Sami Järvinen

SQL-tietokantojen suojaaminen tietoturvauhilta

Tietotekniikan kandidaatintutkielma

30. huhtikuuta 2021

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Sami Järvinen

Yhteystiedot: saemjuja@student.jyu.fi

Ohjaaja: Tuomo Rossi

Työn nimi: SQL-tietokantojen suojaaminen tietoturvauhilta

Title in English: Protecting SQL databases from security threats

Työ: Kandidaatintutkielma

Opintosuunta: Kaikki opintosuunnat

Sivumäärä: 22+0

Tiivistelmä: Viimeisen kolmenkymmenen vuoden aikana tietotekniikkaympäristö on käynyt läpi monia muutoksia, ja tietokantatutkimusyhteisöt ovat yrittäneet pysyä askeleen edellä tietokannan turvallisuusuhkia. Turvallisuuden merkitystä on lisännyt yritysten kriittisten toimintojen digitalisoituminen. Tietokannat usein pitävät sisällään arkaluontoisia tietoja ja ne ovat olennainen osa kaikkia tietojärjestelmiä. Yleisimmät tietokantaohjelmistot perustuvat relaatiomalliin, johon voi tehdä hakuja SQL-kielellä. Tietokantojen tietoturvallisuus on tärkeää, jos tapahtuu tietovuoto, siitä on haittaa tietokannan omistajalle ja kohteelle, josta tietoa kerättiin. Tässä tutkielmassa käsitellään SQL-injektiota, palvelunestohyökkäystä, liiallisia oikeuksia ja väärin määritettyä tietokantaa ja kuinka näiltä voidaan suojautua.

Avainsanat: SQL-injektio, palvelunestohyökkäys, tietoturva, tietokanta, SQL, liialliset oikeudet

Abstract: Over the past thirty years, the IT environment has undergone many changes, and database research communities have tried to stay one step ahead of database security threats. The importance of security has been increased by the digitalization of companies' critical functions. Databases often contain sensitive information and are an essential part of all information systems. The most common database software is based on a relational model that can be searched in SQL. The information security of databases is important if a data leak occurs it is detrimental to the owner of the database and the object from which the informa-

tion was collected. This paper discusses SQL injection, denial of service attack, excessive privileges, and a misconfigured databases and how to protect against these threats.

Keywords: SQL injection, denial of service attack, information security, database, SQL, excessive permissions

Sisällys

1 Johdanto

Viimeisen kolmenkymmenen vuoden aikana tietotekniikkaympäristö on käynyt läpi monia muutoksia, ja tietokantatutkimusyhteisöt ovat yrittäneet pysyä askeleen edellä tietokannan turvallisuuteen kohdistuvista uhista. Yritysten kriittisten toimintojen digitalisoituminen on lisännyt turvallisuuden merkitystä tietokantatutkimuksessa. Tietokannat usein pitävät sisällään arkaluontoisia tietoja ja ne ovat olennainen osa kaikkia tietojärjestelmiä. Datan turvallisuus riippuu fyysisestä turvallisuudesta, käyttöjärjestelmän turvallisuudesta ja tietokannanhallintajärjestelmän turvallisuudesta. Tietokannan suojaus voi vaarantua pääsemällä käsiksi arkaluontoiseen tietoon, tietojen muuttumisella tai jos tietokannan saatavuus heikentyy (Lesov 2010). Kansainvälinen standardi ISO / IEC 27002 vuodelta 2005 määrittelee tietoturvan tietojen luottamuksellisuuden, eheyden ja saatavuuden säilyttämiseksi (Von Solms ja Van Niekerk 2013). Tietoturvauhat ovat uhkia, jotka vaarantavat jonkin osan tai kaikki standardin määritelmästä.

Tässä tutkielmassa käydään läpi, mikä tietokanta on ja mitä SQL tarkoittaa. Määritelmien jälkeen tutkitaan tietokantoihin liittyviä turvallisuusuhkia ja selvitetään, kuinka niiltä pystytään suojautumaan. Luvussa kaksi käsitellään termit tietokanta, SQL ja käydään hieman niiden historiaa läpi, sekä annetaan esimerkki SQL-lauseesta. Luvussa kolme käsitellään SQL-injektiota, selvitetään mikä se on, annetaan koodi esimerkki siitä, kerrotaan sen mahdollisista seurauksista ja esitellään tapoja, kuinka tältä turvallisuusuhalta voidaan suojautua. Neljännessä luvussa käydään läpi mikä on palvelunestohyökkäys, miten se toimii, minkälaisia vahinkoja se voi tuottaa ja kuinka siltä suojaudutaan. Viides luku käsittelee liiallisia oikeuksia ja oikeuksien hallintaa, luvussa kerrotaan kuinka voi hallita oikeuksia hyvin ja turvallisesti ja annetaan käytännön esimerkki liiallisesta oikeuksien käytöstä. Kuudennessa luvussa selvitetään minkälainen on huonosti tai väärin määritetty tietokanta ja mitä haittoja siitä voi seurata. Viimeinen ja seitsemäs luku on yhteenveto, jossa esitellään johtopäätökset ja tiivistetään tutkielman olennaisimmat asiat.

2 SQL-tietokanta

Tavallisimmin tietokanta määritellään datan ja informaation koosteeksi, joka on tietokoneen tallentama. Elmasrin ja Navathen (Elmasri ym. 2000) mukaan data tarkoittaa asioita, joita pystytään tallentamaan tai mittaamaan ja joilla on viitteellinen tarkoitus. Kokoelmaa, joka sisältää yhteenkuuluvaa dataa, kutsutaan tietokannaksi. Käytettäessä termiä tietokanta sen määritelmä on yleensä rajatumpi. Elmasrin ja Navathen määrittelevät, että tietokannalla tulee olla seuraavat implisiittiset ominaisuudet;

- Tietokanta edustaa jotakin tosielämän näkökohtaa.
- Tietokannan data kokoelman tulee olla johdonmukainen ja sillä tulee olla merkitystä.
- Tietokannan pitää olla suunniteltu ja sisältää tietoja jotain tiettyä tarkoitusta varten.

Tietokannalle on myös esitetty tarkentavia määritelmiä, kuten Darwen (Darwen 2009) tarkentaa tietokannan määritelmää. Tietokanta on organisoitu, koneellisesti luettavissa oleva symbolien kokoelma, joka on tulkittava yrityksen todellisena selostuksena ja tyypillisesti tietokanta on avoinna käyttäjien yhteisölle, mahdollisesti vaihtuvien vaatimusten kanssa. Määritelmissä korostuvat tietojen sijainti samassa paikassa ja on tärkeää muistaa, että tietokantoja voi olla esimerkiksi samalla yrityksellä useita.

Käyttäjä pääsee vaikuttamaan tietokantojen tietoihin erlaisista ohjelmistoista koostetulla tietokannan hallintajärjestelmällä. Arkikielessä yleensä tietokannan hallintajärjestelmän katsotaan sisältyvän sanaan tietokanta.

Tietokantojen historia alkaa aikaa ennen tietokoneita, kun tarvittiin tallettaa esimerkiksi potilastietoja paperisena. Tietokoneiden tultua maailmaan, tietokannat alkoivat muuttumaan sähköisiksi (Berg, Seymour ja Goel 2013). Tietokantojen relaatiomallin suunnitteli E.F Codd 1970-luvulla, ennen relaatiomallia tietokannat olivat linkitetyn listan tyylisiä. Relaatiomalli vaati, että sovellukset etsisivät tietoja sisällön mukaan, eikä linkkejä seuraamalla (Berg, Seymour ja Goel 2013). Relaatiomallissa tietokannan data jaetaan pienempiin osiin, joita kutsutaan relaatioiksi. Arkikieleen on yleistynyt termi SQL-tietokanta, mitä käytetään kun tarkoitetaan relaatiotietokantaa.

2.1 SQL

Merkittäviä tapahtumia informaatioteknologian alalla tapahtui 1990-luvulla kuten; internet, liiketoimintatiedon hallinnan ja data varastoimisen kasvaminen. Edellä mainituilla asioilla on yhteisenä ominaisuutena relaatiotietokantatekniikka ja lisänä tietokannanhallintajärjestelmätuotteita, jotka perustuvat kieleen nimeltä SQL (Melton 2003). SQL tulee sanoista structured query language ja sitä käytetään, kun halutaan käsitellä tai hakea tietokannan dataa. SQL on kyselykieli ja sen luomilla kyselyillä voidaan muokata, hakea, lisätä ja poistaa dataa tietokannasta (Alenezi, Nadeem ja Asif 2021).

Esimerkiksi haetaan hinta ja saatavuus tuotteista, joiden hinta on alle 100 euroa. SQL-lauseena tämän esimerkin rakenne on yleinen, jonka lisänä on ehtolause. Select-osa luettelee ne tiedot mitä halutaan hakea, From-osassa määritetään, mistä tauluista tiedot halutaan. Where-osa asettaa ehdon, tässä esimerkissä, jos tuotteen hinta on suurempi kuin 100 euroa niin sitä ei oteta mukaan tulostauluun.

```
SELECT Hinta, Määrä
FROM Tuote
WHERE Hinta < 100;
```

3 SQL-injektio

SQL-injektio on koodihyökkäys, jota käytetään erityisesti hyökättäessä verkkosovelluksiin tai verkkosivuille. Injektiossa hyökkääjä käyttää SQL-kyselykäskyä ja lisää siihen SQL-merkkejä tai avainsanoja, joka muuttaa aiotun kyselyn logiikkaa. Injektiohyökkäyksen uhka on mahdollinen jokaisessa verkkosovelluksessa, joka käyttää tietokantaa SQL-käskyjen kautta, jotka ovat koostettu ulkoisesta syötetiedosta. Hyväksikäyttämällä ulkoisesti syötettyä dataa, voi hyökkääjä muokata SQL-käskyjä, joka johtaa sovelluksen antamaan mielivaltaisia komentoja ja täten vaarantaa koko tietokannan (Singh ym. 2016). SQL-injektio voidaan tehdä esimerkiksi verkkosivun sisäänkirjautumiskentästä.

SQL-injektiosta on monia eri variaatioita, mutta keskeisenä teemana pysyy pahantahtoisen käyttäjän käyttävän tietoansa SQL-kielestä muokatakseen käskyänsä tietokannalle. Hyökkäyksen tavoite on lähettää tietokannalle kysely tavalla, johon sovelluksen ohjelmoija ei sitä ollut tarkoittanut (Buehrer, Weide ja Sivilotti 2005).

3.1 Esimerkki

Otetaan esimerkkinä verkkosivusto, johon pitää kirjautua. Kun käyttäjä1 kirjoittaa tietonsa sisäänkirjautumis- kenttään, joka asettaa tiedot SQL-kyselyksi, mikä voi näyttää esimerkiksi tältä:

```
SELECT * FROM käyttäjät
WHERE käyttäjänimi = 'käyttäjä1' and salasana = 'xxxxx'
```

Lause kysyy tietokannalta, onko siellä käyttäjää, jonka nimi on ”käyttäjä1” ja salasana ”xxxxx”. Tässä tulee huomata, että heittomerkki ei ole osa käyttäjän syötettä vaan osa SQL-kieltä. Heittomerkin avulla voidaan rikkoa SQL-syntaksi, jotta saadaan virheviesti. Virheviestin jälkeen hyökkääjä tietää sivun olevan altis injektiolle, joka voidaan suorittaa esimerkiksi näin:

```
SELECT * FROM käyttäjät
WHERE käyttäjänimi = 'käyttäjä1' or 1=1 -- and salasana = 'xxxxx'
```

Lauseen logiikka muuttuu ja se on aina tosi, koska yksi on samansuuruinen kuin yksi. Tulee huomata, että salasanan kysyminen sivuutetaan, koska syntaksin mukaan tässä se on muutettu hyökkääjän toimesta kommentiksi.

3.2 Mahdolliset seuraukset

SQL-injektio on yksi tuhoisimmista haavoittuvuuksista yritykselle ja sen tietokannoille, koska se voi johtaa kaiken arkaluonteisen tiedon vuotoihin ulkopuolisille (Clarke-Salt 2009). Nykypäivänä melkein jokaisella isolla yrityksellä on paikka, verkkosivu tai sovellus, joka usein käyttää relaatiotietokantaa (Buehrer, Weide ja Sivilotti 2005). Tietovuodot voivat aiheuttaa yritykselle tulon sekä maineen menetystä ja ihmisille jopa identiteettivarkauksia.

Singh, Dayal, Raw ja Kumar (Singh ym. 2016) määrittelevät tärkeimmät ominaisuudet, jotka liittyvät SQL-injektioon;

1. Todennuksen ohittaminen: Hyökkääjä, jolla ei ole aitoa käyttäjätunnusta tai salasanaa, voi saada pääsyn verkon kautta manipuloimalla SQL-käskyn logiikkaa.
2. Arkaluonteisen tiedon levittäminen: Kun hyökkääjä on saanut luvattoman pääsyn verkon kautta, hyökkääjällä on pääsy arkaluonteisen tiedon luo tietokannassa.
3. Datan eheyden menettäminen: Hyökkääjä ei ainoastaan muokkaa tietokannan tietoja vaan myös lisää sinne haitallista tietoa ja täten tietokannan eheys on vaarassa.
4. Datan saatavuuden menettäminen: Hyökkääjä saa täyden pääsyn järjestelmään ja poistaa tärkeän datan tietokannasta, mikä johtaa tiedon menetykseen ja datan saatavuus on menetetty.
5. Koodin etäsuorittaminen: Tässä hyökkääjä ei ainoastaan muokkaa olemassa olevaa dataa vaan lisää uusia käyttäjiä, joilla on täydet oikeudet, jolloin isännöintijärjestelmät ovat vaarantuneet.

3.3 Suojautuminen

On monia tapoja suojautua SQL-injektiolta. Yksi tapa on parantaa ohjelmointitekniikoita. Yleisinä käytäntöinä toimivat esimerkiksi yksittäisten lainausmerkkien välttäminen, ra-

ja syötteen merkkien määrässä, ja suodattamalla virheilmoitukset. Vaikka nämä ovat hyviä käytäntöjä niin niistä huolimatta haavoittuvuudet nousevat esiin (Boyd ja Keromytis 2004).

Toinen tapa suojautua SQL-injektiolta on jäsennyspuu. Jäsennyspuu on tietorakenne, joka sisältää SQL-lauseen jäsennellyn esityksen. Vertaamalla ja analysoimalla kahden lauseen jäsennyspuita pystytään selvittämään ovatko ne samanlaisia. Kun hyökkääjä syöttää onnistuneesti kyselyn tietokantaan, niin tarkoitetun kyselyn jäsennyspuu ja tuloksena syntyvän SQL-kyselyn jäsennyspuu eivät täsmää. Tarkoitetulla SQL-kyselyllä tarkoitetaan lausetta, jonka rakenteen tietokannan ohjelmoija on muotoillut ja kirjoittanut koodiin (Buehrer, Weide ja Sivilotti 2005).

Lwin Khin Shar ja Hee Beng Kuan Tanin mukaan SQL-injektiolta suojaavat keinot voidaan jakaa kolmeen eri luokkaan (Shar ja Tan 2013):

- Puolustava koodaaminen. Kyselyt ovat parametroitu ja menettelyt ovat tallennettu. Dynaamisten kyselyiden korvaaminen asianmukaisesti koodatuilla parametrisoiduilla kyselyillä tai tallennetut menettelyt saisivat kehittäjät ensin määrittämään SQL-koodin rakenteen ennen parametrien sisällyttämistä kyselyyn. Kun on määritetty SQL-rakenne johon parametrit ovat sidottu, ei ole mahdollista injektoida ylimääräistä SQL-koodia. Kehittäjien tulisi myös vahvistaa tietotyypit ennen niiden käyttöä. Esimerkiksi onko syöte merkkijono vai numeerinen. Syötteen tietotyyppien vahvistaminen auttaisi hylkäämän kelpaamattomat syötteet.
- SQLIV (SQL injection vulnerabilities) havaitseminen. Tutkijat ovat kehittäneet useita eri tapoja havaita SQL-injektiohaavoittuvuuksia. Koodipohjaisessa haavoittuvuustestauksessa tavoitteena on luoda riittävät testit, jotta havaitaan haavoittuvuudet. Tämä toimintatapa kuitenkin edellyttää myös manuaalista tarkastusta, koska se ei löydä haavoittuvia ohjelmapisteitä. MUSIC (mutation-based SQL injection vulnerability checking) on esimerkki koodipohjaisesta testauksesta. MUSIC käyttää yhdeksää mutaatio-operaattoria, joilla korvataan verkkosovelluksen alkuperäiset SQL-kyselyt mutatoiduilla kyselyillä.
- SQLIA (SQL injection attack) suorituksenaikainen ehkäiseminen. On kehitetty työkaluja sekä tekniikoita, joilla pystytään estämään kaikki SQL-injektiohyökkäykset tarkistamalla ohjelman ajonaikaa verrattuna oikeisiin SQL-kyselyihin. Kuitenkin tämä

tapa vaatisi koodin rakentamista, jotta se mahdollistaisi ajonaikaisen tarkistuksen, mikä voi tehdä tietoturvahaavoittuvuuksien virheenkorjauksista jopa vaikeampaa.

4 Palvelunestohyökkäys

Tietokoneiden verkostot ovat nykypäivänä tärkeä osa tietotekniikan infrastruktuuria. Tietokannatkin ovat usein verkossa toimivia alustoja, eikä vain yhdellä tietokoneella käytettäviä. Palvelunestohyökkäykset ovat nousseet yhtenä vakavimmista häiriöstä verkossa toimiville palveluille (Carl ym. 2006).

Palvelunestohyökkäys on hyökkäys, jossa tarkoitus on estää luvallisten käyttäjien mahdollisuutta käyttää jotakin verkkosivua tai sovellusta. Hyökkäys voi tapahtua erilaisilla tekniikoilla. Hyökkääjä voi saada pääsyn tietokantaan ja yrittää kaataa palvelimen. Resurssien ylikuormitus, verkontulva ja datan korruptio voivat olla tapoja, joilla luodaan olosuhteet palvelunestohyökkäykselle (Basharat, Azam ja Muzaffar 2012).

Palvelunestohyökkäyksen voi aloittaa tarkoituksenmukaisesti hyväksikäyttämällä uhrin järjestelmän haavoittuvuuksia esimerkiksi hukuttamalla uhrin suureen määrään turhia palvelupyyntöjä (Carl ym. 2006). Useimmat palvelunestohyökkäykset perustuvatkin ruuhkautumiseen, jossa hyökkääjä lähettää suuria määriä dataa uhrille (Dong, Yang ja Wang 2006).

4.1 Vahingot

Palvelunestohyökkäys hidastaa tietokantapalveluita ja voi tehdä käyttäjille mahdottomaksi käyttää palveluita. Se ei kuitenkaan paljasta tietokannan tietoja. Vaikka palvelunestohyökkäys ei paljastakaan tietoja niin se vie uhrilta paljon aikaa ja resursseja (Mousa, Karabatak ja Mustafa 2020). Ne eivät pelkästään vahingoita järjestelmien saatavuutta vaan uhkaavat niiden luottamuksellisuutta ja eheyttä (Carl ym. 2006).

Esimerkkinä palvelunestohyökkäyksenä on tietokanta, johon kirjaudutaan sisään ja kun käyttäjä on kirjannut salasanan liian monta kertaa väärin, tietokanta lukitsee tilin. Kun tili on lukittu, tulee käyttäjän odottaa tietty aika, kunnes voi yrittää sisäänkirjautumista uudelleen. Hyökkääjä voi käyttää tällaista haavoittuvuutta hyväkseen ja yrittää kirjautua tietokantaan sisään oikeilla käyttäjätunnuksilla mutta väärillä salasanoilla ja tämä johtaa siihen, että kukaan luvallinen käyttäjä ei pysty kirjautumaan tietokantaan sisään (Natan 2005).

4.2 Suojautuminen

On monta vastakeinoa palvelunestohyökkäystä vastaan. Esimerkiksi lisäämällä oikeat rekisteröintiasetukset maksimoidakseen tietoliikenne protokollan TCP:n yhteyslistan koon ja vahvistaa TCP / IP pinoa. Suojaavia toimenpiteitä on myös relaatioiden luomisen nopeuttaminen, dynaamisten rakenteiden käytöllä voidaan varmistaa, ettei jonon linkki ei koskaan ole loppuun käytetty. Tunkeutumisen havaitsemisjärjestelmän (IDS) käyttö auttaa myös paljastaamaan tunkeutujat (Mousa, Karabatak ja Mustafa 2020).

Thakare ja Kaur (Carl ym. 2006) jakavat palvelunestohyökkäyksen havaitsemisjärjestelmät kahteen kategoriaan;

1. Isäntäpohjaiset tunnistusjärjestelmät
2. Verkkopohjaiset tunnistusjärjestelmät

Isäntäpohjaisissa järjestelmissä, järjestelmän isäntä on vastuussa hyökkäysten havaitsemisessa ja havaitsemisjärjestelmä on otettu käyttöön isännän koneella. Verkkopohjaisessa tunnistusjärjestelmässä tulevaa tietoliikennettä seurataan hyökkäysten varalta. Verkkopohjaiset järjestelmät voidaan vielä luokitella kahteen eri luokkaan;

1. Väärinkäyttöön perustuvat tunnistusjärjestelmät
2. Poikkeavuuksiin perustuvat havaitsemisjärjestelmät

Väärinkäyttöön perustuvissa tunnistusjärjestelmissä on allekirjoitustietokanta, joka sisältää hyökkäysten ja sovellusten allekirjoitusmäärittelyt, joiden avulla voidaan tunnistaa sovelluksia palomuurikäytäntöjen seuraamiseksi. Tietoliikennettä seurataan ja tarkistetaan, onko yhtään paria allekirjoitustietokannan kanssa. Tämänkaltaiset järjestelmät toimivat hyvin jo tunnettuja hyökkäyksiä vastaan, mutta eivät tuntemattomia. Poikkeavuuksiin perustuvassa havainnointijärjestelmässä saapuvaa tietoliikennettä tutkitaan kaikenlaisen häiritsevän käyttäytymisen havaitsemiseksi. Tämän järjestelmän havaitsemisen nopeus on hyvä niin tunnettujen kuin tuntemattomien hyökkäysten tapauksessa.

Tehokas keino suojautua on yhteyden esto ja kieltäminen erityisesti, jos on mahdollista estää yhteys monen eri parametrin mukaan kuin vain esimerkiksi käyttäjätunnuksen. Yhteydeneston voi tehdä käyttämällä ulkoista turvallisuusjärjestelmää kuten tietokannan palomuuria.

Kun palomuuri on käytössä ja hyökkääjä yrittää esimerkiksi tehdä luvussa 4.1 esitettyä hyökkäystä, palomuuri ottaa hyökkääjän IP-osoitteen mistä palvelupyyntö tulee ja estää yhteyden muodostamisen siitä IP-osoitteesta. Näin hyökkääjä onnistuu ainoastaan estämään oman IP-osoitteen palvelupyynnöt eikä pysty vahingoittumaan luvallisia käyttäjiä (Natan 2005).

5 Liialliset oikeudet

Tietokannan käyttäjille määritetään käyttöoikeuksia, joita käyttämällä he voivat suorittaa tehtäviään. Eri käyttäjille voivat olla erilaiset käyttöoikeudet tietokantaan, esimerkiksi mitä tietoja näkee ja mitä tietoja voi muokata. Ne oikeudet mitkä käyttäjälle suodaan, riippuu käyttäjän tehtävistä. Esimerkiksi jos käyttäjä haluaa muuttaa järjestelmän aikaa, niin hän tarvitsee siihen ajan vaihtamisoikeuden (Shen ym. 2013).

Käyttäjälle saatetaan virheellisesti määrittää käyttöoikeudet, jotka sallivat hänet suorittamaan hänen toimeksiantoonsa kuulumattomia tehtäviä. Sellaisten tehtävien kautta, jotka eivät kuulu käyttäjän työkuvaan, voidaan löytää haitallinen aikomus, tällaisten etuoikeuksien väärinkäyttöön (Basharat, Azam ja Muzaffar 2012).

Tietokannan käyttöoikeuksia voi hyväksikäyttää monella eri tavalla luvattomiinkin tarkoituksiin. Liiallisten käyttöoikeuksien myöntäminen on Malikin ja Patelin (Malik ja Patel 2016) mukaan ongelmallista kahdesta syystä. Noin kahdeksankymmentä prosenttia hyökkäyksistä, jotka kohdistuvat yrityksen dataan ovat yrityksen nykyisen tai entisen työntekijän tekemiä. Liiallisten oikeuksien antaminen tai niiden mitätöinti liian myöhään tekevät heidän väärinkäytöksistään tarpeettoman yksinkertaiset toteuttaa.

5.1 Suojautuminen

Eri roolien oikeudet voidaan esittää hierarkkisesti, korkeimpana olevalla oikeudella käyttäjä pystyy tekemään mitä tahansa ja näkemään kaiken. Kun hierarkiassa mennään alaspäin niin käyttäjiltä jää pois joitakin oikeuksia ja näkyvyyksiä.

Pienimmän oikeuden periaatetta on hyvä käyttää, eli käyttäjän oikeudet tulisi olla mahdollisimman niukat mutta kuitenkin sellaiset, että tehtävä on suoritettavissa. Tämä takaisi sen, ettei käyttäjä pystyisi tekemään mitään muuta kuin oleellista tehtävän suorittamisessa. Periaate vähentää mahdollisia tahattomia, ei toivottuja tai epäasiallisia oikeuksien käytöksiä, joten niiden esiintymisen todennäköisyys laskee (Schneider 2003).

Sen lisäksi, ettei käyttäjälle anneta liian hyviä oikeuksia lisävarmuutta suojautumiseen tuo

hyvät kirjausketjut. Kirjausketju on ketju, jossa on esimerkiksi kirjattu tietoja tapahtumista ja niistä tuotetusta tiedosta. Kirjausketjun ansioista tietokannan järjestelmävalvojat pystyvät puuttumaan ajoissa käyttöoikeuksien väärinkäyttöön (Pevnev ja Kapchynskyi 2018).

5.2 Käytännön esimerkki

Esimerkkinä liiallisista oikeuksista voidaan käyttää yliopiston ylläpitäjää, jonka tehtävä vaatii ainoastaan oikeuksia vaihtaa opiskelijan yhteystietoja, mutta jos ylläpitäjällä on enemmän oikeuksia kuin hän tarvitsee hän voi käyttää sitä hyväksi ja esimerkiksi vaihtaa oppilaiden arvosanoja. Tietokannan käyttäjälle päätyy liiallisia oikeuksia hyvinkin yksinkertaisesta syystä. Tämä syy voi olla, että tietokannan järjestelmävalvojalla ei ole aikaa määrittää ja päivittää käyttöoikeuksia yksittäisille käyttäjille ja tämän seurauksena oikeudet myönnetään oletusarvoisesti käyttäjäryhmille, jotka voivat ylittää tarvitut vaatimukset tehtäville (Pevnev ja Kapchynskyi 2018).

6 Väärin määritetty tietokanta

Tietokantajäjestelmillä on suuri määrä kokoonpanoparametreja, jotka ohjaavat muistin jakelua, I/O-optimointia ja monia näkökohtia kirjaamiseen, palautukseen, sekä muuhun käyttäytymiseen (Duan, Thummala ja Babu 2009). On tavallista löytää haavoittuvia tai päivittymättömiä tietokantoja tai tietokantoja, jotka vielä sisältävät oletustilit sekä määritysparametrit. Nämä ovat asioita, joita hyökkääjät osaavat käyttää hyväksi hyökättäessä organisaation tai yrityksen kimppuun (Malik ja Patel 2016).

Organisaatioilla on yleensä vaikeuksia pysyä tahdissa tietokantamääritysten ylläpidossa, vaikka olisikin saatavilla korjaustiedostoja. Useimmiten tämä johtuu suurista työkuormista ja aikaa vievistä vaatimuksista korjaustiedostojen testaamiseen, mistä aiheutuu vaikeus löytää aikaikkuna, jossa yrityskriittinen järjestelmä suljetaan päivitysten tai korjausten ajaksi. Tämä vaikeus johtaa siihen, että yrityksillä voi kestää jopa kuukausia tietokantojen korjaaminen ja näinä aikoina ne ovat haavoittuvia (Ahirwar, Saxena ja Yadav, n.d.).

Tyypilliset tietokannanhallintajärjestelmät tarjoavat useita mekanismeja tietojensuojaamiseksi, esimerkiksi käyttäjänoikeudet, kuitenkin useimmissa järjestelmissä suojausmekanismien tehokkuus riippuu suurelta osin niiden määrityksistä (Araújo Neto ja Vieira 2008). Esimerkiksi riskejä aiheuttaa liian suuret käyttäjäoikeudet, tietokantaan jääneet oletustilit ja salaamatta tai varmuuskopioimatta oleva data.

6.1 Huonosti määritellyn tietokannan seuraukset

Jos tietokannantoimittaja ei ole julkaissut korjaustiedostoa ja tietokannassa on haavoittuvuuksia, ne voivat altistaa korkean riskin hyökkäyksiin kuten SQL-injektio ja palvelunestohyökkäys. Virtuaalisten korjaustiedostojen käyttö estää haavoittuvuuksien hyödyntämisyritykset vaatimatta todellisia korjauksia tai muutoksia tietokannan tai palvelimen määrityksiin. Virtuaalinen korjaustiedosto suojaa tietokantaa haavoittuvuuksien hyväksikäyttöyrityksiltä, kunnes oikea korjaustiedosto on saatu käyttöön (Ahirwar, Saxena ja Yadav, n.d.).

Mikäli tietokannan tai järjestelmän turvallisuusasetukset ovat määritelty väärin tai huonos-

ti, ne jättävät hyökkääjille mahdollisuuden hyödyntää haavoittuvuuksia usealla eri tavalla ja laukaista erilaisia hyökkäyksiä, jotka eivät rajoitu vain injektioon tai palvelunestohyökkäykseen. Turvallisuusasetusten väärin määrittäminen voi johtaa riskeihin, kuten luvaton pääsy järjestelmätietoihin ja toimintoihin tai täydellinen järjestelmän vaarantuminen (Eshete, Villafiorita ja Weldemariam 2011).

7 Yhteenveto

Tutkielmassa selvennettiin, mitä tarkoittaa tietokanta ja SQL ja käsiteltiin neljää siihen kohdistuvaa tietoturvauhkaa, jotka voivat aiheuttaa haittaa tietokannan käyttäjille, sekä sen haltijalle.

Jos esimerkiksi verkkosivulla kirjaudutaan sisään kentästä, missä ei ole riittäviä tarkistimia syötteille, altistaa tämä verkkosivun käyttämän tietokannan SQL-injektiolle. Injektiolla hyökkääjä voi päästä käsiksi tietokannan tietoihin ja aiheuttaa jopa tietovuodon. Injektiolta voidaan suojautua monella eri tavalla kuten parametroiduilla kyselyillä ja rajaamalla syötteen pituutta. Palvelunestohyökkäykset vahingoittavat tietotekniikan infrastruktuuria ja estävät luvallisten käyttäjien mahdollisuutta esimerkiksi päästä tietokannan tietoihin käsiksi ja täten vahingoittaa tietojen saatavuutta. Palvelunestohyökkäykset voidaan estää käyttämällä ulkoista turvallisuusjärjestelmää kuten palomuuria ja estää hyökkääjän IP-osoitteesta tulevat ruuhkauttavat palvelupyynnöt.

Kun tietokannan käyttäjille määritellään liian laajat käyttöoikeudet, kuin hänen tehtävänsä vaatisivat, puhutaan liiallisista käyttöoikeuksista. Hyökkäyksistä, jotka kohdistuvat yrityksen dataan, suurimmassa osassa tekijänä on yrityksen nykyinen tai entinen työntekijä. Liiallisten oikeuksien antaminen tekee heidän väärinkäytöksistään tarpeettoman yksinkertaisia toteuttaa. Hyvä periaate, joka suojaa liiallisilta oikeuksilta on pienimmän oikeuden periaate.

Huonosti määritetyllä tietokannalla tarkoitetaan tietokantaa mikä sisältää oletusasetuksia ja tilejä tai päivittämätöntä tietokantaa. Jotta yritys saa tietokannan päivitettyä voi aikaa kulua useita kuukausia, joten tietokantaa voi suojata virtuaalisella korjaustiedostolla, joka estää haavoittuvuuksien hyödyntämisyritykset. Mikäli tietokannan turvallisuusasetukset ovat määritelty väärin tai huonosti niin ne jättävät hyökkääjille mahdollisuuksia esimerkiksi SQL-injektioon tai palvelunestohyökkäykseen.

Lähteet

Ahirwar, Rekha, Rashi Saxena ja Pankaj Yadav. n.d. “Challenges to Data Base Security–A Futuristic View”.

Alenezi, Mamdouh, Muhammad Nadeem ja Raja Asif. 2021. “SQL injection attacks countermeasures assessments”. *Indonesian Journal of Electrical Engineering and Computer Science* 21 (2): 1121–1131.

Araújo Neto, Afonso, ja Marco Vieira. 2008. “Towards Assessing the Security of DBMS Configurations”, 90–95. Heinäkuu. ISBN: 978-1-4244-2397-2. https://doi.org/10.1109/DSN.2008.4630074.

Basharat, Iqra, Farooque Azam ja Abdul Wahab Muzaffar. 2012. “Database security and encryption: A survey study”. *International Journal of Computer Applications* 47 (12).

Berg, Kristi L, Tom Seymour ja Richa Goel. 2013. “History of databases”. *International Journal of Management & Information Systems (IJMIS)* 17 (1): 29–36.

Boyd, Stephen W., ja Angelos D. Keromytis. 2004. “SQLrand: Preventing SQL Injection Attacks”. Teoksessa *Applied Cryptography and Network Security,* toimittanut Markus Jakobsson, Moti Yung ja Jianying Zhou, 292–302. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-24852-1.

Buehrer, Gregory, Bruce Weide ja Paolo Sivilotti. 2005. “Using parse tree validation to prevent SQL injection attacks”, 106–113. Tammikuu. https://doi.org/10.1145/1108473.1108496.

Carl, G., G. Kesidis, R. R. Brooks ja Suresh Rai. 2006. “Denial-of-service attack-detection techniques”. *IEEE Internet Computing* 10 (1): 82–89. https://doi.org/10.1109/MIC.2006.5.

Clarke-Salt, Justin. 2009. *SQL injection attacks and defense.* Elsevier.

Darwen, Hugh. 2009. *An introduction to relational database theory.* Bookboon.

Dong, K., S. Yang ja S. Wang. 2006. “Analysis of low-rate TCP DoS attack against FAST TCP”. Teoksessa *Sixth International Conference on Intelligent Systems Design and Applications,* 3:86–91. Lokakuu. https://doi.org/10.1109/ISDA.2006.12.

Duan, Songyun, Vamsidhar Thummala ja Shivnath Babu. 2009. “Tuning database configuration parameters with iTuned”. *Proceedings of the VLDB Endowment* 2 (1): 1246–1257.

Elmasri, R, Shamkant B Navathe, R Elmasri ja SB Navathe. 2000. *Fundamentals of Database Systems</Title.* Springer.

Eshete, B., A. Villafiorita ja K. Weldemariam. 2011. “Early Detection of Security Misconfiguration Vulnerabilities in Web Applications”. Teoksessa *2011 Sixth International Conference on Availability, Reliability and Security,* 169–174. Elokuu. https://doi.org/10.1109/ARES.2011.31.

Lesov, Paul. 2010. “Database Security: A Historical Perspective”. *arXiv preprint arXiv:1004.4022.*

Malik, Mubina, ja Trisha Patel. 2016. “Database Security - Attacks and Control Methods”. *International Journal of Information Sciences and Techniques* 6 (maaliskuu): 175–183. https://doi.org/10.5121/ijist.2016.6218.

Melton, 1946-, Jim. 2003. *Advanced SQL, 1999 : understanding object-relational and other advanced features.* Toimittanut 1946- Melton Jim. 563. Morgan Kaufmann series in data management systems. Amsterdam ; Boston, Mass.: Morgan Kaufmann Pub. http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=210503.

Mousa, A., M. Karabatak ja T. Mustafa. 2020. “Database Security Threats and Challenges”. Teoksessa *2020 8th International Symposium on Digital Forensics and Security (ISDFS),* 1–5. Kesäkuu. https://doi.org/10.1109/ISDFS49300.2020.9116436.

Natan, Ron Ben. 2005. *Implementing database security and auditing.* Elsevier.

Pevnev, Vladimir, ja Serhii Kapchynskyi. 2018. “DATABASE SECURITY: THREATS AND PREVENTIVE MEASURES”. *Advanced Information Systems* 2 (toukokuu): 69–72. https://doi.org/10.20998/2522-9052.2018.1.13.

Schneider, F. B. 2003. “Least privilege and more [computer security]”. *IEEE Security Privacy* 1, numero 5 (syyskuu): 55–59. ISSN: 1558-4046. https://doi.org/10.1109/MSECP.2003.1236236.

Shar, L. K., ja H. B. K. Tan. 2013. “Defeating SQL Injection”. *Computer* 46 (3): 69–77. https://doi.org/10.1109/MC.2012.283.

Shen, Mou, Mengdong Chen, Min Li ja Lianzhong Liu. 2013. “Research of least privilege for database administrators”. *International Journal of Database Theory and Application* 6 (6): 39–50.

Singh, Nanhay, Mohit Dayal, RS Raw ja Suresh Kumar. 2016. “Sql injection: Types, methodology, attack queries and prevention”. Teoksessa *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom),* 2872–2876. IEEE.

Von Solms, Rossouw, ja Johan Van Niekerk. 2013. “From information security to cyber security”. *computers & security* 38:97–102.