Atte Juhani Rajalahti

Perehdytys SQL-injektioihin sekä niitä vastaan toimimiseen

Tietotekniikan kandidaatintutkielma

20. kesäkuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Atte Juhani Rajalahti

Yhteystiedot: atjuraja@student.jyu.fi

Ohjaaja: Tuomo Rossi

Työn nimi: Perehdytys SQL-injektioihin sekä niitä vastaan toimimiseen

Title in English: Introduction to SQL injections and how to work against them

Työ: Kandidaatintutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 22+0

Tiivistelmä: Nykymaailmassa useat organisaatiot säilyttävät kriittistä dataa erinäisissä tietokannoissa. Vahinkoa tavoittelevat hyökkääjät pyrkivät joko saamaan käyttöoikeuksia, tuottamaan vahinkoa, tai palauttamaan arkaluontoista dataa näistä tietokannoista. Yksi yleisimmistä hyökkäystavoista ovat injektiot, ja näistä yksi esimerkki on SQL-kyselykieltä hyväkseen käyttävät SQL-injektiot. Tässä tutkielmassa käsitellään mitä SQL-injektiot ovat, miten ne toimivat, sekä miten niitä vastaan voidaan turvautua.

Avainsanat: SQL, tietokanta, ohjelmistoturvallisuus, tietokantaturvallisuus, SQL-injektiot

Abstract: In today's world, many organizations store critical data in various databases. Malicious attackers aim to either gain access rights, cause damage, or recover sensitive data from these databases. One of the most common attack methods are injections, and one example of these is SQL injections exploiting the SQL query language. This treatise discusses what SQL injections are, how they work, and how to protect against them.

Keywords: SQL, database, software security, database security, SQL injections

Kuviot

Sisällys

1 Johdanto

Samalla kun nykymaailma muuttuu vuosi vuodelta digitaalisemmaksi, kasvaa myös tarve säilyttää suuria määriä dataa. Erinäiset tietokannat on todettu tämänhetkisessä maailmassa parhaaksi ratkaisuksi tähän ongelmaan, ja useat organisaatiot ja yritykset turvautuvatkin tietokantoihin datan varastoinnin kohdalla. Koska näissä tietokannoissa usein säilytetään yritysten liiketoiminnan kannalta kriittistä ja elintärkeää dataa, ovat ne usein otollinen kohde mahdollisille pahantahtoisille hyökkääjille, jotka pyrkivät aiheuttamaan joko vahinkoa itse tietokannalle, tai palauttamaan sieltä mahdollista kriittistä dataa. Yksi yleisimmistä hyökkääjien käyttämistä keinoista palauttaa tietokannasta salaista informaatiota on hyödyntää SQL-injektioita, jotka ovat historiallisesti aiheuttaneet jopa miljoonien eurojen vahinkoja suurempien organisaatioiden kohdalla. Jotta SQL-injektioilta voitaisiin turvautua mahdollisimman hyvin, on tärkeää ymmärtää miksi ja miten ne toimivat, sekä kuinka tietokanta voidaan suojata niiden toimintaa vastaan.

Tämä kandidaatintutkielma antaa yleiskatsauksen SQL-injektioihin, sekä kuinka niitä vastaan kannattaisi toimia. Aluksi lukijalle selvitetään relaatiotietokannan ja SQL-kielen rakennetta yleisellä tasolla, sekä kerrotaan pikaisesti mitä injektiot ovat, jotta voidaan paremmin ymmärtää SQL-injektion rakennetta ja toimintaa (luku 2). Tämän jälkeen käydään tarkemmin läpi SQL-injektiota, käsitellään se yleisellä tasolla, sekä käsitellään millaisia SQL-injektion eri tyypit ovat (luku 3). Seuraavaksi käsitellään miten tehokkaita erilaiset SQL-injektio haavoittuvuuden havaitsemistekniikat ovat, sekä arvioiden niiden huonoja puolia, jonka jälkeen käydään läpi SQL-injektioiden ehkäisemiseen liittyviä hyväksi todettuja tekniikoita (luku 4). Lopuksi vielä esitellään muutama viime vuosikymmenellä tapahtunut SQL-injektio tapaus, sekä selvitetään millaista vahinkoa ja mille tahoille ne ovat tapahtuneet (luku 5).

2 Taustatietoja

Jotta voidaan ymmärtää SQL-injektiot ja niiden toiminta, on tärkeää ymmärtää sekä tietokantojen, että SQL-kielen rakenne. Tässä luvussa käydään läpi tietokannat, yksinkertainen esimerkki relaatiotietokannan tauluista, sekä SQL-kyselykieli yksinkertaisella tasolla.

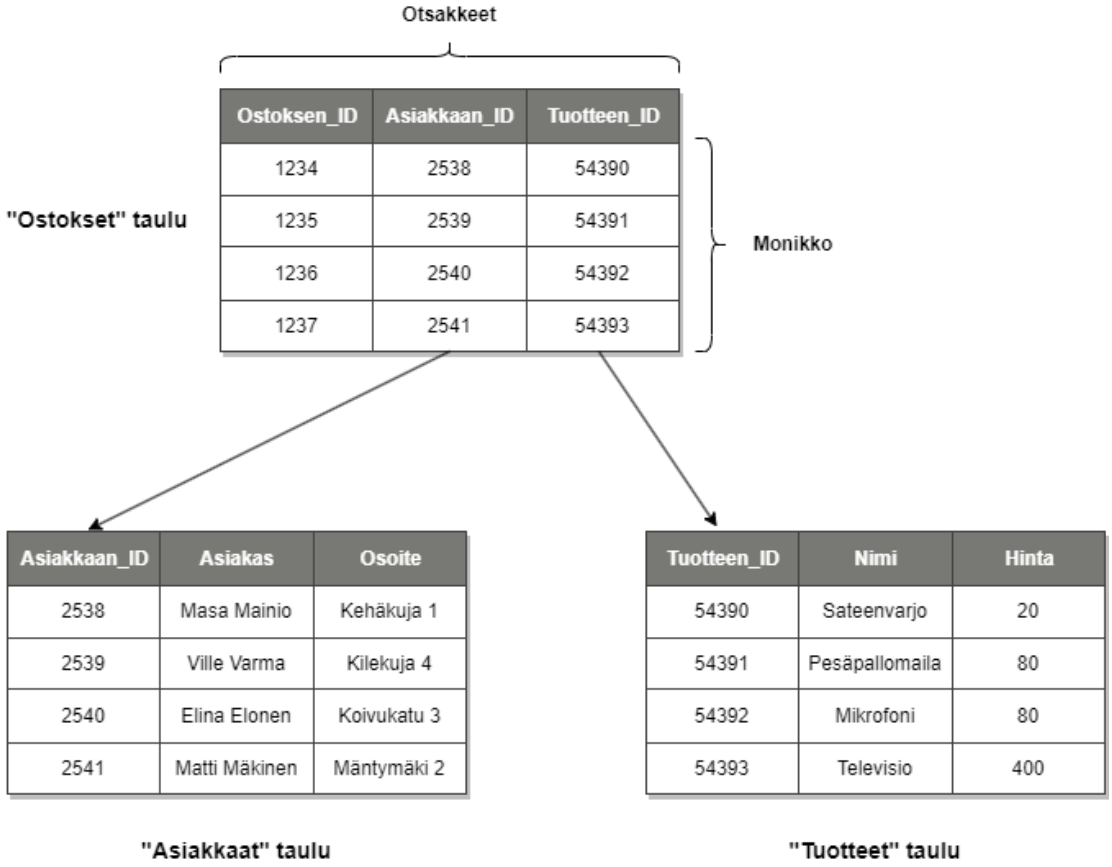
2.1 Tietokanta

Tietokannat ovat järjesteltyjä kokoelmia jäsenneltyä tietoa tai dataa, ja ne ovat yleisin tapa säilyttää dataa yritysten ja organisaatioiden keskuudessa. Usein ne voivat sisältää yrityksen kannalta kriittistä dataa, kuten esimerkiksi käyttäjätunnuksia ja salasanoja, joten niiden hyvin suojaamista voidaan pitää organisaation toiminnan kannalta elintärkeänä. Säilytettävän datan määrän noustessa suuremmaksi nousee myös tietokantojen tarve, sillä niitä pidetään parhaimpana tapana säilyttää suuria määriä dataa tehokkaasti.

Eri tietokantatyyppejä on satoja, mutta näistä yleisimpänä pidetään relaatiotietokantoja (DB-Engines 2023). Relaatiotietokannat käyttävät lähes poikkeuksetta SQL-kieltä kyselykielenään, joten on tärkeää ymmärtää myös relaatiotietokannan rakennetta yksinkertaisella tasolla, jotta ymmärtäisi hyvin sekä SQL-kielen, että SQL-injektioiden toiminnan.

Relaatiotietokantojen teoreettinen perusta perustuu Edgar F. Coddin alun perin vuonna 1970 julkaisemaan teoriamalliin (Codd 1983). Relaatiotietokannoissa dataa säilytetään tauluissa (engl. Table), jotka sisältävät otsakkeita (engl. Header). Nämä otsakkeet sisältävät yksittäisiä attribuuttien arvoja, joiden monikko muodostaa monikon (engl. Tuple). Jotta relaatiotietokannassa voidaan muodostaa relaatio taulujen välille, täytyy tauluista valita viiteavain (engl. Foreign Key). Taulun viiteavaimeksi voidaan valita taulun attribuuttijoukko, jonka arvot ovat myös toisen taulun attribuuttijoukko. Nämä relaatiotietokannan viiteavaimet mahdollistavat taulujen väliset yhteydet, sekä niiden välillä liikkumisen SQL-kielen avulla.

Alla esitetty esimerkki relaatiotietokannan tauluista ja niiden välisistä yhteyksistä, joita kuvataan taulujen välisillä nuolilla. Mukailtu lähteestä Researchgate (2016).



Kuvio 1. Esimerkki relaatiotietokannasta.

2.2 SQL

SQL (engl. Structured Query Language) on yhdysvaltalaisen IBM:n kehittämä kyselykieli, jonka kautta voidaan hallinnoida ja käsitellä relaatiotietokantoja. Se kehitettiin 1970-luvun alussa, ja siitä on tullut maailman yleisin tietokantojen hallinnointiin käytettävä kyselykieli (StackOverflow 2020). SQL-kielen syntaksi sisältää mm. englanninkielisiä sanoja, vertailuoperaattoreita, sekä erikoismerkkejä, joita yhdistelemällä luodaan käskyjä, joiden avulla tietokantaa hallinnoidaan, sekä sieltä haetaan dataa. Tarkastellaan SQL-kielen rakennetta yksinkertaisen esimerkin kautta.

```
SELECT *
FROM tuotteet
WHERE Tuotteen_ID>54390 AND Hinta<200
```

```
ORDER BY Nimi;
```

Edellä esitetyn esimerkin ensimmäisellä rivillä SELECT käskyllä käsketään palauttamaan tietokannasta kaikki rivit, jotka täyttävät lauseen seuraavan kahden rivin ehdot. Toisella rivillä rajataan mihin tietokannan tauluista haluamme hakumme kohdistuvan, tässä tapauksessa tietokannasta löytyvä tuotteet taulu. Kolmannella rivillä lisätään kaksi ylimääräistä vaatimusta tietokannalle lähtevään käskyyn, joissa tarkennetaan, että TuotteenID:n täytyy olla suurempi kuin 54390, sekä hinnan täytyy olla vähemmän kuin 200. Lopuksi viimeisellä rivillä käsketään järjestämään palautetut rivit nousevaan suuruusjärjestykseen nimen mukaan. Joten kokonaisuudessaan kysely palauttaa tietokannasta tuotteet taulusta kaikki rivit, joissa tuotteen ID on suurempi kuin 54390, sekä hinta on vähemmän kuin 200, ja lopuksi ne järjestetään nousevaan suuruusjärjestykseen nimen mukaan.

SQL:stä löytyvillä komennoilla voidaan myös muokata tietokannan taulujen rakennetta, sekä niiden sarakkeiden arvoja. Näistä komennoista yleisimpiä ovat ”UPDATE”, jolla muutetaan tietyn taulun sarakkeen arvoa, ”INSERT”, jolla lisätään taulurivi jo olemassa olevaan tauluun, sekä ”DELETE”, jolla voidaan poistaa taulusta tietty taulurivi.

3 SQL-Injektio

Tässä luvussa käsitellään yleisellä tasolla mitä SQL-injektiot ovat, miltä yksinkertainen esimerkki SQL-injektiosta näyttää, sekä mihin eri kategorioihin SQL-injektiot voidaan jakaa.

3.1 Yleisesti SQL-Injektioista

SQL-injektiot ovat injektioita, jotka hyödyntävät SQL-kyselykieltä apunaan päästäkseen muokkaamaan ja tutkimaan verkkosovelluksen takana sijaitsevaa tietokantaa. Injektio viittaa hyökkäystapaan, jossa hyökkääjä syöttää järjestelmälle sen loogisen toiminnan kannalta haitallista syötettä, ja pyrkii näin saamaan järjestelmän suorittamaan komentoja, jotka ovat hyökkääjälle hyödyllisiä, kuten käyttöoikeuksien muuntaminen, tai salaisen informaation paljastaminen. Verkkosovelluksen ja tietokannan välisessä kommunikaatiossa tietokanta odottaa saavansa SQL-kyselykielen syntaksia noudattavia käskyjä, ja näitä verkkosovelluksen kautta lähetettyjä manipuloituja, mutta syntaksisesti korrekteja SQL-kyselyitä apuna käyttäen hyökkääjä voi ohittaa sovellusten väliset turvatoimet. Esitetään yksinkertainen esimerkki SQL-kyselystä, sekä kuinka tätä kyselyä manipuloimalla voidaan suorittaa yksinkertainen SQL-injektio.

```
SELECT *
FROM users
WHERE name = '" + userName + "' AND password = '" + password + "'
```

Edellä esitetty SQL-lause toimii yksinkertaisena esimerkkinä lauseesta, jolla verkkosivu hakee tietokannasta käyttäjän tiedot. Verkkosovellus odottaa, kunnes käyttäjä syöttää käyttäjänimelle ja salasanalle varattuihin pakkoihin tiedot, jonka jälkeen ne syötetään ennalta luotuun SQL-lauseeseen niille tarkoitetuille paikoille, jonka jälkeen suoritetaan haku tietokannasta. Ongelma lauseen kanssa syntyy, jos hyökkääjä tuntee SQL-kielen, sillä sekä käyttäjätunnuksen, että salasanan tarkistuksen voidaan tämän esimerkin tapauksessa ohittaa yksinkertaisesti, esimerkiksi syöttämällä molempiin kenttiin syötteeksi "' OR '1'='1". Näin ollen tietokannalle lähtevän SQL-lauseen uusi viimeinen rivi näyttäisi seuraavalta.

```
SELECT *
FROM users
WHERE name = '' OR '1'='1' AND password = '' OR '1'='1'
```

Tässä tapauksessa molemmat oikeellisuus tarkistukset on läpäisty, ja tietokanta palauttaa hyökkääjälle ”users” taulusta kaikkien käyttäjien kaikki tiedot.

3.2 Erilaiset SQL-Injektiot

Erityyppiset SQL-injektiot voidaan luokitella eri kategorioihin monilla eri perusteilla, mutta yleisimpiä tapoja on luokitella ne hyökkäyksen tarkoituksen sekä injektiomekanismin perusteella (Nasereddin ym. 2021). Tässä osiossa erityyppiset SQL-injektiot on luokiteltu kolmeen eri kategoriaan: Kaistan sisäiset SQL-injektiot (engl. In-band SQLi), sokeat SQL-injektiot (engl. Blind SQLi), sekä kaistan ulkopuoliset SQL-injektiot (engl. Out-of-band SQLi) (Rai ym. 2021).

Kaistan sisäiset SQL-injektiot viittaavat SQL-injektiotyyppeihin, joissa käytetään samaa viestintäkanavaa sekä hyökkäyksen suorittamiseen, että tietokannalta saatujen vastausten vastaanottamiseen. Tätä injektiotyyppiä voidaan pitää yksinkertaisimpana ja nopeimpana SQL-injektiotyyppinä (Rai ym. 2021). Kaistan ulkopuoliset SQL-injektiot sen sijaan viittaavat SQL-injektioihin, joissa hyökkääjä käyttää eri viestintäkanavaa hyökkäyksen suorittamiseen ja vastausten vastaanottamiseen. Sokeissa SQL-injektioissa hyökkääjä ei vastaanota tietokannalta takaisin SQL-kyselyn vastausta, eikä mitään tietoa liittyen tietokannassa tapahtuneeseen virheeseen. Sokea SQL-injektiohyökkäys on yleisesti tehottomampi, ja se vie hyökkääjältä enemmän aikaa, mutta sitä pidetään silti yhtä vaarallisena kuin muitakin SQL-injektioita.

3.2.1 Kaistan sisäiset SQL-injektiot

Kaistan sisäisten SQL-injektioiden (joskus myös klassiset SQL-injektiot) tyypillisimpinä esiintyminä pidetään virheperusteisia SQL-injektioita (engl. Error-based SQLi) sekä yhdistysoperaation SQL-injektiota (engl. Union-based SQLi).

Virheperusteisessä SQL-injektiossa hyökkääjän tavoitteena on lähettää tietokannalle syötettä, joka tuottaa virheilmoituksen. Palautetusta virheilmoituksesta saadun informaation perusteella hyökkääjä voi saada selville esimerkiksi tietokannan rakennetta. Virheperusteista SQL-injektiota vastaan turvautumisen kannalta on tärkeää, ettei käyttäjälle palauteta suoraan järjestelmän tuottamaa virheilmoitusta, vaan sen sijaan palautetaan jokin ohjelmoijan itse luoma virheilmoitus, josta ei ilmene tietokannan rakenteeseen tai tietoturvaan liittyviä ominaisuuksia.

Yhdistysoperaation SQL-injektiossa hyökkääjä käyttää SQL-kielen UNION operaattoria hyväkseen, jotta hänelle palautettaisiin enemmän dataa kuin mitä alkuperäiseltä kyselyltä odotettiin. UNION operaattorilla voidaan yhdistää useita SELECT kyselyitä yhdeksi kyselyksi, ja tätä tekniikkaa hyväksi käyttämällä hyökkääjä voi mahdollisesta palauttaa dataa tauluista, joihin hänellä ei alun perin ollut oikeuksia.

3.2.2 Sokeat SQL-injektiot

Totuusarvoihin perustuvassa SQL-injektiossa (engl. Boolean-based SQLi) hyökkääjä lisää alkuperäiseen kyselyyn totuusarvoisia ehtolauseita, joista saatujen vastausten perusteella hän pystyy arvioimaan tietokannan rakennetta ja toimintaa. Tätä tekniikkaa käyttämällä on mahdollista saada selville esimerkiksi taulujen otsakkeiden nimiä, sekä yksittäisten attribuuttien arvoja, sillä hyökkääjä pystyy käymään jokaisen merkin yksi kerrallaan läpi vertaillen sitä itse syötettyyn merkkiin, kunnes totuusarvon vertailu palauttaa arvon tosi. Sokeisiin SQL-injektioihin perustuvaa tekniikkaa hyödyntäen löytyy myös automatisoituja työkaluja, kuten SQLmap sekä SQLninja (Rai ym. 2021).

Ajan viivästys SQL-injektio (engl. Time-based SQLi) on totuusarvoihin perustuvan SQL-injektion kanssa hyvin samantapainen, mutta sen sijaan että odotetaan tosi tai epätosi vastauksia ja tehdään johtopäätöksiä niiden perusteella, käsketään tietokantaa odottamaan vastauksen kanssa, jos syöttämämme kysely on oikein. Esimerkiksi jos salasanan ensimmäinen merkki on ’a’, pyydetään tietokantaa odottamaan 5 sekuntia, mutta jos merkki ei olekaan ’a’, pyydetään tietokantaa odottamaan vain 2 sekuntia. Viiveestä päättelemällä voidaan arvioida kyselyn oikeellisuus, ja näin käymällä läpi merkki kerrallaan voidaan saada haltuun

käyttäjän salasana.

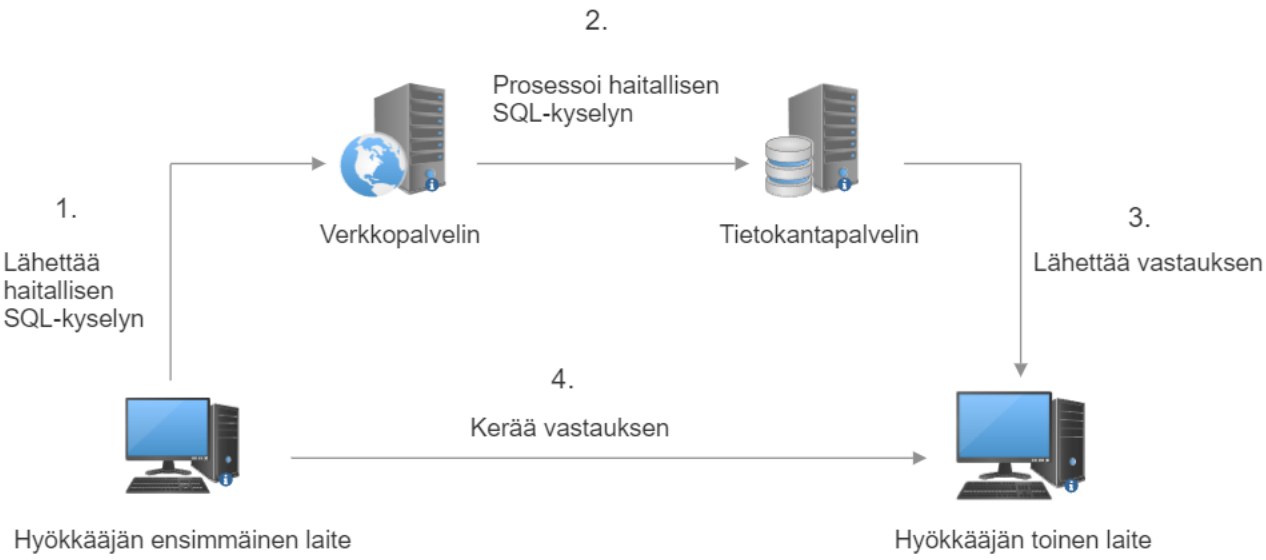
3.2.3 Kaistan ulkopuoliset SQL-injektiot

Kaistan ulkopuoliset SQL-injektiot eroavat kaistan sisäisistä siinä, että niissä hyökkääjä ei vastaanota vastausta tietokannalta samaa viestintäkanavaa hyödyntäen, jolla hän kommunikoi tietokannalle. Sen sijaan hyökkääjä ohjaa tietokannan palauttamat datat ja virheilmoitukset toiselle hänen hallitsemalleen laitteelle. Kaistan ulkopuoliset SQL-injektiot hyödyntävät tietokannan ominaisuutta, jolla voidaan lähettää dataa esimerkiksi sähköpostina, tai HTTP-protokollaa apuna käyttäen (Voitovych, Yuvkovetskyi ja Kupershtein 2016).

Kaistan ulkopuolinen SQL-injektio ei ole yleensä hyökkääjän ensisijainen vaihtoehto, mutta esimerkiksi tapauksissa, joissa palvelin on epävakaa tai liian hidas toimiakseen hyvin muiden SQL-injektioiden kohdalla, voidaan yrittää käyttää tietokannan kykyä ohjata palautteet toiselle laitteelle hyväksi (Imperva 2023).

Kaistan sisäisten sekä kaistan ulkopuolisten SQL-injektioiden eroa voidaan havainnollistaa seuraavien kuvien avulla.

Kuvio 2. Havainnollistava kuva kaistan sisäisestä SQL-injektiosta.



Kuvio 3. Havainnollistava kuva kaistan ulkopuolisesta SQL-injektiosta. Mukailtu lähteestä (How 2019).

4 SQL-injektiota vastaan suojautuminen

Tässä luvussa käydään läpi SQL-injektiohaavoittuvuuksiin liittyviä mahdollisia havaitsemistekniikoita, sekä SQL-injektioihin liittyviä hyväksi todettuja ehkäisymenetelmiä.

4.1 Havaitsemistekniikat

Verkkosovellukset voivat sisältää SQL-injektiohaavoittuvuuksia vuosia ilman kenenkään huomaamatta. Jotta näiden haavoittuvuuksien olemassaolo voitaisiin minimoida, on tarpeellista kehittää hyviä havaitsemistekniikoita, jotka ovat kriittisiä varsinkin vanhempia verkkosovelluksia käsiteltäessä, jolloin SQL-injektioihin liittyvä tietoisuus ja ehkäisy ei ole ollut yhtä yleistä.

Shar ja Tan (2013) arvioivat useita eri SQL-injektiohaavoittuvuuden havaitsemiseen kehitettyjä ohjelmistoja, mutta vaikka ne havaitsivat useita oikeita haavoittuvuuksia, havaittiin niiden käytössä myös useita huonoja puolia, kuten esimerkiksi teknologian huono skaalautuvuus suurempiin järjestelmiin, erilaiset väärät hälytykset, sekä jatkuva ohjelmoijan väliintulon tarve. Tästä askeleen eteenpäin otti Junjin (2009), joka kehitti oman ohjelmansa haavoittuvuuksien havaitsemiseen, jossa hän hyödynsi useita eri tekniikoita haavoittuvuuksien tunnistamiseen. Junjin (2009) totesi hänen ohjelmansa hyvin toimivaksi erityisesti vanhemmissa sovelluksissa, joiden kohdalla ei ollut käytetty hyväksi todettuja nykyaikaisia ohjelmointikäytänteitä, tuotti myös hänen ohjelmansa vähäisiä vääriä hälytyksiä, joka johtaa ylimääräiseen tuhlattuun aikaan ohjelmoijan kohdalla.

4.2 Ehkäiseminen

Koska SQL-injektiot ovat suora seuraus ohjelmoijien huonoista ohjelmointikäytänteistä, on vastuu SQL-injektioita vastaan suojautumisessa viime kädessä ohjelmoijalla itsellään. Tästä syystä on tärkeää ymmärtää mitä hyväksi todetut käytänteet ovat, sekä millaisia eri teknologioita ohjelmoija voi käyttää hyväkseen SQL-injektioiden ehkäisemiseksi. Ehkäisemiseen liittyvät tekniikat on jaettu kahteen eri kategoriaan, jotka ovat hyvät ohjelmointikäy-

tänteet, sekä suorituksenaikaisten SQL-injektioiden ehkäisy (engl. Runtime SQLIA prevention) (Shar ja Tan 2013).

Hyvissä ohjelmointikäytänteissä ohjelmoija keskittyy käyttäjän syötteen oikeaoppiseen käsittelyyn. Yksinkertaisin esimerkki tästä on tiettyjen sanojen ja merkkien valko- tai mustalistaus. Esimerkiksi käyttäjän etunimeä, sukunimeä tai ikää kysyttäessä ei ole välttämättä tarpeellista vastaanottaa tiettyjä erikoismerkkejä tai sanoja, joten näiden merkkien ja sanojen estäminen on helppo tapa estää yksinkertaisimmat mahdolliset haavoittuvuudet.

Toinen yleinen hyväksi todettu ohjelmointikäytänne on parametrisoitujen kyselyiden (engl. Prepared Statements) hyödyntäminen. Parametrisoiduissa kyselyissä muodostetaan SQL-kyselyitä, joissa käyttäjän syötteen käsittely erotellaan muusta käskyn rakenteesta. Koska käyttäjän syöte käsitellään erillään ennalta luodusta käskystä, ei käyttäjän syötteellä ole mahdollisuutta vaikuttaa alkuperäisen käskyn logiikkaan (Thomas, Williams ja Xie 2009). Tarkastellaan yksinkertaista esimerkkiä parametrisoidusta kyselystä.

```
SELECT *
FROM users
WHERE name = ? AND password = ?
```

Edellä esitetyn SQL-koodin kääntäminen ilman käyttäjän syötettä ei tule tuottamaan virheitä, ja kysymysmerkkien paikalle voidaan tämän jälkeen lisätä käyttäjän syöte, jolla ei ole enää mahdollisuutta vaikuttaa alkuperäisen lauseen loogisen merkityksen muuttumiseen.

Suorituksenaikaisessa SQL-injektioiden ehkäisyssä hyökkäys pyritään paikantamaan järjestelmän muodostaessa kyselyä annetuilla syötteillä. Yksi esimerkki suorituksenaikaisesta SQL-injektion ehkäisystä on SQLrand, joka käyttää hyväkseen verkkosovelluksen ja tietokannan välille asettuvaa välipalvelinta, joka pakottaa ohjelmoijaa käyttämään normaalien avainsanojen sijaan satunnaistettuja avainsanoja (Boyd ja Keromytis 2004). Välipalvelin pysäyttää SQL-kyselyn etenemisen tietokannalle, ja purkaa avainsanojen satunnaistamisen ennen kuin kysely lähetetään tietokannalle. Näin ollen hyökkääjän on mahdotonta viedä tulkille syntaktisesti järkevää lausetta, jollei hän itse tiedä satunnaistamiseen käytettyä logiikkaa. Suorituksenaikainen ehkäisyn huonoina puolina pidetään sen hidastavaa vaikutusta järjestelmän suorituskykyyn, sekä joissakin tapauksissa virheiden paikantamisen vaikeutumisen lisäänty-

minen.

Tietokannan turvallisuuden kannalta järkevää on myös rajoittaa eri käyttäjien käyttöoikeuksia tietokantaan liittyvien komentojen kohdalla (Mavromoustakos ym. 2016). Esimerkiksi kaikilla käyttäjille ei välttämättä tarvitse olla oikeuksia tietokannan tiettyihin käskyihin, joilla voidaan poistaa attribuutteja tai kokonaisia tauluja.

5 Esimerkkejä SQL-injektioilla toteutetuista tietomurroista

Tässä luvussa esitellään muutama tunnettu SQL-injektiohyökkäys viime vuosikymmeniltä, sekä kehen ne ovat kohdistuneet, ja millaisia vahinkoja ne ovat aiheuttaneet.

Injektiota pidetään yhtenä kriittisimmistä verkkosovelluksiin kohdistuvista tietoturvariskeistä (OWASP 2021). 2000-luvun aikana suurin osa SQL-injektiohyökkäyksistä on keskittynyt suurien yrityksien verkkosivuihin sekä sosiaalisen median alustoihin. Jotkin näistä hyökkäyksistä johtivat suuriin tietomurtoihin, joiden seurauksena on voitu vuotaa jopa tuhansien käyttäjien henkilökohtaisia tietoja (Johny ym. 2021). Ensimmäisen SQL-injektio haavoittuvuuden esiintymän on dokumentoinut tietoturva asiantuntija Jeff Forristal vuonna 1998 (Horner ja Hyslip 2017). Vaikka Forristal osoitti kuinka hyökkääjä pystyisi palauttamaan Microsoftin SQL-palvelimelta arkaluontoista informaatiota, ei Microsoft pitänyt haavoittuvuutta vakavana uhkana (Malwarebytes 2023).

SQL-injektio hyökkäyksiä on kohdistunut niin kaupallisiin yhtiöihin, kuin valtioiden hallinnollisiin elimiin. Yhtenä merkityksellisimmistä SQL-injektio hyökkäyksistä voidaan pitää elokuussa 2009 tapahtunutta hyökkäystä, jossa varastettiin dataa 130 miljoonasta pankki- ja luottokortti järjestelmästä. Näihin järjestelmiin sisältyi muun muassa Heartland Payment System, sekä päivittäistavarakauppa 7Eleven, ja tapauksen ajatellaan olevan suurin identiteettivarkaus Yhdysvaltojen historiassa (Johny ym. 2021).

Ei-kaupallisiin organisaatioihin kohdistuneista SQL-injektio hyökkäyksistä yksi tunnetuimmista tapauksista on Rhode Islandin osavaltion hallitukseen kohdistunut hyökkäys. Haavoittuvuus mahdollisti henkilökohtaisten tietojen varastamisen käyttäjiltä, jotka olivat käyneet kauppaa osavaltiolle kuuluvien virastojen kanssa (Johny ym. 2021). Muita ei-kaupallisten organisaatioiden hyökkäyksen kohteita ovat olleet muun muassa useat yliopistot, joihin kohdistuneessa hyökkäyksessä vuodettiin oppilaiden sekä työntekijöiden henkilökohtaisia tietoja (Computerworld 2012), sekä Turkin hallintoon kohdistunut hyökkäys, jossa väitetysti pyyhittiin asiakkaiden velkoja pois (Kozik ja Choraś 2018).

Muita tunnettuja SQL-injektio hyökkäyksen uhreja ovat olleet mm. Microsoftin yhdistyneiden kuningaskuntien nettisivut, yhdysvaltojen armeijan verkkopalvelimet, malesialaisen Kaspersky antiviruksen nettisivut, sekä Yahoo, jossa vuodettiin yli 450 000 käyttäjän kirjautumistiedot (Johny ym. 2021).

6 Yhteenveto

SQL-injektiot ovat yksi suurimmista ja vakavimmista tietokantoihin ja verkkosovelluksiin kohdistuvista tietoturvauhista (OWASP 2021). Ne ovat vaikuttaneet niin kaupallisten organisaatioiden, kuin hallinnollisten elinten verkkosovelluksiin ja tietokantoihin. SQL-injektioiden vakavuus nousee esille varsinkin useissa vanhemmissa ohjelmistoissa, jolloin yleinen tietoisuus haavoittuvuuden olemassaolosta sekä niitä vastaan toimimisesta on ollut vähäisempää. Lisäksi yksi mahdollinen ongelma, joka mahdollistaa SQL-injektio haavoittuvuuden on näkökulma, jossa ohjelmistoon liittyvät tietoturvaominaisuudet nähdään yrityksen kannalta rahan hukkana, sillä niiden kehittäminen ei varsinaisesti paranna tuotteen tuottavuutta, joten niiden laiminlyönti voi tuottaa hetkellisesti enemmän voittoa, mutta tulevaisuudessa voi olla katastrofaalista organisaation kannalta (ZDNET 2021).

Viime kädessä SQL-injektioita vastaan toimiminen jää ohjelmoijien sekä ohjelmistosuunnittelijoiden harteille, sillä ainut tapa välttyä SQL-injektioilta on käyttää hyväksi todettuja käytänteitä. SQL-injektio haavoittuvuuksien olemassaoloa voi olla vaikea tiedostaa, sillä niistä ei aiheudu vahinkoa ennen kuin hyökkääjä käyttää niitä hyväkseen. Tutkielmassa esiteltiin muutama automaattinen tarkistin, mutta niiltäkin löytyi omat huonot puolensa, jotka muun muassa vaativat ohjelmoijien väliintuloa tai muuta ylimääräistä työtä. Parhaaksi todettu tapa välttää SQL-injektiota on siis toimia ennakoivasti jo ohjelmistoa suunniteltaessa ja toteutettaessa, sillä jälkeenpäin haavoittuvuuksia paikatessa työn määrä on huomattavasti suurempi.

Lähteet

Boyd, Stephen W, ja Angelos D Keromytis. 2004. “SQLrand: Preventing SQL injection attacks”. Teoksessa *Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004. Proceedings 2,* 292–302. Springer.

Codd, E. F. 1983. “A Relational Model of Data for Large Shared Data Banks”. *Commun. ACM* (New York, NY, USA) 26, numero 1 (tammikuu): 64–69. ISSN: 0001-0782. https://doi.org/10.1145/357980.358007. https://doi-org.ezproxy.jyu.fi/10.1145/357980.358007.

Computerworld. 2012. *Group says it hacked systems at 100 major universities.* Saatavilla WWW-muodossa, ://www.computerworld.com/article/2491950/group-says-it-hacked-systems-at-100-major-universities.html, viitattu 10.5.2023.

DB-Engines. 2023. *DB-Engines DBMS popularity broken down by database model.* Saatavilla WWW-muodossa, https://db-engines.com/en/ranking_categories, viitattu 19.4.2023.

Horner, Matthew, ja Thomas Hyslip. 2017. “SQL injection: the longest running sequel in programming history”. *Journal of Digital Forensics, Security and Law* 12 (2): 10.

How, Lee Chun. 2019. *Out-of-Band (OOB) SQL Injection.* Saatavilla WWW-muodossa, https://infosecwriteups.com/out-of-band-oob-sql-injection-87b7c666548b, viitattu 26.4.2023.

Imperva. 2023. *SQL (Structured query language) Injection.* Saatavilla WWW-muodossa, ://www.imperva.com/learn/application-security/sql-injection-sqli/, viitattu 26.4.2023.

Johny, Joanna Hazaline Binti, Wafa Athilah Fikriah Binti Nordin, Nurrina Mizana Binti Lahapi ja Yu-Beng Leau. 2021. “SQL Injection prevention in web application: a review”. Teoksessa *Advances in Cyber Security: Third International Conference, ACeS 2021, Penang, Malaysia, August 24–25, 2021, Revised Selected Papers 3,* 568–585. Springer.

Junjin, Mei. 2009. “An Approach for SQL Injection Vulnerability Detection”. Teoksessa *2009 Sixth International Conference on Information Technology: New Generations,* 1411–1414. https://doi.org/10.1109/ITNG.2009.34.

Kozik, Rafał, ja Michał Choraś. 2018. “Protecting the application layer in the public domain with machine learning methods”. *Logic Journal of the IGPL* 27, numero 2 (syyskuu): 149–159. ISSN: 1367-0751. https://doi.org/10.1093/jigpal/jzy029. eprint: https://academic.oup.com/jigpal/article-pdf/27/2/149/28246840/jzy029.pdf. https://doi.org/10.1093/jigpal/jzy029.

Malwarebytes. 2023. *What is SQL injection.* Saatavilla WWW-muodossa, ://www.malwarebytes.com/sql-injection, viitattu 9.5.2023.

Mavromoustakos, Stephanos, Aakash Patel, Kinjal Chaudhary, Parth Chokshi ja Shaili Patel. 2016. “Causes and Prevention of SQL Injection Attacks in Web Applications”, 55–59. Joulukuu. https://doi.org/10.1145/3026724.3026742.

Nasereddin, Mohammed, Ashaar ALKhamaiseh, Malik Qasaimeh ja Raad Al-Qassas. 2021. “A systematic review of detection and prevention techniques of SQL injection attacks”. *Information Security Journal: A Global Perspective,* 1–14.

OWASP. 2021. *A03:2021 – Injection.* Saatavilla WWW-muodossa, https://owasp.org/Top10/A03_2021-Injection/, viitattu 19.2.2023.

Rai, Aditya, MD. Mazharul Islam Miraz, Deshbandhu Das, Harpreet Kaur ja Swati. 2021. “SQL Injection: Classification and Prevention”. Teoksessa *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM),* 367–372. https://doi.org/10.1109/ICIEM51511.2021.9445347.

Researchgate. 2016. *ResearchGate example of relational table.* Saatavilla WWW-muodossa, https://www.researchgate.net/figure/example-of-relational-table_fig2_316216480, viitattu 19.4.2023.

Shar, Lwin Khin, ja Hee Beng Kuan Tan. 2013. “Defeating SQL Injection”. *Computer* 46 (3): 69–77. https://doi.org/10.1109/MC.2012.283.

StackOverflow. 2020. *Stack Overflow Developer Survey 2020.* Saatavilla WWW-muodossa, https://insights.stackoverflow.com/survey/2020, viitattu 13.3.2023.

Thomas, Stephen, Laurie Williams ja Tao Xie. 2009. “On automated prepared statement generation to remove SQL injection vulnerabilities”. *Information and Software Technology* 51 (3): 589–598. ISSN: 0950-5849. https://doi.org/https://doi.org/10.1016/j.infsof.2008.08.002. https://www.sciencedirect.com/science/article/pii/S0950584908001110.

Voitovych, O. P., O. S. Yuvkovetskyi ja L. M. Kupershtein. 2016. “SQL injection prevention system”. Teoksessa *2016 International Conference Radio Electronics Info Communications (UkrMiCo),* 1–4. https://doi.org/10.1109/UkrMiCo.2016.7739642.

ZDNET. 2021. *Bosses are reluctant to spend money on cybersecurity. Then they get hacked.* Saatavilla WWW-muodossa, ://www.zdnet.com/article/too-many-bosses-are-reluctant-to-spend-money-on-cybersecurity-then-they-get-hacked/, viitattu 10.5.2023.