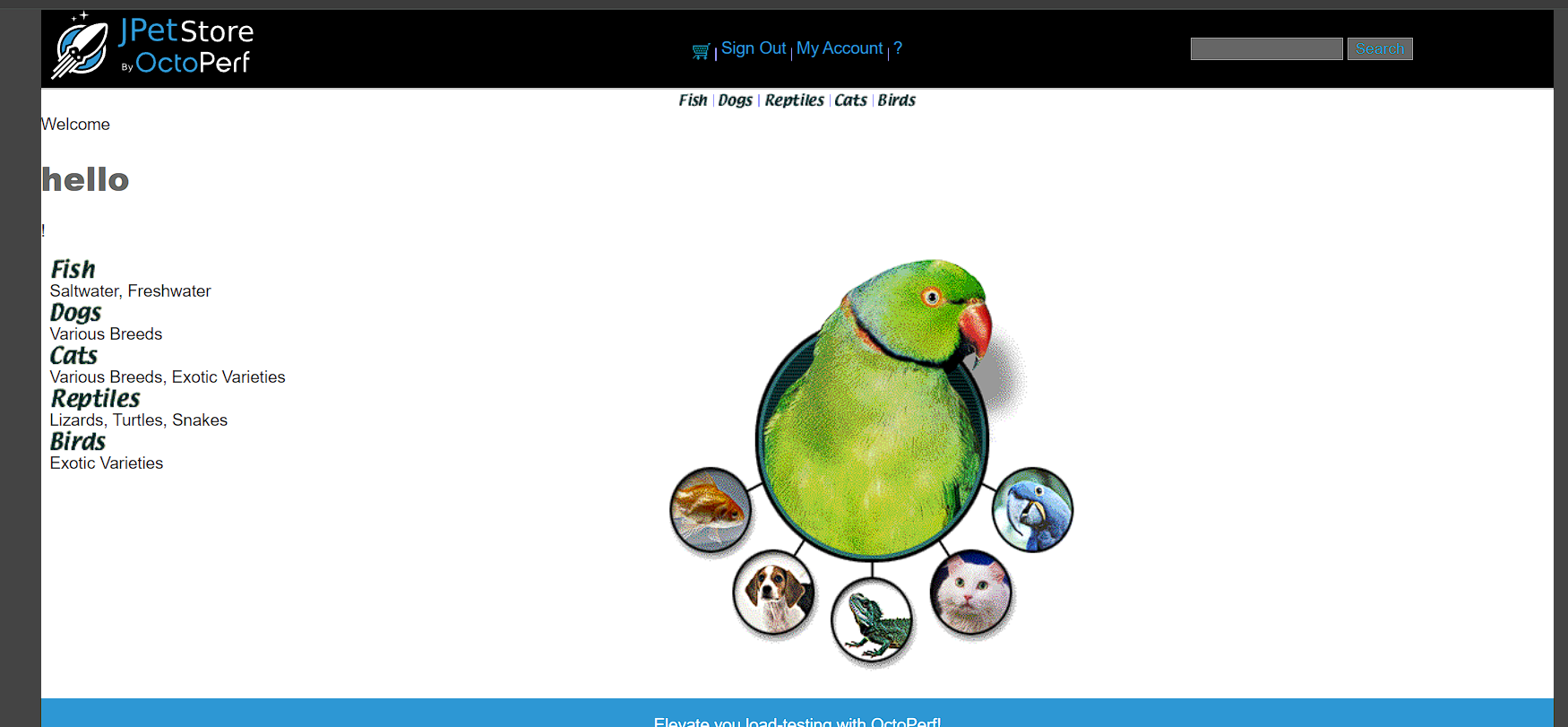
**6.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Account.action?newAccountForm=>

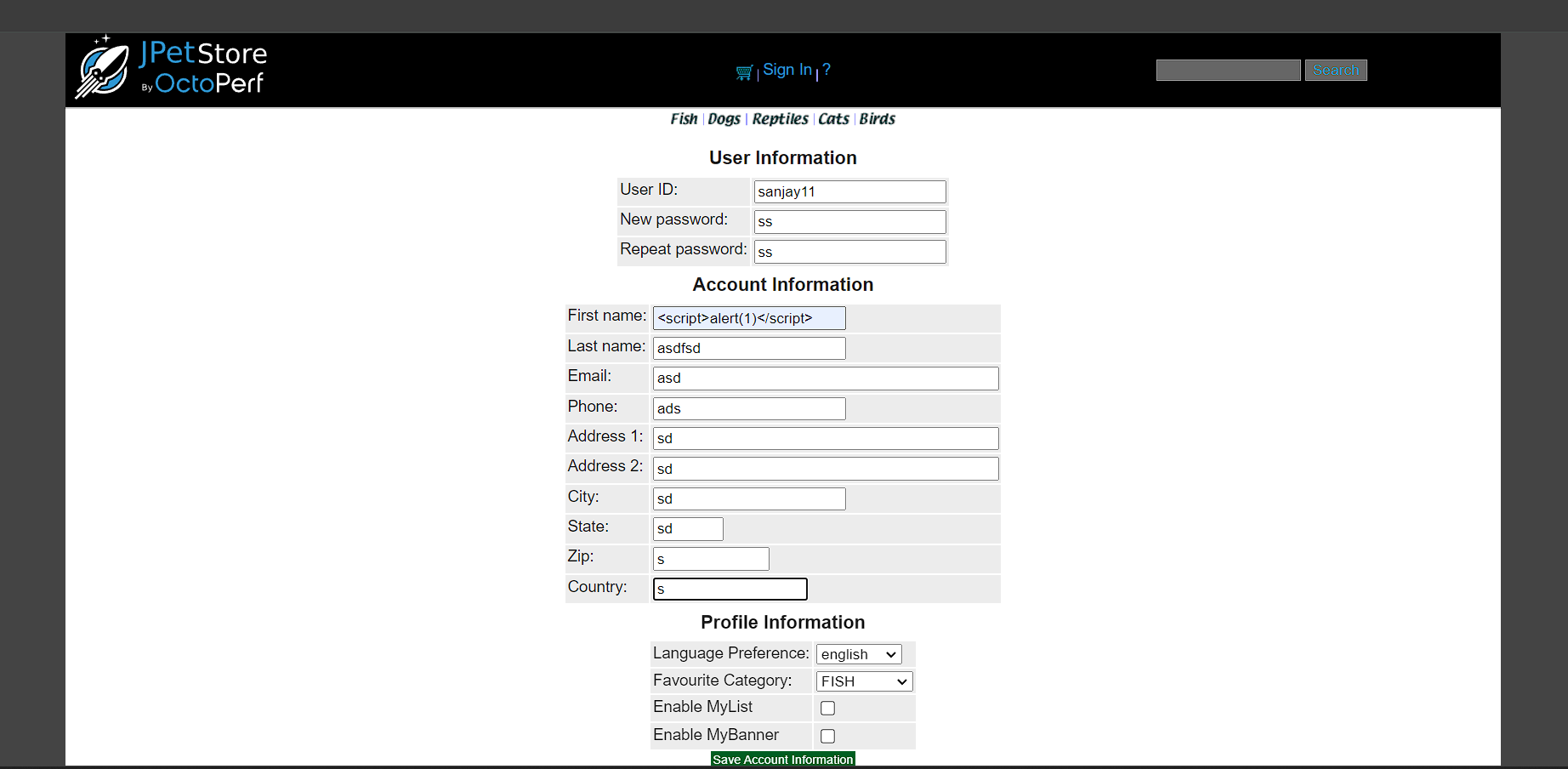
## **6.3 Proof of concept.**

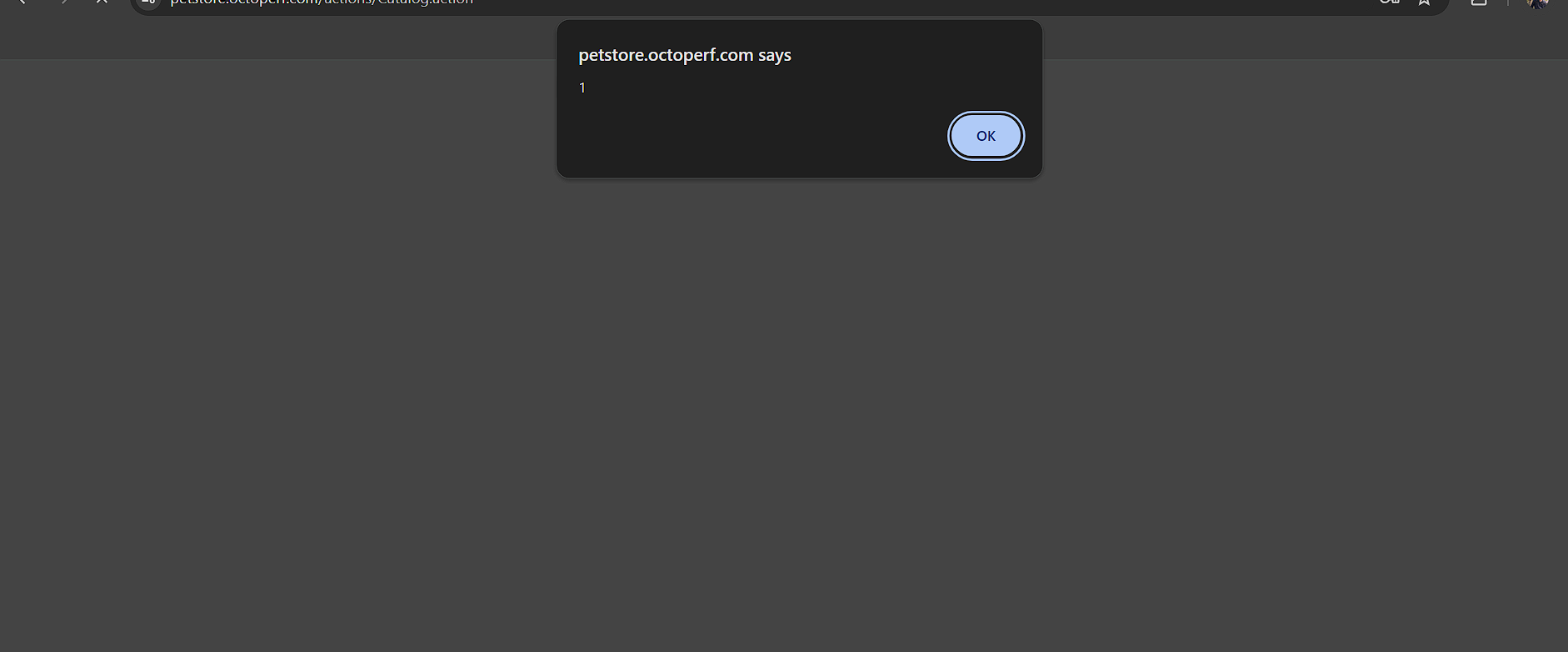
**Step 1:** go to the registration page . add this script”<h1>hello</h1>” in the input box.

**Step 2 :** We can see that the script is executed on the web page .



Step 3 : go to the register page and add ‘<script>alert(1)</script>” this script to the input box .





We can see the script is running without any validation.

## **6.4 Mitigation**

**Validate User Input:** Implement robust validation on the server-side to ensure user input conforms to expected formats .

**Sanitize User Input:** Before displaying or storing user input, sanitize it to remove any harmful code or scripts

# **7.improper validation**

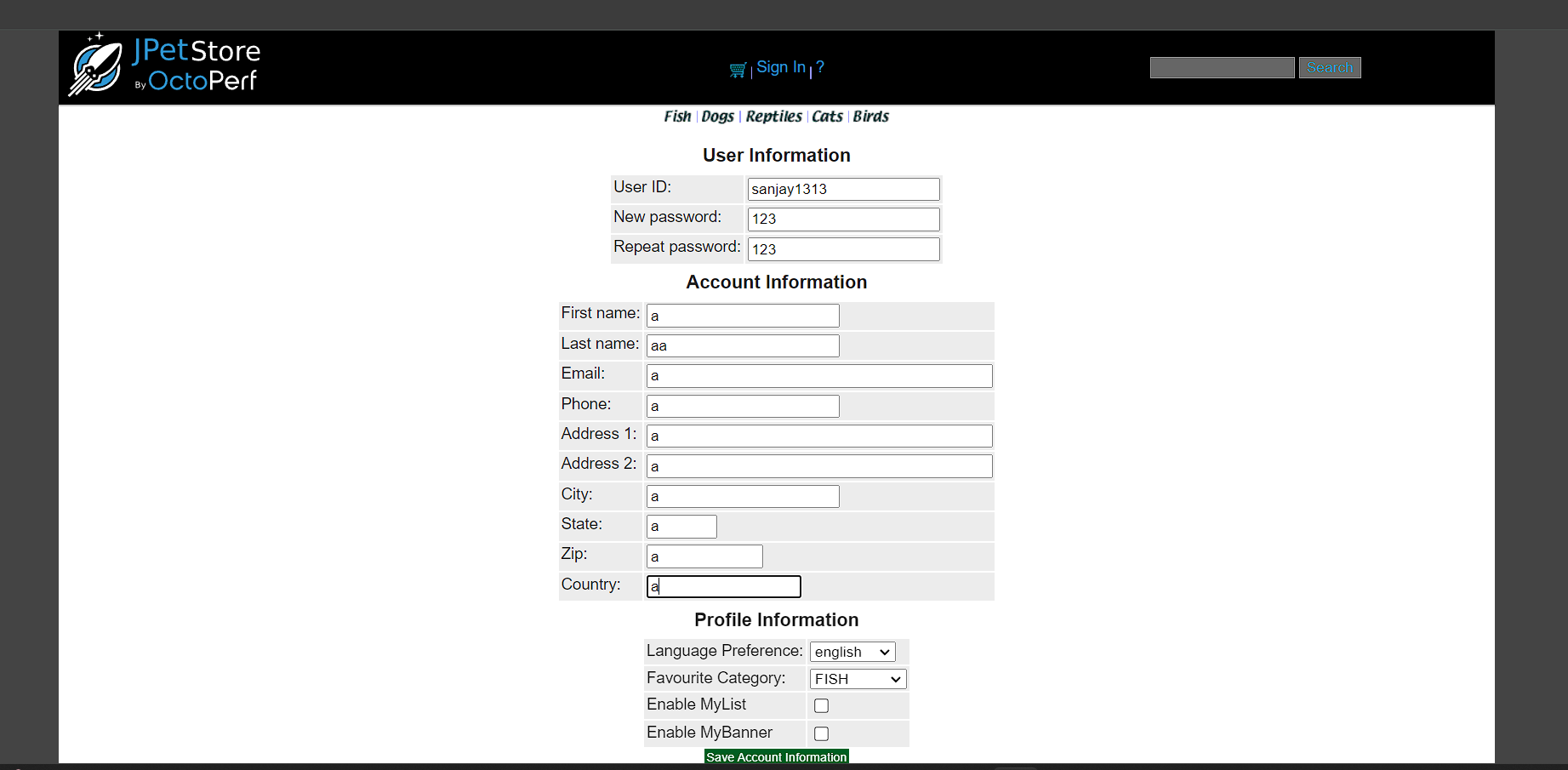
## **7.1 Description**

The web application suffers from a security vulnerability due to improper input validation. Specifically, the registration process doesn't validate phone numbers and email addresses effectively. This means a user could enter anything, including letters or symbols, in these fields.

## **7.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Account.action?newAccountForm=>

## **7.3 Proof of concept.**

In the restoration page there is no validation . we can see that we can add any data in the email,phone number options no need of proper form.

## **7.4 Mitigation**

**Email Verification:** For email addresses, consider sending a verification email to confirm the user's ownership and prevent fake accounts.

**Length Checks:** Enforce minimum and maximum character lengths for phone numbers and email addresses to prevent incomplete or nonsensical entries.

**Sanitize user input:** After validation, sanitize user input to remove any remaining malicious code or characters that might have slipped through. This helps prevent further vulnerabilities down the line.

# **8.Session management error**

## **8.1 Description**

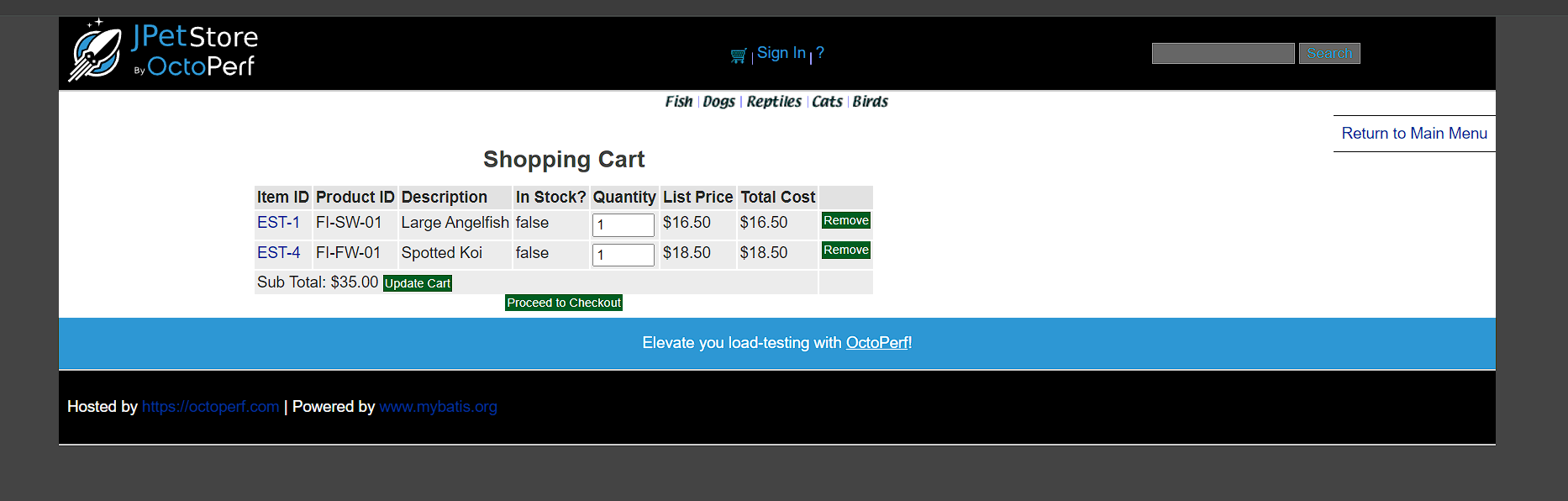
The web application seems to have a functionality issue related to shopping carts. When users add items to their cart, log out, and then log back in, the cart appears empty. This means the items they added previously are not saved and cannot be retrieved after logging out.

## **8.2 Vulnerable instance**

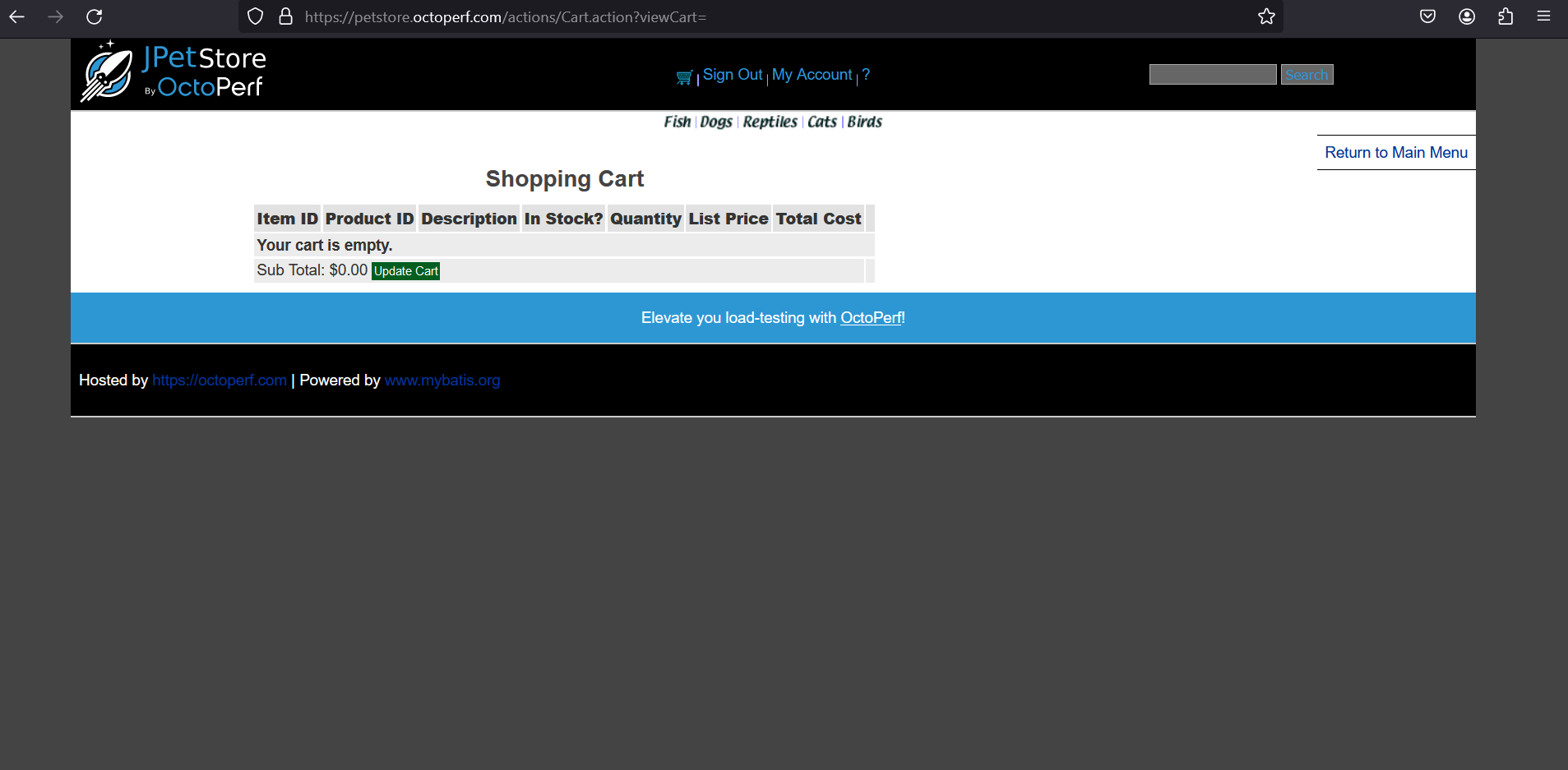
<https://petstore.octoperf.com/actions/Cart.action?viewCart=>

## **8.3 Proof of concept.**

**Step 1 :** Firstlogin then add some products in the cart.



**Step 2:** Then logout and login once more,we can see the cart is empty.



## **8.4 Mitigation**

**Implement a session-based cart system.** When a user visits the website, a unique session ID is generated and stored on their browser. This session ID can then be used to link any added items to that specific user, even if they haven't logged in yet.

# **9.Lack of rate limiting**

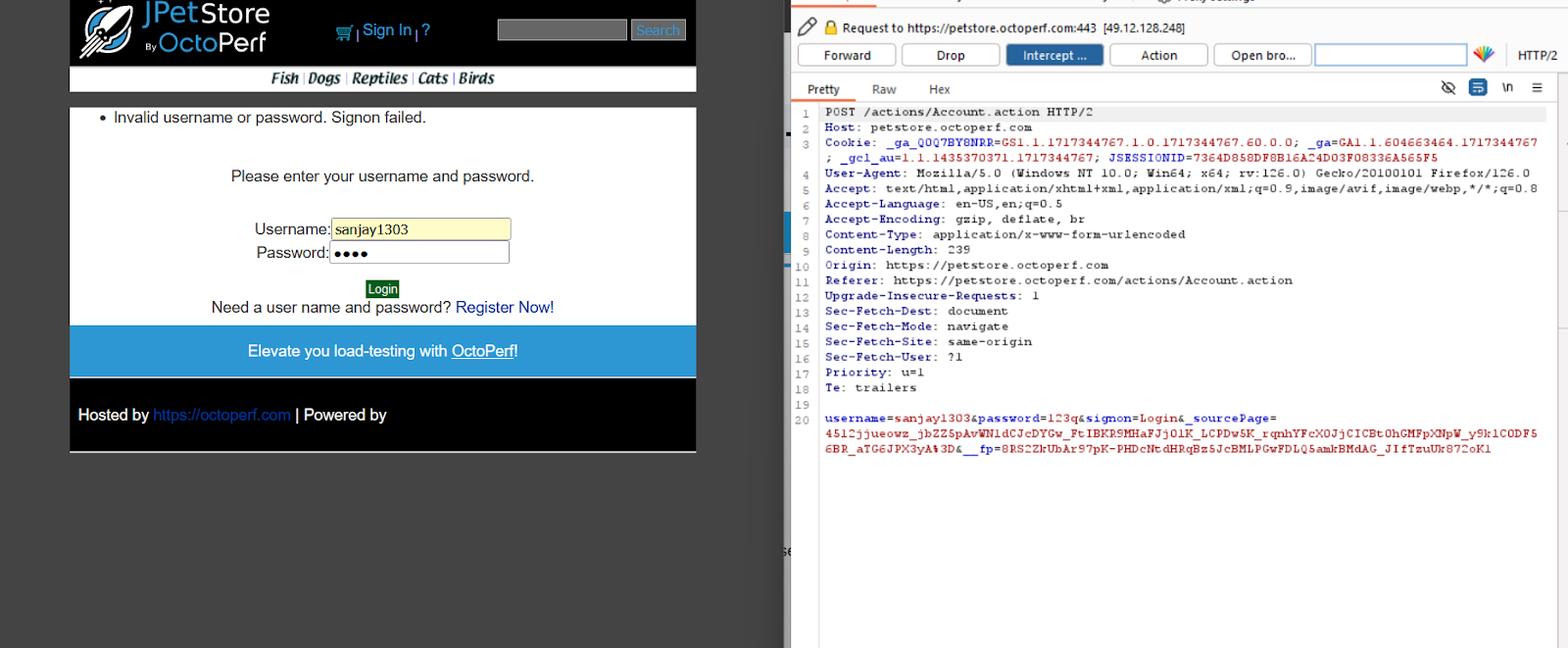
## **9.1 Description**

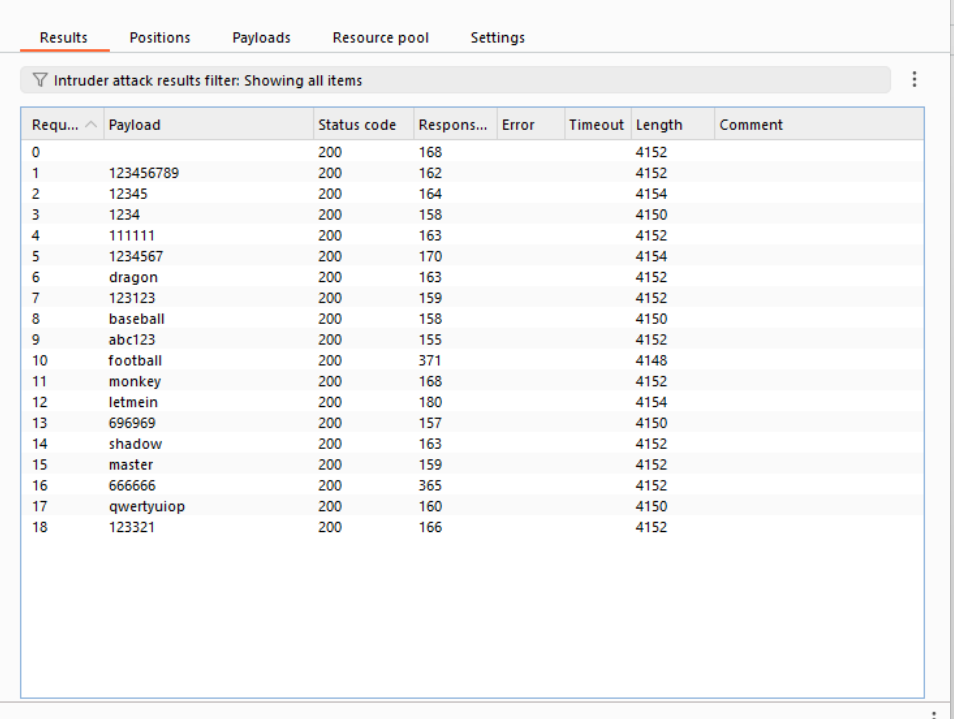
The website's login page appears to lack a security measure called "rate limiting." This feature restricts the number of login attempts a user can make within a specific timeframe. Without rate limiting, an attacker could exploit a vulnerability by repeatedly trying to guess a user's password through a "brute-force attack."

## **9.2 Vulnerable instance**

<https://petstore.octoperf.com/actions/Account.action?signonForm=>

## **9.3 Proof of concept.**

**Step 1 :** go to the login, add password and username and capture the request 

Step 2: transfer the request to the intruder and start an attack. After starting the attack we can see that the request made and the response getting 200 OK for all the request that is sent through the intruder. Every request is getting a 200 OK so we can confirm it’s vulnerable to brute force.

## **9.4 Mitigation**

**Introduce CAPTCHA Challenges:** While rate limiting is essential, consider adding an extra layer of security with CAPTCHAs.

**Implement Rate Limiting:** This is the most common and straightforward approach.

## 