OWASP – TOP 10

1. Injection

*Svako polje u modelu koje je tipa string anotira se **@SqlInjectionSafe** anotacijom koja sprečava unos izvršivih SQL naredbi (Drop table, Insert into, ...). **Hibernate**, takođe u sebi ima **ugrađene mehanizme** koji sprečavaju SQL Injection napade.*

2. Broken Authentication

*Prilikom registracije korisnika na sistem od njega se zahteva da unese lozinku koja prati prethodno utvrđenu **polisu na osnovu trenutne najbolje prakse** (minimalni broj karaktera, slovo, broj, da ne sadrži korisničko ime itd.). Nakon unosa lozinke, potrebno je ponoviti je gde se proverava da li se lozinke **podudaraju**. Svaka lozinka se "**hešira**" pomocu **BCrypt** mehanizma.*

3. Sensitive Data Exposure

***ACL-om (Access Control List)** će se onemogućiti pristup fajlovima sa osetljivim podacima.*

4. XML External Entities (XXE)

***Onemogućavanjem DTD-ova (Document Type Definition)** i eksternih entiteta se sprečava ovakva vrsta napada.*

5. Broken Access Control

***RBAC model** podrazumeva da svaki korisnik u sistemu ima svoju ulogu a svaka uloga u sistemu ima svoje permisije. Pristup svakoj metodi se obezbeđuje tako što se svaka metoda anotira sa **@PreAuthorize** anotacijom gde se proverava uloga korisnika.*

6. Security Misconfiguration

*Na mikroservisima se implementira **Spring Security**. U svakom kontroleru se implementira **mehanizam upravljanja greškama.***

7. Cross-Site Scripting XSS

*Problem XSS zaštite se delimično rešava tako što se na input poljima na frontendu postavlja **validacija kroz Regex**, a na backendu kroz **Patterne**. **Nije dozvoljen unos specijalnih karaktera** kao što su <, >, što predstavlja vid zaštite od ovakvih napada.*

8. Insecure Deserialization

*Problem nesigurnih deserijalizacija rešava se upotrebom **novijih verzija dependency-ja**.*

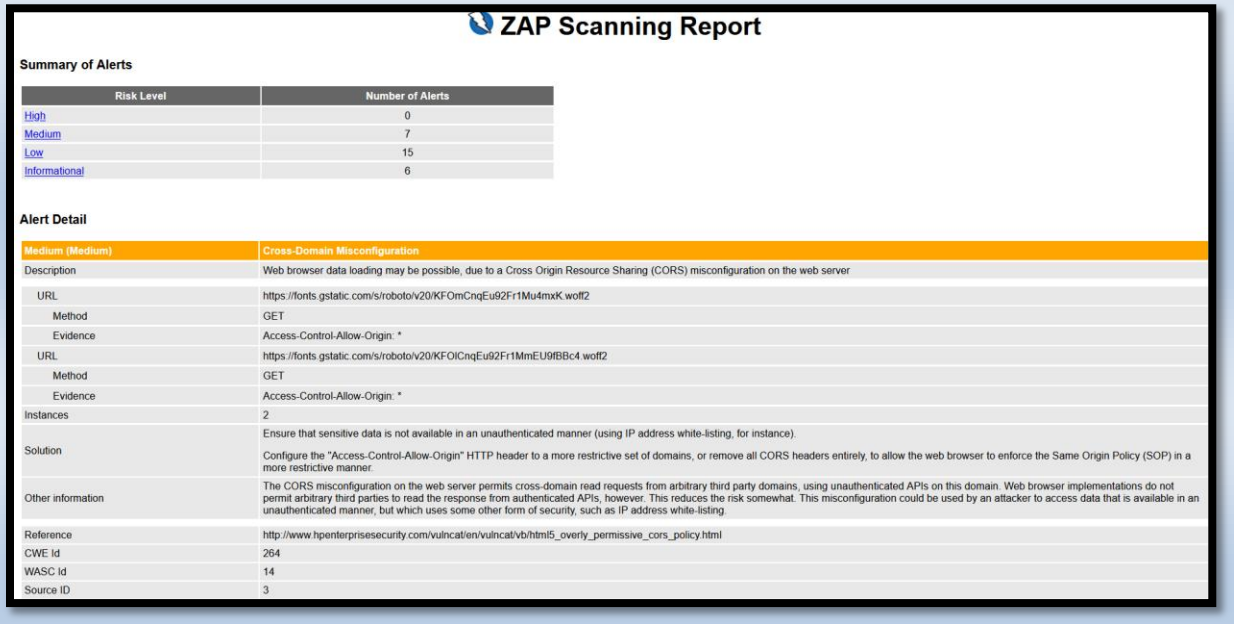
9. Using Components with Known Vulnerabilities

***Check Dependency** i razrešavanje ranjivosti.*

10. Insufficient Logging & Monitoring

***Logging mehanizam** se implementira tako da su ispunjeni zahtevi kao što su **kompletnost, pouzdanost, upotrebljivost i konciznost**. Koristi se tri tipa Log fajova: **Info, Warn I Error**.*

Skeniraćemo ranjivosi Spring Boot aplikacija uz pomoć OWASP ZAP alata koji će formirati izveštaj o svim ranjivostima koje je pronašao i rangiraće ih po kritičnosti.



ZAP Scanning Report

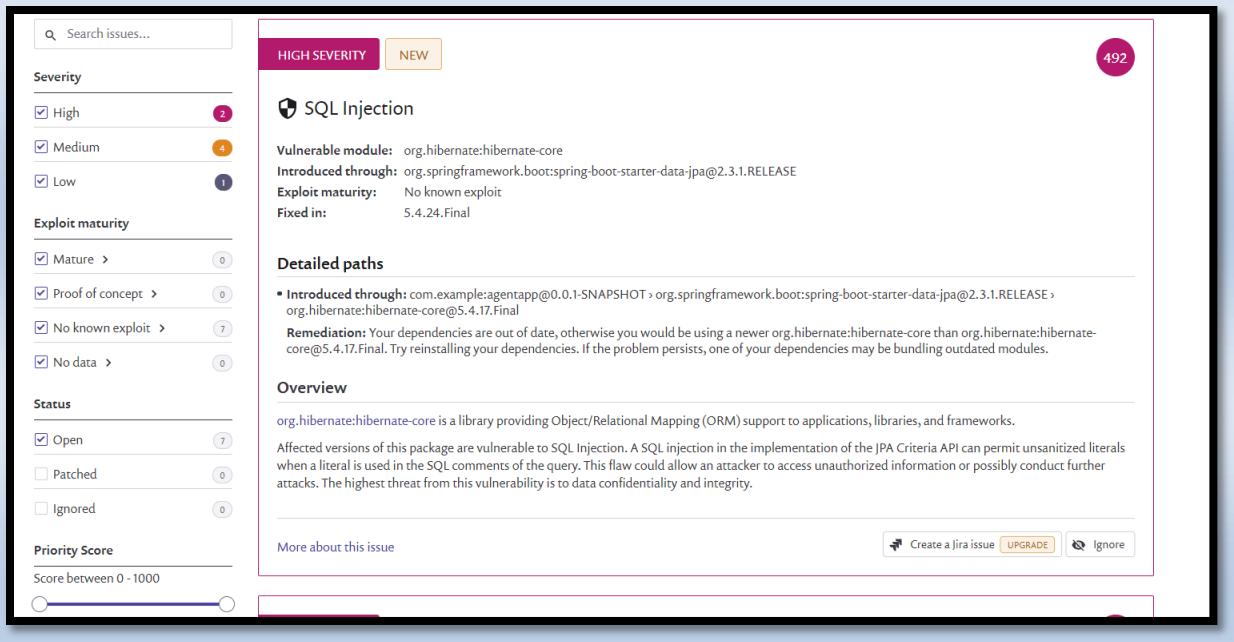
Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	7
Low	15
Informational	6

Alert Detail

Medium (Medium)	Cross-Domain Misconfiguration
Description	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server
URL	https://fonts.gstatic.com/s/roboto/v20/KFOmCnqEu92Fr1Mu4mxK.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/roboto/v20/KFOlCnqEu92Fr1MmEU9fBBc4.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
Instances	2
Solution	Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
Other information	The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.
Reference	http://www.hpenterprisesecurity.com/vulncat/en/vulncat/vb/html5_overly_permissive_cors_policy.html
CWE Id	264
WASC Id	14
Source ID	3

Takođe, sa repozitorijumom je integrisana platforma za proveravanje bezbednosti Spring Boot aplikacija, kao i React.js aplikacija - Snyk.



Search issues...

Severity

- High 2
- Medium 4
- Low 1

Exploit maturity

- Mature › 0
- Proof of concept › 0
- No known exploit › 7
- No data › 0

Status

- Open 7
- Patched 0
- Ignored 0

Priority Score

Score between 0 - 1000

HIGH SEVERITY NEW 492

SQL Injection

Vulnerable module: org.hibernate:hibernate-core
Introduced through: org.springframework.boot:spring-boot-starter-data-jpa@2.3.1.RELEASE
Exploit maturity: No known exploit
Fixed in: 5.4.24.Final

Detailed paths

- **Introduced through:** com.example:agentapp@0.0.1-SNAPSHOT › org.springframework.boot:spring-boot-starter-data-jpa@2.3.1.RELEASE › org.hibernate:hibernate-core@5.4.17.Final

  Remediation: Your dependencies are out of date, otherwise you would be using a newer org.hibernate:hibernate-core than org.hibernate:hibernate-core@5.4.17.Final. Try reinstalling your dependencies. If the problem persists, one of your dependencies may be bundling outdated modules.

Overview

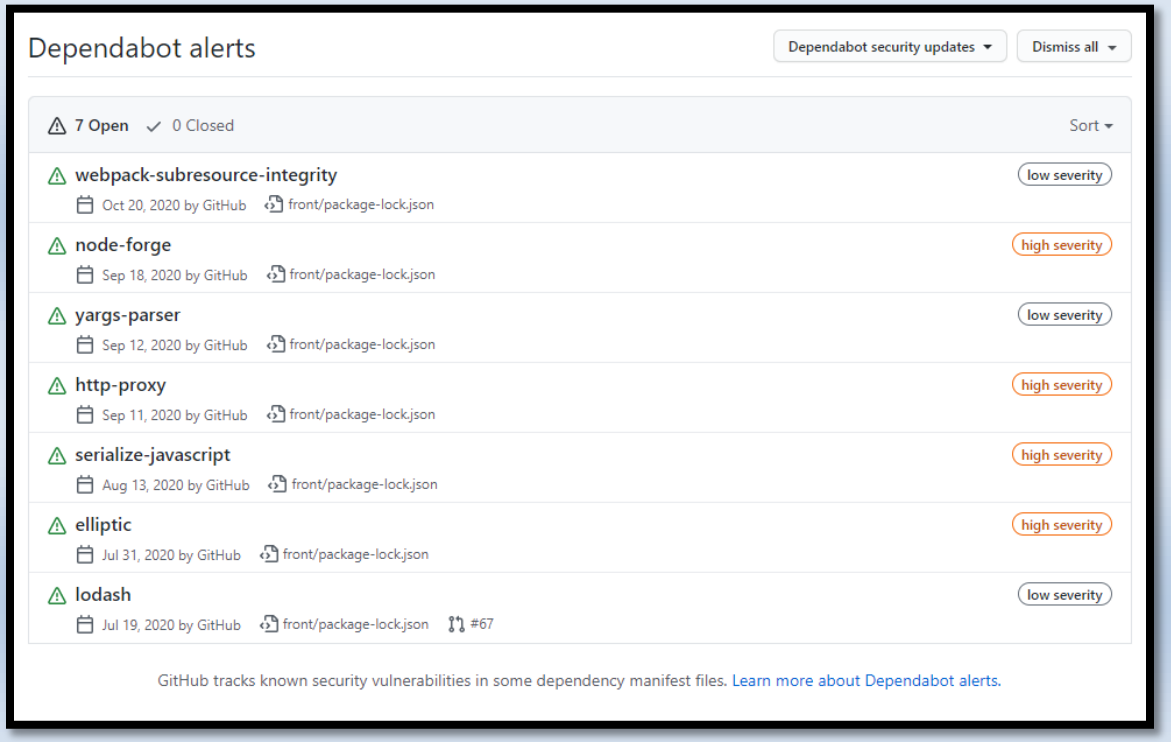
org.hibernate:hibernate-core is a library providing Object/Relational Mapping (ORM) support to applications, libraries, and frameworks.

Affected versions of this package are vulnerable to SQL Injection. A SQL injection in the implementation of the JPA Criteria API can permit unsanitized literals when a literal is used in the SQL comments of the query. This flaw could allow an attacker to access unauthorized information or possibly conduct further attacks. The highest threat from this vulnerability is to data confidentiality and integrity.

More about this issue

Create a Jira issue UPGRADE Ignore

Ujedno, aktiviran je Dependabot na Githubu koji takođe proverava ranjivosti projekta tokom celog njegovog životnog veka.



Opšte poznato, NPM ima ugrađenu podršku za skeniranje ranjivosti paketa za radno okruženje Node.js.