Code Review Report

Repository Name: ascart-vuln/hello

Scanned At: 2023-10-24 16:57:15

File Name	Vulnerability Name	Description	Severity	Vulnerable Code Snippet	Remediation
bl.py	Cross-Site Scripting (XSS)	The code uses user-supplied input in a call to the render_template() function without proper validation, allowing for potential cross-site scripting attacks.	Medium	`render_template('withdraw.html', username = user.User, balance = str(user.Balance), message = withdraw_msg)`	Use a secure template engine or properly sanitize user input before passing it to the render_template() function.
bl.py	Insecure Configuration	The code is using the debug=True flag, which can expose sensitive information and allow for potential attacks.	Low	`app.run(debug=True)`	Set the debug flag to False in production environments to prevent sensitive information from being exposed.
bl.py	Unvalidated Input	The code is using user-supplied input without proper validation, allowing for potential attacks.	High	`withdrawAmount = float(paramURL.get('amount'))`	Use input validation or type checking to ensure user-supplied input is safe and appropriate for the intended use.
bl.py	Sensitive Information Exposure	The code is using the debug=True flag, which can expose sensitive information and allow for potential attacks.	Medium	`app.run(debug=True)`	Set the debug flag to False in production environments to prevent sensitive information from being exposed.
met.rb	Missing Authentication	The code does not require authentication for accessing sensitive data or performing privileged actions, making it vulnerable to unauthorized access or manipulation.	High	`return unless Rails.env.production?`	Implement proper authentication mechanisms, such as user login and access control, to restrict access to sensitive data and actions.
met.rb	Hardcoded Credentials	The code contains hardcoded credentials, such as API keys or passwords, which can be easily accessed by attackers.	High	`$statsd.send(push_type, "#{key_prefix}_#{event}", value)`	Store sensitive credentials in a secure location, such as environment variables or a separate configuration file, and access them securely in the code.
met.rb	Code Injection	The code uses user-supplied input in a call to the push_data_to_stat… function without proper validation, allowing for potential code injection attacks.	High	`data.each do |event, value|`	Use prepared statements or input validation to prevent user-supplied input from being executed as code.
met.rb	Sensitive Data Exposure	The code sends sensitive data, such as database credentials or API keys, to an external service or server without proper	Medium	`push_data_to_statsd(data, :count)`	Encrypt sensitive data before sending it to external services or servers, and use secure protocols for communication.

		encryption or protection.			
met.rb	Insecure Data Storage	The code stores sensitive data, such as database credentials or API keys, in an insecure location or format, making it vulnerable to unauthorized access or manipulation.	Medium	`key_prefix = "model_stats_#{Rails.env}"`	Store sensitive data in a secure location, such as environment variables or a separate configuration file, and use encryption or hashing to protect it.
xs.html	Cross-Site Scripting (XSS)	The code uses user-supplied input in a call to the Exploit() function without proper validation, allowing for potential cross-site scripting attacks.	High	`XSSpayload = '<img src onerror=alert(document.domain)>'`	Use input validation or encoding to prevent user-supplied input from being executed as script.