OULUN SEUDUN
AMMATTIKORKEAKOULU

Niko Koski

OHJELMISTOKEHITTÄJÄN TIETOTURVAOHJE

OHJELMISTOKEHITTÄJÄN TIETOTURVAOHJE

Niko Koski
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä(t): Niko Koski
Opinnäytetyön nimi: Ohjelmistokehittäjän tietoturvaohje
Työn ohjaaja(t): Juha Alakärppä
Työn valmistumislukukausi ja -vuosi: Kevät 2013
Sivumäärä: 39

Tässä opinnäytetyössä tutkittiin tietoturvauhkia web-ohjelmistokehityksen kannalta. Tavoitteena oli tuottaa tilaajalle dokumentaatio yleisimmistä web-sovelluksissa olevista tietoturvauhista ja menetelmistä niiltä suojautumiseen.

Ensimmäinen vaihe oli tutustua web-sovellusten tietoturvauhkiin ja kerätä tietoa niistä. Kerätyn aineiston perusteella aloitettiin tilaajan tietoturvadokumentin tekeminen. Tietoturvaohjeen teon ohella testattiin tuotannossa olevaa sovellusta tietoturvauhkia vastaan.

Tuloksina syntyi tämä julkinen dokumentaatio, tietoturvaohje, testaustuloksia tilaajan tuotannossa olevalle sovellukselle sekä esimerkkisovelluksia, joissa havainnollistetaan tässä työssä esiteltyjä tietoturvauhkia. Näistä keskeisimpinä mainittakoon SQL-injektiot ja XSS-hyökkäykset.

Asiasanat: tietoturva, ohjeistus, OWASP

ALKULAUSE

Tämän opinnäytetyön aiheesta on kiittäminen Sebitti Oy:tä, joka tarjosi resurssit ja työkalut työn tekemiseen. Työ antoi minulle tilaisuuden tutustua ohjelmisto-kehityksessä hyvin tärkeään osa-alueeseen eli tietoturvaan.

Sebitti Oy on Microsoft-teknologioihin perehtynyt ohjelmistotalo, ja niinpä tässä työssä esitellyt tutkimustulokset pohjaavat vahvalti ASP.NET-kehitysympäristöstä saatuihin kokemuksiin.

Työtä tehdessä yllätyin, kuinka hyvin ASP.NETissä nykyisellään on suojauduttu tässä työssä esitettyjä tietoturvauhkia vastaan. Ohjelmistokehittäjältä vaadittu työn määrä on pyritty selvästi minimoimaan, mikä on hyvä asia, mutta ei poista sitä tosiasiaa, että kehittäjän tulee tiedostaa vaarat ja ymmärtää niiden mahdolliset seuraukset.

Oulussa 1.5.2013

Niko Koski

SISÄLLYS

SANASTO

ACL	Access Control List, lista jonkin objektin käyttöoikeuksien haltijoista.
ASP.NET	Microsoftin web-ohjelmointiin tarkoitettu rajapinta.
Authentication	Todennus, järjestelmälle tunnistautuminen jollain menetelmällä, esim. käyttäjätunnus ja salasana.
Authorization	Valtuuttaminen, rajataan tunnistautuneen käyttäjän pääsyä ohjelmiston eri toimintoihin.
CLR	Common Language Runtime, Microsoftin kehittämä ajoympäristö, joka muun muassa mahdollistaa eri kielellä toteutettujen olioiden kommunikoinnin keskenään.
Data	Tekstiä, numeroita, kuvia, ääntä.
Forms Authentication	ASP.NETistä löytyvä luokka autentikoinnin hallintaan.
Framework	Ohjelmistokehitysrajapinta, luokkien kirjasto, jonka tarkoituksena on tehostaa ohjelmistokehitystä.
Hakkeri	Henkilö, joka koettaa aiheuttaa aineellista tai aineetonta vahinkoa tai hyötyä järjestelmän aukkoja hyväksikäyttäen.
IIS	Internet Information Service, Microsoftin web-palvelinympäristö, jota käytetään Windows-palvelimissa.
Interface	Rajapinta, tarjoaa työkalun tai sovitun toteutuksen jollekin palvelulle tai toiminnolle ja sen käytölle.

Istunto	Web-sovelluksessa käyttäjälle luodaan istunto ja sille yksilöllinen ID, jonka avulla pysytään perillä, kuka tekee ja mitä.
Luokkakirjasto	Kokoelma luokkia, joiden tehtävänä on tehostaa ohjelmistokehitystä tarjoamalla valmiita funktioita ja toteutustapoja.
Komponentti	Sovelluksen jonkin toiminnon tai toiminnallisuuden toteuttava osa.
Natiivisovellus	Jollekin tietylle alustalle tehty, yleensä binäärimuotoinen ajettava sovellus.
OWASP	The Open Web Application Security Project. Järjestö, joka on erikoistunut web-sovellusten tietoturvaan.
Ohjelmisto	Sovelluskokonaisuus, joka voi pitää sisällään niin asiakaspäätteen kuin palvelinympäristönkin.
Pollaus	Ohjelmistokehityksessä pollaamisella tarkoitetaan jonkin muuttuvan tiedon lukemista, kun luettavan tiedon tapahtuma-ajalla ei ole merkitystä.
Web	Käytetään lyhentämään fraasia ”World Wide Web”. Puhekielessä Internet.

1 JOHDANTO

Tänä päivänä ohjelmistokehittäjältä vaaditaan paljon. Ei välttämättä riitä, että saavutetaan tietyt tavoitteet aikataulun puitteissa, vaan ohjelmiston tai tuotteen tilaaja saattaa vaatia jo tiettyjä teknisiä ratkaisuja sekä toteutusmalleja. Yksittäiselle ohjelmoijalle haasteista suurin lienee aikataulussa pysyminen, kuitenkaan laadusta tinkimättä. Ohjelmointityön laatua ja onnistumista voidaan mitata monella tapaa, mutta yrityksen jatkuvuudelle ja uskottavuudelle tärkein kaikista lienee tietoturva.

Sovelluksen tietoturvaan vaikuttaa monta tekijää. Tärkein on varmasti ohjelmistokehittäjän tapa ohjelmoida, ja se on myös samalla vaikein korjattava. Toisena voitaisiin nostaa esille käytettävät työkalut eli käytetyt ohjelmointirajapinnat. Näiden niin kutsuttujen frameworkien tietoturvaan yksittäisen ohjelmistokehittäjän on vaikea vaikuttaa, mutta niiden mahdolliset aukot ja puutteet tulee tiedostaa.

Tietoturvaan liittyvistä uhista on saatavilla paljon tietoa niin internetistä kuin myös alan lehdistä, ja tämän tietävät myös asiakkaat, joille ohjelmistotalo sovelluksia toimittaa. Asiakkaat ovat entistä määrätietoisempia vallitsevista trendeistä tietoturvan suhteen ja osaavat jo vaatia, millaisia menetelmiä ohjelmistotoimittajan tulisi toteuttaa sovelluksissaan. Näin ollen jokaisella ohjelmistotalolla tulisi olla jokin tietty tietoturvamalli, jota se toteuttaa ohjelmistoissaan, olkoonpa se sitten jonkin isomman organisaation hyväksyttämä tai sitten ohjelmistotalon itse tekemä.

Tilaajalla oli tarve tutkimukselle, jossa olisi kerrottu kaikista yleisimmistä tietoturvauhista, ja niinpä tämä opinnäytetyö käsittelee sovellusten tietoturvaan liittyviä tekijöitä ja valottaa samalla millaisia tietoturva-aukkoja on olemassa ja miten niitä voidaan torjua.

2 TIETOTURVA

Ensisijaisesti tietoturvallisuus lähtee yrityksestä itsestään ja sen arvoista. Yrityksellä voi olla tietyt standardit, jotka jokaisen tuotoksen pitää täyttää. Näitä voivat olla muun muassa riittävä dokumentoinnin taso, testausmenetelmät ja noudatettava arkkitehtuuri ohjelmistossa. Yksittäisen ohjelmistokehittäjän vastuulla on näiden standardien toteuttaminen ja noudattaminen. (1.)

Tietoturvasta puhuttaessa ohjelmistokehittäjän pitää pystyä jo suunnitteluvaiheessa kartoittamaan järjestelmässä yksittäisten toimintojen mahdolliset aukot ja se miten joku voi niitä hyödyntää. Tietoturvalähtöisyyden tulisi näkyä kaikkialla sovelluksen osa-alueissa. (1.)

Ohjelmistokehittäjä on viime kädessä vastuussa ohjelmiston sisältämän datan

- **luottamuksellisuudesta**: dataa pystyvät käsittelemään vain sellaiset henkilöt, joilla on siihen oikeus
- **eheydestä:** tieto ei saa muuttua jonkin hyökkäyksen tai vastaavan tilanteen seurauksena
- **saatavuudesta:** data pitää olla saatavilla silloin kun käyttäjä sitä tarvitsee. (1.)

Näiden **kolmen kulmakiven** tulisi heijastua kaikkiin ohjelmiston komponentteihin.Yhtenä esimerkkinä voidaan käyttää sovelluksessa tai järjestelmässä vaadittavaa sisäänkirjautumista eli käyttäjäautentikointia. Käyttäjän täytyy ensin tunnistautua ohjelmistolle voidakseen käsitellä dataa ja suorittaa toimenpiteitä. Tunnistautumisen edellytyksenä on, että järjestelmä on **saatavilla**, eli käyttäjä pystyy lähettämään kirjautumistietonsa ja tämän jälkeen vastaanottamaan dataa. Järjestelmän täytyy pitää huoli, että tämä data säilyy **eheänä** transaktioiden välillä. Näiden ehtojen täyttyminen mahdollistaa käyttäjän pääsyn järjestelmään ja näin ollen **luottamukselliseen** dataan käsiksi. (1.)

3 WEB-SOVELLUS

Alkuaikoina www-sivut rakentuivat lähinnä staattisista web-sivuista. Staattinen sivusto ei mahdollista vuorovaikutusta käyttäjän kanssa, joten sellaisenaan niitä ei voi käyttää toteuttamaan ohjelmistoa. Tässä apuun tulevat palvelinympäristössä suoritettavat ohjelmointikielet kuten PHP ja ASP.NET. Näiden kahden kaltaiset ohjelmointirajapinnat mahdollistavat vuorovaikutteisten web-sovellusten tekemisen. (2; 3.)

Web-sovelluksella tarkoitetaan yleensä sovellusta, jota käyttäjät voivat käyttää internetin yli tai vaihtoehtoisesti yrityksen intrajärjestelmässä. Yksinkertaisimmillaan tämä voi tarkoittaa sitä, että käyttäjän tarvitsee vain avata järjestelmänsä web-selain ja siirtyä osoitteeseen, jonka takaa sovelluksen web-käyttöliittymä löytyy. (2; 3.)

Tällaisten sovellusten etuna on alustariippumattomuus, eli sovellus ei ota kantaa siihen, millä laitteella käyttäjä sovellusta käyttää. Web-sovellus on aina saavutettavissa, jos käyttäjällä on jokin web-selain käytettävissä. Koodin kannalta se voi muistuttaa paljon tavallista työpöytäsovellusta, erona se, että sitä käytetään käyttäjän valitseman web-selaimen kautta. Sovelluskehittäjän harteille jää pitää huoli, että sovellusta pystyy käyttämään erityyppisillä laitteilla, hyvänä esimerkkinä laitteet, joissa on kosketusnäyttö. (2; 3.)

Tietoturva

Tietoturvasta huolehtiminen on ensisijaisen tärkeää web-sovelluksissa, koska ne ovat helpommin saavutettavissa. Mikäli web-sovellus on saatavilla julkisessa verkossa, sen potentiaalinen käyttäjämäärä kasvaa automaattisesti seitsemään miljardiin. Näin ollen on tärkeää, että itse palvelinympäristön suojaus on kunnossa mutta myös sovellus täyttää tietyt laatukriteerit. Jokaisessa web-sovelluksessa, jossa on luottamuksellista dataa, tulisi toteuttaa vähintäänkin käyttäjätunnistukseen liittyvät osa-alueet, joihin kuuluu autentikointi, valtuutus ja istunnonhallinta. (4.)

Autentikointi

Autentikoinnilla varmistetaan käyttäjän identiteetti ja se määrää mitkä sivut ja toiminnot käyttäjällä on käytössään. Jos sovelluksen käyttäjä väittää olevansa Jaska, järjestelmässä täytyy olla tapa autentikoida, että hän todella on Jaska ja hän saa Jaskalle tarkoitetut työkalut käyttöön. Autentikointitapoja voivat esimerkiksi olla

- **käyttäjätunnus-salasana-yhdistelmä**, jossa käyttäjällä on oma henkilökohtainen käyttäjätunnus sekä tälle hänen itsensä asettama salasana. Käyttäjän syöttämiä merkkejä verrataan tietokannassa oleviin.
- jonkin **kolmannen osapuolen tarjoama identiteettipalvelu**, esimerkiksi Facebook
- **Windows-autentikointi** Microsoftin sovellusympäristöissä. (4; 5.)

Valtuuttaminen

Järjestelmä on varmistanut kirjautuneen käyttäjän olevan Jaska. Jaskalla on valtuudet tiettyihin sovelluksen osa-alueisiin ja järjestelmässä täytyy pitää kirjaa koko istunnon ajan Jaskan tekemisistä ja menemisistä. Tämä on ohjelmistokehittäjän näkökulmasta huomattavasti haastavampi toteutettava osa-alue kuin autentikointi koska sovelluksessa voi olla monikerroksisia käyttäjätasoja. (4.)

Istunnonhallinta

Web-sovelluksen kehityksessä on erilaiset lähtökohdat kuin jollekin tietylle järjestelmälle tuotetussa natiivisovelluksessa, ja kehittäjä joutuu miettimään erilaisia tapoja datan käsittelyyn web-sovellusten tilattoman luonteen vuoksi (3).

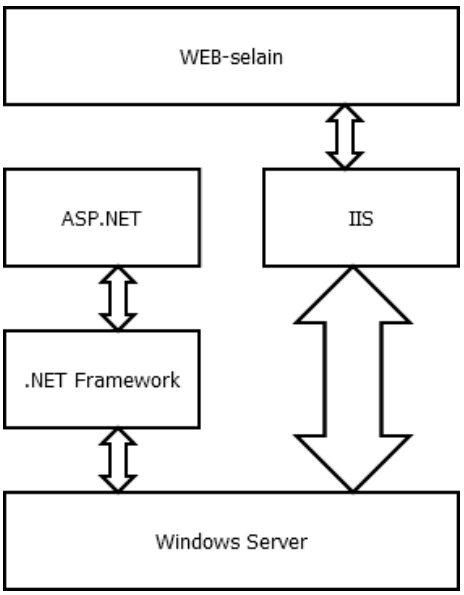
Tietylle alustalle tehdyssä natiivisovelluksessa tietoa voidaan käsitellä ajettavan järjestelmän muistissa kun taas web-sovellus on vain kokoelma ladattavia sivuja jotka eivät tiedä toistensa olemassaolosta mitään. HTTP ei mahdollista muistinvarauksia, ja tästä johtuen web-sovelluksissa täytyy toteuttaa jonkinlainen tilataltio jossa käsiteltävä data ja käyttäjän tekemät toiminnot pysyvät tallessa sivulatausten välillä. (4.)

4 ASP.NET:N TIETOTURVA

ASP.NET on Microsoftin .NET-tuoteperheeseen kuuluva web- ohjelmistorajapinta. Se kehitettiin ASP:n seuraajaksi, ja tällä hetkellä se on PHP:n ohella yksi käytetyimmistä web-sovelluskehitysalustoista. Sen vahvuuksina voidaan pitää Windows-työpöytäsovelluskehittäjille entuudestaan tuttuja kontrolleja ja niiden kokoelmia sekä tapahtumankäsittelijöihin nojaavaa kehitystapaa joka on poikkeuksellinen web-sovelluksissa. (5; 6.)

Erilaisten ohjelmistorajapintojen tehtävänä on antaa tarvittavat työkalut jotta ohjelmistokehittäjä selviäisi tehtävistään mahdollisimman tehokkaasti. Ne antavat myös valmiita ratkaisuja sovelluksen arkkitehtuurin toteutukseen. ASP.NETissä näitä on esimerkiksi Web Forms ja MVC. (5; 6.)

Kuvassa 1 nähdään ASP.NET-sovelluksen elinkaari. IIS-palvelimen saatua pyynnön ladattavasta sivusta siihen liitetty ASP.NET-kielinen ohjelmakoodi suoritetaan ja tuloksena renderöidään html-kuvauskielinen web-sivu jonka käyttäjän web-selain näyttää.

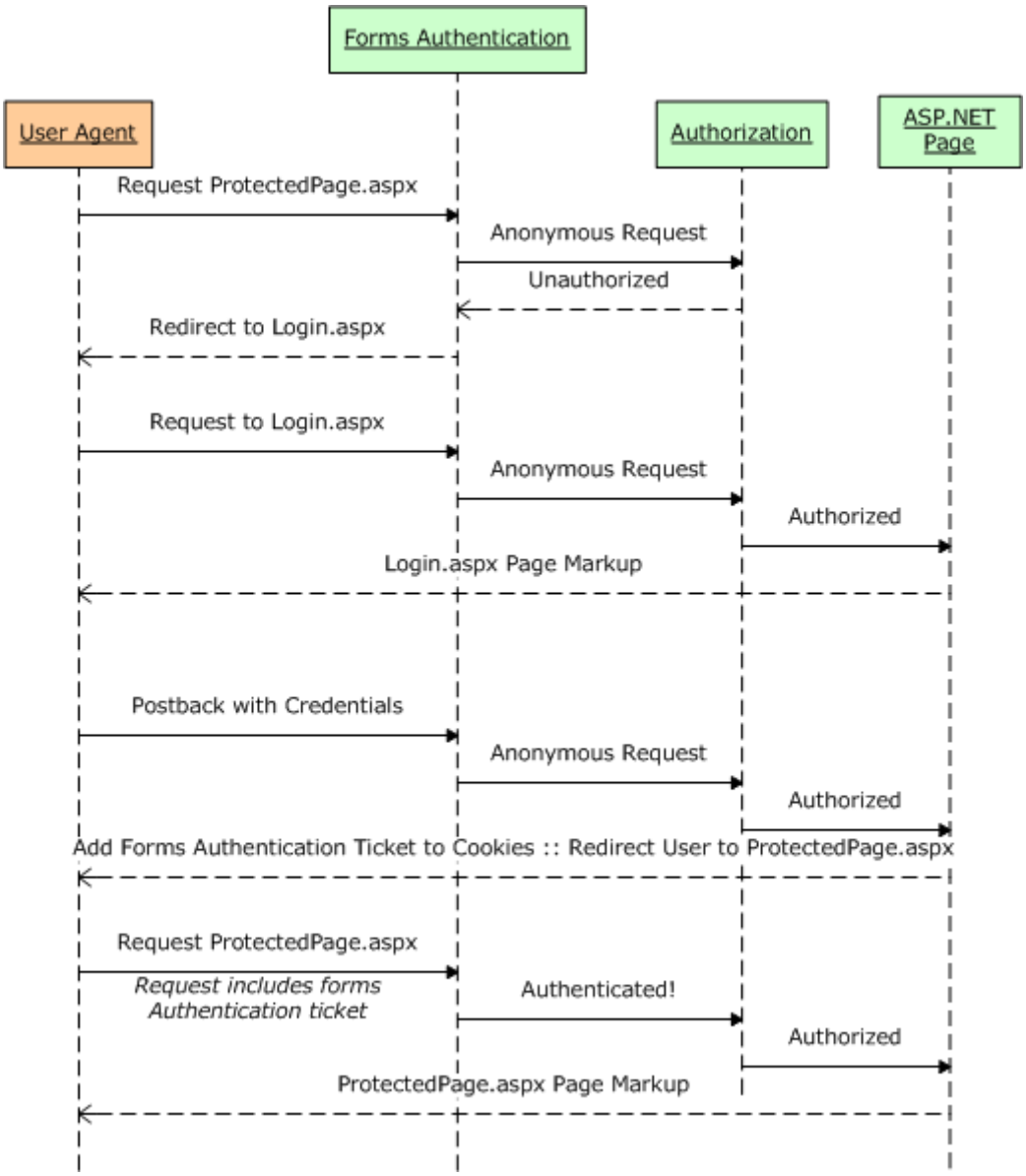


KUVA 1. ASP.NET-sovelluksen arkkitehtuuri (6)

Autentikointi

Ohjelmistokehittäjä voi käyttää .NET-kirjastosta löytyvää Forms Authentication -komponenttia ASP.NET-sovelluksessa autentikoinnin toteuttamiseen. Tosin mikään ei estä toteuttamasta omaa tunnistautumismenetelmää kehittäjän niin halutessa. (5; 6.)

Jos sovelluksen käyttäjä yrittää päästä käsiksi johonkin sovelluksen osa-alueeseen, johon hänellä ei ole pääsyä ilman kirjautumista, Forms Authentication ohjaa käyttäjän kirjautumissivulle (kuva 2). Käyttäjän antamia kirjautumistietoja voidaan verrata tietokantaan ja jos vastaavuus löytyy, voidaan luoda Forms Authentication -tiketti. Tämä tiketti pitää sisällään käyttäjän tunnistamiseen tarvittavan datan, kuten käyttäjänimen, ja se säilötään käyttäjän evästeisiin, jotta myöhemmin HTTP-pyyntöjä tehdessä palvelin tunnistaa kirjautuneen käyttäjän. (5.)



KUVA 2. Käyttäjän pyytäessä sivustoa, jonne vaaditaan Forms Authentication -tiketti, hänet ohjataan kirjautumissivulle (5)

Valtuuttaminen

ASP.NET tarjoaa natiivina kaksi tapaa valtuuttamiselle:

1. File authorization -moduuli, joka tarkastaa ladattavan sivun ACL:stä onko pyynnön tekijällä oikeutta kyseiseen sivuun.

2. URL authorization -moduuli, jota voidaan käyttää kokonaisten polkurakenteiden oikeuksienhallintaan. (6; 10.)

Ohjelmistokehittäjä voi käyttää näitä kahta menetelmää toteuttaessaan käyttöoikeuksienhallintaa web-sovelluksessaan. Mutta kuten todettua, oman käyttöoikeudenhallinan toteuttaminen on myös mahdollista, mikäli kokee edellä mainitut riittämättömiksi. (5.)

Istunnonhallinta

ASP.NETissä on useita istunnonhallintaa helpottavia tekniikoita joita ohjelmistokehittäjä voi käyttää hyödykseen toteuttaessaan vuorovaikutteista web-sovellusta.

Selainpuolen tekniikoita ovat seuraavat:

1. View State: käytetään säilyttämään nykyisen sivun kontrollien ja käyttäjän syöttämien tietojen tila saman sivun uudelleenlatausten välillä. Tiedot hashataan merkkijonoksi ja asetetaan piilotettuun kenttään.
2. Control State: sama kuin edellinen mutta yksittäisen kontrollin tilan säilömiseen.
3. Query Strings: parametreja jotka asetetaan selaimen osoiteriville tiedon siirtämistä varten sivulatausten välillä.

Palvelinpuolella voidaan käyttää näitä:

1. Application State: globaali tilataltio joka on saatavilla kaikkien sivulatausten välillä. Tämä on yhteinen kaikkien web-sovelluksen käyttäjien välillä.
2. Session State: istuntokohtainen tilataltio, eli rajattu kyseisen käyttäjän avaamaan istuntoon. (5; 6; 10.)

Kaikki edellä mainitut tekniikat ovat vain kokoelma avain-arvoasetuksia ja sellaisenaan niiden käyttäminen on työlästä, mutta ne mahdollistavat oman luokkakirjaston rakentamisen niiden päälle. Tällöin niihin voidaan luoda instansseja luokista jotka säilyttävät tilansa sivulatausten välillä.

5 TOP 10 -TIETOTURVAUHAT

Lukuisat järjestöt pyrkivät pitämään ohjelmistokehittäjät varpaillaan vallitsevista trendeistä tietoturvan saralla. Yksi näistä on OWASP. Se on voittoa tavoittelematon järjestö jonka kantava ajatus on pyrkiä tuottamaan oppaita ja ohjeistuksia ohjelmistokehittäjille ja näin ollen parantamaan web-sovellusten tietoturvaa. Sen tärkein portaali on owasp.org-sivusto joka on Wikipedian kaltainen avoin wiki-sivusto, josta löytyy tietoturvaan liittyviä kirjoja ja muuta materiaalia. Yksi tärkeä opas on OWASP:n ylläpitämä Top 10 -lista. Tämä lista kokoaa yksiin kansiin tietoa yleisimmistä tietoturvauhista web-sovelluksissa. (7.)

Listaa päivitetään säännöllisesti vastaamaan nykypäivän trendejä (kuva 3).

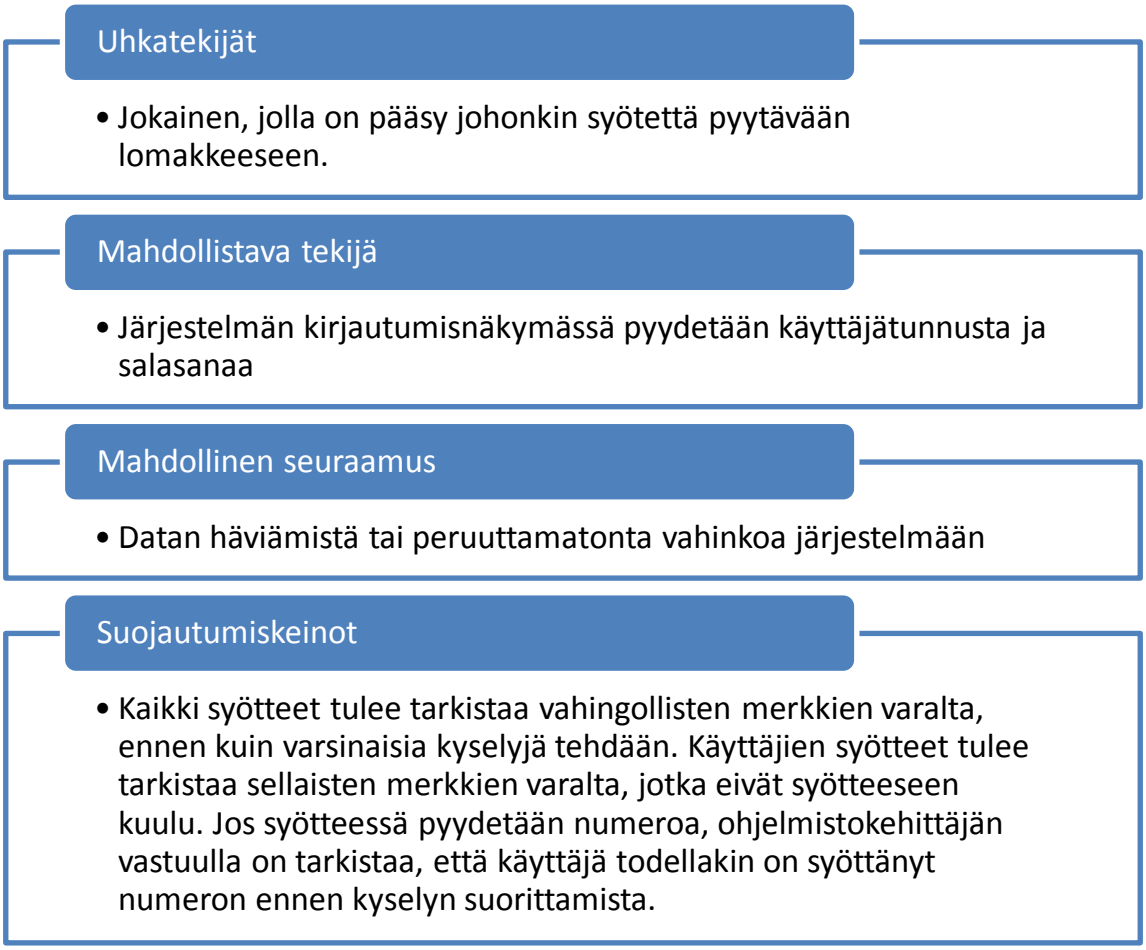
Mapping from 2007 to 2010 Top 10

OWASP Top 10 – 2007 (Previous)		OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	↑	A1 – Injection
A1 – Cross Site Scripting (XSS)	↓	A2 – Cross Site Scripting (XSS)
A7 – Broken Authentication and Session Management	↑	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	=	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	=	A5 – Cross Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	+	A6 – Security Misconfiguration (NEW)
A10 – Failure to Restrict URL Access	↑	A7 – Failure to Restrict URL Access
<not in T10 2007>	+	A8 – Unvalidated Redirects and Forwards (NEW)
A8 – Insecure Cryptographic Storage	↓	A9 – Insecure Cryptographic Storage
A9 – Insecure Communications	↓	A10 – Insufficient Transport Layer Protection
A3 – Malicious File Execution	-	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	-	<dropped from T10 2010>

KUVA 3. Kahden viimeksi julkaistun Top 10 -listan erot (12)

5.1 Injektio

Injektio on menetelmä ohjelmistossa olevien käyttäjäsyötettä vaativien lomakkeiden hyödyntämiseksi järjestelmiin tunkeutumisessa. Sovelluksessa voidaan pyytää käyttäjältä jotain syötettä ja tunkeutuja voi käyttää tätä hyödykseen syöttämällä haitallista koodia kyselyyn. Useimmiten puhutaan SQL-injektioista, mutta myös LDAP-injektiot ovat yleisiä. Sovelluksessa voi olla hakutoiminto, joka pyytää käyttäjältä merkkijonoa jota sitten haetaan palvelimelta. Tunkeutuja voi tietynlaisella syötteellä katkaista ohjelmistokehittäjän kyselyn ja sen sijaan ajaa oman haitallisen kyselyn. Jos käyttäjän antamia syötteitä ei tarkasteta, tunkeutujan on mahdollista syöttää omia kyselyjä näihin muutoin tavallisia kirjautumistietoja vaativiin kenttiin. Kuvassa 4 nähdään tiivistelmä injektiosta. (9.)

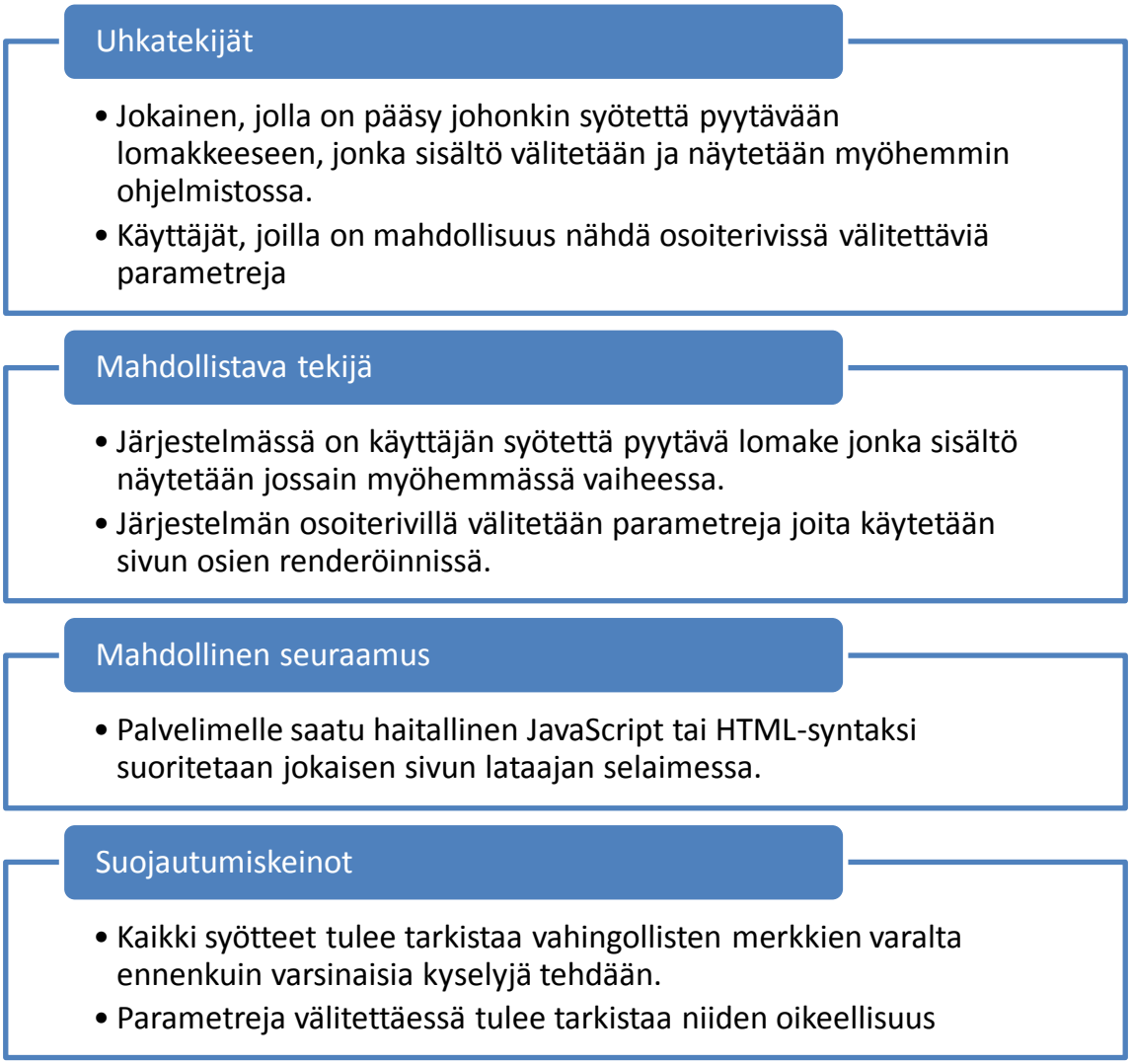


KUVA 4. SQL-injektion uhkatekijät ja suojautumiskeinot

5.2 Cross-Site Scripting

XSS:stä eli Cross-Site Scriptingistä puhuttaessa tarkoitetaan sivuston sisällön muokkaamista hakkerin tarkoitusperien mukaiseksi. Tämä tarkoittaa yleensä oman merkkikielen tai skriptin saamista ajetuksi sovelluksessa (kuva 5) (7; 9).

Kaksi yleisintä keinoa tähän on haittakoodin saaminen palvelimelle syötettä pyytävän lomakkeen avulla tai osoiterivin kautta, jos sitä käytetään parametrien välitykseen (9).

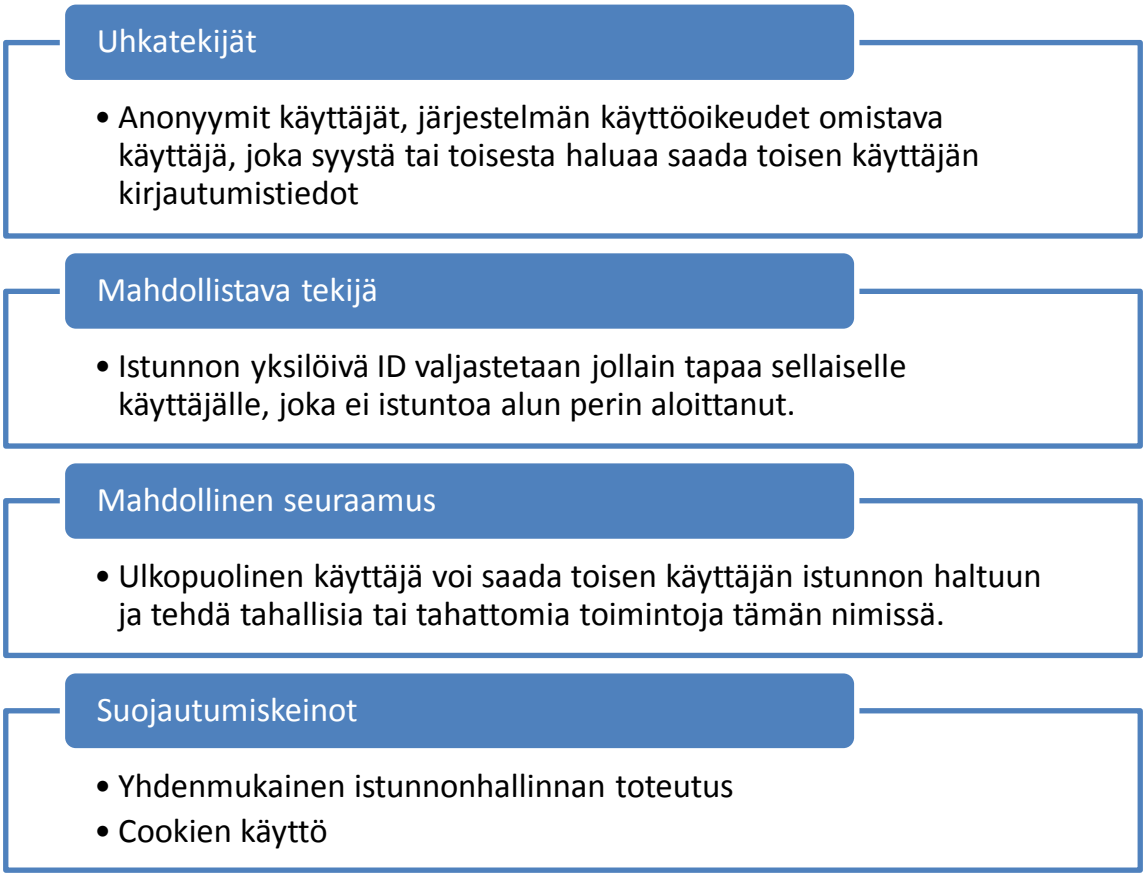


KUVA 5. XSS tietoturvariskinä

5.3 Autentikointi- ja istuntohallintaongelmat

Ohjelmiston autentikointi- ja istuntohallinnan toteutuksessa voi olla puutteita, jotka voivat avata tunkeilijalle mahdollisuuden saada toisen käyttäjän istunto haltuun (7; 9).

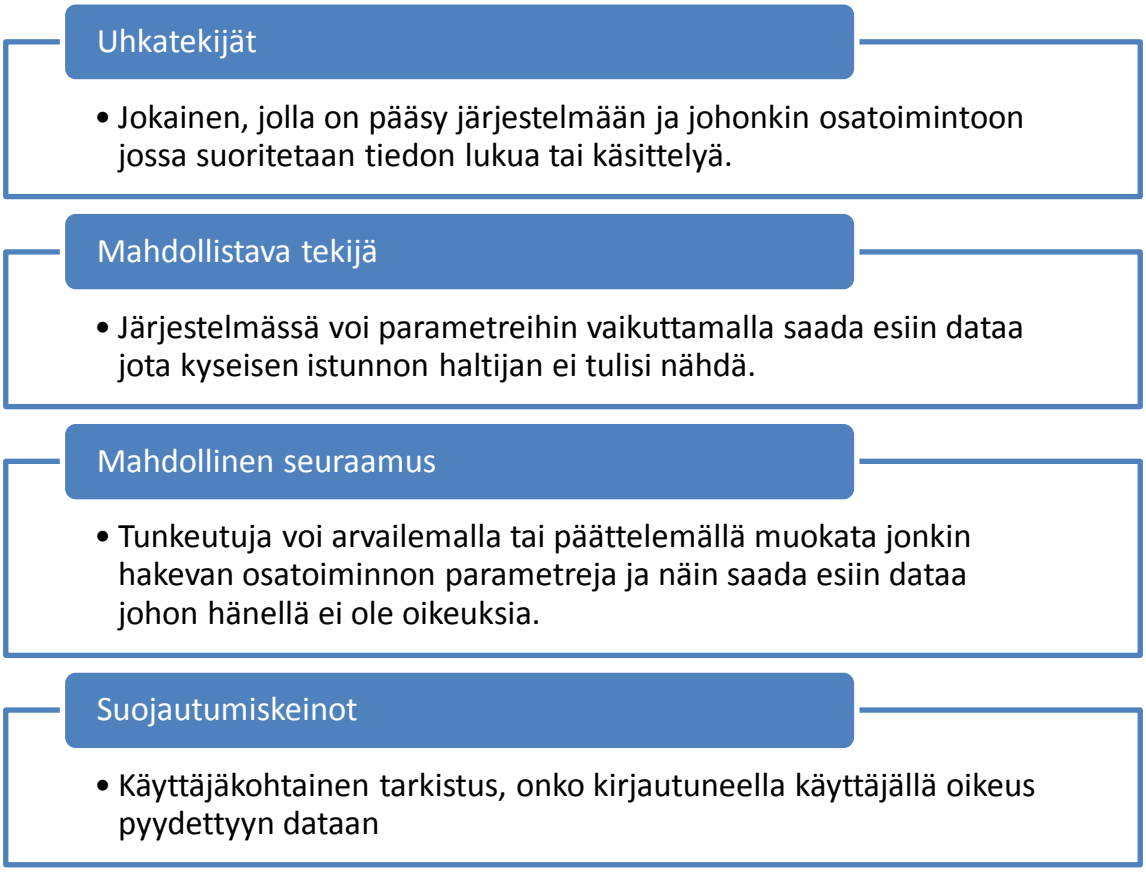
Yleinen harhaluulo on, että evästeet heikentäisivät jotenkin sovelluksen tietoturvaa, mutta tosiasiassa ne lisäävät sitä. Evästeet ovat järkevin menetelmä taltioida käyttäjän istuntoon liittyvää dataa, kuten yksilöllinen istunto-ID. Toinen vaihtoehto olisi taltioida istunto-ID selaimen osoiteriville. Tämä tekee sen tahallisesta tai tahattomasta jakamisesta helpompaa (kuva 6) (7).



KUVA 6. Autentikoinnin- ja istunnonhallinnan peruselementit

5.4 Tarkistamattomat dataviittaukset

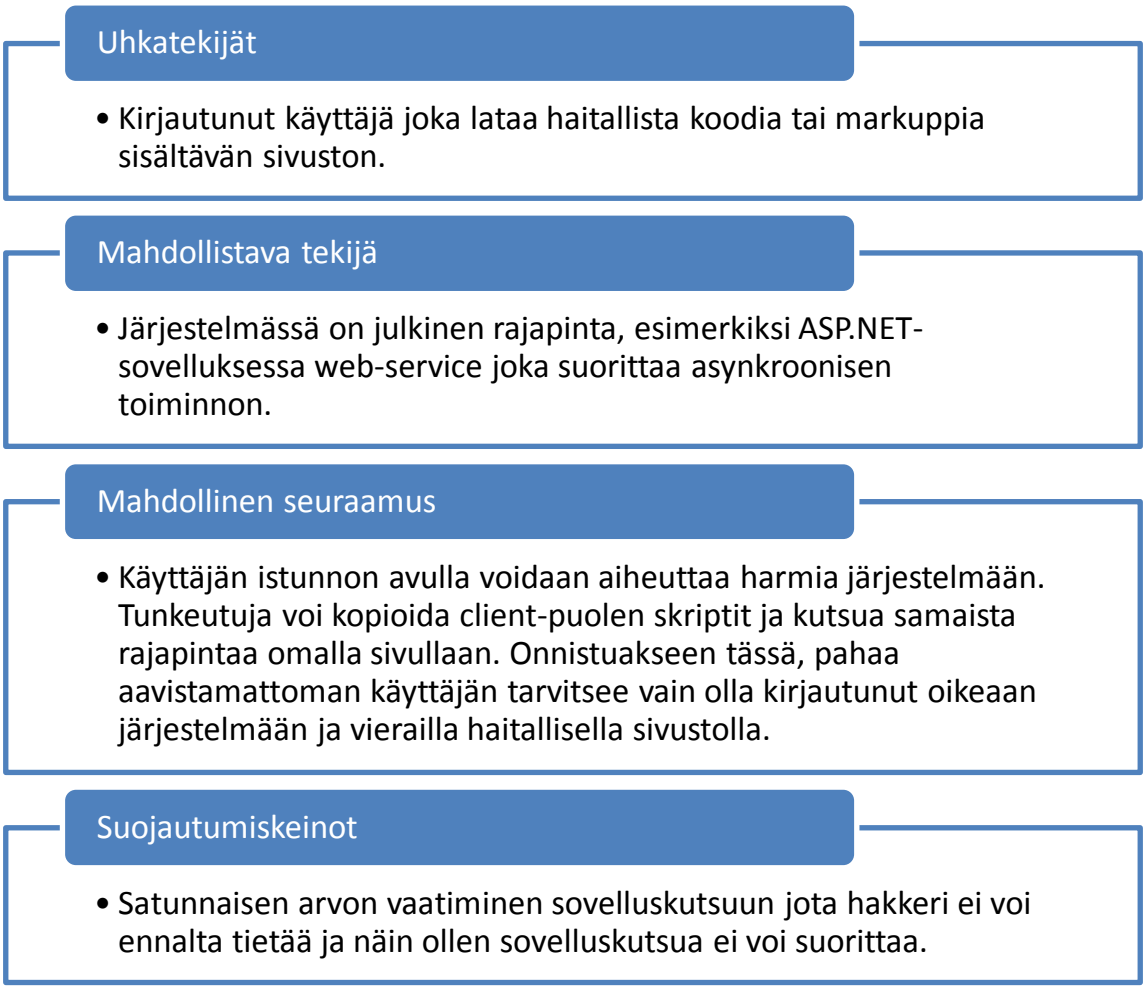
Ohjelmistossa tulisi aina dataa tarkastellessa varmistaa, että pyynnön lähettäneellä käyttäjällä on siihen käsittelyoikeudet. Tunkeutuja voi arvailemalla tai muuttamalla parametreja saada esille dataa johon hänellä ei ole käyttöoikeuksia. Esimerkiksi kun ohjelmistossa ladataan näkymään jonkin tiedoston sisältö (kuva 7) (9).



KUVA 7. Tarkistamattomat dataviittaukset, uhkatekijät ja suojautumiskeinot

5.5 Cross-Site Request Forgery

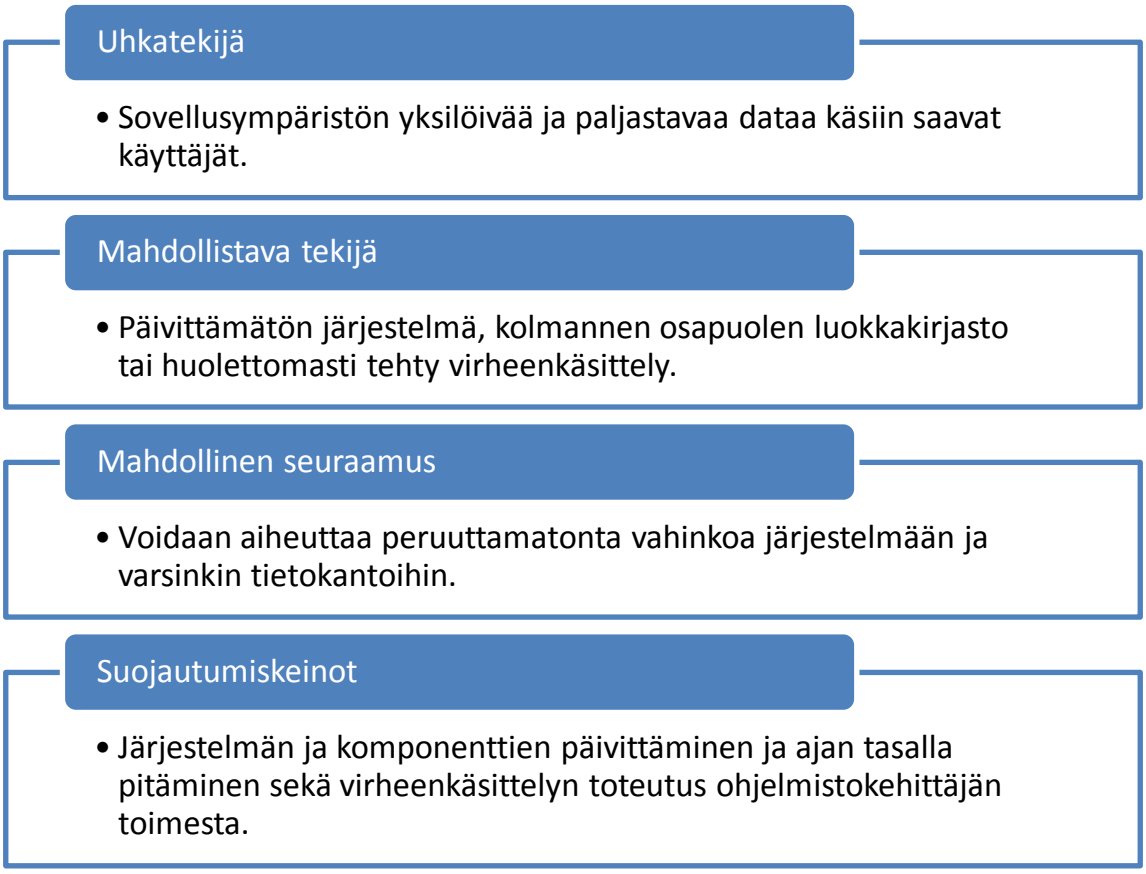
Cross-Site Request Forgeryssä, tuttavallisemmin CSRF:ssä, pyritään saamaan luotettuun järjestelmään kirjautunut käyttäjä suorittamaan haitallista koodia toisella ei-luotetulla sivustolla. AJAX:n suosion kasvaessa tämä menetelmä on tullut suosituksi hyökkäysmenetelmäksi. Asynkronisessa sisällönpäivityksessä joudutaan tekemään julkinen rajapinta jota voidaan kutsua esimerkiksi JavaScriptillä. Näin ollen joudutaan luopumaan joistain tarkistusmenetelmistä ja tämä avaa ovet CSRF:lle (kuva 8) (9.).



KUVA 8. Cross-Site Request Forgery kiteytettynä

5.6 Järjestelmän virheellinen ylläpito

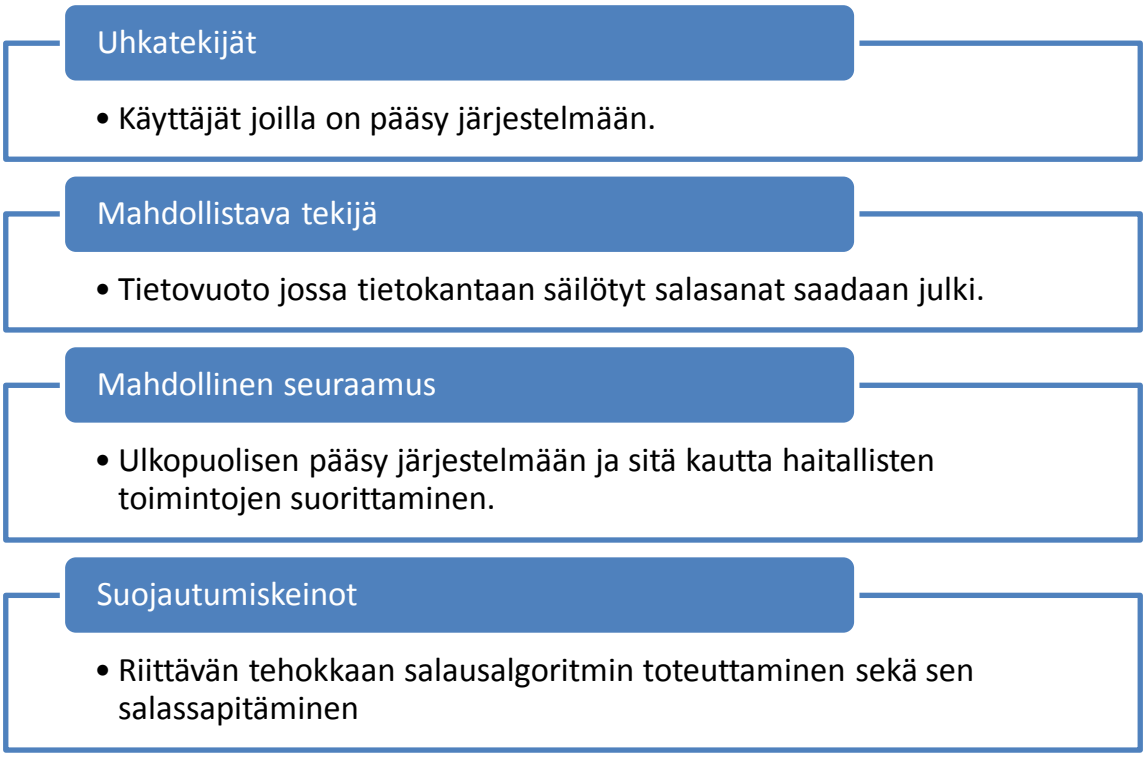
Järjestelmän virheellinen ylläpito on laaja käsite ja se pitää sisällään paljon sovelluksen ulkopuolisia tekijöitä. Näitä on esimerkiksi palvelinympäristön ajan tasalla pitäminen eli yksinkertaisesti päivitysten ajaminen järjestelmään, kun niitä on saatavilla. Toinen tärkeä tekijä on käytettyjen frameworkien ajan tasalla pitäminen, esimerkkinä .NET-rajapinta ja sovelluksessa mahdollisesti käytettävät kolmannen osapuolen luokkakirjastot. Kolmantena voitaisiin mainita ohjelmistossa toteutettu virheenkäsittely, mitkä virheet näytetään, miten ja kenelle (kuva 9) (9).



KUVA 9. Järjestelmän virheellisen ylläpidon tekijät ja seuraamus

5.7 Virheellinen datakryptaus

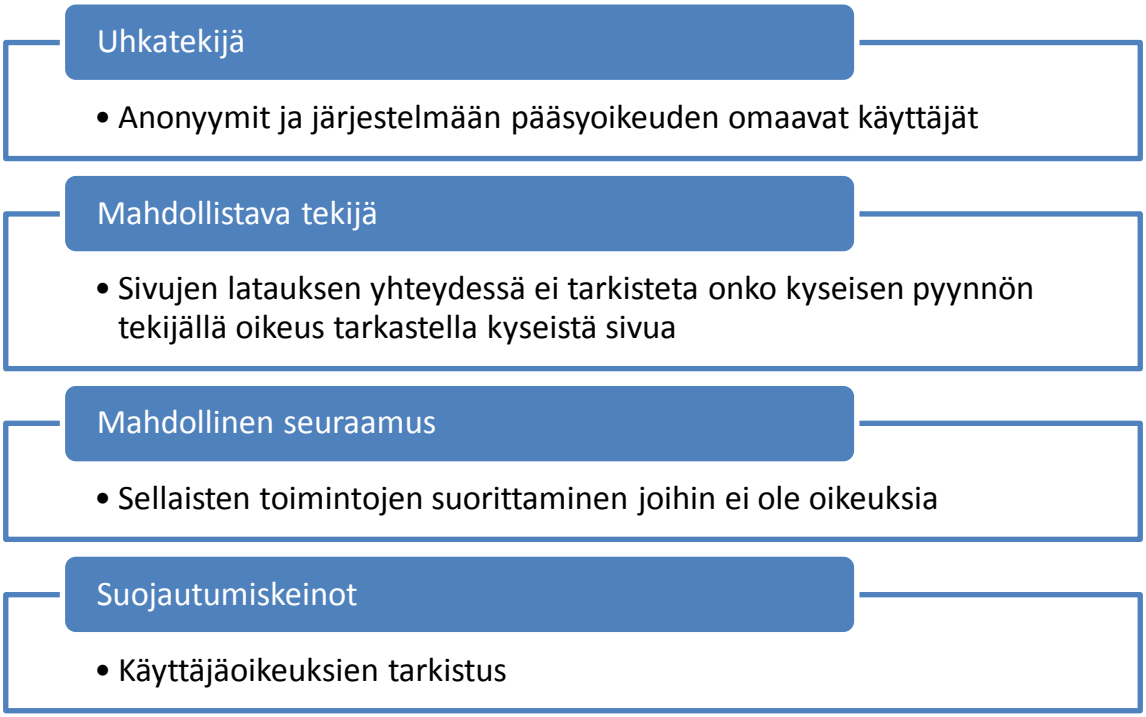
Järjestelmässä tulisi luottamuksellinen tieto kryptata jollain tavalla, hyvänä esimerkkinä tietokannassa olevat salasanat. Ohjelmointirajapinnoista löytyy yleensä jokin kryptauksen mahdollistava luokkakirjasto. Kryptauksen lisäksi tulee pitää huoli, että ulkopuolisen ei ole mahdollista selvittää kryptaukseen käytettyä menetelmää (kuva 10) (9).



KUVA 10. Datan kryptaamisessa muistettavaa

5.8 Suojaamattomat web-osoitteet

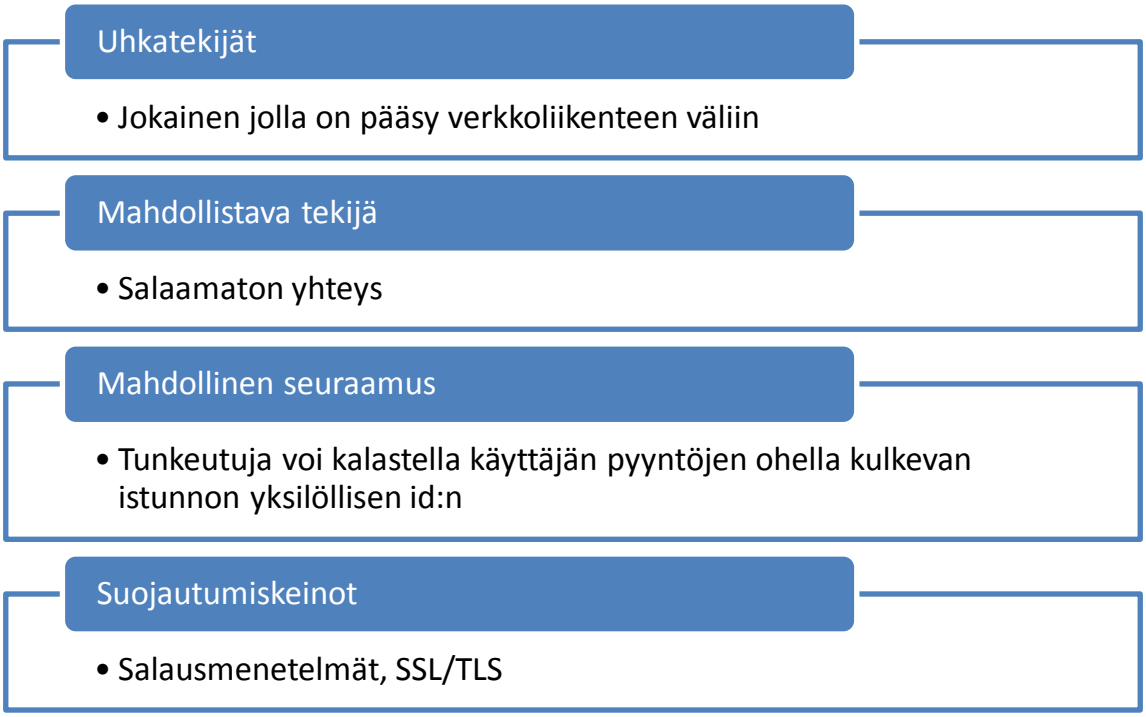
Suojaamaton web-osoite tarkoittaa että järjestelmässä ei tarkasteta onko kirjautuneella tai kirjautumattomalla käyttäjällä oikeus tarkastella pyytämäänsä sivua. Järjestelmässä täytyy jokaisella sivulatauksella varmistaa käyttäjän identiteetti eli tarkistaa, että sivun pyytäjällä on oikeus katsella haluttua sivua. Kehittäjän ei tulisi luottaa siihen ettei sivua löydetä tai että pelkkä piilottaminen riittäisi (kuva 11) (9).



KUVA 11. Suojaamattomat web-osoitteet

5.9 Luottamuksellisen tiedon siirtäminen suojaamatta

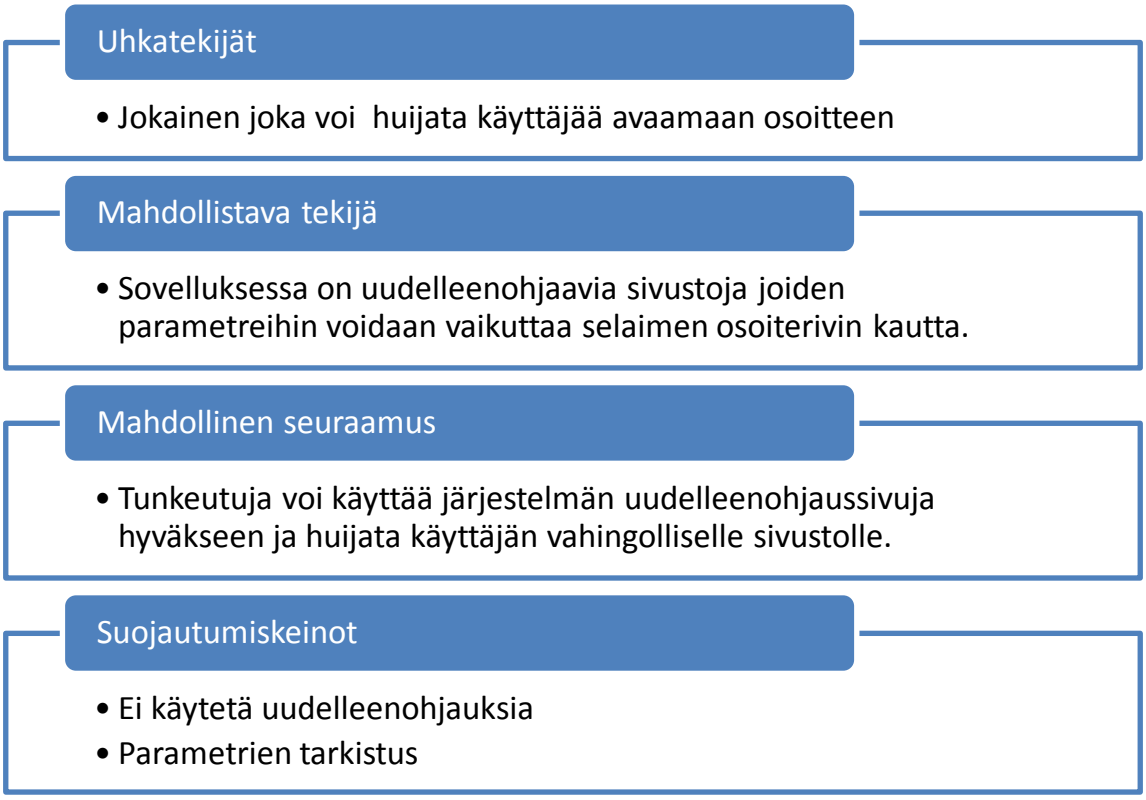
Jos tietoa siirretään järjestelmässä tai järjestelmien välillä salaamatta, tunkeutuja voi onkia tietoa kuuntelemalla verkkoliikennettä tähän tarkoitukseen tehdyllä työkalulla (kuva 12) (9).



KUVA 12. Salaus tiivistettynä

5.10 Tarkastamattomat uudelleenohjaukset

Joskus ohjelmistossa voi olla tarve toiminnolle joka uudelleenohjaa käyttäjän jollekin toiselle sivulle sen sijaan että lataus käynnistyisi käyttäjän valitseman linkin pyytämänä. Tällaiset uudelleenohjaukset väärin toteutettuna voivat olla tietoturvariski järjestelmälle (kuva 13) (9).



KUVA 13. Tarkastamattomat uudelleenohjaukset

6 WEB FORMS -TESTAUS

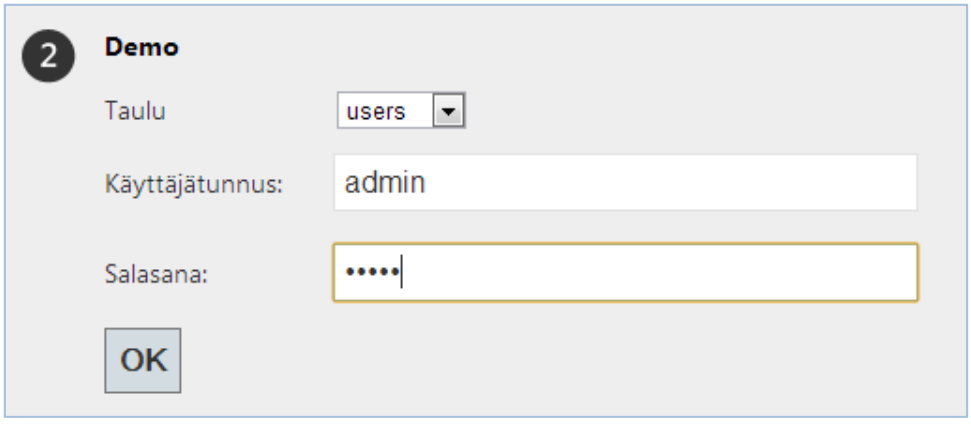
Tilaajalle tehtiin testaustyö tuotannossa olevalle sovellukselle käyttäen apuna OWASPin Top 10 -listaa. Tuotantosovelluksen testaustuloksista ei voi kirjoittaa tässä julkisessa dokumentaatiossa, joten tässä työssä esitellyt testaustulokset on kerätty Web Forms -tyyppisestä demosovelluksesta. Demosovellus jää havainnollistamisvälineeksi tilaajalle ja sen työntekijöille.

Demosovelluksessa käytetään vakioasetuksia jolloin saadaan realistisia havaintoja siitä miten ASP.NETillä toteutettu web-ohjelmisto on suojautunut aiemmin esitettyjä tietoturvauhkia vastaan. Sovellukseen tehtiin toiminnallisuuksia jotka mahdollistaisivat aiemmin esiteltyjen tietoturvauhkien toteutumisen. Tuloksina syntyi havaintoja siitä miten ASP.NETissä on suojauduttu aiemmin esitettyjä tietoturvauhkia vastaan ja mitä toimenpiteitä ohjelmistokehittäjältä vaaditaan jottei riskiskenaariot pääsisi toteutumaan.

6.1 Kirjautuminen

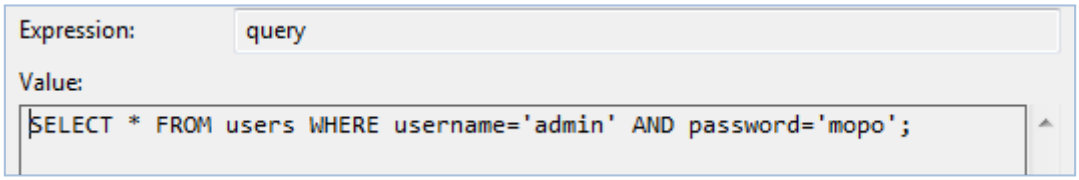
Tapahtumaskenaario

Sovelluksessa on käyttäjän kirjautumistietoja pyytävä lomake, joka sisältää kaksi tekstikenttää. Ensimmäisessä tekstikentässä pyydetään käyttäjän käyttäjätunnusta ja toisessa salasanaa (kuva 14).



KUVA 14. Kirjautumislomake, jossa pyydetään käyttäjätunnuksia

Käyttäjän syöttäessä pyydetyt tiedot ja painaessa kirjautumispainiketta ohjelmistossa aloitetaan SQL-kyselyn suorittaminen.



KUVA 15. SQL-kysely

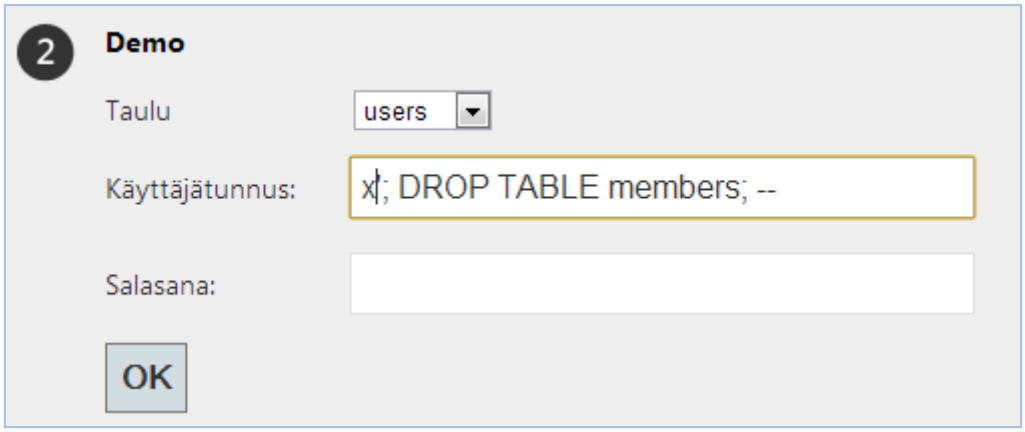
Riskitekijät

Varsin yleinen tapa luoda SQL-kyselyjä millä tahansa ohjelmistoalustalla on kirjoittaa valmis SQL-kysely käyttöliittymäkoodissa ja täydentää sitä käyttäjän syötteillä (kuva 16).

```
Dim query As String = "Select * FROM users WHERE user = '" &
    txtUser.Text & "' AND password = '" &
    txtPassword.Text & "';"
```

KUVA 16. SQL-kyselyn muodostaminen käyttöliittymäkoodissa

Tällaisenaan kyselyn käyttäminen on erittäin vaarallista, koska ei voida luottaa siihen että syötteet ovat turvallisia, kuten kuvassa 17 nähdään.



KUVA 17. Tämä merkkijono joutuessaan SQL-kyselyn parametriksi aiheuttaisi alkuperäisen kyselyn katkaisemisen ja members-taulun tuhoamisen.

Havainnot

ASP.NET ei ota kantaa miten käyttäjä tekee SQL-kyselyt, mutta tarjoaa pari tapaa tehdä niistä turvallisempia.

1. Whitelistin käyttö eli väärien merkkien parsiminen syötteistä regular expressionia hyväksi käyttäen. Esimerkiksi jos käyttäjältä pyydettäisiin lomakkeessa positiivista kokonaislukua, sovelluksessa tulee tarkistaa että kyseinen syöte on positiivinen kokonaisluku (kuva 18).

```
Dim positiveIntRegex = New Regex("^0*[1-9][0-9]*$")
If positiveIntRegex.IsMatch(txtUser.Text) Then
```

KUVA 18. Oheinen regular expression -määrittely sallisi vain positiivisen kokonaisluvun syötteessä

2. Stored Procedure -komentojen ja nimettyjen parametrien käyttö kuvan 19 mukaisesti.

```
CREATE PROCEDURE GetUser
@user_id INT
|
```

KUVA 19. Ennalta luotu proseduuri, joka kelpuuttaa vain kokonaisluvun parametrikseen. Muut tyypit aiheuttavat virheen eikä kyselyä suoriteta.

Stored-proseduurien käyttö parsii vääriä datatyyppejä ja lisää tietoturvaa. Se on whitelistin kanssa lähes pettämätön tapa suojautua injektioita vastaan.

```
Dim conn = New SqlConnection(connstring)
Using command = New SqlCommand("GetUser", conn)
    command.CommandType = CommandType.StoredProcedure
    command.Parameters.Add("@user_id", SqlDbType.Int).Value = user
    command.Connection.Open()
    data = command.ExecuteReader()
```

KUVA 20. Stored-proseduurin kutsuminen käyttöliittymäkoodissa

Yhteenveto

ASP.NET ei tarjoa täysin valmiita ratkaisuja injektioita vastaan, ja kokematon käyttäjä voikin tehdä vääriä ratkaisuja kyselyjen muodostamisessa. Whitelistien käyttö on ensisijaisen tärkeää, mutta se voi myös olla haastavaa syötteissä joissa pyydetään merkkijonoja, koska pitää katsoa aina tapauskohtaisesti, mitä merkkejä syötteessä sallitaan.

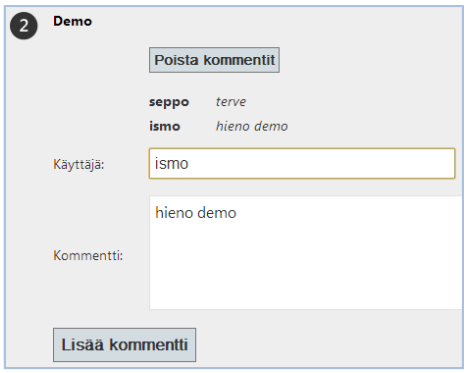
6.2 Sisällön lisääminen

Tapahtumaskenaario

Sovelluksessa on käyttäjän syötettä pyytävä lomake jolla on mahdollista lisätä kommentteja blogikirjoitukseen. Kirjoitukset kirjoitetaan kantaan josta ne myöhemmin saadaan ladattua blogikirjoituksen kommenttiosioon.

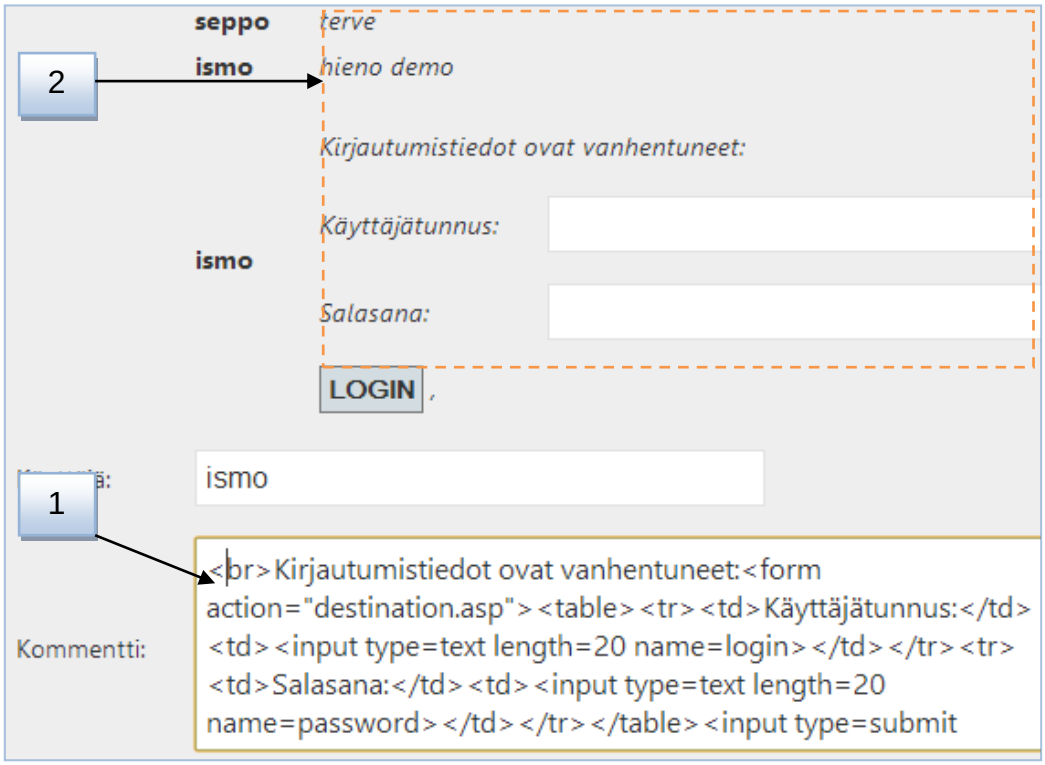
Riskitekijät

XSS:ssä yhteistä SQL-injektion kanssa on että molemmissa pyritään saamaan haitallista merkkijonosyntaksia järjestelmään. Tässäkin vaarana on että syötteet jäävät tarkastamatta haitallisten komentojen varalta. Kuvassa 21 on normaali kommenttilisäystoiminnallisuus joka avaa mahdollisuuden XSS:lle.



KUVA 21. Lomake jolla voi lisätä kommentteja sivustolle, nämä näkyvät kaikille sivulla vierailleville

Käyttäjän on mahdollista syöttää haitallista kuvauskieltä tai omia skriptejä syötteisiin kuvan 22 tapaan.



KUVA 22. Tunkeutujan kommentti-syötteeseen kirjoittama haitallinen HTML-syntaksi (1) siirtyi palvelimelle ja sitä myötä jokaisen sivulatauksen tekijän selaimeen (2).

Havainnot

ASP.NETissä on valmis toteutus potentiaalisten XSS-yritysten kitkemiseen, ValidateRequest-niminen property (kuva 23), joka löytyy automaattisesti jokaisesta Page-luokasta perityltä formilta. Oletuksena se on päällä, mutta ei ole poikkeuksellista että se kytketään pois käytöstä, jotta voidaan helpottaa jotain client-puolen toiminnallisuuden lisäämistä.

```
CodeBehind="Login.aspx.vb" Inherits="OpariSovellus.Login" ValidateRequest="true" %>
```

KUVA 23. Kun ValidateRequest-property on päällä, ASP.NET-parseri käy läpi sivupyynnöt haitallisten merkkien varalta, eikä päästä niitä pääsyä tietokantaan.

Tämän lisäksi syötteitä tulisi verrata Whitelistiin SQL-injektion tapaan, jolloin voidaan varmistua ettei palvelimelle asti pääse haitallista syntaksia.

Yhteenveto

Testiympäristössä haitallisten kyselyjen suorittaminen oli mahdotonta jos pyyntöjen validointi oli päällä.Tämä yhdessä whitelistin kanssa poisti kokonaan XSS-haavoittuvuuden mahdollisuuden.

6.3 Istunto

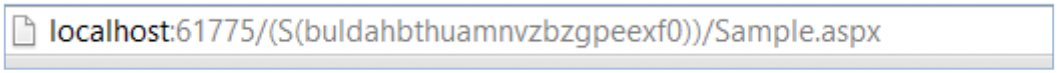
Istunnon käsittelyyn liittyvää testausta on melko haastava raportoida koska se ei ole täysin yksiselitteinen. Kaikille web-sovelluksille kuitenkin yhteistä on HTTP:n tilaton luonne ja sen aiheuttamat haasteet istunnonhallinnassa ja säilytyksessä.

Tapahtumaskenaario

Käyttäjä syöttää kirjautumistietonsa järjestelmään. Järjestelmä vertaa annettuja tietoja tietokantaan ja vastaavuuden löytyessä voidaan luoda istunto autentikoituneelle käyttäjälle ja suorittaa sisäänkirjautuminen.

Riskitekijä

Järjestelmän asetusten vuoksi, käyttäjän yksilöllinen istunto-ID voidaan säilöä selaimen osoiteriville. Käyttäjä linkkaa ohjelmistossa sivun ystävälleen tietämättä että hänen nykyinen istuntonsa kulkee mukana. Näin ollen osoitteen saanut käyttäjä saa lähettäjän istunnon käyttöönsä (kuva 24).



KUVA 24. Järjestelmässä säilytetään yksilöllistä SessionID:tä url:ssa.

Havainnot

ASP.NETissä on istunnonhallintaa varten valmis toteutus, jota kutsutaan Forms Authenticationiksi. Jokaiselle vierailijalle sivustolla luodaan oma istunto ja sille yksilöllinen ID. Tämä ID kulkee selaimen ja palvelimen välityksellä http-pyyntöjen välillä. ASP.NETissä kaikki evästeet on merkattu HttpOnly:ksi joka estää niiden muuntelun client-puolella ja näin ollen lisää tietoturvaa merkittävästi.

Mikäli jostain syystä järjestelmässä ei haluta käyttää evästeitä, joudutaan istunnon yksilöivä ID säilömään selaimen osoiteriville. ID:n kuljettaminen osoiterivillä mahdollistaa istunnon jakamisen toisen käyttäjän kanssa, joko tahallisesti tai tahattomasti.

```
<sessionState mode="InProc" customProvider="DefaultSessionProvider" cookieless="false">
```

KUVA 25. Tarvittaessa evästeet voi kytkeä pois asettamalla Web.Config:sta cookieless-propertyn arvoksi "true"

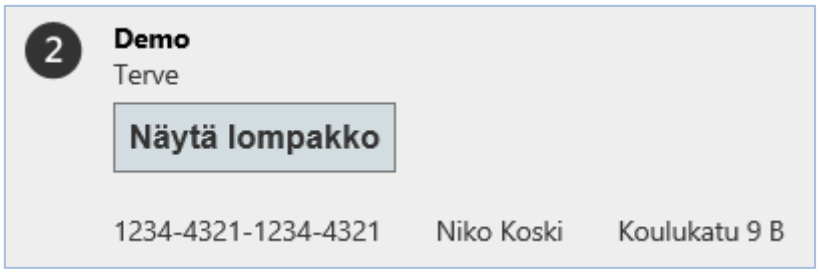
Yhteenveto

ASP.NETin Forms Authentication hoitaa istunnonhallinnan tehokkaasti ilman suurempaa arkkitehtuurisuunnittelua ohjelmistokehittäjältä. Edellytyksenä on että evästeet ovat käytössä sovelluksessa. Kehittäjän vastuulle jää lähinnä oikeuksien tarkistelu ja vertaaminen kannasta löytyvää tietoa vasten.

6.4 Luottamuksellisen tiedon näyttäminen

Tapahtumaskenaario

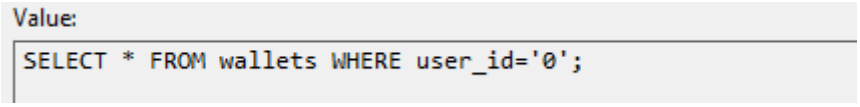
Sovelluksessa voidaan suorittaa ostoksia ja näin ollen siinä on mahdollisuus taltioida käyttäjän luottokortin tiedot. Käyttöliittymässä on painike jolla voidaan näyttää oman lompakon tiedot (kuva 26).



KUVA 26. Napin painaminen lataa tämänhetkisen käyttäjän lompakon sisällön.

Riskitekijä

Näytettävä lompakko kuuluu jollekin käyttäjälle ja tällä käyttäjällä tulisi olla tietokannassa yksilöllinen ID. Painike laukaisee kyselyn joka hakee kirjautuneen käyttäjän lompakon tiedot. Parametrina toimitetaan kirjautuneen käyttäjän ID (kuva 27).



KUVA 27. Kyselyn parametriksi saadaan tällä hetkellä kirjautuneen käyttäjän ID.

Esimerkki uhkatilanteesta

Jos kirjautunut käyttäjä pystyisi tavalla tai toisella muokkaamaan parametrina kulkevaa käyttäjä-ID:tä, voisi hän saada esille jonkun toisen käyttäjän lompakon sisällön sovelluksessa.

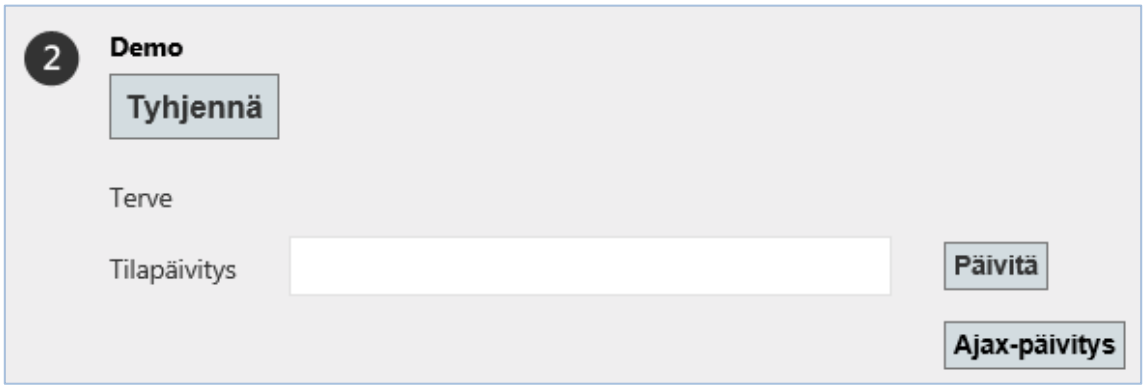
Yhteenveto

ASP.NET ei tarjoa suoraa suojaa tätä uhkaa vastaan, vaan on ohjelmistokehittäjän harteilla varmentaa, että istunnonhaltija ei pääse väärään dataan käsiksi. Väärän datan näyttämisen vaaraa ei synny, mikäli käyttöliittymässä ei anneta mahdollisuutta vaikuttaa parametreihin. Mikäli on tilanne, jossa vaaditaan parametri käyttöliittymän puolelta, tulee ohjelmistokehittäjän tehdä tarkistukset, että kyseisellä istunnonhaltijalla on oikeus pyydettyyn dataan.

6.5 Asynkroninen päivitys

Tapahtumaskenaario

Sovelluksessa on toiminto joka mahdollistaa tilapäivityksen tekemisen Facebookin tapaan. Kuvassa 28 oleva päivityspainike laukaisee SQL-kyselyn joka lisää tilapäivityksen tietokantaan.



KUVA 28. Asynkronisen tilapäivityksen mahdollistava syötelomake

Riskitekijä

Käyttöliittymäkokemuksen parantamiseksi kuvan 22 mukainen tilapäivitystoiminto voidaan suorittaa asynkronisesti. Asynkronisten funktioiden käyttäminen edellyttää että funktiot ovat saatavilla julkisen rajapinnan kautta. ASP.NETissä tällaisen voi toteuttaa Web Service -tekniikalla.

```
<OperationContract()>
Public Sub UpdateStatus(statusText As String)
    If HttpContext.Current.User.Identity.IsAuthenticated Then
        If statusText <> "" Then
            Dim db As New SQLiteDatabase
            Dim data As New Dictionary(Of String, String)
            data.Add("text", statusText)
            data.Add("user_id", 1)
            db.Insert("updates", data)
        End If
```

KUVA 29. Jos järjestelmässä tarkistetaan vain, onko käyttäjä kirjautunut, avaa se mahdollisuuden CSRF:lle, koska rajapinta ei nykyisellään ota kantaa siihen, mistä pyyntö on tullut.

Havainnot

Julkisten rajapintojen kutsujen yhteydessä tulee käyttää jotain satunnaisesti luotua parametria.

Yhteenveto

Web Servicen käyttö ASP.NET-sovelluksissa voi altistaa sovelluksen CSRF-hyökkäyksille.

7 YHTEENVETO

Tehtävänä oli tutkia ja tehdä selvitys yleisimmistä tietoturvauhista web-ohjelmistokehityksessä. Pohjana tutkimukselle käytettiin OWASP:n TOP-10 -dokumentaatiota, koska siitä on tullut eräänlainen de facto -standardi web-ohjelmistokehityspiireissä. Sen tuntevat useimmiten niin sovellusten tilaajat kuin ohjelmistokehittäjät.

Tässä työssä tehdyt testaukset antoivat tärkeää pohjatietoa, jota tekijä hyödynsi tilaajan tuotannossa olevan sovelluksen testaukseen. Näitä tuloksia ei tässä julkisessa dokumentaatiossa nähdä, mutta demosovelluksen tulokset antavat myös realistisen kuvan Web Forms -sovelluksen tietoturvasta.

Tekijän oma tietämys web-sovelluskehityksestä ja siihen liittyvistä tekniikoista kasvoi merkittävästi tätä työtä tehdessä. Etenkin sovelluksen tietoturvaan liittyvissä tekijöissä tuli paljon uutta eteen. Työtä tehdessä tuli tutuksi useita tietoturvaan liittyviä käsitteitä ja avainsanoja.

Vaikka työ nojautuikin OWASP:n määrittelemiin Top 10 -tietoturvauhkiin, varsinaista sovelluksen tietoturvaa ei tulisi rakentaa siten, että se toteuttaa vain ja juurikin nämä kymmenen kohtaa. On olemassa paljon muitakin uhkatekijöitä sovelluksen tietoturvan saralla, mutta TOP-10 on hyvä lähtökohta, kun halutaan tehdä turvallinen ohjelmisto.

LÄHTEET

1. Sovelluskehityksen tietoturvaohje, VAHTI 1/2013. Saatavissa: http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtionhallinnon_tietoturvallisuus/20130207Sovell/VAHTI_1_Sovelluskehityksen_tietoturvaohje_NETTI.pdf. Hakupäivä 8.2.2013.

2. Hypertext Transfer Protocol. 1999. Saatavissa: http://www.w3.org/Protocols/rfc2616/rfc2616.html. Hakupäivä 25.1.2013.

3. Web Applications: What are They? Saatavissa: http://www.acunetix.com/websitesecurity/web-applications/. Hakupäivä 22.1.2013.

4. Guide To Building Secure Web Applications and Web Services. 2005. Saatavissa: http://freefr.dl.sourceforge.net/project/owasp/Guide/2.0.1/OWASPGuide2.0.1.pdf. Hakupäivä 6.12.2012.

5. Mitchell, Scott 2008. An Overview of Forms Authentication. Saatavissa: http://www.asp.net/web-forms/tutorials/security/introduction/an-overview-of-forms-authentication-vb. Hakupäivä: 5.3.2013.

6. Mitchell, Scott 2008. Security Basics and ASP.NET Support. Saatavissa: http://www.asp.net/web-forms/tutorials/security/introduction/security-basics-and-asp-net-support-cs. Hakupäivä: 17.4.2013.

7. Hunt, Troy 2011. OWASP Top 10 for .NET developers. Saatavissa: https://asafaweb.com/OWASP%20Top%2010%20for%20.NET%20developers.pdf. Hakupäivä 1.4.2013.

8. About The Open Web Application Security Project. 2013. Saatavissa: https://www.owasp.org/index.php/About_OWASP. Hakupäivä: 20.4.2013

9. OWASP Top 10 2010. 2010. Saatavissa: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf. Hakupäivä 5.12.2012.

10. Prosise, Jeff 2002. ASP.NET Security: An Introductory Guide to Building and Deploying More Secure Sites with ASP.NET and IIS. Saatavissa: http://msdn.microsoft.com/en-us/magazine/cc301387.aspx. Hakupäivä 30.3.2013.

11. Singh, Rahul 2012. Understanding Session Management Techniques in ASP.Saatavissa: http://www.codeproject.com/Articles/416137/Understanding-Session-Management-Techniques-in-ASP. Hakupäivä: 10.3.2013.

12. Ferguson, Dave 2009. OWASP Top Ten Changing for 2010. Saatavissa: http://appsecnotes.blogspot.fi/2009/11/owasp-top-ten-changing-for-2010.html. Hakupäivä: 4.5.2013.