Open Web Application Security Project (OWASP) - Izveštaj

1. *Injection napad*

U Injection napade spadaju SQL, OS, NoSQL i LDAP napadi, koji se trigeruju kada se nepoverljivi podaci salju interpreteru u okviru komandi ili upita. Podaci poslati od strane napadača mogu da prouzrokuju da interpreter izvrši neželjene komande, kao i da pristupi podacima bez ispravne autorizacije.

Rešenje problema:

Ovi napadi se mogu spreciti validacijom podataka dobijenih od korisnika, što podrazumeva odbijanje sumnjivih podataka. Admin takodje može da postavi kontrole kako bi minimizovao količinu informacija koja može biti izložena prilikom napada.

Potrebno je koristiti API security koji podrazumeva implementaciju REST API-ja, zajedno sa HTTPS URL protokolom (Hyper Text Transfer Protocol Secure),uvodjenjem SSL sertifikata.

Korišćenje Hibernate validatora i parametrizovanih upita uz upotrebu najnovijih verzija biblioteka za OR mapiranje.

Upotreba White list input validation, gde se ne dozvoljava korisniku da sam formira SQL upit. Za svaki korisnikov unos moramo imati regularni izraz koji će proveriti da li je taj unos ispravan.

2. *Broken Authentification*

Web aplikacije sa slabom autentifikacijom su laka meta napadačima, zato što su podložne kompromitovanju lozinki, tokena sesije i ključeva.

Rešenje problema:

Korišćenje algoritama za kriptovanje lozinki koje moraju da imaju bar 10 karaktera. Omogućena je zamena i reset lozinke.

Upotreba REST arhitekture koja je stateless.

Cela aplikacija se sprovodi preko HTTPS protokola,uvodjenjem SSL sertifikata.

Generisanje jedinstvenog tokena za svakog ulogovanog korisnika u cilju praćenja sesije. Token ističe nakon odredjenog vremena. Kada se korisnik izloguje, sesija se invalidira.

3. *Sensitive Data Exposure*

Mnoge web aplikacija i API-ji ne štite na pravi način osetljive podatke, poput onih vezanih za finansije ili zdravstvo. Napadači mogu da ukradu ili modifikuju takve slabo zaštićene podatke. Takvi podaci zahtevaju posebnu zaštitu prilikom prenosa. Popularan metod za kradju osetljivih podataka je man-in-the-middle attack.

Rešavanje problema:

Komunikacija izmedju servera i klijenta se odvija preko HTTPS protokola, uvodjenjem SSL sertifikata.

Upotreba enkripcije za lozinke (BCrypt).

Upotrebom ACL-a se može dodatno ograničiti pristup korisniku za read/write fajlova, npr. log-fajlova.

4. *XML External Entities (XEE)*

Ovo je napad na web aplikacije koji parsira XML unos, u kome se nalaze podaci o entitetu. Najbolji način da se spreči ovakav napad je korišćenjem manje složenih tipova podataka kao što je JSON ili izbegavanjem serijalizacije osetljivih podataka.

Rešavanje problema:

Za razmenu poruka preko SOAP-a se koristi verzija 1.2 koja je sigurnija po pitanju XXE napada.

Upotreba JSON-a za prenos podataka.

5. *Broken Access Control*

Kontrola pristupa podrazumeva ograničavanje pristupa stranicama i sekcijama korisnicima u zavisnosti od njihovih potreba. Primeri ovih napada su pristup neautorizovanim funkcionalnostima.

Rešenje problema:

RBAC model - Korišćene su permisije kako bi se obezbedilo da odredjeni korisnik može da pristupi samo metodama koje su mu dozvoljene.Svaki tip korisnika poseduje sopstvene privilegije.

Upotreba tokena koji ograničava korisniku pristup podacima koji nisu vezani za njega.

Korišćenje logger-a koji obezbedjuje praćenje stanja aplikacije.

6. *Security Misconfiguration*

Pogrešna konfiguracija je najčešća ranjivost na listi i često je rezultat korišćenja zadatih konfiguracija ili prikazivanja prekomerno iscrpnih grešaka. Na primer, ukoliko aplikacija previše opisno pokazuje korisnikove greške, to može da otkrije ranjivost u aplikaciji. Ovo se može ublažiti uklanjanjem neiskorišćenih funkcija u kodu i uopštenijim ispisom grešaka.

Rešenje problema:

U slučaju greške, ne prikazivati previše informacija. Stoga su implementirane globalne funkcije za ecxeption handler, što obezbedjuje da ukoliko se desi greška, korisnik će moći da vidi samo poruku da je doslo do greške a ne i samu grešku koja se dogodila.

Implementirana je poslednja proverena verzija TLS-a 1.2 koja se smatra dovoljno bezbednom. Redovno je vršeno proveravanje da li je konfiguracija i dalje važeća i da li je bezbedna.

7. *Cross-Site Scripting*

Ovaj vid ranjivosti nastaje kada web aplikacije omoguće korisnicima da dodaju prilagodjeni URL u adresu ili na web lokaciju koju će videti drugi korisnici. Ovo se može iskoristiti za pokretanje zlonamernog JavaScript koda na pregledaču žrtve. Strategije za ublažavanje ove ranjivosti su izbegavanje nepouzdanih HTTP zahteva kao i provera sadržaja koji je kreirao korisnik. Postoje tri vrste XSS tipova, Stored, Reflected i DOM-Based. Stored XSS se realizuje kada se maliciozni podaci sačuvaju na server. Reflected XSS se dogadja ukoliko aplikacija ili API uključuje nevalidni i neograničeni korisnični unos kao deo HTML izlaza. DOM-Based XSS ranjivost nastaje kada JavaScript frejmvorci i API-ji dinamički uključuju podatke koje kontroliše napadač.

Rešenje problema:

Korišćen je Angular framework, koji vrši zaštitu aplikacije od XSS napada preskakanjem specijalnih karaktera pre nego što browser obradi podatke. Implementirana je i validacija korisničkog unosa i na frontend-u i na backend-u.

8. *Insecure Deserialization*

Nebezbedna deserijalizacija često može da dovede do toga da se kod izvršava preko klase koje se menjaju tokom deserijalizacije. Napad može da se izvrši tako što napadač deserijalizuje objekat uspešno, onda ga modifikuje tako što da sebi rolu admina, pa ga serijalizuje opet. Ovaj skup akcija može da dovede do kompromitovanja cele aplikacije.

Rešenje problema:

Jedini siguran način za zaštitu od nesigurne deserijalizacije je zabrana deserijalizacije podataka od nepoverljivih izvora. Zapis u log fajlove izuzetaka i gresaka prilikom deserijalizacije, npr kada dolazni tip nije očekivani tip.

9. *Using Components With Known Vulnerabilitie*

Komponente, kao što su biblioteke, framevorci i drugi softverski moduli imaju iste privilegije kao aplikacija. Ove komponente mogu da sadrže ranjivosti, koje ako se eksploatišu mogu da prouzrokuju ozbiljan gubitak podataka. Aplikacije i API-ji koji koriste komponente sa ranjivostima mogu da oslabe odbranu aplikacije i da omoguće razne napade.

Rešenje problema:

Uklanjanje svih nekorišćenih dependenci-ja. Upotreba komponeneti samo sa proverenih izvora i koje su up to date. Rešiti se komponenti koje se aktivno ne održavaju.

10. *Insufficient Logging And Monitoring*

Nedovoljno evidentiranje i praćenje u kombinaciji sa neposostojećom ili neefektivnom integracijom sa reakcijom na incident omogućuju napadačima dalje napade na sistem. Većina istraživanja pokazuju da je vreme potrebno da se detektuje upad preko 200 dana.

Rešenje problema:

U log fajlovima implementiranimu u monolitnoj aplikaciji i u miktoservisima se vrši zapis svega što desi u aplikaciji.