OWASP TOP 10

IzveŠtaj

**Autori**: *Tim 20*

* *RA 99/2016 Igor Gligorević*
* *RA 221/2016 Nenead Marin*
* *RA 223/2016 Kristina Radojević*
* *RA 215/2015 Branislav Letić*

**Korišćene tehnologije:**

* *BackEnd: Sprig Boot, NodeJS(Express)*
* *FrontEnd: ReactJS*
* *Database: SQL, NoSQL(Mongo)*

**Aplikacija:** *Rent a car*

# 1. Injection

Injection napadi predstavljaju jedne od najopasnijih napada koji mogu da ugroze sigurnost aplikacije. U celoj aplikaciji smo izbegavali kreiranje upita konkatenacijom stringova sa korisničkim unosima i time zaštitili aplikaciju od mogućeg eksploatisanja ovim tipom napada. Upiti u agentskoj aplikaciji i u svim mikroservisima su formirani pomoću *prepared statement*-a ili ugradjene procedure (*stored procedures*). Pored toga, oslanjajući se na već proverene alate ORM(Hibernate) i ODM(Mongoose) prevazišli smo ranjivosti injection napada. Ovakvim rešenjem postigli smo da upiti ka bazi budu sigurni i da korisnik svojim unosom ne može da izmeni suštinu konkretnog upita. Klijentska i serverska strana imaju ugrađenu validaciju korisničkog unosa, pa se korisnik ograničava na unos u traženom formatu što utiče na povećanje sigurnosti i doprinosi prevenciji od injection napada.

# 2. Broken Authentication

Aplikacija poštuje najbolje prakse vezane za prevenciju od ovog tipa ranjivosti. Eksploatisanje ove ranjivosti napadač može izvršiti na više načina. Pre čuvanja, lozinka se hešira, prilikom heširanja koristi se nasumično generisani string koji se naziva salt i on se pripaja uz generisani heš. Time se sprečava mogućnost da, ukoliko dođe do kompromitovanja osetljivih podataka iz skladišta, se ti podaci mogu zloupotrebiti.

Kada korisnik kreira lozinku od njega se zahteva da ona sadrži minimalno 10 karaktera u okviru kojih se nalaze minimalno po jedan specijalan karakter, broj, veliko i malo slovo. Na ovaj način se smanjuje mogućnost da se uspešno izvrši “Brute force” napad i pogodi korisnikova lozinka. Ograničenjem broja neuspešnih logovanja i blokiranjem korisnika kada dosegne njihov limit na određeno vreme dodatno se osigurava od “Brute force” napada.

Prilikom odjave token, koji je služio da se korisnik identifikuje u toku obrade zahteva koje je poslao serveru, se briše.

# 3. Sensitive Data Exposure

Zaštita od izloženosti osetljivih podataka obezbeđena je kroz više aspekata aplikacije. Karakteristične ranjivosti i izvori problema izloženosti osetljivih podataka i načini sa kojima smo te ranjivosti i probleme prevazišli u aplikaciji su:

* **Man in the middle attack**: Razmena poruka između klijentske i serverske strane zasnovana je na HTTPS komunikaciji. Time se garnatuje poverljivost, integritet, autentičnost i neporecivost. Ukoliko postoji napadač koji osluškuje mrežu preko koje se odvija komunikacija, poruke koje može da vidi su zaštićene jer je sav saobraćaj enkriptovan. Pre uspostave HTTPS komunikacije prilikom razmene sertifikata ukoliko napadač pokuša da izmeni sertifikat(npr. ubaci svoj javni ključ umesto onog koji je na sertifikatu), proverom digitalnog potipsa bi se ustanovilo da sertifikat nije validan, jer bi se novoizračunati hesh razlikovao. Na ovaj način se štite osetljivi podaci u tranzitu.
* **Čuvanje osetljivih podataka u vidu običnog teksta:** Ova ranjivost je prevaziđena tako što se osetljivi podaci zaštićeno čuvaju. Jedan od osetljivih podataka je lozinka korisnika, koja se pre skladištenja u bazu hešira. Kao dodatna mera zaštite pri heširanju koristi se i salt, koji se automatski generiše i konkatenira na heširan deo. Kako bi logovi aplikacije bili upotrebljivi, čuva se informacija o izvršiocu neke akcije, ukoliko je korisnik ulogovan. Kako bi se zaštitio osetljiv podatak po kojem se može prepoznati izvršilac, on se takođe hešira na isti način kao i lozinka korisnika.
* **Korišćenje slabih kriptografskih algoritama:** Aplikacija se štiti od ove ranjivosti, tako što poštuje najbolje prakse koje se trenutno koriste a tiču se kriptografskih algoritama. Tako se za generisanje ključeva sertifikata koristi ECC (Elliptic Curve Cryptography). Praktikovano je sigurno čuvanje sertifikata i ključeva, kao i ograničavanje njihovom pristupu preko ACL lista.

# 4. XML External Entities (XXE)

XML se u našoj aplikaciji koristi u kontekstu SOAP protokola, što je čini ranjivom na XXE napade. Jedan od najčešćih primera ovakve vrste napada predstavlja *Billion Laughs Attack,* vrsta DoS napada, koji ima zadatak da preoptereti memoriju XML parsera i time naruši dostupnost servera. Način odbrane od ovakvog napada predstavlja zabrana korišćenja DTD-ova u XML parserima, ili ako to nije moguće, onemogućiti eksterne entitete i deklaracije tipa dokumenta.

# 5. Broken Access Control

Implementacija RBAC modela koja se prožima kroz aplikaciju i time štiti metode od neautorizovanog pristupa. RBAC model je realizovan kroz skup rola koje su vezane za permisije. Ispunjen je princip najmanje privilegije, tako što neki korisnik može da koristi samo potrebne funkcionalnosti, bez mogućnosti da poziva autorizovane metode namenjene isključivo drugim tipovima korisnika. Prilikom registracije, korisniku se dodeljuje odgovarajuća rola. Po pozivu metoda koje imaju autorizovani pristup izvršava se provera da li je rola korisnika vezana za određenu permisiju i da li je korisnik blokiran za tu permisiju. Prilikom kreiranja zahteva ka serveru, server generiše access token koji se dalje prosleđuje mikroservisima, a u koji su upisane sve permisije korisnika.

Zaštita resursa od neautorizovanog pristupa izvršena je upotrebom ACL liste. Njom se omogućava da se se pojedinačnim korisnicima dodaju ili oduzmu određene permisije za rad sa resursima. Štiti fajlove koji sadrže osetljive podatke.

# 6. Security Misconfiguration

Spring boot tehnologija zajedno sa Spring security-em pruža lak način da se konfigurišu podešavanja koja se tiču bezbednosnog aspekta aplikcije. Problem loše konfiguracije nije izazvan, jer nisu korišćene nepotrebne sigurnosne konfiguracije, korisnicima se ne vraća “stack trace” grešaka i konfiguracija je pažljivo kreirana tako da obskrbi aplikaciju sa potrebnim bezbednosnim karakteristikama.

# 7. Cross-Site Scripting (XSS)

Kad je reč o XSS-u, aplikacija preventira sve tipove XSS napada. Prvenstveno zato što aplikacija na front-end strani nigde ne izvrsava bilo kakav korisnički unos kao html. Samim tim se ne može izvršiti bilo kakva javascript skripta ili maliciozan kod. Pored navedenog razloga koji preventira sve tipove XSS napada, dodatno je povećana sigurnost i pospešena prevencija od:

* **Stored XSS**: Validacijom korisničkog unosa, korisnik se obavezuje da unese podatke u traženom formatu. Na osnovu definisanih formata korisničkih unosa pre skladištenja podataka vrši se provera da li konkretan unos zadovoljava traženi fomrat. Time se uklanja mogućnost za skladištenjem bilo kakvog teksta koji u sebi sadrži maliciozan kod.
* **Reflected XSS**: Ukoliko bi se ukazala potreba da se korisnički unos prikazuje na stranici u vidu html-a, kada bi parametri bili prosleđivani kao “query” parametri, uvela bi se upotreba alata koji bi vršio sanitizaciju tog unosa. Oslanjanje na alat ili third-party biblioteku nekad nije dobra praksa, međutim pravljenje svog alata je podložno propustima i greškama, pa je prihvatljivo osloniti se na već postojeću biblioteku koja pruža potrebne funkcionalnosti. Jedna od takvih biblioteka koja služi za sanitizaciju korisničkog unosa je DOMPurify, koja vrši filtriranje teksta izbacivajući sve html tagove koji mogu izazvati maliciozno ponašanje. U slučaju ovog tipa xss napada sanitizaciju korisničkog unosa je moguće vršiti na serverskoj i/ili na klijentskoj strani.
* **DOM Based XSS**: Iako se ovaj napad može izvršiti na sličan način kao i Reflected XSS, prosleđivanjem “query” parametra koji sadrži u sebi maliciozan kod koji se izvršava u pretraživaču žrtve, razlikuje se po tome što se prevencija može izvršiti samo na klijentskoj strani. Pa je prevenciju ovog tipa xss napada moguće izvršiti upotrebom alata pomenutog u pasusu iznad.

# 8. Insecure Deserialization

Deserijalizacija predstavlja konvertovanje podataka iz različitih formata (JSON, XML...) u Java objekte, pri čemu napadač može da unese maliciozan kod koji će se prilikom konvertovanja izvršiti. U našoj aplikaciji nema slučajeva deserijalizacije, što znači da ova grupa napada nije relevantna u tom kontekstu. U slučaju da postoji potreba sprečavanja ovakvog napada jedini pouzdan način bio bi ne prihvatanje objekata od nepouzdanih izvora ili korišćenje medijuma koji prihvataju samo primitivne tipove podataka. Isto tako, omogućiti praćenje deserijalizacije podataka.

# 9. Using Components With Known Vulnerabilities

Koristeći alat “OWASP dependency check” i “snyk” za pronalaženje ranjivosti bilblioteka i alata na koje se oslanja aplikacija, pronašli smo sledeće ranjivosti i rešenja kojima se otklanjaju:

* **Spring Security(**[spring-security-core-5.3.2.RELEASE.jar](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Agent%20Application%20Dependency%20Check.html#l54_83fed9c3ee4a014b6a6cacfc7042eb325dca6766), [spring-security-rsa-1.0.9.RELEASE.jar](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Auth%20Service%20Dependency%20Check.html#l90_7a506772756fe7669fa9538e7f195f832fab8559), [spring-security-crypto-5.2.4.RELEASE.jar](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Eureka%20Dependency%20Check.html#l79_524e0f34f07c502b1806da73180de966f9d228d8)**)**:

Problem: Postoji propust u authorizaciji kada se koristi “method security”. Tada neautorizovani maliciozni korisnik može dobiti neautorizovani pristup metodi koja ima ograničen pristup.

Rešenje: Ažuriranjem verzije spring security-a prevazilazi se ova ranjivost.

* **Logging (**[log4j-api-2.12.1.jar](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Auth%20Service%20Dependency%20Check.html#l62_a55e6d987f50a515c9260b0451b4fa217dc539cb)**)**:

Problem: Neispravna validacija sertifikata sa neusklađenim host-om u Apache Log4j SMTP(Simple Mail Transfer Protocol) appender-u može da izazove presretanje komunikacije i izvršavanje napada MITM(Man in the middle) i time curenje bilo kog loga koji je poslat kroz taj appender.

Rešenje: Upotrebom novije verzije (*2.13.2*) prevazilazi se ova ranjivost.

* **Tomcat (**[tomcat-embed-core-9.0.34.jar](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Auth%20Service%20Dependency%20Check.html#l62_a55e6d987f50a515c9260b0451b4fa217dc539cb)**)**:

Problem: Kada je reč o tomcatu za određene verzije, uključujući i onu koju smo mi prvobitno koristili, napadaču je omogućeno da dobije kontrolu nad sadržajem i nazivom fajla koji se nalazi na serveru.

Rešenje: Prelaskom na noviju verziju(>= 9.0.35) ova ranjivost se ne može više ekspolatisati.

* **Eureka – Service Registry (**[spring-cloud-netflix-eureka-server-2.2.2.RELEASE.jar: wro.js](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Eureka%20Dependency%20Check.html#l77_e2d04bbfcb7121b54b37f984242bba3756da09ab)**)**:

Problem: Ova ranjivost se odnosi na jQuery čije verzije manje od 3.0.0 su ranjive na XSS napade.

Rešenje: Ažuriranjem na trenutno najnoviju verziju (3.5) uklanja se mogućnost za xss napad.

* **Guava (**[spring-cloud-netflix-eureka-server-2.2.2.RELEASE.jar: wro.js](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/Eureka%20Dependency%20Check.html#l77_e2d04bbfcb7121b54b37f984242bba3756da09ab)**)**:

Problem: Nepovezana alocirana memorija omogućava udaljenim napadačima da sprovedu DoS na servere koji koriste ovu biblioteku. Napadačima se dozvoljava da alociraju memoriju i dodele joj prosleđene podatke bez prethodne provere.

Rešenje: Koristiti verziju guave koja je veća od 24.1.1.

* **PKI - Dom (**[dom4j-2.1.1.jar](file:///C:/Users/Kristina/Desktop/Projekti/Projekat%20XML/XWS/Dependency%20checks/PKI%20Dependency%20Check.html#l23_3dce5dbb3571aa820c677fadd8349bfa8f00c199)**)**:

Problem: Dopušta XXE napade, odnosno obradu DTD-ova i eksternih entiteta.

Rešenje: Koristiti verziju 2.1.3.

* **React – elliptic (**elliptic@6.5.2**)**:

Problem: Omogućeno je da dođe do neispravnosti u enkodingu, vodećim bajtovima ili “integer overflow“ prilikom upotrebe ECDSA potpisa.

Rešenje: Unaprediti verziju eliptic-a na verziju 6.5.3.

* **React – serialize\_javascript (**serialize-javascript@2.1.2**)**:

Problem: Prilikom serializacije u JSON format, može doći do „Code injection“ napada

Rešenje: Ažurirati trenutnu verziju na 3.1.0

* **React – yargs\_parser(**yargs-parser@11.1.1**)**:

Problem: Slanjem nevalidnog izraza parseru, produžava se vreme izvršavanja provere regularnosti prosleđenog izraza, ova ranjivost se može eksploatisati tako što bi napadač izvršio DoS napad.

Rešenje: Ažurirati trenutnu verziju na 13.1.2.

# 10. Insufficient Logging And Monitoring

Kvalitetno i sigurno logovanje događaja korisnika ili samog sistema prožima se kroz čitavu aplikaciju.

U aplikaciji svaki servis ima svoje logove, i separaciju po fajlovima za svaki tip događaja (Debug, Info, Warn, Error), što omogućava lakše snalaženje pri traženju podataka.

Postupak loginga zahteva da podaci o logu budu upotrebljivi što uključuje čuvanje informacije o tome ko je inicirao događaj. Kako bi se zaštitila izloženost osetljivih podataka, ukoliko u nekom slučaju dođe do kompromitovanja log fajla, informacija vezana za identitet je heširana i time zaštićenja. Kada ulogovan korisnik izvrši neku akciju koja se tiče nekog bezbednosnog svojstva ili čije je beleženje od interesa vrši se kreiranje pojedinačnog loga u kojem je navedeno vreme kad se izvršila akcija, komponenta koja ju je izvršila, njen opis i heširan identitet korisnika.

Zabeleženo vreme ispunjava vremenski format koji prati preporuku koristeći ISO 8601 standard.

Brisanje i modifikacija dozvoljeni su samo servisima koji vrše kreiranje novih logova. Direktorijum unutar kog se nalaze log fajlovi, zaštićen je od brisanja. Pristup log fajlovima dozvoljen je samo nadležnom korisniku. Ovakav vid zaštite ostvaren je upotrebom ACL liste.