OWASP TOP 10 izveštaj - TIM 17

Uvod

U okviru projekta iz predmeta “Bezbednost u sistemima elektronskog poslovanja” potrebno je napraviti izveštaj o tome kako i gde u projektu je rešena konkretna grupa problema sa OWASP top 10 liste kao i problemi mehanizama za logovanje, autentifikaciju i autorizaciju.

S obzirom na to, u ovom dokumentu ćemo proći kroz svih 10 OWASP problema sa liste kao i kroz probleme mehanizama za logovanje, autentifikaciju i autorizaciju koji su implementirani u projekat kao i način i mesto njihove implementacije.

OWASP TOP 10

OWASP 10 - Nevalidirana preusmerenja i prosleđivanja (Unvalidated redirects and forwards)

Ovaj propust rešava sam Play framework jer sva preusmerenja idu preko 302 i nisu hardkodovane putanje.

OWASP 9 - Korišćenje komponenti sa poznatim ranjivostima (Using components with known vulnerabilities)

Ovo smo pokušali da izbegnemo tako što smo koristili komponente koje su se pokazale kao dovljno dobre za industriju, a za one za koje znamo da imaju ranjivosti, pokušali smo da ih zakrpimo.

Primer toga je DOM parser XML-a koji nam obezbeđuje Java. Njegove nedostatke koji omogućavaju XML entity injection smo premostili upotrebom validacije putem šeme.

OWASP 8 - Cross-site request forgery

Play framework podržava mehanizme za praćenje sesije među kojima su i oni koji služe za odbranu od ovog napad. Sam framework podržava mehanizam za rad sa secret key-om. Na žalost, zbog načina na koji smo implementirali frontend, ovo nam nije od koristi tako da smo morali da koristimo drugi standard.

Za rešavanje ovog problema smo koristili JWT (JSON Web Token) kao tajni token i koristili smo GET i POST metode onako kako HTTP standard nalaže. Ovo znači da problem CSRF-a rešili ubacivanjem tajnog tokena u svaki POST zahtev.

OWASP 7 - Missing function level access control

Rešeno putem @Before tagova kako bi se neautorizovanim korisnicima onemogućio pristup stranicama za koje nemaju autorizaciju.

Kako bismo došli do svih @Before tagova koji su potrebni, najpre smo napravili Model pretnji, pa nakon toga ustanovili gde su potrebne ove restrikcije.

UI se takođe brine da na osnovu RBAC modela i nivoa pristupa prijavljenog korisnika prikaže samo elemente ka kojima korisnik ima pristup. Ovo je realizovano putem centralizovanog servisa za proveru autorizacije i prava pristupa.

Centralizovani servis za proveru autorizacije i prava pristupa se poziva pri svakom izvršavanju nekog servisa.

OWASP 6 - Sensitive data exposure

Aplikacija je na production mode-u, sve je sakriveno, a poruke o greškama minimalne.

Play framework ima dva režima rada. Jedan je developement mode, a drugi je production mode. U developement mode-u, pri izvršavanju koda koji ima grešku, uzrok te greške se ispisuje na ekran što i nije baš poželjan scenario jer potencijalni napadači mogu da otkriju tačne nedostatke aplikacije.

Iz tog razloga, nakon završetka aplikacije, ona se stavlja u production mode čime se ovi ispisi onemogućavaju.

Drugi vid ispisa je onaj koji nam govori u konzoli browser-a podatke o grešci koja je nastala. Svođenjem ovih ispisan na minimum, kao i davanje što manje direktnih informacija o grešci koja je nastala, sakrivamo arhitekturu sistema od potencijalnih napadača.

Što se tiče zaštite osetljivih podataka korisnika, kao što su čuvanje šifre komunikacija između klijenta i servera, to smo rešili na sledeće načine.

Komunikacija između klijenta i servera

Sva komunikacija između klijenta i servera se obavlja preko TLS/SSL, odnosno HTTPS konekcije. Ukoliko bi korisnik pokušao da pristupi stranici koja nije https, bio bi redirektovan na https stranicu.

Takođe, pored ovoga, aplikacija podržava i enkriptovanje sadržaja koji se šalje. Ova enkripcija je u vidu XML Standard enkripcije.

Čuvanje šifre korisnika

Čuvanje šifre korisnika je implementirano Hash and Salt metodom.

Hash algoritam na serveru je SHA1 i on se izvršava 31450 puta za šta je potrebno oko 3 sekunde na modernom i5 računaru čime drastično povećavamo vreme proračuna rainbow tabela. Pored toga, klijent serveru nikada ne daje šifru već je sam klijent prvo hash-uje SHA256 algoritmom pa onda šalje hash šifre.

OWASP 5 - Security misconfiguration

Play Framework je pravljen sa bezbednošću u vidu. Kao takav, ne dozvoljava neovlaštenim korisnicima da listaju sadržaj direktorijuma.

Administratorski nalog je iskonfigurisan tako da mu korisničko ime i lozinka ne budu podrazumevani i predvidivi dok je sample aplikacija uklonjena kako ne bi otvarala potencijalan back door.

Takođe su minimalno i oprezno korišćene eksterne komponente kako se ne bi nasledili njihovi potencijalni propusti. Ovo se ogleda u tome što, pored istraživanja bezbednosti komponenti i paketa koje smo koristili, nisu uvezeni celi paketi sa svim klasama već samo oni koji su neophodni za funkcionalnost date klase.

Framework je takođe podešen da ne bude u dev modu, koji je default za razvoj, već u prod modu koji uklanja sve logove koji su korisni za debug-ovanje i zatvara debug port aplikacije.

Veoma je važno napomenuti da je neophodno da se odradi konfiguracija sledećih parametara koji bi trebali da budu tajni, a trenutno se nalaze na javnom repozitorijumu:

- U aplication.conf fajlu potrebno promenuti sledeće stvari:
  - Podaci o bazi podataka (korisničko ime i lozinka)
  - Application.secret
- Podaci o sertifikatu u klasama koje ga koriste i sam sertifikat
  - Ukoliko se ne promene podaci o sertifikatima a promene sertifikati, ovo se jednostavno manifestuje nemogućnošću potpisivanja dokumenata niti korišćenjem same aplikacije jer je rad aplikacije preko https konekcije.

OWASP 4 - Insecure direct object references

Reference na objekte nisu kao nisu vidljive kao paramepri korisnicima, a pravo pristupa se proverava za svaku vrednost parametara.

OWASP 3 - Cross-site scripting

Kako bi se izbegli Cross-site scripting napade, bilo je potrebno validirati korisnički unos i enkodovati unos korisnika kao i odgovor servera.

Za enkodovanje unosa i odgovor servera se pobrinuo Play framework tako što je pružio mogućnost escape-ovanja stringova. Ova opcija se omogućava postavljanjem zastave

future.escapeInTemplates=true

nutar aplication.conf fajla aplikacije.

Validacija je urađena na više načina i na više mesta.

Prva vid validacije se obavlja na unosu korisnika kao i njegovom logovanju na samom frontendu. Pošto nam je administrator čovek od poverenja, on se brine o tome da se u bazu ne unese korisnik koji ne ispunjava uslove validacije, odnosno očekuje se od njega da ne zaobilazi frontend.

Drugi vid validacije je na backend-u i realizovan je kao whitelist. Radi se o validaciji unosa XML fajlova. Validacija XML fajlova je realizovana kao validacija preko .xsd šeme. Naravno, o validnom unosu XML-a brine i frontend tako što se sam XML unosi putem XML editora koji ima postavljen template validnog dokumenta.

OWASP 2 - Broken authentication and session management

Ovaj problem smo u najvećoj meri rešili time što smo koristili REST servise, odnosno Play framework koji je stateless.

Drugi način kako smo ovo obezbedili jeste tako što imamo centralizovan sistem za autentifikaciju koji se bavi autentifikacijom i praćenjem “sesija”.

Lozinke se čuvaju na na bezbedan Hash and Salt način dok se sama autentifikacija kao i ceo rad aplikacije odvija korišćenjem TLS odnosno HTTPS protokola.

Server takođe generiše i tajni token koji služi za praćenje sesije tako da nije moguće da se neko drugi predstavi kao naš korisnik.

Identifikator sesije nije dostupan putem URL-a.

Sesije se uništavaju logout metodom koja se nalazi unutar Login klase čime se invalidira sesija i na backend-u.

Tokeni takođe imaju vek trajanja tako da se posle određenog vremena sami invalidiraju.

OWASP 1 - Injection

SQL injection

Play framework, koji je korišćen kao drugi sloj aplikacije, odnosno server koji podržava i koristi metode višeg nivoa upita nad SQL bazom. Ove metode rade nad JPA, a ne direktno nad bazom i kao takve su već escape-ovane.

Ni u jednom trenutku se bazi ne pristupa metodama niskog nivoa, odnosno čistim “SELECT” upitima tako da ne postoji način da dođe do SQL injection-a.

Validacija ulaza je takođe urađena na frontend-u, tako da bi to trebalo da otera većinu n00b “hack-era”, dok backend čuva upit visokog nivoa, pa mu validacija nije neophodna.

Validacija ulaza je implementirana whitelist-om zbog dodatne sigurnost.

SQL smuggling

Ovo je rešeno jednostavnim podešavanjem play framework konfiguracije da normalizuje sve input-e.

Konkretna linija od značaja je: *future.escapeInTemplates=true*

Ovaj problem takođe rešava i korišćenje JPA umesto čistog SQL-a nad bazom.

XML entity injection

Rešeno je validacijom putem šeme.

Problemi logovanja, autentifikacije i autorizacije

Logovanje

Logovanje akcija kako korisnika tako i servera omogućava sam play framework.

Autentifikacija

Autentifikacija se vrši na dva nivoa.

Prvi nivo autentifikacije je pri prijavi korisnika na sistem. Korisnik je dužan da unese svoje korisničko ime i lozinku.

Drugi nivo autentifikacije je pri unošenju dokumenata kod kojih, pored potrebe da korisnik bude prijavljen na sistem, mora da potpiše dokument, a za to mu je neophodan važeći sertifikat.

Autorizacija

Autorizacija je vršena po RBAC modelu. Sistem podržava 3 vrste prijavljenih korisnika i 4. ne prijavljenog. Svaka vrsta korisnika ima svoja ovlašćenja o čemu se brine centralizovan sistem za evidenciju pristupa.