OWASP TOP 10 – 2017

Predmet: Bezbednost u Sistemima Elektronskog Plaćanja

Zadatak: Proći kroz OWASP TOP 10 listu za 2017 godinu

Autor: Nikola Dakić SW59/2013

Napomena: Treba napomenuti da ovaj rad predstavlja samo dobru osnovu za razumevanje toga koliko zahtevan posao može predstavljati, šta to sve znači i koliko je značajan posao obezbediti jednu aplikaciju. Gledano je da se prođe kroz svaku tačku sa što više razumevanja i da se primene odgovarajuće mere koje svaka tačka zahteva. Sigurno je da je ovde zagreban samo vrh ledenog brega i da obezbeđivanje jednog sistema predstavlja težak posao.

Osnovna struktura svake tačke:

- Pretnja
- Sigurnosni propusti
- Uticaj napada
- Kada je aplikacija ranjiva
- Mere zaštite
- Primena

OWASP Top 10 Application Security Risks – 2017

- **A1 Injection**
- **A2 Broken Authentication**
- **A3 Sensitive Data Exposure**
- **A4 XML External Entities (XXE)**
- **A5 Broken Access Control**
- **A6 Security Misconfiguration**
- **A7 Cross-Site Scripting (XSS)**
- **A8 Insecure Deserialization**
- **A9 Using Components with Known Vulnerabilities**
- **A10 Insufficient Logging & Monitoring**

A1 Injection

Pretnja

Bilo koji izvor podataka može biti ugrožen a samim tim i svi tipovi korisnika.
Injection propusti se javljaju kada napadač ima mogućnost da pošalje “neprijateljske” podatke interpreteru.

Sigurnosni propusti

Injection propusti mogu biti široko rasprostraljeni, naročito u nasleđenom kodu.
Obično se nalaze u SQL, LDAP (protokol za traženje određenih informacija od servera), Xpath, NoSQL ili ORM upitima. Lako su uočljivi prilikom posmatranja koda.
Scanners i Fuzzers (softver za unos neočekivanih podataka) mogu pomoći napadaču da uoči Injection propuste.

Uticaj napada

Može dovesti do gubitka ili oštećenja podataka.
Ukidanja privilegija korisnicima.
Kao i do preuzimanja čitave web aplikacije.

Kada je aplikacija ranjiva?

Kada podaci koje korisnik unosi, nisu ni na koji način validirani, filtrirani ili sakcionisani od strane aplikacije.
Kada se prave upiti ka interpreteru van dozvoljenog konteksta.
Kada se 'nepoverljivi' podaci direktno koriste ili spajaju sa upitima.

Mere zaštite

Koristiti sigurne API-je, koji vrše parametrizaciju osetljivih podataka.
Koristiti ORMs.
Koristiti whitelist-u dozvoljenih unosa, filtrirati nepoverljive podatake.

Primena

- Konverzija nepoverljivih podataka.
- Provera validnosti nepoverljivih podataka korišćenjem regularnih izraza.
- Korisćenje ORMs (Hibernate) i njihove prirodne sposobnosti parametarizacije.

A2 Broken Authentication

Pretnja

Napadač ima ogroman broj validnih korisničkih imena i lozinki, koje koristi za bruto force ili za dictionary napad.
Takođe ima dobro razumevanje o upravljanju sesijama.

Sigurnosni propusti

Uglavnom se odnose na dizajn i implementaciju autentifikacije koja obično nije laka za realizovanje.
Loše upravljanje sesijama.

Uticaj napada

Napadač može da dobije pristup ka nekoliko naloga ili samo jednom admin nalogu i tako kompromituje čitav sistem.
Što dalje može izazvati opasnosti kao što su: iznuđivanje novca, krađa identiteta, otkrivanje osetljivih podataka..

Kada je aplikacija ranjiva?

Kada dozvoljava napad kao što je Credential stuffing - korisnik ima određen broj ispravnih credencijala sa nekog sajta, i sada te iste pokušava da primeni na neki drugi.
Kada dozvoljava Bruto Force ili neke druge automatske napade.
Dozvoljava default, slabe ili standardne šifre, kao što su : password1, admin/admin
Koristi slab proces za obnovu zaboravljenih šifri.
Koristi obične, enkriptovane ili slabo hashovane šifre.
Nema više faktora za autentifikaciju. Recimo lokaciju.
Otkriva ID sesije u URL-u.
Ne generiše novi id sesije posle svakog uspešnog logovanja.
Ne validira ispravno id sesije ili token.

Mere zaštite

Implementirati autentifikaciju sa više faktora kako bi se sprečili automatizovani napadi kao što su credential stuffing i brute force.
Ne raditi deploy aplikacije sa default credential-ima.
Implementirati proveru slabih i standardnih lozinki prilikom registracije.
Prilikom registracije ili obnove credential-a prikazivati iste poruke za sve, kako se iz njih ne bi moglo ništa zaključiti.
Ograničiti broj neuspešnih logovanja/zahteva. Sve logovati i obavestiti admina kada je možda došlo do credential stuffing ili brute force napada.
Generisati svaki put ponovo id sesije/tokene. Ne držati id sesije u URL-u. Poništiti im validnost prilikom logout-a. Postaviti vreme isticanja sesije.
Koristiti već gotova rešenja upravljanja autentifikacijom (Spring Security, ASP.NET)

Primena

- Korišćen Spring Security.
- Generiše se token prilikom svakog uspešnog logina.
- Uvedeno vreme isticanja tokena (1h).
- Validiranje tokena prilikom svakog requesta.
- Implementirana provera slabih i standradnih lozinki prilikom registracije.
- Uvedeno blokiranje korisnika posle 10 neuspešnih pokušaja logovanja.
- Logovanje osetljivih podataka.

A3 Sensitive Data Exposure

Pretnja

Radije umesto napada na enkripiciju, napadač ukrade ključeve i izvrši man-in-the-middle napad ili ukrade podatke sa servera/ tranzitu i zatim proba bruto-force napadom da provali lozinke.

Sigurnosni propusti

Glavni propust ovde je nedostatak ekriptovanja osetljivih podataka.
A čak i kada je enkripcija implementirana, nesigurno generisanje i čuvanje ključa ili korišćenje slabih algoritama, predstavljaju glavni propust.

Uticaj napada

Ovi propusti mogu da kompromituju sve bitne podatke kao što su: kreditne kartice, lični podaci..

Kada je aplikacija ranjiva?

Kada se prilikom skladištenja ili tranzita osetljivih podataka (npr. poodaci koji spadaju pod GDP regulaciju):

- podaci šalju kao čist tekst
- koristi neki stari ili slab kriptografski algoritam
- koriste default ili slabi kripto ključevi

Mere zaštite

Identifikovati osetljive podatke i klasifikovati ih.
Primeniti bezbednosne mere po klasifikaciji.
Ne skladištiti osetiljive podatke bez potrebe, već ih odbaciti što je to pre moguće.
Enkriptovati sve osetljive podatke koji su uskladišteni.
Pratiti najnovije standarde za enkripciju.
Enkriptovati sve osetljive podatke u tranzitu sa sigurnim protokolom kao što je TLS.
Skladištiti šifre upotrebom jake salt hashing funckije sa delay faktorom.

Primena

- Hashovanje lozinke uz pomoć BCryptPasswordEncoder sa jačinom 12
- Omogućen HTTPS Protokol

A4 XML External Entities (XXE)

Pretnja

Napadač može da iskoristi ranjive XML procesore/parsere, ako može da uploaduje XML ili ubaci neprijateljski sadržaj u XML dokument.
Time napadač može da izazove Denial of Service (DOS) napad, ili pristupa lokalnim/udaljenim fajlovima/servisima.

Sigurnosni propusti

Ne proverava se validnost unosa, koji može ukazivati (reference) na neki eksterni sadržaj i koji se kasnije prilikom parsiranja XML dokumenta, može izvršiti neki nepoželjan način.

Uticaj napada

Ovakvi propusti mogu biti iskorišćeni za ekstratovanje podataka, izvršavanje udaljenog zahteva sa servera, skeniranje unutrašnjeg sistema, izvršavanje denial-of-service napada, kao i nekih drugih napada.

Kada je aplikacija ranjiva?

Kada aplikacija prihvata XML direktno ili XML uploadovanje, naročito od nepoverljivih izvora, ili omogućava ubacivanje podataka u XML dokument koji se onda parsira od stranje XML procesora.

Ako aplikacija koristi SOAP pre verzije 1.2, verovatno je podložna XXE napadima.
Ako aplikacija koristi SAML za obradu identiteta u okviru federalne bezbednosti. SAML koristi XML za tvrdnje o identitetu i može biti ranjiv.

Mere zaštite

Kad god je moguće, koristiti manje komplikovan format podataka kao što je JSON i izbegavati serializaciju osetljivih podataka.
Upgrade-ovati sve XML procesore i biblioteke koje koristi aplikacija.
Implementirati whitelistu dozvoljenih unosa, kako bi se izbeglo dodavnje neprijateljskog sadržaja.
Validirati uploadovne XML ili XSL fajlove korišćenjem XSD validacije ili slično.

Primena

- Korišćen JSON

A5 Broken Access Control

Pretnja

Ekploatacije kontrole pristupa je ključna tačka napada.
Određeni alati mogu detektovati nedostatak kontrole pristupa, ali ne u potpunosti.

Sigurnosni propusti

Nedostatci kontrole pristupa su uobičajeni usled nedostatka efektivnog funkcionalnog testirana od strana developer-a.

Uticaj napada

Napadač može da se ponaša kao privilegovan korisnik i izvršava privilegovane funkcionalnosti.

Kada je aplikacija ranjiva?

Kada je omogućena modifikacija URL-ova, tj pristupanje nedozvoljenim stranicama pa i funkcionalnostima.
Kada je moguće ponašati se kao ulogovan korisnik bez logovanja ili kao admin sa pravima običnog korisnika.
Kada je moguća manipulacija meta podacima tj. tokenima, kolačićima.

Mere zaštite

Implementirati access control mehanizam i na klientskoj i na serverskoj strani.
Logovati neuspešne kontrole pristupa.
Uvesti limit pristupa kontrolerima, kako bi se izbegli automatizovani napadi.
JWT token bi trebao da postane nevalidan posle logout-a.

Primena

- Client - Restrikcija URL-ova na osnovu uloga
- Server - Restrikcija zahteva na osnovu ovlašćenja

A6 Security Misconfiguration

Pretnja

Napadač pristupa default nalozima, nekorišćenim stranicama, nezaštićenim fajlovima i ostalim manama, kako bi saznao više o sistemu i iskoristio to znanje.

Sigurnosni propusti

Pogrešno konfigurisanje se može desiti na bilo kom nivou aplikacije, uključujući platformu, web server, frejmwork itd.

Uticaj napada

Ovakvi propusti omogućavaju napadaču neovlašćen pristup sistemskim podacima ili funkcionalnostima.

Kada je aplikacija ranjiva?

Kada su uključeni ili instalirani nepotrebni dodaci (features), portovi, servisi, nalozi i privilegije.
Kada se greške obrađuju na takav način da ih napadač može videti.
Kada postoje default nalozi sa default šiframa.
Kada sistemi koji podržavaju upgrade-ovanje dodataka, ne sadrže najnovije verzije.
Kada server ne šalje sigurnosna zaglavlja ili nisu postavljene sigurne vrednosti.

Mere zaštite

Napraviti sigurno okruženje i automatizovati sva naredna objavljivanja kako ne bih slučajno dolazilo do nekih grešaka.
Koristiti samo neophodne dodatke.
Redovno updejtovanje frejmworka, biblioteka itd.
Zaštiti osetljive konfiguracione podatke.
Koristiti custom error poruke.

Primena

- Nedozvoliti prikaz serverskih grešaka napadaču.
- Koristiti custom error poruke.

A7 Cross-Site Scripting (XSS)

Pretnja

Napadač šalje skriptu koja iskorišćava interpreter browsera.
Bilo kakav izvor podataka može biti ulazni vektor napada, uključujući i podatke iz baze.

Sigurnosni propusti

XSS je najčešća mana web aplikacija. Ova mana se aktivira kada aplikacija šalje podatke browseru bez validnog encodiranja izlaznih podataka.

Uticaj napada

Napadač može izvršiti skriptu u žrtvinom browseru i uzeti korisniku sesiju, redirektovati ga ili ubaciti neki nepoželjan sadržaj.

Kada je aplikacija ranjiva?

Postoje 3 tipa XSS, koji napadaju žrtvin browser:

- Reflected XSS - Aplikacija dozvoljava korisniku da unese nepoželjan unos i prikaže njegov HTML izlaz. Uspešan napad može omogućiti napadaču da izvrši proizvoljan HTML i JavaScript kod u žrtvinom browseru.
- Stored XSS - Aplikacija dozvoljava da napadač sačuva nepoželjnu skriptu u bazi i da je posle neki drugi korisnik ili administrator izvrši.
- DOM XSS - Stranice koje dinamički učitavaju podatke na stranici mogu biti kontrolisane od strane napadača. Aplikacija ne bi trebala da šalje podatake nesigurnim JavaScript API-ima.

Mere zaštite

Koristiti frejmworke koji po dizajnu imaju zaštitu od XSS, kao što su Ruby on Rails, ReactJS..
Izbegavati neproverene HTTP zahteve.
Primeniti konteksno osetljivo encodovanje kada se prikazuje podatak na klijentskoj strani.
Kodirati aplikaciju kao da imamo XSS sadržaj u bazi.
Očekivati da napadač koristi sakriveni URL.
Whitelista dozvoljenih unosa.
Validiranje Regeksima.

Primena

- Validacija pojma pretrage regeksom.
- Korišćenje AngularJS-a koji ima po defaultu zaštitu od prikaza podataka (output encoding)
- Validacija dodavanja artikla na klijentskoj strani od strane angulara, na serverskoj provera (SafeHtml) prilikom mapiranja na java objekat.

A8 Insecure Deserialization

Pretnja

Napadač ubacuje serializovane podatke koji se mogu izvršiti na napoželjan način prilikom deserijalizacije.

Sigurnosni propusti

Ne proverava se uneseni sadržaj (user-input) prilikom deserijalizacije.

Uticaj napada

Uticaj propusta deserijalizacije ne može se proceniti. Ove greške mogu dovesti do napada na daljinu, što je jedan od najtežih napada.

Kada je aplikacija ranjiva?

Aplikacija i APIs će biti ugroženi ako deserializuju neprijateljski podatke koje je podmetnuo napadač. Ovo može dovesti do dve vrste napada:

- Napadač modifikuje logiku aplikacije ili postiže izvršavanje udaljenog koda ako postoje klase koje dozvoljavaju da aplikacije promeni svoje ponašenje za vreme ili posle deseralizacije.
- Tipični napadai kao što je access-control-related napad, gde se koristi postojeća struktura ali se sadržaj izmeni.

Serijalizacija se može koristiti u aplikacijama za:

- Udaljenu i međusobnu komunikaciju (RPC/IPC)
- Wire protocols, web services, message brokers
- Caching/Persistence
- Databases, cache servers, file systems
- HTTP cookies, HTML form parameters, API authentication tokens

Mere zaštite

Jedini bezbedan arhitektonski šablon je ne prihvatanje serijalizacije iz nepouzdanih izvora ili korišćenje medijuma koji prihvata samo primitivne tipove podataka. Ako to nije moguće:

- Sprovesti proveru integriteta kao što digitalni potpis na bilo kom serijalizovanom objektu kako bi se izbeglo dodavanje neprijateljskog sadržaja.
- Sprovođenje strogih ograničenja tipa tokom deseralizacije pre kreiranja objekta.
- Izolovanje i pokretanje koda koji se deserijalizuje u okruženju sa što manjim privilegijama ako je to moguće.
- Logovanje grešaka prilikom deserijalizacije.
- Ograničavanje ili nadgledanje dolazne i odlazne mreže povezivanja.
- Praćenje deserijalizacije, upozorenje da li se korisnik stalno deserijalizuje.

Primena

- /

A9 Using Components with Known Vulnerabilities

Pretnja

Napadač koristi već prethodne poznate ranjivosti ili traži nove.

Sigurnosni propusti

Opseg ranjivosti je velik jer developeri uglavnom ne razumeju koje sve komponente koriste.

Uticaj napada

U zavisnosti od veličine propusta, uticaj može biti jako mali ili jako velik.

Kada je aplikacija ranjiva?

Kada ne znamo verzije svih komponenata koje koristimo, na klijentskoj i serverskoj strani. Komponente koje direktno koristimo, i one ugnježdene (nested) zavisnosti.
Kada je softver ranjiv, nema više podršku, ili out of date. Ovo uključuje OS, WEB/Aplication server, database management sistem, biblioteke itd.
Ako se ne vrši redovno skeniranje ranjivosti.
Ako se ne ispravljaju (update) frejmvorci, zavisnosti itd.

Mere zaštite

Ukloniti sve nepotrebne zavisnosti, komponente, fajlove.
Kontinuirano pratiti sve update.

Primena

- Korišćenje samo neophodnih komponenti

A10 Insufficient Logging&Monitoring

Pretnja

Nedostatak logovanja i nedovoljno praćenje je osnova svakog većeg incidenta.

Napadač se oslanja na nedostatak praćenja i pravovremenog reagovanje kako bi postigao svoje ciljeve bez otkrivanja.

Sigurnosni propusti

I ovaj nedostatak je uključen u top 10 na osnovu istraživanje iz industrije.
Jedna strategija za utvrđivanje efikasnosti praćenja je ispitati logove nakon testiranja penetracije. Akcije treba da budu doboljno zabeležene kako bi se razumela kakva je potencijalna šteta.

Uticaj napada

Najuspešniji napadi počinju sa ispitivanjem ranjivosti.Omogućavanje takvih ispitivanje može dovesti do uspešnog napada i do 100%.

Kada je aplikacija ranjiva?

Kada se događaji kao što su logovanje, neuspelo logovanje, određene bitne transakcije ne loguju.
Kada se greške i upozorenja ne loguju.
Kada se logovi ne nadgledaju tako da obaveste na neke sumnjive aktivnosti.
Kada se logovi skladište samo lokalno.
Kada aplikacije nije u mogućnosti da upozori na napade u realnom ili blizu realnog vremenu.

Mere zaštite

Omogućiti logovanje svih bitnih akcija.
Omogućiti generisanje logova u formatu koji je lak za korišćenje od strane centralizovanog sistema za upravljanje logovima.
Obezbediti da se logovi ne mogu menjati ili brisati.
Omogućiti da sistem obaveštava sumnjiva ponašanja.
Uspostaviti plan reagovanja na incidente.

Primena

- Logovanje svih bitnih akcija

Spisak implementiranih stvari

- ✓ Konverzija nepoverljivih podataka.
- ✓ Provera validnosti nepoverljivih podataka korišćenjem regularnih izraza.
- ✓ Korisćenje ORMs (Hibernate) i njihove prirodne sposobnosti parametarizacije.
- ✓ Korišćen Spring Security.
- ✓ Generiše se token prilikom svakog uspešnog logina.
- ✓ Uvedeno vreme isticanja tokena (1h).
- ✓ Validiranje tokena prilikom svakog requesta.
- ✓ Implementirana provera slabih i standradnih lozinki prilikom registracije.
- ✓ Uvedeno blokiranje korisnika posle 10 neuspešnih pokušaja logovanja.
- ✓ Hashovanje lozinke uz pomoć BCryptPasswordEncoder sa jačinom 12
- ✓ Omogućen HTTPS Protokol
- ✓ Client - Restrikcija URL-ova na osnovu uloga
- ✓ Server - Restrikcija zahteva na osnovu ovlašćenja
- ✓ Nedozvoliti prikaz serverskih grešaka napadaču.
- ✓ Korišćene custom error poruke.
- ✓ Validacija pojma pretrage regeksom.
- ✓ Korišćenje AngularJS-a koji ima po defaultu zaštitu od prikaza podataka (output encoding)
- ✓ Validacija dodavanja artikla na klijentskoj strani od strane angulara, na serverskoj provera (SafeHtml) prilikom mapiranja na Java objekat.
- ✓ Korišćenje samo neophodnih komponenti
- ✓ Logovanje svih bitnih akcija
- ✓ Generisanje Anti-Forgery tokena (CSRF)