VULNERABILITY CHECK

11. TOUKOKUUTA 2025

JYRI PAPPINEN

Sisällys

Start

This thesis examines a previously completed hobby project from an information security perspective. The aim is to identify and fix potential vulnerabilities as well as to consider possible future development solutions. The project was chosen due to its sufficient scope and personal interest, making it a suitable subject for a security analysis and offering an opportunity for learning to support future development work.

Software

The application is a simple patient information system, which begins with the patient registering in the system. After registration, a doctor can add diagnoses to the patient's profile. The system consists of a separate frontend, backend, and a JSON-based data store.

Since the application deals with patient data, it involves special categories of personal data as defined by the GDPR. Collecting this data requires the user's explicit consent and a legitimate purpose—such as processing information for the purpose of issuing a diagnosis.

Security has been enhanced in several ways. The application includes different user roles to restrict access based on privileges. Additionally, a social security number-based check is implemented during registration to allow people with the same name to register uniquely.

At the code level, security is improved by storing only hashed versions of passwords and treating all user input as plain text, which helps prevent potential SQL injection attacks. When logging in, a token is generated for the user with an expiration time of one hour. This ensures that sessions left open cannot be misused after a certain time has passed.

The application has been thoroughly tested using unit and integration tests, as well as end-to-end testing to ensure the system functions correctly as a whole.

Vulnerabilities

The work began by creating an SBOM (Software Bill of Materials) file, which lists all the dependencies used in the application. During the initial review, several vulnerabilities were discovered in both the frontend and backend, some of which were classified as critical. These issues were quickly resolved by updating the dependencies to their latest versions. After the updates, no vulnerabilities were found in the frontend.

However, one critical and one undefined vulnerability remained in the backend. These were examined separately, and it was found that the vulnerabilities were still present even in the latest versions of the dependencies. As a result, the code was reviewed in more detail to determine whether these vulnerabilities could be exploited within this specific project. The review concluded that the identified vulnerabilities were not applicable in this context and did not pose a direct risk to the system's operation.

	Risk Score	Critical	High	Medium	Low	Unassigned
Before update fronend	88	0	1	1	0	16
After update frontend	0	0	0	0	0	0
Before update backend	87	0	3	4	0	12
After update backend	8	0	0	1	0	1

TABLE 1. Before and after vulnerabilities

PICTURE 1. Backend before update

PICTURE 2. Backend after update

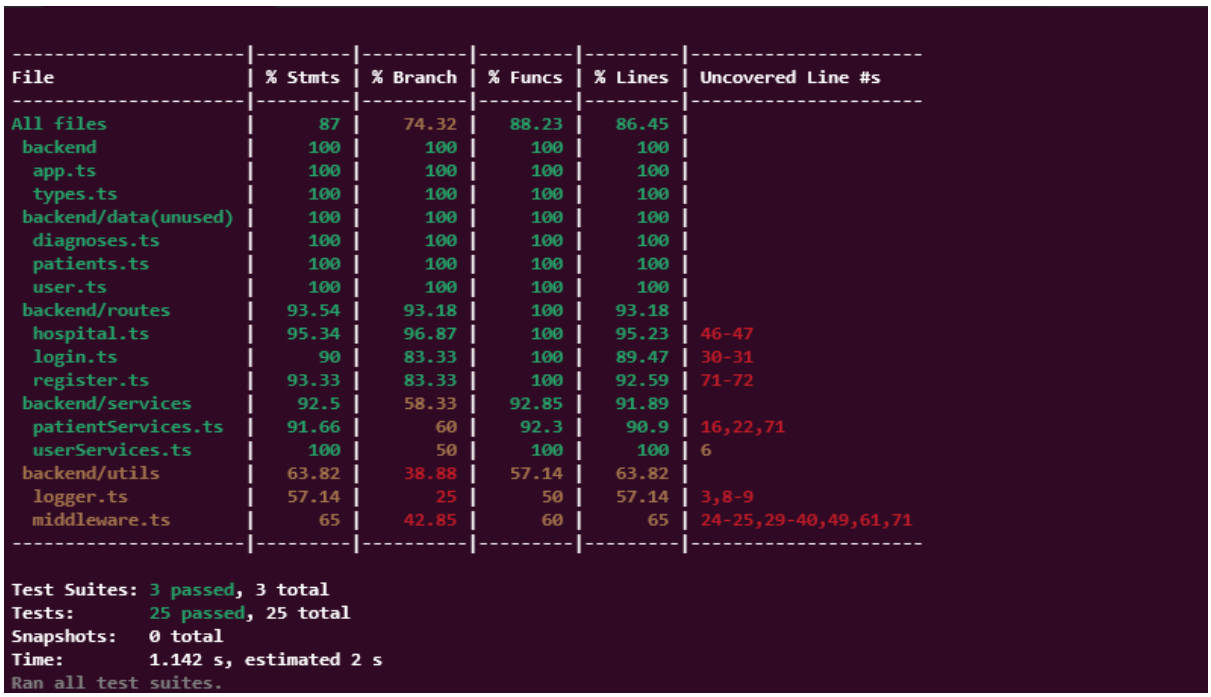
PICTURE 3. Frontend before update

PICTURE 4. Frontend after update

Other updates

After all the changes were made, unnecessary logging was removed to ensure that the application would not output excess information to the browser console. This helps reduce the risk of exposing system logic or sensitive data that could be misused.

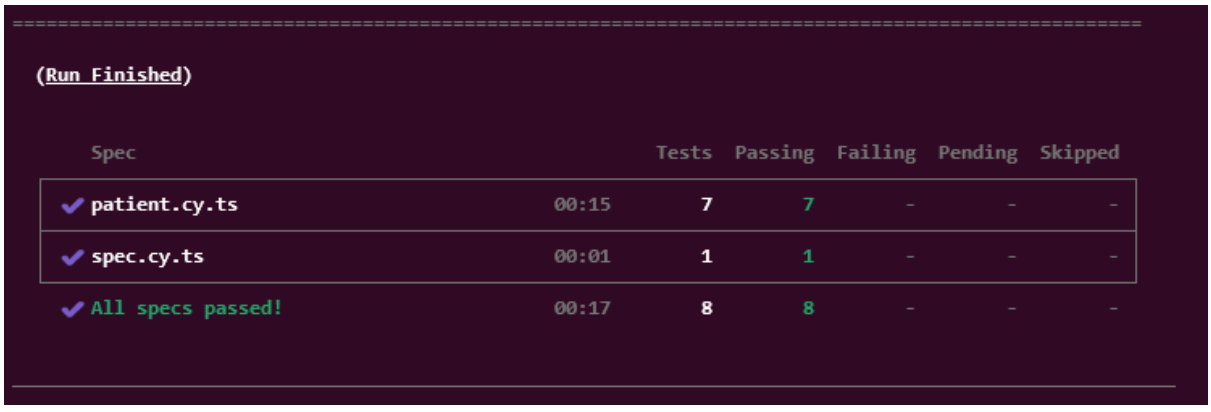
Finally, it was verified that the implemented changes did not break the functionality of the application. Manual testing was first conducted on key features, followed by automated tests. All tests passed without significant issues, confirming the stability of the application after the updates.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	87	74.32	88.23	86.45	
backend	100	100	100	100	
app.ts	100	100	100	100	
types.ts	100	100	100	100	
backend/data(unused)	100	100	100	100	
diagnoses.ts	100	100	100	100	
patients.ts	100	100	100	100	
user.ts	100	100	100	100	
backend/routes	93.54	93.18	100	93.18	
hospital.ts	95.34	96.87	100	95.23	46-47
login.ts	90	83.33	100	89.47	30-31
register.ts	93.33	83.33	100	92.59	71-72
backend/services	92.5	58.33	92.85	91.89	
patientServices.ts	91.66	60	92.3	90.9	16,22,71
userServices.ts	100	50	100	100	6
backend/utils	63.82	38.88	57.14	63.82	
logger.ts	57.14	25	50	57.14	3,8-9
middleware.ts	65	42.85	60	65	24-25,29-40,49,61,71

Test Suites: 3 passed, 3 total
Tests: 25 passed, 25 total
Snapshots: 0 total
Time: 1.142 s, estimated 2 s
Ran all test suites.

PICTURE 5. Backend integration tests



(Run Finished)

Spec		Tests	Passing	Failing	Pending	Skipped
✔ patient.cy.ts	00:15	7	7	-	-	-
✔ spec.cy.ts	00:01	1	1	-	-	-
✔ All specs passed!	00:17	8	8	-	-	-

PICTURE 6. Frontend end-to-end tests

Future

The audit revealed a critical security vulnerability: the data repository was not protected in any way. If an external user discovers the address of the data repository, they can immediately access all patient data. This creates a serious cybersecurity risk that should be addressed as soon as possible.

Next, consideration could be given to limiting the number of requests a single user can make within a certain time period. This limitation would help prevent potential botnet attacks, such as brute-force and DDoS attacks.

Although the program is currently running locally, it is important to ensure that the program uses the HTTPS protocol in the future, which would protect data transmission and prevent data leakage.

Finally, a logging system could be added to the program to record important events, such as failed login attempts or other actions performed within the program. This would enhance the ability to detect potential misuse and assist in investigating anomalous situations if they arise.

Summary

Some aspects of security had already been taken into account in the project, but significant shortcomings were also identified. When it comes to handling people's personal data, caution is never excessive – every protective measure is a step toward a more reliable and secure system.