# OWASP TOP 10

## 1.Insufficient Logging & Monitoring

Logovi su poprilicno vazni kako za debagovanje izvrsavanja odredjenog dela sistema tako i u smislu bezbednosti. Logovi sami po sebi bi trebali da sadrze dovoljno sadrzaja, ni previse ni premalo. Developer koji ce odrzavati aplikaciju ili security team bi trebao da ima uvid u sve potrebne informacije u jednom logu za jedan event od znacaja. Prosecno vreme otkad je doslo do penetracije u sistem do saznavanja security team-a za isti usled implementiranja loseg sistema za logging je 190 dana. Zbog takvih situacija se mora implementirati pouzdan, efikasan, brz i informativan nacin logginga u sistemu.

U nasem slucaju logovi ce ispisivati datum i tacno vreme (hh:mm:ss:SSS), nit koja je obavljala taj posao, servis iz kog je logger koji je ispisao odredjeni log i poruka koja je customize-ovana po ukusu developera u cilju lakog, brzog i efikasnog dolaska do potrebnih informacija pri iscitavanju logova.

Logovi su organizovani u nasem projektu u 4 datoteke (ERROR, WARNING, INFO i TRACE-DEBUG logove). Buduci da trace i debug logovi ce se cesto desavati na istom mestu smatrali smo da je najbolje da se nalaze na jednom mestu. Za proces logginga je koriscen logback logging mehanizam zato sto je log4j dependency imao ranjivosti a logback mehanizam nije pokazivao iste. Ova 4 fajla su takodje rolling fajlovi sto znaci da nakon odredjene granice, sto memorijske sto vremenske, ce se roll-ovati tj. u sustini zipovati kako bi omogucili stalnu dostupnost ispisivanja logova, manje memorijsko zauzece i manje "natrpavanje" logova unutar aktivnog fajla. Rolling fajl ce se takodje u ponoc automatski roll-ovati i napraviti fajl sa nasim custom nazivom i timestampom danasnjim u zip formatu.

Takodje pored ovih 4 rolling fajl mehanizama imamo ispis u konzolu kao i veoma vazni SMTP Logger. SMTP Logger nam sluzi da se izvesti "admin" sistema ili odrzavac ili security team ukoliko bi se desio ERROR u nasem sistemu sa hitnim prioritetom popravka iste. Logback mehanizam koristi 5 osnovnih nivoa logova, 1. ERROR 2. WARNING 3. INFO 4.DEBUG 5.TRACE, nivoi su poredjani tako da je od 1 do 5 poredjano po vaznosti za sistem a u obrnutom redosledu bi bili ako je u pitanju ucestalost u kodu tj. njihovom pojavljivanju. Ukratko cemo opisati sta koji log oznacava u nasem sistemu:

- Error - ne koristi se toliko cesto tj ne bi trebao da se koristi toliko cesto. Koristi se samo ukoliko se desi greska koja korisnicima onemogucuje da obavljaju odredjene funkionalnosti na sajtu i tada se odrzavac softvera treba odma izvestiti (mail). Takodje vazno je iskoristiti ERROR log kada, usled nekog bug-a je nemoguce odraditi neku jako vaznu biznis logiku koja je okosnica sistema.

- Warn - ukoliko se desi warning takodje bi trebao da se izvesti dev osoba, ali sa manjim prioritetom hitnog popravka. Warning bi trebao da se izbaci ukoliko sistem ima vremena da se oporavi ili i dalje ima svoju normalnu funkcionalnost ali bi taj "problem" trebao da se resi sto pre, najkasnije u narednih par dana.
- Info - treba da se koristi da se dokumentuju promene u stanju aplikacije (dodavanje, brisanje, izmena odredjenih bitnih delova sistema). Vazno da bi se pratilo sta se zapravo desavalo u sistemu u nekom trenutku (Korisnik Y se ulogovao) (Oglas taj i taj je presao iz Pending u Reserved itd itd)
- Debug - generalno sluzi za ispisivanje poruka ukoliko se desi exception tj. u situacijama kada bi se vratio status sa 4xx kodom.
- Trace - sluzi za ispis sadrzaja zahteva, ukoliko za to ima potrebe radi debagovanja sistema. Takodje moze da sadrzi vreme pocetka rada metode i/ili kraj ili samo ukupno trajanje metode (u nasem slucaju).

Ovakvi logovi bi trebali tako da se prave da su pogodni za slanje monitoring alata poput SIEM alata kako bi se na odredjen broj ili odredjen nivo loga obavila odredjena vrsta bezbedonosne akcije.

## 2.Injection

Neki od hakerskih načina za ubrizgavanje SQL izraza jesu kroz parametre url patha ili kroz input parametre formi. U cilju sprečavanja ove vrste napada, na serverskoj strani se vrši validacija. Kreirali smo specifikacije strukture JSON objekata u vidu JSON šeme. Na taj način omogućili smo validaciju, dokumentaciju i kontrolu interakcije sistema sa json podacima.

Radi obezbeđenja dodatne zaštite, svaki prihvaćen request smo obradili u smislu da smo proveravali da li sadrži maliciozan sadržaj koji bi omogućio injection napad. Za ovo smo koristili third part biblioteku koja je open source i dostupna je na sledećem linku: https://github.com/rkpunjal/sql-injection-safe. U projekat smo ovu biblioteku importovali kao dependency, a koristili smo njenu metodu IsSQLInjectionSafe kojoj smo prosleđivali sadržaj nekog input polja, u okviru requesta. Ukoliko se prepozna maliciozan sql injection sadržaj, korisniku se odgovara sa http statusom 400, a u loger se ispisuje informacija o pokušaju attack na sistem.

Takođe, koristili smo hibernate i jpa sloj, koji imaju ugrađene mehanizme koji sprečavanju sql injection napade.

## 3.Sensitive Data Exsposure

Definisali smo set permisija nad osetljivim podacima koristeći ACL(Access Control List). Na ovaj način smo zaštitili sve važne fajlove. ACL je implementiran na nivou operativnog sistema.

## 4.Security Misconfiguration

U svim kontrolerima i servisnim metodama obezbeđeno je rukovanje potencijalnim greškama.

## 5.Cross-Site Scripting XSS

Korisniku nije dozvoljen unos specijalnih karaktera, mogucih skript tagova. Validacijom input parametara obezbeđena je ova ranjivost sistema.

## 6.Broken Autentification

Prilikom registracije korisnik je morao da unese jaku lozinku (jedno veliko slovo, broj, specijalni karakter, minimalno deset karaktera), a zatim da je ponovi. Lozinke se heširaju, dodaje se salt i takve se čuvaju u okviru sistema.

Prilikom logovanja, ukoliko neko pokuša više od 3 puta da se uloguje sa neispravnim kredencijalima, dalji nastavak akcije se obustavlja. Vraća se HTTPStatus.CONFLICT (409) , a neulogovani korisnik sa te IP adrese se udaljava od logina na određeno vreme. U tom periodu odbija se svaki pokušaj logina. Ovo predstavlja dodatni sloj u zaštiti.

## 7.XML External Entities(XEE)

Ova vrsta napada je izbegnuta tako što je onemogućeno korišćenje eksternih entiteta i Document Type Definition fajlova.

## 8.Using components with Known Vulnerabilities

Ranjivosti sistema su detektovane i rešene. Za više informacija pogledati “DepedencyCheck” u okviru dokumentacije.

## 9.Insecure Deserialization

Korišćenjem novijih verzija third part biblioteka, ova potencijalna ranjivost našeg sistema je razrešena. Za više informacija pogledati “DepedencyCheck” u okviru dokumentacije.

## 10.Broken Access Control

Ova ranjivost aplikacije je rešena implementacijom RBAC-a(Role Based Access Control). Korisnicima sistema se dodeljuju role a njima definišemo koja prava nad resursima imaju određenje grupe korisnika. Dodavanjem i uklanjanjem korisnika politika rola se ne mora menjati. Sa @PreAuthorize anotacijom, postavljenom iznad metoda, proveravaju se uloge korisnika. Pored obezbeđenja kontrole pristupa, eskalacijom privilegija i ograničavanjem pristupa određenim resursima, čuvaju se i smanjuju upotreba mreže i memorije.