OWASP TOP 10

1) Injection

Od injection napada smo se zaštitili validacijom svih podataka koji se šalju na bek – whitelist pristup. Sve dto-ove u našoj aplikaciji smo validirali koristeći anotacije iz javax.validation.constraints paketa. Iznad svih obaveznih polja u dto-u smo stavili *@NotBlank* i *@NotNull* anotacije koje proveravaju da li je uneta vrednost za dato polje. Takođe, iznad svih polja je dodata *@Size* anotacija koja određuje minimalnu i maksimalnu dužinu ukoliko se nalazi iznad String-a, ili ukoliko se nalazi iznad kolekcije minimalan i maksimalan broj elemenata u kolekciji. Najvažnija anotacija je *@Pattern* kojoj se prosleđuje regex koji dato polje mora da zadovolji. Zahvaljujući regex-u onemogućeno je slanje neželjenih karaktera na back i zaštićeni smo od injection napada. Na isti način smo se zaštitili od injection napada preko *RequestParams* i *PathVariable*
Zaštitu od injectiona na frontu smo implementirali pomoću *Validators* modula iz @angular/forms.
Takođe, pošto smo omogućili filtriranje pomoću regexa, gde se od korisnika očekuje da unese regularni izraz, razmatrali smo da li da validiramo i sam uneti regularni izraz. Nakon istraživanja, otkrili smo da je validacija regularnog izraza rekurzivna i vremenski zahtevna, te smo se odlučili da je ne implementiramo da ne bismo nepotrebno omogućili ReDoS (RegularExpressionDenialOfService).
Kako koristimo Hibernate ORM, nigde nemamo dinamičko ubacivanje neprocesovanog korisničkog unosa u same upite nad bazama.

2) Broken-Authentication

Brute-force napad – keycloak je konfigurisan na takav način da nakon 10 neuspešnih prijava sa istog korisničkog naloga privremeno zaključava taj nalog, takođe, u slučaju dve prijave u jako kratkom vremenskom intervalu (koji je moguće samo programski ostvariti), takođe se zaključava nalog.

Jačina lozinke – postoje ograničenja nad lozinkama koje korisnik može da unese (minimalna dužina, neophodno prisustvo barem jednog broja i specijalnog karaktera, lozinka se ne sme ponoviti u prethodne tri promene, lozinka se ne sme nalaziti na spisku najčešćih lozinki – blacklist).

Generisanje tokena – JWT tokeni se potpisuju pomoću RSA256 ključa koji se može rotirati, skladište se u sessionstorage i nakon logout-a se brišu iz skladišta i takođe invalidiraju na serverskoj strani (ako je neko preuzeo token, nakon logout-a neće više biti validan). Takođe, trajanje tokena je 5 minuta, nakon čega mora da se refresh-uje.

Skladištenje lozinki – lozinke se ne čuvaju u plain text formatu već se heširaju i koristi se i salting mehanizam.

3) Sensitive Data Exposure

Prilikom rada sa pacijentima u našoj bolničkoj aplikaciji smatrali smo da je njihov medicinski karton osetljiva informacija kojoj ne bi trebao svako da ima pristup. Svaki novi pacijent koji je dodat u bazu je prethodno šifrovan (koristimo AES simetrični ključ dužine 256 bita – ispoštovane su sve trenutne preporučene vrednosti – ključ se skladišti u JCEKS skladištu koje je zaštićeno lozinkom, kao i sam ključ), i time je i prilikom potencijalnog napada na bazu aplikacije i krađe podataka obezbeđena tajnost njegovih ličnih podataka. Jedino doktori aplikacije imaju pristup pacijentima – doktor pacijent poverenje. Zaštitu podataka o pacijentima u tranzitu obezbeđuje https.

Keycloak obezbeđuje sigurno rukovanje korisnicima u sistemu (snažan hashing algoritam i primena salting mehanizma za lozinke).

4) XML External Entities

Ne koristimo XML format nigde u aplikaciji.

5) Broken Access Control

Svi resursi za koje postoji mogućnost pristupa tokom korišćenja naše aplikacije su zaštićeni pomoću keycloak-a. Kompletan RBAC sistem je implementiram korišćenjem rola i permisija vezanih za te role.

Na frontu smo podesili guards za rute, da ne bi mogao bilo koji korisnik da ukuca bilo koju rutu u url i redirektuje se na nju.

6) Security Misconfiguration

Sve komunikacije su konfigurisane tako da se odvijaju preko https-a, pa i komunikacija sa keycloak serverom.

7) Cross-Site Scripting XSS

Zaštita od XSS napada je sastavni deo Angular aplikacije. Kako ne manipulišemo DOM stablom, ne patimo od DOM XSS napada. U WebSecurityConfiguration konfiguracionoj klasi je reimplementirana *configure* metoda i dodata je zaštita od xss napada:

```
http.headers().xssProtection()
.and().contentSecurityPolicy("script-src 'self'");
```

8) Insecure Deserialization

Svaka deserijalizacija objekta je odrađenja u *try catch* bloku i ukoliko dođe do neispravne deserijalizacije to je bilo zabeleženo. Takođe, DTO objekti su validirani, te se u njihovim poljima ne može naći bilo šta. Uz to, komunikacija između MedicalDevice i Hospital aplikacija je digitalno potpisana, te smo potpuno sigurni da podaci nisu izmenjeni u toku transporta.

9) Using Components with Known Vulnerabilities

Ne koristimo komponente koje imaju poznate bezbednosne mane.

10) Insufficient Logging & Monitoring

Svaki izlaz iz aplikacije je pokriven logovima koristeći *Logger* klasu iz org.slf4j paketa. U sistemu postoje 3 tipa logova: informacijoni logovi – info, logovi upozorenja – warn i logovi grešaka aplikacije – error. Većina logova u aplikaciji je bila informaciona, oni su standardni logovi koji beleže normalan tok korišćenja aplikacije. Logovi upozorenja predstavljaju neku grešku sa klijentove strane – pogrešni kredencijali prilikom logovanja na aplikaciju, pokušaj pristupa resursu po nepostojećem id… Logovi greške su interne serverske greške – pogrešna putanja do direktorijuma, loš ključ prilikom šifrovanja/dešifrovanja podataka, neuspela konekcija ka bazi..

Svi logovi sadrže informaciju o:

- Datumu – tačan datum kada se događaj desio
- Vremenu – tačno vreme kada se događaj desio
- Izvor događaja – koji program, komponenta ili korisnički nalog je prouzrokovao događaj
- Tip događaja – info, warning, error
- Opis događaja – opis koji bliže opisuje konkretan događaj
- Status kod/rezultat – status kod koji je prouzrokovao izvršen događaj
- Opis status koda – poruka koja bliže opisuje razlog koji je doveo do status koda

Logovi obezbeđuju neporecivost.

Primeri logova:

a. Info log

```
logger.info(String.format("%s called method %s with status code %s: %s",
        "Medical device", "receivePatientStatus", HttpStatus.OK, "patient status
received"));
```

b. Warn log

```
  logger.warn(String.format("%s called method %s with status code %s: %s",
        "Medical device", "receivePatientStatus", HttpStatus.NOT_FOUND, "non
existent patient id"));
```

c. Error log

```
  logger.error(String.format("%s called method %s with status code %s: %s",
        "Medical device", "receivePatientStatus", HttpStatus.BAD_REQUEST, "untrusted
certificate"));
```

Admin bolničke aplikacije i super admin adminske aplikacije imaju pregled logova i takođe imaju pregled alarma koje su ti logovi izazvali i mogu blagovremeno da reguju ukoliko dođe do napada na aplikaciju.

Postoje predefinisani alarmi u aplikaciji koji se okidaju u slučaju određenih napada, to jest, prisustva određenih logova u određenom broju (više od 5 neuspešnih prijava sa istog naloga u roku od 1 minuta,…). Takođe, admin bolničke aplikacije ima mogućnost kreiranja novih pravila koja okidaju alarme i na taj način mu je obezbeđen dodatan vid monotirnga nad aplikacijom.