OULUN AMMATTIKORKEAKOULU

Henrik Mourujärvi

ELFCUBEN TIETOTURVALLISUUDEN VARMISTAMINEN JA TUOTANTOON VIEMINEN

ELFCUBEN TIETOTURVALLISUUDEN VARMISTAMINEN JA TUOTANTOON VIEMINEN

Henrik Mourujärvi
Opinnäytetyö
Syksy 2020
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, laite- ja tuotesuunnittelun suuntautumisvaihtoehto

---

Tekijä: Henrik Mourujärvi
Opinnäytetyön nimi suomeksi: elfCUBEn tietoturvallisuuden varmistaminen ja tuotantoon vieminen
Opinnäytetyön nimi englanniksi: Ensuring the information security of elfCUBE and deployment to production
Työn ohjaaja: Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Syksy 2020
Sivumäärä: 23

Tämä opinnäytetyö käsittelee elfCUBEn tietoturvatestausta ja sen infrastruktuurin dokumentointia. Työn toimeksiantaja toimii oululainen elfGROUP Kyberturvallisuuspalvelut Oy, joka on tietoturvaan ja sähköisten tietojen turvalliseen hallintaan erikoistunut yritys. Työssä suoritettiin tietoturvatestaus ja infrastruktuurin dokumentointi elfGROUPin luomalle elfCUBE-palvelulle.

Työn tavoite on varmistaa elfCUBEn tuotantoturvallisuus tietoturvatestaamalla palvelu ja tehdä kattava infrastruktuurin dokumentointi. Työssä käydään läpi yleisimpiä web-sovelluksiin liittyviä haavoittuvuuksia ja testauksessa käytettyä työkalua. Lisäksi käydään läpi tietoturvatestauksessa raamina käytettyä OWASP-raporttia, jolle testaus perustuu.

Tässä työssä ei tulla paljastamaan löydettyjä tietoturvahavaintoja, sekä työn tulokseksi syntynyttä infrastruktuurin dokumenttia ei tulla jakamaan yleiseen tietoon arkaluotoisten tietojen vuoksi.

Asiasanat: pääsynvalvonta, haavoittuvuus, tietoturvatestaus

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Device and Product Design

---

Author: Henrik Mourujärvi
Title of thesis: Ensuring the information security of elfCUBE and deployment to production
Supervisor: Kari Jyrkkä
Term and year when the thesis was submitted: Fall 2020
Pages: 23

This thesis addresses the security testing of elfCUBE and the documentation of its infrastructure. The employer of this thesis is elfGROUP Cybersecurity Services Ltd which is based in Oulu. elfGROUP is a company specialising in security and the secure management of electronic data. Security testing and infrastructure documentation at this work was carried out for the elfCUBE service created by elfGROUP.

The goal of the work is to ensure the production security of elfCUBE by security testing the service and to do comprehensive infrastructure documentation. The work goes through the most common vulnerabilities associated with web applications and the tool used in testing. There is also a review of the OWASP report which was used as a framework for security testing.

Neither the security findings nor the infrastructure document created as a result of the work will be shared with the public since they contain delicate data.

Keywords: Access Control, Vulnerability, Security Testing

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on suorittaa tietoturvatestaus toimeksiantaja elfGROUPin uudelle tietoturvapoikkeavuuksien raportointipalvelulle elfCUBElle. elfCUBE on elfGROUPin rakentama verkkopalvelu, johon kirjataan tietoturvatestauksessa löydetyt haavoittuvuudet ja raportit tehdystä tietoturvatestauksesta. elfGROUPin asiakkaat voivat nettisivuilta käydä tarkistamassa heidän palvelullensa tehdystä tietoturvatestauksesta löytyneet havainnot ja raportit. Työn tavoitteena on varmistaa elfCUBEn tietoturvallisuus ja tuotantovarmuus, mikä sisältää kattavan tietoturvatestauksen tuotannon infra- ja sovellustasolla sekä tarkan infrastruktuurin dokumentoinnin englannin kielellä.

Tietoturvatestaus toteutetaan alalla yleisessä käytössä olevien metodien ja työkalujen, kuten Burp Suite Professional -työkalun avulla. Tietoturvatestauksessa tullaan ottamaan erityishuomioon ”Access control” eli pääsynhallintaan liittyvät haavoittuvuudet sekä yleisimmät web-applikaatioon liittyvät haavoittuvuudet. Kyseisiä haavoittuvuuksia tullaan avaamaan seuraavassa kappaleessa, ja testaamaan käytännössä Tuotantoturvallisuuden varmistaminen -alaluvussa.

Käyttöoikeuksiin liittyvät haavoittuvuudet ovat nostettu kriittisiksi tässä sovelluksessa, koska on tärkeää, ettei käyttäjäoikeuksien ulkopuolista dataa onnistuta lukemaan, poistamaan tai muokkaamaan. Tämä johtaisi siihen, että eri yritykset voisivat saada käsisään toisten ja jopa kilpailevien yrityksien arkaluontoisia tietoja ja pahimmassa tapauksessa korjaamattomia haavoittuvuuksia, joilla voidaan tehdä suurta tuhoa kohde organisaatiolle.

Raportointi-osiossa selitettiin tietoturvahaavoittuvuuksien raportoinnin perusperiaatteet, ja miten ne on hoidettu tämän opinnäytetyön yhteydessä. Tässä tullaan myös avaamaan, miten alalla keskimäärin raportit valmistetaan ja mitä ne sisältävät. Raportointiosiossa kerrotaan myös, mitä infrastruktuurin dokumentointi sisältää ja mitä asioita täytyy ottaa huomioon hyvän infrastruktuurin dokumentoinnissa. Dokumentointi tullaan tekemään englannin kielellä ja tätä dokumenttia ei tulla jakamaan yleiseen jakoon arkaluontoisten tietojen takia.

2 WEB-SOVELLUSTEN YLEISIMMÄT HAAVOITTUVUUDET

Tässä luvussa tullaan avaamaan, mitä ovat web-sovellusten yleisimmät haavoittuvuudet ja miten ne ilmenevät sovelluksissa. Tiedot ovat suurimmilta osin hankittu OWASP TOP 10 -listauksesta, joka on tietoturva-alan suosituin haavoittuvuusstandardisointilistaus. Listaus sisältää kymmenen yleisintä sovellushaavoittuvuutta, sekä listaa haavoittuvuuksien riskit, vaikutukset ja vastatoimenpiteet. The Open Web Application Security Project eli ”OWASP” on voittoa tavoittelematon säätiö, jonka tarkoitus on parantaa ohjelmistojen turvallisuutta (1.)

2.1 Injektiot

Tietoturvassa injektioilla tarkoitetaan hyökkäyksiä, joilla on tarkoitus saada syötettyä haitallista dataa ohjelmaan tai verkkopalveluun. Tämä data voi olla haitallista koodia tai komentoja. Koodit tai komennot suoritetaan frontend-osassa eli käyttäjälle näkyvissä olevassa osassa, kuten nettisivuilla, tai backend-osassa eli käyttäjälle näkymättömässä osassa, kuten tietokannassa. (2.)

Tyypillisiä injektioita ovat SQL- tai OS-injektiot, joissa haitallista dataa saadaan SQL:ssä tietokantaan tai OS:ssa käyttöjärjestelmään. (2.)

2.1.1 SQL-injektio

SQL-injektiot ovat hyökkäyksiä, joilla on tarkoitus manipuloida SQL-lauseketta, niin että hyökkääjä voi hakea, muokata tai poistaa dataa tietokannasta haluamansa mukaan ilman asiaankuuluvaa oikeutta. SQL-injektiossa hyökkääjä voi esimerkiksi muokata hakukyselyä niin, että kysely palauttaakin ylimääräistä tietoa alkuperäisen kyselyn lisäksi. Tämä tieto voisi olla esimerkiksi käyttäjien nimet ja salasanat. Näin hyökkääjä pääsisi käsiksi toisten palvelun käyttäjien tunnuksiin ilman oikeuksia (3.)

SQL-injektiota voidaan myös käyttää esimerkiksi autentikoinnin ohittamiseen, eli hyökkääjä voi ohittaa kokonaan kirjautumissivun ja päästä käsiksi kohdepalveluun ilman tunnuksia tai toimivaa salasanaa.

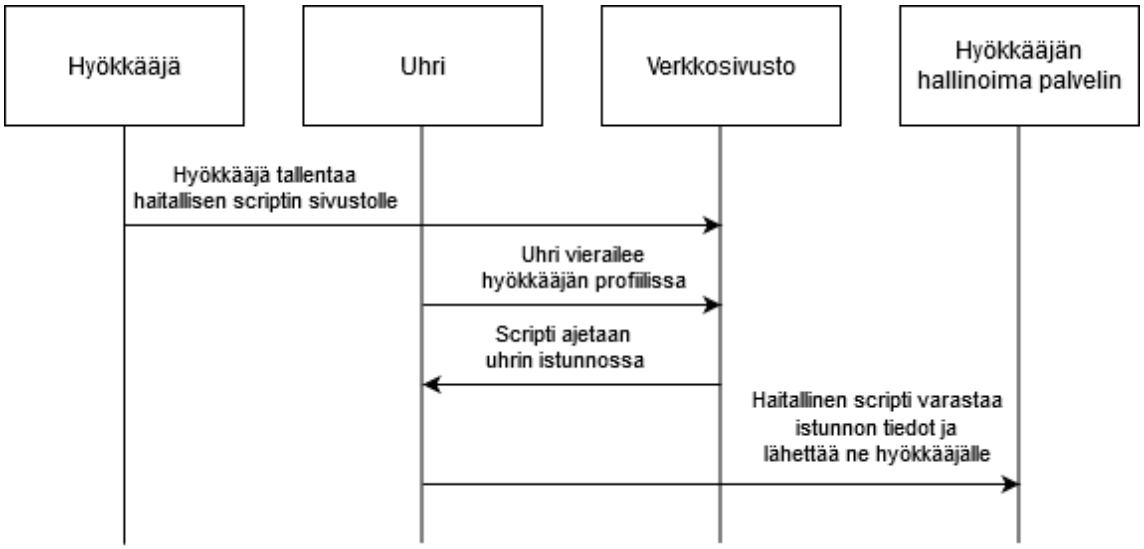
2.1.2 Cross Site Scripting (XSS)

Cross Site Scripting eli XSS on yksi yleisimmistä ja tunnetuimmista hyökkäystekniikoista. Cross Site Scripting -hyökkäyksessä hyökkääjä pystyy ajamaan haitallisia skriptejä verkkopalvelussa. Hyökkäyksessä kohde verkkopalvelu ei tee syötteentarkistusta ennen datan käsittelyä, mikä johtaa skriptin ajamiseen (3.)

2.1.3 XSS Tyypit

XSS-hyökkäyksiä on kolmenlaisia, heijastettu, tallennettu tai DOM-pohjainen. Heijastetussa XSS-hyökkäyksessä hyökkääjä muotoilee sivuston URL-osoitteen niin, että se sisältää haitallisen skriptin. Uhrin vieraillessa tai klikatessa hyökkääjän muotoilemaa linkkiä skripti ajetaan ja se lähettää uhrin istunnon tiedot hyökkääjälle. Näin hyökkääjä saa haltuunsa uhrin istunnon ja voi esiintyä palvelulle uhrina.

Tallennetussa XSS-hyökkäyksessä haitallinen skripti on saatu tallennettua sivustolle esimerkiksi hyökkääjän profiilin nimeksi ja uhrin vieraillessa hyökkääjän profiilissa haitallinen koodi ajetaan ja hyökkääjä saa haltuunsa uhrin istunnon. Kuvassa 1 visualisoidaan tallennetun XSS-hyökkäyksen kulku, miten hyökkääjä saa käsiksi uhrin istunnon.



KUVA 1. Tallennetun XSS-hyökkäyksen kulku

DOM-pohjaisessa XSS-hyökkäyksessä haitallinen skripti on jo kohdepalvelussa ohjelmointivirheen tai aiemman hyökkäyksen ansiosta. Hyökkäys toimii samalla tavoin kuin heijastettu XSS-hyökkäys, mutta DOM XSS -hyökkäyksessä hyökkääjän ei tarvitse syöttää itse haitallista skriptiä palvelulle vaan se on jo palvelussa ennestään. DOM-pohjainen XSS vaatii toimiakseen parametrin, jonka välittämää ei tarkisteta ennen sivulle lisäämistä.

2.2 Virheellinen käyttäjän tunnistaminen ja istunnonhallinta

Käyttäjän tunnistamisessa varmistetaan, että HTTP-pyynnön lähettäjä on todellakin sama käyttäjä, kuin kuka hän väittää olevan. Toiminta tehdään yleisesti käyttäjätunnuksen ja käyttäjälle annetun tunnisteen avulla. Istunnonhallinnassa palvelin pitää yllä tilaa käyttäjän istunnosta ja tietää näin, kuka käyttäjä on kyseessä. Tämä tunniste on uniikki jokaiselle käyttäjälle ja yleisesti se poistetaan käyttäjän kirjauduttua ulos palvelusta.

Virheellisesti toteutetussa käyttäjän tunnistamisessa ja istunnonhallinnassa hyökkääjä onnistuu saamaan haltuunsa uhrin istuntotunnisteen esimerkiksi XSS-hyökkäystä käyttämällä ja näin onnistuu esiintymään palvelimelle käyttäjänä joka hän ei oikeasti ole.

Kyseiset haavoittuvuudet ovat yleisiä ja vakavia, koska toimivan autentikoinnin toteuttaminen on monimutkaista. Lisäksi Kyseisten haavoittuvuuksien löytäminen on yleensä hankalaa, koska jokaisella palvelulla on ainutlaatuinen toteutus istunnonhallinnalle ja käyttäjän tunnistamiselle.

2.3 Virheellinen pääsynvalvonta

Virheellisessä pääsynvalvonnassa hyökkääjä onnistuu pääsemään käsiksi sellaisiin sovelluksin osiin, joihin hänen ei ole tarkoitus päästä. Hyökkäys onnistuu esim. muokkaamalla URL:ää niin, että käyttäjiltä piilotettu ominaisuus avautuu. Muita tapoja päästä käsiksi piilotettuihin ominaisuuksiin on manipuloida HTTP-kutsua lennosta, jolloin hyökkääjä voisi päästä esim. ylläpitäjäsivustolle tai lukemaan toisen henkilötietoja.

2.4 Virheitä sisältävien komponenttien käyttö

Sovelluksessa käytettävissä kirjastoissa, kehyksissä ja ohjelmistomoduuleissa voi olla komponentteja, joissa piilee haavoittuvuuksia. Kyseisillä komponenteilla on yleensä täydet käyttöoikeudet sovelluksiin ja tämän takia haavoittuvuudet komponenteissa voivat johtaa vakaviin seurauksiin, kuten tietovuotoihin tai backend-järjestelmien haltuunottoon. Yleisesti kyseiset haavoittuvuudet ovat tunnettuja, mikä johtaa niiden helppoon käyttöön, koska haavoittuvuuksiin on kirjoitettu valmiita hyökkäystyökaluja ja koodeja, joka helpottaa hyökkääjän toimintaa. (3.)

Opinnäytetyön yhteydessä tehtävässä käytännön testauksessa tullaan testaamaan kohdepalvelu edellä mainittuja haavoittuvuuksia vastaan ja testauksessa tullaan erityisesti ottamaan huomioon pääsynhallintaan liittyvät haavoittuvuudet.

3 TIETOTURVATESTAUS

Tietoturvatestauksella tarkoitetaan työtä, jolla pyritään selvittämään sovelluksen, sivuston tai palvelun tietoturvatoimien taso. Tällaisia toimia voivat olla esimerkiksi luvattoman pääsyn estäminen, haitallisen koodin ajaminen ja oikeuksien hallinta. Tietoturvatestaus tulisi suorittaa säännöllisesti, jopa vuosittain. Etenkin ennen uusien palveluiden käyttöönottoa sekä jokaisen suuremman päivityksen yhteydessä tietoturvatestauksen tekeminen on hyvin tärkeää.

Tietoturvatestauksessa eli penetraatiotestauksessa selvitetään kohteen tietoturvallisuustaso käytännön toimien perusteella. Testausprosessi lähtee liikkeelle aloituspalaverin kautta, jossa sovitaan yhteiset toimintasäännöt asiakkaan kanssa. Palaverissa käydään myös yleisesti läpi kohteen laajuus sekä kohteen rakenne: mille alustalle sovellus on rakennettu, mitä taustapalveluita kyseinen kohde käyttää tai onko testauksessa osa-alueita, joita ei haluta ottaa mukaan testaukseen. Aloituspalaverissa toimitetaan lisäksi testaajalle kohteen osoitteet ja tunnukset, joilla testausta lähdetään toteuttamaan.

Rajoituksia testaukselle asiakas voi asettaa myös silloin, jos kohdepalvelulla on keskeneräisiä tai salaisia osa-alueita, joita ulkopuolisille ei haluta paljastaa. Testaajan on tärkeä ymmärtää, missä raja menee ettei yhteisiä sääntöjä rikota. Välinpitämättömyys annettuja sääntöjä kohtaan voi johtaa jopa luottamuksen menettämiseen, mikä on erittäin haitallista tietoturva-alalla.

3.1 Metodit

Yhtä selkeää polkua tietoturvatestaukselle ei ole, koska kohteet ovat aina ainutkertaisia. Näin ollen ei ole yhtä oikeaa tapaa, jota voitaisiin käyttää jokaisessa testauksessa. Tämä johtaa siihen, että jokaiselle tietoturvatestaajalle syntyy omanlaisensa testausrutiini. Tällöin kohteet tulee tarkistettua monipuolisemmin haavoittuvuuksilta, varsinkin jos samaa kohdetta testaa useampi tietoturvatestaaja.

Tietoturvatestausta voidaan toteuttaa kolmella eri tavalla: valkolaatikko-, harmaalaatikko-, ja mustalaatikkotestauksella.

Valkolaatikkotestauksessa tietoturvatestaaja saa käyttöönsä kohteen lähdekoodin sekä muita testausta helpottavia ennakkotietoja, kuten järjestelmän arkkitehtuurikuvauksen. Kyseiset tiedot voivat auttaa tietoturvatestaajaa tunnistamaan haavoittuvuuksia suoraan koodista tai ymmärtämään kohteen toimintojen logiikkaa testauksen edetessä. (4.)

Harmaalaatikkotestaus on lähellä white box -testausta, mutta näissä testaajalle ei toimiteta lähdekoodia. Harmaalaatikkotestauksessa tyypillisesti tarjotaan testaajalle jotain tietoa järjestelmästä, mutta ei suoranaista lähdekoodia. Tällaisia tietoja voivat olla esimerkiksi arkkitehtuurikuvaus tai tunnukset backend -järjestelmiin. (4.)

Mustalaatikkotestaus on lähimpänä oikean hyökkääjän lähestymistapaa; tällöin testaajalle ei tarjota minkäänlaista sisäistä ennakkotietoa kohteesta. Testaaja joutuu etsimään avoimista lähteistä tietoa kohteesta, mikäli sellaista on tarjolla julkisessa verkossa. (4.)

Edellä mainituissa tyyleissä on sekä hyvät että huonot puolensa. Valkolaatikkotestauksessa testaaja voi löytää helpommin haavoittuvuuksia, mutta se ei kuvaa yhtä hyvin oikean hyökkääjän toteuttamia hyökkäystapoja kuin mustalaatikkotestaus kuvaisi. Harmaalaatikkotestaus on hyvä välimalli näille toimintatavoille; testaajalla ei ole liikaa tietoa järjestelmästä, mutta hänellä on testausta nopeuttavia tietoja, jolloin testaus on laadukkaampaa. (4.)

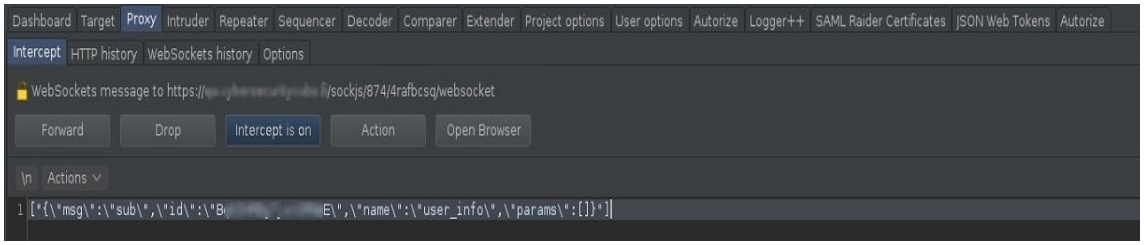
Yhteenvetona tässä opinnäytetyön yhteydessä tehdyssä tietoturvatestauksessa käytettiin valkolaatikkotestaustyyliä, koska saatavilla oli kohdepalvelun kaikki tiedot ja pääsy projektinhallintajärjestelmään, mistä löytyy lähdekoodi palvelulle. Yleisesti ottaen asiakkaille tehtävissä tietoturvatestauksissa käytetään harmaalaatikkotestausmetodia.

3.2 Työkalut

Tässä työssä tietoturvatestaus toteutettiin Burp Suite -työkalulla, joka on PortSwiggerin luoma työpöytäsovellus web-sovellusten tietoturvatestaukseen. Burp Suite on monipuolinen työkalu, joka sisältää tieturvatestauksessa hyödylli-

siä ominaisuuksia, kuten proxyn, scannerin, repeaterin ja decoderin. Nettisivuillaan PortSwigger kuvailee työkaluaan termillä: ”A Swiss Army knife for hackers” (5), joka on osuva kuvaus, sillä Burp Suite sisältää lähes kaiken mitä web-sovellusten testaaja voi toivoa. Edellä mainittujen ominaisuuksien lisäksi Burp Suitessa on ”BApp Store” (6), joka on kokoelma yhteisön kehittämiä lisäosia Burp Suitelle. Nämä lisäosat vievät tietoturvatestauksen mielekkyyden uudelle tasolle esimerkiksi automatisoimalla tiettyjä rutiininomaisia toimintoja.

Proxy on työkalun yleisimmin käytettyjä toimintoja. Tämän ominaisuuden avulla voidaan keskeyttää palvelimen ja asiakkaan välisiä HTTP- ja websocket-pyyntöjä. Kyseiset HTTP- ja websocket-pyynnöt sekä -vastaukset ovat keskeisessä osassa testausta, koska kyseisiä kutsuja manipuloimalla voidaan toteuttaa aiemmin esiteltyjä hyökkäyksiä frontend-rajoituksista huolimatta. HTTP- ja websocket-pyynnössä voi myös kulkea syöttökenttiä, joita testattavassa kohteessa ei ole käyttäjille näkyvissä. Tämä voi avata uusia hyökkäyskohteita testaajalle. Kuvassa 2 on havainnollistettu millaiselta keskeytetty websocket-kutsu näyttää sekä mitä tietoja tämä kutsu sisältää, ja minkä tyyppistä dataa testaaja manipuloi testauksessaan.

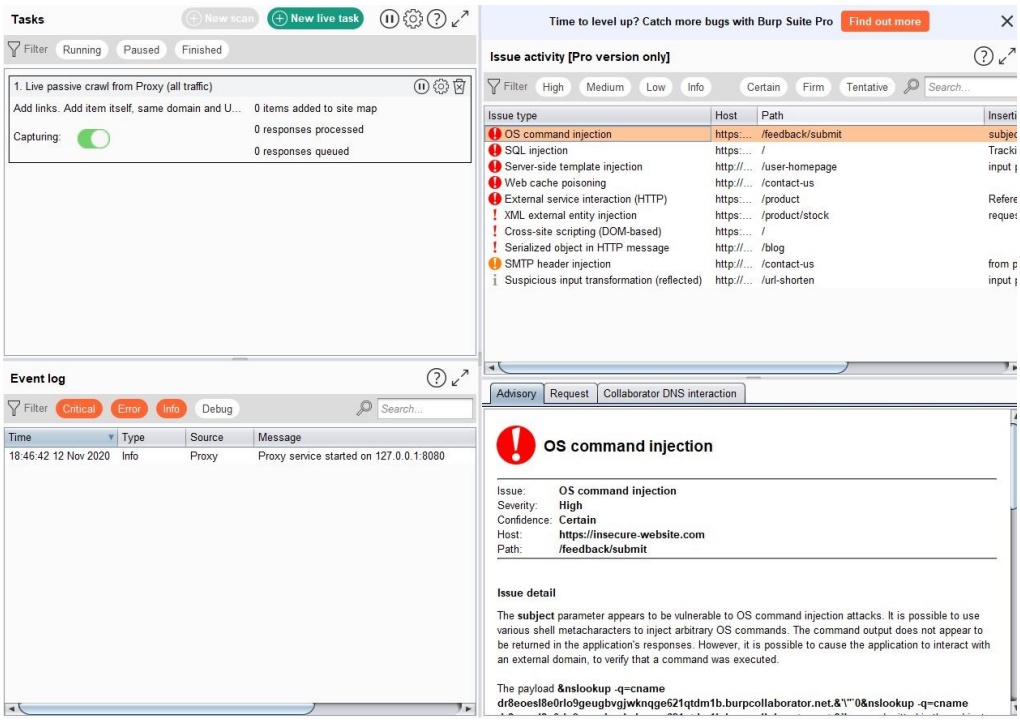


KUVA 2 Burp Suiten proxy-työkalun intercept-ominaisuudella keskeytetty websocket-pyyntö.

Scanner on nimensä mukaisesti automaattinen skanneri, joka ottaa sovelluksen historiasta osoitteita, joissa testaaja on vieraillut. Näihin osoitteisiin skanneri suorittaa automaattista testausta eli manipuloi historiassa olevia pyyntöjä ja lähettää ne automaattisesti testauksessa olevalle kohteelle. Skanneri on oiva tapa selvittää, onko kohteessa niin sanottuja low-hanging fruit -haavoittuvuuksia eli helposti löytyviä haavoittuvuuksia. Vaikka tällaiset haavoittuvuudet ovat helposti havaittavissa, se ei tarkoita, etteivätkö ne voisi olla kriittisiä tietoturvan kannalta. Skanneri

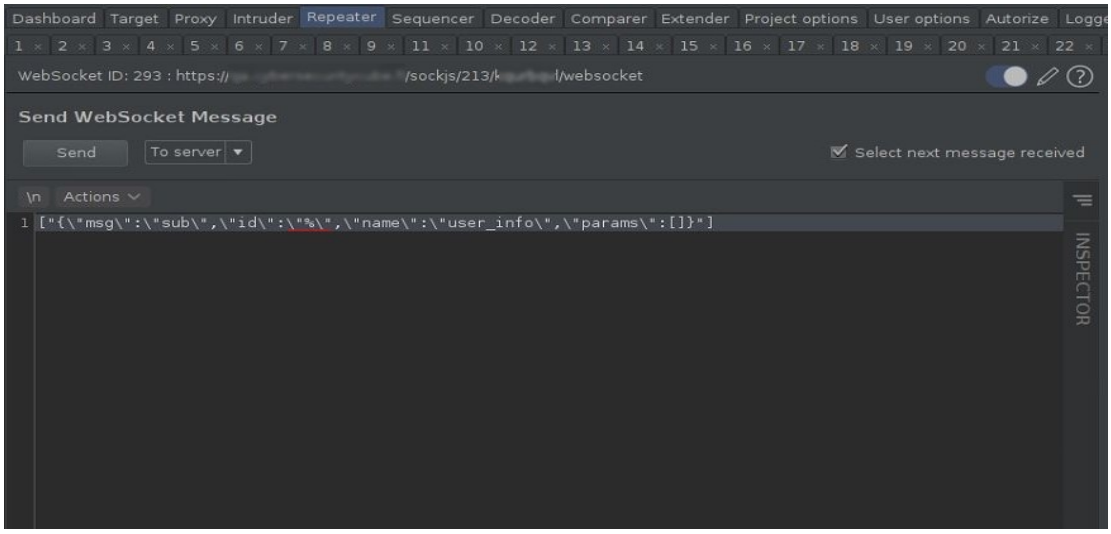
myös ”lokittaa” eli kirjaa jokaisen havainnon talteen myöhäisempää tarkastelua varten.

Automaattiskannerin löydökset on aina tarkistettava manuaalisella testauksella mahdollisten False Postive -tulosten vuoksi. Näin ollen automaattiskannerit osoittavat mahdollisia heikkoja kohtia ja haavoittuvuuksia sovelluksesta, jotka testaaja käy läpi manuaalisesti ja todentaa ne oikeiksi. Kuvassa 3 Burp Suiten esimerkkikuva skannerin tallentamista havainnoista. Oikealla alakulmassa näkyvästä ruudusta selviää, että esimerkkisivun ”subject”-parametri on OS-injektiohaavoittuvainen.



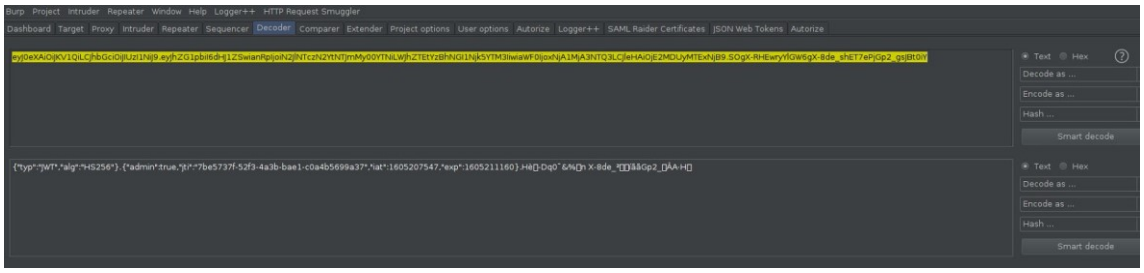
KUVA 3 Esimerkkikuva Burp Suiten skannerin tallentamista havainnoista

Repeater on toinen yleisesti käytetty ja tärkeä työkalu, jolla voidaan kopioida Proxyn historiassa olevia HTTP- tai websocket-kutsuja. Näin historiassa olevia kutsuja voidaan manipuloida helposti jälkikäteen ja lähettää ne uudelleen kohteelle. Repeaterin etuina on nopea ja helppo tapa manipuloida yksittäistä kutsua ja tarkastella, miten kohde reagoi manipuloituihin pyyntöihin. Kuvassa 4 on kuvassa 2 olevan websocket-pyynnön manipulointi repeater-työkalulla.



KUVA 4 Proxystä otettu Websocket-pyyntö (kuva 2), jossa id-arvo manipuloituna ja lähetettynä kohteelle.

Decoder on työkalu, jolla voidaan dekoodata pyynnöissä kulkevia merkkijonoja. Monesti HTTP- ja websocket-pyynnöissä ja vastauksissa kulkevat arvot ja etenkin otsakkeissa olevat evästeet ovat koodattuna base64-muotoon. Evästeet muunnetaan base64-muotoon, jotta ne eivät korruptoituisi tiedonsiirrossa palvelimen ja asiakkaan välillä (7.) Kuvassa 5 on esimerkki JWT-evästeestä dekoodattuna base64-muodosta selväkieliseen muotoon Burp Suiten decoder-työkalua käyttäen.



KUVA 5 Esimerkki JSON Web Token dekoodattuna base64-muodosta selkokieliseen

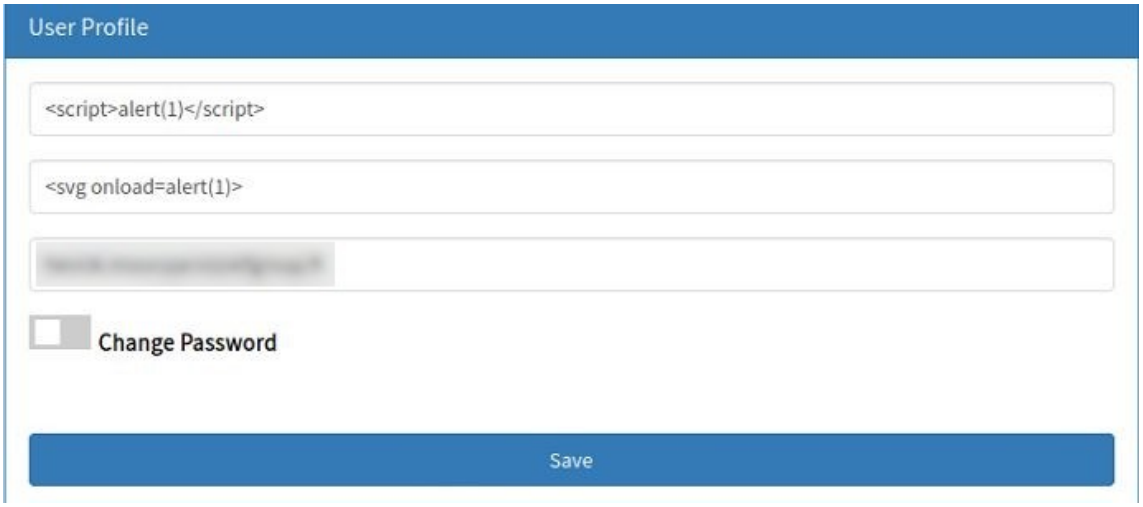
4 TUOTANTOTURVALLISUUDEN VARMISTAMINEN

Tässä tekstissä esitetyt vaiheet on kuvattu samassa järjestyksessä, kuin elfCUBEn tietoturvatestaus käytännössä toteutettiin. Aluksi testaus aloitettiin tutustumalla kohdepalveluun selaamalla läpi elfCUBE -sivusto ja klikkailemalla toimintoja, joita se piti sisällään. Samalla Burp Suite -työkalu toteutti automaattista skannausta kohdepalvelulle. Burp Suiten sisältämä skanneri ilmoittaa havainnoista, joita se tekee selailun yhteydessä. Automaattiskannerin etu on, että se testaa satoja HTTP-kutsuja testaajan vierailemiin osoitteisiin ja datan syöttöjä sekunneissa. Lisäksi skanneri voi mahdollisesti löytää piilotettuja ominaisuuksia.

Kohteen läpikäynnin ensimmäisillä kerroilla nousee usein esiin testaajia kiinnostavia alueita. Itselläni nousi kiinnostuksen kohteeksi eri organisaatioiden välinen ”access control” eli pääsynhallinta, koska pelkona oli, että yritys A voi päästä käsiksi yrityksen B tietoihin. Huomioksi nousi, voidaanko toisen organisaation tietoja lukea, poistaa tai muokata ilman asiaankuuluvia oikeuksia. Kyseistä haavoittuvuutta testattiin manipuloimalla sovelluksessa kulkevia websocket-kutsuja hyödyntämällä äsken mainittua proxy-työkalua. Testauksessa koitettiin muokata id-arvoa websocket-pyynnössä yritys A:n id:stä yritys B:n arvoksi. Tämä pyyntö palautti ”403 forbidden” -vastauksen, joka indikoi, että pääsynhallinta tämän ominaisuuden osalta on asetettu oikein.

Toinen huomioni herättänyt osa-alue oli yritykseen kutsuminen. elfCUBEen tullaan lisäämään monia asiakasyrityksiä ja tämä tapahtuu lähettämällä kutsumislinkki asiakkaan kontaktihenkilölle. Tätä kutsumislinkkiä manipuloimalla olisin voinut onnistua kutsumaan itse itseni yrityksen organisaatioon ilman, että järjestelmänhallitsija lähettäisi kutsua minulle todellisuudessa. Lähdin testaamaan tätä ottamalla Burp Suiten historiasta kyseisen kutsumislinkin lähettävän websocket-pyynnön. Pyyntö lähetettiin Repeater-työkalulle ja lähdin manipuloimaan websocket-pyynnön alkuperäisen yrityksen id-arvon tilalle uhrin yrityksen id:n. Manipuloitu websocket-pyyntö palautti virheilmoituksen ”Not authorized”. Tämä implikoi, että kutsumislinkin toiminnallisuus on toteutettu oikein tietoturvan näkökulmasta ja manipuloitu kutsumislinkki ei koskaan saapunut perille.

Seuraavaksi ryhdyttiin suorittamaan rutiininomaista tietoturvatestausta. Kohdepalvelussa oli Oma profiili -sivusto, jossa käyttäjä voi hallita oman käyttäjänsä tietoja. Käyttäjä pystyi hallitsemaan etu- ja sukunimeään sekä sähköpostiosoitettaan. Näihin kenttiin koitettiin syöttää aiemmin mainittua haitallista XSS-koodia. Kuvassa 6 etu- ja sukunimikenttiin on syötetty haitallista JavaScript-koodia, ja jos kyseisiä kenttiä ei ole implementoitu oikein, kyseiset koodit ajetaan ja näytölle ilmestyy ilmoituslaatikko. elfCUBEssa kyseisten kenttien syötteentarkistus oli tehty oikein ja ladatessa sivu uudestaan ilmoituslaatikko ei avautunut. Tämä tarkoittaa, että kentät on toteutettu oikein tietoturvan näkökulmasta.



KUVA 6 Etu- ja sukunimikentissä tallennettuna tyypilliset XSS-koodit. Kyseiset koodit suorittaessaan luovat vain ilmoituslaatikon, joka sisältää tekstin ”1”, eli haitallista koodia ei ajeta näissä skripteissä. Ainoastaan havainnollistetaan, että haavoittuvuus on olemassa.

Tämän tyyppistä syöttökenttien tarkistusta testaan kaikista sovelluksen syöttökentistä. Syöttökenttiä on muuallakin kohdepalvelussa, joita ei pystytä paljastamaan elfCUBEn julkaisemattomuuden takia. Kohdepalvelusta testattiin edellä mainittujen testien lisäksi logiikkavirheitä ja muita ominaisuuksia kuten pääsynhallintaa sekä istunnonhallintaa.

Tämän tietoturvatestauskierroksen aikana tulleita havaintoja ei tulla julkaisemaan tässä opinnäytetyössä arkaluontoisten tietojen vuoksi.

5 RAPORTOINTI

Tässä osiossa kerrotaan, miten tietoturva-alalla suoritetaan yleisesti tietoturvatestauksen jälkeinen raportointi. Tietoturvatestaus on tieteellinen prosessi, joten tieteellisen prosessin tavoin tehtyjen havaintojen pitää olla toistettavissa. Tämän takia raportti on merkittävä osa onnistuneessa tietoturvatestauksessa, koska sen avulla melkein kuka tahansa voi toistaa testauksen vaiheet ja todeta havainnot todeksi. Lisäksi asiakkaan on saatava konkreettinen tulos testauksesta, ja asiakkaiden kautta tulevat kolmannet osapuolet haluavat usein virallisen dokumentin asiakkaan tietoturvatasosta.

Toinen osio kertoo elfCUBEn infrastruktuurin dokumentoinnista, joka tulee virralliseksi dokumentaatioksi elfGROUPille ja tuotokseksi tästä opinnäytetyöstä. elfGROUP ylläpitää korkeaa tietoturvatasoa ja monet yrityksen laatustandardit (kuten ISO-27001-standardi, joka on standardi mikä käsittelee tietoturvallisuuden hallintajärjestelmän toteuttamista, ylläpitämistä, luomista ja jatkuvaa parantamista koskevia vaatimuksia (8)) vaativat kattavaa dokumentaatiota ja seurantaa tuotekehitysprosessista varsinkin tuotteita kehittäessä (9.)

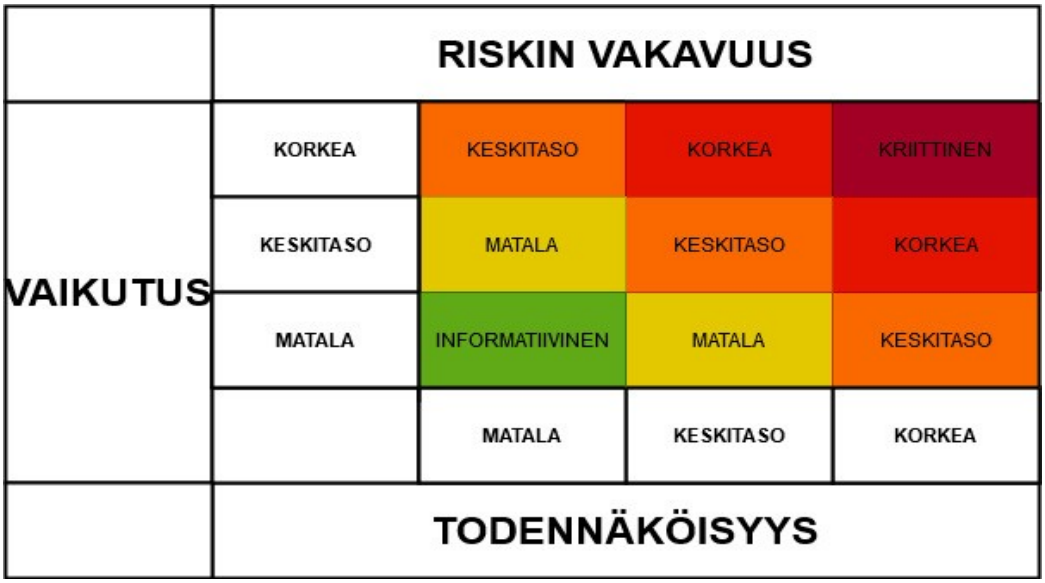
5.1 Tietoturvapoikkeavuuksien raportoiminen

Tietoturvapoikkeavuuksien loppuraportin kirjoittamisprosessissa tulee ottaa huomioon, että kyseistä dokumenttia lukevat todennäköisesti myös ei-tekniset henkilöt, kuten yrityksen ylin johto, joka ei todennäköisesti ymmärrä tai välitä, jos esimerkiksi http-pyynnön body-osuudessa on kenttä, johon voidaan upottaa haitallista JavaScript-koodia. He haluavat vain tietää vastauksen yhteen yksinkertaiseen kysymykseen: ”Olemmeko turvassa vai emme?”

Toisaalta dokumenttia lukevat myös tekniset henkilöt, jotka ovat vastuussa havaittujen ongelmien korjaamisesta. He todennäköisesti haluavat tietää kolme asiaa: missä vika on, kuinka vakava haavoittuvuus on ja kuinka se korjataan. (9)

Hyvän tietoturvatestauksen loppuraportin tulisi edetä loogisesti. Raportin alussa tulisi olla tietenkin kansilehti, joka sisältää testausyrityksen nimen ja logon sekä asiakkaan nimen ja palvelun. Tämän jälkeen olisi hyvä olla seuraavat osiot:

- **Yhteenvetotiivistelmä** – Tiivistelmän tulisi sisältää kohdepalvelun tiedot, mikä yritys kyseisen palvelun on rakentanut ja yleiskuvauksen kohdepalvelusta, mihin ja miten kyseistä palvelua käytetään. Tämän lisäksi tiivistelmän tulisi kertoa, mitä testauksessa tehtiin, minkä tyyppisiä haavoittuvuuksia kohteesta pyrittiin löytämään, yleiskuva haavoittuvuuksista sekä tieto, kuinka paljon haavoittuvuuksia löytyi ja millä kriittisyydellä. Lopussa olisi hyvä olla yhteenveto kohdepalvelun tietoturvatasosta.
- **Yhteenveto haavoittuvuuksista** – Haavoittuvuuksien yhteenvedosta tulisi selvitä yhdellä silmäyksellä, kuinka paljon haavoittuvuuksia on ja millä kriittisyystasolla ne ovat. Kyseiset tiedot voidaan ryhmitellä esimerkiksi vakavuuden tai CVSS-pistemäärän mukaan, joka on alan standardi haavoittuvuuksien vakavuuksien arviointiin (10). Tähän voidaan käyttää grafiikoita, kuten taulukkoja tai kaavioita. Kriittisyydet voidaan määrittää kuvan 7 OWASP:in tarjoamalla kaaviolla.



KUVA 7 Kriittisyyden laskemiseen käytettävä kaavio, jolla voidaan laskea kriittisyys todennäköisyyttä ja vaikutusta käyttämällä (10).

- **Tehdyt havainnot** – Tehdyistä havainnoista tulisi olla yksittäin listattuna jokainen havainto, joka on löytynyt testauksessa. Tässä listauksessa olisi hyvä olla tiedot, mitä kohdepalvelun osaa havainto koskee, kuvaus havainnoista, kuinka havainto korjataan ja mahdolliset kuvankaappaukset havainnollistamaan, kuinka havainto toistetaan.
- **Päätelmät** – Tässä osiossa olisi hyvä olla kaikki, mitä on kirjoitettu raporttiin, mutta tieto tulisi kertoa ytimekkäästi ja keskittyen seuraaviin mahdollisiin vaiheisiin.

Tämän tyyppistä loppuraporttia ei tehty tämän opinnäytetyön yhteydessä tehdyllä testauskierroksella, vaan löydetyt havainnot raportointiin suoraan elfGROUPissa käytössä olevaan projektinhallintajärjestelmään. Lopuksi tehtiin keveämpi dokumentti testatuista toiminnoista sisäiseen käyttöön.

5.2 elfCUBEn infrastruktuurin dokumentointi

Tämä osio keskittyy työn tulokseen eli elfCUBEsta tehtävään dokumenttiin elfCUBEn takana olevasta infrastruktuurista. Dokumentti tulee sisältämään kuvauksen arkkitehtuurista ja teknologioista sovelluksen takana. Kyseisen dokumentin tarkoitus on helpottaa elfCUBEn kehittämistä ja ylläpitämistä jatkossa. Dokumentti sisältää vastaavat tiedot, kuin pilottivaiheessa olevassa versiossa on. Nämä tiedot tulevat muuttumaan palvelun kehittyessä ja dokumenttia päivitetään sitä mukaan, kun uusia versioita tai muutoksia infrastruktuuriin tehdään.

elfCUBEa ajetaan elfGROUPin omilla palvelimilla ja dokumenttiin on kuvattuna diagrammina palvelinympäristö. Kuvasta selviää, mitä palvelimia verkossa on ja missä osoitteissa. Lisäksi kuvasta selviää virtuaaliverkot, aliverkot sekä sallittu liikenne. Kuvassa myös visualisoitu liikenteen reitti palvelimelta julkiseen verkkoon sekä minkä kytkimien ja palomuurien kautta liikenne kulkee. Dokumentissa on myös listattu palomuurisäännöt elfCUBEn osalta, minkä porttien ja osoitteiden on sallittua ottaa yhteyttä elfCUBEn palvelimille ja mikä liikenne elfCUBEsta on sallittu julkiseen verkkoon.

Sovellustasolta dokumenttiin on kirjattuna tarkat versiot ohjelmistoista, jotka pyörittävät elfCUBEa, kuten WWW-palvelimen, tietokannan ja JavaScriptin ajamisen

mahdollistavan ympäristön versio. Myös elfCUBEn eriytykseen muusta backend -järjestelmästä käytetyn sovelluksen versiot ja konfiguraatiot ovat kuvattuna tähän dokumenttiin.

Dokumentista tuli hyvä alusta infrastruktuurin dokumentointiin ja siitä on helppo tarkistaa, ovatko käytettyjen teknologioiden versiot ajan tasalla vai vaativatko ne päivitystä.

6 YHTEENVETO

Tässä opinnäytetyössä käsiteltiin elfCUBEn tietoturvallisuuden varmistamista ja tuotantoon viemistä. Tietoturvatestauksessa on tarkoitus parantaa kohdejärjestelmän tai palvelun tietoturvatasoa ja tietoturvatestaus on metodina oiva työkalu tämän toteuttamiseen. Tietoturvatestauksessa nousee esille todellisia haavoittuvuuksia, joita vihamieliset hyökkääjät voisivat oikeasti käyttää, ja tietoturva ei jää pelkästään teoriatasolle.

Tietoturvatestaajalta vaaditaan laajaa osaamista eri teknologioista, koska hyökkäyspintoja on valtava määrä ja jokainen kohde on ainutlaatuinen. Onneksi tietoturvatestaajan elämää on helpottamassa paljon erilaisia työkaluja. Kyseisiä työkaluja käyttämällä testaajalta huomaamatta jäävät asiat voivat nousta esille ja tietoturvatasoa saadaan edelleen nostettua paremmalle tasolle.

elfCUBEn tietoturvatasoa onnistuttiin kohentamaan tämän opinnäytetyön aikana ja saatiin varmuus palvelun luotettavuudesta julkaisupilottia varten. Testauksessa todettiin kriittisten osa-alueiden tietoturvataso hyväksi ja muutenkin koko sovelluksen tietoturvataso on korkealla. Tämä olikin oletettu tulos, koska yrityksessä tietoturva on pääroolissa ja tietoturvallinen ohjelmistokehitys ja ylläpito on keskeisessä osassa koko palvelun elinkaaren ajan.

Haastavimpiin ja monimutkaisempiin hyökkäysketjuihin ei tässä tietoturvatestauksessa syvennytty, mutta tietoturvatestausta jatketaan palvelun kehittyessä ja erikoisemmilta haavoittuvuuksilta testejä tehdään tulevaisuudessa.

LÄHTEET

1. OWASP Top Ten. 2017. OWASP The Open Web Application Security Project. Saatavissa: https://owasp.org/www-project-top-ten/. Hakupäivä 2.10.2020.
2. Muscat, Ian 2019. What Are Injection Attacks. Acunetix. Saatavissa: https://www.acunetix.com/blog/articles/injection-attacks/. Hakupäivä 2.11.2020.
3. Ahmed Ansari, Juned 2015. Web Penetration Testing With Kali Linux Second Editions. United Kingdom: Packt Publishing Ltd.
4. Poston, Howard 2020. What are Black Box, Grey Box, and White Box Penetration Testing? Infosecinstitute. Saatavissa: https://resources.infosecinstitute.com/topic/what-are-black-box-grey-box-and-white-box-penetration-testing/. Hakupäivä 12.11.2020.
5. Burp Suite Professional. 2020. PortSwigger. Saatavissa: https://portswigger.net/burp/pro. Hakupäivä 12.11.2020.
6. BApp Store. 2020. PortSwigger. Saatavissa: https://portswigger.net/bappstore. Hakupäivä 12.11.2020.
7. Li, Vickie 2019. Powering the Internet with Base64. Saatavissa: https://medium.com/swlh/powering-the-internet-with-base64-d823ec5df747. Hakupäivä 12.11.2020.
8. ISO/IEC 27001:2013(en). 2013. ISO. Saatavissa: https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en. Hakupäivä 17.11.2020.
9. Sheward, Mike 2012. The Art of Writing Penetration Test Reports. Saatavissa: https://resources.infosecinstitute.com/topic/writing-penetration-testing-reports/. Hakupäivä 19.11.2020.
10. Common Vulnerability Scoring System. Saatavissa: https://www.first.org/cvss/. Hakupäivä 19.11.2020.