SQL INJECTION

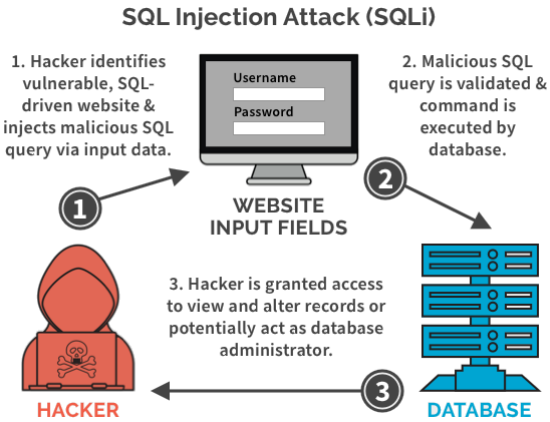
Ajdin Hukara, Fakultet informacijskih tehnologija, Mostar

Sadržaj – SQL injection (u daljnjem tekstu SQLi) je tehnika umetanja malicioznog koda unutar SQL izraza, a samim tim i jedan od najrasprostranjenijih propusta u Web aplikacijama. Ovaj napad se koristi u cilju krae, izmjene ili uništavanja osjetljivih podataka iz baza podataka, ili u cilju neautorizovanog pristupa sistemu i izvršavanja administratorskih funkcija. U nastavku su opisane vrste i najčešće korištene metode ovog napada, prevencije i zaštita, metode testiranja na ranjivost aplikacije, kao i korištenje Regex izraza za otkrivanje SQL injection napada.

Ključne riječi : sigurnost, SQL, baze podataka, web aplikacije, regular expressions (Regex)

1 Uvod

SQL je skraćenica od Structured Query Language i predstavlja jezik za upravljanje bazama podataka. Skoro svi moderni sistemi za upravljanje bazama podataka kao što su MySQL, MS SQL Server, Microsoft Access i Oracle koriste SQL jezik. Upiti su glavni mehanizam SQL injection napada od kojih se razlikuju DDL (Data Definition Language), DML (Data Manipulation Language) i DCL (Data Control Language).



Slika 1 Implementacija SQLi

SQLi se koristi u sistemima koji se oslanjaju na baze podataka, gdje se uz ulazne podatke ubacuje maliciozni kod. Ovaj napad se prvi put spominje 1998. godine u članku koji je napisao Jeff Forristal, trenutno vlasnik kompanije Bluebox Security " . On je tada istakao da može potpuno promijeniti način na koji SQL jezik funkcionira. Open Web Application Security Project (OWASP) je 2007. i 2010. godine objavio da se SQLi nalazi u top 10 napada u svijetu. 2013. godine, SQLi je dostigao prvo mjesto na toj listi. Takoer, istraživanje napravljeno od strane Garther Group " je došlo do saznanja da su od 300 testiranih web stranica čak 97% njih ranjive na SQL injection.
O

vaj rad prezentuje različite mehanizme odbrane od SQL injection napada, metode testiranja sistema na ranjivost i metode za sprečavanje napada. Na slici 1. prikazan je način implementacije SQLi.

2 Kategorije SQL injection napada

Posljedice SQLi napada mogu biti različite, od prikupljanja ili modifikovanja osjetljivih podataka, pa do neautorizovanog logiranja na sistem koji napadaču može dozvoliti izvršavanje administratorskih funkcija. Postoje 3 glavne kategorije SQLi napada, a to su:

1. Union-based SQLi
2. Error-based SQLi
3. Blind SQLi

Union-based SQLi je tip SQL injection napada koji koristi UNION izraz. Ovaj izraz omogućava spajanje dva ili više SELECT izraza, u kojem je jedini uslov da svi SELECT izrazi imaju isti broj kolona. Na ovaj način, napadač može da iskoristi podatke iz drugih tabela, ukoliko su nazivi tabela u bazi poznati.

Error-based SQLi oslanja se na poruke greške koje nam pruža server baze podataka, u cilju dobijanja informacija o strukturi baze podataka. U nekim slučajevima, error-based SQLi je dovoljan za napadača da preuzme kontrolu nad bazom podataka. Obzirom da su poruke greške jako korisne tokom

razvoja web aplikacija, one bi trebale biti isključene u produkciji. Takoer, dobra praksa je poruke greške spašavati u log fajl.

Implementacija **blind SQLi** napada može trajati duže za napadača, ali može biti jako opasna. U ovoj vrsti SQLi, napadač ne može vidjeti rezultate umetnutog koda, ali se otvara mogućnost rekonstrukcije baze podataka (izmjena podataka u tabelama, dodavanje novih tabela, brisanje podataka, izmjena relacija i sl.)

3 Najčešće metode SQL injection napada

3.1 Tautologija

Cilj napada: Zaobilaženje autentifikacije i izvlačenje podataka

Rezultati upita u SQL bazi podataka bazirani su na tautologiji, i poznato je da upiti vraćaju vrijednost TRUE ili FALSE. Cilj ove metode napada jeste da SQL upit vrati vrijednost TRUE kako bi napadač dobio informacije.

Primjer: Ukoliko se u bazi podataka nalazi korisnik 'user01' sa lozinkom 'password01', nakon unosa vrijednosti u polja 'username' i 'password' u login formi, sljedeći upit se izvršava:
SELECT username, password FROM users WHERE username='user01' AND password='password01';

U prethodnom upitu, izraz poslije komande WHERE vraća vrijednost TRUE, nakon čega se odobrava pristup aplikaciji (login). Umetanjem koda koji će vratiti vrijednost TRUE bez obzira na polja username i password, dobija se pristup aplikaciji. Ako napadač u polje 'username' unese vrijednost u obliku ' OR 1=1; –", sljedeći upit će se generisati:
SELECT username, password FROM users WHERE username=' ' OR 1=1; –' AND password=' ';

Kod koji je ubačen u polje 'username' pretvara čitav WHERE izraz u vrijednost TRUE, obzirom da je 1=1. Nakon izvršavanja upita, web aplikacija zaključuje da je autentifikacija korisnika uspješna, i napadač dobija pristup web aplikaciji kao korisnik ili administrator.

3.2 Union upiti

Korištenjem UNION upita, napadač može dohvatiti podatke i iz drugih tabela, a ne samo iz one iz koje je programer napisao kod. Za login formu, obično je to tabela sa korisnicima. U SQL jeziku, UNION izraz spaja dvije ili više tabela u jednu tabelu. Jedino ograničenje je da broj kolona u svim tabelama bude isti. Naslovi kolona bit će jednaki naslovima tabele iz prvog SELECT izraza.

Primjer: U polje za login, napadač unosi vrijednost ' UNION SELECT * FROM tabela2; –", nakon čega se sljedeći kod izvršava nad bazom:
SELECT username, password FROM users WHERE username=' ' UNION SELECT kolona1, kolona2 FROM tabela2; –' AND password=' ';

Sa pretpostavkom da ne postoji korisnik sa praznim korisničkim imenom, UNION upit će dohvatiti podatke iz druge tabele (tabela2). Važno je naglasiti da u prethodnom primjeru tabela2 mora imati dvije kolone (kolona1 i kolona2), jer su u prvom SELECT upitu navedene dvije kolone (username i password).

3.3 Uskladištene procedure

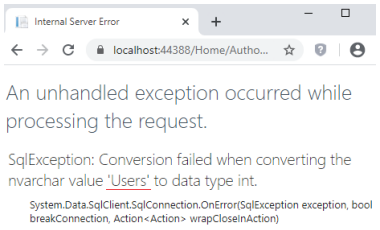
Većina DBMS danas koristi uskladištene procedure koje povećavaju funkcionalnosti baze podataka i dozvoljavaju interakciju sa operativnim sistemom. U ovoj metodi SQLi, napadači pokušavaju izvršavati već kreirane uskladištene procedure. Iako uskladištene procedure mogu biti način da se zaštiti od SQLi, pogrešnom implementacijom uskladištenih procedura, web aplikacija može ostati ranjiva na SQLi napad. Jednom kada se napadač detaljno upozna sa bazom podataka, on može izvršavati procedure pa čak i na nivou operativnog sistema.

3.4 Korištenje sistemskih tabela

Cilj napada: Prikupljanje informacija o strukturi baze podataka i identifikacija ranjivih parametara
Prikupljanje informacija je prva i osnovna metoda svakog sigurnosnog napada. Napadač ne može iskoristiti podatke ukoliko ne poznaje strukturu baze podataka, nazive tabela ili relacije. Prilikom razvoja aplikacije, poruke o greškama (error messages) mogu biti korisne programerima. One su i jedan od načina putem kojih se napadač može detaljno upoznati sa bazom podataka i njenom strukturom.

Ukratko, svaki DBMS (Database Management System) sadrži korisničke i sistemske tabele. Sistemske tabele izmeu ostalog čuvaju i podatke o korisničkim tabelama. Na taj način napadač može doći do naziva tabela u bazi podataka. Tabela sysobjects" čuva podatke o korisničkim tabelama.
Primjer: Da bi napadač dobio informacije o nazivu tabele, dovoljno je da u polje 'username' unese sljedeći kod: ' UNION SELECT name,id,xtype FROM sysobjects WHERE xtype='u';–", a u polje 'password' bilo koju vrijednost. Nakon izvršavanja, dobija se poruka prikazana na slici 2.

:



Slika 2 Primjer opisa greške

Na slici 2. prikazan je Server Error koji ukazuje na to da je potrebno izvršiti konverziju nad tabelom Users “ , te da opis greške itekako može pomoći napadaču u prikupljanju informacija.

Kolona xtype “ u tabeli sysobjects je tipa nvarchar(2) i predstavlja tip tabele, gdje vrijednost u “ označava User Tables (Korisničke tabele). Ukoliko napadač ispravi prethodnu grešku, i uskladi tipove podataka ili uradi konverziju, dobit će listu svih tabela u bazi podataka, nakon čega se otvara mogućnost izvršavanja upita i povlačenja podataka iz tabela.

4 Zaštita od SQL injection napada

SQL injection napadi su nažalost vrlo česti i to iz dva razloga: Značajna rasprostranjenost ranjivosti SQL baza podataka i atraktivnost cilja (baza podataka obično sadrži sve zanimljive ili kritične podatke aplikacije). U današnje vrijeme može se reći da je na neki način sramotno ostati izložen SQLi napadu, obzirom da je poprilično lako zaštititi se od istog. SQL injection je baziran na dinamičkim upitima (slika 3.) koje pišu programeri, tako da je osnova zaštite od SQLi upravo izbjegavanje dinamičkih upita i sprečavanje zlonamjernog korisničkog unosa. Dinamički upiti sastoje se od spajanja SQL izraza i korisničkog unosa (string concatenation). U nastavku slijede tehnike zaštite sistema od SQL injection napada.

```
var cmd = new SqlCommand(
    @" SELECT username, password
       FROM users WHERE username='" + user_input +
    "' AND password='" + password_input + "';");
```

Slika 3 Dinamički upit

4.1 Korištenje parametrizovanih upita

Parametrizovani upiti su prva i osnovna stvar na koju se programeri moraju osloniti. Osim toga što sprečavaju SQL injection napad, parametrizovani upiti omogućavaju čišći “ i čitljiviji kod. Jednostavni su za pisanje, i ne razlikuju se mnogo od dinamičkih upita. Parametrizovani upiti prvo zahtjevaju definiranje SQL koda, a zatim se svaki potrebni parametar proslijedi tom upitu. Ovaj stil kodiranja omogućava bazi podataka da razlikuje SQL kod i podatke koje korisnik unosi. Primjer: Ukoliko bi napadač u polje username“ unio vrijednost ’ OR 1=1; –“, parametrizovani upit ne bi bio ranjiv i umjesto modifikovanog koda pretražio bi korisničko ime koje doslovno odgovara cijelom unesenom nizu karaktera. Na slici 4. prikazan je primjer korištenja parametrizovanog upita u programskom jeziku C. Svi moderni programski jezici podržavaju parametrizovane upite.

```
const string query = @"SELECT account_balance
                FROM user_data
                WHERE user_name = ?";

var command = new SqlCommand(query);
command.Parameters.Add(
    new SqlParameter("user_name", username_input));
```

Slika 4 Prametrizovani upit

4.2 Korištenje uskladištenih procedura

Uskladištene procedure nisu uvijek sigurne od SQLi napada, ali ukoliko ih pravilno iskoristimo, mogu se ponašati slično kao i parametrizovani upiti. Osnovna razlika izmeu uskladištenih procedura i parametrizovanih upita je ta što su uskladištene procedure definisane i pohranjene u samoj bazi, a zatim se pozivaju iz aplikacije.

4.3 Validacija korisničkog unosa

Validacija, provjera ili pročišćavanje “ korisničkog unosa može biti jako efikasna metoda odbrane od SQLi napada. Meutim, pored velikog broja jako dobro obučenih napadača, uvijek postoji šansa da korisnički unos proe proces validacije iako sadrži maliciozan kod. Validaciju u svakom slučaju treba koristiti, ali za odbranu od SQL injection napada, korištenje parametrizovanih upita je neizbježno i mnogo sigurnije.

5 Metode testiranja

Bez obzira koliko programeri nastojali da implementiraju zaštitu od SQLi napada i kreiraju što sigurniji sistem, jako često se dešavaju nenamjerni propusti. Kako bi se propusti eliminisali, potrebno je izvršiti testiranje sistema.

Statička analiza: Predstavlja skeniranje koda odreenim alatima (code-scanning). Ovi alati pretražuju moguće greške u kodu i upozoravaju na mane i ranjivosti. Nemaju mogućnost praćenja stanja u aplikaciji, ali takoer nemaju problema u pronalasku grešaka poput SQL naredbi bez parametrizovanih upita. Postoje i besplatni alati za statičko skeniranje koda, kao što je alat FindBugs “ koji je namjenjen za pronalazak ranjivosti u Java aplikacijama.

Penetracijsko testiranje: Testeri jako često koriste alate za automatizovano testiranje, meutim, alat neće uvijek pronaći sve ranjivosti. Nekad je potrebno angažovati profesionalnog testera koji će ručno otkriti daleko više informacija od većine automatizovanih alata. Česta tehnika testiranja sistema na ranjivost jeste fuzz“ testiranje aplikacije, koja je bazirana na namjernom upisivanju pogrešnih podataka, čime se izazivaju greške koje je potrebno otkloniti.

6 Sprečavanje SQLi napada u realnom vremenu (Regex)

Regex (Regular Expressions) predstavljaju moćan alat za pretragu i testiranje tekstualnih podataka. Regex se koriste u procesu validacije, i mogu biti korisni u detekciji malicioznog koda u web aplikacijama. Primjer: Polje username“ nema potrebe da na prvom mjestu sadrži jednostruke navodnike. Takoer, svaki dio teksta poslije znaka ; “ (tačka-zarez), mora biti ignorisan. Dvostruke crtice — koje označavaju komentare u SQL jeziku, moraju biti detektovane. Sve navedene provjere mogu biti izvršene korištenjem Regex izraza.

7 Zaključak

Regex (Regular Expressions) predstavljaju moćan alat za pretragu i testiranje tekstualnih podataka. Regex se koriste u procesu validacije, i mogu biti korisni u detekciji malicioznog koda u web aplikacijama. Primjer: Polje username“ nema potrebe da na prvom mjestu sadrži jednostruke navodnike. Takoer, svaki dio teksta poslije znaka ; “ (tačka-zarez), mora biti ignorisan. Dvostruke crtice — koje označavaju komentare u SQL jeziku, moraju biti detektovane. Sve navedene provjere mogu biti izvršene korištenjem Regex izraza.

Literatura

[1] https://www.imperva.com/blog/five-sql-best-practices/
[2] https://mistscholars.wordpress.com/2014/08/07/sql-injection/
[3] http://index-of.es/Failed-attack-techniques/SQL%20Injection.pdf
[4] https://pt.slideshare.net/ashish20012/ppt-on-sql-injection/9
[5] http://www.sqlinjection.net/category/advanced/tools/
[6] https://www.cs.montana.edu/courses/csci476/topics/sql_injection.pdf
[7] https://research.cs.wisc.edu/mist/SoftwareSecurityCourse/Chapters/3_8_1-SQL-Injections.pdf