# Gap analiza između implementiranog sistema i OWASP Top 10 rizika

## A1: Injection

- **Opis:**

  Injection napad je moguće realizovati kada podaci koje korisnik šalje nisu validirani, provereni ili filtrirani od strane aplikacije. Konkretno, injection napad predstavlja direktno slanje malicioznih upita ili komandi interpreteru, što prouzrokuje nepredviđen rad aplikacije kao sto je čitanje, izmena i brisanje iz baze podataka za koje nije autorizovan. Postoji više vrsta Injection napada: SQL , NoSQL, OS, LOG, LDAP, EL i OGNL Injection.

  Primer: Napadač direktno prosleđuje SQL upit preko ulaznih parametara aplikacije,

  ```
  String query = "SELECT * FROM  user WHERE userId='" +
  request.getParameter("id") + "'";

  http://example.com/app/accountView?id=' or '1'='1
  ```

- **Pristup problemu:**

  Zaštitna mera od injection napada je razdvajanje podataka od upita i komandi.Potrebno je koristiti API koji koristi parametrizovan interpreter ili ne koristi interpreter. Upotreba "White list" koji predstavlja validaciju ulaznih parametara. Korišćenje LIMIT i ostalih SQL naredbi unutar upita kako bi sprečili masovno otkrivanje podataka u slučaju SQL injection.

- **Implementirano rešenje:**

  Rešenje je otporno na injection napad jer korisnik nije ni u jednom slučaju u direktnoj vezi sa interpreterom.
  Kod komunikacije sa bazom (SQL Injection) korisnikov sadržaj poruke na ulazu biva iskontrolisan Hibernate-om.
  Kod XXE ranjivosti JAX-B void računa o ovom problemu.

## A2: Broken Authentication

- **Opis:**

  Provera i potvrđivanje identifikacije korisnika, autentifikacija i upravljanje sesijama predstavlja ključnu odbranu od napadača i napada oslonjenih na autentifikaciju. Slabosti koje se mogu pronaći za realizaciju ovog napada su dozvoljeni brute force ili neki drugi automatski napadi. Dozvoljena

podrazumevana, slaba, česta lozinka ili lozinka koja se formira na osnovu poznatih algoritama. Neimplementirana višefaktorska autentifikacija itd.

- **Pristup problemu:**

  Implementacija višefaktorske autentifikacije kako bi se zaustavio automatizovan pristup kao i brute force napadi.

  Uklanjanje i zabrana podrazumevanih kredencijala, primena paterna za formiranje lozinki svih korisnika aplikacije, ograničavanje neuspešnih pokušaja logovanja.

- **Implementirano rešenje:**

  Jaka korisnička lozinka. Lozinka sadrži minimum 10 karaktera od čega mora sadržati bar jedan broj i veliko slovo.
  Zaštita korisničkih kredencijala upotrebom hash i salt mehanizma. U ovo svrhu korišćen BCryptPasswordEncoder sa $2^{10}$ iteracija. Koristi se 16 bitni salt.
  Osetljivi URL-ovi poput promene lozinke i aktivacionog email-a (gde je potrebno identifikovati korisnika) su šifrovani kako ne bi svako mogao da izmeni stanje sistema već samo onaj kome je dostavljeno.
  Konfigurisan https koji obezbeđuje zaštićen transport osetljivih korisničkih podataka.

## A3: Sensitive Data Exposuer

- **Opis:**

  Prikazivanje osetljivih podataka, slanje i njihovo čuvanje predstavlja opasnost za aplikaciju. U kategoriju osteljivih podataka svstavamo podatke poput broja kreditne kartice, lozinke, ličnih i zdravstvanih informacija. Napadač presecanjem komunikacije može doći do osetljivih podataka u slučaju da podaci nisu šifrovani ili komunikacija nije zaštićena.
  Primer: Aplikacija enkriptuje brojeve kreditnih kartica upotrebom automatskog algoritma baze podataka. Međutim ovi podaci je autmatski dešifruju prilikom preuzimanja. Ako je moguće realizovati SQL injection napad podaci mogu biti kompromitovani.

- **Pristup problemu:**

  Klasifikacija podataka čuvanih u okviru baze i prepoznavanje osetljivih podataka. Izbegavanje čuvanja osetljivih podataka u slučaju da to nije neophodno, enkriptovanje svih osetljivih podataka koji se čuvaju u bazi validinim metodama enkripcije.

- **Implementirano rešenje:**

  Lozinke su heširane i kao takve sačuvane u bazi.
  Komunikacija se šifrovana korišćenjem https protokola.
  Podaci su čuvani samo onda kada je to zaista i potrebno.

## A4: XML External Entities(XXE)

- **Opis:**

  XML External Entities je vrsta napada na aplikaciju koja parsira ulazni XML fajl. Do napada dolazi kada ulazni XML je prihvaćen direktno ili poslat od starne nepouzdanog, malicioznog izvora obrađen sa slabim XML parserom.

  Napadač izvršava denial-of-service napad uključujući beskonačni fajl.

  ```
  <!ENTITY xxe SYSTEM "file:///dev/random" >]>
  ```

- **Pristup problemu:**

  Obuka programera je od esencijalnog značaja za identifikovanje i zaustavljanje XXE napada. Upotreba jednostavnijih formata poput JSON i izbegavanje serijalizacije osetljivih podataka. Provera XML ili XSL poslatog fajla, validacija XML upotrebom XSD validacije. Ažuriranje biblioteka programa za obradu XML-a. Ažuriranje SOAP na verziju 1.2 ili napredniju.

- **Implementirano rešenje:**

  Konfigurisanje JAX-B da onemogući procesiranje eksternih entiteta. Koristili smo verzija 4.3.12 Springa gdje je po defaultu processExternalEntities false.

## A5: Broken Access Control

- **Opis:**

  Kontrola pristupa služi za zabranu pristupa određenim delovima i mogućnostima aplikacije u skladu sa dozvolama koje određeni korisnik poseduje. Nedostatak kontrole pristupa dovodi do neovlašćenog pristupa informacijama, čitanja, izmene ili brisanja podataka. Do toga može doći prevazilaženjem kontrola pristupa izmenom URL-a, internog stanja aplikacije ili HTML stranice. Dozvolom promene primarnog ključa na drugog korisnika, dozvola izmene tuđeg naloga. Primer: Izmena URL-a i pristup tuđem nalogu

  ```
  http://example.com/app/accountInfo?acct=notmyacct
  ```

- **Pristup problemu:**

  JWT tokeni treba da budu poništeni nako što se korisnik izloguje. Proslediti upozorenje administratoru nakon neuspešne kontrole pristupa. Implementacija mehanizama kontrole pristupa i njihova ponovna upotreba kroz aplikaciju.

- **Implementirano rešenje:**

  Implementiran RBAC sistem za kontrolu pristupa. Kontrola pristupa na osnovu URL-a proverava u bazi da li korisnik ima pravo za pristup.

## A6: Security Misconfiguration

- **Opis:**

  Nedostatak i propust u bezbednosnoj konfiguraciji mogu da se prikažu na različitim nivoima aplikacije kao sto je u bazi, serveru itd. Zbog toga je od ključnog značaja u napred definisana i implementirana a potom i redovno ažurirana bezbednosna konfiguracija . Do ove vrste napada može doći u slučaju instaliranih ili dozvoljenih nepotrebnih komponenti i funkcija. Poruka o greškama sadrži previše informacija.

- **Pristup problemu:**

  Aplikacija treba da se modeluje tako da pruži efikasnu separaciju među komponentama kao i njihovo međusobno funkcionisanje. Potrebna je minimalna platforma bez nepotrebnih funkcija, komponenti dokumentacije i primera. Ukloniti svaku komponentu i framework koji se ne koristi. Implementacija procesa koji poroverava da li je konfiguracija važeća i bezbedna.

- **Implementirano rešenje:**

  Upotreba najnovih verzija operativnog sistema i najnovija verzija ssl (TSL). Ne postoji predefinisani naloga sa visokim privilegijama (admin/admin). Rukovanjem greškama ne otkrivamo previše stvari korisniku (npr. kod logovanja pri pogrešnom unosu ne prikazujemo gde je tačno greska, već se pruža mogućnost za novim pokušajem (poklapanje nije uspelo) ).

## A7: Cross-Site Scripting(XSS)

- **Opis:**

  Cross-Site Scripting(XSS) napad se izvršava kada aplikacija koristi podatke unesene od strane korisnika u nekom prevodiocu pretraživača kao što su JavaScript ili Flash. Napadač može da unese script na osnovu kog može izmeniti obrisati ili pristupiti podacima za koje nije autorizovan. Postoje tri vrste XSS napada, Stored XSS, Reflected XSS i DOM XSS.

- **Pristup problemu:**

  Izbegavanje neproverenog HTTP request zasnovanog na kontekstu izlaza HTML(body, attribute, JavaScript, CSS, or URL). Implementirati validacioni sistem za zabranu unosa određenih specijalnih karaktera.

- **Implementirano rešenje:**

  XSRF token se smešta na frontend i sa svakim requestom šalje na bekend. Konfigurisanjem spring security-a csrf je omogućen i framework vodi računa ovoj vrsti napada. Angular vrši sanitaciju vrednosti, koda koji se ubacuje u DOM.

## A8: Insecure Deserialization

- **Opis:**
  Insecure Deserialization je ranjivost koja se javlja kada se neprovereni, maliciozni podaci koriste za zloupotrebu logike aplikacije, za prouzrokovanje denial of service (DoS) napada. Ovo može takođe da egzistira tipičnim access control napadima.
- **Pristup problemu:**
  Sprovođenje provere integriteta upotrebom digitalnih potpisa. Pokretanje koda koji se deserijalizuje u okruženjima sa malim privilegijama.
- **Implementirano rešenje:**
  Jedini slučaj u našem sistemu deserijalizacije objekta je u soap xml poruke. Rješenje je implemantirnao tako da prihvatamo samo serijalizovane objekte od pouzdanih izvora. Vršili smo proveru integriteta na osnovu digitalnog potpisa nad (datim serijalizovanim objektima) ,tako što smo potpisivali objekte koje šaljemo i pri prijemu poruke vršili smo validaciju tog potpisa.

## A9: Using Components with Known Vulnerabilities

- **Opis:**
  Aplikacije koriste razne biblioteke i framewok-ove da bi izvšavale sve funkcije uspešno. Upotreba raznih komponenti i biblioteka donosi ranjivost i rizik zbog neproverenosti navedenih materijala i mogućnosti da imaju neku slabost koju napadač može da iskoristi za backdoor pristup osetljivim podacima.
- **Pristup problemu:**
  Pre upotrebe nove biblioteke potrebno je proveriti njenu sigurnost i pouzdanost, ako to nije moguće najbolje je izbeći upotrebu te biblioteke. Preuzimanje komponenti isključivo sa zvaničnih i proverenih izvora. Uklanjanje nekorišćenih biblioteka i komponenti.

- **Implementirano rešenje:**
  JWT biblioteka sadrži implementiranu dodatnu zaštitu obraćajući pažnju na poznate ranjivosti (NONE hashing algorithm & token sidejacking).

  **NONE hashing algorithm**
  Problem nastaje kada napadač preuzme token i promeni mu heš algoritam na “none”. Neke biblioteke ovakav token tretiraju kao validan pa je tako napadaču omogućen pristup.
  Naše rešenje je otporno na ovu ranjivost proverom potpisa.
  Ne postoji mogucnost za probijanje ovog tipa jer se svakom tokenu proverava potpis i ukoliko ne postoji ili je neispravan ne propušta se dalje u funkcionalnosti (dobijamo exception “UnsupportedJwtException”)

**Token sidejacking**

Problem nastaje kada napadač ukrade token i koristi ga kako bi dobio dozvolu za određene funkcionalnosti umesto korisnika.

Rešenje se štiti od ovog napada tako što ubacuje korisnički kontekst u token. Koraci ove zaštite su:

1. Random string se generiše prilikom autentifikacije i uključuje se u token.
2. Klijentu se string šalje kao cookie.
3. Heš stringa se čuva u tokenu kako bi se zaštitili od krađe stringa iz tokena i setovanja cookie-a istom vrednošću.
4. Tokom token validacije, ako prihvaćeni token ne sadrži dobar kontekst dolazi do odbijanja.

# A10: Insufficient Logging&Monitoring

- **Opis:**

  Nedostatak evidencije i praćenja aktivnih akcija se dešava u svakom trenutku. To napadaču pruža veću šansu za pristup i uništavanje aplikacije. Nedostatak praćenja akcija prijava, zahteva za velikim transakcijama. Nejasne i neadekvatne poruke greške ili upozorenja. Nemogućnost aplikacije da u realnom vremenu otkrije napadača.

- **Pristup problemu:**

  Omogućiti da se logovi generišu u formatu koje se lako može koristiti prilikom upotrebe softvera koji se bave analizom logova. Omogućiti da transakcije velikih vrednosti imaju naznačen trag sa kontrolom integriteta radi sprečavanja neautorizovanog brisanja i izmene podataka.

- **Implementirano rešenje:**

  Napravljena je klasa logger koja je zadužena za upisivanje u fajl kada se dogodi određena akcija,postoje tri metode koje rade upisivanje u info fajl i upisivanje u fajl ako dođe do greške, na taj način smo olakšali analiziranje logova, i osigurali neporecivost kao bitnu osobinu bezbednosne kontrole.