Fortify Security Report

11/3/25

Executive Summary

Issues Overview

On Nov 3, 2025, a source code review was performed over the 001 code base. 119 files, 19,641 LOC (Executable) were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 4 reviewed findings were uncovered during the analysis.

Issues by Fortify Priority Order	
Critical	2
Low	2

Recommendations and Conclusions

The Issues Category section provides Fortify recommendations for addressing issues at a generic level. The recommendations for specific fixes can be extrapolated from those generic recommendations by the development group.

Project Summary

Code Base Summary

Code location: C:/Users/Lq0ne/source/repos/e0e1/result/YSL
Number of Files: 119
Lines of Code: 19641
Build Label: <No Build Label>

Scan Information

Scan time: 01:03
SCA Engine version: 25.3.0.0014
Machine Name: MORGAN2-WIN
Username running scan: Morgan Wang

Results Certification

Results Certification Valid

Details:

Results Signature:

SCA Analysis Results has Valid signature

Rules Signature:

There were no custom rules used in this scan

Attack Surface

Attack Surface:
Private Information:
null.null.null

System Information:
null.null.error
null.null.n
null.null.resolve

Filter Set Summary

Current Enabled Filter Set:
Security Auditor View

Filter Set Details:

Folder Filters:
If [fortify priority order] contains critical Then set folder to Critical
If [fortify priority order] contains high Then set folder to High
If [fortify priority order] contains medium Then set folder to Medium
If [fortify priority order] contains low Then set folder to Low

Audit Guide Summary

Audit guide not enabled

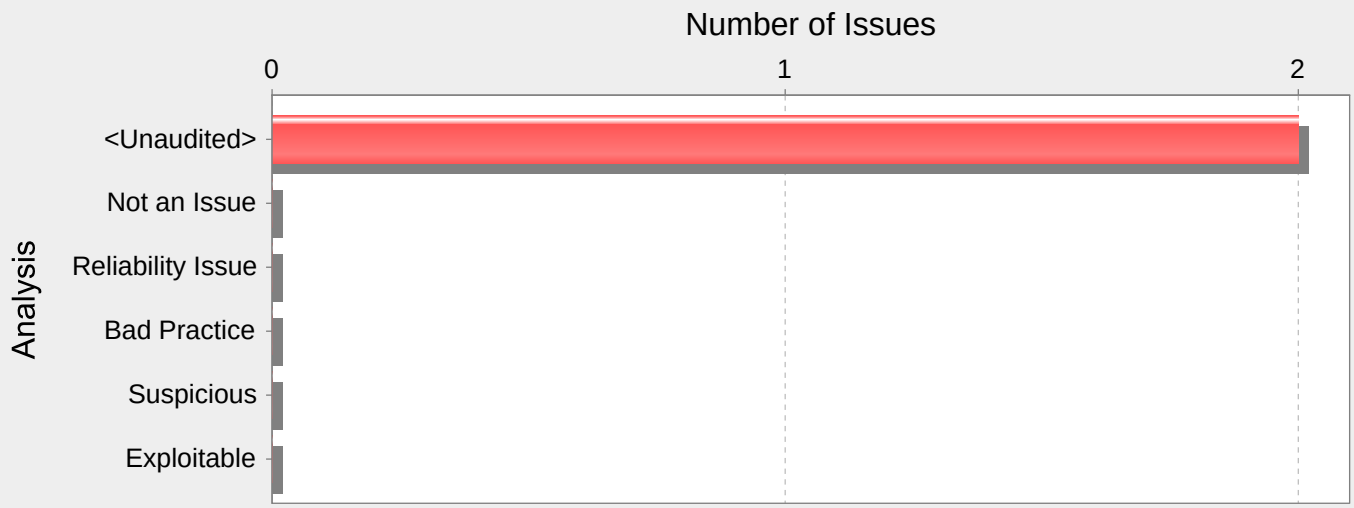
Results Outline

Overall number of results

The scan found 4 issues.

Vulnerability Examples by Category

Category: Key Management: Hardcoded Encryption Key (2 Issues)



Abstract:

Hardcoded encryption keys can compromise security in a way that is not easy to remedy.

Explanation:

Never hardcode an encryption key because it makes the encryption key visible to all of the project's developers, and makes fixing the problem extremely difficult. Changing the encryption key after the code is in production requires a software patch. If the account that the encryption key protects is compromised, the organization must choose between security and system availability.

Example 1: The following example shows an encryption key inside a .pem file:

```
...
-----BEGIN RSA PRIVATE KEY-----
MIICXwIBAAKBgQCtVacMo+w+TFOm0p8MlBWvwXtVRpF28V+o0RNPx5x/1TJTlKEl
...
DiJPJY2LNBQ7jS685mb6650JdvH8uQl6oeJ/aUmq63o2zOw=
-----END RSA PRIVATE KEY-----
...
```

Anyone with access to the code can see the encryption key. After the application has shipped, there is no way to change the encryption key unless the program is patched. An employee with access to this information can use it to break into the system. Any attacker with access to the application executable can extract the encryption key value.

Recommendations:

Never check in encryption keys to your source control system, and never hardcode them. Always obfuscate and manage encryption keys in an external source. Storing encryption keys in plain text anywhere on the system enables anyone with sufficient permissions to read and potentially misuse the encryption key.

app.js, line 2343 (Key Management: Hardcoded Encryption Key)

Fortify Priority:	Critical	**Folder**	Critical
Kingdom:	Security Features		
Abstract:	Hardcoded encryption keys can compromise security in a way that is not easy to remedy.		

Sink: app.js:2343

```
2341                                return t.match(RegExp(n, "g")).join("\n")
2342                            }, e.prototype.getPrivateKey = function() {
2343                                var t = "-----BEGIN RSA PRIVATE KEY-----\n";
2344                                return (t += e.wordwrap(this.getPrivateBaseKeyB64()) + "\n") +
"-----END RSA PRIVATE KEY-----"
```

```
2345                                    }, e.prototype.getPublicKey = function() {
```

Issue Count by Category	
Issues by Category	
Cross-Site Request Forgery	2
Key Management: Hardcoded Encryption Key	2

Issue Breakdown by Analysis

Issues by Analysis