**Cybersecurity Report**

01.05.2024

Technology used in the project during the testing:

Backend

* Java 21
* Spring Boot 3.2.4
* io.jsonwebtoken 0.11.2

Frontend

* Javascript ES6
* React 18.2.0
* Tailwind CSS 3.4.1

**Static testing**

**Vulnerability for Json Web Tolken**

An issue here is the potential version of the json token. Currently, the version available is 0.12.5 and it is a good idea that the json version is updated. Right now, 32 vulnerabilities are found related to this token.

1. CVE-2023-5072 with CVSS score 7.8 - **HIGH**

* The vulnerability allows DoS attack on prior versions of Java (before 22) and Json (before 0.12). A bug in the parser means that an input string of modest size can lead to indefinite amounts of memory being used. (MITRE)
* To fix the vulnerability update Java and Json to their latest versions.

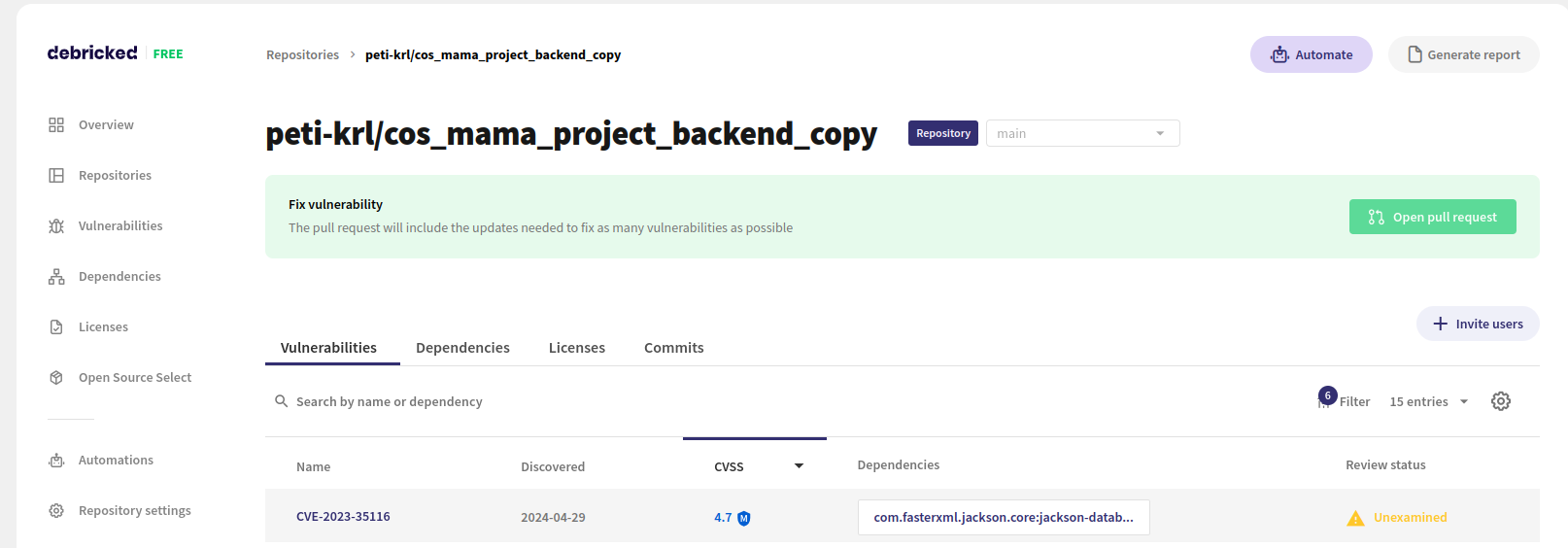
1. CVE-2023-6378 with CVSS score 7.5 - **HIGH**

* A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data. (NIST)
* To fix the vulnerability update Java and Json to their latest versions.

**Vulnerability for jackson-databind**

1. CVE-2023-35116 with CVSS score 4.7 - **MEDIUM**

* jackson-databind through 2.15.2 allows attackers to cause a denial of service or other unspecified impact via a crafted object that uses cyclic dependencies. (NIST)
* To fix the vulnerability update jackson-databind to version 2.16.0.



**Vulnerability for ViteJS**

There could be potential vulnerability with vite for versions like cite@5.0.5, vite@4.5.1, and vite@4.4.12. At the moment, the version of vite is 4.2.1 which is not affected by this vulnerability but there has to be made attention to it when updating the code. Vite v5 was published last year so it is a good idea at some point to consider updating it because it has more features securing the development of the code like it automatically removes depreciated APIs.

Tools used for the static testing - <https://debricked.com/>

More information about “debricked“. It is an online platform that has both free and paid versions. How it works is you connect your GitHub repository to the project you want to test for vulnerabilities. It scans the code and whatever it finds, it could be fixed with a pull request that can be made from the programs interface.

Also, code review methodology was used.

**Non-static testing**

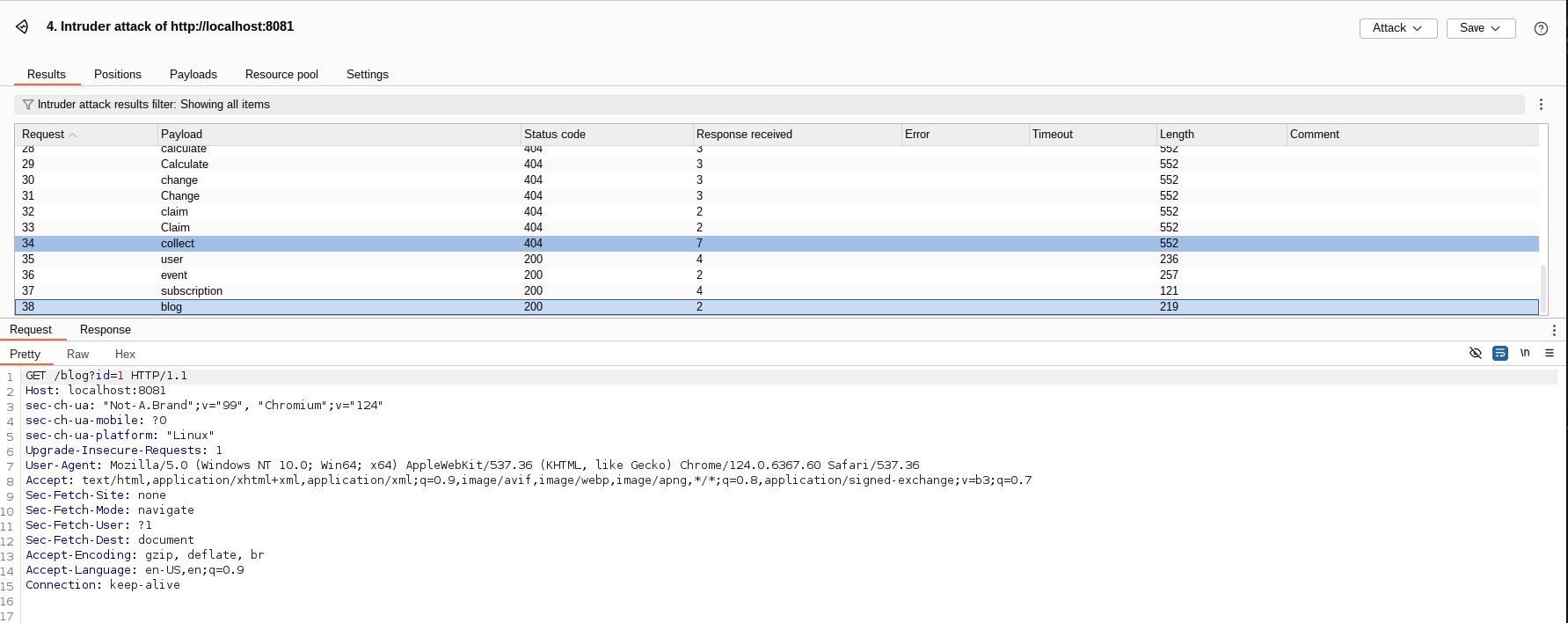
For the non-static testing Burp Suite Community Edition was used. Burp is a great tool for Penetration Testing. For the attack the “Intruder” of the Burp was used which could be created a brute force attack on the endpoints. I used an online wordlist of commonly used endpoints and was able to deduct four: blog, user, event, subscription.

Currently, for /blog there are 5 blogs found, which have an id from 0 to 4

For the /user, there are 5 users found, which have an id from 0 to 4.

For the /event, there are 5 events found, which have an id from 0 to 4.

For the /subscription there is only 1 response with id 0.



**Improvements:**

There are not any critical vulnerabilities or exploitations of the code, considering that it is not fully ready. There should be implemented error handling when the endpoint has an id that is not present in the database it should return status 404 but now it returns 200, which might DoS the server with too many requests.

Also, the technologies should be updated to the latest versions to ensure the safety of the users’ sensitive data.

There is plenty of penetration testing to be done and it would be most suitable to be done on a launched product so that the connection with the database is tested and any other security problem that may appear to be caught before it becomes a zero-day vulnerability.