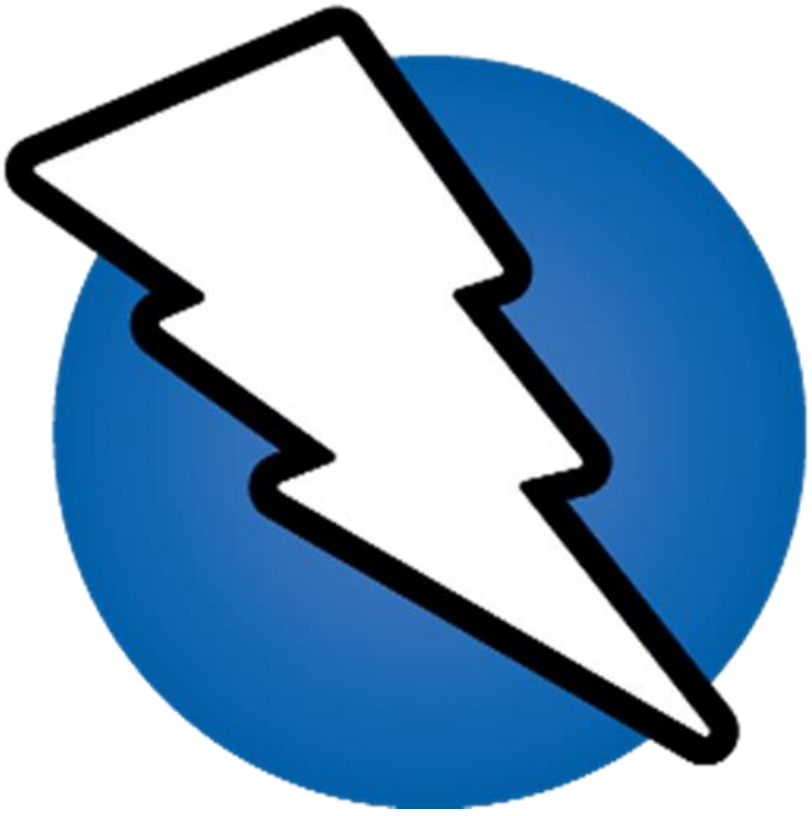
Saastamoinen Miika

# OWASP ZAP -tietoturvatestaustyökalun käyttö websovelluskehityksen tukena



Insinööri

Tieto- ja viestintätekniikka

Syksy 2018

KAMK • University of Applied Sciences

## Tiivistelmä

**Tekijä**: Saastamoinen Miika

**Työn nimi:** OWASP ZAP -tietoturvatestaustyökalun käyttö websovelluskehityksen tukena

**Tutkintonimike:** Insinööri (AMK), tieto- ja viestintätekniikka

**Asiasanat:** OWASP, OWASP ZAP, web, tietoturva

Tietosuoja ja -turva on aina ollut tärkeä asia ihmisten yksityisyydensuojan turvaamiseksi. Ennen Internetiä, ulkopuolisten ihmisten oli vaikea päästä toisten ihmisten tietoihin käsiksi, sillä tiedot olivat todennäköisesti lukkojen takana paperitulosteena. Kun palvelut siirtyvät huimaa vauhtia verkkoon, on kaikilla Internetin käyttäjillä potentiaalinen mahdollisuus päästä sellaiseen tietoon käsiksi, johon heillä ei ole oikeutta. Opinnäytetyön suurena tavoitteena on laajentaa lukijan käsitystä tietoturvan tärkeydestä.

Opinnäytetyössä tutkittiin, miten OWASP ZAP -tietoturvatestaustyökalu (myöhemmin työssä ZAP) toimii, ja miten sitä voidaan hyödyntää sovelluskehityksessä. Jotta ZAP:n käytölle olisi jokin syy, täytyi ensin tutkia mitä tietoturva on ja millaisia säädöksiä tietoturvalle on asetettu. Ennen itse ohjelman käyttöä opinnäytetyössä käydään läpi tietoturvan perusteita ja GDPR-lainsäädäntöä lyhyesti.

ZAP toimii kaikilla yleisimmillä käyttöjärjestelmillä, mutta tämä työ keskittyy sen käyttöön Linux Mint -käyttöjärjestelmällä. Työn lopputuloksena on valmis käyttöohje, jonka avulla luodaan turvallinen ympäristö ZAP:n käyttöön ja päästään alkuun tietoturvatestauksen saralla. Ohjeessa käydään läpi muutama yleinen hyökkäystapa, joita käytetään verkkosovelluksia vastaan. Niiden avulla lukija voi tehdä päätelmiä siitä, millaisia testejä voi tehdä omille sovelluksille haavoittuvuuksien löytämiseksi. Käyttöohje on suunnattu sellaisille sovelluskehittäjille ja -testaajille, joilla on kokemusta websovellusten parissa työskentelystä.

KAMK • University of Applied Sciences

## Abstract

**Author:** Saastamoinen Miika

**Title of the Publication:** Using OWASP ZAP security testing tool during web application development

**Degree Title:** Bachelor of Engineering, Information technology

**Keywords:** OWASP, OWASP ZAP, web, information security

Data protection and information security has always been important part of people's privacy. Before the Internet people couldn't access confidential data as easily because the information was kept in a printed form inside a locked physical location. Services are quickly moving to the Internet, which opens the possibility for every web user to potentially get access to the personal information people confidentially share with the service provider. Goal of this thesis was to expand the reader's viewpoint on information security.

This thesis examines how OWASP ZAP security testing tool (later referred as ZAP) works and how it could be used during development of a web application. To give a reason to use ZAP, first was necessary to investigate regulations set on data protection and what data protection is. First, before using ZAP this thesis introduces basics of data protection and explains the European Union's General Data Protection Regulation shortly.

ZAP works on all major operating systems, and this thesis concentrates on its usage on Linux Mint. The final thesis was a guide on how to create a safe environment to test and practice the usage of ZAP. This environment is a great place to start learning about security testing. The guide uses couple of well-known attack types against web applications as an example. The guide is targeted towards web developers and testers who are familiar with web applications.

Sisällys

**Symboliluettelo**

**GDPR**: General Data Protection Regulation. Euroopan unionin asettama yleinen tietoturva-asetus.

**HTTP**: Hypertext Transfer Protocol. Protokolla, jonka avulla verkkoselaimet keskustelevat verkkopalvelimien kanssa.

**OWASP ZAP**: OWASP Zed Attack Proxy. Myöhemmin tässä työssä pelkkä ZAP. Avoimen lähdekoodin tietoturvaskanneri. Käytetään verkkosovelluksen ja verkkopalvelimien välisen http-viestinnän tutkimiseen.

**OWASP**: Open Web Application Security Project. Internet-yhteisö, joka tuottaa ilmaiseksi saataville artikkeleita, metodeja, dokumentaatiota, työkaluja ja teknologioita verkkosovellusten tietoturvasta.

**SQL**: Structured Query Language. Standardoitu kyselykieli, jonka avulla voidaan hakea tietoa riippuvuustietokannoista.

**Välityspalvelin/Proxy**: Verkkosovelluksen ja palvelimen välissä oleva palvelin, joka tutkii tai käsittelee niiden välillä kulkevaa viestintää.

**Virtuaalitietokone**: Tietokoneympäristö, jota voidaan ajaa käyttöjärjestelmän sisällä.

# 1 Johdanto

Ohjelmistot siirtyvät jatkuvasti perinteisiltä käyttöjärjestelmäalustoilta pilveen verkkopohjaisiksi sovelluksiksi. Vuonna 2018 Iso-Britannian väestön aikuisista 78 % käyttää Internetiä älypuhelimellaan [1]. Vuonna 2017 suomalaisten Android-puhelinten kymmentä asennetuinta sovellusta ei voi edes käyttää ilman Internetyhteyttä [2]. Tästä johtuen yhä suurempi käyttäjäjoukko on sidoksissa Internetiin ja miltei kaikki verkkosivut ja -sovellukset keräävät jotain tietoa käyttäjistään. Tämän trendin on huomannut myös Euroopan unioni, joka asetti 25.05.2018 täytäntöön yleisen tietosuoja-asetuksen, GDPR:n (General Data Protection Requlation). GDPR-lainsäädännöllä on tarkoitus taata EU-kansalaisille parempaa verkkoyksityisyydensuojaa.

Jotta verkkosovellukset olisivat tietoturvan kannalta vankkoja, täytyy sovellusten ohjelmoijien olla ajan tasalla siitä, miten hyökkäyksiltä pystytään suojautumaan ja miten hyökkääjät toimivat. Tietoturva on monisäikeinen asia, johon vaikuttaa lukematon määrä eri tekijöitä. Kuitenkin viimekädessä Internetissä toimivien palveluiden henkilötiedoista ovat vastuussa henkilöt, jotka kirjoittavat näitä tietoja käsittelevät sovellukset. Lainsäädännöt ja asetukset ovat mitättömiä, jos kanssaihmisille Internetin palveluita tuottavilla ihmisillä ei ole tarpeeksi tietotaitoa. Ylläpitäjät ja ohjelmoijat ovat ihmisten verkkoyksityisyyden suojakuori. He ovat niitä, jotka määräävät miten heille luovutettu tieto säilytetään tietojärjestelmissä, ja miten se pidetään sellaisten ihmisten ulottumattomissa, joilla ei ole siihen tietoon mitään oikeutta. Yksi tietoturvan peruspilareista on säilyttää data vain asianmukaisten osapuolten saatavilla.

# 2 Vastuuvapauslauseke

Tässä opinnäytetyössä laadittu materiaali on tehty informatiiviseen tarkoitukseen. OWASP ZAP on hyvin tehokas työkalu verkkoliikenteen seuraamiseen ja verkkosivustojen sisällön tutkimiseen, ja sen käytön tulee aina olla luvanvaraista. ZAP:n käyttö julkisverkossa oleviin verkkosivuihin ja -sovelluksiin on ehdottomasti kielletty. Luvaton verkkosivujen systemaattinen tutkiminen voidaan tulkita tietomurron yritykseksi. Rikoslain mukaan tietomurto sekä tietomurron yritys ovat rangaistavia tekoja. [3]

Ote rikoslaista:

*" **8 §** **(10.4.2015/368)***

***Tietomurto***

*Joka käyttämällä hänelle kuulumatonta käyttäjätunnusta taikka turvajärjestelyn muuten murtamalla oikeudettomasti tunkeutuu tietojärjestelmään, jossa sähköisesti tai muulla vastaavalla teknisellä keinolla käsitellään, varastoidaan tai siirretään tietoja tai dataa, taikka sellaisen järjestelmän erikseen suojattuun osaan, on tuomittava tietomurrosta sakkoon tai vankeuteen enintään kahdeksi vuodeksi.*

*Tietomurrosta tuomitaan myös se, joka tietojärjestelmään tai sen osaan tunkeutumatta*

*1) teknisen erikoislaitteen avulla tai*

*2) muuten teknisin keinoin turvajärjestelyn ohittaen, tietojärjestelmän haavoittuvuutta hyväksi käyttäen tai muuten ilmeisen vilpillisin keinoin*

*oikeudettomasti ottaa selon 1 momentissa tarkoitetussa tietojärjestelmässä olevasta tiedosta tai datasta.*

*Yritys on rangaistava.*

*Tätä pykälää sovelletaan ainoastaan tekoon, josta ei ole muualla laissa säädetty ankarampaa tai yhtä ankaraa rangaistusta."* [3]

# 3 Mitä tietoturva on?

Tietoturva koostuu monesta eri tekijästä, ja tässä luvussa kuvataan millaisia tietoturvakysymyksiä verkkosovellusten ohjelmoijat voivat kohdata. Tietoturvatoimia ohjeistavat ainakin lainsäädännöt, businessajattelu sekä yleisesti hyväksytyt hyvät käytänteet. Tietoturva-asioihin kantaa ottavat kansalliset lait, kuten Sähköisen viestinnän tietosuojalaki sekä EU-tasolla GDPR. Verkkosovellusten kannattavuus perustuu usein käyttäjämäärään. On yrityksen edun mukaista, että käyttäjät kokevat yrityksen tarjoaman alustan turvalliseksi ja että he voivat turvallisin mielin jakaa henkilötietojansa yrityksen käyttöön. Käyttäjämäärien kasvaessa myös sijoittajien kiinnostus yrityksiin nousee. Comparitechin tutkimuksen mukaan yrityksen joutuessa tietomurron kohteeksi, ja riippuen vuodettujen tietojen arkaluontoisuudesta, yrityksen kurssi useimmiten kokee lyhyellä aikavälillä notkahduksen. Tietomurrosta toipumiseen voi mennä huomattavan pitkä aika. [4]

## 3.1 GDPR lyhyesti

GDPR:n tavoitteena on antaa EU-kansalaisille suurempi valta hallinnoida heidän henkilökohtaista dataansa esimerkiksi Internetissä. Vain 15 % ihmisistä uskoo, että heillä on täysi kontrolli omista tiedoistaan Internetin palveluissa. Henkilökohtaiseksi dataksi luokitellaan muun muassa nimi, osoite, sijainti, onlinetunnus, terveystiedot, tulotaso sekä kulttuuri. Ennen näiden tietojen keräämistä yritysten täytyy kertoa, mikä taho tietoja kerää, miksi niitä kerätään, kuinka kauan tietoja säilytetään ja kuka vastaanottaa tiedon. Palvelun ylläpitäjä saa kerätä näitä tietoja vain, jos henkilö antaa siihen luvan. Henkilöllä on myös oikeus päästä tietoihinsa käsiksi. Mikäli tapahtuu tietomurto ja henkilötietoja vuotaa, on näistä tietomurroista ilmoitettava kaikille asianosaisille. Jos henkilötietoja pyydetään poistamaan, ne täytyy poistaa, koska kansalaisilla on oikeus tulla niin sanotusti ”unohdetuksi”. [5] [6]

Jos esimerkiksi lainahakemuksen käsittelyprosessissa käytetään koneellista profilointia, tulee ihmisen tarkistaa se, mikäli se päätyy evätyksi. Tästä profiloinnista tulee myös ilmoittaa hakijalle etukäteen sekä antaa mahdollisuus valittaa päätöksestä. Henkilöille täytyy antaa mahdollisuus jättäytyä pois sellaisesta suoramarkkinoinnista, joka hyödyntää heidän tietojaan. Yritysten pitää olla erityisen tarkkaavaisia niistä tiedoista, jotka liittyvät terveydentilaan, rotuun, seksuaaliseen suuntautumiseen tai poliittisiin näkemyksiin. Riippuen jäsenmaiden laeista, 13–16-vuotiaiden lasten tietojen tallentamiseen tarvitaan huoltajan lupa. Mikäli yritys ei kykene täyttämään GDPR:n

mukaisia ohjeistuksia, se voi saada sakon, jonka suuruus on joko 20 miljoonaa euroa tai 4 % yhtiön vuosittaisesta liikevaihdosta, kumpi näistä luvuista osoittautuukin suuremmaksi. [5] [6]

## 3.2 Tietoturvan peruspilarit

Tietoturvan perusteet pystytään pilkkomaan helposti kolmeen peruspilariin, joihin kaikki tietoturvan osa-alueet loppujen lopuksi nojaavat. Nämä kolme pilaria ovat luottamuksellisuus, eheys ja saatavuus.

**Luottamuksellisuus** tarkoittaa sitä, että tieto on saatavilla vain niille henkilöille, joilla on siihen oikeus. Esimerkkinä verkkosivun käyttäjätunnukset. Tilin luonut henkilö tietää omat tunnuksensa, mutta ongelmatilanteissa voi tukeutua palveluntarjoajan tekniseen tukeen, jolla on myös tiedot käyttäjästä. Jos tähän verkkopalveluun tehdään tietomurto ja kaikki sivuston luottamukselliset tiedot vuodetaan julkisuuteen, informaation luotettavuus on kadonnut. Sitä ei voida enää poistaa niiden ulkopuolisten henkilöiden tietoisuudesta, jotka ovat siihen päässeet käsiksi. [7]

**Eheys** tarkoittaa sitä, että pyydetty data on oikeellista eikä sitä ei ole muokattu. Esimerkiksi sähköpostipalvelun käyttäjän täytyy pystyä luottamaan, että vastaanotettu sähköposti tulee siitä osoitteesta, mistä se on lähetetty eikä sitä ole matkan varrella muutettu ilman lupaa. Esimerkkitilanne: Työntekijä avaa toimitusjohtajalta saapuneen sähköpostin, joka sisältää uutisia henkilöstön irtisanomisista. Toimitusjohtajan alkuperäisessä viestissä onniteltiin työntekijöitä saavutetuista tavoitteista. Näin ollen jokin ulkopuolinen taho on päässyt käsiksi työntekijän ja toimitusjohtajan sähköpostiviestinnän väliin, ja onnistuu aiheuttamaan sekaannusta. [7]

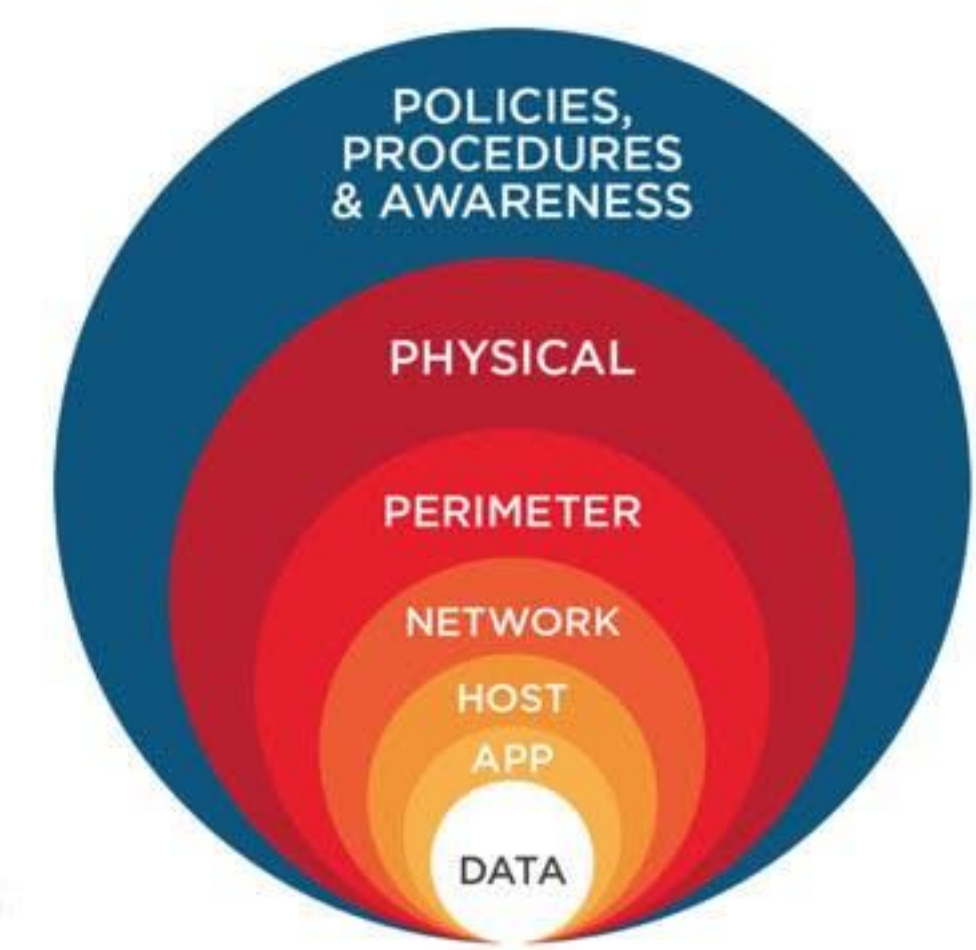
**Saatavuus**. Käyttäjän tallentama tai hänelle kuuluva tieto tulee olla hänen saatavillaan. Esimerkiksi kun verkkopankkisovelluksen käyttäjä haluaa tarkistaa pankkitilinsä saldon, on varmistuttava siitä, että käyttäjälle annetaan hänen tilitietonsa. Tai jos käyttäjällä on esimerkiksi tilaus pilvipalveluun, joka tarjoaa kovalevytilaa varmuuskopiointia varten, käyttäjän on päästävä näihin kopioihin käsiksi. Kopiot on saatava siinä muodossa, kun ne on tallennettu, jos alkuperäiset versiot ovat esimerkiksi päässeet tuhoutumaan. [7]

### 3.3 Mistä tietoturva koostuu?

Tietoturva on hyvin laaja ala, joka koostuu useista eri tekijöistä. Informaatioteknologian lisäksi tietoturvaa on myös Internetin ja verkon ulkopuolella. Myös arkistoidut fyysiset asiakirjat vaativat turvallista säilytystä, ja tällöin sen turvaamiseksi on luonnollisesti käytettävä aivan eri menetelmiä kuin verkkopalvelimella sijaitseviin käyttäjätietoihin. Julkisverkossa olevalla palvelimella oleva tieto on todennäköisesti suuremmassa vaarassa, koska siihen osuva hyökkäysvektori internetin globaalin luonteen vuoksi on huomattavasti suurempi.

On otettava huomioon, mistä verkosta palvelimia pääsee hallitsemaan ja voiko niihin ottaa etäyhteyden. Pidetäänkö palvelimet oman organisaation sisällä ja tehdään oma palvelininfrastruktuuri, vai luotetaanko ulkopuoliseen palveluntarjoajaan? On mietittävä, kuka saa mennä palvelinhuoneeseen ja miten ulkopuolisten fyysinen tunkeutuminen palvelinkeskukseen on estetty. Millainen pääsy sovelluksella on tähän dataan, joka palvelimella sijaitsee? Pystyykö sovelluksesta tunkeutumaan sisäverkkoon? Kuinka organisaatioiden työkoneet valitaan ja suojataan? Nämä ovat vain murto-osa niistä asioista, joita nykypäivän tietoyhteiskunnassa on otettava huomioon, kun yhä enenevissä määrin kuluttajat ja käyttäjät luottavat omaa henkilökohtaista tietoaan eri palveluihin. Tietoturvaa ei voi rakentaa vain yhden osa-alueen varaan, koska hyökkäys voi tulla miltei mistä tahansa; jos esimerkiksi lääkärin vastaanottohuoneen siivooja näkee lukitsemattomalta näytöltä luottamuksellisia potilaskertomuksia, sen voi tulkita tietovuodoksi.

Mikäli linnakkeen ainoana suojausmekanismina on vain jykevä ulkomuuri, sortuessaan se jättää kaiken sisällään haavoittuvaiseksi. Tämän vuoksi tietoturva koostuu aina kerroksista. Yhden kerroksen murtuessa joutuu hyökkääjä menemään toisen puolustusmekanismin läpi. Tämä hidastaa hyökkääjän toimintaa ja antaa puolustajille enemmän aikaa reagoida. Kuvasta 1 nähdään, että turvattava data on käärittyeri kerroksiin.




Kuva 1. Tietoturvan rakentuminen kerroksissa [8]

Uloimpana kuvassa 1 (policies, procedures & awareness) ovat ihmiset ja heidän käytänteensä; kuinka tietoisia henkilöstö on erilaisista uhista? Miten henkilöstö toimii, kun ulkopuolinen henkilö kysyy organisaatioon liittyvää tietoa? Seuraavana kerroksena on fyysinen (physical) turvallisuus, miten rajoitettu pääsy ulkopuolisilta on toimistotiloihin tai palvelinsaleihin? Kuinka henkilöstön työvälineet on suojattu ja missä niitä säilytetään? Kuinka avainhenkilöiden turvallisuus hoidetaan?

Verkon ulkoreuna (perimeter) on piste, jossa organisaation verkko kohtaa julkisen verkon. Ulkoreunaa suojaavat muiden muassa palomuurit, virustorjunta ja virtuaaliverkot. Sisäverkon (network) suojauksessa on otettava huomioon kuinka laajalle alalle mahdollinen hyökkääjä voi päästä. Miten verkossa olevien palvelinten, työasemien, kytkinten, reitittimien ja muiden laitteiden asetukset on säädetty (host)? [8]

Viimeisenä kerroksena ennen datakerrosta on se ohjelmisto, joka pääsee tähän dataan käsiksi ja jota verkon käyttäjät käyttävät. Kehittäjä tai kehitystiimi on täten käytännössä suoraan vastuussa

siitä, miten datakerroksen data suojataan. Ohjelmiston lähdekoodi täytyy analysoida uhkien varalta (app). Ohjelmalle ja koodille on tehtävä kattavaa testausta sekä penetraatiotestausta. Käyttäjäsyöte on käsiteltävä siten, ettei ohjelma kaadu odottamattomalla tavalla saadessaan merkkejä, joita se ei osaa hallita. Nämä ovat vain muutamia esimerkkejä tietoturvakontrolleista, jotka pitää ottaa sovelluskehityksessä huomioon. Harvoin kuitenkaan vain sovelluksella on pääsy tähän dataan. Esimerkiksi ylläpito- ja asiakaspalveluhenkilöillä voi myös olla pääsy dataan. Tällöin organisaatiolla täytyy olla kontrolli siihen, mitä dataan käsiksi pääsevä henkilö voi nähdä, ja mitä sillä datalla saa tehdä. Myös sen varalta, että sovelluskerros romahtaa ja data vuotaa ulkopuolelle, datan tulisi olla salattua. Tällöin hakkerit eivät pääse lukemaan dataa ilman salausavainta tai käyttämättä huomattavaa aikaa datan salauksen purkamiseen (data). [8]

Kuten näkyy, ohjelmoijilla on suuri vastuu siitä, miten käyttäjädataa hallitaan, sillä heidän hallinnoimansa kerros on lähimpänä itse dataa. Mutta inhimillisiä virheitä tapahtuu, eikä jokaista pienintä reikää voida havainnoida, ohjelmakerros ei voi olla ainoa tapa suojata dataa. Hyökkääjien taidot kehittyvät, prosessointinopeus kehittyy, ohjelma-alustat vanhenevat, ja niistä löytyy haavoittuvuuksia. Uhkia on lukematon määrä eikä kaikkea voi ottaa huomioon. Kun muut kerrokset ovat vankkoina tukemassa ohjelmaa, voivat kehittäjät keskittyä tietoturvan lisäksi myös tuottamaan sellaista ohjelmakoodia, joka tuottaa hyötyä kanssaihmisille.

# 4 OWASP ja OWASP ZAP

Luvussa esitellään, mitä OWASP ja OWASP ZAP ovat. Katsotaan, millaisia sisältöjä OWASP tarjoaa ja millainen on ZAP:n toimintaperiaate. Kohdasta 4.3 eteenpäin on käytännön ohjeita, joita seuraamalla luodaan turvallinen testausympäristö ZAP:n käytön harjoittelua varten. Ohjeen aikana asennetaan ZAP, virtuaalitietokoneiden hallintaohjelma VirtualBox, harjoitteluympäristö OWASP Broken Web Applications Project sekä tietokantainjekoita hyväksikäyttävä sqlmap.

## 4.1 OWASP

OWASP eli Open Web Application Security Project on kansainvälinen voittoa tavoittelematon järjestö, jonka tavoitteena on tuoda sovelluskehityksen turvallisuusnäkökulmia näkyvimmiksi sekä kehittää sovellusten turvallisuutta. OWASP tarjoaa erilaisia verkkosovellusten turvallisuuteen liittyviä materiaaleja ilmaiseksi. Tällaisia materiaaleja ovat muun muassa OWASP Top Ten Project, OWASP Cheat Sheets ja OWASP Security Testing Guide. OWASP Top Ten Project -listaan on kerätty kymmenen kriittisintä tietoturvahaavoittuvuutta, joihin webkehittäjien kannattaa kiinnittää erityistä huomiota. Tässä opinnäytetyössä käydään läpi muutamia näistä haavoittuvuuksista. OWASP Cheat Sheets eli lunttauslistat ovat kokoelma eri tapoja testata ohjelmistoa tietoturvahaavoittuvuuksien löytämiseksi. Nämä lunttauslistat sisältävät muistutuslistoja niistä seikoista, joita tietyissä tilanteissa, esimerkiksi sisäänkirjautumisessa, kannattaa ottaa huomioon, ja miten näitä seikkoja voi testata. OWASP Testing Guide eli testausopas antaa viitekehyksen sille, mitkä ovat websovelluksen turvallisuustestauksen parhaat käytänteet. [9]

Näiden materiaalien lisäksi OWASP-yhteisö kehittää myös sovelluksia kuten OWASP Zed Attack Proxy (OWASP ZAP), jota käsitellään myöhemmin tässä työssä, sekä OWASP Dependency Check. OWASP Dependency Check eli riippuvuustarkastus sen sijaan on ohjelmisto, joka käy läpi ohjelman riippuvuuslistan tietoturvauhkien varalta. Riippuvuudella tarkoitetaan tässä kolmannen osapuolen julkisessa jaossa olevaa kirjastoa tai pakettia, jonka kehittäjä voi ladata osaksi omaa projektiaan tai ohjelmaa. Tällä hetkellä Dependency check tukee virallisesti .NET ja Java-sovelluksia [10].

Joihinkin riippuvuuden hallintatyökaluihin tällainen tietoturvatarkastus on sisäänrakennettu, kuten Javascript-pakettien hallintatyökalu NPM:iin (https://www.npmjs.com/). PHP-sovelluksissa

usein käytetty Composer (https://getcomposer.org/) ei suoraan tarjoa pakettien tietoturvatarkastusta, minkä vuoksi näissä tilanteissa tulee tukeutua Composer-yhteisön ylläpitämiin ratkaisuihin kuten esimerkiksi Symfony Security Monitoring -ohjelmaan. (https://github.com/sensiolabs/security-checker)

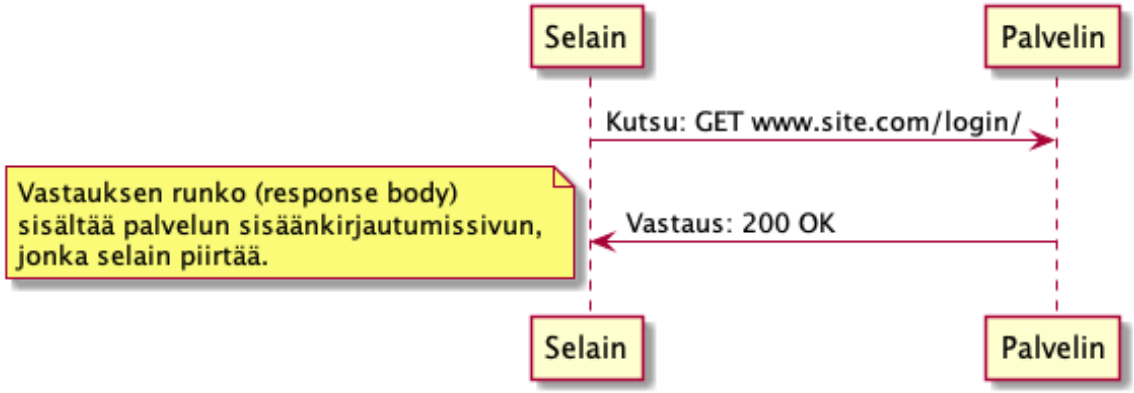
OWASP:n tarjoamat työkalut ja materiaalit ovat hyvä lähtökohta kaikille kehittäjille, jotka ovat kiinnostuneita tietoturvallisen ohjelmistokoodin tekemisestä. Jotta ohjelmista voidaan tehdä entistäkin kestävämpiä, tulisi jokaisen ohjelmoijan nähdä kuinka helposti tietoturvaton ohjelma voidaan murtaa ja miten vakavia tietyt haavoittuvuudet ovat, kuten esimerkiksi tietokantainjektio. Vähänkään kokeneempi hakkeri näkee jo verkkosivun ulkoasusta, mitä haavoittuvuuksia sivustossa mahdollisesti on. Koska Internet on globaali alusta, kokeneita hakkereita on loputon määrä ja heillä on kollektiivisesti loputon aika murtaa mikä tahansa tietojärjestelmä. Mikäli Internetissä haluaa tehdä liiketoimintaa, on huomioitava, että tässä informaatiosodassa yhtä kehittäjää vastaan on aina vähintään legioona ilkeämielisiä verkon käyttäjiä, jotka haluavat hyötyä muiden virheistä.

## 4.2 OWASP ZAP

OWASP ZAP eli OWASP Zed Attack Proxy on tietoturvan testausohjelma, jonka avulla voidaan tutkia selaimen ja palvelimen välillä kulkevaa http-liikennettä. ZAP on avoimen lähdekoodin ohjelmisto, ja se on ladattavissa kaikille yleisimmille käyttöjärjestelmille, eli Windowsille, Linuxille ja MacOS:lle (https://github.com/zaproxy/zaproxy/wiki/Downloads). Tämän työn esimerkeissä käytetään Linux-versiota, mutta todennäköisesti käytännön erot eri alustojen välillä ovat minimaaliset.

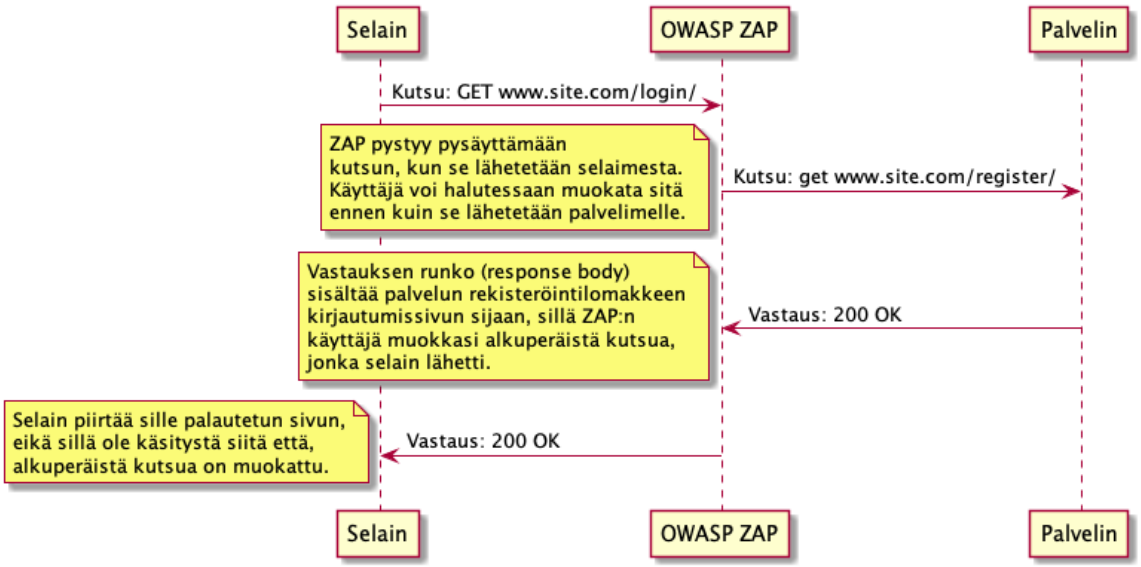
## 4.3 ZAP:n toimintaperiaate

Normaalissa tilanteessa selain välittää pyynnöt palvelimelle, joka tulkitsee pyynnön ja vastaa takaisin selaimelle jollakin ohjelmoidulla tavalla. Selaimen tehtävä on tulkita palvelimelta saatu data niin, että se on ihmiselle helposti luettavassa muodossa, joten se tulkitsee palvelimen välittämän datan, esimerkiksi HTML-sivun, ja piirtää sen visuaaliseen muotoon. Tämä tilanne on kuvattu kuvassa 2.




Kuva 2. Selaimen ja palvelimen välinen liikenne normaalitilanteessa

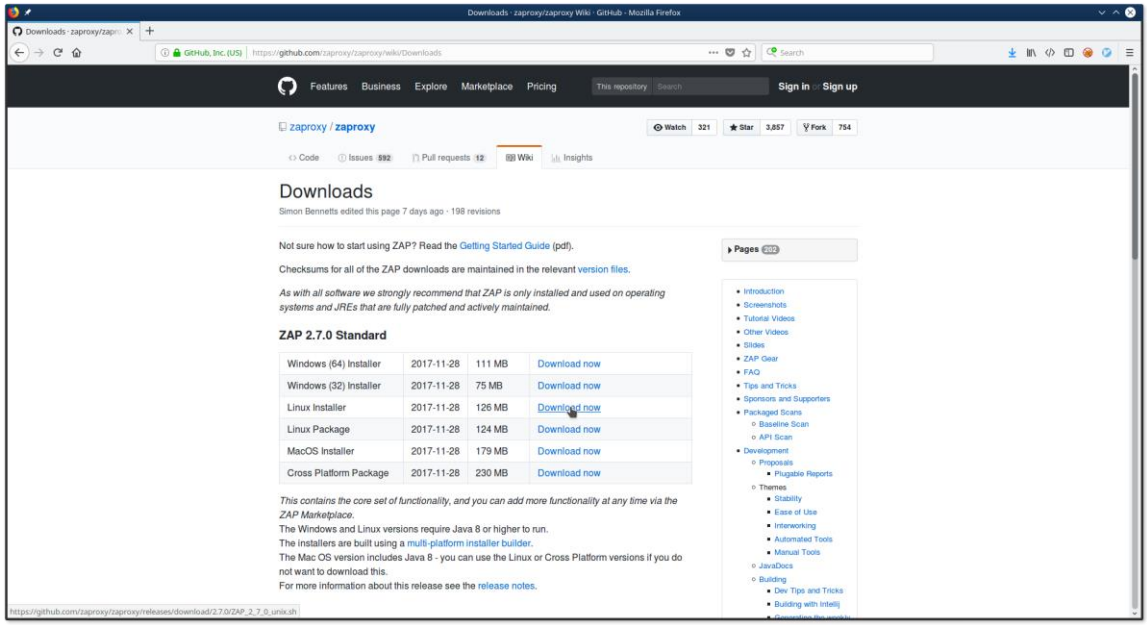
Jotta ZAP:ia voi hyödyntää http-viestinnän tutkimisessa, on selaimen asetuksia muutettava niin, että se lähettää kutsut ZAP:lle, joka taas välittää kutsun palvelimelle. Täten ZAP toimii välityspalvelimena eli proxyna. ZAP kerää kaiken liikenteen, mikä kulkee selaimen ja palvelinten välissä ja kirjaa ne muistiin. Näin ZAP:n käyttäjän on helppo tutkia liikennettä jälkeenpäin tai vaikka kutsua palvelinta uudestaan jollain vanhemmalla kutsulla. ZAP:n toiminta välityspalvelimena on kuvattu kuvassa 3.




Kuva 3. Selaimen ja palvelimen välinen liikenne, kun ZAP on kytketty välityspalvelimeksi

## 4.4 ZAP:n asentaminen

1. ZAP:n asennustiedosto löytyy verkkosivulta *https://github.com/zaproxy/zaproxy/wiki/Downloads.* Kuvassa 4 näkyy latauslinkki kursorin alla.




Kuva 4. ZAP:n lataussivu avoimen lähdekoodin jakopalvelu GitHub:ssa [11]

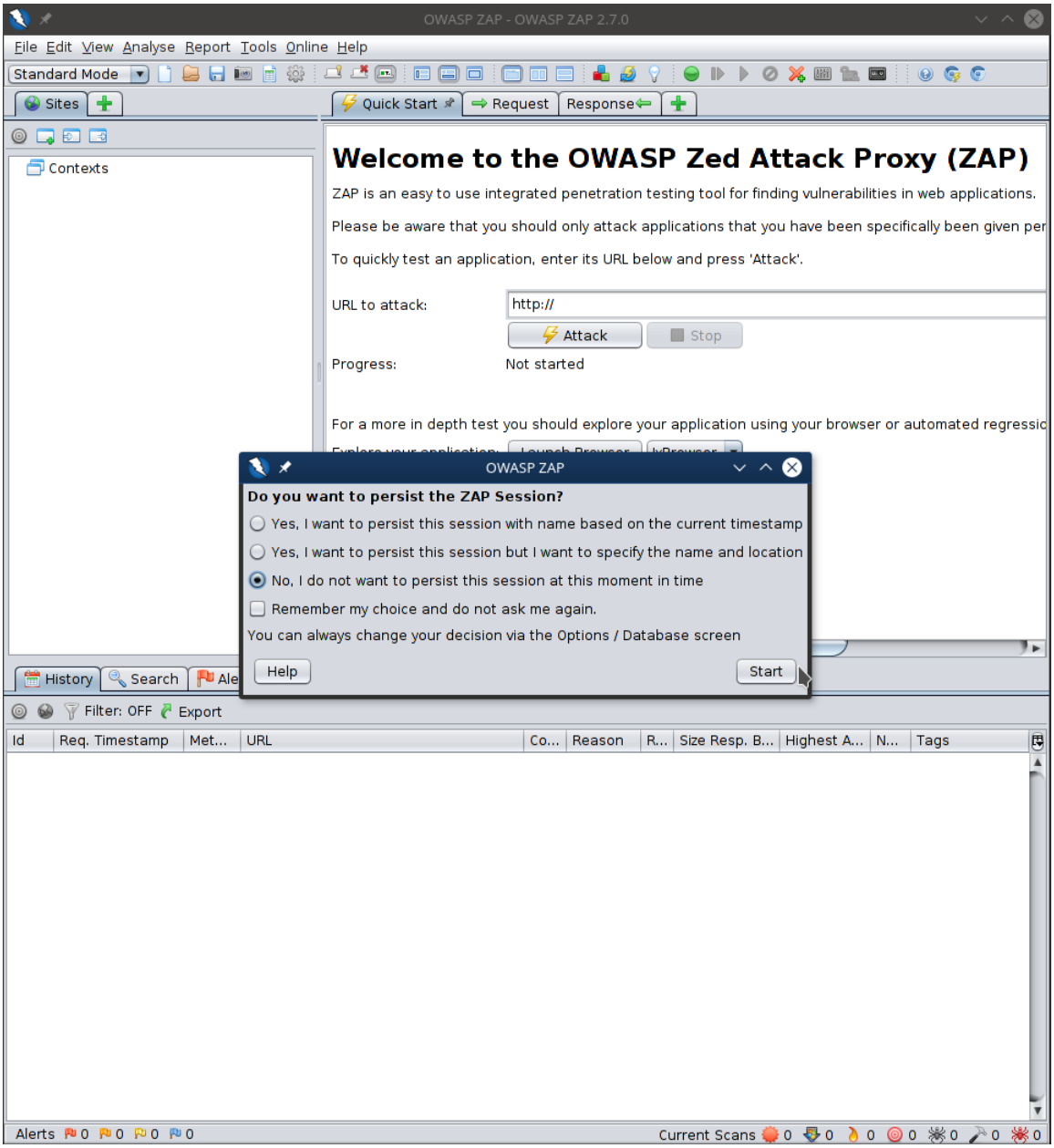
2. Kun asennusohjelma on latautunut, *Konsole*-pääteohjelmalla voidaan navigoida latauskansioon komennolla *cd ~/Downloads.*

3. Jotta asennusohjelma voidaan ajaa, pitää sille antaa käyttäjäoikeudet: *chmod +x ZAP_2_7_0_unix.sh* (ZAP_2_7_0_unix.sh on asennusohjelman nimi työn kirjoitushetkellä, versio tai nimi voi muuttua).

4. Asennusohjelma suoritetaan komennolla: *sudo ./ZAP_2_7_0_unix.sh.*

5. Kun OWASP ZAP on asennettu, sen voi käynnistää ohjelmavalikosta.

6. Kun ZAP käynnistyy, aukeaa valintaikkuna, josta valitaan "*No, I do not want to persist this session at this moment in time*" ja painetaan "*Start*".




Kuva 5. ZAP:n käynnistäminen ensimmäistä kertaa

Nyt ZAP on onnistuneesti asennettu. Koska sitä ei saa käyttää mielivaltaisesti mihin tahansa ympäristöön, tarvitaan sellainen ympäristö, mitä vastaan ZAP:n käyttöä saa harjoitella aivan vapaasti.

## 4.5 OWASP Broken Web Applications Project

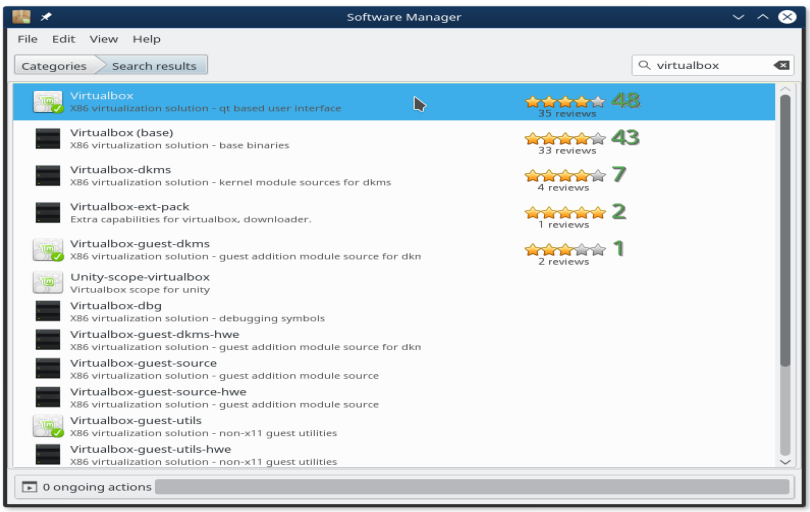
OWASP Broken Web Applications Project (BWAP) on tarkoituksenmukaisesti rikkonainen verkkopalvelualusta, jota vastaan hyökkääminen on täysin sallittua ja se tapahtuu omalla henkilökohtaisella tietokoneella. BWAP on virtuaaliympäristö, jonka käynnistämiseen tarvitaan virtuaalitietokoneiden ajoalusta. Tätä tarkoitusta varten hyvä vaihtoehto on Oraclen VirtualBox.

BWAP on jo ikääntynyt projekti, eikä sitä enää ylläpidetä. Se kuitenkin sisältää monia haavoittuvuuksia, jotka ovat edelleen hyvin ajankohtaisia ja tähän demonstraatiotarkoitukseen BWAP on sopiva. [12]

### 4.5.1 VirtualBoxin asentaminen

1. Komennolla *sudo apt-get update && sudo apt-get dist-upgrade -y* päivitetään Linuxin pakettitiedot, sekä päivitetään jo asennetut sovellukset. Tämän komennon jälkeen voidaan olla varmoja siitä, että saatavilla on uusimmat versiot haettavista sovelluksista. [13]

2. Virtualbox asennetaan Linux Mintin ohjelmanhallintatyökalu ”Software Manager”:in kautta. Kuvasta 6 nähdään kuinka hakutermillä ”virtualbox” ohjelmanhallinta löytää ”VirtualBox”-ohjelman. Kaksoisklikkaamalla tätä elementtiä avautuu ikkuna, jossa on nappi ”Install”. Tätä nappia painamalla Virtualboxin asentaminen käynnistyy. [13]

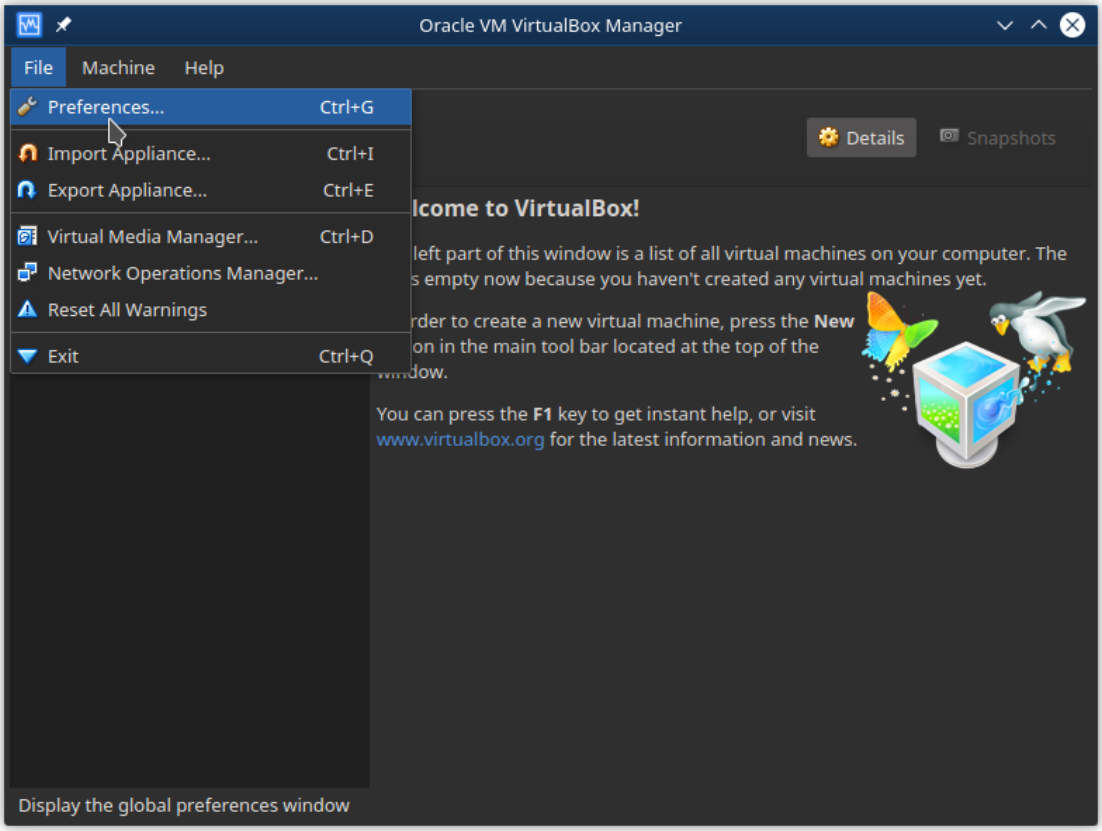



Kuva 6. VirtualBox Linux Mintin ohjelmistohallintatyökalussa

### 4.5.2 BWAP:n käynnistäminen VirtualBoxissa

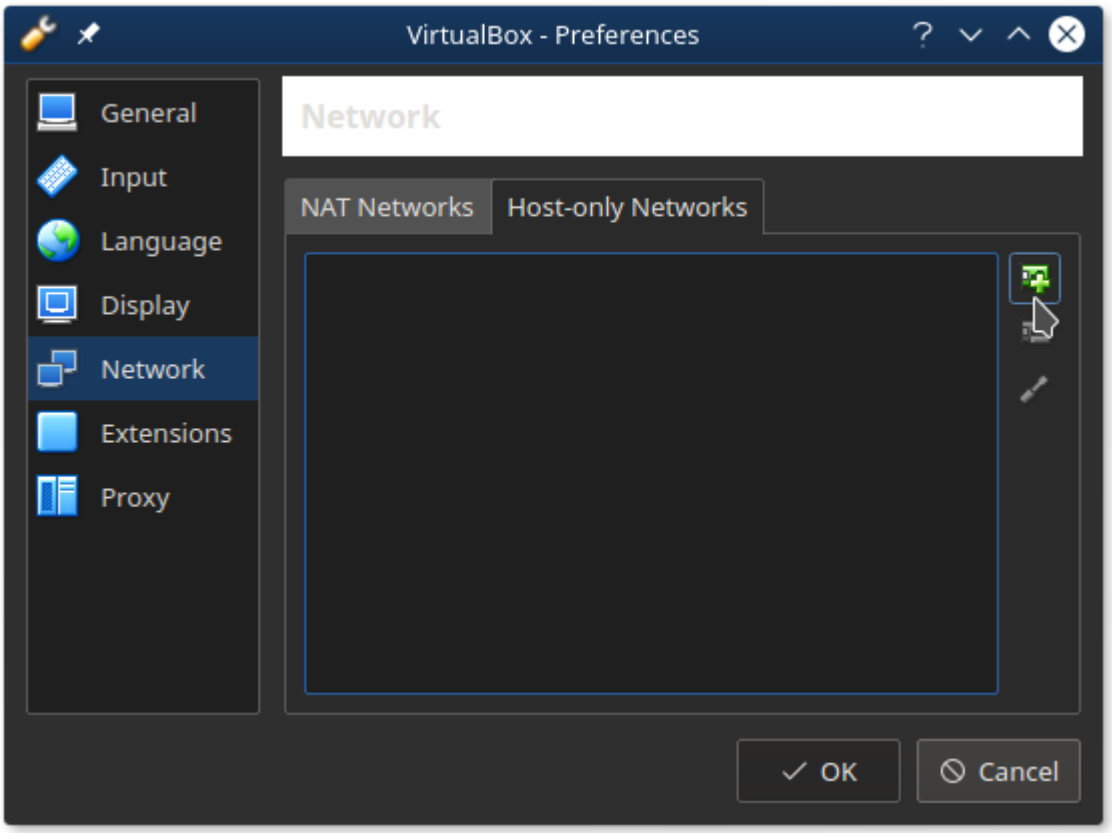
1. OWASP BWAP -virtuaalitietokone löytyy verkkosivulta: *https://sourceforge.net/projects/owaspbwa/*. Latauksen valmistuttua .7z-paketti täytyy purkaa.

2. Jotta OWASP BWAP -virtuaalitietokone voidaan käynnistää, täytyy ensin avata Virtualbox ja säätää muutamaa asetusta. Kuvassa 7 nähdään kursorin alla asetusvalikon painike.




Kuva 7. VirtualBoxin asetusten avaaminen ohjelman alivalikoista

3. Asetusikkunassa valitaan ensin "*Network*" ja sitten verkkoasetuksista välilehti "*Host-only Networks*". Kuvassa 8 nähdään kursorin alla ”+”-painike, josta lisätään uusi virtuaaliverkon adapteri. Painiketta klikattaessa uusi adapteri tulee näkyviin nimellä “vboxnet0”. Verkkoasetukset voi sulkea “Ok”-painikkeesta.

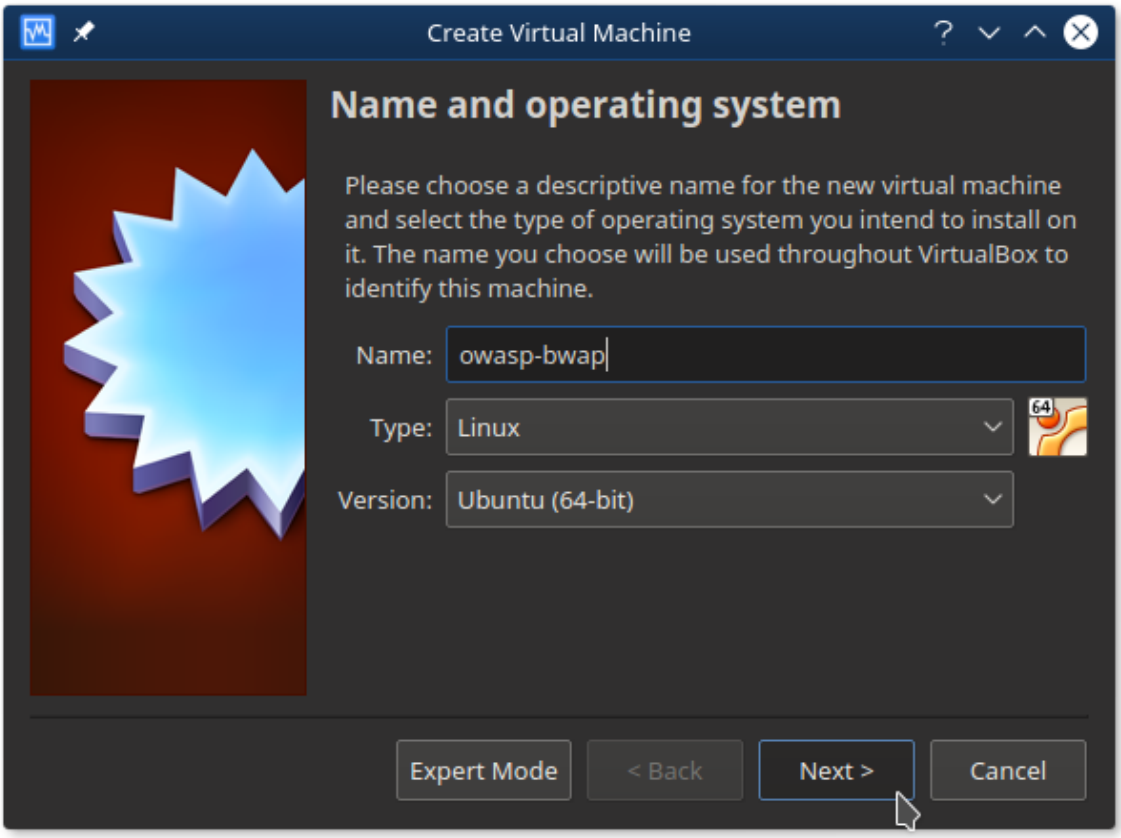



Kuva 8. VirtualBoxin verkkoasetukset

4. Uusi virtuaalikone luodaan painamalla sinistä tähtisymbolia vasemmassa yläkulmassa "*New*". Avautuva ikkuna nähdään kuvassa 9.

    - Nimeksi "*Name*" asetetaan: owasp-bwap.
    - Tyypiksi "*Type*" valitaan: Linux.
    - Versioksi "*Version*" valitaan: Ubuntu (64-bit).

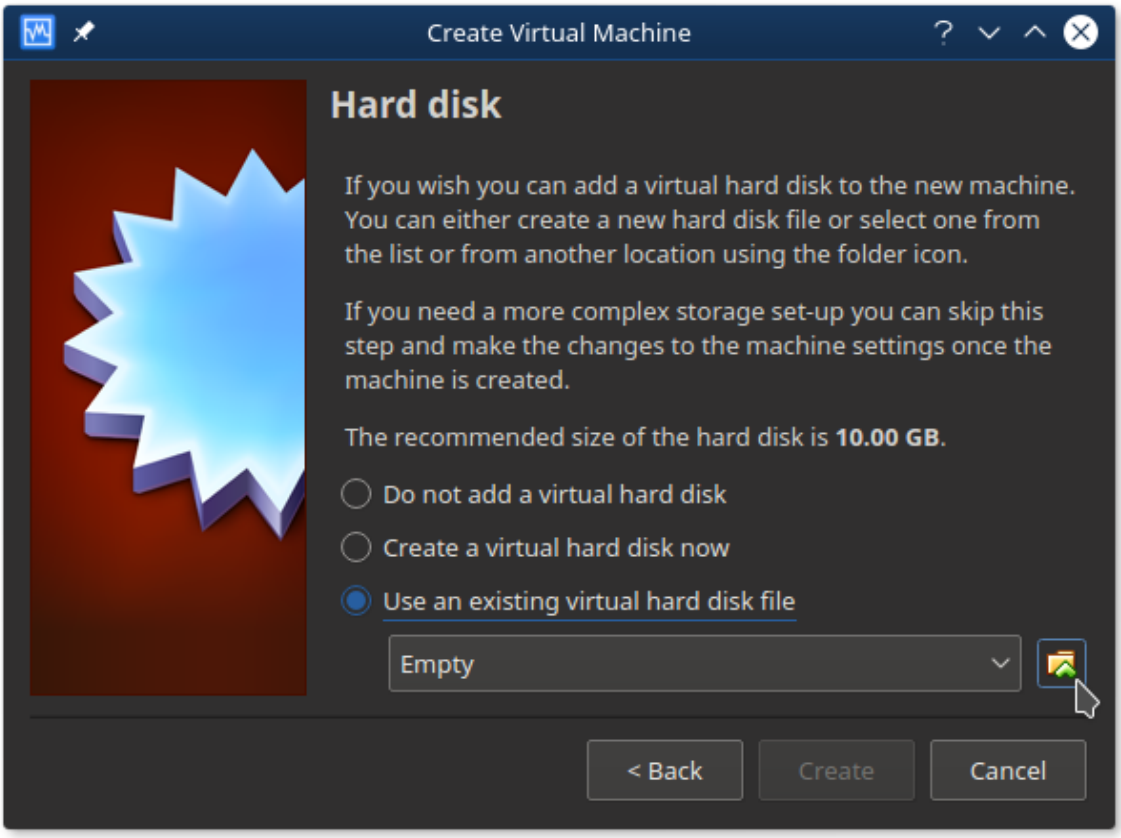
    Seuraavaan vaiheeseen siirrytään painamalla “Next”.




Kuva 9. OWAS BWAP -virtuaalikoneen luominen

5. Muistin “Memory” määräksi asetetaan 512 mb ja siirrytään seuraavaan valikkoon painamalla “Next”.

6. Kuvan 10 mukaisesti valitaan "*Use an existing virtual hard disk drive*". Seuraavaan vaiheeseen siirrytään klikkaamalla kansion kuvaa, joka näkyy kuvassa 10 kursorin alla.

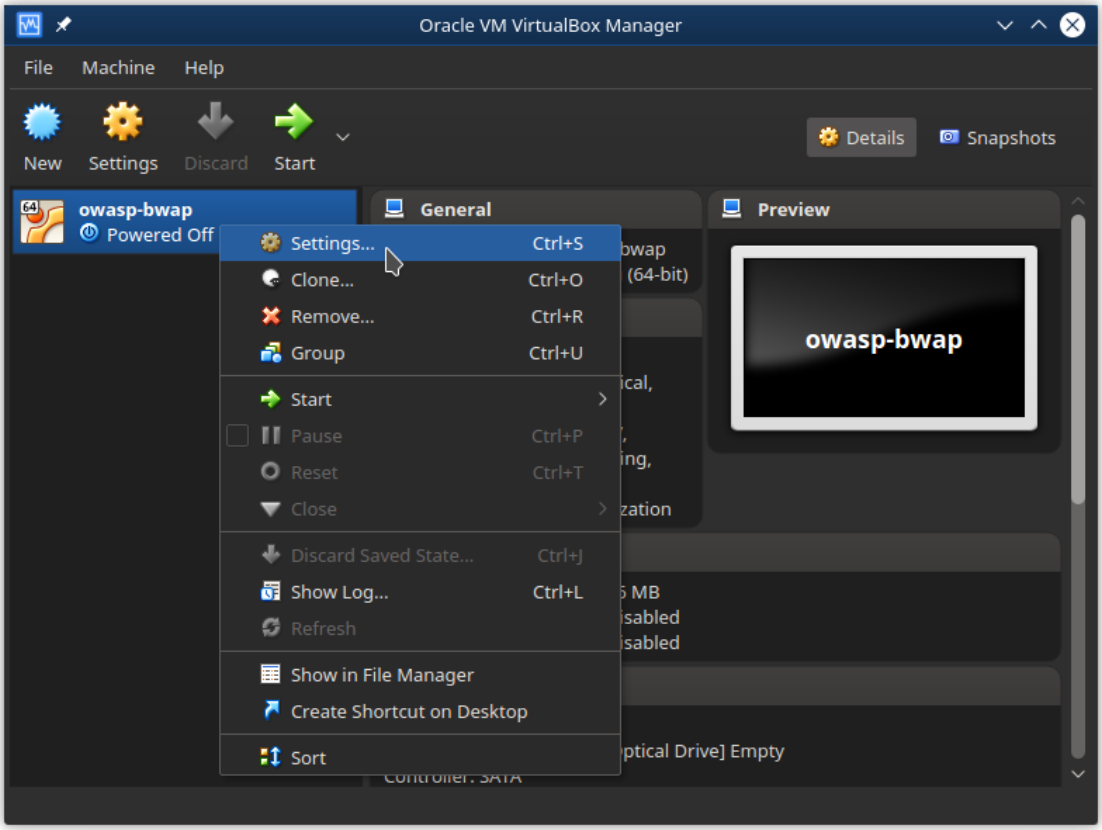



Kuva 10. OWASP BWAP -virtuaalikonetiedoston osoittaminen VirtualBoxille

7. Jotta Virtualbox voi käynnistää OWASP BWAP –virtuaalitietokoneen, täytyy sen sijainti osoittaa Virtualboxille. Etsitään "*OWASP Broken Web Apps-cl1.vmdk*" niminen tiedosto siitä kansiosta, minne tiedosto purettiin kohdassa 1. Täytyy huomioida, ettei valittavan tiedoston tiedostopäätteen edessä ole *s001, s002* tai vastaavaa.

8. Virtuaalitietokone luodaan painamalla "Create".

9. Jotta virtuaalitietokone ei ota yhteyttä internetiin, täytyy sen verkkoasetuksia muokata. Asetuksiin päästään klikkaamalla luodun virtuaalitietokoneen nimeä oikealla hiiren painikkeella. Kuvassa 11 näkyy avautuva alavalikko. Virtuaalitietokoneen asetuksiin pääsee painamalla kuvassa 11 kursorin alla näkyvää "Settings..."-painiketta.




Kuva 11. OWASP BWAP -virtuaalitietokoneen asetusvalikon sijainti alavalikoissa

10. Valitaan avautuvan ikkunan vasemmasta reunasta “Network”.

11. Välilehdeltä "*Adapter 1*".

    a. Merkitään "*Enable Network Adapter*".

    b. Verkkosovittimen liitospisteeksi "*Attached to*" valitaan: "*Host-only Adapter*".

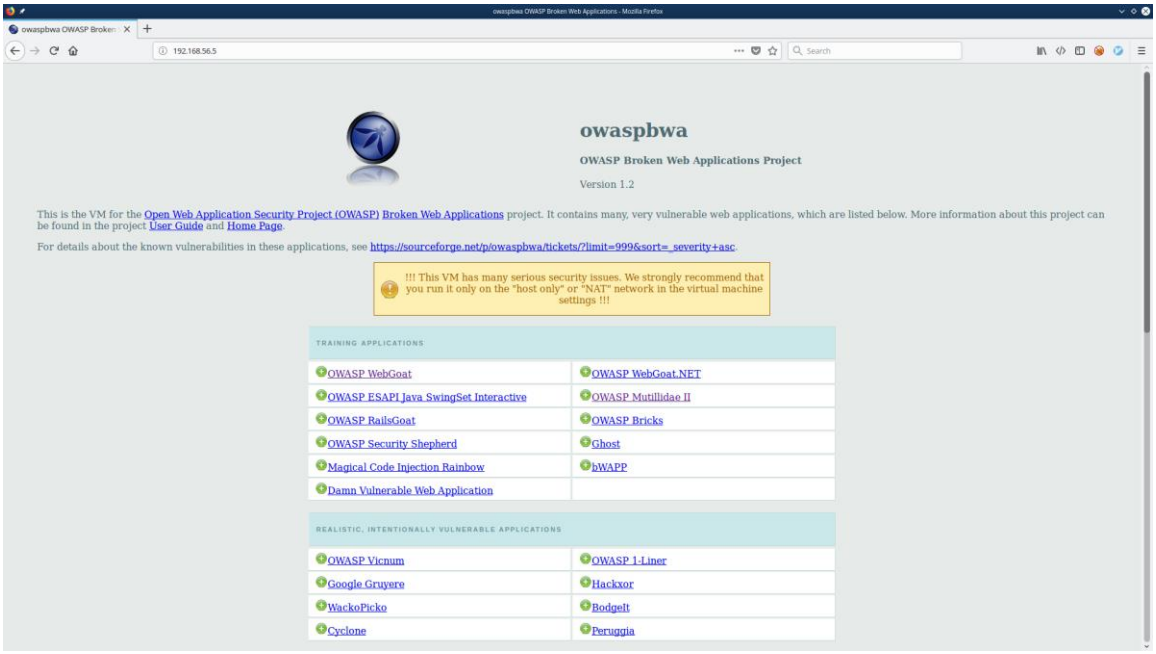
    c. Nimeksi "*Name*" asetetaan kohdassa 5. luotu verkkosovitin: "*vboxnet 0*".

    Painetaan ”Next” ja virtuaalitietokone on nyt käyttövalmis.

12. Virtuaalitietokone käynnistyy kaksoisklikkaamalla. Käynnistyminen vie hetken. Kun virtuaalitietokone on käynnistynyt, se ilmoittaa missä IP-osoitteessa OWASP BWAP -verkkosovellus sijaitsee, esimerkiksi ” *You can access the web apps at http://192.168.56.6/*”.

13. Sovelluksiin päästään käsiksi asettamalla kohdassa 12. ilmoitettu osoite ”*http://192.168.56.6”* verkkoselaimeen isäntäkoneella. On huomioitava, että tämä kyseinen osoite ei välttämättä aina ole sama.

[14]



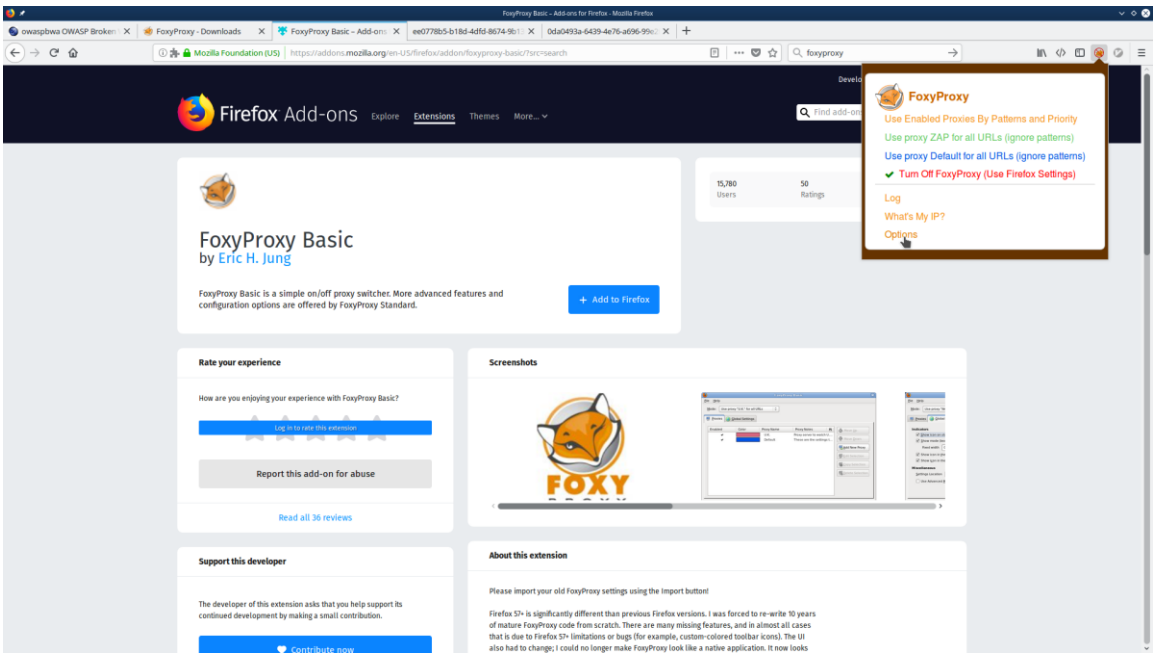

Kuva 12. OWASP BWAP -ohjelmiston etusivu selaimessa

## 4.6 Selaimen liikenteen ohjaaminen ZAP:iin

Kun sekä ZAP että sovellusympäristö on saatu pystytettyä, täytyy selaimen liikenne ohjata ZAP:n läpi. Tähän tarkoitukseen on kätevintä asentaa selaimeen laajennus, esimerkiksi FoxyProxy. FoxyProxyn voi asentaa useimmille selaimille, mutta tässä esimerkissä selaimena toimii Firefox. Jotta välityspalvelinta ei tarvitsisi jatkuvasti kytkeä pois päältä normaalin internetin selailun takia, kannattaa pitää välityspalvelin päällä toisessa selaimessa ja selata Internetiä toisessa. FoxyProxyyn voi myös tehdä asetuksia, joilla liikenne saadaan välitettyä vain tietyistä osoitteista, mutta on todennäköisesti helpompaa pitää ZAP:lla tehtävä toiminta kokonaan erossa normaalista internetin selaamisesta.

1. FoxyProxy:n voi ladata osoitteesta *https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-basic/?src=search.* Laajennus asennetaan selaimeen painamalla ”*+ Add to Firefox*” -nappia.

2. Kun lisäosa on asennettu, sen ikoni ilmestyy selaimen oikeaan yläkulmaan. Jotta FoxyProxy osaa välittää liikenteen oikeaan osoitteeseen, täytyy sen asetuksia vaihtaa. Kuvassa 13 nähdään FoxyProxy -ikonia klikattaessa avautuva valikko, jossa kursorin alla näkyy ”Option”-painike.



Kuva 13. FoxyProxyn asetuksiin siirtyminen

3. Vasemmassa yläkulmassa olevasta ”+ Add” painikkeesta luodaan uusi välityspalvelin.

4. Asetetaan arvot:

    a. Välityspalvelimen tyyppi, "*Proxy type*": http.

    b. Nimi tai kuvaus, "*Title or Description*": ZAP.

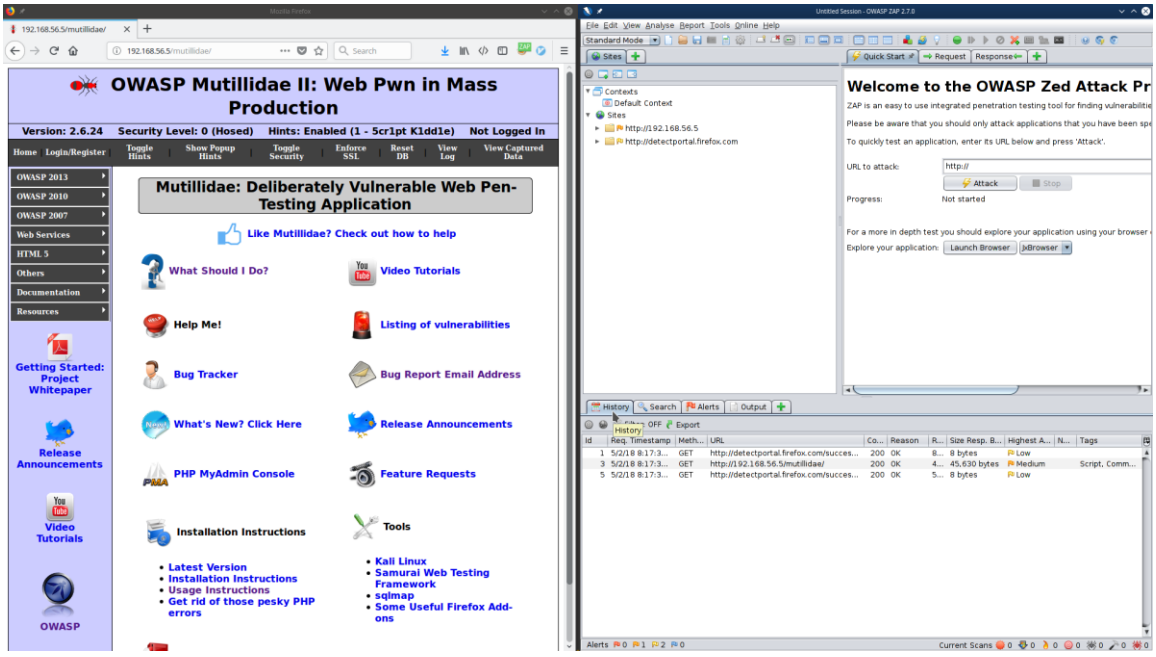
    c. Ip-osoite, "*IP address, DNS name, server name*": 127.0.0.1.

    d. Portti, "*Port*": 8080.

Tallennetaan asetukset painamalla ”Save”.

5. Jotta selain käyttää luotua välityspalvelinta, täytyy se asettaa aktiiviseksi klikkaamalla FoxyProxyn kettusymbolia ja valitsemalla valikosta ”*Use proxy ZAP for all URLs (ignore patterns*)”.

6. Jotta liikenne ohjautuu oikein, täytyy ZAP käynnistää. Nyt kun selaimessa siirrytään kappaleen 4.5.2 kohdan 13 osoitteeseen, tulisi verkkoliikenteen näkyä ZAP:n History-välilehdellä.




Kuva 14. Selaimen liikennettä ZAP:ssa

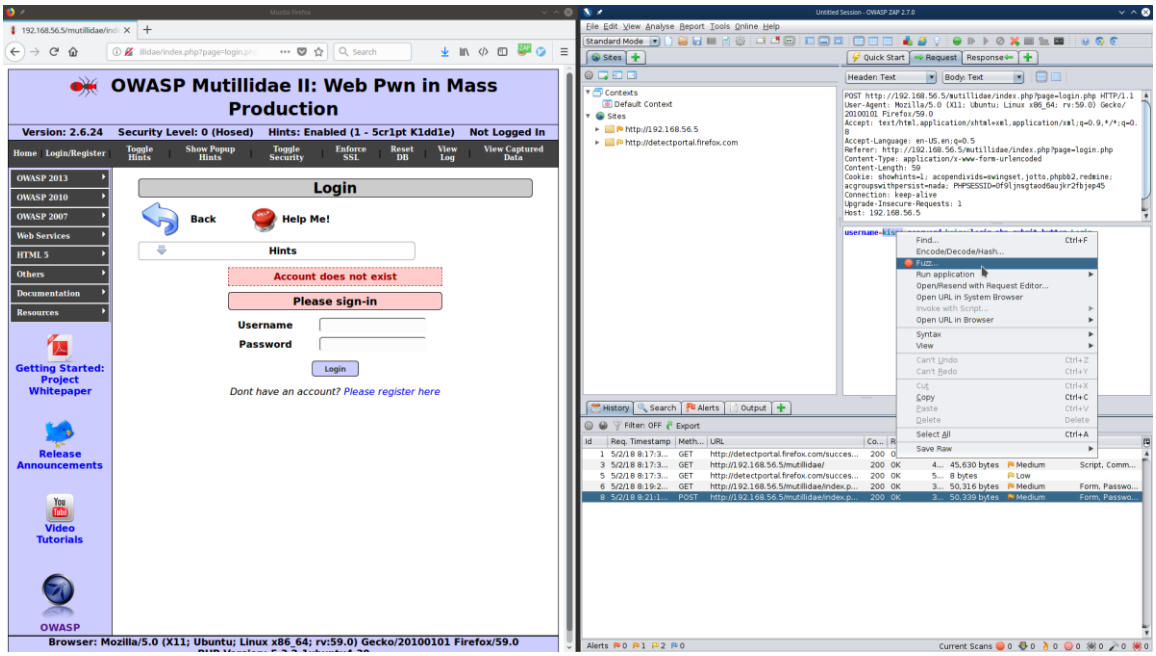
## 4.7 ZAP:n käytön esimerkki kirjautumissivulla

Seuraavassa esimerkissä pyritään löytämään haavoittuvuus kirjautumissivulta antamalla kirjautumislomakkeelle ilkeämielistä dataa. Tällaiset haavoittuvuudet ovat nykypäivänä hyvin tiedossa, mutta mikäli esimerkissä vastaan tulevia virheitä pääsee oikeisiin tuotantopalveluihin, voivat seuraukset olla vakavia.

1. Esimerkkinä käytetään kirjautumissivua, joka löytyy BWAP:n osoitteesta: *http://<**OWASP BWAP osoite**>/mutillidae/index.php?page=login.php.* Liitetään URL:iin osoite kappaleen 4.5.2 kohdasta 13.

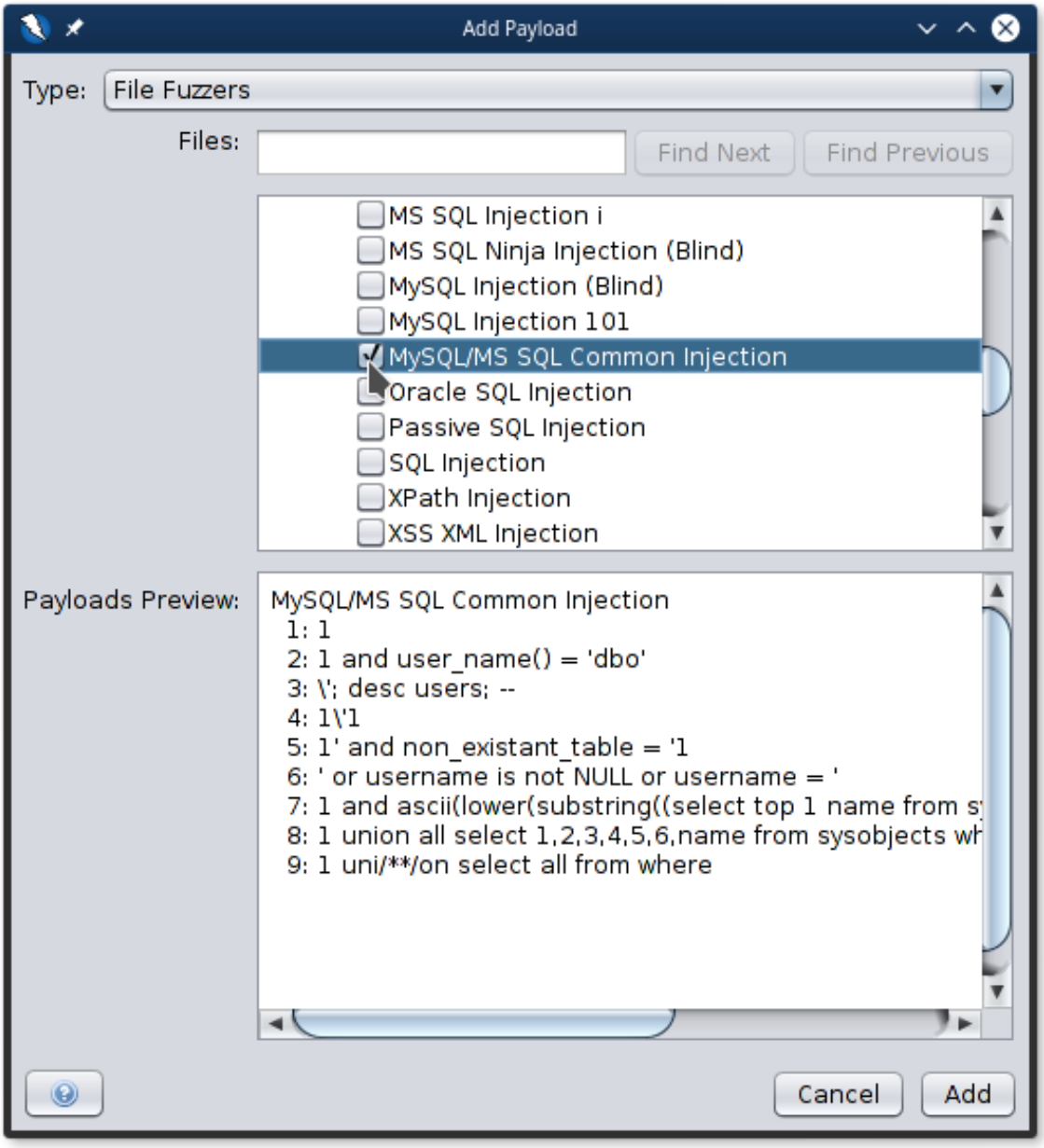
2. Kun kirjautumislomakkeeseen kenttiin ”*Username*” ja ”*Password*” asetetaan mitä tahansa ja lähetetään lomake painamalla ”*Login*”, ZAP:n historiavälilehdellä näkyy POST-kutsu. Tätä kutsua voidaan tutkia tarkemmin klikkaamalla sitä. Yläoikealla ikkunassa näkyy kutsu, jonka kirjautumispainike lähetti palvelimelle.

3. Lähetetään tämä sama kutsu uudestaan, kuitenkin siten että ZAP vaihtaa lomakkeen arvoja automaattisesti sellaisella datalla, joka on laadittu murtautumistarkoituksessa. Tällaista satunnaisdatan syöttöä kutsutaan fussaukseksi, englanniksi fuzzing. Jotta kutsua voidaan fussata, täytyy alkuperäisestä kutsusta maalata se parametri, jonka ZAP:n halutaan korvaavan. Kun parametri on maalattu ja siitä klikataan hiiren oikealla painikkeella, avautuu kuvassa 15 näkyvä alivalikko. Fussaustyökalu avautuu, kun painetaan ”Fuzz…”




Kuva 15. Fussaustyökalun sijainti ZAP:n alivalikoissa

4. Maalatun parametrin tilalle fussattavat arvot lisätään painamalla ”Payloads…”. Avautuneessa ikkunassa on painike ”Add…”, jonka klikkaaminen aukaisee uuden ikkunan. Tästä avautuneesta ikkunasta valitaan ”Type”-pudotusvalikosta ”File Fuzzers”. ZAP:iin on sisäänrakennettu kokoelmia erilaisista fussauslistoista. Kun ”jbrofuzz”:n vieressä olevaa nuolta klikataan, avautuu alavalikko. ”Injection” rivin viereistä nuolta painaessa avautuu jälleen alavalikko, josta valitaan kuvan 16 mukaisesti ”*MySQL/MS SQL Common Injection*”. Tämä lista sisältää sellaisia SQL-kyselyitä, jotka voivat jollain tapaa rikkoa kyselyn oikeellisuuden.

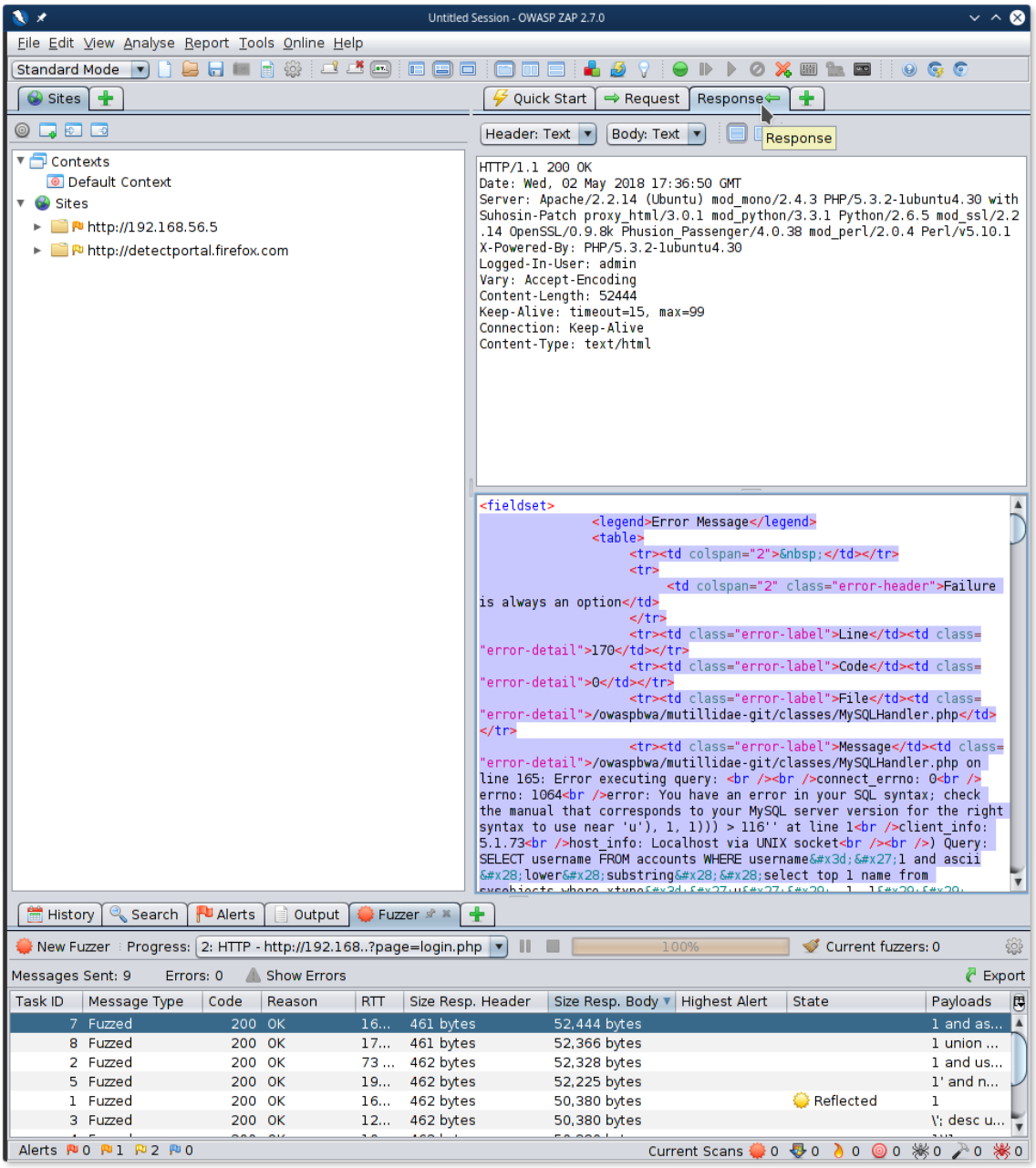



Kuva 16. ZAP:n sisäänrakennettu SQL-injektioiden löytämiseen tarkoitettu fussauslista

5. Lisätään lista painamalla ”Add” ja aloitetaan fussaus painamalla ”Start Fuzzer”. ZAP käy listan läpi, ja jokaisen rivin kohdalla se lähettää kohdassa 2 lähetetyn kirjautumislomakkeen, kuitenkin siten, että korvaa kohdassa 3 maalatun parametrin listassa olevalla rivillä.

6. Kun ZAP on suorittanut fussauksen, voidaan sen käymiä kutsuja käydä läpi ZAP:n alalaidassa olevasta ”Fuzzer”-välilehdestä. Tässä tapauksessa todetaan, että parametri "`1 and ascii(lower(substring((select top 1 name from sysobjects where xtype='u'), 1, 1))) > 116`" aiheuttaa tietokantakyselyssä virheen

tutkimalla vastausta, jonka palvelin palautti. Kuvassa 17 nähdään, että tämä rivi on lisätty lomakkeeseen seitsemännessä kutsussa (Task ID).




Kuva 17. Tietokantavirhe ZAP:ssa

1. Sama tulos voidaan todeta myös manuaalisesti testaamalla, asettamalla "`1 and ascii(lower(substring((select top 1 name from sysobjects where xtype='u'), 1, 1))) > 116`" kirjautumislomakkeen "*Username*"-kenttään ja lähettämällä lomake painamalla "*Login*".

Kuten huomataan, fussaamalla ZAP voi tehdä kutsuja huomattavasti nopeammin kuin ihminen pystyisi kirjoittamaan niitä käsin. ZAP:lle syötetty lista sisälsi yksinkertaisia SQL-injektioita, joiden tarkoitus on saada aikaiseksi virhe palvelimen koodissa, joka kutsuu palvelimella ajettavaa MySQL-tietokantaa. Lyhyesti sanottuna käyttäjäsyötettä ei ole hallittu oikein. Kun kirjautumisominaisuus yrittää hakea "`1 and ascii(lower(substring((select top 1 name from sysobjects where xtype='u'), 1, 1))) > 116`" nimistä käyttäjää tietokannasta, menee tämä parametri suoraan ajettavaan tietokantakyselyyn. Tämä aiheuttaa kyselyssä syntaksivirheen, joka taas aiheuttaa sen, ettei ohjelma voi jatkaa, koska MySQL ei osaa suorittaa sille annettua virheellistä kyselyä. Käydään hyökkäystapoja tarkemmin hieman myöhemmin.

### 4.7.1 SQL-virheen hyväksikäyttö

Edellisen kohdan perusteella tiedämme, että käyttäjänimikentässä annettu data menee suoraan osaksi MySQL-kyselyä. Tämä tarkoittaa sitä, että hakkerit pystyvät tuon kentän avulla tekemään omia kyselyitään palvelimelle. Koska tämä on hyvin tunnettu ja tehokas keino saada käyttäjädataa murretuilta sivustoilta, on sitä varten olemassa ohjelmisto automatisoimaan tietokannan haku. Sqlmap (https://github.com/sqlmapproject/sqlmap) on avoimen lähdekoodin ohjelmisto, joka on kehitetty nimenomaan hyväksikäyttämään SQL-injektiohaavoittuvuuksia. Se tekee automaattisesti erilaisia tietokantakyselyitä ja yrittää täten rakentaa kuvan siitä, millainen kanta ohjelmiston takana on. Seuraava esimerkki osoittaa, kuinka tehokas työkalu se on. On kuitenkin huomioitava, että Sqlmapia ei saa käyttää luvattomasti, sillä sen luvaton käyttö on rangaistava teko, aivan kuten ZAP:kin. Seuraava on esimerkki siitä, miten vakava SQL-injektion löytyminen on. Yleisesti ottaen ei ole olemassa pieniä tai suuria SQL-injektiohaavoittuvuuksia, on vain SQL-injektiohaavoittuvuuksia.

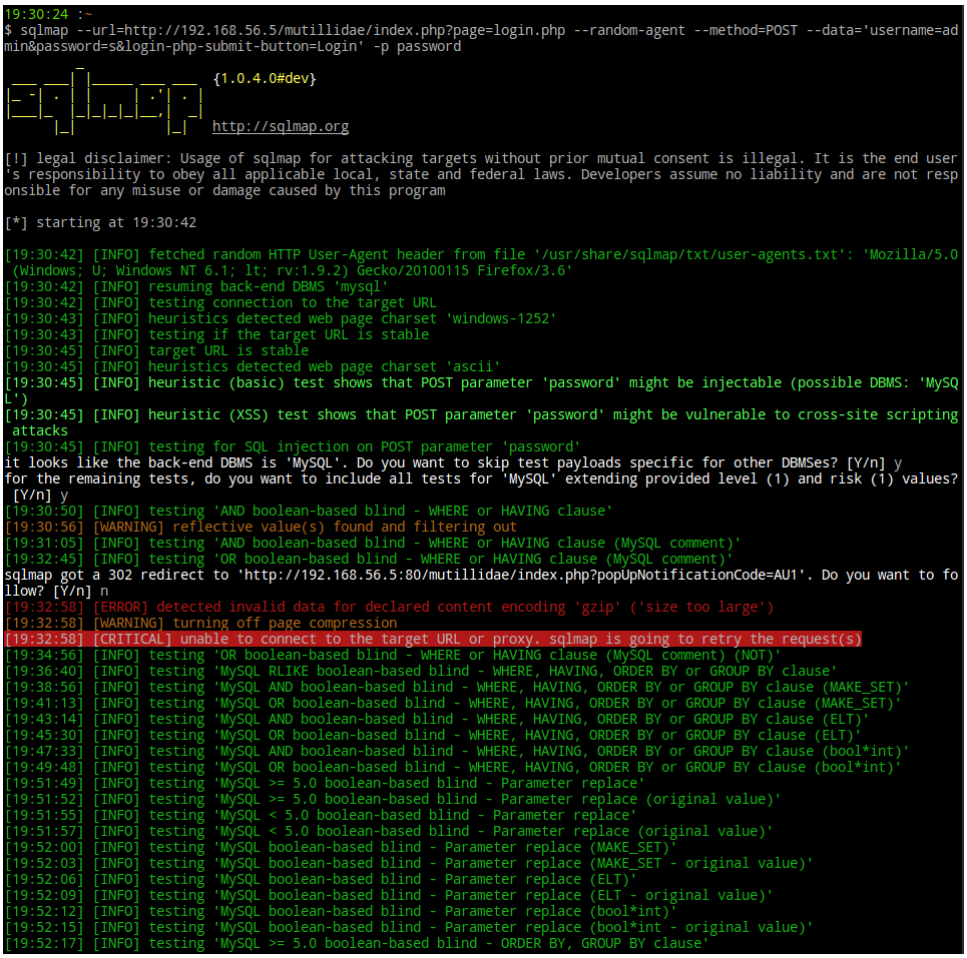
### 4.7.2 Sqlmapin asennus ja käyttö

1. Asennetaan sqlmap komentorivikomennolla: *sudo apt-get install sqlmap.*

2. Ajetaan sqlmap komennolla:

*sqlmap --url=http://192.168.56.5/mutillidae/index.php?page=login.php --random-agent --method=POST --data='username=admin&password=s&login-php-submit-button=Login' -p password*.

Kuvassa 18 nähdään sqlmapin tuottamaa konsolitulostetta.




Kuva 18. Sqlmapin toimintaa

3. Kun sqlmap kysyy käyttäjäsyötettä, vastataan seuraavasti:

   - *it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y*
   - *for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y*
   - *sqlmap got a 302 redirect to 'http://192.168.56.5:80/mutillidae/index.php?popUpNotificationCode=AU1'. Do you want to follow? [Y/n] n*

- *POST parameter 'password' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n*

4. Sqlmap on todennut, että kanta on injektoitavissa. Katsotaan seuraavaksi, mitä se sisältää, lisäämällä edellisen komennon perään: *--dbs*

    - Kun sqlmap on valmis, huomataan, että se löysi kaikkien palvelimilla olevien sovellusten tietokannat. Keskitytään kuitenkin mutillidaen kantaan, koska se on murron kohteena.

5. Korvataan edellisestä komennosta *--dbs* seuraavalla: *--tables -D "mutillidae".*

    - Nähdään, millaisia tauluja mutillidaen kanta pitää sisällään, koska murron tavoitteena on saada käyttäjätietoja. Kiinnostavin taulu on todennäköisesti "*accounts*" eli tilit.

6. Korvataan edellisestä komennosta *--tables -D "mutillidae"* seuraavalla: *--dump -D "mutillidae" -T "accounts".*

Koska salasanat on tallennettu tietokantaan salaamattomina, on helppo kirjautua sisään käyttäjänä "*adrian*" ja salasanalla "*somepassword*". Nyt pystytään tekemään samoja asioita, joita tämä käyttäjä pystyy sovelluksessa tekemään. Tietokanta on nyt täysin hyökkääjän vallassa, ja hän voi tehdä sillä mitä haluaa, esimerkiksi ladata kaiken itselleen, tuhota kaikki käyttäjät tai luoda itsellensä oman käyttäjän. Näin räikeät virheet ovat toki nykyään harvassa, mutta virheitä voi sattua kenelle tahansa. Koska ZAP:n ja sqlmap:n kaltaisia tehokkaita työkaluja on vapaassa jakelussa, voi kuka tahansa verkon käyttäjä ottaa tilaisuuden omiin käsiinsä ja yrittää tietomurtoa.

Jokaisen kehittäjän tulee tietää mitä ovat ne aseet, joita vastaan he joutuvat taistelemaan, ja näin rakentamaan itselleen mahdollisimman kestävän linnakkeen. Tällaisten esimerkkien tarkoituksena on herätellä kehittäjiä siihen, että hakkerointi voi olla pahimmillaan erittäin helppoa. Nykyaikaisissa ohjelmistoalustoissa ja -rungoissa (framework) yleisimmät tietoturva-aukot on otettu huomioon, mutta toivottavasti seuraavaksi läpikäytävät esimerkit avaavat, millaisia nykyaikaiset hyökkäysmenetelmät ovat ja miten niitä vastaan voidaan suojautua.

# 5 Esimerkkejä hyökkäystavoista

Kuten aikaisemmin on mainittu, OWASP tarjoaa erilaisia tietoturva-aiheisia sisältöjä, ja näistä yksi on OWASP Top Ten -lista. Tämä lista sisältää ne kymmenen tietoturvauhkaa, jotka ovat OWASP:n mukaan kriittisimpiä. Kriteerit tälle listalle ovat arvioita siitä, kuinka hyväksikäytettäviä tai helposti löydettäviä tietoturvauhkat ovat, ja kuinka vakavat niiden seuraukset voivat olla. [15] Listan päämäärä on opettaa sovelluskehittäjiä, heidän tiimejään sekä organisaatioita yleisimpien tietoturva-aukkojen seuraamuksista. Hyökkäystapoja on lukemattomia määriä ja ne kehittyvät päivä päivältä enemmän, joten jokaisen tutkimiseen menisi vuosia. Käydään kuitenkin läpi muutama hyökkäystapa OWASP Top Ten -listalta, jotta saamme asioille kontekstia.

## 5.1 SQL-injektio (SQL-injection)

SQL-injektiot lienevät yleisin ja tunnetuin tietomurtojen syy, mutta nykypäivänä injektiohyökkäys on käsitteenä hyvin tunnettu ja hyökkäysten estäminen helppoa. Kuitenkin SQL-injektio on pysynyt jo vuosia OWASP Top Ten -listan kärjessä, koska kun järjestelmästä löytyy injektiomahdollisuus, ovat seuraamukset todennäköisesti vakavat, kuten kohdassa 4.6.1 läpikäydyssä esimerkissä tuli ilmi. Injektioksi nimitetään sellaista tilannetta, missä jokin vaihtaa tarkoituksenmukaisen ja harmittoman ohjeen tilalle tuhoisan ohjeen. Esimerkiksi ohjelman on tarkoitus hakea tietokannasta kaikki käyttäjät, mutta injektion myötä tietokantaohjetta on muokattu poistamaan kaikkien käyttäjien tietueet. [16, s. 249]

SQL-injektio toimii yksinkertaisimmillaan siten, että käyttäjäsyöte saadaan verkkosivulta ujutettua osaksi tietokannan hakuehtoa. Hyvänä esimerkkinä toimii OWASP BWAP:n Mutillidae II:n sisäänkirjautumissivu (*http://<**OWASP BWAP osoitteesi**>/mutillidae/index.php?page=login.php* katso osoite kappaleen 4.4.2 kohdasta 20). Siinä molemmat, sekä käyttäjänimi että salasana, ovat suoria parametreja tietokantakyselylle. Tämä voidaan todeta, mikäli käyttäjänimeksi asetetaan "*admin*" ja salasanaksi lainausmerkki (‘). Tällöin tietokantakyselyn syntaksi on virheellinen, ja siitä palautuu palvelimelta virheviesti:

```
/owaspbwa/mutillidae-git/classes/MySQLHandler.php on line 165:
Error executing query:

connect_errno: 0
errno: 1064
error: You have an error in your SQL syntax; check the manual
that corresponds to your MySQL server version for the right syn-
tax to use near ''''' at line 2
client_info: 5.1.73
host_info: Localhost via UNIX socket

) Query: SELECT * FROM accounts WHERE username='admin' AND pass-
word=''' (0) [Exception]
```

Virheviestin osana on viallinen kysely: ”`Query: SELECT username FROM accounts WHERE username='admin' AND password='''`. Salasanakenttään syötetty lainausmerkki `'` sulkee hakuehdon, jonka johdosta viimeinen `'`-merkki rikkoo kyselyn oikeellisuuden. Tietokanta heittää poikkeuksen, jota ohjelma ei osaa käsitellä. Poikkeus lopulta kaataa ohjelman, ja palvelin on säädetty siten, että se palauttaa kaiken, mitä ohjelma tulostaa ruudulle. Tässä tapauksessa ohjelma tulostaa virheviestin. Nyt tiedetään, että käyttäjän tunnistautuminen toimii todennäköisesti vain siten, että käyttäjän tulee syöttää sellainen käyttäjätunnus-salasanayhdistelmä, joka löytyy tietokannasta.

Koska käyttäjän syöte menee raakana osaksi kyselyä, voimme muotoilla sitä hieman tarkoitustamme sopivammaksi. Käytetään edelleen käyttäjänimenä *"admin"*, ihan vain valistuneena arvauksena siitä, että kannassa on sellainen käyttäjä. Asetetaan salasanakenttään seuraava: `' or password like '%`. Kyselystä tulee täten huomattavasti erilainen kuin alkuperäisestä: "`SELECT username FROM accounts WHERE username='admin' AND password='' or password like '%'`". Tämä tarkoittaa käytännössä, että kannasta haetaan käyttäjänimi "*admin*", jonka salasana on joko tyhjä tai mikä tahansa merkki. Toisin sanoen mikäli "*admin*" käyttäjällä on salasanana mikä tahansa merkki, se on tälle kyselylle sopiva. SQL:ssä prosenttimerkki-% on niin sanottu ”jokerimerkki”, joka vastaa mitä tahansa merkkiä ja kuinka monta kertaa tahansa.

## 5.2 Rikkonainen tunnistautuminen (Broken authentication)

Hyvin usein verkkopalveluita käytettäessä käyttäjät joutuvat luomaan palveluun tunnukset ja kirjautumaan sisään omilla tunnuksillaan. Yleisin tunnistautumistapa on käyttäjätunnus - salasanayhdistelmä. Tämä on rikkonaisen tunnistautumisen keskiössä. Yksittäisen käyttäjän tulee keksiä jokaiseen palveluun uusi salasana ja muistaa se. Kun tilejä ja tunnuksia on kymmeniin palveluihin, monet kokevat salasanojen muistamisen hankalaksi ja täten kokevat parhaaksi käyttää samaa salasanaa monessa palvelussa. Samaan aikaan tietomurtoja tapahtuu kaiken aikaa, isoihinkin palveluihin, esimerkkinä Yahoo!:n vuoden 2013 tietomurto, jossa yli miljardin käyttäjätilin nimet, sähköpostiosoitteet, osoitteet, puhelinnumerot, syntymäajat, salatut salasanat ja joissain tapauksissa salaamattomat turvakysymykset vuotivat julkisuuteen [17] [18]. Kun tällaisia vuotoja tapahtuu ja vuodetut tiedot päätyvät julkisiksi, pystyvät hakkerit automaatiohyökkäyksillä yrittämään näitä vuodettuja yleisiä salasanoja ja käyttäjänimiä taukoamatta eri palveluihin siinä toivossa, että jokin osuu oikeaan.

Hyökkäyksissä voidaan käyttää myös palveluiden ylläpitoon tarkoitettujen sovellusten oletussalasanoja ja -tunnuksia. Hyvä esimerkki siitä, kuinka oletustunnuksia voidaan käyttää, on keväällä 2016 hetken aikaa ylläpidetty VNC Roulette -niminen palvelu. Tämän palvelun ylläpitäjä oli skannannut julkisen verkon osoitteita löytääkseen julkisverkosta sellaisia portteja, joihin VNC oli kytketty. VNC eli Virtual Network Computing -ohjelmistoa käytetään useimmiten tietokoneiden etähallintaan. Kävi ilmi, että verkossa on huomattava määrä avoimia VNC-palvelimia, ja tämä kyseinen hakkeri väitti, että hän sai saaliiksi 23 gigatavun edestä kuvankaappauksia erinäisistä VNC-syötteistä. Avoimien VNC-palvelimien joukossa oli yksityishenkilöiden lisäksi tehdaslaitteiston hallintajärjestelmiä, potilastietokantoja sekä valvontakameroiden hallintapaneeleita. [19] [20] Tämä on erityisen hyvä esimerkki siitä, että kukaan verkon käyttäjä tai mikään verkkopalvelu ei ole liian pieni hyökkäyskohde.

## 5.3 Luvaton ohjelmakoodin ajaminen (Cross site scripting, XSS)

Kun SQL-injektiolla pyritään ujuttamaan virheellisiä ohjeita tietokantaohjelmistoon, XSS hyökkäyksen tavoite on saada verkkosivulla sinne kuulumatonta ohjelmakoodia. XSS-hyökkäyksissä hyödynnetään käyttäjän luottamusta verkkosovellukseen. Pyrkimys on saada käyttäjä tai selain huijattua tekemään jokin toiminto tai lähettämään tietoa ulkopuoliseen osoitteeseen. [16, s. 263]

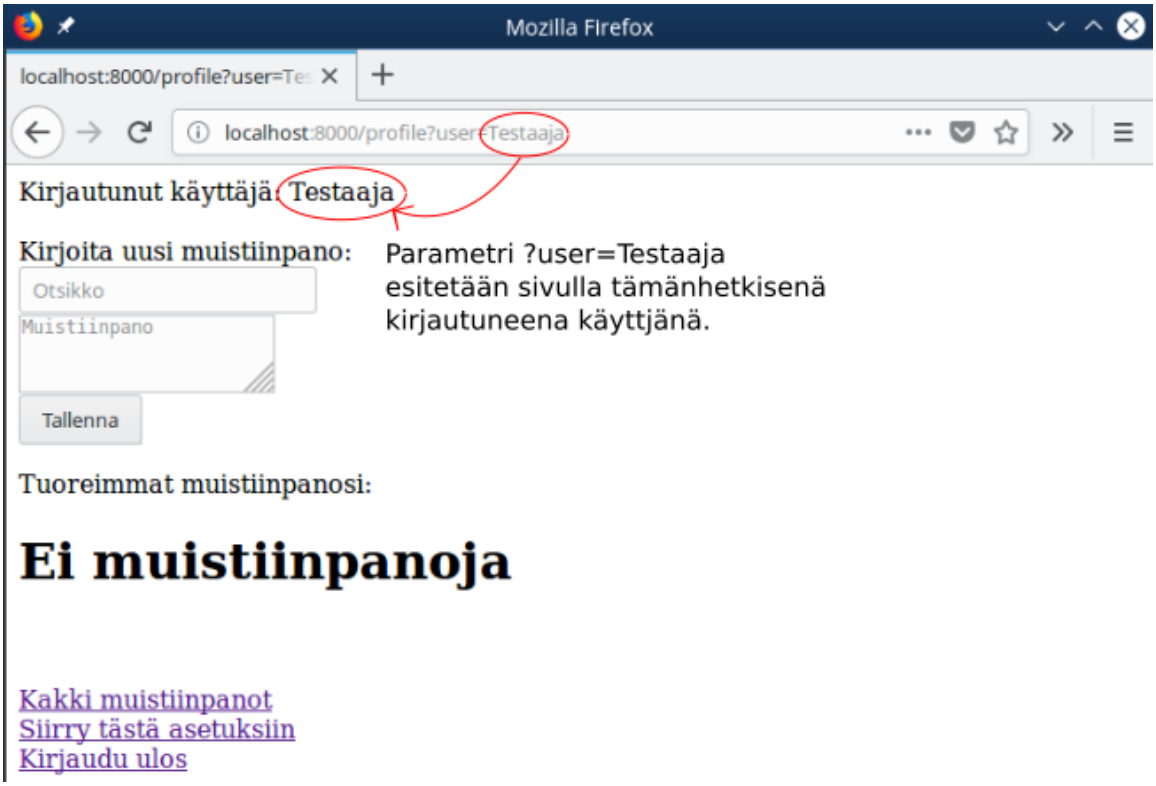
XSS-hyökkäykset voidaan jakaa kahteen kategoriaan: väliaikaiset (reflected XSS) tai pysyvät (stored XSS). Käydään seuraavaksi kolme esimerkkiä, joissa on hyödynnetty sovelluksen haavoittuvuutta JavaScriptillä tehtyihin XSS-hyökkäyksiin.

### 5.3.1 Väliaikainen XSS

Väliaikainen XSS pyrkii hyväksikäyttämään sellaisia verkkosivuja, jotka ottavat verkko-osoitteen mukana tulevia parametrejä osaksi sivun lähdekoodia. Eli käytännössä hyökkääjä joutuu käsittelemään linkkiä siten, että hän saa ujutettua ohjelmakoodin sitä kautta osaksi sivun lähdekoodia. Tällaiset hyökkäykset vaativat aina sitä, että hyökkääjä on löytänyt uhrin, joka klikkaa hyökkääjän kopeloimaa linkkiä.

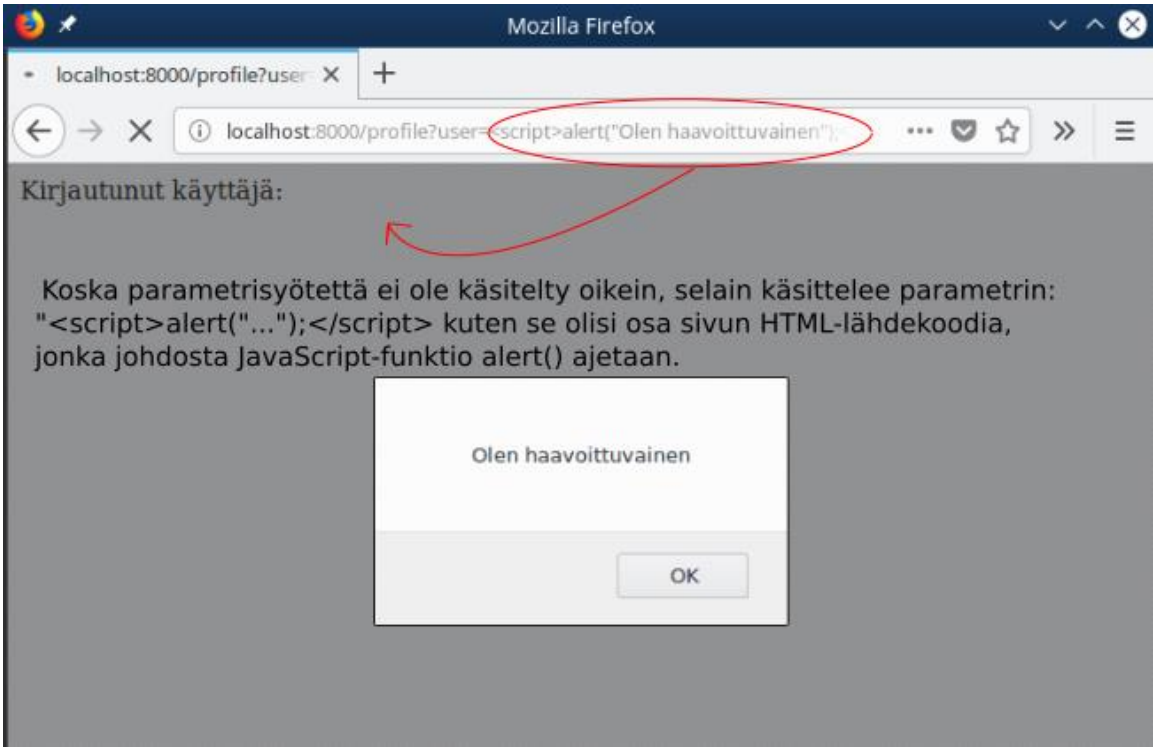
Esimerkki:

Sivusto ottaa vastaan parametrejä URL-osoitteen mukana: *http://localhost:8000/xss-test?user=Testaaja*




Kuva 19. URL-parametrin esiintyminen webohjelmassa

Hyökkääjä on todennut, että vastaanotettua parametriä ei siivota ja se on altis XSS-hyökkäykselle. Sivun alttius väliaikaiselle XSS-hyökkäykselle voidaan todentaa muun muassa siten, että parametrin tilalle kirjoitetaan pätkä JavaScriptiä, esimerkiksi: `<script>alert("Olen haavoittuvainen");</script>`. Tämä tarkoittaa käytännössä sitä, että mikäli ohjelmaa ei ole suojattu tällaista hyökkäystä vastaan, selaimen aloittaessa kirjoittamaan käyttäjänimen kohdalle tätä parametria, huomaa se siinä käytetyn ”<script>” tagin. Se kertoo selaimelle, että seuraava syöte, joka tulee ennen sulkevaa ”</script>” tagia on JavaScript-ohjelmakoodia, joka selaimen tulisi ajaa.

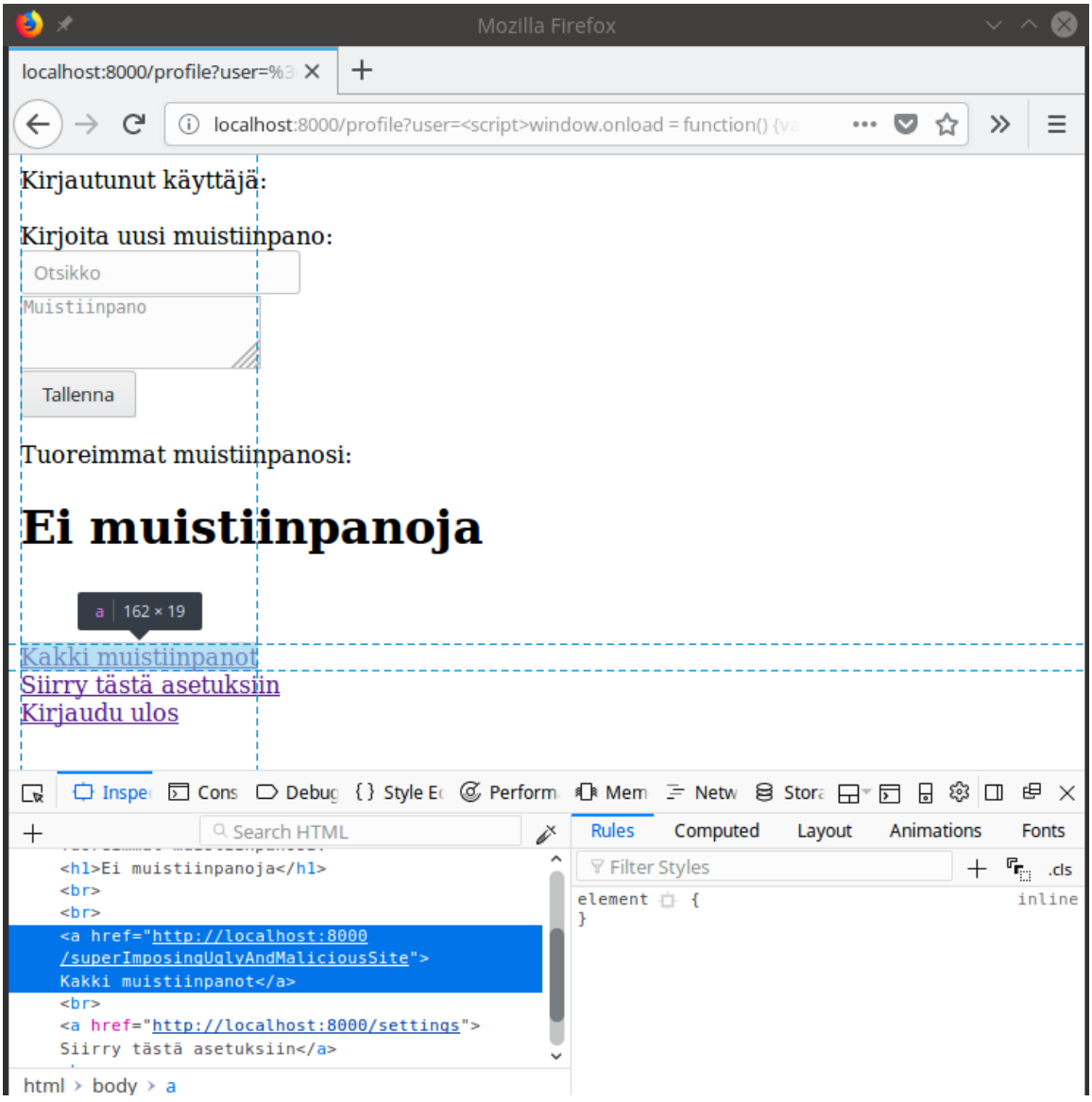



Kuva 20. XSS-haavoittuvuuden löytäminen hyödyntäen URL-parametrissä olevaa JavaScriptiä

Kun haavoittuvuus on todennettu, hyökkääjä rakentaa itselleen hyötyä tuottavan haitallisen ohjelmanpätkän ja etsii itselleen uhrin, jolle hän lähettää pahantahoisen ohjelman sisältävän linkin. Täten väliaikainen XSS-hyökkäys tarvitsee toimiakseen inhimillisen virheen. Hyökkääjä voi esimerkiksi rakentaa itsestään ensin luotettavan kuvan uhrin kanssa, vaihtamalla vaikka useita sähköposteja uhrin kanssa. Kun hyökkääjä on saanut uhrin suojauksia hieman laskettua, hän lähettää viallisen linkin, jonka uhri aukaisee. Linkki voi olla esimerkiksi seuraavanlainen:

```
http://localhost:8000/xsstest?user=<script>window.onload = func-
tion()      {var      AllLinks=document.getElementsByTagName("a");
AllLinks[0].href      =      "http://localhost:8000/xsstest-mali-
cious"}</script>
```

Linkissä oleva ohjelmapalanen muuttaa ensimmäisen sivulla olevan linkin hyökkääjän haluamaksi, säilyttäen kuitenkin alkuperäisen ulkomuodon.



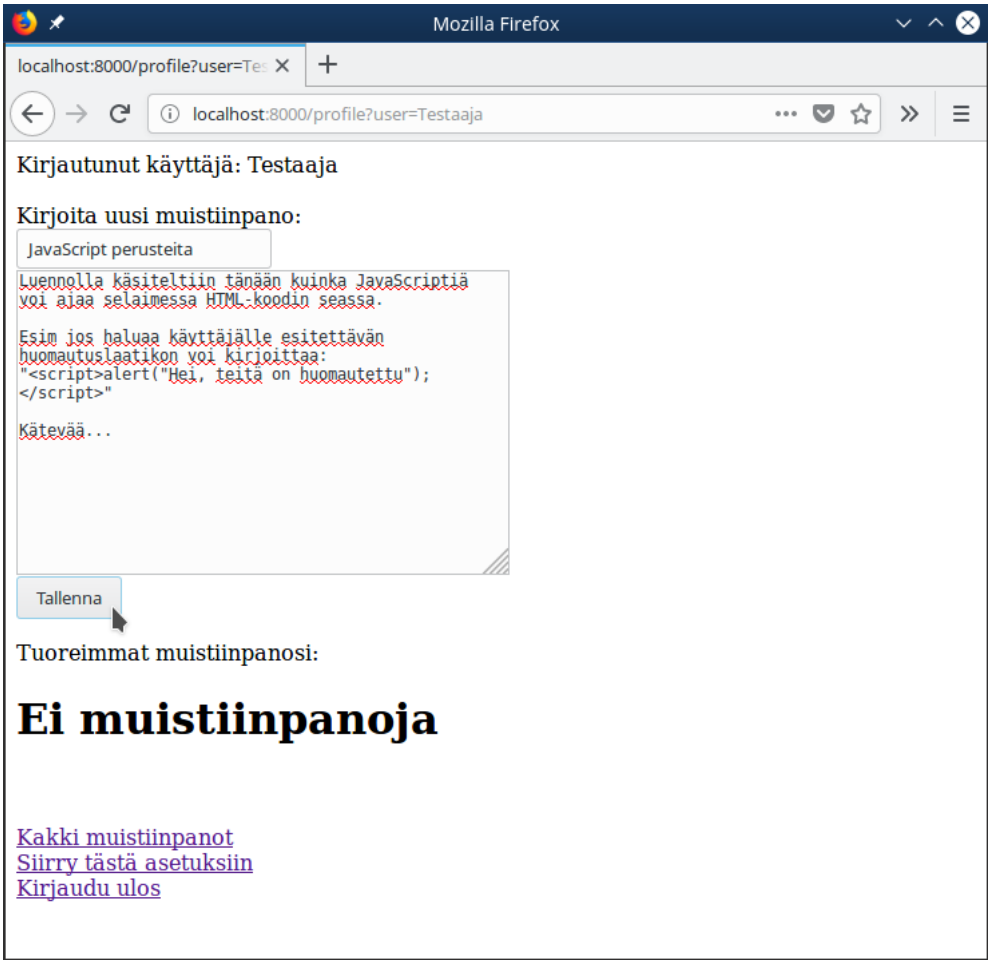

Kuva 21. XSS-hyökkäyksen johdosta muutettu linkin osoite

### 5.3.2 Pysyvä XSS

Pysyvä XSS:n toimintaperiaate on lähtökohtaisesti hyvin samanlainen kuin väliaikaisen XSS:n. Hyökkääjän pyrkimyksenä on siis saada ajettua sivustoon kuulumatonta ohjelmakoodia. Termi pysyvä tulee siitä, että tässä tapauksessa hyökkääjä on saanut vietyä pahantahtoista syötettä ohjelman tietokantaan. Kun ohjelmisto hakee tietokannasta viallisen tietueen, jotta voi näyttää sen sivuston käyttäjälle, selain ajaakin hyökkääjän ohjelmakoodin.

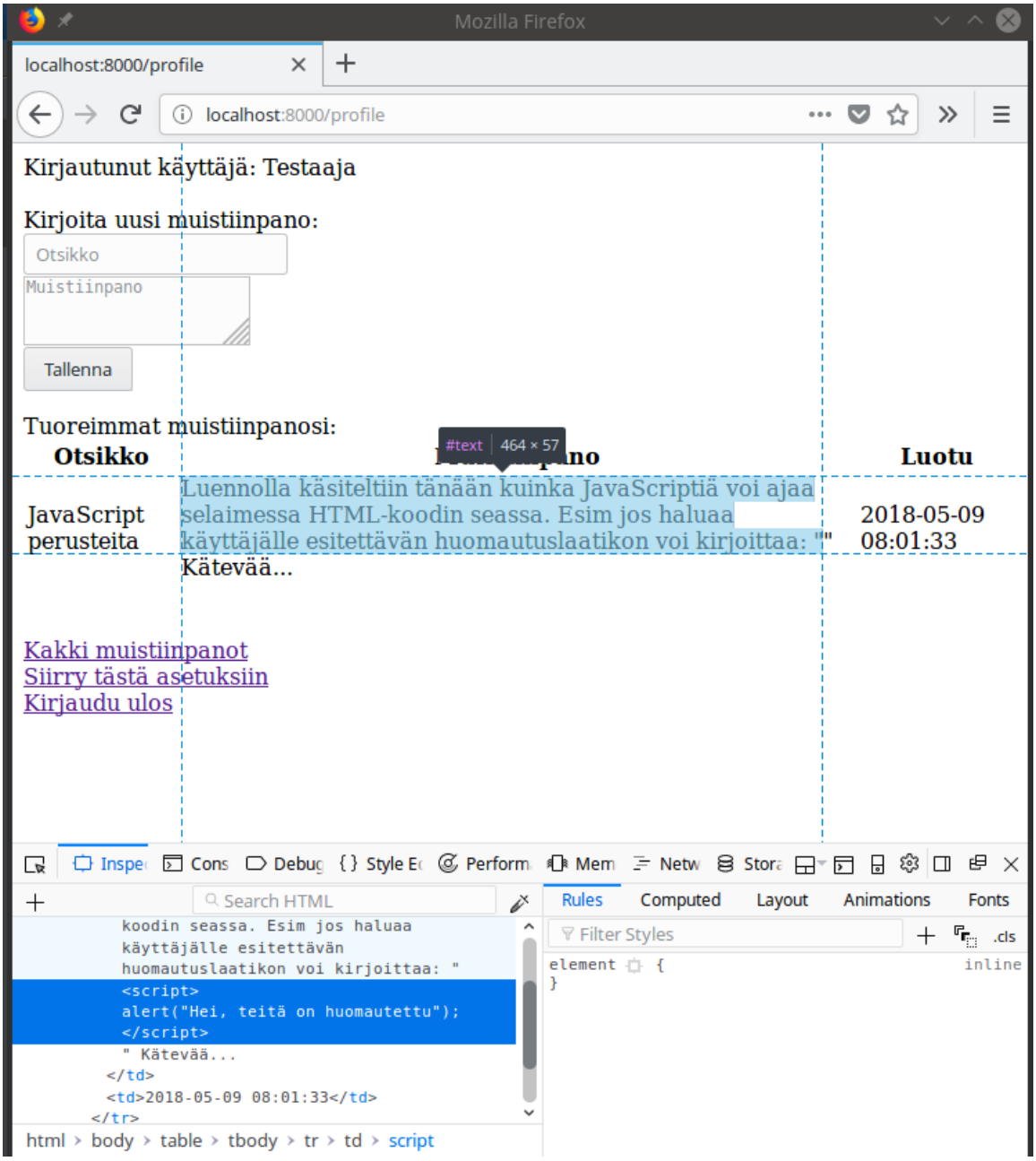
Esimerkki:

Käyttäjä voi syöttää sivustolle muistiinpanoja. Muistiinpanojen tietue on mielivaltainen merkkijono, eikä sitä käsitellä millään tavalla. Käyttäjäsyöte menee sellaisenaan tietokantaan. Käyttäjä syöttää muistiinpanoonsa virheellistä dataa.




Kuva 22. Esimerkki, kuinka pysyvä XSS-hyökkäys voi toteutua

Kun käyttäjä haluaa nähdä muistiinpanonsa myöhemmin ja se haetaan tietokannasta, ajetaan siihen kirjoitettu ohjelmanpätkä aivan kuten kohdan 5.3.1 esimerkissä. Erona kuitenkin se, että tämä ohjelmakoodi ajetaan nyt jokaisella kerralla, kun käyttäjä haluaa nähdä nimenomaisen muistiinpanon.




Kuva 23. Kuvassa 22 annettu JavaScript on osana HTML-koodia eikä täten näy osana muistiinpanoa. Sivua ladatessa tulee samanlainen huomautusikkuna kuin kuvassa 20.

Esimerkin mukainen tilanne ei todennäköisesti tuota haittaa muuta kuin muistiinpanosovelluksen käyttäjälle. Huomattavasti vakavampi tilanne olisi, jos palvelun ylläpitotyökaluissa, joissa käsitel-

lään käyttäjäsyötettä, olisi tällainen haavoittuvuus. Esimerkiksi käyttäjien palaute- tai tukipyyntölomakkeiden esittelysivuilla. Tällöin hyökkääjät voisivat suoraan viestiä ylläpidolle pahantahtoisia ohjelmakoodeja hankaloittaen heidän työtään ja pahimmassa tapauksessa kaappaamalla ylläpitotunnukset itselleen.

### 5.3.3 DOM-pohjainen XSS

DOM-pohjainen XSS on myös väliaikainen XSS-hyökkäys, mutta HTML-koodin sijaan se pyrkii hyökkäämään sivulla jo valmiiksi olevaan JavaScript-koodiin. DOM eli Document Object Model on käytäntö, kuinka esimerkiksi HTML-sivun objekteja voidaan käsitellä. Jokaisella HTML-dokumentilla on siihen liittyvä DOM, joka sisältää dokumenttiin sisältyvän tiedon selaimen näkökulmasta. Kun selaimessa ajetaan skriptejä, antaa selain sille pääsyn HTML:n DOM:iin antaen skriptille tietoa sivun tietueista ja niiden arvoista. Täten skriptit pystyvät muokkaamaan sivun ominaisuuksia siten, että selaimen ei tarvitse keskustella palvelimen kanssa, vaan sivun muokkaukset tapahtuvat ainoastaan käyttäjän selaimessa. [21]

Esimerkki:

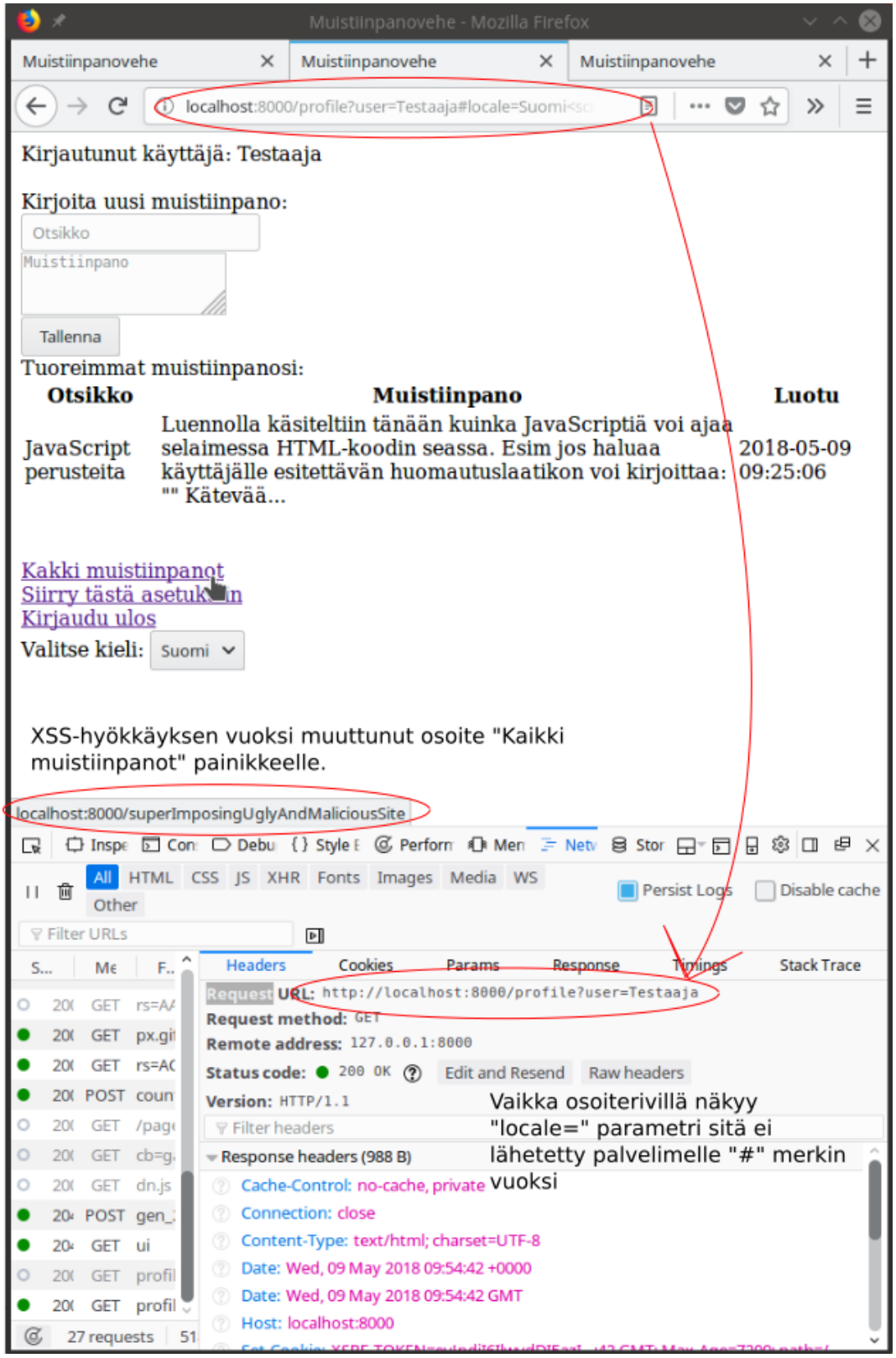
Sivustolla on JavaScriptillä luotu kielenvalintapudotuslaatikko, jonka oletusarvo saadaan osoiterivin parametristä "locale":

```
 http://localhost:8000/profile?user=Testaaja#locale=Suomi.

<select>

    <script>

          document.write(decodeURI("<OPTION      value=1>"+docu-
ment.location.href.substring(document.location.href.indexOf("lo-
cale=")+7)+"</OPTION>"));

         document.write("<OPTION value=2>English</OPTION>");

     </script>

</select>
```

Huomioitavaa tässä osoitteessa on, että "#" merkin jälkeistä tekstiä ei oteta mukaan http-kutsuun, eli sitä ei lähetetä palvelimelle. Mikäli hyökkäys tehdään näin, palvelun ylläpitäjän lokitietoihin ei jää mitään tietoa, että kyseinen hyökkäys on tapahtunut. Hyökkääjä toimii samoin kuin väliaikaisessa hyökkäyksessä ja luo haitallisen linkin:

```
http://localhost:8000/profile?user=Testaaja#lo-
cale=Suomi%3Cscript%3Ewindow.onload%20=%20func-
tion()%20{var%20AllLinks=document.getElementsByTag-
Name(%22a%22);%20AllLinks[0].href%20=%20%22http://local-
host:8000/xsstest-malicious%22}%3C/script%3E.
```

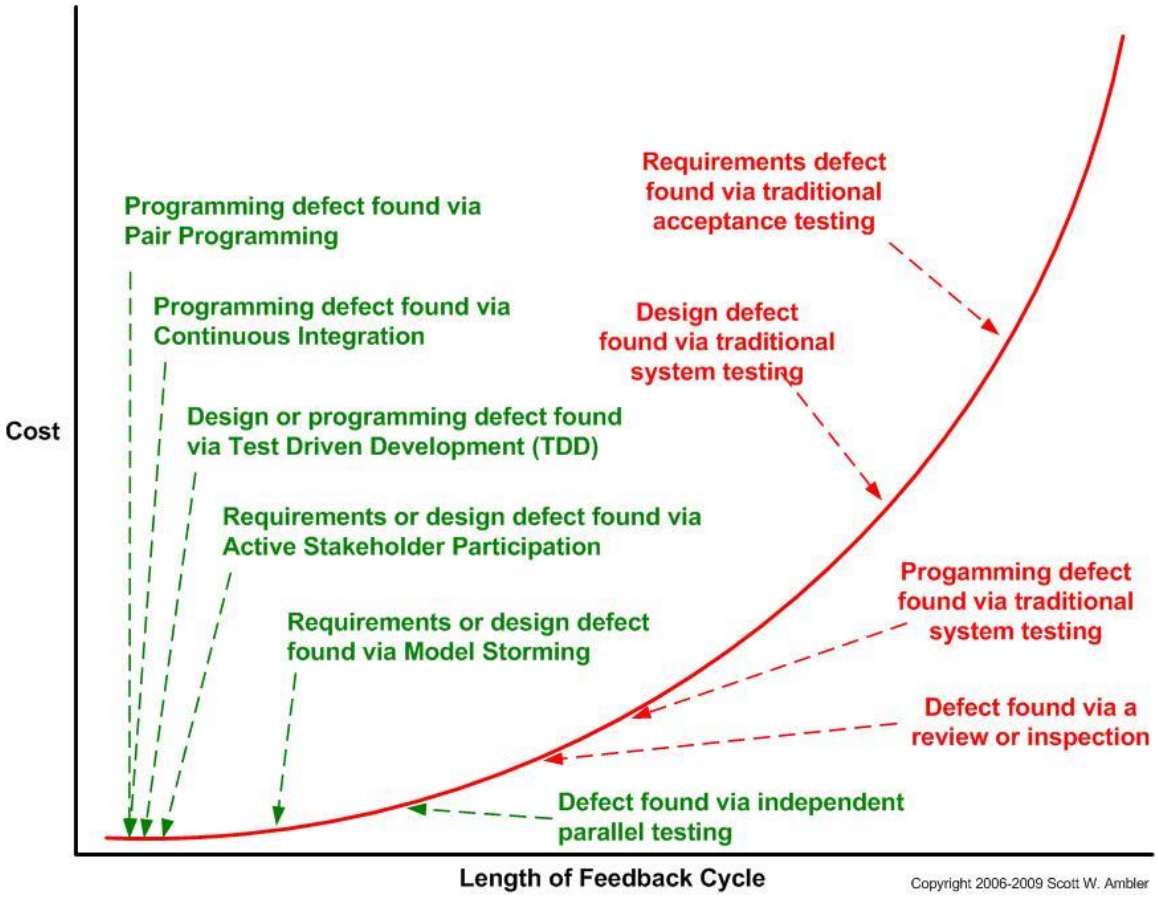
Tämä linkki toimii samoin kuten kohdassa 5.3.1, eli se muuttaa ensimmäisen löytämänsä linkin osoittamaan haittasivustoon.




Kuva 24. Esimerkki siitä, kuinka URL-osoitteessa ”#”-merkin jälkeen tulevaa tietoa ei lähetetä palvelimelle

# 6 OWASP ZAP:n käyttö sovelluskehityksen yhteydessä

Virheiden korjaamiseen kuluvat resurssit kasvavat sitä mukaa mitä pidemmällä sovelluksen elinkaarta mennään. Jos mahdolliset virhe- ja ongelmatilanteet voidaan havaita jo suunnitteluvaiheessa ja muokata suunnitelmaa, virheiden korjaamiseen menee huomattavasti vähemmän aikaa, sillä mitään ohjelmointityötä ei ole vielä tehty. Mikäli vajavaisesti suunniteltua sovellusta aletaan tekemään ja ohjelmisto koodataan ilman kehittäjien testausta eikä laadunvarmistus ole saanut kehittäjiltä tarpeeksi tietoa mitä sovelluksen pitäisi tehdä, ilmenee ongelmia varmasti. Kun sovellusta ylläpidettäessä huomataan, että jokin on perustavanlaatuisesti pielessä, on sovellusta hankala alkaa korjaamaan, kun samalla täytyy pitää huoli siitä, että sovelluksen käyttäjät kykenevät käyttämään toimintoja ilman suurempia katkoja.



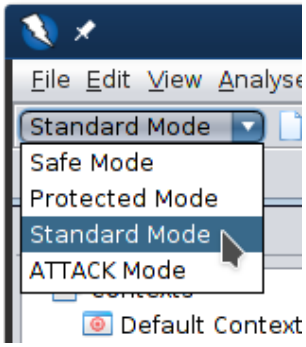

Kuva 25. Virheen korjauksen kulujen kasvu palautesyklin pituuden suhteen [22]

Kuten kuvasta huomataan, että mitä pidemmälle sovelluskehityksen prosessi on edennyt, sitä korkeammalle korjauksen kulut kasvavat. Tämän vuoksi on erityisen tärkeää, että kehittäjät tes-

taavat omaa lähdekoodiaan mahdollisimman huolellisesti. Näin kehittäjän palautesykli pysyy nopeana, sillä hänen ei tarvitse käydä jokaista virhettä tai epäloogisuutta testaajan kanssa erikseen läpi. ZAP on tällaiseen kehittäjän testaustyöhön erittäin hyvä apukeino. Testaajaan verrattuna kehittäjällä on se etu, että kehittäjällä on suurempi tieto siitä, mikä ohjelmakoodissa voi mennä pieleen. Koska kehittäjä näkee muutakin kuin käyttöliittymän, on hänen hyvä suorittaa penetraatiotestausta omaan ohjelmakoodiinsa. Penetraatiotestauksen tarkoituksena on varmistua sovelluksen tietoturvasta ja löytää mahdollisia tietoturva-aukkoja. Penetraatiotestauksen ohessa löytyy monesti myös aivan selviä logiikkavirheitä, kun kehittäjä joutuu miettimään koodiansa eri näkökulmasta.

## 6.1 ZAP:n skannerit

ZAP:ssa on sisäänrakennettuja skannereita, jotka tutkivat automaattisesti verkkosivuston sisältöä. Passiiviskanneri on oletuksena päällä, ja se kerää kutsuja sitä mukaa, kun käyttäjä selailee sivua. Tulokset näkyvät *”History”*-välilehdellä. Spider-skanneri käy läpi sivustolla olevat linkit ja rakentaa niistä sivukartan. Aktiiviskanneri käy myös linkit läpi, mutta toisin kuin spider, se lähettää näiden linkkien osoitteisiin tietoa tarkoituksena löytää virheitä. Näiden skannereiden ajaminen on hyvä ensimmäinen askel sovelluksen tietoturvatestauksessa. Jotta skannereita voidaan käyttää, pitää ZAP:n vasemmasta yläkulmasta valita "*Standard Mode*".



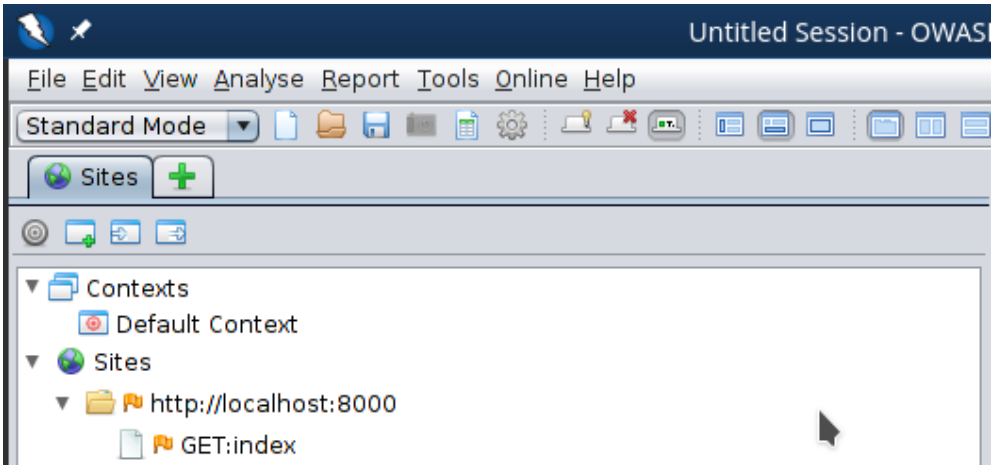

Kuva 26. ZAP:n moodin asettaminen ”Standard Mode”

### 6.1.1 Spider

Spider-skannaus käy verkkosivun HTML-koodin läpi ja seuraa linkkejä, HTML:n sisään upotettuja JavaScript-lähteitä ja CSS-tiedostoja, eli käytännössä se pyrkii löytämään kaikki lähteet, joista

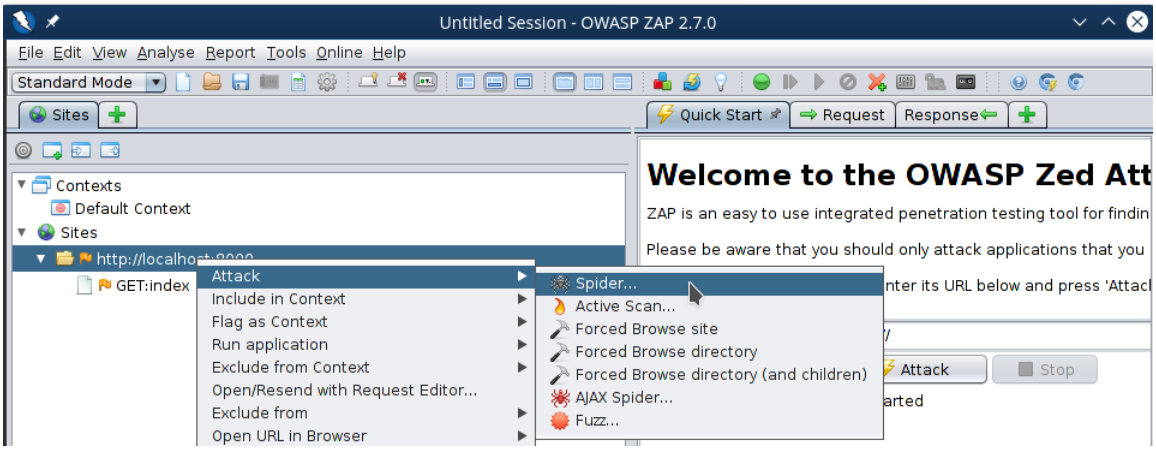
verkkosivu pystyy keskustelemaan palvelimelleen tai muille palvelimille. Kun skannaus on tehty, ZAP rakentaa löydöksistään sivukartan.

Esimerkki spider-skannauksesta. Kun haluttu sovellus on selaimessa auki ja selaimen liikenne on ohjattu ZAP:iin, ZAP rakentaa jo automaattisesti sivukarttaa niistä sivuista, joissa käyttäjä on jo käynyt.




Kuva 27. Sivukartta ZAP:ssa

Kun sivu halutaan spider-skannata, valitaan sivukartasta kyseisen sivun osoite ja avataan aliva-likko oikealla hiiren klikkauksella.

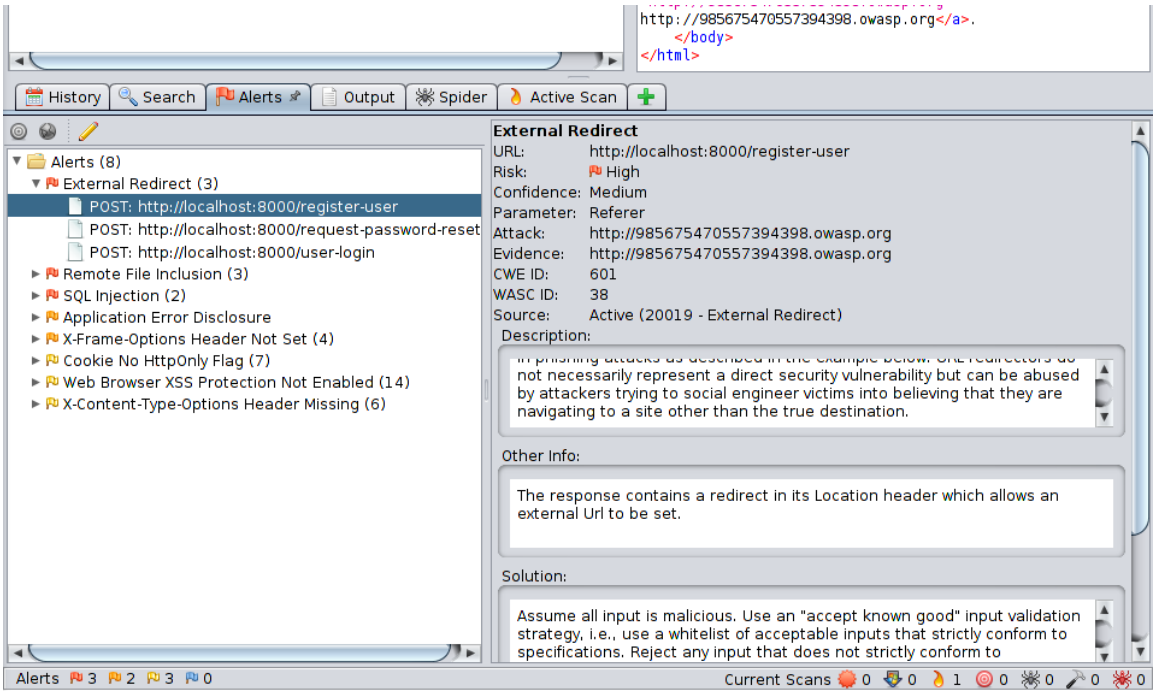



Kuva 28. Spider-skannaustyökalun sijainti ZAP:n ohjelmavalikossa

Avautuvasta ikkunasta painetaan "*Start scan*". Sivukartta kasvaa sitä mukaa kun skanneri löytää linkkejä sivustolta. Tämän skannerin avulla voidaan löytää linkkejä sivuille, jotka ovat voineet unohtua sivulle esimerkiksi piilotettuina elementteinä.

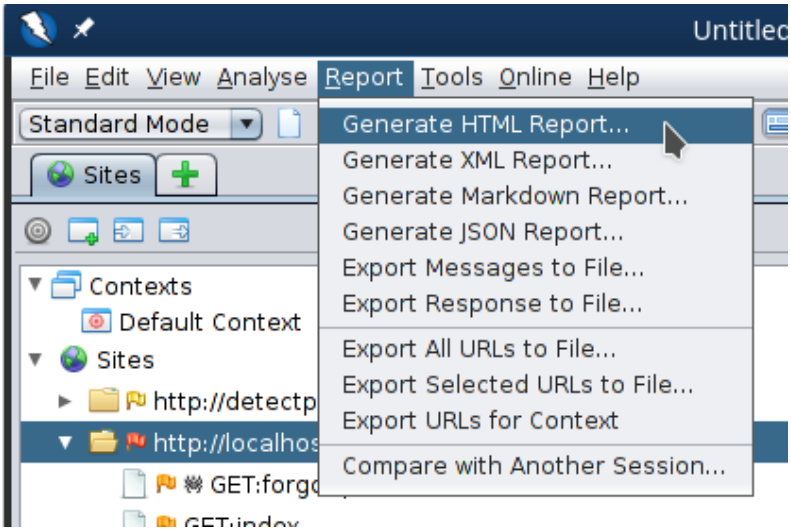
### 6.1.2 Aktiiviskanneri (Active scan)

Aktiiviskanneri toimii hyvin samaan tapaan kuin spider-skanneri, mutta se tekee kaikkiin löytämiinsä linkkeihin kutsuja ja seuraa uudelleenohjauksia. Aktiiviskanneri on huomattavasti tehokkaampi työkalu virheiden löytämisessä kuin spider-skanneri, sillä se fussaa (ks kappale 6.3) löytämiensä kutsujen parametrejä. Jos applikaatiossa on lokitiedosto, johon ohjelma kirjoittaa virheistään, on sitä hyvä pitää auki esimerkiksi Linuxin komennolla "*tail -f kansio/loki.txt*", jolloin nähdään reaaliajassa lokitiedostoon tulevat virhetilanteet. Aktiiviskanneri ajetaan samaan tapaan kuin spider-skanneri, mutta tällä kertaa "*Attack...*" alivalikosta valitaan "*Active scan...*". Avautuvasta ikkunasta painetaan "*Scan*". Kun skannaus on valmis, ZAP:n "*Alerts*"-välilehdeltä voi tarkastella millaisia tuloksia skannaus tuotti.




Kuva 29. ZAP:n aktiiviskannerin löytämiä hälytyksiä

Jokaisessa hälytyksessä näkyy vasemmassa reunassa pieni lippu, ja mitä punaisempi lippu on, sitä vaarallisempana ZAP tätä hälytystä pitää. Hälytyksistä voi luoda raportin, jotta löydöksiä on helpompi jakaa ja tulkita.




Kuva 30. Aktiiviskannerin löydöksien vieminen HTML-raportiksi

Raportista ilmenee millainen kutsu aiheutti hälytyksen, kuvauksen virheestä, mahdollisen korjausehdotuksen sekä lisätietoja aiheesta. Raportin virheet kannattaa käydä yksitellen läpi manuaalisesti testaamalla, siten että pyrkii toistamaan virheen selaimesta ja sitten eliminoimaan virheen.

## 6.2 Lunttauslistat (cheat sheets)

Skannereiden aiheuttamien hälytysten jatkotutkimuksissa kannattaa hyödyntää erilaisia tietoturvaan ja testaukseen liittyviä lunttauslistoja, joita muun muassa OWASP tarjoaa (https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series). Näihin listoihin on kerätty kategorioittain asioita, mitä kannattaa ottaa kehitysvaiheessa huomioon ja miten niitä voi jälkeenpäin testata. Lunttilistat pyrkivät esittelemään niin sanottuja parhaita toimintatapoja (best practices) kustakin kategoriasta. Ohjelmaa suunniteltaessa on kuitenkin otettava huomioon, mitkä riskit ovat hyväksyttäviä, koska harvoin pystytään täyttämään kaikkia hyviä toimintatapoja. Lunttilistat ovat siltäkin kantilta hyviä, että ihmisillä voi olla vaihteleva muistikyky ja lunttauslistoja läpi käydessä voi löytyä sellaisia huomioita, jotka ovat yksinkertaisesti unohtuneet. Esimerkiksi miten käyttäjän tunnistautumisessa kannattaa huomioida käyttäjätunnusten hallinta? Millainen salasanan tulee olla? Salasanan olisi hyvä olla pitkä, tarpeeksi monimutkainen arvattavaksi ja yleisesti käytettyjen salasanojen käyttö voisi olla hyvä kieltää. [23] Käyttäjäystävällisyyden nimissä olisi myös hyvä, että käyttäjälle näytetään, mitkä ovat kriteerit hyvälle salasanalle.

## 6.3 Parametrien fussaus (Fuzzing)

Kun verkossa on näkyvillä osoite, joka ottaa vastaan käyttäjäsyötettä, on tällöin oltava valmis ottamaan vastaan ihan mitä tahansa. Nyrkkisääntö on, että älä koskaan luota käyttäjäsyötteeseen. Monesti käyttäjät saattavat aivan vahingossa antaa sovelluksen ajon kannalta haitallista dataa, joka voi aiheuttaa ohjelman kaatumisen. Kuitenkin suurempi uhka on vihamielisten internetin käyttäjien roskadata ja sen satunnainen syöttäminen (fuzzing, suom. fussaus). Koska muun muassa merkistöjä, ohjelmasyntakseja, hyökkäysmetodeja ja injektiokikkoja on loputon määrä, voisi tällaisen roskadatan manuaalitestausta suorittaa loputtomiin. Tästä esitettiin esimerkki aiemmin kohdassa 4.6: OWASP BWA:n Mutillidae II -ohjelman sisäänkirjautumisen murtamisessa.

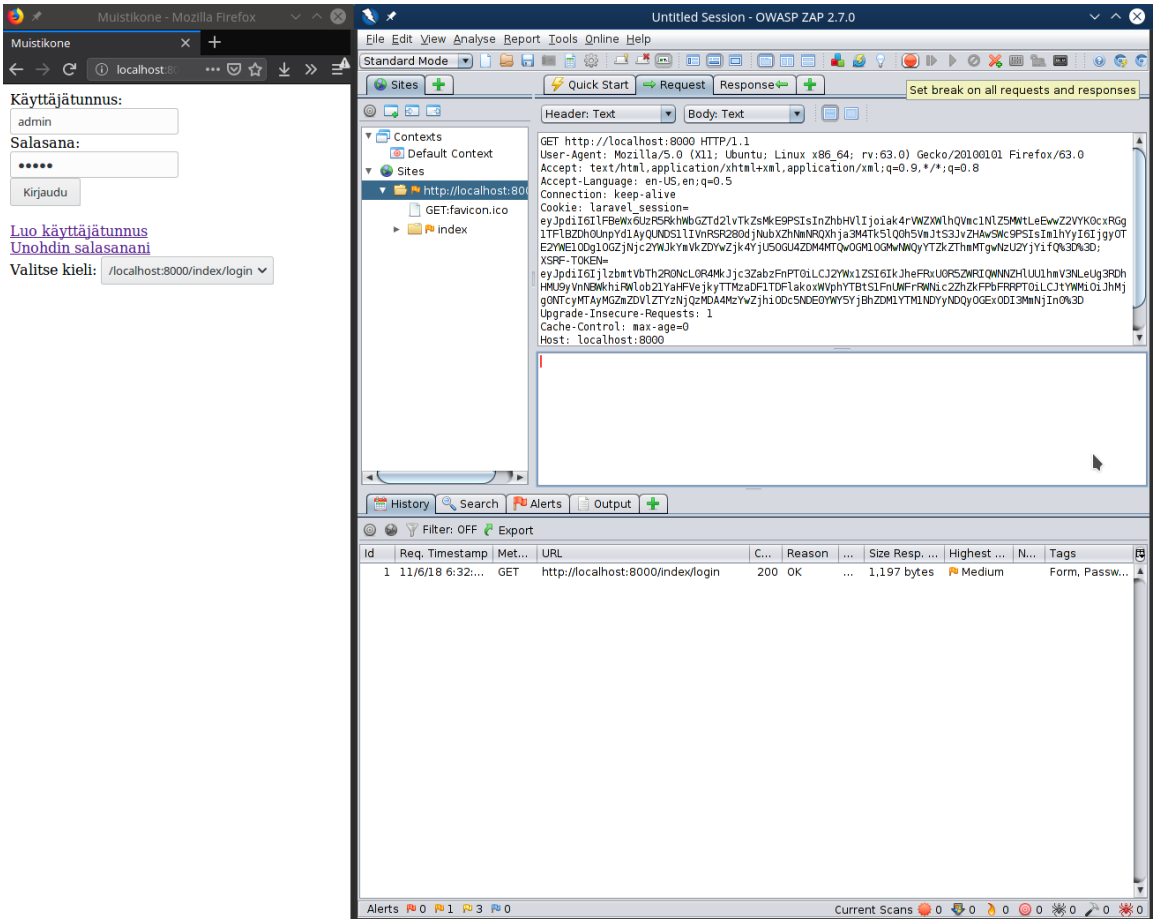
Fussaamalla voidaan löytää tilanteita, jolloin ohjelma ei voi ymmärtää, mitä pitäisi tehdä: "*To invoke the hive-mind representing*":llä kun se odottaa saavansa numeron yhden ja kymmenen väliltä. Sovellusta suunniteltaessa voi miettiä, millainen data on hyväksyttävää missäkin tilanteessa, ja tehdä niin sanottua valkolistausta. Valkolistaus tarkoittaa, että jätetään huomioimatta kaikki sellainen data, mikä ei tue toiminnallisuuden tarkoitusta. Kun käyttäjäsyötettä testataan fussaamalla, kannattaa testata jokainen parametri, parametrin nimi sekä lisätä kutsuun uusia parametrejä ja fussata niiden arvoja ja nimiä. Fussausta voi ja kannattaa tehdä mihin osaan tahansa http-kutsua.

Suorien sovellusvirheiden lisäksi fussaamalla voidaan löytää sellaisia kohtia, joista hyökkääjät voivat tehdä rajattomasti pyyntöjä palvelimelle samalla tukkien palvelimelle menevän liikenteen. Tällaisia hyökkäyksiä kutsutaan palvelunestohyökkäyksiksi, joiden tavoitteena on estää sovelluksen käyttö ns. oikeilta käyttäjiltä. Kehittäjän huomatessa, että tietystä pisteestä voi tehdä rajattoman määrän pyyntöjä, voi hän luoda sinne esimerkiksi IP-osoitteen rajoituksen, jolloin yhdestä IP-osoitteesta sallitaan esimerkiksi 50 kutsua tunnissa. Tällöin yksi hyökkääjä ei voi tehdä palvelinestohyökkäystä yksin vaan tarvitsee avukseen bottiverkon (botnet), joka koostuu kaapatuista tietokoneista, joilla on eri IP-osoitteet.

## 6.4 Pysäytyspiste (Breakpoint)

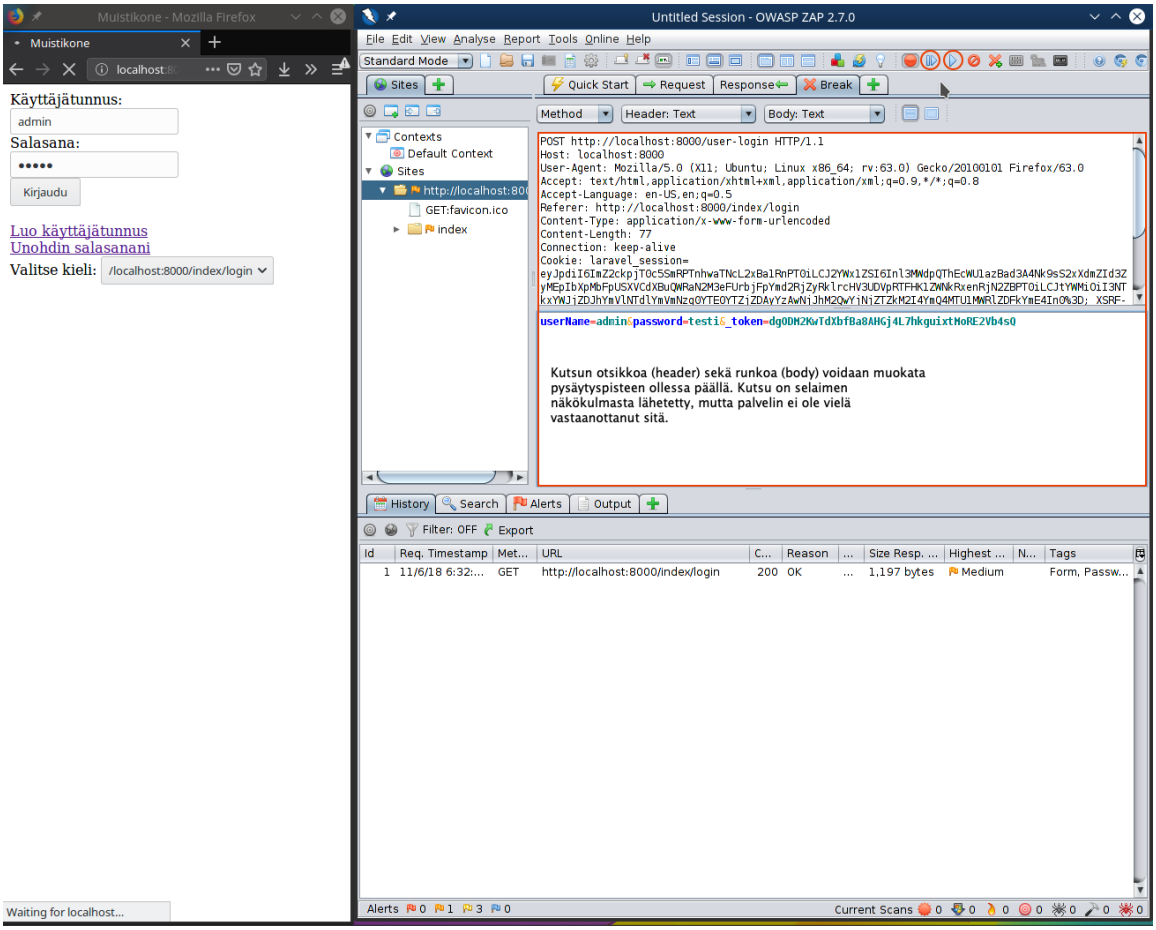
Mikäli ZAP:lla halutaan toistaa kuvan 3 tilanne, täytyy tällöin käyttää apunaan pysäytyspistettä. Kun selain lähettää pyynnön palvelimelle, se ei voi tietää, kuinka kauan vastauksen saamiseen

menee, jolloin selain jää viestin lähettämisen jälkeen rauhassa odottamaan vastausta. Tämä toiminnallisuus mahdollistaa sen, että ZAP:n avulla voidaan selaimen ja palvelimen välistä viestintää tutkia reaaliajassa. Kun ZAP on asetettu selaimen välityspalvelimeksi, voi käyttäjä estää viestin kulkemisen eteenpäin asettamalla pysäytyspisteen.




Kuva 31. Pysäytyspisteen päälle kytkeminen. Ympyräsymboli muuttuu vihreästä punaiseksi pysähdyspisteen ollessa päällä

Kun pysäytyspiste on asetettu päälle, se estää seuraavan kutsun automaattisen etenemisen.




Kuva 32 Http-kutsu pysähtyneenä ZAP:ssa

Kuvassa 32 näkyy, kuinka selaimelta lähetetty palvelun sisäänkirjautumislomake on pysähtynyt ZAP:iin. Kun kutsu on tässä tilassa, voi ZAP:n käyttäjä muokata viestiä ennen kuin lähettää sen eteenpäin selaimelle. Kuvassa 32 ympyröitynä vasemmalla oleva play/pause-painike päästää tällä hetkellä pysäytetyn kutsun eteenpäin, ja pysäyttää seuraavan ZAP:iin saapuvan kutsun tai vastauksen. Kun taas oikealla ympyröity play-painike vapauttaa pysäytetyn kutsun eteenpäin, eikä pysäytä jatkossa ZAP:n läpi kulkevia viestejä. Pysäytyspistettä hyödyntämällä kehittäjä voi tutkia tarkasti sitä, miten selaimen sovellus keskustelee palvelinsovelluksen kanssa. Kuten huomataan, http-kutsun muokkaaminen on suhteellisen helppoa sen jälkeen, kun selain lähettää kutsun. Tästä syystä selaimessa tehtyyn syötteen validointiin ei voi täysin luottaa, ja kaikki syöte tulee validoida myös palvelimella, jolloin voidaan olla täysin varmoja siitä, että data ei enää muutu.

# 7 Yhteenveto

Opinnäytetyön tavoitteena oli tutkia OWASP ZAP:n käyttöä ja sen hyödyntämistä sovelluskehityksen tukena ja kirjata löydökset ja havainnot. Jotta ZAP:n käytön hyödyt voi hahmottaa, tulee tietää syyt, miksi sitä kannattaa käyttää. Tätä tietoa tukemaan täytyi opiskella tietoturvan perusteita, käsitteitä sekä erilaisia hyökkäystapoja. Tietoturva on suuri kokonaisuus ja hyvä tietoturvan taso on kaikkien asianosaisten vastuulla.

Työn lopputuloksena on opas, jonka avulla tietoturvasta ja tietoturvatestauksesta kiinnostunut henkilö pääsee alkuun tietoturvatestauksen saralla. Opas on sopiva sellaisille sovelluskehittäjille ja -testaajille, joilla on kokemusta websovellusten parissa työskentelystä. Työssä käytetyt esimerkit ovat yleispäteviä, eivätkä ne ole riippuvaisia back end -teknologioista.

ZAP on tehokas työkalu, jota käytettäessä oppii paljon siitä, kuinka verkkosovellukset toimivat. Sen käyttäminen on kuitenkin hieman haastavaa kehnon käyttöliittymän takia, ja monet ominaisuudet on kätketty epäloogisesti monien painikkeiden taakse. Jatkotutkimuksena tähän ongelmaan ratkaisuksi voisi olla ZAP:n API:n (application programming interface) tutkiminen. Tämän API:n avulla ZAP:n toimintoja voisi kutsua vaikka Python- tai Ruby-kielillä. Myös API:n automaattinen tietoturvatestaus esimerkiksi Jenkins-automaatioalustan avulla voisi olla mahdollista.

Vaikka tämä opinnäytetyö keskittyy OWASP ZAP:iin, ei se ole ainoa tietoturvatestaustyökalu. Muita samankaltaisia työkaluja ovat esimerkiksi BurpSuite, Vega ja Arachni. [24]

Tietoturva koskee jokaista ihmistä. Kaikkien Internetin käyttäjien käyttäytymistä seurataan ja käyttäytymismallien ymmärtäminen on suuri bisnes. Esimerkiksi Google ja Facebook suorittavat jatkuvasti profilointia maksimoidakseen mainostulot kohdentamalla mainoksia mahdollisimman tarkasti. Ihmisten käyttäytymisestä Internetissä on tullut tuote, jota voidaan myydä eri tahoille, jotta ne voivat löytää kohdeyleisönsä mahdollisimman tehokkaasti. Tämä ei välttämättä ole kuluttajan kannalta huono asia, sillä he saavat aina sellaista sisältöä mikä heille sopii. Hinta osuvasta sisällöstä on kuitenkin yksityisyyden heikkeneminen, sillä jokainen liike verkossa on haluttavaa tietoa. Nykypäivänä ei enää voi oikein olla varma siitä, että kuka tietää sinusta ja kuinka paljon. Oman henkilökohtaisen tiedon varjeleminen tulisi olla jokaiselle tärkeää.

Lähteet


1 Office for National Statistics: Internet access – households and individuals, Great Britain: 2018 [WWW-julkaisu] <https://www.ons.gov.uk/peoplepopulationandcommunity/householdcharacteristics/homeinternetandsocialmediausage/bulletins/internetaccesshouseholdsandindividuals/2018> viitattu 10.11.2018

2 Netradar: Most used mobile applications [WWW-julkaisu] <http://www.netradar.com/most-used-mobile-applications-in-the-finnish-market/> viitattu 10.11.

3 Finlex: Tieto- ja viestintärikoksista. [WWW-julkaisu] <https://www.finlex.fi/fi/laki/ajantasa/1889/18890039001#L38> viitattu 30.4.2018

4 Comparitech: Analysis: How data breaches affect stock market share prices (2018 update) [WWW-julkaisu] <https://www.comparitech.com/blog/information-security/data-breach-share-price-2018/> viitattu 28.10.2018

5 European Comission: Data protection – Better rules for small business [WWW-julkaisu] <https://ec.europa.eu/justice/smedataprotect/index_en.htm> viitattu 28.10.2018

6 EU GDPR: GDPR Key Changes [WWW-julkaisu] <https://eugdpr.org/the-regulation/> viitattu 28.10.2018

7 Mozilla Developer Network: Confidentiality, Integrity and Availability [WWW-julkaisu] <https://developer.mozilla.org/en-US/docs/Web/Security/Information_Sec00urity_Basics/Confidentiality,_Integrity,_and_Availability> viitattu 30.4.2018

8 Aligned Energy: Security in the Data Center and Beyond – A Layered Security Approach [WWW-julkaisu] <https://www.alignedenergy.com/post/security-in-the-data-center-and-beyond-a-layered-security-approach/> viitattu 14.5.2018

9 Open Wed Application Security Project: About OWASP [WWW-julkaisu] <https://www.owasp.org/> viitattu 1.5.2018

10 OWASP Dependency Check [WWW-julkaisu] <https://www.owasp.org/index.php/OWASP_Dependency_Check> viitattu 26.10.2018

11 GitHub: zaproxy downloads: [WWW-julkaisu] <https://github.com/zaproxy/zaproxy/wiki/Downloads> viitattu 8.11.2018

12 codementor: Setting Up OWASP-BWA With VirtualBox [WWW-julkaisu] <https://www.codementor.io/jayadityagupta/setting-up-owasp-bwa-with-virtualbox-a1wgz26vq> viitattu 3.5.2018

13 PCsteps: How To Install VirtualBox in Linux Mint / Ubuntu Linux [WWW-julkaisu] <https://www.pcsteps.com/184-install-virtualbox-linux-mint-ubuntu/> viitattu 3.5.2018

14 OWASP Broken Web Applications Project [WWW-julkaisu] <https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project#tab=Main> viitattu 8.11.2018

15 Open Web Application Security Project: OWASP Top 10, 2017 Introduction [WWW-julkaisu] <https://www.owasp.org/index.php/Top_10-2017_Introduction> viitattu 5.5.2018

16 Chris Snyder, Michael Southwell. Pro PHP Security. 2005

17 Reuters: Yahoo says one billion accounts exposed in newly discovered security breach [WWW-julkaisu] <https://www.reuters.com/article/us-yahoo-cyber-idUSKBN1432WZ> viitattu 5.5.2018

18 The Guardian: Yahoo hack: 1bn accounts compromised by biggest data breach in history [WWW-julkaisu] <https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached> viitattu 5.5.2018

19 Fossbytes: VNC Roulette: How One Hacker Is Exposing Thousands Of Hackable Computers [WWW-julkaisu] <https://fossbytes.com/vnc-roulette-hacker-exposed-hackable-computers/> viitattu 5.5.2018

20 hahasecurity: Hack millions of devices with 0 skills! [WWW-julkaisu] <http://hahasecurity.blogspot.fi/> viitattu 07.5.2018

21 Acunetix: DOM XSS: An explanation of DOM-based Cross-Site Scripting [WWW-julkaisu] <https://www.acunetix.com/blog/articles/dom-xss-explained/> viitattu 10.5.2018

22 Agile Modeling: Examining the Agile Cost of Change Curve [WWW-julkaisu] <http://www.agilemodeling.com/essays/costOfChange.htm> viitattu 10.5.2018

23 OWASP Authentication Cheat Sheet [WWW-julkaisu] <https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Implement_Proper_Password_Strength_Controls> viitattu 27.10.2018

24 OWASP: Vulnerability Scanning Tools [WWW-julkaisu] <https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools> viitattu 9.11.2018