2021

Static Code Analysis

STATIC CODE ANALYSIS ON SYSTEM OF YU ZHICHENG

MICHEL BIJNEN, KRISTIAN HANSEN, ALEXANDER LAMBOOIJ, ROELLUCASSEN, YOURI SAMAN, THOMAS VAN SCHIJNDEL

FHICT ICS7 | Rachelsmolen 1 Eindhoven | 08/01/2021

Introduction

This document has been set-up in order to communicate what vulnerabilities have been found within the git repository of Yu Zhicheng.

This list of vulnerabilities was established by running SonarQube Scanner analyses and performing manual code reviews.

The severity levels are defined using the CVSS version 3. This system calculates a severity score from 0.1 to 10.0 for every found vulnerability. Severity scores are categorized as follows:

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization’s attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Table of contents

Code analysis result

Critical (9.4)	01 - Plain Text password
Location	backend -> res.controller.js -> lines 22-30 backend -> server.js -> lines 39/47
Problem	A plaintext username and password are present in the code.
Recommendations	Move the credentials to environmental variables.

Informational	02 - Metadata missing
Location	backend -> package.json -> lines 2-10
Problem	Metadata about the server is missing.
Recommendations	Fill in proper metadata about the server.

Informational	03 - Comparison with ‘==’
Location	backend -> res.controller.js -> line 128/138
Problem	Comparison verification == req.query.id may cause unexpected type coercion.
Recommendations	Replace with ‘===’.

Informational	04 - A lot of commented code
Location	backend -> everywhere
Problem	A lot of commented code is present which makes the code unreadable.
Recommendations	Remove commented code so the code becomes readable and only add comments which improves readability.

Informational	05 - Wrong formatted robots.txt
Location	frontend -> robots.txt
Problem	robots.txt is formatted wrong.
Recommendations	Add a ‘/’ after disallow.

Informational	06 - No best git practices
Location	Git repository
Problem	Unreadable repository, not according to git best practices. Multiple projects are stored in one repository. No branches are used for different user stories or tasks.
Recommendations	Refactor git repositories according to best practices.

Informational	07 - Duplicate code
Location	backend -> res.controller.js -> lines 125-127/135-137
Problem	Duplicate code is present.
Recommendations	Turn the duplicated code into a reusable function.

Informational	08 - Hardcoded strings
Location	backend -> res.controller.js -> line 67/152
Problem	Hardcoded strings are present.
Recommendations	Move hard coded strings (like urls and emails) to config file or environment variables.

Informational	09 - Assertion with ‘==’
Location	backend -> res.controller.js -> line 49/121/125/135
Problem	Assertion with string boolean ‘active == “true”.
Recommendations	remove ‘==’ and “true” and use if statements like “if (results.active)” or “if (!results.active)”.

Informational	10 - Unnecessary return
Location	backend -> res.controller.js -> line 247
Problem	Return is unnecessary as the last statement in a function with no return value.
Recommendations	Remove return statement.

Informational	11 - Wrong status code
Location	backend -> res.controller.js -> line 52
Problem	Return is unnecessary as the last statement in a function with no return value.
Recommendations	Remove return statement.

Informational	12 - Wrong naming conventions
Location	backend -> res.controller.js -> line 33/64/96/111/150/241 backend -> res.service.js -> line 4/17/30/44/57/84
Problem	Inconsistent naming convention for methods.
Recommendations	Change every method name to camel casing.

Informational	13 - Bad SOLID practices
Location	backend -> res.controller.js backend -> res.service.js
Problem	According to SOLID, classes and methods should not be long and only have a single responsibility. This is not done in these classes and thus decreases readability.
Recommendations	Make sure every method has a single responsibility and all ‘utils’ methods are divided in other classes.

Conclusion

There is one big vulnerability present in the system. This vulnerability has to be fixed before deploying to a production environment.

Take the code smells (category “Informational”) into account. These decrease readability and maintainability. Some of these code smells may cause indirect security issues.