

Security Report

Fontys University of Applied Science

Software

Project Plan Individual project:



Henaknowledge

Tutor: Tim Kurvers & Maja pesic

Student: Mohammed Al Harbi

4089553

[Contents](#)

Introduction:.....	3
TOP 1: SQL Injection.....	4
TOP 2: Broken Authentication.....	4
TOP 3: Sensitive Data Exposure.....	5
TOP 4: XML external entities.....	5
TOP 5: Broken Access Control.....	5
TOP 6: Security Misconfiguration.....	6
TOP 7: Cross-site scripting.....	6
TOP 8: Insecure Deserialization.....	6
TOP 9: Using Components with Known Vulnerabilities.....	7
TOP 10: Insufficient Logging and Monitoring.....	7
Dependency inject usage proof:.....	8

Introduction:

This document aims to show how Henaknowledge prevents OWASP Top 10 Vulnerabilities.

TOP 1: SQL Injection

Henaknowledge controllers receives data through Data transfer Objects which mimics what an entity class does, but created only for purpose of transferring the data. Typing in one of henaknowledge fields a SQL statement would not perform the SQL action since there are no direct connection between the controllers and the database layer at all.

TOP 2: Broken Authentication

Henaknowledge require a JWT for every request apart from authentication request. It requires the username and the password given in the request then Henaknowledge will send a JWT to the side that requested the authentication. Provided that JWT with BEARER before it is mandatory for every request.

TOP 3: Sensitive Data Exposure

Henaknowledge also uses BCryptPasswordEncoder to increase the difficulty of being hacked or in case of a hacked managed to obtain the database, the users are encrypted at least as last bearer before the hacker sees the information.

Along with the token being mandatory for every request, it makes it even difficult to get any data without it.

TOP 4: XML external entities

Henaknowledge uses Java classes as entities and does not use any external XML file as an entity. Henaknowledge has Model folder which contain all model classes with the annotation Entity.

TOP 5: Broken Access Control

Authentication and Authorization plays huge part in this. Henaknowledge requires clients to first authenticate via a post method providing an object that contains username and password. Then, Henaknowledge either returns the generated JWT if the user exists in the database or returns an error to the client. In case, of a successful authentication and that the username and password belongs to a STUDENT profile, then the generated JWT can NOT have access to ADMIN end points such as GET: "localhost:8080/Admin" to get all the available Admins. Whereas, a successful authentication of a user profile ADMIN will generate a JWT that is able to access all endpoints in the system.

TOP 6: Security Misconfiguration

In Henaknowledge the security configuration are set to disable CSRF to prevent such attacks and configured to have no sessions, STATELESS so that every request apart from the ones that has been configured to be permitted to all users are mandatory to have BEAR JWT in the authorization in the headers of the request.

TOP 7: Cross-site scripting

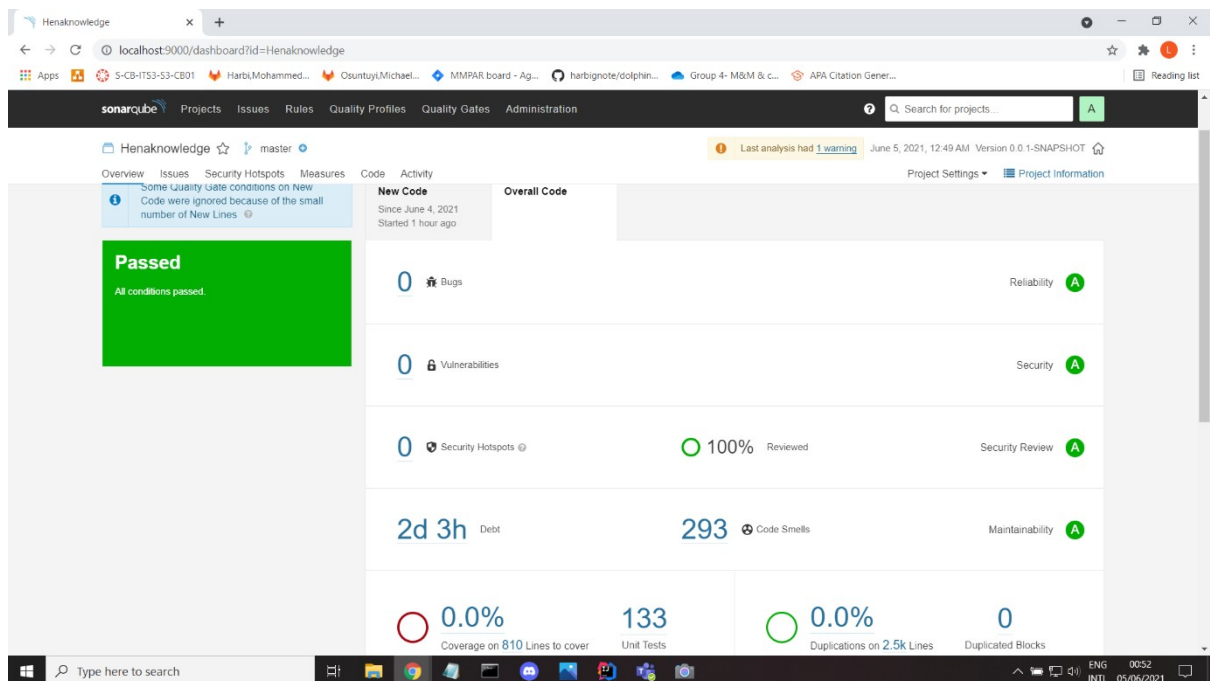
In Henaknowledge frontend it is not possible to retrieve information at all without being logged in. In addition to that only if you are logged in, you have a few privileges for instance as a Student, you can not retrieve admins information so on so forth. Thus, It is kind of hard to use the DOM to obtain data that the user has no privileges to access them.

TOP 8: Insecure Deserialization

Henaknowledge uses DTO as a way to get the data, then use them in the entity class that is connected to the database. Thus, having no deserialization meaning that this not a vulnerability and it cant be leveraged to obtain data.

TOP 9: Using Components with Known Vulnerabilities

This is the latest Henaknowledge version and it has 0 vulnerabilities according to Sonarqube. Jocaco report is also available in the sprint 5 Document folder.



TOP 10: Insufficient Logging and Monitoring

Once a user tries to log in, the client side sends an authentication request and if the provided credentials does not match the database encrypted details the client side reserves an error which then are reflected in the frontend UI. Access-control in this case fails, server-side input validation fails as well.

Dependency inject usage proof:

