OWASP TOP 10 GAP ANALIZA

1. Injection

Injection je problem koji se javlja u situacijama kada korisnik unese zlonameran sadržaj u aplikaciju sa namerom da je eksploatiše. Napadač pokušava da podmetne kod koji može izazvati narušavanje poverljivosti, dostupnosti i integriteta aplikacije. Ukoliko odbrana od ovakvog vida napada ne bude implementirana na adekvatan način kod koji napadač unese će biti interpretiran na drugačiji način od onog koji je predviđen implementiranjem aplikacije.

Relevantna ranjivost za naš sistem je SQL injection.

SQL injection rešavamo upotrebom JPA Repository-a koji koji štiti aplikaciju od SQL Injection napada.

[Link](https://stackoverflow.com/questions/33772332/prevent-sql-injection-java)

1. Broken autentification and session management

Napadi ove vrste su fokusirani na pristupanje nalogu redovnog korisnika sistema. Napadač pokušava da pristupi nalogu i privilegijama koje su predviđene za rolu korisnika vlasnika naloga, tako što pokuša da na određeni način sazna kredencijale korisnika ili tako što pristupi sesiji ulogovanog korisnika.

Ranjivost za ovu tačku OWASP top 10 liste jeste gubitak poverljivosti, integriteta, pa i dostupnosti korisničkih podataka, takođe lako može doći do krađe računa, ugrožavanje privatnosti.

Gore navedene opasnosti smo rešili na više načina. Omogućili smo odjavu sa sesije, postavljanje samo kompleksne šifre, usporavanje procesa logovanja, radi smanjenja mogućnosti za brute force napad. I na kraju krajeva smo zaštitili lozinku korisnika koja se čuva u bazi podataka tako heširanjem po [Bcrypt algoritmu](https://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html). Postoji dosta načina zaštite od napada vezanih za ovu tačku, a to su: sigurnosno pitanje, izmena računa, obaveštavanje korisnika na mejl prilikom logovanja...

1. Cross-site scripting (XSS)

XSS je česta ranjivost sistema kod koje se nedovoljno vodi računa o proveri podataka koje unosi korisnik. Ukoliko ti podaci stignu do web čitača bez prethodne provere, lako se web čitaču može proslediti skripta koja će se izvršavati paralelno sa kodom kreiranim od strane programera. Na taj način lako može doći do ubacivanja nepoželjnog sadržaja. U najvećem broju slučajeva podmeće se JavaScript kod.

Moguće je ukrasti korisničke podatke, podatke o sesiji korisnika, izmeniti sadržaj web stranice, uticati na pretraživač korisnika.

Zaštitili smo se od XSS-a korišćenjem adekvatne verzije [AngularJS-a](https://docs.angularjs.org/guide/security), na adekvatan način.

1. Broken Access Control

Svodi se na restrikcije ulogovanih korisnika nad funkcionalnostima aplikacije. Ukoliko su neke privilegije suvisne za neke korisnike ili ne postoji odgovarajuca kontrola pristupa ili je lose implementirana, moguce je od strane hakera izvrsiti eksploataciju osetljivih podataka drugih korisnika.

Regulacija ranjivosti podrazumeva implementaciju Role based access control modela. Putem ovog mehanizma se za učesnike u sistemu vezuju odgovarajuće privilegije i pri tom vrši restrikcija odgovornosti korisnika sistema nad aplikacijom.

Linkovi: <http://www.stevenlanders.com/2014/07/02/spring-aop-annotation.html>

<https://www.tutorialspoint.com/springaop/springaop_custom_annotation.htm>

<http://www.baeldung.com/role-and-privilege-for-spring-security-registration>

1. Security Misconfiguration

Ukoliko je neka komponenta zasnovana na nesigurnoj konfiguraciji, ona može biti izložena napadu, samim tim spada u 5. tačku OWASP Top 10 liste. Bez obzira da li se komponenta nalazi na serveru, u bazi podataka ili nekom trećem mestu, ukoliko je zasnovana na nesigurnoj konfiguraciji, ona spada pod ovu kategoriju.

Ukoliko postoji klasičan admin nalog, napadač bi jednostavnom pretragom na google-u mogao doći do kredencijala takvog naloga. Sam framework može biti ranjiv. Komponente koje nisu od koristi mogu biti meta napada takođe.

Načini za regulaciju ranjivosti podrazumevaju brisanje:

* Brisanje klasičnog admin naloga.
* Pre same implementacije istražiti najsigurniji framework trenutno.
* Proweriti da li su obrisane sve nekorišćene komponente poput naloga, privilegija, portova..
* Automatizovati sve što se može automatizovati jer su ljudi češće podložni greškama.
* Redovno skeniranje sistema radi pronalaženja loše konfigurisanih fajlova.

1. Sensitive Data Exposure

Odnosi se na zaštitu osetljivih podataka koji, ukoliko budu eksploatisani, mogu ozbiljno narušiti funkcionisanje i pouzdanost sistema.

Podaci od važnosti koji se čuvaju u tekstualnom formatu kao što su šifre, podaci o računima u banci i sl. mogu biti podložni ranjivosti.

Regulacija ranjivosti se vrši hešovanjem osetljivog sadržaja u bazi podataka. Prilikom transfera podataka nikad ne slati sadržaj u otvorenom formatu. Koristiti heš funkcije za koje se smatra da su najbolje u praksi.

Linkovi: <http://www.stevenlanders.com/2014/07/02/spring-aop-annotation.html>

1. Insufficient Attack Protection

Nedovoljna zaštita od napada se odnosi na nemogućnost samog sistema da detektuje, spreči I odgovori na različite vrste napada. Ideja koja stoji iza A7 jeste da svaki profesionalac koji se bavi sigurnošću treba da prevede svaku sigurnosnu arhitekturu i svaki model pretnji u kod koji ce se moći pokretati tokom razvoja aplikacije, kako bi se aplikacija testirala koliko je podložna napadima.

1. Cross-Site Request Forgery(CSRF)

CSRF napadom se prisiljava web čitač prijavljene žrtve na slanje zahteva ranjivoj web aplikaciji, koja zatim obavlja odabranu akciju u žrtvino ime, u korist napadača. Ukoliko se aplikacije oslanjaju isključivo na korisničke podatke ili oznake koje se automatski šalju od strane web čitača, mogu lako biti napadnute.

Ranjivosti iz XSS odnose se i na CSRF. Podmetanje HTTP zahteva u HTTP objekte (img, button često).

Transakcije se potpisuju i na taj način osigurava izvornost zahteva. Kod procesiranja osetljivih podataka (prilikom logovanja npr.) koristimo POST metodu koju je teže falsifikovati od GET. Takođe koristimo Spring Security CSRF Protection koji uključuje skrivene tokene.

Linkovi: <https://spring.io/blog/2015/01/12/the-login-page-angular-js-and-spring-security-part-ii>

<https://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html>

1. Using Components with Known Vulnerabilities

Komponente sa propustima mogu biti pronađene i napadnute od strane automatizovanih alata napadača. Te komponente su najčešće operativni sistemi, web servisi, biblioteke (plugin-ovi),... Napadač koristi činjenicu da je komponenta ranjiva i iskorišćena ili povezana sa aplikacijom. U sprečavanju ovog napada potrebno je pronaći sve ranjive komponente koje se koriste i zaštiti ih od mogućih napada.

Relevantna ranjivost za nas je korišćenje biblioteka koje su ranjive ili pozivaju druge ranjive biblioteke, kao i korišćenje verzije biblioteke koja nije najnovija.

Ranjivosti regulišemo proverama ranjivosti korišćenih biblioteka i praćenjem njihovih verzija. Većim delom koristimo Spring Security biblioteke. Najnovije biblioteke pratimo na Maven repozitorijumu.

Link: <https://www.owasp.org/index.php/OWASP_Dependency_Check#tab=Main>

1. Underprotected APIs

Kao i na serverskoj strani, tako treba i na klijentskoj voditi računa o sigurnosti i ranjivosti biblioteka koje se koriste. API-ji koriste protokole i formate podataka kao što su SOAP/XML, REST/JSON, RPC, GWT,... Ovi API-ji su često nezaštićeni i sadrže brojne ranjivosti.

Obzirom da podatke razmenjujemo putem SOAP-a (XML) i REST-a (JSON) ovo su relevantne ranjivosti za nas.

Poruke koje se razmenjuju putem SOAP-a su šifrovane i potpisane tako da napadač ne može da ih pročita ili preusmeri. Sistem sa API-jima komunicija preko https. Na API-ju je uključena mera autorizacije putem permisija i autentifikacije.