Petri Monola

Injektiot ja oikeuksien eskaloituminen luottamuksellisen tiedon turvaamisen uhkana

Tietotekniikan kandidaatintutkielma

16. toukokuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Petri Monola

Yhteystiedot: petri.k.monola@student.jyu.fi

Ohjaaja: Sanna Mönkölä

Työn nimi: Injektiot ja oikeuksien eskaloituminen luottamuksellisen tiedon turvaamisen uhkana

Title in English: Injections and privilege escalation as a threat for protection of confidential information

Työ: Kandidaatintutkielma

Sivumäärä: 24+0

Tiivistelmä: Nykyaikana käytetään entistä enemmän tietojärjestelmiä, joihin tallennetaan luottamuksellista tai arkaluontoista tietoa. On myös pahantahtoisia käyttäjiä, jotka haluavat päästä käsiksi tähän arkaluontoiseen tietoon. Tässä tutkielmassa tarkastellaan injektioita ja oikeuksien eskaloitumista keinona saada arkaluontoista tietoa tietojärjestelmästä. Tarkemmin injektioista käsitellään kolmea eri tyyppiä: shell-, skripti- ja SQL-injektioita. Oikeuksien eskaloitumiseen liittyen, tutkielmassa tarkastellaan järjestelmien oikeuksien hallintaa sekä miten oikeuksia jaetaan eri tasoihin. Eskaloitumisen estämisen keinona esitellään lyhyesti oikeuksien erottelumenetelmää.

Avainsanat: injektiot, oikeuksien eskaloituminen, arkaluontoinen tieto, tietojärjestelmä, cross-site scripting, SQL-injektio, shell-injektio, skripti-injektio

Abstract: Nowadays information systems and databases are constantly growing even more popular, and confidential information are being trusted in the hands of such systems. There are also malicious users who want to get their hands on this confidential information. This paper looks into injections and privilege escalation as a way of extracting confidential information out of information systems. Three types of injections are included: shell, script and SQL-injections. This paper inspect ways of controlling privileges in a system and how privileges are divided into levels. Privilege separation is briefly introduced as a method of preventing privilege escalation.

Keywords: injections, privilege escalation, SQL-injections, confidential information, information system, cross-site scripting, shell-injection

Kuviot

Sisältö

1 Johdanto

Nykyaikana käytetään entistä enemmän tietojärjestelmiä joihin tallennetaan luottamuksellista tai arkaluontoista tietoa. Luottamuksellisella tiedolla tarkoitetaan tässä esimerkiksi järjestelmässä tai kannassa olevien henkilöiden henkilötietoja tai yksityiseen henkilöön tai yritykseen liittyvää ei-julkista tietoa. Tätä tietoa on tärkeää suojata kyberhyökkäyksiltä, joissa yritetään päästä käsiksi järjestelmässä sijaitseviin tietoihin (Tsegaye ja Flowerday 2014). Turvallisten ja luotettavien järjestelmien tulisi suojata siellä sijaitsevaa tietoa väärinkäytöltä ("ISO27001: Information Security Management System (ISMS) standard" 2005).

Tässä tutkielmassa tarkastellaan kyberhyökkäyksiä ja niitä mahdollistavia haavoittuvuuksia. Haavoittuvuus on järjestelmässä oleva vika, joka altistaa järjestelmän kyberhyökkäykselle (Tsegaye ja Flowerday 2014). Tutkielmassa käsitellään syvällisemmin sellaisia haavoittuvuuksia, jotka mahdollistavat luottamuksellisten tietojen vuotamisen tietojärjestelmästä. Tällaisia tyypillisiä keinoja ovat injektiot ja oikeuksien eskaloituminen. Oikeuksien eskaloitumisella tarkoitetaan tapahtumaa jossa järjestelmän hyökkääjä saa korotetut käyttöoikeudet järjestelmään ja siten pääsee käsiksi arkaluontoiseen tietoon (Provos, Friedl ja Honeyman 2003). Injektioilla tarkoitetaan tilannetta jossa hyökkääjä voi jonkin haavoittuvuuden avulla injektoida koodia järjestelmään, ja siten pyrkii saamaan tietoja toisistä käyttäjistä, heidän antamista tiedoista ja itse järjestelmästä.

Varsinkin web-sovelluksissa vaarallisimpia hyökkäyksiä ovat SQL-injektiot ja skripti-injektiot (Gupta, Govil ja Singh 2014). Injektioiden ja oikeuksien eskaloitumisen avulla hyökkääjä voi päästä käsiksi järjestelmän resursseihin ja sitä kautta voi paljastaa arkaluontoista tietoa. Tutkielmassa tarkastellaan siis koodi-injektioita ja oikeuksien eskaloitumista, niiden syitä ja seurauksia, sekä keinoja miten järjestelmä voidaan suojata näiltä hyökkäyksiltä.

Luvussa 2 käsitellään koodi-injektioita ja niistä kolmea eri tyyppiä: shell, skripti ja SQL. Jokaisesta injektiotyypistä esitetään esimerkkitapauksia. Tässä osiossa pyritään selvitäämään mitä injektioilla tarkoitetaan ja minkälaiset haavoittuvuuden mahdollistavat injektiohyökkäykset sekä käydään läpi mitä ovat koodi-injektioiden seuraukset pahimmillaan ja miten niitä vastaan suojaudutaan. Luvussa 3 käsitellään oikeuksien eskaloitumista, tapoja hallita

oikeuksia ja esimerkkejä siitä minkälainen oikeusrakenne järjestelmässä voi olla. Tässä luvussa selitetään mitä oikeuksien eskaloitumien tarkoittaa ja miten eskaloituminen voi edetä. Luvun lopussa käydään läpi mitä voivat olla oikeuksien eskaloitumisen seuraukset sekä miten injektioita vastaan voidaan suojautua. Luvussa 4 esitellään tutkielman johtopäätökset sekä tiivistetään tutkielman olennaisimman asiat.

2 Koodi-injektiot

(Lin ja Chen 2007) ovat todenneet, että koodi-injektiot usein sivuuttavat järjestelmän alkuperäiset toimintatavat tai muokkaavat niitä. Tällöin hyökkääjä voi mahdollisesti saada informaatiota järjestelmästä tai luvattomia oikeuksia järjestelmään. Jotta injektiot olisivat mahdollisia, täytyy järjestelmässä olla jokin haavoittuvuus, jota voidaan hyödyntää koodin injektoimiseen. Tällainen tyypillinen järjestelmän haavoittuvuus on tarkistamattomat syötteet. Sen sijaan turvallisemmat järjestelmät rakentavat syötteiden perusteella tehdyt kyselyt dynaamisesti, siten että ne koostuvat tarkoituksellisista komennoista ja sinne sijoitetuista syötteistä (Su ja Wassermann 2006). Yllättävän monet sovellukset ovat edelleen haavoittuvaisia koodi-injektoille.

Seuraavissa aliluvuissa käsitellään kolmea eri tyyppistä injektioita: shell-, skripti- ja SQL-injektioita. Näistä käsitellään myös esimerkkitapauksia sekä tarkastellaan yleisesti injektioiden seurauksia, syitä ja niitä vastaan suojautumista.

2.1 Shell-injektiot

Unix shell on komentotulkki unix-järjestelmissä. Järjestelmän sovellus voi antaa komentoja suoritettavaksi komentotulkille. Tällöin on mahdollista injektoida komentoja komentotulkin suoritettavaksi, jos sovelluksessa on jokin ohjelmointivirhe tai muu haavoittuvuus. Haavoittuvuus komentotulkki-injektiolle antaa käyttäjälle mahdollisuuden suorittaa komentoja root-käyttäjänä, vaikka käyttäjällä olisikin rajoitetut käyttöoikeudet (Su ja Wassermann 2006). Tällöin käyttäjä pystyy haavoittuvuuden kautta esimerkiksi asettamaan järjestelmän osiin alemmat käyttöoikeudet, tai asettaa itselleen korkeammat käyttöoikeudet järjestelmään, jolloin korotetaan eli eskaloidaan oikeuksia. Oikeuksien eskaloitumista käsitellään tarkemmin luvussa 3.

Jos sovellus lähettää toimintoja komentoriville suoritettavaksi, niin hyökkääjä voi käyttää puolipistettä (*;*) lopettamaan järjestelmän komennon ja sen jälkeen voi kirjoittaa oman skriptin, joka toteutetaan korkeammilla oikeuksilla. Lisäksi esimerkiksi pystyviivalla (|) voidaan tehdä komentosarjoja eli laittaa peräkkäin monta komentoa, jolloin edellisen komennon tu-

los toimii syötteenä seuraavalle. Shell-injektioita käsitellään esimerkkien avulla aliluvussa 2.4.

2.2 Skripti-injektiot

Skripti-injektiot ovat usein web-sovlluksien yhteydessä esiintyvä ongelma. Kun tarkoitetaan web-sovelluksissa tapahtuvia injektioita, käytetään usein nimitystä Cross-site scripting (XSS). Ne seuraavat siitä kun sovellukselle annetut syötteet sekoittuvat sovelluksen tai web-sivun koodeihin. Tämä on mahdollista silloin kun web-sovellus ei tarkista käyttäjän antaman syötteen oikeellisuutta (Gupta, Govil ja Singh 2014). Skriptejä voidaan injektioida suoraan HTML-koodin sekaan web-sivulla tai web-sovelluksessa, mikä mahdollistaa skriptin suorittamisen jonkun toisen käyttäjän sessiossa, jolloin skripti pystyy lähettämään käyttäjän evästeitä hyökkääjälle.

Skriptit kirjoitetaan web-sovelluksissa tyypillisesti JavaScriptillä. Injektoidut skriptit voivat vuodattaa tietoa erinäisin keinoin: skripti-injektioiden avulla voidaan esimerkiksi kaapata käyttäjien tilejä, varastaa tietoa ja evästeitä, manipuloida sivujen sisältöä sekä niitä voidaan myös hyödyntää palvelunestohyökkäyksissä (Gupta, Govil ja Singh 2014; Thompson ja Chase 2005). Evästeistä voidaan lukea sessiokohtaista tietoa, joihin on voitu tallentaa käyttäjän antamia tietoja. Tiedostoista lukeminen saattaa jo itsessään paljastaa arkaluontoista tietoa.

(Thompson ja Chase 2005) kertovat, että Javascriptillä voidaan hakea sivujen asettamia evästeiden arvoja samassa domainissa missä skriptiä ajetaan. Tällöin hyökkääjä voi julkaista skriptin, joka kerää sessioevästeen ja vie sen CGI-parametrinä kolmannen osapuolen sivulle. Kun tavallinen käyttäjä ottaa yhteyden sivulle, käyttäjän sessioeväste kaapataan ja lähetetään hyökkääjän sivulle. Tämän evästeen avulla hyökkääjä voi esittää olevansa käyttäjä, jolta evästeen hyökkääjä kaappasi. Tässä tilanteessa hyökkääjä saa oikeudet toisen käyttäjän sessioon, jolloin hyökkääjä voi lukea toisen käyttäjän antamia tietoja tai suorittaa järjestelmässä toimintoja.

2.3 SQL-injektiot

SQL-injektiot ovat injektiohyökkäyksiä, joilla pyritään hyökkäämään tietokantoihin käyttämällä SQL-kyselykieltä tietojen noutamiseen. SQL-injektio on yksi tyypillinen web-sovelluksen haavoittuvuus ja tätä haavoittuvuutta hyödyntämällä hyökkääjä saattaa tehdä kyseilyita tietokannasta ja näin saada arkaluontoista tietoa käsiinsä (Sadeghian, Zamani ja Ibrahim 2013).

Kuten shell-injektioissa hyökkääjä voi suorittaa omaa koodia käyttämällä puolipistettä, jolla voidaan keskeyttää järjestelmän oma kysely, ja sen perään injektoida mielivaltainen kysely. Käyttämällä SQL-kielen avainsanoja (esim. *OR* ja *AND*), voidaan muokata alkuperäistä kyselyä toisenlaiseksi esimerkiksi antamalla ehtoja, jotka ovat automaattisesti totta, kuten luvun 2.4 kuviossa 2.

Kuten muutkin injektiotyppit niin myös SQL-injektiot usein johtuvat tarkastamattomista syötteistä (Gupta, Govil ja Singh 2014). Tällöin jos käyttäjän syöte sijoitetaan suoraan kyselyyn tarkastamatta, niin siitä seuraa vääränlaisia SQL-kyselyitä, mitkä saattavat johtaa arkaluontoisen tiedon paljastamiseen.

2.4 Esimerkkejä

Tässä luvussa käsitellään esimerkkejä SQL-, skripti-, ja shell-injektioista. Jokaisesta tyypistä annetaan esimerkki siitä millainen injektoitu koodi voi olla sekä selitetään miten koodi toimii.

Esimerkki SQL-injektiosta, mukailtu lähteestä (Qian ym. 2015):

```
SELECT  nimi, sukupuoli, osoite, syntymaaika
FROM asiakas_taulu
WHERE puhelin_nro=”123456”
```

Kuvio 1. SQL-tietokantakysely

```
SELECT nimi, sukupuoli, osoite, syntymaaika
FROM asiakas_taulu
WHERE puhelin_nro=”123456” OR 1=1
```

Kuvio 2. SQL-injektio

Kuvion 1 mukaisessa kyselyssä haetaan kannasta käyttäjän tiedot puhelinnumeron mukaan. Kuitenkin jos käyttäjä syöttäisi ”123456 or 1=1”, kuten kuviossa 2, niin tällöin koska 1=1 on aina totta, kysely palauttaa taulusta kaikkien käyttäjien tiedot.

Esimerkki skripti-injektiosta, mukailtu lähteestä (Su ja Wassermann 2006):

```
><script>document.location='http://www.xss.com/cgi-
bin/cookie.cgi?'%20+document.cookie</script
```

Kuvio 3. Skripti-injektio

Hyökkääjä voi käyttää hyväkseen syötekenttää, jonka sisältö asetetaan sivulle. Esimerkiksi foorumi- tai huutokauppa-sivustoille voidaan lisätä tekstiä tai linkkejä. Jos sivusto on altis skripti-injektioille voidaan injektoida kuvion 3 kaltainen skripti, joka lähettää käyttäjän evästeet sivulle *http://www.xss.com*. Kuvion 3 HTML-koodi ei ole validia, sillä ensimmäinen >-merkki lopettaa edeltävän HTML-tagin.

Esimerkki shell-injektiosta:

```
$ ./ohjelma teksti.txt

    Tuloste:
        Tämä on tekstitiedoston sisältöä...
```

Kuvio 4. Shell ohjelman toiminta

$ ./ohjelma "teksti.txt; ls"

Tuloste:

Tämä on tekstitiedoston sisältöä...
teksti.txt tiedosto1.dat muuta2.dat
tunnuksia.dat muuta1.dat muuta3.dat

Kuvio 5. Shell injektiolla listaus tiedostoista

$ ./ohjelma "teksti.txt; cat tunnuksia.dat"

Tuloste:

Tämä on tekstitiedoston sisältöä...
Käyttäjä_1: s&alaSanA123
Käyttäjä_2: hRg3Gs_&2
Käyttäjä_3: 133_25231aG2

Kuvio 6. Shell injektiolla tiedoston luku

Kuvion 4 ohjelman tarkoitus on tulostaa käyttäjän antaman tekstitiedoston sisältö. Ohjelma hyväksyy syötteeksi siis vain tekstitiedostoja. Käyttäjällä ei ole oikeuksia lukea järjestelmän muita tiedostoja, mutta on oikeus suorittaa tätä ohjelmaa. Ohjelman tarkoituksena on antaa käyttäjälle mahdollisuus lukea tekstitiedostoja ohjelman avulla. Kuitenkin jos sovellus omaa korkeammat oikeudet, esimerkiksi oikeudet suorittaa komentoja kometorivillä, niin tällöin käyttäjä voi mahdollisesti väärinkäyttää sovelluksen oikeuksia ajaakseen omia komentoja sen kautta. Kuviossa 5 injektoidaan komento, joka näyttää hakemistossa olevat tiedostot. Käyttäjä on antanut syötteeksi oikeanlaisen tiedoston, joten sovellus on tyytyväinen, mutta sen lisäksi käyttäjä on injektoinut omaa koodia. Kuviossa 6 ohjelma suorittaa injektoidun komennon ja tulostaa .dat -tyyppisen tiedoston sisällön, mikä sisältää käyttäjien tunnuksia ja salasanoja selkokieliseen muotoon tallennettuna.

2.5 Injektioiden seuraukset

Injektioista voi seurata luottamuksellisten tietojen vuotamista, tietojen tuohamista, oikeudetonta tunnistautumista järjestelmään sekä järjestelmän oikeuksien muokkaamista (Johari ja

Sharma 2012). Tunnistautuminen ja oikeuksien muokkaaminen voidaan luokitella oikeuksien eskaloitumiseksi, tätä käsitellään tarkemmin luvussa 3. Suoraan tietokannasta injektioiden avulla luettu tieto sekä käyttäjien sessiosta saatu tieto voi olla luottamuksellista tietoa.

Useita haavoittuvuuksia yhdistelemällä voidaan saada aikaiseksi vakavampia hyökkäyksiä. Esimerkiksi jos järjestelmään tallennetaan tunnukset ja salasanat selkokielisenä, toisin sanoen käyttämättä minkäänlaista salausalgoritmia, ja järjestelmä on altis injektioille niin voidaan käyttää kuvion 2 kaltaista injektiota, joka palauttaisi kannasta useita tunnuksia ja salasanoja. Tällöin pystyttäisiin tunnistautumaan järjestelmään toisten käyttäjien tunnuksilla ja päästä käsiksi heidän tietoihin. Siispä usean eri haavoittuvuuden yhdistelmä voi moninkertaistaa hyökkäyksien tuhoa. Myös (Thompson ja Chase 2005) toteavat, että salasanojen ja tunnuksien tallentaminen selkokielellä on vaarallista, koska tällöin tunnistautumistiedot voidaan varastaa. Tunnistautumistietojen salauksesta voi lukea lisää teoksesta *The Software Vulnerability Guide*, Thompson ja Chase 2005.

2.6 Suojautuminen injektioita vastaan

Koska injektiot johtuvat usein tarkastamattomista syötteistä, niin injektioita vastaan voidaan suojautua tekemällä järkeviä syötteen tarkastuksia eli validoidaan käyttäjän antamat syötteet. Siispä täytyy tarkastella sovelluksen kohtia joissa käyttäjä syöttää dataa. (Lin ja Chen 2007) esittelevät niin kutsuttua mustalaatikkotestausta (engl. *black-box testing*). Se antaa jo hyviä tuloksia, jos järjestelmää osataan testata oikeilla injektiosyötteillä. Ei voida aina olettaa, että kaikki käyttäjät antavat tiettyjä tai oikeinlaisia syötteitä. Hyökkääjä voi tarkoituksella kokeilla erinäköisiä syötteitä, jotka saattavat rikkoa syötteentarkistusta. Esimerkiksi erikoismerkit ja escape-merkit (engl. *escape characters*) ovat hankalia syötteen tarkastuksen kannalta. Jotta haavoittuvuudet voidaan löytää, niin täytyy asettaa tarkastelijat hyökkääjän näkökulmaan. Voidaan olettaa, että hyökkääjä ei tunne tarkkaa järjestelmän toteutusta, mutta kuitenkin tuntee yleiset haavoittuvuudet joita esiintyy järjestelmien syötteentarkastuksissa.

On olemassa syötteen tarkastuksia, jotka estävät vääräntyyppiset ja haitalliset syötteet, siten että ne tunnetaan jo entuudestaan. Toisin sanoen nämä syötteet ovat merkattuja, tästä tulee myös nimitys *signature-based detection* eli tunnistetaan näitä merkattuja syötteitä. Injektiot

voidaan estää myös rajoittamalla syötteiden pituutta tai käyttämällä säännöllisia lausekkeita (engl. *regular expression*) (Su ja Wassermann 2006).

(Wei, Muthuprasanna ja Kothari 2006; Qian ym. 2015) esittelevät proseduurit (engl. *stored procedures*) yhtenä keinona estää SQL-injektiot. Proseduurit lisäävät abstraktin kerroksen järjestelmään ja niitä käytetään rajapinnan kautta. Proseduureilla pystytään piilottamaan kannan taulujen rakenteet ja määrittelyt. SQL-kyselyt luodaan sijoittamalla parametrina annettu data kyselyyn, siten että myös SQL-komennot pysyvät vain tekstimuotoisena datana, ja siten kyselyjen rakenne säilyy eivätkä injektiot ole mahdollisia.

Web-sovelluksissa ensiintyvien haavoittuvuuksien tunnistamiseen on kehitetty monia, laajasti käytettyjä työkaluja, esimerkiksi Web Application Vulnerability and Error Scanner (WAVES), WebSSARI (Web application Security Analysis and Runtime Inspection), AppScan, WebInspect ja ScanDo (Su ja Wassermann 2006; Lin ja Chen 2007). Näiden työkalujen avulla voidaan löytää injektioita mahdollistavia haavoittuvuuksia web-sovelluksista.

(Sadeghian, Zamani ja Manaf 2013) ovat esitelleet keinoja suojautua SQL-injektioita vastaan, muunmuassa merkkijonojen analysointityökalulla *String Analyzer* (Wassermann ja Su 2007). *Ardilla*, joka generoi mahdollisia hyökkäyksiä sovellukseen, mitkä voidaan siten ennakoida ja estää (Kieyzun ym. 2009). *SQLUnitGen*, jolla valvotaan käyttäjän syötteitä (Shin, Williams ja Xie 2006). *MUSIC*, jolla tutkitaan syntaktisesti virheellisiä injektioita ja niiden aiheuttamia muunnoksia järjestelmässä (Shahriar ja Zulkernine 2008). *SISE* on *SUSHI*-algoritmia hyödyntävä työkalu, jolla voi löytää SQL ja skripti-injektio haavoittuvuuksia web-sovelluksissa (Fu ja Li 2010). *Vulnerability and Attack Injetion* analysoi sovellusta sekä injektoi sovellukseen koodia, jonka jälkeen se tarkistaa oliko hyökkäys onnistunut ja toteaa haavoittuvuuden (Fonseca, Vieira ja Madeira 2009). *PHPMiner* louhii PHP-kielellä toteutetuista web-sovelluksista dataa löytääkseen haavoittuvuuksia (Shar ja Tan 2012). Lisäksi (Sadeghian, Zamani ja Manaf 2013) esittävät työkaluja myös ajonaikaiseen haavoittuvuuksien tunnistamiseen. Ajonaikaisessa tunnistamisessa saatetaan löytää haavoittuvuuksia, jotka eivät selviä koodia analysoimalla. Myös (Johari ja Sharma 2012) esittelevät erilaisia injektioiden estämiskeinoja. Web-sovelluksien haavoittuvuuksista on runsaasti kirjallisuutta ja tässä on mainittu vain osa.

3 Oikeuksien eskaloituminen

Oikeuksien eskaloituminen (engl. *privilege escalation*) on tapahtuma, jossa käyttäjä hyödyntää järjestelmästä löytynyttä haavoittuvuutta eli jotain ohjelmointi- tai suunnitteluvirhettä saadakseen korkeamman tason oikeudet järjestelmään. Näin käyttäjä pääsee käsiksi resursseihin jotka olivat aikaisemmin käyttäjän ulottumattomissa. Korkeamman tason oikeuksilla käyttäjä saattaa suorittaa toimintoja, jotka voivat vahingoittaa järjestelmää, tai päästä käsiksi järjestelmässä sijaitsevaan arkaluontoiseen tietoon.

Aliluvuissa käsitellään muun muassa kuinka nykyaikaisissa järjestelmissä hallitaan oikeuksia, millaisia oikeustasoja järjestelmässä voi olla, mistä eskaloituminen voi johtua sekä miten sitä vastaan voidaan suojautua.

3.1 Oikeuksien hallinta

Sovelluskehittäjät usein implementoivat rooleja keinona hallita oikeuksia (Monshizadeh, Naldurg ja Venkatakrishnan 2014). Järjestelmässä on useita käyttäjiä. Oikeuksia jaetaan sen mukaan mitä käyttäjän on tarpeen tehdä. Eri rooleja luodaan eri käyttötarkoituksia varten. Yleensä suositaan periaatetta jossa käyttäjällä on alimmat mahdolliset käyttöoikeudet (engl. *principle of least privilege*). Tämä tarkoittaa sitä, että käyttäjä voi suorittaa ja käsitellä vain sallittuja toimintoja, tiedostoja ja sovelluksia, mitkä ovat järjestelmän käytön kannalta oleellisimpia.

Usein järjestelmään on implementoitu pääkäyttäjä, jolla on korkeammat oikeudet järjestelmään. Pääkäyttäjän tarkoitus on hallita järjestelmää ja valvoa oikeuksia. Pääkäyttäjästä käytetään myös nimitystä järjestelmänvalvoja. Järjestelmänvalvoja voi esimerkiksi muokata käyttäjien oikeuksia, selata vapaasti kaikki resursseja ja suorittaa kaikkia toimintoja.

(Monshizadeh, Naldurg ja Venkatakrishnan 2014) mukaan oikeuksien eskaloitumisessa käyttäjä saa oikeudet toiseen ryhmään tai pystyy manipuloimaan ryhmään kuuluvien oikeuksia tai omia oikeuksia. Oikeuksien eskaloituminen voi olla joko vertikaalista tai horisontaalista. Vertikaalisessa eskaloitumisessa käyttäjä saa itselleen korkeammat oikeudet, esimerkiksi

järjestelmänvalvojan oikeudet. Horisontaalisessa eskaloitumisessa käyttäjä yrittää päästä käsiksi toisten käyttäjien resursseihin järjestelmän osiin ilman että käyttäjä saa korkeamman tason oikeuksia. Esimerkiksi käyttäjä voi web-sovelluksessa päästä toisen käyttäjän sessioon sisälle.

Oikeuksien hallintaan on olemassa malleja, esimerkiksi rinkimalli, jota käsitellään luvussa 3.2. Malleissa suositaan periaatetta, jossa järjestelmä jaetaan käyttäjiin ja resursseihin. Käyttäjille voidaan silloin luoda rooleja. Näitä rooleja voidaan hallita sekä muokata tarpeen mukaan ja siten pyritään estämään oikeuksien väärinkäyttö.

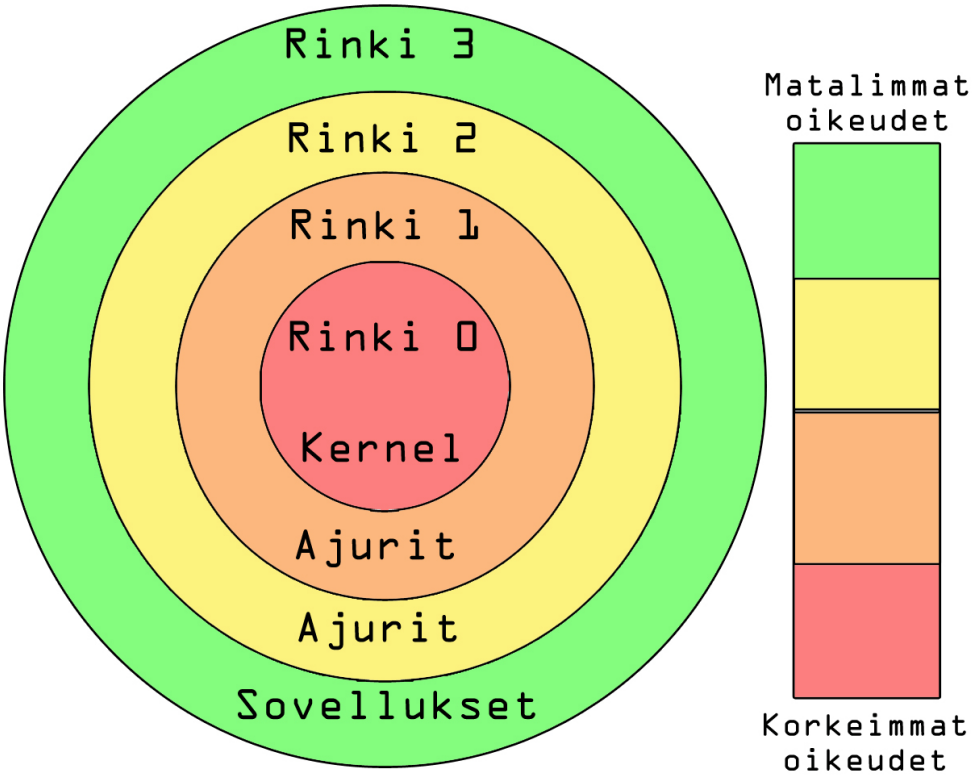
3.2 Oikeuksien tasot

Oikeudet voidaan esittää tasojen avulla. Korkeimman tason oikeuksilla voidaan suorittaa järjestelmässä mitä tahansa toimintoja. Muilla järjestelmän käyttäjillä on matalemman tason oikeudet, jolloin osa toiminnoista ei ole käytössä tai toimintoja voidaan suorittaa jonkin rajapinnan kautta, esimerkiksi sovelluksen kautta. Sovelluksilla voi siis olla korkeammat oikeudet, jotta sovelluksen tarvitsemia toimintoja voitaisiin suorittaa. Jos sovelluksessa on jokin haavoittuvuus, joka mahdollistaa esimerkiksi injektiot, niin käyttäjä voi hyödyntää näitä sovelluksen korkeampia oikeuksia suorittaakseen muita, mahdollisesti haitallisia, toimintoja, jotka vaativat korkeampia oikeuksia kuin mitä käyttäjällä on.

Esimerkiksi x86-järjestelmissä oikeudet ovat jaettu oikeusrinkeihin joista sisimmäisimmällä ringillä on täydet käyttöoikeudet ja ulommaisimmalla on vähimmät oikeudet järjestelmään (Maniatakos 2013). Jos sovelluksen kautta saa järjestelmään korkeampia oikeuksia, on koko järjestelmä vaarassa, sillä käyttäjä saattaa korkeampien oikeuksien avulla päästä järjestelmän muihin osiin käsiksi. Kuviossa 7 havainnollistetaan rinkimallia, jossa sismmässä ringissä on suurimmat oikeudet ja uloimmassa pienimmät oikeudet. Jos sovellus vaatii korkeamman tason oikeuksia kuin mitä käyttäjällä on, niin sen käyttö evätään.

Rinkimallin lisäksi on myös olemassa muita hierarkisia malleja. Nämä kaikki mallit noudattavat samoja periaatteita, joissa järjestelmä on jaettu käyttäjiin ja resursseihin (Thompson ja Chase 2005). Käyttäjilla on tietyt toiminnot jotka on käytettävissä, tyypillisesti luku-, kirjoitus- ja suoritusoikeudet. Näitä oikeuksia voidaan kuvata ja hallita tietyillä rakenteilla

tai malleilla.



Kuvio 7. Oikeuksien tasot rinkeihin jaettuna, mukaillen (Ellison ym. 2003)

3.3 Eskaloitumisen syyt ja seuraukset

Oikeuksien eskaloituminen voi johtaa siihen, että käyttäjä saa järjestelmänvalvojan oikeudet, jolloin käyttäjällä on täydet oikeudet koko järjestelmään. Käyttäjä pääsee vapaasti käsiksi järjestelmän resursseihin ja voi paljastaa arkaluontoista tietoa. Myös sovellus voi saada oikeudet järjestelmän resursseihin. Jos sovelluksen oikeuksia ei tarkasteta, silloin kun se yrittää käyttää järjestelmän resursseja tai suorittaa jotain toimintoa, niin sovellus voi käyttää sen oikeuksien ulkopuolisia resursseja hyväkseen (Chan, Hui ja Yiu 2011). Tällöin syynä eskaloitumiselle on oikeuksien tarkistuksen puuttuminen.

(Monshizadeh, Naldurg ja Venkatakrishnan 2014) toteavat, että usein web-sovellukset ottavat yhteyden tietokannan resursseihin pääkäyttäjänä, jolla on kaikki oikeudet tietokantaan, ja pienikin tunnistautumisvirhe (engl. *authorization error*) saattaa johtaa katastrofisiin tietomurtoihin. Tyypillisessä valtuuksien tarkastuksessa web-sovelluksissa tarkastetaan onko valtuutetulla käyttäjällä tietty rooli, joka vaaditaan tietokannan taulun tietojen lukemiseen. Jos sovellus käyttää jatkuvasti samoja tunnistautumiskäytänteitä kun yritetään päästä käsiksi samoihin resursseihin eri reittejä pitkin, niin tällöin tunnistatuminen on johdonmukaista. Jos tunnistautumistarkastukset ovat epäjohdonmukaisesti suoritettu, niin kulunvalvonnassa voi tapahtua virheitä, tämä voi johtaa sellaisiin heikkouksiin, jotka mahdollistavat oikeuksien eskaloitumisen. Käyttäjä voisi esimerkiksi saada oikeudet lukea tietokannan tauluja ilman valtuutusta.

Korkeampien oikeuksien omaavassa sovelluksessa olevan ohjelmointivirheen tai haavoittuvuuden kautta voidaan eskaloida oikeuksia (Thompson ja Chase 2005). Voi olla esimerkiksi mahdollista, että jokin virhe sovelluksessa mahdollistaa shell-injektion. Tällöin voidaan injektoida shell-skriptejä suoritettavaksi kernel-tasolla, jolloin hyökkääjä saattaa päästä suoraan muokkaamaan omia oikeuksia. Tällaisessa tapauksessa oikeudet voivat eskaloitua hyvinkin nopeasti.

Eskalointihyökkäyksiä pystytään toteuttamaan myös laitetasolla hyödyntämällä *address space indentifier*-korruptiota moderneissa prosessoreissa. Tätä aihealuetta ei kuitenkaan käsitellä tässä tutkielmassa, mutta muun muassa (Maniatakos 2013) on tutkinut tämänkaltaisia hyökkäyksiä.

3.4 Suojautuminen eskaloitumista vastaan

Hallitsemalla oikeuksia voidaan pitää huolta ettei järjestelmän käyttäjät saa liian korkeaita oikeuksia. Tunnistautuminen järjestelmään sekä valtuuksien tarkastaminen ovat olennaisia osia oikeuksien hallinnassa. Ohjelmointi- tai konfigurointivirheet, jotka esiintyvät näissä osissa voivat johtaa oikeuksien eskaloitumiseen (Thompson ja Chase 2005). Siispä järjestelmän toimintaa tai rakennetta tulisi tarkastella ja testata, jotta voitaisiin löytää näitä virheitä. On tärkeää myös suojata järjestelmä erilaisia haavoittuvuuksia vastaan, sillä esimerkiksi haa-

voittuvuudet jotka mahdollistavat injektiot, voivat johtaa oikeuksien eskaloitumiseen. Siispä myös injektioiden estäminen auttaa suojautumisessa oikeuksien eskaloitumista vastaan.

(Provos, Friedl ja Honeyman 2003) esittelevät oikeuksien erottelun (engl. *privilege separation*) keinona ehkäistä oikeuksien eskaloitumista. Tällöin sovellus jaetaan kahteen osaan, joista toisella on oikeudet suorittaa tiettyjä toimintoja ja toisella ei ole näitä oikeuksia. Osa jolla ei ole oikeuksia johonkin koodiin pyytää lupaa siihen oikeudelliselta osalta. Tämän tarkoituksena on vähentää koodia joihin tarvitaan erityisoikeudet niiden ajamiseen. Koska luvattomat oikeudet käyttäjälle voivat olla ohjelmointivirheiden lopputulos, voidaan estää ohjelmointivirheistä johtuvaa oikeuksien eskaloitumista erottelemalla järjestelmä oikeudelliseen ja oikeudettomaan osaan.

4 Yhteenveto

Tutkielmassa käsiteltiin kahden tyyppistä hyökkäystä, jotka mahdollistavat nykyaikaisissa järjestelmissä tietojen vuotamisen. Tarkastamattomista syötteistä johtuvat haavoittuvuudet voivat johtaa injektiohyökkäyksiin, jotka puolestaan voivat johtaa oikeuksien eskaloitumiseen. Web-sovelluksissa skripti- ja SQL-injektiot ovat yleisiä haavoittuvuuksia, joita hyökkääjä voi hyödyntää tietojen urkkimiseen. Jos hyökkääjä pääsee vapaasti käsiksi järjestelmän resurssehin ja toimintoihin, niin hyökkääjä voi aiheutaa suuria tuhoja ja tietovuoroja.

Järjestelmien, jotka tallentavat luottamuksellista dataa, tulisi varmistaa ettei sitä joudu vääriin käsiin. Siksi on tärkeää suojata tietojärjestelmissä sijaitsevaa arkaluontoista tietoa kyberhyökkäyksiltä. Injektiot ja oikeuksien eskaloituminen ovat uhkia arkaluontoisen tiedon turvaamiselle. Korjaamalla ohjelmointivirheitä ja haavoittuvuuksia, jotka mahdollistavat tämänkaltaiset hyökkäykset, voidaan suojata tietojärjestelmissä sijaitsevaa arkaluontoista tietoa. Injektioiden estäminen ja järkevästi toteutettu oikeuksien hallinta järjestelmässä varmistaa ettei tietoa joudu vääriin käsiin, jolloin voidaan turvata yksilöiden ja yritysten yksityisyyttä.

Lähteet

Chan, P. P. F., L. C. K. Hui ja S. M. Yiu. 2011. “A privilege escalation vulnerability checking system for android applications”. Teoksessa *Communication Technology (ICCT), 2011 IEEE 13th International Conference on,* 681–686. Syyskuu. doi:10.1109/ICCT.2011.6157963.

Ellison, C.M., R.A. Golliver, H.C. Herbert, D.C. Lin, F.X. McKeen, G. Neiger, K. Reneris, J.A. Sutton, S.S. Thakkar ja M. Mittal. 2003. *Executing isolated mode instructions in a secure system running in privilege rings.* US Patent 6,507,904, tammikuu. https://www.google.com/patents/US6507904.

Fonseca, J., M. Vieira ja H. Madeira. 2009. “Vulnerability & attack injection for web applications”. Teoksessa *2009 IEEE/IFIP International Conference on Dependable Systems Networks,* 93–102. Kesäkuu. doi:10.1109/DSN.2009.5270349.

Fu, X., ja C. Li. 2010. “A String Constraint Solver for Detecting Web Application Vulnerability”. Teoksessa *Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering,* 535–542. doi:10.1.1.204.8145.

Gupta, M. K., M. C. Govil ja G. Singh. 2014. “Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey”. Teoksessa *Recent Advances and Innovations in Engineering (ICRAIE), 2014,* 1–5. Toukokuu. doi:10.1109/ICRAIE.2014.6909173.

Johari, R., ja P. Sharma. 2012. “A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection”. Teoksessa *Communication Systems and Network Technologies (CSNT), 2012 International Conference on,* 453–458. Toukokuu. doi:10.1109/CSNT.2012.104.

Kieyzun, A., P. J. Guo, K. Jayaraman ja M. D. Ernst. 2009. “Automatic creation of SQL Injection and cross-site scripting attacks”. Teoksessa *2009 IEEE 31st International Conference on Software Engineering,* 199–209. Toukokuu. doi:10.1109/ICSE.2009.5070521.

Lin, J. C., ja J. M. Chen. 2007. “The Automatic Defense Mechanism for Malicious Injection Attack”. Teoksessa *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on,* 709–714. Lokakuu. doi:10.1109/CIT.2007.21.

Maniatakos, M. 2013. “Privilege escalation attack through address space identifier corruption in untrusted modern processors”. Teoksessa *Design Technology of Integrated Systems in Nanoscale Era (DTIS), 2013 8th International Conference on,* 161–166. Maaliskuu. doi:10.1109/DTIS.2013.6527798.

Monshizadeh, M., P. Naldurg ja V. N. Venkatakrishnan. 2014. “MACE: Detecting Privilege Escalation Vulnerabilities in Web Applications”. Teoksessa *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security,* 690–701. CCS ’14. Scottsdale, Arizona, USA: ACM. ISBN: 978-1-4503-2957-6. doi:10.1145/2660267.2660337. http://doi.acm.org/10.1145/2660267.2660337.

“ISO27001: Information Security Management System (ISMS) standard”. 2005. *Online: http://www.27000.org/iso-27001.htm* (lokakuu).

Provos, N., M. Friedl ja P. Honeyman. 2003. “Preventing Privilege Escalation”. Teoksessa *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12,* 16–16. SSYM’03. Washington, DC: USENIX Association. http://dl.acm.org/citation.cfm?id=1251353.1251369.

Qian, L., Z. Zhu, J. Hu ja S. Liu. 2015. “Research of SQL injection attack and prevention technology”. Teoksessa *Estimation, Detection and Information Fusion (ICEDIF), 2015 International Conference on,* 303–306. Tammikuu. doi:10.1109/ICEDIF.2015.7280212.

Sadeghian, A., M. Zamani ja S. Ibrahim. 2013. “SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques”. Teoksessa *Informatics and Creative Multimedia (ICICM), 2013 International Conference on,* 265–268. Syyskuu. doi:10.1109/ICICM.2013.52.

Sadeghian, A., M. Zamani ja A. A. Manaf. 2013. “A Taxonomy of SQL Injection Detection and Prevention Techniques”. Teoksessa *Informatics and Creative Multimedia (ICICM), 2013 International Conference on,* 53–56. Syyskuu. doi:10.1109/ICICM.2013.18.

Shahriar, H., ja M. Zulkernine. 2008. “MUSIC: Mutation-based SQL Injection Vulnerability Checking”. Teoksessa *2008 The Eighth International Conference on Quality Software,* 77–86. Elokuu. doi:10.1109/QSIC.2008.33.

Shar, L. K., ja H. B. K. Tan. 2012. “Mining Input Sanitization Patterns for Predicting SQL Injection and Cross Site Scripting Vulnerabilities”. Teoksessa *Proceedings of the 34th International Conference on Software Engineering,* 1293–1296. ICSE ’12. Zurich, Switzerland: IEEE Press. ISBN: 978-1-4673-1067-3. http://dl.acm.org/citation.cfm?id=2337223.2337399.

Shin, Y., L. Williams ja T. Xie. 2006. “SQLUnitGen: SQL Injection Testing Using Static and Dynamic Analysis”. Teoksessa *The 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006).* Marraskuu. doi:10.1.1.143.5685.

Su, Z., ja G. Wassermann. 2006. “The Essence of Command Injection Attacks in Web Applications”. Teoksessa *Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages,* 372–382. POPL ’06. Charleston, South Carolina, USA: ACM. ISBN: 1-59593-027-2. doi:10.1145/1111037.1111070. http://doi.acm.org/10.1145/1111037.1111070.

Thompson, H. H., ja S. G. Chase. 2005. *The Software Vulnerability Guide.* Nide 1st ed. Course PTR. ISBN: 9781584503583. http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=233205&site=ehost-live.

Tsegaye, T., ja S. Flowerday. 2014. “Controls for protecting critical information infrastructure from cyberattacks”, 24–29. IEEE. doi:10.1109/WorldCIS.2014.7028160. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7028160.

Wassermann, G., ja Z. Su. 2007. “Sound and Precise Analysis of Web Applications for Injection Vulnerabilities”. Teoksessa *Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation,* 32–41. PLDI ’07. San Diego, California, USA: ACM. ISBN: 978-1-59593-633-2. doi:10.1145/1250734.1250739. http://doi.acm.org/10.1145/1250734.1250739.

Wei, K., M. Muthuprasanna ja S. Kothari. 2006. “Preventing SQL injection attacks in stored procedures”. Teoksessa *Software Engineering Conference, 2006. Australian.* Huhtikuu. doi:10.1109/ASWEC.2006.40.