Sistemi za upravljanje bazama podataka

Sigurnost baza podataka – SQL Server

Mentor:
prof. dr Aleksandar Stanimirović

Student:
Marija Milošević 1045

Sadržaj

1. Uvod

Poslovanje modernih kompanija sve je više zavisno od IT aplikacija i sistema, a budući da se danas za skoro sve koristi internet, javlja se sve više bezbednosnih pretnji. Zato je neophodno implementirati odgovarajuću zaštitu aplikacija i sistema, što podrazumeva i obraćanje pažnje na sigurnost baza podataka. Pojam sigurnosti baze podataka odnosi se na različite mere koje organizacije sprovode kako bi zaštitile svoje baze od internih i eksternih pretnji. Ovo uključuje zaštitu same baze, podataka koje skladišti, njenog sistema i razne aplikacije koje joj pristupaju. Organizacije moraju da osiguraju baze od namernih napada poput sajber pretnji, ali i od zloupotrebe podataka i baze od strane onih koji joj mogu pristupiti. Fokus je na sprečavanju ugrožavanja tajnosti, integriteta i dostupnosti.

U poslednjih par godina, broj napada je značajno porastao. Studija Ponemon instituta iz 2016. godine pokazuje da organizacije odvajaju veliki deo svog budžeta za mrežnu bezbednost (40%) a samo 19% za bezbednost baza podataka. Uz određenu štetu koju ove pretnje predstavljaju po reputaciju kompanije i njihov skup korisnika, postoji i ogroman broj regulativa i kazni za ovakve napade sa kojima se organizacije moraju izboriti, poput onih u GDPR-u[1], gde su neke dosta skupe. Efikasna sigurnost baze podataka je ključna za održavanje sklada, zaštitu reputacije i zadržavanja korisnika kompanije.

U ovom radu će biti predstavljeni osnovni koncepti koji se tiču sigurnosti baza podataka, nakon čega će se preći na obradu SQL Servera. Biće razmotrene najveće i najčešće pretnje, kao i bezbednosne opcije koje SQL Server nudi. Svaka od navedenih pretnji biće propraćena primerom koji ilustruje na koji se način SQL Server s njom suočava.

[1] General Data Protection Regulation

2. Sigurnost baza podataka

Sigurnost baza podataka tiče se upotrebe širokog spektra bezbednosnih kontrola informacija u cilju zaštite baza podataka (koje potencijalno uključuju podatke, aplikacije ili uskladištene funkcije, sisteme baza podataka, servere i mrežne veze). Neophodno je bazu zaštiti od ugrožavanja:

- Tajnosti – Tajnost se odnosi na upotrebu politika, procedura i modela za kreiranje i održavanje privatnosti i diskrecije informacija i sistema. Kako bi sistem pružio tajnost, on mora osigurati da informacije budu privatne ograničavanjem autorizovanog pristupa tim resursima i mora blokirati neautorizovan pristup resursima;
- Integriteta – Integritet podrazumeva upotrebu politika, procedura i modela za kreiranje i održavanje pouzdanih, konzistentnih i kompletnih informacija i sistema. Integritet se štiti sprečavanjem neautorizovanih, ali i autorizovanih modifikacija (bilo namernih ili slučajnih) koje mogu da izazovu nepouzdanost ili nekonzistentnost u bazi. To je najteža stavka koju ovde treba proceniti, jer podaci ne nedostaju, oni su prisutni ali su modifikovani;
- Dostupnosti – Dostupnost se odnosi na održavanje resursa ne mreži ili u okviru baze dostupnim. Ovo uključuje podatke, aplikacije, druge baze, računare, servere, fajlove, pristup mreži, itd. Za razliku od tajnosti i integriteta, posao ne može funkcionisati bez dostupnosti, jer nemogućnost pristupa kritičnim stavkama dovodi do nemogućnosti izvršenja i najjednostavnijih zadataka.

Sigurnost će uključivati različite tipove i kategorije kontrola: tehničke, fizičke i proceduralne/administrativne.

2.1. Česte pretnje i izazovi

Uprkos skupim alatima koji pomažu zaštiti podataka, ne možemo se odbraniti od zlonamernih napadača bez kompletnog razumevanja svega onoga što predstavlja pretnju. Baze koje su previše restriktivne su podjednako neefikasne kao one kojima se lako pristupa. Baze kojima se lako pristupa mogu da vode do problema sa bezbednošću, integritetom i privatnošću, dok restriktivne baze dovode do frustracije korisnika i neefikasnosti. Da bi se dobio određeni balans, moramo razumeti od čega se štitimo. Postoji zabluda da su najveće pretnje po sredstva one koje su van i pokušavaju da „upadnu“. Međutim, postoji dosta pretnji i unutar mreže.

1. *Hakeri* – iako termin ima negativnu konotaciju, po definiciji on se odnosi na one koji su odlično upućeni u softver i hardver modernih računarskih sistema i bave se istraživanjem i analizom mrežne bezbednosti bez ikakvih namera da izazovu štetu. S druge strane, krekeri (*eng. crackers*) su oni koji upadaju u mreže bez autorizacije u nadi da će uništiti i/ili ukrasti informacije.
2. *Socijalni inženjeri* – oni koriste interakciju kako bi manipulisali ljudima i dobili pristup sistemima, neautorizovanim oblastima i poverljivim informacijama. Najčešći način za

dobijanje pristupa sistemu je jednostavno pitanje osobe za šifru. U ljudskoj je prirodi želja da se pomogne drugom, tako da ovu činjenicu socijalni inženjeri često koriste.

3. *Korisnici računara* – Više od polovine bezbednosnih upada na mreži uključuje njene korisnike. Manjak edukacije ili zanemarivanje politika dva su najveća razloga za to. Okolnosti koje se pružaju za korisničke greške su beskonačne, pa je bitno naučiti što više o korisnicima sistema i edukovati ih kontinualno. Osobe sa slabim poznavanjem rada na računaru takođe mogu izazvati razne probleme. Pored toga, ne treba zanemariti ljute ili nezadovoljne korisnike, koji su već upoznati s radom sistema pa namerno mogu izazvati grešku.
   Lista najčešćih grešaka uključuje:
   a. Loše navike – ostavljanje računara nenadgledanog tokom sastanka, pauze, itd.
   b. Greške vezane za lozinke – biranje lozinki koje se lako pogađaju, zapisivanje lozinki i ostavljanje na vidnom mestu.
   c. Zanemarivanje politika kompanije – poseta neautorizovanih sajtova i skidanje neautorizovanog softvera, uključivanje neautorizovane opreme, poput USB-a i eksternih hard diskova, na računar, prijavljivanje na servis kompanije preko neodobrenih personalnih računara.
   d. Otvaranje nepoznatih mejlova – dovodi do čestih *phishing* napada.
   e. Neprimereno obelodanjivanje podataka.
4. *Administratori baza i mreže* – na njih se retko gleda kao na pretnju, ali se i na administrativnom nivou mogu javiti problemi. Kako ovi timovi kreiraju, održavaju, upravljaju i nadgledaju celu bazu i/ili mrežnu arhitekturu, greške koje mogu da nastanu na ovom nivou skoro sigurno će imati posledice po integritet, dotupnost i pouzdanost mreže. Pri dodavanju, brisanju i menjanju korisnika, njihovih privilegija, ali i podataka, mogu se javiti bezbednosne mane gde ih nije bilo. Zbog toga je bitno vršiti česte revizije.
5. *Internet* - Iako ima dosta prednosti, na ovom polju dovodi do velikih problema i to vezanih za različite stvari: veb strane, veb brauzere (HTTP problemi, SQL injekcije, DNS *poisoning*), mejlovi (*phishing*, zlonamerni prilozi), malver (koji može dovesti do incidenata poput neautorizovanog pristupa, curenja ličnih ili vlasničkih podataka, brisanja ili oštećenja podataka ili programa, prekida ili nemogućnosti autorizovanog pristupa bazi, napada na druge sisteme, neočekivanog pada usluga baze), itd.
6. *Defekti u softveru* – Ovde spadaju mane u dizajnu i greške u kodiranju. Čak 35 posto uspešnih napada koristi neku od ovih grešaka. Mane u dizajnu uključuju odluke donete pri dizajniranju koje kreiraju suštinski nebezbedan sistem. Greške u kodiranju se odnose na bagove, ali i stavke koje nisu ubačene zbog dizajna nego intuicijom (ili kao rezultat nerazmišljanja o potencijalnim posledicama). Ovde spada *buffer ovevrflow*, uslov trke, pa čak i neispravni generatori slučajnih brojeva.

Kritičan deo za bezbednost je obezbeđivanje da radnici, partneri i preduzetnici s pristupom bazi ne zloupotrebe svoje akreditive. Protiv ovih ranjivosti teško je se sačuvati jer korisnici s legitimnim pristupom mogu uzeti podatke u svoje svrhe. Organizacije moraju da utvrde da korisnici s pristupom sistemu baza i aplikacijama to koriste samo za informacije njima neophodne za posao.

2.2. Najbolje prakse

Svaka kompanija će definisati svoje korake u obezbeđivanju baze podataka u zavisnosti od modela koji koriste i poslovnih prioriteta. U nastavku je dat par preporuka na koje se može obratiti pažnja, a koje bi pomogle pri obezbeđvanju baze podataka.

- Fizička bezbednost – kritično je ne posmatrati i fizički hardver na kom se podaci skladište i održavaju. Ovaj aspekt uključuje zaključavanje soba u kojima su baze i njihovi serveri. Pored toga, uključuje i tim koji će nadgledati fizički pristup toj opremi. Krucijalni aspekt ove prakse je postojanje bekapa i mera za oporavak od katastrofe ukoliko do neke fizičke katastrofe dođe. Takođe, ne treba hostovati veb servere i aplikacije na istom serveru kao baza koja se obezbeđuje.
- Web aplikacije i *firewall*-ovi – *firewall*-ovi sprečavaju napadače da pristupe IT mreži organizacije preko interneta i krucijalni su prerekvizit za sajber bezbednosna pitanja. Web aplikacije koje komuniciraju s bazama mogu se štititi softverom za upravljanje pristupom. Ovaj softver određuje ko i na koji način sme pristupiti web appu.
- Enkripcija baze podataka – enkripcija je jedna od najefikasnijih praksi jer se implementira tamo gde su podaci u bazi. Međutim, organizacije mogu kriptovati i statičke i dinamičke podatke, tako da su oni zaštićeni dok se kreću između IT sistema u organizaciji. Enkriptovani podaci deluju kao nerazumljiva sekvenca dok se ne dekriptuju odgovarajućim ključevima. Zbog toga, čak i ako neko pristupi kriptovanim podacima, njima će oni biti beznačajni. Enkripcija je ključna i za održavanje bezbednosti podataka i može biti efikasna kod bezbednosti IoT-a.
- Upravljanje lozinkama i dozvolama – ovo je kritično za održavanje bezbednosti baze. Ovaj zadatak uglavnom nadgledaju IT timovi ili tim za bezbednost. Nekada praksa uključuje i liste kontrole pristupa. Organizacija može preduzeti par različitih koraka za upravljanje lozinkama, npr. upotrebu dvofaktorske ili višefaktorske autentifikacije, ili ograničavanje vremena za unos akreditiva. Ovakve prakse će, međutim, zahevati konstantnu modifikaciju listi pristupa i dozvola. To može oduzeti dosta vremena, ali su se te prakse pokazale kao efikasne.
- Izolovanje osetljivih baza podataka – teško je probiti bezbednost baze ukoliko su one osetljive izolovane. U zavisnosti od tehnika izolacije koje se koriste, neautorizovani korisnici možda neće ni znati da osetljive baze postoje. Softverski definisani parametri dobar su način izolacije osetljivih baza kako se one ne bi pojavljivale na mreži određenog korisnika. Ovaj pristup otežava preuzimanje baze bočnim napadom, a efikasan je i protiv *zero-day* napada. Strategije izolacije jedan su od najboljih načina za učvršćivanje bezbednosti baze podataka na nivou pristupa. Često se kombinuju s ostalim sigurnosnim slojevima poput javnih ključeva i enkripcije.
- Upravljanje promenama – ovo upravljanje zahtevaće nacrte, i to idealno unapred, procedura koje se moraju obaviti tokom promene a u cilju očuvanja bezbednosti baze. Primeri promena uključuju spajanja, nabavke ili jednostavno korisnike koji dobijaju pristup različitim IT resursima. Neophodno je dokumentovati koje će se promene desiti za

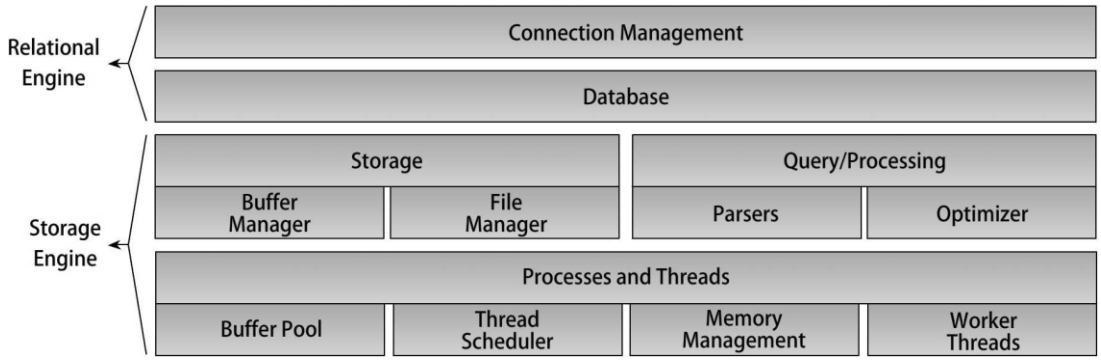
bezbedan pristup bazama i aplikacijama. Takođe je bitno identifikovati sve aplikacije i sisteme koji će koristiti tu bazu.

- Revizija baze – revizija uglavnom zahteva redovno čitanje log fajlova baza i aplikacija. Ove informacije otkrivaju ko je pristupio kom repozitorijumu ili aplikaciji, kada i koje su akcije preduzeli. Ako je bilo neautorizovanog pristupa podacima, blagovremene revizije mogu pomoći u smanjenju celokupnog uticaja tih upada alarmiranjem administratora baze. Što brže organizacija odreaguje na upad, više će vremena imati da obavesti sve uključene klijente i ograniči napravljenu štetu. Revizija daje centralizovan pregled bezbednosti baze kao poslednji korak zaštite.

Kao dodatak implementiranju slojeva sigurnosnih pravila u celom mrežnom okruženju, sigurnost baza zahteva i utvrđivanje politika za pristup samoj bazi. To će uključivati administrativna pravila, preventivna pravila i pravila detekcije. Administrativna pravila odnose se na upravljanje instalacijom, promenama i konfiguracijama baze. Preventivna pravila se bave pristupom, enkripcijom, tokenizacijom i maskiranjem. Na kraju, pravila detekcije se bave nadgledanjem aktivnosti baze i alatima za prevenciju gubitka podataka.

3. SQL Server

SQL Server je sistem za upravljanje relacionim bazama podataka razvijen od strane Majkrosofta s ciljem pružanje brze, bezbedne i skalabilne platforme za pristup podacima. Skalabilnost ove arhitekture je najbolja crta jer je razvijena da podjednako dobro funkcioniše kroz široki spektar okruženja, od baze podataka na personalnom računaru za samo jednog korisnika, pa sve do baze na klasteru servera koja opslužuje par hiljada korisnika. Na slici 1 prikazana je arhitektura SQL Servera. On je razvijan za upotrebu na Windows platformi, a klijenti se povezuju sa serverom koristeći Majkrosoftov format – TDS (*eng. Tabular Data Stream*). To je protokol koji opisuje kako klijent i server komuniciraju – daje specifikacije tipova podataka i načine njihovog toka. TDS se najčešće enkapsulira u transportni protokol poput TCP/IP-a



Slika 1. Arhitektura SQL Servera.

SQL Server se može podeliti na dve glavne komponente:

- Relaciona mašina (*eng. Relational Engine*) – primarna odgovornost je obrada upita i vraćanje podataka. Zadatke koja relaciona mašina izvršava inicirani su korisničkim upitima i uključuju parsiranje SQL-a, optimizaciju plana izvršenja i vraćanje rezultata korisniku.
- Mašina za skladištenje (*eng. Storage Engine*) – primarna odgovornost je obezbeđivanje efikasnog upravljanja fajlovima, memorijom, oporavkom, logovanjem i transakcijama. On obezbeđuje da je memorija dostupna i prikladno alocirana, da se fajlovi efikasno koriste i skladište, a bekap radi dovoljno često.

SQL Server obezbeđuje i par alata za upravljanje. Ovde će biti pomenuta dva alata koja će biti korišćena u primerima: *SQL Server Configuration Manager* i *SQL Server Management Studio. Configuration Manager* se koristi za konfiguraciju instalacije SQL Servera. On pruža administratorima opciju da konfigurišu mrežne protokole koje aplikacija koristi za upravljanje uslugama koje se vezuju za SQL Server. *Management Studio* predstavlja primarni administrativni interfejs ka SQL Server bazi. Ovaj alat omogućuje administratorima da iz jedne konzole konfigurišu i komuniciraju s bazom.

3.1. Najčešće sigurnosne pretnje

Developeri moraju da razumeju bezbednosne pretnje, alate koji su dati za njihovo suzbijanje, ali i kako izbeći bezbednosne propuste koje su sami napravili. Bezbednost se može posmatrati kao lanac, pri čemu slom jedne karike ugrožava jačinu celog sistema. Neke od najčešćih bezbednosnih pretnji su:

- SQL Injection – ovo je proces u kome zlonameran korisnik unosi *Transact-SQL* upite umesto validnog unosa. Ako se unos direktno prosledi serveru, bez ikakve procene, i ukoliko aplikacija izvrši taj kod, napad može potencijalno oštetiti ili uništiti podatke. Ovakvi napadi mogu se sprečiti upotrebom uskladištenih procedura i parametrizovanih komandi, izbegavanjem dinamičkog SQL-a i ograničavanjem dozvola na nivou svih korisnika. Detaljnije će o ovome biti reči u poglavlju 3.2.4.
- EOP (*eng. Elevation of Privilege*) napad – do ovog napada dolazi kada napadač može da pretpostavi privilegije pouzdanog naloga, na primer vlasnika ili administratora. Što se tiče prevencije, treba uvek upotrebljavati korisničke naloge sa najmanje privilegija za izvršenje i dodeljivati samo neophodne dozvole. Uvek treba izbegavati upotrebu administrativnih naloga ili vlsničkog naloga za izvršenje koda. Ovo će ograničiti količinu štete do koje može doći ako se napad desi. Za izvršavanje zadataka za koje su neophodne dodatne dozvole, koristite potpisivanje procedura ili lažno predstavljanje (*eng. impersonation*) samo u toku trajanja zadatka. Možete potpisati procedure sertifikatima ili koristiti lažno predstavljanje za privremeno davanje dozvola. O potpisanim procedurama i lažnom predstavljanju biće reči u poglavlju 3.2.4.
- Ispitivanje i obzervacija - *Probe* napad može koristiti poruke o grešci koje generiše aplikacija kako bi tražila bezbednosne ranjivosti. Treba uneti rukovanje greškama u sav kod kako bi se sprečilo vraćanje informacija o greškama krajnjem korisniku.

- Autentifikacija – Injection napad na konekcioni string se može desiti pri prijavljivanju na SQL server ukoliko se konekcioni string baziran na korisničkom unosu kreira u toku izvršenja. Ako se on ne proveri za validne parove ključnih reči, napadač može ubaciti dodatne karaktere, time potencijalno ugrožavajući osetljive podatke ili druge resurse na serveru. Cilj je koristiti Windows autentifikaciju gde god je to moguće, o čemu će biti reči u poglavlju 3.2.1.
- Lozinke – Mnogi napadi uspeju jer je napadač uspeo da dobije ili pogodi lozinku privilegovanog korisnika. Lozinke su prva linija odbrane od napadača, pa je podešavanje jake lozinke krucijalno za bezbednost sistema. Neophodno je kreirati i forsirati politike za lozinke za *mixed mode authentication.* Pored toga, uvek treba dodeliti jaku lozinku nalogu sistemskog administratora, čak i pri korišćenju Windows autentifikacije.

Česta bezbednosna razmatranja poput krađe podataka ili vandalizma su prisutna, nevezano za verziju SQL Servera koja se koristi. Integritet podataka takođe se treba razmatrati kao bezbednosni problem. Ako podaci nisu zaštićeni, može se desiti da postanu beznačajni ako je manipulacija podacima dozvoljena, a podaci se nehotice ili zlonamerno modifikuju netačnim vrednostima ili kompletno izbrišu. Pored toga, često postoje pravni zahtevi kojih se treba pridržavati, poput ispravnog skladištenja i poverljivosti informacija. Skladištenje neke vrste ličnih podataka je potpuno zabranjeno, što zavisi od zakona koji su na snazi.

3.2. Potencijalna rešenja

SQL Server ima dosta funkcija koje podržavaju kreiranje bezbednih baza podataka. Bitno je razumeti da same bezbednosne opcije ne garantuju bezbednost baze podataka. Svaka primena baze je jedinstvena u svojim zahtevima, izvršnom okruženju, razvojnom modelu, fizičkoj lokaciji i korisničkoj populaciji. Neke primene koje su lokalnog opsega će možda zahevati samo minimalni nivo bezbednosti, dok će druge lokalne aplikacije ili one razvijane preko interneta zahtevati strožije mere bezbednosti i neprestano nadgledanje i evaluaciju.

Ne postoji jedinstven tačan način za obezbeđivanje SQL Server klijentske aplikacije. Aplikacija koja je solidno obezbeđena pri inicijalnom razvoju može postati manje bezbedna tokom vremena. Nemoguće je s bilo kojom tačnošću predvideti koje se pretnje mogu pojaviti u budućnosti.

Bezbednosni zahtevi primene SQL Servera trebaju se razmotriti za vreme dizajniranja, ne kasnije. Procena pretnji u ranijim fazama razvoja daje priliku da se potencijalna šteta pri detekciji ranjivosti izbegne. Kreiranjem višestrukih linija odbrane oko baze, može se smanjiti šteta izazvana bezbednosnim prodorom. Developeri moraju da dođu do kombinacije opcija i funkcionalnosti koje su najprikladnije za suočavanje s poznatim pretnjama, ali i da predvide pretnje koje se mogu javiti u budućnosti.

Neke od stavki koje će biti pominjane u narednim poglavljima su autentifikacija, autorizacija, enkripcija i revizija. Na slici 2 je dat prikaz ovih slojeva i rečeno je na šta se koji proces odnosi. Autentifikacija ustanovljava identitet korisnika ili procesa koji se autentifikuje. S druge strane,

autorizacija je proces određivanja kojim osiguranim resursima se sme pristupiti i koje su operacije za te resurse dozvoljene. Enkripcija predstavlja transformaciju podataka iz čitljivog u kriptovano stanje nekim od poznatih algoritama za šifriranje. Revizija se odnosi na pregled logova kako bi se utvrdilo da li je došlo do problema i ako jeste, ko je za njega odgovoran.

Slika 2. Prikaz procesa autentifikacije, autorizacije, enkripcije i revizije.

3.2.1. Autentifikacija

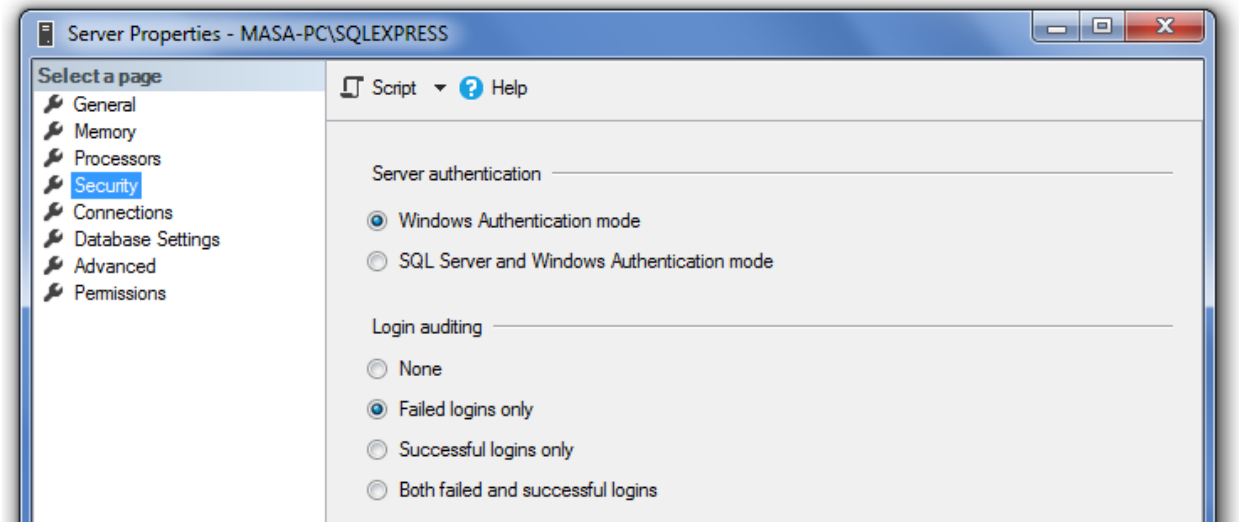
Pre nego što se krene s pričom o autentifikaciji, treba reći da instanca SQL Servera sadrži hijerarhijsku kolekciju entiteta, počevši od samog servera. Svaki server sadrži više baza, a svaka baza sadrži kolekciju osiguranih stavki (*eng. securable*). Svaka od njih uz sebe ima dozvole (*permissions*) koje se mogu dati korisniku, što može biti omogućen pristup za pojedinca, grupu ili proces.

SQL Server podržava dva režima autentifikacije:

- Windows režim - on je podrazumevani i često se pominje kao integrisana bezbednost jer je ovaj bezbednosni model čvrsto povezan sa Windows-om. Specifični Windows korisnici i grupni nalozi mogu da se prijave na SQL Server. Korisnici koji su već autentifikovani ne moraju da unose bilo kakve dodatne akreditive. Preporučuje se upotreba Windows autentifikacije gde god je to moguće. Ona koristi niz enkriptovanih poruka za autentifikaciju korisnika na SQL Serveru. Bezbednija je od alternative, koristi *Kerberos* protokol, zahteva jake lozinke i definiše njihovo trajanje.
- Kombinovani režim se odnosi na autentifikaciju i od strane Windows-a i od strane SQL Servera. Parovi korisničkog imena i lozinke čuvaju se u okviru SQL Servera. Korisnička imena i kriptovane lozinke prenosiće se preko mreže, što ih izlaže riziku. Sa Windows autentifikacijom, korisnici koji su već prijavljeni i autentifikovani ne šalju svoje akreditive.

Ovaj režim je pogodan za okruženja sa starijim ili kombinovanim operativnim sistemima. Ako se koristi kombinovani režim, neophodno je kreirati SQL Server login koji se smešta na server.

Po pokretanju *Configuration Manager-a*, podrazumevani vid autentifikacije je Windows autentifikacija. To se može izmeniti odabirom u stavci *Security*, koja je prikazana na slici 3. Ovde se takođe vidi da se može definisati za koje se akcije upisuju podaci o prijavljivanjima – ovde će se pratiti samo neuspešna prijavljivanja. Njihova revizija može otkriti određene bezbednosne propuste.

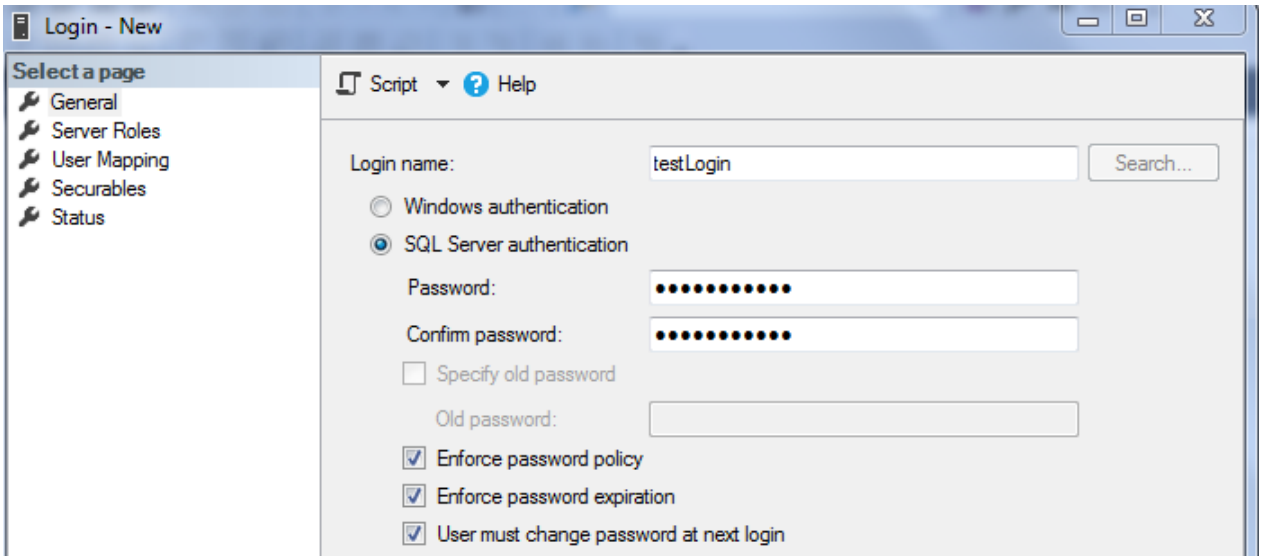


Slika 3. Biranje režima autentifikacije.

SQL Server bezbednosni model ima dva nivoa: server i bazu podataka. Login je nalog definisan na nivou servera, s dodeljenim dozvolama za povezivanje i potencijalno izvođenje administravnih zadataka i pristup bazama na instanci servera. Kreiranje login-a je prvi korak ka izdavanju dozvola u SQL Serveru. Login se kreira na nivou servera, a da bi mu se dozvolio pristup bazi, treba se kreirati korisnik u toj bazi i njega treba mapirati na postojeći login. Nakon toga se mogu dodeljivati uloge. SQL Server omogućava tri tipa logina:

- Lokalni Windows korisnički nalog ili nalog proverenog domena – SQL Server se u ovom slučaju uzda u Windows da autentifikuje Windows korisničke naloge.
- Windows grupa – Davanje pristupa grupi podrazumeva pristup svim prijavljivanjima korisnika koji su članovi te grupe.
- SQL Server login – Smešta se korisničko ime i heš vrednost lozinke u master bazu i koriste se interne metode autentifikacije za proveru pokušaja prijavljivanja.

Kreiranje novog login-a je prikazano na slici 4. Ovog puta je odabrana SQL Server autentifikacija, što iziskuje unos lozinke koja će se pri prijavljivanju koristiti. Da bi login bio validan, neophodno je restartovati SQL Server. Pri prijavljivanju s ovim korisničkim imenom, specificiraće se vid autentifikacije i tražiće se unos lozinke. Na slici se vide i dodatne stavke vezane za politiku za lozinke. Ova politika je korisna za sprečavanje *brute-force* napada.



Slika 4. Kreiranje login-a.

Login se može kreirati i SQL naredbom, kao što je prikazano na slici 5. SQL Server login je kreiran, s definisanim parametrima politike lozinki, kao što je bilo definisano na slici 4.

```
CREATE LOGIN Marija WITH PASSWORD = 'vrlo jaka lozinka' MUST_CHANGE,
CHECK_EXPIRATION = ON, CHECK_POLICY = ON, DEFAULT_DATABASE =
Books, DEFAULT_LANGUAGE = us_English;
```

Slika 5. Kreiranje logina SQL naredbom.

Lozinka za login može se menjati naredbom na slici 6. Neophodno je navesti i staru lozinku ukoliko ne postoje izričite dozvole koje dozvoljavaju njeno nenavođenje.

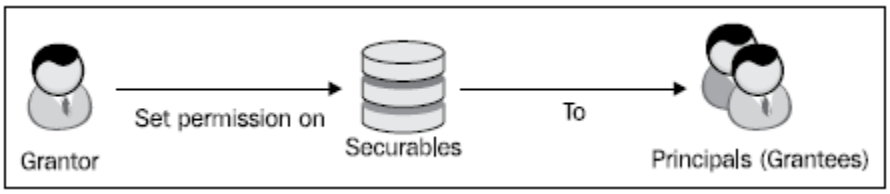
```
ALTER LOGIN Marija WITH
PASSWORD = 'jos jaca lozinka'
OLD_PASSWORD = 'vrlo jaka lozinka';
```

Slika 6. Promena lozinke za login.

3.2.2. Autorizacija

Kada su podaci autentifikovani ili je sistem potvrdio identitet korisnika ili logina, predefinisane dozvole će odredi kojim bazama ili objektima se sme pristupiti. Proces u kome se dozvole primenjuju na korisnika poznat je kao autorizacija. To je proces obezbeđivanja da korisnici ili aplikacije koje zahtevaju pristup okruženju ili objektu unutar okruženja imaju dozvolu za isto. Upravljanje korisnicima je kritično, pa će biranje najprikladnijih privilegija za svako korisnika pomoći u održavanju bezbednosti baze.

Pri kreiranju objekata baze mora se eksplicitno navesti njihove dozvole, kako bi oni bili dostupni korisnicima. Svaki *securable* ima dozvole koje se mogu dati korisniku upotrebom neke od *permission* naredbi. Na slici 7 je prikazan sistem dodeljivanja dozvola.



Slika 7. Princip dodeljivanja dozvola.

Izvršavanjem naredbe sa slike 8 može se videti koje se dozvole odnose na koju klasu *securable* objekata. Deo rezultata izvršenja dat je na slici 9. Najčešće dozvole su ALTER, CONNECT, DELETE, EXECUTE, IMPERSONATE, INSERT, SELECT, TAKE OWNERSHIP, itd.

```
SELECT * FROM sys.fn_builtin_permissions(DEFAULT);
```

Slika 8. Naredba za prikaz dozvola.

	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
55	DATABASE	ALTER ANY DATABASE DDL TRIG...	ALTG	ALTER	SERVER	CONTROL SERVER
56	DATABASE	ALTER ANY DATABASE EVENT NO...	ALED	ALTER	SERVER	ALTER ANY EVENT NOTIFICA...
57	DATABASE	ALTER ANY DATABASE AUDIT	ALDA	ALTER	SERVER	ALTER ANY SERVER AUDIT
58	DATABASE	VIEW DATABASE STATE	VWDS	CONTROL	SERVER	VIEW SERVER STATE
59	DATABASE	VIEW DEFINITION	VW	CONTROL	SERVER	VIEW ANY DEFINITION
60	DATABASE	TAKE OWNERSHIP	TO	CONTROL	SERVER	CONTROL SERVER
61	DATABASE	ALTER	AL	CONTROL	SERVER	ALTER ANY DATABASE
62	DATABASE	CONTROL	CL		SERVER	CONTROL SERVER
63	OBJECT	SELECT	SL	RECEIVE	SCHEMA	SELECT
64	OBJECT	UPDATE	UP	CONTROL	SCHEMA	UPDATE
65	OBJECT	REFERENCES	RF	CONTROL	SCHEMA	REFERENCES

Slika 9. Deo rezultata izvršene naredbe.

Postoje 3 Transact-SQL *permission* naredbe, GRANT, REVOKE i DENY. Na slici 10 je prikazana upotreba naredbe GRANT.

```
GRANT SELECT ON OBJECT::dbo.Published TO Marija;
```

Slika 10. Upotreba naredbe GRANT.

Bitan deo defanzivne strategije protiv bezbednosnih pretnji je razvoj aplikacije upotrebom LUA pristupa (*eng. Least-privileged User Account*). On obezbeđuje da se korisnici prijavljuju na naloge sa ograničenim pristupom. Ovu strategiju treba koristiti pri davanju dozvola korisnicima baze – uvek treba izdavati minimum dozvola koji će korisniku biti dovoljan da izvrši neophodan zadatak.

Unutar SQL Servera, *uloge (roles)* se definišu na nivou servera ili baze. Uloge na nivou servera daju dozvole za manipulisanje serverskim okruženjem, i ove dozvole se daju login nalozima. S

druge strane, uloge na nivou baze služe za dobijanje pristupa objektima baze i dodeljuju se korisničkim nalozima. Davanje dozvola ulogama umesto pojedinačnim korisnicima može pojednostaviti administraciju. Dozvole koje se definipu na nivou uloge nasledиće svi članovi tog skupa. Lakše je dodavati ili uklanjati korisnike iz uloge nego kreirati odvojene skupove dozvole za inidividualne korisnike. Uloge se i mogu ugneždavati, ali to može degradirati performanse.

3.2.3. Enkripcija

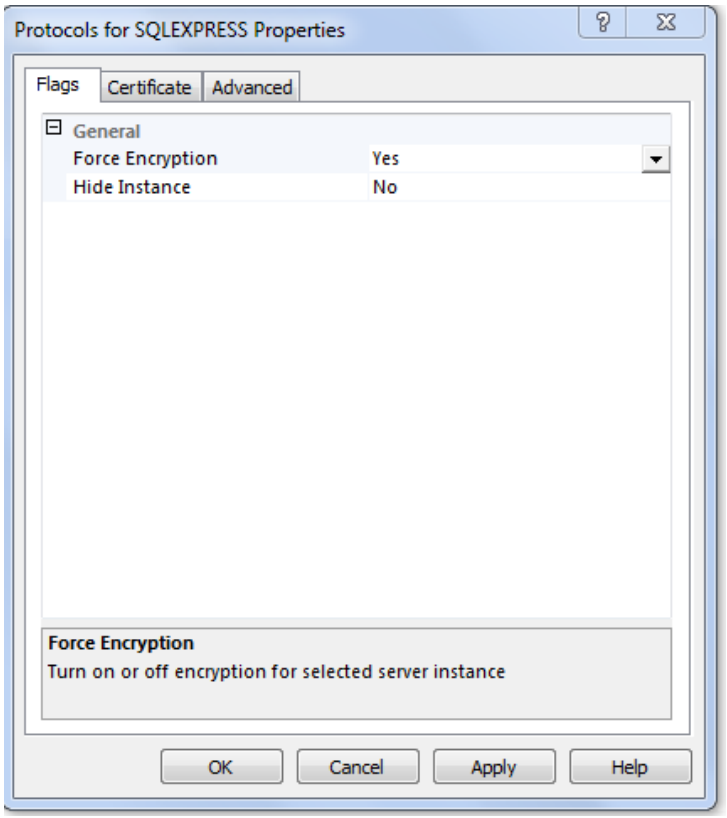
SQL Server pruža mogućnost enkripcije i dekripcije podataka upotrebom sertifikata, asimetričnih ili simetričnih ključeva. Ovim parametrima upravlja u internom skladištu za sertifikate. Ono koristi hijerarhiju koja obezbeđuje sertifikate i ključeve na jednom nivou slojem iznad u hijerarhiji.

Najbolji podržani režim enkripcije je enkripcija simetričnim ključem, koja je pogodna za velike količine podataka. Ključevi mogu biti kriptovani sertifikatima, lozinkama ili drugim simetričnim ključevima. SQL Server podržava par algoritama za enkripciju simetričnim ključem: DES, 3DES, RC2, RC4, 128bitni AES, itd. Algoritmi su implementirani upotrebom *Windows Crypto API-ja.*

U opsegu konekcije SQL Server može da drži više aktivnih simetričnih ključeva. Ključ se uzima iz skladišta i dostupan je za dekripciju. Kada se deo podataka dekriptuje, nije neophodno definisati koji se simetrični ključ koristi, jer svaka kriptovana vrednost sadrži GUID ključa koji se upotrebljava. Proveravaće se strim kriptovanih bajtova sa simetričnim ključem. Ako korektan ključ nije aktivan, vratiće se NULL.

Pored pomenutih algoritama, podržano je još par opcija za enkripciju:

- SSL (*eng. Secure Sockets Layer*) – kriptuje saobraćaj između instance servera i klijentske aplikacije. Pored toga, klijent može potvrditi identitet servera pomoću njegovog sertifikata. SSL enkripcija može se uključiti kao na slici 11, u *Configuration Manager*-u. Treba napomenuti da je za ovaj korak neophodno pribavljanje SSL sertifikata od *Certificate Authority (CA)* kao što su Verisign, Comodo, itd. Nakon biranja forsirane enkripcije, biće neophodno unošenje putanje do pribavljenog sertifikata.



Slika 11. Uključivanje SSL enkripcije.

- TDE (*eng. Transparent Data Encryption*) – kriptuje podatke na disku, tj. sve podatke i log fajlove. TDE se može aktivirati na sledeći način:
    - Kreirati master ključ i odmah ga bekapovati na bezbednu lokaciju, kao što je prikazano na slici 12. Lozinka mora biti kreirana tako da podržava politiku Windows-a za lozinke.

```
USE master;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'lozinka';

BACKUP MASTER KEY TO FILE = '\\Masa\SQL_master.key' ENCRYPTION BY PASSWORD = 'jaka lozinka1@';
```

Slika 12. Kreiranje master ključa i bekapa.

    - Kreirati serverski sertifikat i bekapovati ga. Naredbe za ovo date su na slici 13.

```
CREATE CERTIFICATE TDECert WITH SUBJECT = 'TDE Certificate';

BACKUP CERTIFICATE TDECert TO FILE = '\\Masa\SQL_TDECert.cer'
WITH PRIVATE KEY (FILE = '\\Masa\SQL_TDECert.pvk',
ENCRYPTION BY PASSWORD = 'jaka lozinka1@22');
```

Slika 13. Kreiranje i bekap sertifikata.

- U bazi koju želimo da koristimo kreiramo ključ za enkripciju baze i aktiviramo enkripciju, kao na slici 14. Ovde je kao algoritam naveden AES_128, a kao opcije se nude i AES_192, AES_256 i TRIPLE_DES_3KEY.

```
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE TDECert;


ALTER DATABASE Books SET ENCRYPTION ON;
```

Slika 14. Generisanje ključa za enkripciju i aktivacija enkripcije.

- *Backup* enkripcija – slično TDE-u, s tim što su sada bekapovi kriptovani.

3.2.4. Pisanje bezbednog dinamičkog SQL-a

Injection napadi se baziraju na činjenici da u mnogim tehnologijama naredbe i podaci nisu striktno razdvojeni. Ovo omogućuje napadaču da prosledi instrukcije na mestu gde programer očekuje podatke. Pomoću prosleđenih instrukcija, napadač može da promeni predviđeno ponašanje programa.

Ovi napadi se najčešće pokreću kroz korisnički interfejs. Da bi izveo napad, napadač pokušava da kroz ulaz u program prosledi podatke koji će se interpretirati kao komande. Da bi napad bio uspešan, tri pretpostavke trebaju biti ispunjene:

- Identifikovana je korišćena tehnologija – Ovakvi napadi su izrazito zavisni od programskog jezika.
- Identifikovani su svi mogući korisnički ulazi – postoje neki očigledni unosi, poput Web formi, ali napadač može identifikovati, recimo, i skrivene HTTP input elemente.
- Nađen je adekvatan korisnički ulaz koji predstavlja slabu tačku.

Svaka procedura koja generiše SQL naredbe trebalo bi biti revidirana za Injection ranjivosti jer će SQL Server izvršiti sve sintaksno validno upite koje primi. Može se manipulisati čak i parametrizovanim podacima. Ukoliko se koristi dinamički SQL, treba parametrizovati komande i nikada ne uključivati vrednosti parametara u *query string.*

Proces ovog napada funkcioniše tako što se preuranjeno okonča tekstualni string i na njega nadoveže nova komanda. Dok god je SQL kod sintaksno ispravan, u programu se ne može detektovati da je on modifikovan. Zato je neophodna validacija korisničkog unosa i pažljiv pregled koda koji izvršava kreirane SQL komande za server koji se koristi. Nikad ne treba konkatenirati korisnički unos koji nije prošao validaciju, jer je to primarna tačka ulaza za ovakav napad.

Dodatni saveti za izbegavanje ovog napada su:

- Validirati korisnički unos testiranjem tipa, dužine, formata i opsega. Koristiti QUOTENAME() funkciju za izbegavanje sistemskih imena ili REPLACE() za izbegavanje karaktera u stringu.
- Implementirati više slojeva validacije u aplikaciji.
- Testirati veličinu i tip ulaznih podataka i forsirati odgovarajuća ograničenja – ovo može pomoći u sprečavanju namernih prekoračenja bafera.
- Testirati sadržaj stringova i prihvatiti samo očekivane vrednosti. Odbiti unose koji sadrže binarne podatke i karaktere za komentare.
- Koristiti Regex izraze u klijentskom kodu za uklanjanje nevalidnih karaktera.

Izvršavanje dinamički kreiranih SQL naredbi u proceduralnom kodu dovešće do toga da SQL Server proverava dozvole u odnosu na objekte kojima se tim SQL-om pristupa. Postoje metode za davanje pristupa podacima upotrebom uskladištenih procedura i korisnički definisanih funkcija koje izvršavaju dinamički SQL.

- Upotreba lažnog predstavljanja preko EXECUTE AS naredbe: SQL Server ne proverava dozvole onog koji poziva proceduru ukoliko ona i tabele imaju istog vlasnika. Međutim, to neće funkcionisati ako su vlasnici različiti ili u slučaju dinamičkog SQL-a. EXECUTE AS naredba se može iskoristiti kada onaj koji je poziva nema dozvole za referencirane objekte iz baze. Efekat naredbe je da se kontekst izvršenja prebacuje na *proxy* korisnika – sav kod i svi pozivi ka ugneždenim procedurama se izvršavaju pod bezbednosnim kontekstom *proxy* korisnika. Tek kada se izda REVERT naredba, kontekst izvršenja vraća se originalnom korisniku. Na slici 15 data je naredba EXECUTE AS – treba imati na umu da mora postojati IMPERSONATE dozvola nad korisnikom ili loginom za koji se lažno predstavljate.

```
EXECUTE AS LOGIN = 'loginName';
```

Slika 15. Naredba EXECUTE AS.

- Potpisivanje uskladištenih procedura sertifikatima – kada se potpisana uskladištena procedura izvrši, dozvole date korisniku sertifikata spajaju se s dozvolama koji ima onaj koji je proceduru pozvao. Kontekst izvršenja ostaće isti: korisnik sertifikata se ne predstavlja lažno u ovom slučaju.

3.3. Dodatno jačanje sigurnosti

Pored svih pomenutih rešenja, SQL Server okruženje bi se moglo ojačati sprovođenjem nekih od narednih koraka:

- Koristiti Windows autentifikaciju umesto kombinovane autentifikacije.
- Koristiti korisnički nalog s niskim nivoom privilegija, ne koristiti administratorski nalog za izvršenja.
- Obezbediti nalog sistemskog administratora jakom lozinkom.
- Pregledati na koji način su uloge dodeljene korisnicima na nivou baze i ograničiti njihovo dodeljivanje na najmanji neophodni skup.
- Fizički obezbediti server na kome se SQL Server nalazi.
- Definisati upotrebu *firewall*-a za instancu SQL Servera.
- Sve SQL Server sistemske fajlove i fajlove s podacima instalirati na NTFS particiji i definisati prikladne dozvole.
- Obrisati fajlove za podešavanje – oni mogu da sadrže osetljive informacije o konfiguraciji koje su logovane tokom instalacije.
- Pregledati sve lozinke ili makar proveriti postoje li *null* lozinke koristeći naredbu na slici 16.

```
SELECT name, password FROM syslogins WHERE password is null;
```

Slika 16. Naredba za proveru lozinki.

- Koristiti *row-level* bezbednost – ovo pojednostavljuje dizajn i kodiranje bezbednosti u aplikaciji. RLS (*eng. Row-Level Security*) implementira restrikcije na pristup redovima podataka. Na primer, studentima se može dati pristup samo onim redovima koji se odnose na njihovu studentsku grupu. Restrikcija se nalazi na nivou baze, a ne dalje od podataka, na nivou aplikacije. Svaki put kada se pokuša pristup podacima, sistem baze će primeniti restrikcije pristupa. RLS podržava dva tipa predikata:
  - Filter predikate, koji filtriraju redove dostupne operacijama čitanja;
  - Blok predikate, koji eksplicitno blokiraju operacije upisa koje ugrožavaju predikat.
- Pokrenuti proces kojim bi se periodično pregledale uloge i grupna članstva.
- Uklonite mrežne biblioteke koje se ne koriste – Većina okruženja se bazira na TCP/IP-u, tako da se sve ostale biblioteke trebaju ukloniti.
- Zahtevati da se sav pristup serveru baze umreži. Ne promovisati udaljeni pristup operativnom sistemu.
- Ukloniti ili ograničiti pristup extended procedurama (xp__ uskladištenim procedurama) – SQL Server extended procedure su DLL-ovi koje može instalirati administrator kako bi se pružila poboljšana funkcionalnost. Međutim, glavni rizik kod njihove moći je mogućnost da pristupe i izazovu akcije na nivou operativnog sistema. Korišćenje ovih procedura zamagljuje granicu između baze i operativnog sistema i mogu dati previše privilegija korisniku prijavljenom na bazu. Neophodno je održati distancu između hosta i baze. Još

jedan rizik se odnosi na ranjivosti unutar samih procedura. Na primer, extended procedure *xp_regread* i *xp_instance_regread* dozvoljavaju ulozi PUBLIC da čita iz sistemskog registra. U njemu postoji dosta informacija koje mogu biti korisne napadaču. Na slikama 17 i 18 prikazane su dve različite naredbe na osnovu kojih se mogu dobiti informacije o operativnom sistemu koje se kasnije mogu zloupotrebiti. Naredbom sa slike 17 dobija se informacija o tome gde je instanca SQL Servera instalirama, a naredbom sa slike 18 koji je podrazumevani login.

```
exec xp_regread
'HKEY_LOCAL_MACHINE',
'SOFTWARE\Microsoft\MSSQLServer\Setup', 'SQLPath'

exec xp_instance_regread
'HKEY_LOCAL_MACHINE',
'SOFTWARE\Microsoft\MSSQLServer\Setup', 'SQLPath'
```

```
exec xp_regread 'HKEY_LOCAL_MACHINE', 'SOFTWARE\Microsoft\
MSSQLServer\MSSQLServer', 'DefaultLogin'

exec xp_instance_regread 'HKEY_LOCAL_MACHINE', 'SOFTWARE\
Microsoft\MSSQLServer\MSSQLServer', 'DefaultLogin'
```

Slike 17 i 18. Primeri naredbi s uskladištenim extended procedurama.

- Ne instalirati korisnički kreirane extended procedure jer se one izvršavaju s punim sigurnosnim pravima na serveru.
- Ne generisati HTML unutar vaših uskladištenih procedura.
- Ne instalirati *full-text* pretragu osim ako aplikacija to ne iziskuje.
- Ugasiti SQL mejl opcije i naći alternativna rešenja kao metode notifikacije.
- Pažljivo pratiti sve neuspešne pokušaje prijavljivanja.
- Omogućiti reviziju kao poslednji korak za jačanje sigurnosti.

4. Zaključak

Sigurnost baze podataka je jako bitna i ne može se očekivati normalno funkcionisanje aplikacije ukoliko ona nije dobro sprovedena. Gubitak određenih podataka bio bi katastrofalan, a posledice nesagledive. Srećom, SQL Server nudi dosta opcija koje se mogu kombinovati po potrebama organizacije, a koje daju različite nivoe zaštite. On nudi i dobru dokumentaciju koju korisnici mogu ispratiti kako bi našli onu meru sigurnosti koja je za njih najpogodnija.

LITERATURA

[1] A. Basta, M. Zgola, *Database Security*, Cengage Learning, 2011.

[2] R. Ben-Natan, *Implementing Database Security and Auditing*, Elsevier Digital Press, 2005.

[3] R. Bruchez, *Microsoft SQL Server 2012 Security Cookbook*, Packt Publishing, 2012.

[4] SQL Server dokumentacija:

https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/sql-server-security

[5] Prezentacije sa sajta predmeta

[6] Database Security, Wikipedia: https://en.wikipedia.org/wiki/Database_security

[7] What is Database Security, sumo logic: https://www.sumologic.com/blog/what-is-database-security/

[8] Database Security Report 2016: https://cybersecurityventures.com/database-security-report-2016/

[9] Database Security, IBM Cloud: https://www.ibm.com/cloud/learn/database-security