ZAŠTITA OD NEOVLAŠTENOG PRISTUPA PODACIMA

1. Zaštita baze podataka

Svaka baza podataka mora biti zaštićena od nedopuštenih radnji u svrhu očuvanja sigurnosti i integriteta baze. Osiguravanje sigurnosti baze podataka odnosi se na osiguravanje ovlaštenosti korisnika za akcije koje pokušavaju izvesti. S druge strane, osiguravanje integriteta odnosi se na osiguravanje da su akcije koje korisnici pokušavaju izvesti ispravne. O oba slučaja moraju biti definirana pravila koja korisnici ne smiju narušiti.

Glavni vid zaštite je postavljanje ograničenja fizičkog pristupa do računala na kojem se podaci nalaze. Osim fizičke postoji i softverski vid zaštite koji se ugrađuje u DBMS. Jedan od načina zaštite je dodjeljivanje ovlaštenja definiranim korisnicima za određene radnje. Oni će podacima moći pristupati isključivo koristeći odgovarajuću identifikaciju, odnosno kombinaciju korisničkog imena i lozinke.

Gotovo svi DBMS-ovi imaju mogućnost rada na Internetu. Bazama se može pristupati s drugih računala koja se nalaze u drugim gradovima ili državama. Navedeno nije samo prednost već je ujedno i mana jer povlači pitanje dodatne zaštite od neovlaštenog pristupa podacima. Komunikacija na Internetu bazirana je na IP protokolu gdje računala međusobno komuniciraju preko paketa čija je struktura točno određena. Svaki IP paket dodjeljuje se jednoj vrsti Internet usluge odnosno komunikacije, a identificira ga se preko broja odnosno porta. Svaka usluga na Internetu ima svoj broj porta (FTP 20/21, SSH 22, Telnet 23, Mail 25, WWW 80, …). Baze podataka također imaju svoje portove za komunikaciju. Tako je port MySQL baze port 3306.

Kako zaštiti bazu podataka na Internetu ?

- Isključiti mrežne mogućnosti DBMS-a
- Dopustiti pristup DBMS-u samo lokalnim programima (localhost)
- Dopustiti pristup DBMS-u samo računalima unutar lokalne mreže
- Dopustiti nekima, ali uz identifikaciju (korisnik/lozinka)
- Koristiti šifriranu komunikaciju (ssl/ssh, dvostruki ključevi,. . .)

U nastavku će biti opisani postupci stvaranja korisnika, dodjeljivanja lozinke te dodjeljivanja ovlaštenja korisnicima za strogo određene radnje u bazi podataka.

2. Stvaranje korisnika

Svaki je korisnik u bazi podataka definiran kombinacijom korisničkog imena te oznake lokacije (adrese klijenta) s koje može pristupati podatcima.

Oblik identifikacijske oznake klijenta:

```
'korisnickoIme'@'adresaKlijenta'
```

Oznaka adrese klijenta može biti:

- IP adresa - primjerice: 31.147.108.24
- Ime poslužitelja – primjerice: localhost
- Puni naziv domene - primjerice: somehost.example.com
- Oznaka % koja označava bilo koju adresu ili skup adresa – primjerice: '%', '34.190.68.%', '%.example.com'

Kako bi se dva korisnika razlikovala, nužno je da imaju različite obje oznake. To povlači da će korisnik 'admin'@'192.168.5.10' biti različit od korisnika 'admin'@'192.168.5.%'. Ako je korisnik definiran na način da mu nije navedena oznaka lokacije s koje se smije spajati (primjerice 'admin'), tada se pretpostavlja da bazi može pristupati s bilo koje lokacije (dakle 'admin'@'%').

Osnovna sintaksa:

```
CREATE USER imeKorisnika [IDENTIFIED BY [PASSWORD] 'lozinka']
        [,imeKorisnika2 [IDENTIFIED BY [PASSWORD] 'lozinka2']] …
```

Svaka promjena stvaranja i brisanja korisnika te dodjeljivanja i brisanja dozvola određenom korisniku zapisana je u DBMS-u u posebnoj bazi podataka naziva *mysql*. Stvaranje novog korisnika rezultirat će dodavanjem nove n-torke u tablici *mysql.user*. Dohvatom podataka iz te tablice moguće je provjeriti specifikacije određenog korisnika.

Primjer:

Potrebno je stvoriti novog korisnika *ihorvat* sa lokalnog poslužitelja te mu dodijeliti lozinku *1234*.

```
CREATE USER 'ihorvat'@'localhost' IDENTIFIED BY '1234';
```

Izvršavanje gornje CREATE USER naredbe rezultirat će sljedećom n-torkom u tablici mysql.user:

Host	User	Password	Select_priv	Insert_priv	Update_priv
localhost	ihorvat	*A4B6157319038724E3560894F7F932C8886EBFCF	N	N	N
			N	N	N

Prema gornjoj sintaksi moguće je stvoriti korisnika te mu opcionalno odmah dodijeliti i lozinku. Ako mu se ne dodijeli lozinka, tada će vrijednost atributa *password* ostati prazno. U sigurnosnom pogledu to je nedostatak u sustavu, te odgovornost ostaje na administratoru da prilikom kreiranja korisnika odnosno najkasnije prilikom dodjeljivanja dozvola, dodijeli korisniku i odgovarajuću lozinku. Ako se lozinka i ne dodijeli u početku, moguće ju je naknadno dodijeliti naredbom SET PASSWORD.

Osnovna sintaksa:

```
SET PASSWORD [FOR imeKorisnika] = PASSWORD('lozinka')
```

U osnovnoj verziji uporabe naredbe SET PASSWORD, lozinka će biti postavljena trenutno spojenom korisniku.

Primjer:

Potrebno je promijeniti lozinku trenutno logiranom korisniku na vrijednost *root*.

```
SET PASSWORD = PASSWORD('root');
```

Lozinke nikada nisu u navedenoj tablici u bazi *mysql* pohranjene u izvornom obliku već su kriptirane. MySQL kriptira lozinke prema vlastitom algoritmu, pri čemu je za svaku predviđen 41 byte. U gornjem kodu korištena je funkcija PASSWORD() koja se brine za kriptiranje lozinke. Kada se lozinka ne bi pohranila u kriptiranom obliku, došlo bi do pogreške prilikom pokušaja spajanja dotičnog korisnika na DBMS jer MySQL očekuje da je lozinka zapisana kriptirano te čitajući iz kolone *user.password* uvijek prvo provodi dekripciju.

Osim promjene lozinke trenutno spojenom korisniku, moguće je promijeniti lozinku nekom zadanom korisniku.

Primjer:

Promijeniti lozinku prethodno stvorenom korisniku *ihorvat* (koji se spaja s lokalnog poslužitelja) na vrijednost *novi1234*.

```
SET PASSWORD FOR 'ihorvat'@'localhost' = PASSWORD('novi1234');
```

Treći je način postavljanja lozinke (osim …IDENTIFIET BY klauzule i SET PASSEWORD naredbe) direktnim ažuriranjem retka u tablici *mysql.user*.

Primjer:

Promijeniti lozinku prethodno stvorenom korisniku *ihorvat* (koji se spaja s lokalnog poslužitelja) na vrijednost *novaLozinka*. Promjenu lozinke potrebno je izvršiti ažuriranjem odgovarajuće tablice u bazi mysql.

```
UPDATE mysql.user
        SET user.password=PASSWORD('novaLozinka')
        WHERE user.user='ihorvat' AND user.host='localhost';
FLUSH PRIVILEGES;
```

Određene promjene vezane uz upravljanje korisnicima i njihovim dozvolama bit će vidljive tek po ponovnom pokretanju baze podataka odnosno nakon izvođenja naredbe FLUSH PRIVILEGES.

3. Dodjeljivanje dozvola

Dozvole korisnicima za rad s podacima u MySQL bazi moguće je dodijeliti na dva načina. Prvi i preporučljiv način je koristeći naredbu GRANT. Sintaksa je čista i jednostavna, te se na brz način može dodijeliti odgovarajuća dozvola nekom korisniku. Kao što je moguće stvoriti novog korisnika izmjenom odnosno unosom podataka u tablici *mysql.user*, moguće je i dodijeliti odgovarajuće

dozvole manipulacijom podataka u odgovarajućim tablicama u bazi *mysql*. To je nešto složeniji način jer je potrebno poznavati strukturu tablica u kojima su dozvole pohranjene. Ta se metoda preporuča samo u delikatnijim zahtjevima i situacijama.

Osnovna sintaksa:

```
GRANT
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO user_specification [, user_specification] ...
    [REQUIRE {NONE | ssl_option [[AND] ssl_option] ...}]
    [WITH with_option ...]
```

U nastavku su navedene opcije za dane dijelove naredbe:

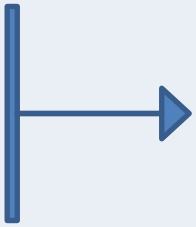
```
object_type:
    TABLE
  | FUNCTION
  | PROCEDURE

priv_level:
    *                          cijeli sustav
  | *.*                        sve baze u sustavu
  | db_name.*                  sve tablice u bazi db_name
  | db_name.tbl_name           tablica tbl_name u bazi db_name
  | tbl_name                   tablica tbl_name
  | db_name.routine_name       pohranjeni zadatak routine_name u bazi db_name

user_specification:
    user
    [
        IDENTIFIED BY [PASSWORD] 'password'
      | IDENTIFIED WITH auth_plugin [AS 'auth_string']
    ]

ssl_option:
    SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'

with_option:
    GRANT OPTION
  | MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
```



Primjer:

Dodijeliti sve dozvole nad svim podacima u sustavu korisniku *admin* sa svih adresa.

```
GRANT ALL ON *.* TO 'admin'@'%';
```

Primjer:

Dodijeliti sve dozvole nad svim podacima u sustavu korisniku *admin* sa svih adresa. Neka navedeni korisnik ima mogućnost dodjeljivanja dozvola drugim korisnicima

```
GRANT ALL ON *.* TO 'admin'@'%' WITH GRANT OPTION;
```

Primjer:

Dodijeliti dozvole čitanja, dodavanja i ažuriranja podataka u tablici *nalog* (baza *autoradionica*) korisniku *korisnik25* sa bilo kojeg poslužitelja u domeni *servis.com*. Korisnik *korisnik25* se spaja koristeći lozinku *mojaLozinka*. Neka *korisnik25* može ostvariti najviše 25 konekcija u satu.

```
GRANT SELECT, INSERT, UPDATE
        ON autoradionica.nalog
        TO 'korisnik25'@'%.servis.com'
        IDENTIFIED BY 'mojaLozinka'
        WITH MAX_CONNECTIONS_PER_HOUR 25;
```

Stanje u tablici mysql.tables_priv nakon izvođenja gornjeg upita:

Host	Db	User	Tab...	Grantor	▲ T	Table_priv
%.servis.com	autoradionica	korisnik25	nalog	root@localhost	:21	Select,Insert,Update,Delete,Create,Drop,References,Ind

Ako se naredbom grant pokušavaju dodijeliti dozvole korisniku koji ne postoji, korisnik će automatski biti kreiran.

Razine dodjeljivanja dozvola

i. Dozvole pristupa cijelom sustavu

- ✓ Evidentirane su u tablici *mysql.user*

Primjeri:

```
GRANT ALL ON *.* TO 'imeKorisnika'@'nazivDomene';
GRANT SELECT, INSERT, DELETE ON *.* TO 'imeKorisnika'@'nazivDomene';
```

ii. Dozvole pristupa bazi podataka

- ✓ Evidentirane su u tablici *mysql.db*

Primjeri:

```
GRANT ALL ON imeBaze.* TO 'imeKorisnika'@'nazivDomene';
GRANT SELECT, INSERT ON imeBaze.* TO 'imeKorisnika'@'nazivDomene';
```

iii. Dozvole pristupa tablici

- ✓ Evidentirane su u tablici *mysql.tables_priv*

Primjeri:

```
GRANT ALL ON imeBaze.mytbl TO 'imeKorisnika'@'nazivDomene';
GRANT SELECT, UPDATE, DELETE ON imeBaze.mytbl TO 'imeKorisnika'@'nazivDomene';
```

iv. Dozvole pristupa atributu

- ✓ Evidentirane su u tablici *mysql.columns_priv*

Primjer:

```
GRANT SELECT (col1), UPDATE (col1,col2) ON imeBaze.mytbl TO
'imeKorisnika'@'nazivDomene';
```

v. Dozvole pristupa pohranjenom zadatku

- ✓ Evidentirane su u tablici *mysql.procs_priv*

Primjeri:

```
GRANT CREATE ROUTINE ON imeBaze.* TO 'imeKorisnika'@'nazivDomene';
GRANT EXECUTE ON PROCEDURE imeBaze.imeProcedure TO 'imeKorisnika'@'nazivDomene';
```

4. Ukidanje dodijeljenih dozvola

Kao i kod dodjeljivanja, dozvole se mogu ukinuti na dva načina. Prvi je način naredbom REVOKE, a drugi ažuriranjem odgovarajućih tablica u bazi MySQL. Sve što je oko izbora načina navedeno za dodjeljivanje dozvola, vrijedi i za ukidanje dozvola.

Osnovna sintaksa (1):

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] ...
```

Osnovna sintaksa (2):

```
REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM user [, user] ...
```

Primjer:

Ukinuti dozvole ažuriranja podataka u tablici *nalog* (baza *autoradionica*) korisniku *korisnik25* sa bilo kojeg poslužitelja u domeni *servis.com*.

```
REVOKE UPDATE ON autoradionica.nalog FROM 'korisnik25'@'%.servis.com';
```

Kao i kod dodjeljivanja dozvola, određene promjene bit će vidljive tek po ponovnom pokretanju baze podataka odnosno nakon izvođenja naredbe FLUSH PRIVILEGES.

Pregled postojećih dozvola

Ako je određenom korisniku potrebno ukinuti određene dozvole, a nismo sigurni koje trenutno ima dodijeljene, potrebno je prvo s naredbom SHOW GRANTS provjeriti trenutno stanje korisnikovih dozvola.

Osnovna sintaksa:

```
SHOW GRANTS FOR imeKorisnika
```

Primjer:

Provjeriti dozvole za ranije stvorenog korisnika *ihorvat*.

```
SHOW GRANTS FOR 'ihorvat'@'localhost';
```

Obzirom da je korisnik samo bio stvoren naredbom CREATE USER a nisu mu kasnije dodjeljivane dozvole, navedeni korisnik ima samo dozvolu USAGE (korištenje) nad sustavom.

```
Grants for ihorvat@localhost
GRANT USAGE ON *.* TO 'ihorvat'@'localhost' IDENTIFIED BY PASSWORD '*
```

Promjena korisničkog imena

Osnovna sintaksa:

```
RENAME USER stariKorisnik TO noviKorisnik
        [,stariKorisnik2 TO noviKorisnik2] …
```

Kod radnje promjene korisničkog imena treba biti oprezan jer će MySQL ažurirati samo podatke u tablici *mysql.user,* odnosno neće promijeniti ostale zapise koji se odnose na korisnikove dozvole. Posljedica toga je da stare dozvole ostaju aktivne bez obzira što stari korisnik više ne postoji. Postoji opasnost da kada bi se u budućnosti stvorio novi korisnik istog imena kao što je bio stari, odjednom bi imao dodijeljene (a potencijalno i opasne) dozvole.

Iz tog se razloga preporuča prvo koristiti naredbu REVOKE pa nakon toga GRANT, odnosno trajno obrisati starog korisnika ako više nije potreban.

Brisanje korisnika

Osnovna sintaksa:

```
DROP USER imeKorisnika;
```