jamk.fi

Web-sovellustason palomuuri osana kerroksellista suojausarkkitehtuuria

Lari Lantela

Opinnäytetyö
Maaliskuu 2017
Tekniikan ja liikenteen ala
Insinööri (AMK), tietotekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Lantela, Lari	Opinnäytetyö, AMK	Maaliskuu 2017
	Sivumäärä 125	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Web-sovellustason palomuuri osana kerroksellista suojausarkkitehtuuria

Tutkinto-ohjelma

Tietotekniikan (tietoverkkotekniikan) koulutusohjelma

Työn ohjaajat

Karo Saharinen

Mika Rantonen

Toimeksiantaja

Kansaneläkelaitos

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Kansaneläkelaitoksen ICT-osasto. Opinnäytetyön tavoitteena ja kehityskohteena oli selvittää web-sovellustason palomuurin tarjoamat mahdollisuudet osana sovelluksien kerroksellista suojautumisarkkitehtuuria. Määrittelyvaiheessa keskityttiin tyypillisen yksinkertaisen web-sovelluksen käyttämään rakenteeseen sekä sovellustasolla tapahtuviin tietoturvauhkiin ja niiden mahdollisiin vaikutuksiin. Web-sovellustason palomuurituotteista opinnäytetyössä keskityttiin kaupalliseen F5 BIG-IP ASM -tuotteeseen.

Opinnäytetyön toteutusvaihe, jossa web-sovelluksiin kohdistettiin erilaisia hyökkäyslausekkeita, suoritettiin eristetyssä laboratorioympäristössä. Toteutusvaiheessa suojattavina web-sovelluksina käytettiin kahta erilaista yksinkertaista PHP-pohjaista sovellusta.

Työn avulla saavutettiin kattava tietopohja web-sovellustason palomuurien tarpeellisuudesta sovellustason uhkien torjunnassa. Dokumentaation avulla toimeksiantajalle luotiin käytännön esimerkkien avulla hyvät lähtökohdat BIG-IP ASM -web-sovellustason palomuurin käyttöönottoon. Käytännön esimerkkien avulla tuotiin esille myös NGFW- ja IPS-tuotteiden heikkoudet web-sovellustason uhkien torjumisessa. Opinnäytetyön pohjalta saatiin myös johtopäätöksiä, joiden avulla web-sovellustason palomuureja voidaan arvioida kriittiseltä pohjalta.

Lopputuloksena saavutettiin laboratorioympäristöön kaksi ASM:n toimivaa tietoturvapolitiikkaa, joiden perusteella ASM-moduulin käyttöönottoa on helppo lähteä suunnittelemaan varsinaiseen tuotantoympäristöön.

Avainsanat (asiasanat)

Web-sovellustason palomuuri, sovellustason tietoturva, WAF, F5 ASM

Muut tiedot

-

jamk.fi

Description

Author(s)	Type of publication	Date
Lantela, Lari	Bachelor's thesis	March 2017
		Language of publication: Finnish
	Number of pages 125	Permission for web publication: x

Title of publication

Web application layer firewall as part of layered security architecture

Degree programme

Information Technology

Supervisors

Karo Saharinen

Mika Rantonen

Assigned by

Social Insurance Institution of Finland

Abstract

The bachelor's thesis was assigned by the ICT department of Social Insurance Institution of Finland. The purpose of the thesis was to investigate opportunities and benefits offered by the web application layer firewall used as a part of a layered security architecture. The definition phase focused on the structure of a typical simple web application, the application layer threats and their potential effects against organization's web applications. As for the web application layer firewalls, the main focus was on the commercial F5 BIG-IP ASM web application firewall.

The implementation phase, where application layer attacks were used against simple web applications, was implemented in an isolated laboratory environment. Web applications in question were simple PHP-language based applications.

A comprehensive knowledge base for the basic need to implement web application firewall to protect organization's web applications was achieved. Practical examples revealed the weaknesses of NGFW and IPS products against web application level threats. The thesis helped to understand the features of the F5 BIG IP ASM product, as well as its effectiveness and its typical use cases. Furthermore, the research provided conclusions to evaluate WAF products critically.

As the end result the thesis achieved to implement two working effective ASM security policies into the development environment. On this basis, it is easy to continue bringing the system to the actual production environment.

Keywords/tags (subjects)

Web Application Firewall, application layer security, WAF, F5 ASM

Miscellaneous

-

Sisältö

Lyhenteet

ADC	Application Delivery Controller
AFM	Advanced Firewall Module
APM	Access Policy Manager
ASM	Application Security Manager
ASVS	Application Security Verification Standard
AV	Antivirus
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheet
CVE	Common Vulnerabilities and Exposures
DLP	Data Loss Prevention
DoS	Denial of Service
DVWA	Damn Vulnerable Web Application
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Security
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IIS	Internet Information Services
KATAKRI	Kansallinen turvallisuusauditointikriteeristö
LDAP	Lightweight Directory Access Protocol
LFI	Local File Inclusion
LTM	Local Traffic Manager
NGFW	Next Generation Firewall
OSI	Open Systems Interconnection Reference Model
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
PCI DSS	Payment Card Industry Data Security Standard
RFC	Request For Comments
RFI	Remote File Inclusion
SIEM	Security Information and Event Management
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transport Layer Protocol
TLS	Transport Layer Security
TMOS	Traffic Management Operation System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VAHTI	Valtiohallinnon tieto- ja kyberturvallisuuden johtoryhmä
WAF	Web Application Firewall
WWW	World Wide Web
XML	Extensible Markup Language
XSS	Cross-Site Scripting

Kuviot

Taulukot

1 Lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Kansaneläkelaitos (Kela). Kelan tehtävänä on hoitaa Suomessa asuvien henkilöiden perusturvaa heidän erilaisissa elämäntilanteissaan. Kelan asiakkaita ovat kaikki Suomessa ja ulkomailla asuvat Suomen sosiaaliturvan piiriin kuuluvat henkilöt. Kela vastaa myös Kansallisen Terveysarkiston (KanTa) palvelujen tuottamisesta. (Toiminta 2016.)

Kelan organisaation muodostavat kuusi erillistä tulosyksikköä**:** asiakkuuspalvelut, etuuspalvelut, kehittämispalvelut, ICT-palvelut, yhteiset palvelut ja esikuntapalvelut. Asiakkuuspalveluiden tulosyksikkö vastaa asiakaspalvelusta sekä Kelan asiakkaiden neuvonnasta ja heidän ohjaamisestaan. Etuuspalvelujen tulosyksikkö puolestaan vastaa Kelan tarjoamien etuuksien ratkaisutoiminnasta. Kehittämispalvelujen tulosyksikön tehtävä on huolehtia asiakas- ja etuusprosessien sekä tietojärjestelmien kehityksestä ja niiden parantamisesta. ICT-palvelujen tulosyksikkö tuottaa ja toimittaa Kelan tarjoamat ICT-palvelut Kelan asiakkaiden, sidosryhmien ja henkilöstön käyttöön. Yhteisten palvelujen tulosyksikkö vastaa toiminnan mahdollistavien palvelujen tuottamisesta ja niiden toimittamisesta. Esikuntapalvelujen tulosyksikön vastuualueeseen puolestaan kuuluu huolehtia johtamisen ja strategian toteuttamisesta. (Organisaatio 2016.)

1.2 Opinnäytetyön tavoitteet ja tutkimusongelma

Kansaneläkelaitoksen ICT-osasto oli nähnyt tarpeelliseksi vahvistaa omien tietojärjestelmien suojaustaan ottamalla käyttöön web-sovellustasolla toimivan palomuurin. Palomuurin käyttöönotto edellyttää kuitenkin kattavaa perehtymistä itse järjestelmään ja sen tarjoamiin ominaisuuksiin. Järjestelmään perehtymisen lisäksi oli luotava teoreettiset lähtökohdat sen tarpeellisuudelle.

Toimeksiantajan vaatimukset opinnäytetyölle olivat aluksi nykypäivän web-sovelluksiin kohdistuvien uhkien määrittely ja niiden luokittelu. Tämän yhteyteen haluttiin myös tietopohja uhkien mahdollisista vaikutuksista organisaatiolle.

Varsinaisena tarkoituksena oli kuitenkin tarkastella sitä, minkälaisen lisäarvon web-sovellustason palomuuri tuo organisaatiolle heidän jo olemassa olevien suojausjärjestelmien tueksi. Tämän yhteydessä esille nousee kerroksellinen suojautuminen ja sen tärkeys nykypäivän tietoverkoissa tapahtuvien uhkien torjumisessa.

Opinnäytetyössä keskityttiin tarkemmin toimeksiantajan määrittelemään kaupalliseen F5 Networksin BIG-IP ASM -web-sovellustason palomuuriin. Järjestelmän osalta toimeksiantajalle tehtiin kattava tietopohja sen ominaisuuksista, ylläpidosta, raportoinnista ja erilaisista käyttöönottoskenaarioista.

Opinnäytetyön toteutusvaiheen pääkohdat olivat ASM-järjestelmän tarjoamien tietoturvapolitiikkatyyppien vertailu, suojauskyvyn arviointi sovellustason uhkia vastaan, järjestelmän tuottamien mahdollisten väärien positiivisten- ja negatiivisten hälytysten tarkastelu sekä järjestelmän käyttöönottoon liittyvien alustavien prosessien suunnittelu. Lisäksi toteutusvaiheessa tutkittiin käytännön esimerkkien avulla web-sovellustason palomuuria osana kerroksellista suojausarkkitehtuuria. Toteutusvaihe ja kaikki sen yhteydessä suoritettavat testit tehtiin eristetyssä laboratorioympäristössä.

1.3 Tilannekatsaus tämän päivän tietoturvaan

Tietojärjestelmien sujuva toimivuus ja tietoturva ovat nykypäivänä kriittisiä tekijöitä eri yritysten ja organisaatioiden yleistoiminnalle. Lyhytkin tietoliikennekatko tai organisaation ydinliiketoiminnan kannalta kriittisen sovelluksen toimimattomuus aiheuttavat lähes poikkeuksetta mittamaattoman suurta haittaa. Sama ajatusmalli on rantautumassa myös tietoturvan puolelle. Lyhytkestoinen palvelunestohyökkäys tai organisaation hallinnoiman palvelun tietoturva-aukon hyväksikäyttö voi aiheuttaa myös suuria imagollisia haittavaikutuksia.

Tietoturvasta on viime kuukausina tullut entistä suuremmissa määrin median käsittelyn kohde. Erilaisista tietovuodoista ja haavoittuvuuksista kirjoitetaan suomalaisissa valtamedioissa jo lähes päivittäin. Uutisoinnin ansiosta tietoturvan ongelma- ja kehityskohteet ovat tulleet myös tavallisen kansan tietouteen. Erilaisten tietovuotojen esille nostaminen konkreettisten esimerkkien avulla onkin kiinnittänyt

kansan huomion tietoturvan merkityksellisyyteen. Tämä on myös osaltaan asettanut paineita eri yrityksille ja organisaatioille kehittää omien tietojärjestelmiensä tietoturvaa. Voidaankin karkeasti sanoa, että median uutisointi on saanut aikaan joitakin positiivisia tietoturvan kehityshankkeita.

Vuonna 2016 viikolla 35 Helsingin Sanomissa uutisoitiin Suomessa yleisesti käytössä olevien verkkolaitteiden haavoittuvuudesta. (Ks. kuvio 1.)

Kuvio 1. Helsingin Sanomien uutinen verkkolaitteen haavoittuvuudesta (Halminen 2016)

Uutinen nousi Helsingin Sanomien päivän luetuimmaksi uutiseksi. Uutisessa tietoturvayhtiö syyttää laitevalmistajaa piittaamattomuudesta sen verkkolaitteesta löydetyn haavoittuvuuden korjaustoimenpidevalmiuksia kohtaan.

Jatkuvan nopean muutoksen alla olevat tietoteknilliset ympäristöt asettavat haasteita tietoturvan tason tehokkaaseen ylläpitämiseen ja kehittämiseen. Yleistyvät pilvipalvelut sekä jatkuvasti kasvava salatun liikenteen määrä tietoverkossa vaativat uusia keinoja tietoturvan ja havainnointikyvyn kehittämiseen. Haasteita tulee tulevaisuudessa lisäämään myös nopeaa kasvua tekevä esineiden internet (engl. *Internet of Things*).

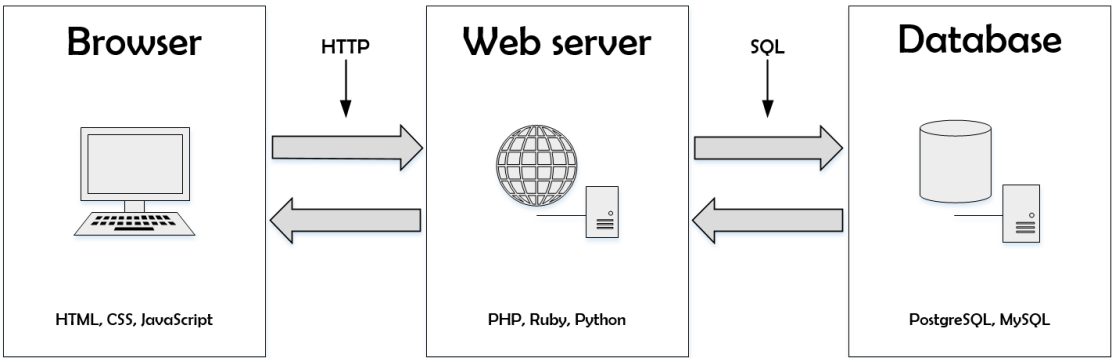
2 Web-sovellukset

2.1 Yleistä

Karkeasti määriteltynä web-sovelluksella tarkoitetaan ohjelmaa, jota suoritetaan palvelimella ja käytetään erillisten web-selainten avulla. Web-sovelluksien mahdollisina toteutustapoina on useita erilaisia tekniikoita ja ohjelmointikieliä. Yritysten ja organisaatioiden käytössä olevien web-sovelluksien tarkoituksena on liiketoiminnan tukeminen sekä asiakas- ja yhteistyökumppaneiden kollaboraation mahdollistaminen. 1990-luvun lopussa ja 2000-luvun alussa web-sovellukset olivat usein yksinkertaisia monoliittisiä järjestelmiä. Nykypäivänä niiden rakenne on kuitenkin muuttunut hajautetuksi, ja niistä on tullut monimutkaisia kokonaisuuksia, joissa yhdistyvät useat eri komponentit ja tekniikat. Web-sovellukset tarjoavat lähes rajattoman suuret mahdollisuudet erilaisissa toteutuksissa ja niitä on mahdollista kustomoida hyvin yksityskohtaisesti. Nämä ovat niitä päätekijöitä, jotka ovat johtaneet myös hyvin liiketoimintakriittisten ja arkaluontoista tietoa sisältävien järjestelmien toiminnan siirtymiseen erilaisten web-sovelluspohjaisten alustojen päälle. (Jokinen 2012, 2-4.)

2.2 Rakenne ja toiminta

Web-sovelluksen yleisarkkitehtuuriin kuuluu kolme erillistä komponenttia: selain, verkko ja palvelin. Sovelluksen käyttäjän selaimella tekemillä pyynnöillä voidaan lukea ja muokata palvelimella olevaa tietoa. Käyttäjän selain visualisoi palvelimen lähettämän resurssin ymmärrettävään muotoon. Lähes jokaisen web-sovelluksen yhteydessä toimii erillinen tietokanta. Tietokannan tehtävänä on säilöä web-sovelluksen käyttämää tietoa. Kun palvelimen yhteyteen lasketaan kuuluvaksi erillinen tietokanta, voidaan puhua kolmitasoisesta rakennearkkitehtuurista. (Ks. kuvio 2.)



Kuvio 2. Yksinkertaisen web-sovelluksen rakennearkkitehtuuri

Kolmitasoinen arkkitehtuuri koostuu selaimesta, web-palvelimesta ja sen yhteydessä toimivasta, usein erillisestä tietokannasta.

Web-sovellusten tyypillinen sisäinen arkkitehtuuri on karkeasti jaettavissa kahteen erilliseen osa-alueeseen: asiakaspäähän (engl. *frontend*) ja palvelinpäähän (engl. *backend*). Asiakaspäässä koodi suoritetaan selaimessa. Siinä suoritetaan kaikki käyttäjälle näkyvä toiminta. Tyypillisimpiä asiakaspään kieliä ovat HTML, CSS ja JavaScript. Palvelinpäässä toteutustapoja on useita. Keskeiseen rooliin nousee yleensä erillinen tietokanta. Palvelinpuolen tapahtumat ovat käyttäjälle läpinäkyviä. Tyypillisimpiä palvelinpään koodeja ovat PHP, Python ja Perl. Asiakaspään ja palvelinpään väliseen liikenteeseen käytetään HTTP-protokollaa tai sen salattua versiota HTTPS-protokollaa.

2.3 Yhteysprotokolla (HTTP)

HTTP (Hypertext Transfer Protocol) on sovellustason siirtoprotokolla, jonka avulla asiakas ja palvelin kommunikoivat keskenään. HTTP-protokolla vaatii liikennöintiin luotettavan yhteyden, joten tavallisesti HTTP-liikenne tapahtuu TCP/IP-yhteyksien päällä. Oletuksena se käyttää liikennöintiin TCP-porttia 80. Tällä hetkellä yleisimmin käytössä oleva HTTP-versio on HTTP/1.1, joka määritellään RFC:ssä 2616. Vastaavasti uusin HTTP-versio on HTTP/2, jonka standardi julkistettiin toukokuussa 2015 (Stenberg 2015, 4). HTTP/2-standardi määritellään tarkasti RFC:ssä 7540. HTTP-protokollan pohjalta on kehitelty myös salattu versio, HTTPS-protokolla (Hypertext Transfer Protocol Security). HTTPS-protokollalla tapahtuva liikennöinti salataan erillisellä TLS-protokollalla ennen sen lähettämistä verkon ylitse. Tämän vuoksi

HTTPS-toteutukset vaativat palvelimelta palvelinvarmenteen eli sertifikaatin sekä yksityisen avaimen. HTTPS-protokolla käyttää oletuksena liikennöintiin TCP-porttia 443.

HTTP-protokolla sisältää useita metodeja, joita käytetään erilaisten toimenpiteiden suorittamiseen asiakkaan ja palvelimen välillä. Yleisimmät HTTP-metodit ovat GET ja POST. GET-metodin avulla asiakas pyytää web-palvelinta lähettämään sille tietyn sivun tai resurssin. GET-metodilla ei ole koskaan tarvetta muuttaa pyydetyn sivuston rakennetta. Perättäiset HTTP GET -pyynnöt tuottavat aina saman lopputuloksen. Vastaavasti POST-metodin avulla asiakas välittää palvelimelle jotakin informaatiota. Kyseinen informaatio voi olla esimerkiksi web-sivuston lomakkeen tekstikenttään syötettävä tieto. HTTP-protokollan yhteydessä on myös mahdollista käyttää useita muita metodeita, mutta niiden käyttö on huomattavasti GET- ja POST-metodeja harvinaisempaa. Kaikki HTTP/1.1-standardin käyttämät metodit ovat lueteltuina yksityiskohtaisesti RFC:ssä 2604. Osa näistä metodeista on kuitenkin suunniteltu tukemaan vain web-sovellusten kehitystä ja testausta, eivätkä ne näin ollen ole yleisessä käytössä. (RFC 2616 1999, 9.3 – 9.5.) Taulukossa 1 on lueteltuna HTTP/1.1-standardin käyttämät kahdeksan metodia ja niiden lyhyet kuvaukset.

Taulukko 1. HTTP/1.1-standardin käyttämät metodit (Humphrys N.d, muokattu).

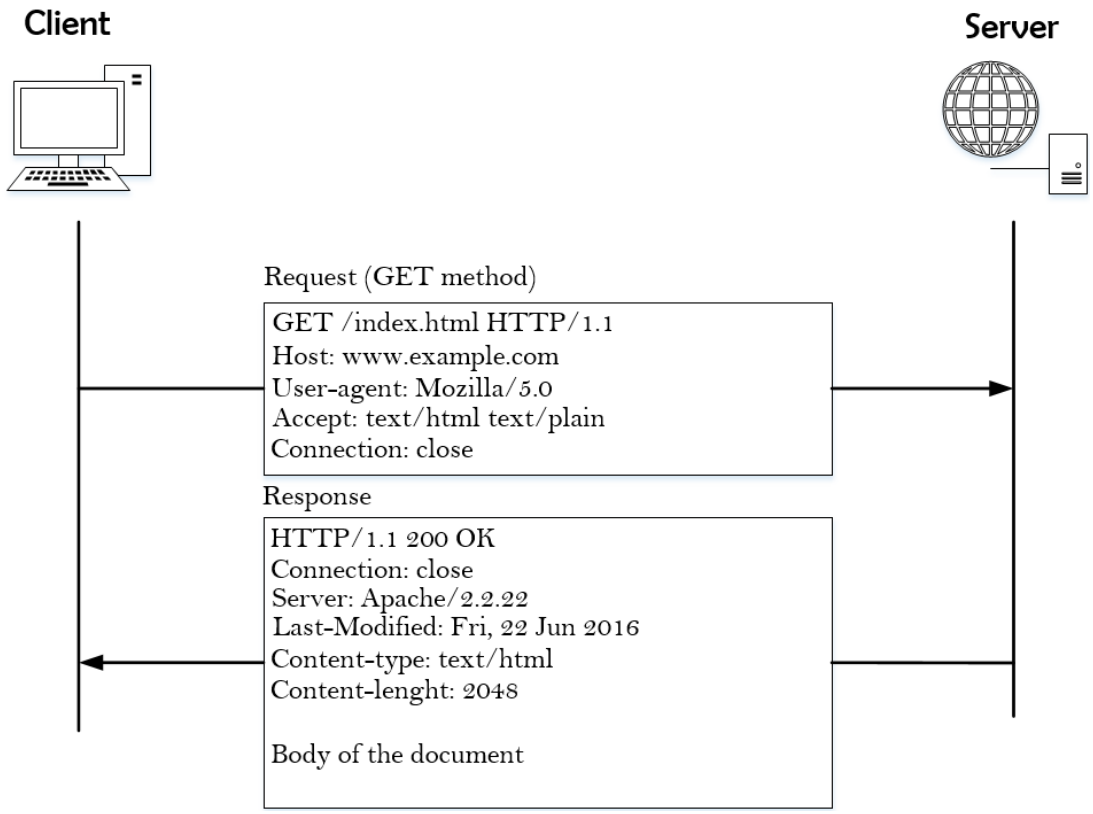
METHOD	DESCRIPTION
GET	Request to read a web page
HEAD	Request to read a web page's header
PUT	Request to store a web page
POST	Append to a named resource (e.g a webpage)
DELETE	Remove the web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

Kun palvelin palauttaa pyydetyn resurssin asiakkaalle, se viestii hakupyynnön toteutumisesta erillisillä statuskoodeilla. Nämä koodit kertovat esimerkiksi pyydetyn resurssin lähettämisen onnistumisesta tai siitä, ettei palvelimella ole tällä hetkellä saatavilla kyseistä resurssia. Yleisimmät HTTP-protokollan käyttämät statuskoodit ovat *200 OK* ja *404 NOT FOUND*. Taulukossa 2 on lueteltuna kaikki yleisimmät HTTP/1.1-standardin käyttämät statuskoodit.

Taulukko 2. HTTP/1.1-standardin statuskoodit (Green 2016, 10, muokattu)

CODE	DESCRIPTION	MEANING
200	OK	Standard success response
201	CREATED	New resource created
301	MOVED PERMANENTLY	Permanent redirect to new URI
304	NOT MODIFIED	Safe to use page stored in cache
307	TEMPORARY REDIRECT	Use new URI now; try old later
401	UNAUTHORIZED	Authentication failed
403	FORBIDDEN	Disallowed, auth will not help
404	NOT FOUND	Resource was not found
405	METHOD NOT ALLOWED	Used GET when should use POST
500	INTERNAL SERVER ERROR	Internal server error

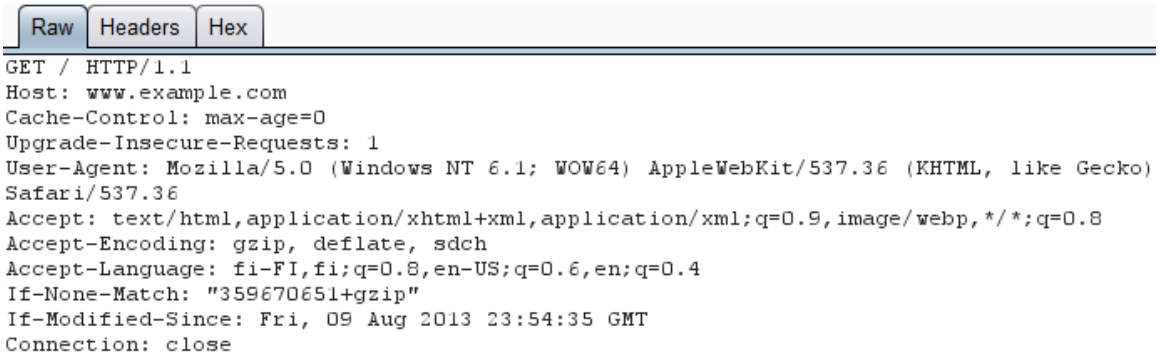
HTTP-protokollan toiminnan pohjalla olevaa asiakas-palvelin-mallia on havainnollistettu esimerkin avulla kuviossa 3.



Kuvio 3. HTTP-viestit asiakkaan ja palvelimen välillä

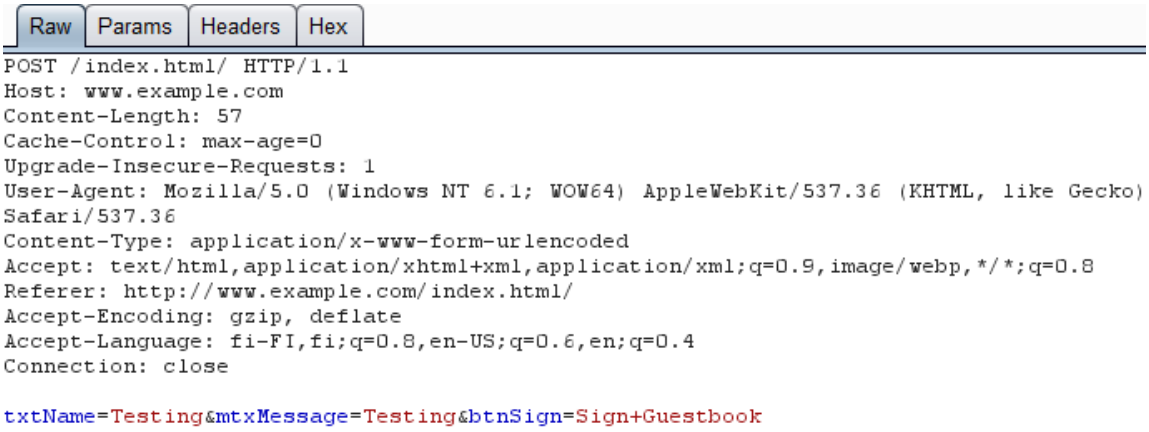
Lähtötilanteessa asiakas haluaa saada luettua palvelimelta tietyn resurssin (index.html). Tätä varten asiakas lähettää web-palvelimelle HTTP GET -viestin, jonka alkuosa koostuu käytettävästä HTTP-metodista, pyydetystä URI:sta (Uniform

Resource Identifier) ja HTTP-protokollan versionumerosta. URI on käytännössä merkkijono, jonka avulla asiakas osoittaa palvelimelta haluamansa resurssin. Web-sovelluksien yhteydessä yleisesti käytetty URI-tyyppi on URL (Uniform Resource Locator), jonka avulla osoitetaan pyydetyn tiedon sijainti. Varsinaisen pyynnön alkuosaa seuraavat viestin muut otsakkeet. Palvelimen vastaanotettua asiakkaan pyynnön se vastaa viestiin asiaankuuluvalla tavalla. Mikäli asiakkaan pyytämä resurssi on saatavissa ja tietyt muut ehdot täyttyvät, palvelin lähettää asiakkaan pyytämän resurssin HTTP-vastauksen sisällä. Vastauksen sisältämät otsaketiedot koostuvat käytettävästä HTTP-versionumerosta, statuskoodista ja muista palvelimen tiedoista. Mikäli pyydettyä resurssia ei ole saatavilla, palvelin lähettää asiakkaalle paluuviestin statuskoodilla 404. Kuvion 4 todennuksessa on esimerkkikaappaus asiakkaalta palvelimelle lähetettävästä HTTP GET -viestistä ja sen sisältämistä HTTP-otsakekentistä.



Kuvio 4. HTTP GET -pyyntö asiakkaalta palvelimelle

Kuvion 5 todennuksessa on kaappaus HTTP POST -viestistä ja sen sisältämistä HTTP-otsakekentistä ja datasta.

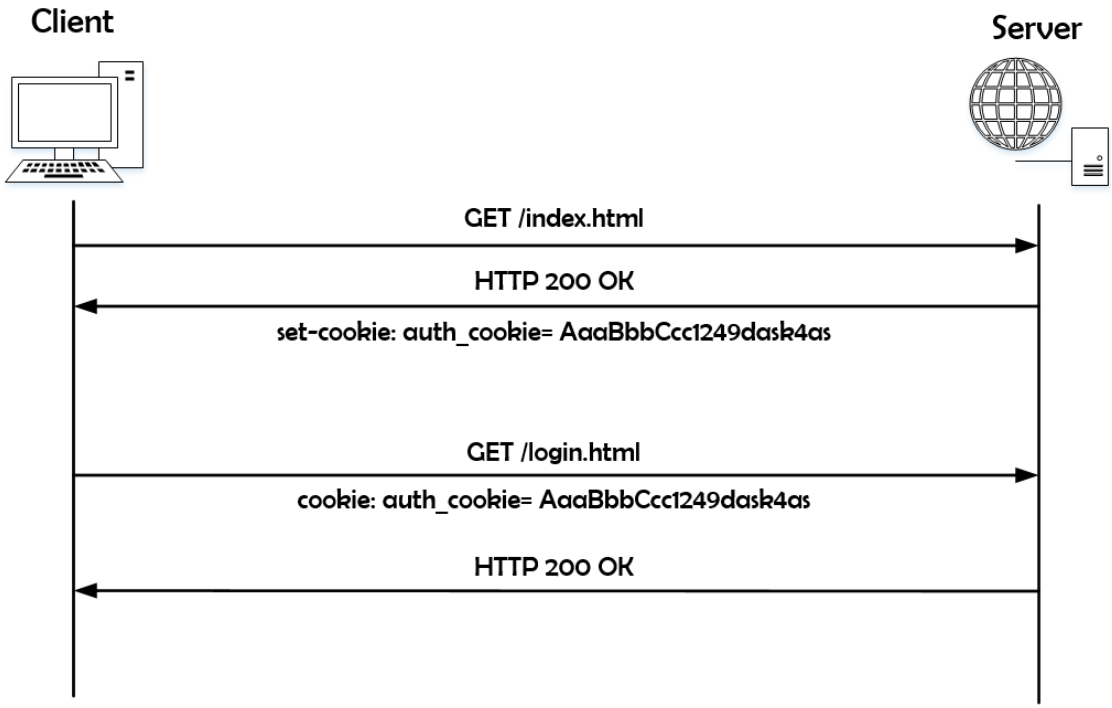


Kuvio 5. HTTP POST -pyyntö asiakkaalta palvelimelle

POST-pyyntöä käytetään tässä esimerkissä apuna kohdepalvelimen päällä toimivan sovelluksen vieraskirjaan kirjoittamisessa.

2.4 Evästeet ja istunnot

Pohjimmiltaan HTTP-protokolla on tilaton. Tämä asettaa sille käytännön toteutukseen useita erilaisia ongelmia. Protokollan tilattomuutta on pyritty paikkamaan ottamalla sen yhteydessä käyttöön erilliset evästeet (engl. *cookies*). Evästeet ovat pieniä koodinpätkiä, jotka kuljetetaan HTTP-viestien kehystiedoissa palvelimelta asiakkaalle ja toisin päin. Evästeillä on moninaisia käyttötarkoituksia: niiden avulla palvelinpää pystyy esimerkiksi identifioimaan ja yhdistämään tietyt asiakkaalta tulevat HTTP-pyynnöt tietyksi asiakkaaksi. Evästeet tallennetaan usein asiakaspään selaimeen ja niillä on olemassa tietty ennalta määritelty elinikä. Evästeiden avulla web-palveluihin voidaan toteuttaa useita kehittyneitä ominaisuuksia, kuten turvallisen kirjautumisen toteutus sekä erilaisia funktioita käyttäjien seurantaan ja analysointiin. Kuviossa 6 on havainnollistettu palvelinevästeiden toimintaa yksinkertaisen esimerkin avulla.



Kuvio 6. Palvelinevästeiden toimintaperiaate

Palvelin vastaa asiakkaan HTTP GET -viestiin HTTP-vastauksella, jonka otsaketietoihin on lisätty palvelimen generoima eväste. Viestin saatuaan asiakaslaite tallentaa palvelimelta vastaanotetun evästeen oman selaimensa välimuistiin ja käyttää sitä jatkossa omissa HTTP-pyynnöissään. Seuraavissa kyselyissä asiakaslaite lähettää HTTP-viestin mukana palvelimelta saadun evästeen. Palvelimen saatua päätelaitteen viestin voi se evästeen avulla päätellä tietyt HTTP-viestit kuuluvaksi samalle asiakaslaitteelle.

Eri käyttäjien ja lähteiden väliset istunnot eritellään toisistaan yleensä erillisen istuntotunnisteen avulla (engl. *session ID*). Kyseisessä toimintamallissa palvelin luo asiakkaalle erillisen istuntotunnisteen ja lähettää sen HTTP-vastauksen yhteydessä asiakkaalle. Jatkossa asiakas välittää istuntotunnisteen palvelimelle aina jokaisen lähetetyn viestin yhteydessä. Tällöin palvelin osaa pitää kirjaa tulevista yhteyksistä ja luokitella saman istuntotunnisteen sisältämät viestit kuuluvaksi samaan istuntoon. Istuntotunniste voidaan välittää palvelimelle HTTP-viestin yhteydessä erillisenä URL-parametrinä, piilotettuna lomakkeen kenttänä tai vaihtoehtoisesti myös evästeen yhteydessä. (Ihantola 2016.)

3 Web-sovelluksiin kohdistuvat uhkatekijät

3.1 Yleistä

Web-sovelluksiin kohdistuvilla uhkilla tarkoitetaan web-sovelluksia kohtaan suoritettavaa haitallista toimintaa, jolla voi olla web-sovelluksen tarjoamalle palvelulle ja sen omistajalle negatiivisia vaikutuksia. Vaikutukset voivat olla omistajalle suoria tai epäsuoria. Web-sovelluksiin kohdistuvat uhkatekijät ovat merkittävä osa kaikesta verkossa tapahtuvasta haitallisesta toiminnasta. Uhkien määrää ovat lisänneet palveluiden nopea siirtyminen web-pohjaisille alustoille sekä web-sovelluksien laaja hyökkäyspinta-ala.

Erilaiset verkkohyökkäykset ovat viime vuosina siirtyneet yhä enemmissä määrin verkkokerrokselta sovelluskerrokselle. Lisäksi hyökkäyksistä on tullut jatkuvasti entistä vaikeammin havaittavia ja torjuttavia. Aikaisemmin web-sovelluksiin suunnatut hyökkäykset kohdistettiin usein palvelujen alustoihin, kuten Linux-puolella suosittuun ja yleisesti käytössä olevaan Apache-palvelinohjelmistoon ja Windows-puolella IIS:ään (Internet Information Services). Alustoihin kohdistuvat hyökkäykset perustuvat niissä oleviin tunnettuihin tietoturva-aukkoihin. Nykyään kuitenkin web-sovellusten pohjana toimivien alustojen tietoturvaan kiinnitetään entistä enemmän huomiota. Säännölliset ja tiheät päivitysvälit ovat osaltaan edesauttaneet tietoturvan kehittymistä sovellusten alustapuolella. Mikäli päivityksistä huolehditaan säännöllisin väliajoin, pienenevät mahdollisuudet alustapuolen hyökkäyksien toteuttamiseen merkittävästi. Tämän seurauksena hyökkääjät ovatkin alkaneet kiinnittää huomiota itse alustojen päällä toimiviin web-palveluihin. Web-palvelujen yksilöllisyys, monipuolisuus ja yleisyys mahdollistavat hyökkääjille suuren hyökkäyspinta-alan ja lähes rajattomat mahdollisuudet erilaisten hyökkäysten toteuttamiselle. Alustoihin kohdistuvia hyökkäyksiä ei kuitenkaan voida rajata ulos kokonaan, vaan niiden päivityksiin ja tietoturvakorjauksiin on myös jatkossa kiinnitettävä erityistä huomiota. (Lehtonen & Siljander 2010, 6.)

Hyökkääjät voidaan usein profiloida kolmeen erilaiseen kategoriaan, jotka ovat heijastettavissa heidän osaamistasoonsa. Osaamistaso ei varsinaisesti paljasta hyökkääjän motiiveja. Amatöörit ovat raportoiduista tietoturvariskeistä ylivoimaisesti

suurin yksittäinen ryhmä. He löytävät web-sovelluksesta yleensä tahallisesti tai tahattomasti tietoturva-aukkoja ja käyttävät niitä hyväkseen. Usein amatööreillä ei ole kattavaa ymmärrystä suorittamistaan toimista. Toiseen ryhmään kuuluvat hakkerit ja kräkkerit, joiden tavoitteena on päästä käsiksi informaatioon, johon heillä ei varsinaisesti ole luvallista oikeutta. Hakkereilla on usein syvällinen ymmärrys eri järjestelmistä. He tutkivat kuitenkin yleensä järjestelmiä ainoastaan kehitys- ja testimielessä sekä raportoivat tietonsa oikeille tahoille. Sen sijaan kräkkereiden toiminta on enemmän vihamielistä, ja he tavoittelevatkin yleensä vain omaa etuaan. Kolmas ryhmä on verkossa tapahtuva ammattirikollisuus, jonka motiivi tietomurtoihin on pääsääntöisesti raha ja valta. (Saarelma 2014, 31.)

Web-sovelluksiin kohdistuvien hyökkäysten motiivit voivat olla hyvin moniosaisia, mutta usein niihin kuitenkin liittyy pyrkimys oman edun saavuttamiseen. Myös yleinen haitanteko on yksi tyypillisistä motiiveista etenkin amatöörien keskuudessa. Viime aikoina pinnalle ovat nousseet myös erilaiset poliittiset ja uskonnolliset motiivit.

3.2 Haavoittuvuudet ja niiden vaikutukset

Web-sovellusten yhteydessä haavoittuvuudella tarkoitetaan järjestelmän suunnittelussa, toteutuksessa tai sen toiminnassa olevaa heikkoutta. Haavoittuvuus mahdollistaa hyökkääjälle web-sovelluksen normaalista toiminnasta poikkeavan toiminnan suorittamisen. Tämä taas voi johtaa suoraan haittavaikutuksiin sovelluksen omistajalle. Haavoittuvuus voi olla dynaamisessa web-sovelluksessa esimerkiksi käyttäjän syöttämän tiedon puutteellinen varmentaminen, väärin toteutettu kirjautumismenetelmä tai tietokantayhteyden puutteellinen sulkeminen session päättyessä. Haavoittuvuutta ei pidä koskaan ajatella itsenäisenä hyökkäyksenä, vaan pikemminkin erillisen hyökkäyksen mahdollistava tekijänä.

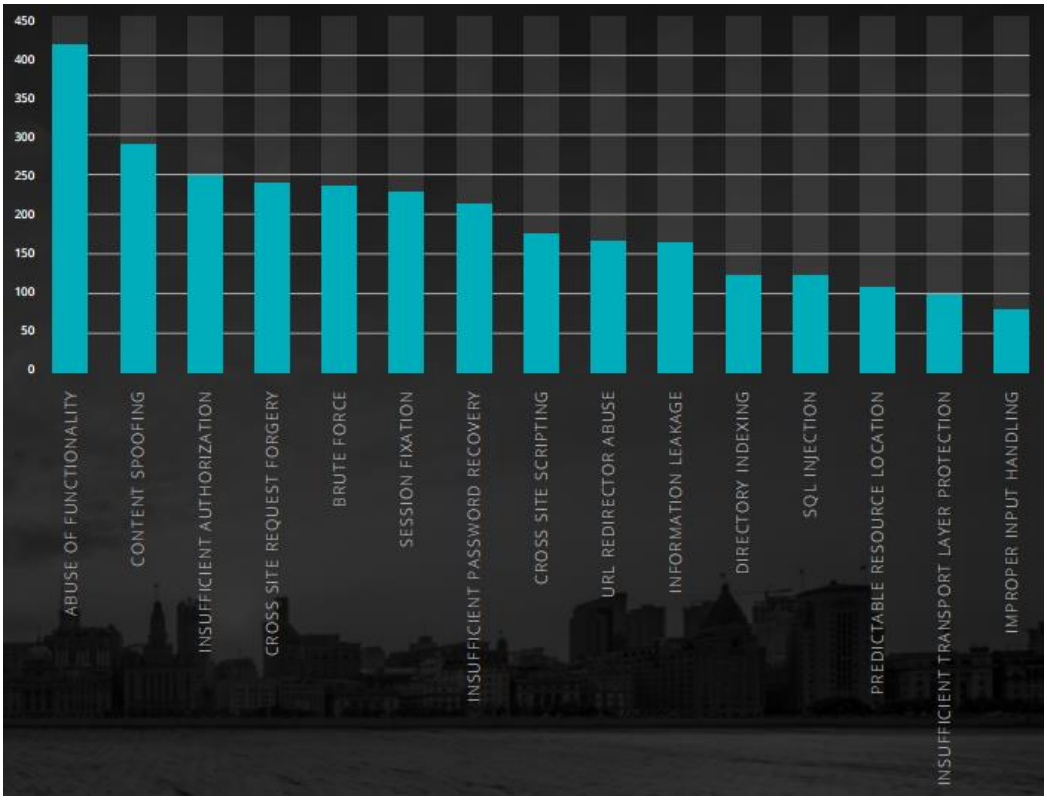
Lähes jokainen web-sovellus on haavoittuvainen. Positive Techonologies -tietoturvayhtiön tekemän tutkimuksen mukaan jopa 95 prosenttia kaikista web-sovelluksista sisältää vähintään yhden keskitason haavoittuvuuden. Korkean tason haavoittuvuus löytyy vastaavasti 62 prosentista kaikista web-sovelluksista (Web Application Vulnerability Statistics 2013). Jokaisella isolla organisaatiolla on useita

kymmeniä, ellei satoja asiakkaiden käytössä olevia palvelukriittisiä web-sovelluksia. Jokaisessa sovelluksessa on keskimäärin yhteensä 5–32 erilaista haavoittuvuutta. Tämän myötä yhden organisaation web-sovelluksissa olevien haavoittuvuuksien kokonaismäärä voi nousta jopa tuhansiin. Kriittisen haavoittuvuuden keskimääräinen elinikä web-sovelluksessa on yli 300 päivää. Vastaavasti korkean tason haavoittuvuus on hyödynnettävissä web-sovelluksessa keskimäärin yli 500 päivää. (Web Application Security Statistics Report 2016.)

Organisaation on tärkeää osata tunnistaa ja luokitella omissa web-sovelluksissaan esiintyvät haavoittuvuudet. Yleensä haavoittuvuudet luokitellaan niiden vakavuuden perusteella luokkiin matala, keskitaso, korkea ja kriittinen. Luokittelu perustuu moneen erilliseen tekijään, kuten haavoittuvuuden tyyppiin, hyödynnettävyyteen ja sen vaikutuksiin. Haavoittuvuuksien luokittelutavat eroavat kuitenkin toisistaan jonkin verran tekijöidensä mukaan. Mahdollisten haavoittuvuuksien osalta on myös pyrittävä huomioimaan haavoittuvan sovelluksen palvelukriittisyys ja hyökkääjän mahdollisuudet haavoittuvuuden hyödynnettävyyteen.

Web-sovelluksissa esiintyville haavoittuvuuksille ei ole määriteltävissä yhtä selkeää syytä. Haavoittuvuudet ovat kuitenkin usein seurausta puutteellisesti suunnitellusta ja testatusta web-sovelluksesta. Vaikka web-sovelluksen käyttämä lähdekoodi olisikin kirjoitettu yleisesti hyväksyttyjen hyvien käytänteiden mukaisesti, voi puutteellinen tietoturvatestaus silti aiheuttaa siinä suuren riskin organisaation tietoturvalle.

Web-sovelluksen käyttämästä lähdekoodista löydettyjä haavoittuvuuksia ei voida korjata automatisoidusti. Kun sovelluksen testaaja, kehittäjä tai käyttäjä löytää ja raportoi siinä olevasta haavoittuvuudesta, kuluu sen korjaamiseen aina tietty aika. Korjaamiseen kuluva aika on riippuvainen itse sovelluksesta, käytettävästä alustasta, haavoittuvuuden vakavuustasosta sekä mahdollisesti organisaation intresseistä ja tietotaidosta. Huomattavia eroja syntyy myös kaupallisten ja avoimeen lähdekoodiin perustuvien ratkaisujen välillä. WhiteHat Security tietoturvayhtiön vuoden 2016 tietoturvaraportti osoittaa, että tyypillisimpien sovellustason haavoittuvuuksien korjaamiseen keskimääräisesti kuluva aika on aina mitattavissa kuukausissa. (Ks. kuvio 7.)



Kuvio 7. Haavoittuvuuksien korjaamiseen kuluva aika (Web Applications Security Statistics Report 2016)

WhiteHat Securityn laatiman raportin perusteella siinä mukana olevien haavoittuvuuksien korjaamiseen kuluva aika eri organisaatioissa on keskimäärin noin 200 päivää.

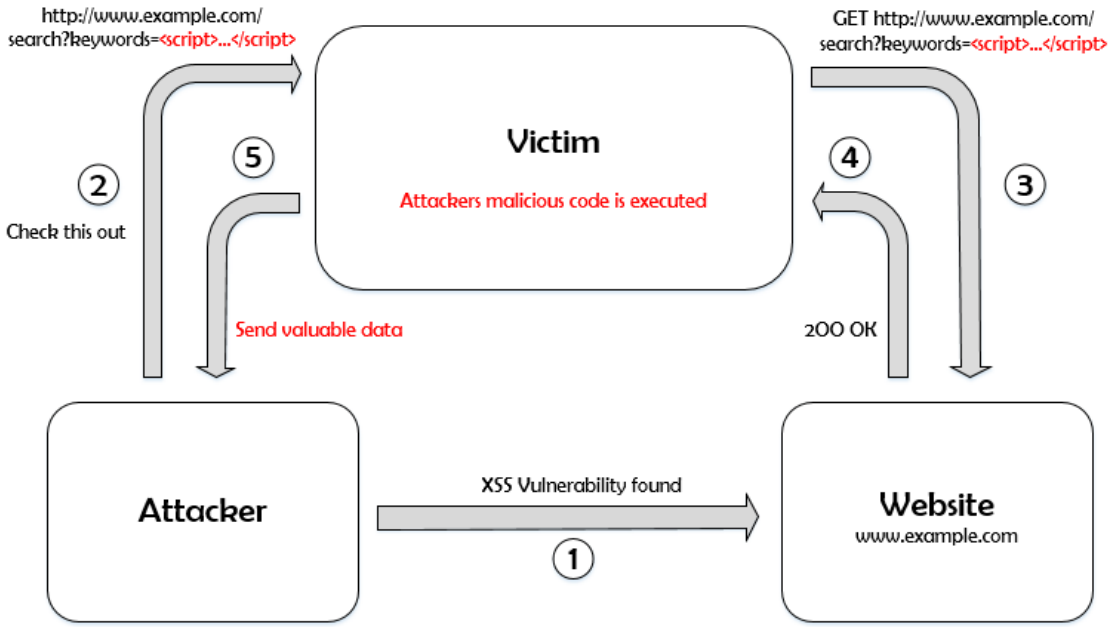
3.3 Yleisimpiä hyökkäystyyppejä

3.3.1 Cross-Site Scripting

Cross-Site Scripting (XSS) on hyökkäystapa, jossa web-sovelluksen haavoittuvaan komponenttiin voidaan syöttää hyökkääjän valitsemaa haitallista koodia. Tällä hetkellä erilaiset XSS-hyökkäykset ovat yksi yleisimmistä web-sovelluksiin kohdistuvista hyökkäyksistä. Erilaiset XSS-hyökkäykset tulevatkin muodostamaan myös tulevaisuudessa erittäin suuren osan organisaatioita vastaan kohdistetuista sovellustason uhkista. XSS-hyökkäykset perustuvat usein web-sovelluksen lomakkeiden kentissä käyttäjän syötetietojen puutteelliseen validiointiin. Sovelluksen käyttäjää ei koskaan voida pitää luotettuna, jolloin ei myöskään voida luottaa hänen

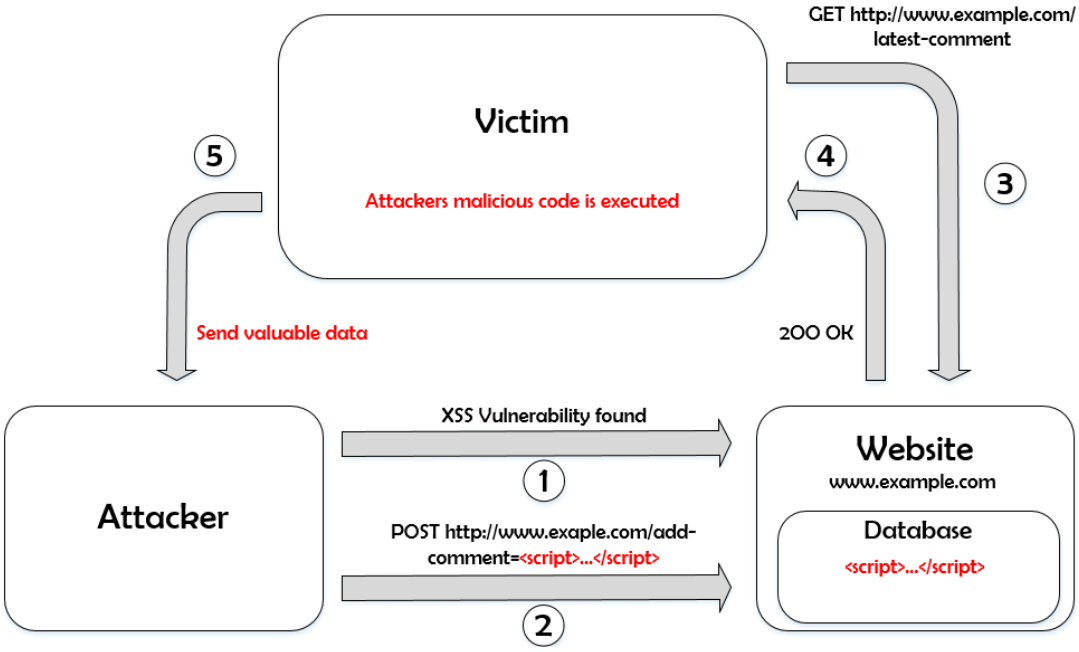
web-sovellukseen kohdistamiinsa syötteisiin. Yhtenä esimerkkinä syötteiden puutteellisesta validioinnista on se, että käyttäjällä on mahdollista syöttää web-sovelluksen lomakkeen kenttään JavaScript-koodia. (Lehtinen, 2015. 19-20.)

XSS-hyökkäykset voidaan jakaa niiden toteutustavan perusteella kahteen erilliseen kategoriaan: heijastettuihin ja tallennettuihin XSS-hyökkäyksiin. Heijastetussa hyökkäyksessä hyökkääjä käyttää hyväkseen web-sovelluksen lomakekentän puutteellista rakenteellista toteutusta ja pystyy näin ollen syöttämään sinne omaa haitallista koodiaan. Haitallista koodia ei tallenneta kohdesivustolle, vaan se heijastetaan käyttäjän selaimeen. Heijastettu XSS-hyökkäys ei yleensä itsessään ole vakava tietoturvauhka, mutta usein sitä käytetään osana laajempaa hyökkäystä. Hyökkääjä voi esimerkiksi lähettää sähköpostin välityksellä kohdehenkilölle linkin, joka sisältää hyökkääjän omaa haitallista koodia. Kohdehenkilön klikatessa linkkiä haitallinen koodi ajetaan hänen selaimessaan. Haitallinen koodi voi esimerkiksi ohjata uhrin niin sanotulle kalastelusivulle, joka vastaa täysin alkuperäistä sivua. Uhrin syöttäessä tunnuksensa kalastelusivulle, siirtyvät ne hyökkääjän haltuun. Toinen tyypillinen heijastetun XSS-hyökkäyksen avulla toteutettu hyökkäys on kohdehenkilön evästeiden varastaminen. Kokonaisuudessaan heijastetut XSS-hyökkäykset ovatkin yleinen ja suosittu tapa toteuttaa erilaisia tietojenkalasteluyrityksiä. Kuviossa 8 on havainnollistettu yksinkertaisten heijastetun XSS-hyökkäyksen toimintamallia. (Benji 2015.)



Kuvio 8. Heijastetun XSS-hyökkäyksen toimintaperiaate

Tallennetussa XSS-hyökkäyksessä hyökkääjä asettaa haavoittuvalle web-sivustolle pysyvästi omaa koodiaan, joka tallennetaan yleensä sivuston käyttämään erilliseen tietokantaan. (Ks. kuvio 9.)



Kuvio 9. Tallennetun XSS-hyökkäyksen toimintaperiaate

Lähes poikkeksetta web-sovellusten toteutuksien yhteydessä käytetään jotakin erillistä tietokantaa. Tämä mahdollistaa tallennettujen XSS-hyökkäysten yleisyyden. Tallennetussa XSS-hyökkäyksessä hyökkääjä pystyy hyödyntämään muun muassa web-sivuston kommenttikenttiä, foorumin keskustelupalstoja ja sivuston muokattavia henkilökohtaisia profiileja. Haitallinen koodi tallennetaan sovelluksen käyttämään tietokantaan ja se ajetaan aina, kun joku lataa selaimeensa haitallisen koodin sisältämän sivun.

3.3.2 Injektio

Erilaiset injektiot ovat eräitä yleisimpiä web-sovelluksiin kohdistuvia uhkia XSS-hyökkäyksien ohella. Kriittisimpiä web-sovelluksen yksittäisiä kohteita ovat käyttäjäsyötteet. Käyttäjän syötteisiin ei pidä koskaan luottaa, sillä syötetty tieto voi olla käytännössä mitä tahansa. Injektiohyökkäykset perustuvat asiakas-palvelin-arkkitehtuurissa asiakkaalta tulevien syötteiden virheelliseen tai puutteelliseen validiointiin. Tämä avaa mahdolliselle hyökkääjälle tilaisuuden syöttää web-

sovelluksen lomakkeiden kenttiin omaa haitallista koodiaan. Tällöin puhutaan järjestelmäkyselyn rakenteen muutoshyökkäyksestä eli injektiosta. (Saarinen 2014, 21.)

Dynaamisten web-sovellusten yhteydessä käytetään yhä useammin datan säilömiseen jotakin erillistä tietokantarajapintaa. Tämä on yksi niistä suurimmista yksittäisistä syistä miksi injektiohyökkäyksien määrä on kasvanut vuosi vuodelta. Kaikista injektiotyypeistä yleisin on tietokantoihin kohdistuva SQL-injektio. Muita yleisiä injektioita ovat muun muassa LDAP-injektio ja XPath-injektio.

Injektiohyökkäysten tavoitteena on saada oma syöte tulkituksi osaksi ohjelmaa ja tällä tavoin muuttaa ohjelman toimintaa. Injektiohyökkäys kohdistuu aina tiettyyn järjestelmän osaan. Tyypillisesti hyökkäys koostuu kolmesta erillisestä vaiheesta. Ensimmäisessä vaiheessa suoritetaan esitutkimus, jossa pyritään tutkimaan web-sovelluksen yhteydessä käytettävät tekniikat. Tämä onnistuu esimerkiksi tulkitsemalla sivuston virheilmoituksia, lähdekoodia tai käyttämällä siihen erityisiä työkaluja. Kun tarvittavat esitiedot on saatu kaivettua esiin, siirrytään seuraavaan vaiheeseen, jossa tulkitaan web-sovelluksen syötekenttiä, ja erityisesti sitä, mitä syötteitä käyttäjällä on mahdollisuus kenttiin syöttää. Viimeisessä ja varsinaisessa hyökkäysvaiheessa hyödynnetään niitä syötteitä, joilla pystytään muokkaamaan ja vaikuttamaan itse sovelluksen toimintaan. (Lehtonen & Siljander 2010, 7.)

Injektiohyökkäys mahdollistetaan usein puutteellisesti tai virheellisesti toteutetulla käyttäjän syötteiden tarkastuksella. Esimerkkinä web-sovelluksen lomakkeeseen toteutetun kirjautumisen tietokantakysely:

```
userName = getRequestString(”Username”);
userPassw = getRequestString(”Password”);
sql = ’SELECT * FROM Users WHERE Name =”’ + userName ’” AND
Pass =”’ + userPassw + ’””
```

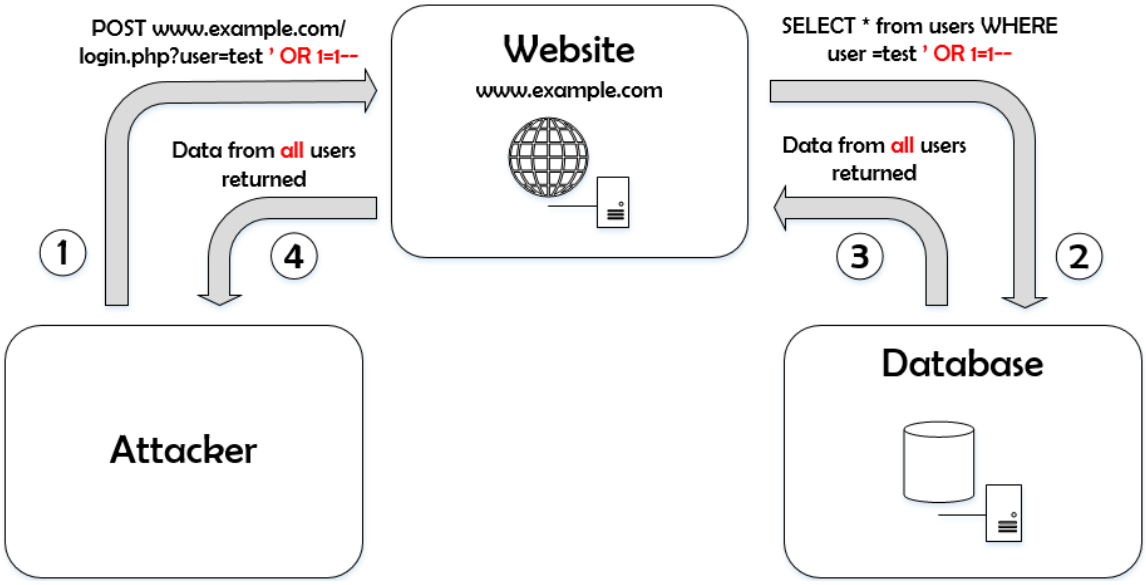
Normaalissa kirjautumisessa lomakkeen avulla haetaan tietokannasta käyttäjän syöttämää käyttäjätunnusta ja salasanaa. Näiden perusteella muodostuu seuraava SQL-kysely:

```
SELECT * FROM Users WHERE Name =”testiuser” AND Pass =”qwerty”;
```

Hyökkääjä voi yrittää hyödyntää kyseisen SQL-kyselyn yhteydessä “”=”” merkintää, joka on aina tosi. Tällöin lomakkeen kenttiin syötetään arvot “ or “”=”. SQL-kyselylausekkeeksi muodostuu tällöin seuraava tietokantakysely, joka on aina tosi:

```
SELECT * FROM Users WHERE Name =”” or “”=”” AND Pass = “” or
“”=””;
```

Kyselyn avulla hyökkääjä voi saada haltuunsa tietokannan kaikki käyttäjätunnukset ja niihin liitetyt salasanat. Kuviossa 10 on havainnollistettu kuvallisen esimerkin avulla tyypillisen SQL-injektion toimintamallia.



Kuvio 10. SQL-injektion toimintaperiaate

Esimerkissä web-sovelluksen lomakkeen kentässä välitetään POST-kyselyn yhteydessä haitallinen SQL-lauseke. SQL-kysely välitetään web-sovelluksen yhteydessä toimivalle tietokannalle, joka palauttaa haitallisen lausekkeen kysymät tiedot.

3.3.3 Rikkoutunut käyttäjän autentikointi ja istunnonhallinta

Käyttäjän autentikoinnilla tarkoitetaan tilannetta, jossa käyttäjä pyrkii todistamaan oman identiteettinsä ja saamaan tätä kautta käyttöönsä hänelle myönnetyt oikeudet ja valtuudet. Tyypillisesti tämä tapahtuu erillisen käyttäjätunnuksen ja salasanan avulla. Vastaavasti istunnonhallinnalla tarkoitetaan tilannetta, jossa asiakas ja palvelin ovat vuorovaikutuksessa keskenään onnistuneen autentikoinnin jälkeen.

Istunnonhallintaa ylläpidetään tavallisesti antamalla käyttäjälle oma uniikki istuntotunniste. Istuntotunnistetta voidaan kuljettaa evästeissä, lomakkeiden näkymättömissä kentissä tai web-sovelluksen generoimissa linkeissä. (Lehtinen 2015, 16.)

Rikkoutuneella autentikoinnilla ja istunnonhallinnalla tarkoitetaan web-sovelluksen rakenteessa olevaa haavoittuvuutta, jonka avulla hyökkääjällä on mahdollisuus ohittaa tai kaapata web-sivuston autentikointi. Kyseiset haavoittuvuudet ovat vakavia, ja niiden monimutkaisesta toteutuksesta johtuen yleensä hankalasti korjattavia.

Esimerkkihyökkäyksessä käyttäjän istuntotunniste siirtyy viestin mukana vääriin käsiin. Esimerkkisivuston haavoittuvuudesta johtuen se tukee URL:n uudelleenkirjoitusta ja siihen liitetään mukana istuntotunniste. Käyttäjä on kirjautunut sisään verkkokauppaan omilla tunnuksillaan, jotka pitävät sisällään myös käyttäjän luottokorttitiedot. Käyttäjä löytää verkkokaupasta mieleisensä kannettavan tietokoneen ja päättää kysyä ostokseen ystävänsä mielipidettä. Jakaessaan linkin, käyttäjä jakaa samalla ystävälleen oman salaisen istuntotunnisteensa.

```
http://www.example.com/sales/items_jsessionid=20C49SDKT65JT4B?i
tem=laptop15
```

Linkin avulla käyttäjän ystävä pystyy käyttämään lähettäjän istuntoa ja tekemään ostoksia hänen luottokortillaan.

Toisessa esimerkissä tietylle sivustolle on asetettu puutteellinen istunnon aikakatkaisu. Henkilö käyttää kirjastossa julkista tietokonetta, joka on vapaasti asiakkaiden käytettävissä. Henkilö kirjautuu web-sivustolle omilla tunnuksillaan. Poistuessaan hän ei kuitenkaan kirjaudu sivustolta ulos, vaan ainoastaan sulkee käytettävän selaimen. Muutaman tunnin kuluttua uusi käyttäjä tulee koneelle ja vierailee samalla sivustolla. Tällöin uusi käyttäjä pystyy käyttämään sivustoa edellisen henkilön tunnuksilla.

3.3.4 Turvaton suora objektiviittaus

Turvattomalla suoralla objektiviittauksella tarkoitetaan web-sovelluksen haavoittuvuutta, jossa sovellus käyttää suoraa viittausta sovelluksen sisäisesti

käyttämään tietoon tai resurssiin. Tällöin esimerkiksi tietokantoihin perustuvat tiedot voivat paljastua käyttäjälle selaimen tekstikentässä näkyvien parametrien arvoina.

Esimerkkihyökkäyksessä web-sivustolla on käytössä käyttäjän henkilökohtainen profiili, jota voidaan muokata ja kustomoida käyttäjän haluamalla tavalla. Käyttäjän profiili identifioidaan selaimen tekstikentässä parametrilla *profileid*, jota seuraa profiilin uniikki tunnistenumero.

```
http://www.example.com/profiles/edit?profileid=3456
```

Mikäli sivusto on haavoittuvainen ja mahdollistaa suoran objektiviittauksen, on hyökkääjällä mahdollisuus muokata *profileid*-parametrin arvoa ja päästä tämän kautta muokkaamaan muiden henkilöiden profiileja.

3.3.5 Tiedoston solutus

Tiedoston solutus (engl. *file inclusion*) on hyökkäystapa, joka hyödyntää muiden hyökkäyksien tapaan web-sovelluksen rakenteellista ja toiminnallista virhettä. Hyökkäys mahdollistaa sekä tiedostojen lataamisen ja suorittamisen kohdepalvelimella sekä palvimen arkaluontoisten tiedostojen lukemisen. Solutushyökkäykset ovat karkeasti jaoteltavissa kahteen erilliseen kategoriaan.

Paikallisen tiedoston solutus (engl. *local file inclusion*) -hyökkäystapa mahdollistaa kohdepalvelimen omien tiedostojen lukemisen ja joskus myös niiden suorittamisen. LFI-hyökkäys mahdollistetaan, mikäli web-sovelluksen lomakkeen kenttä on toteutettu puutteellisesti. Mikäli myös kohdepalvelimen web-palvelua ajetaan korkeilla käyttöoikeuksilla, on hyökkääjällä mahdollisuus päästä käsiksi sen arkaluontoisiin tietoihin. Esimerkkitapauksessa hyökkääjä hyödyntää web-palvelun väärin toteutettua lomakkeen kenttää *picture*-parametrin arvossa. Hyökkääjä syöttää parametrin arvoksi Linuxin salasanojen taltiointiin käytettävän tiedostosijainnin ”*/etc/passwd*”. Onnistuessaan komento palauttaa hyökkääjälle koko passwd-tiedoston sisällön.

```
http://example.com/index.php?picture=../../../../../etc/passwd
```

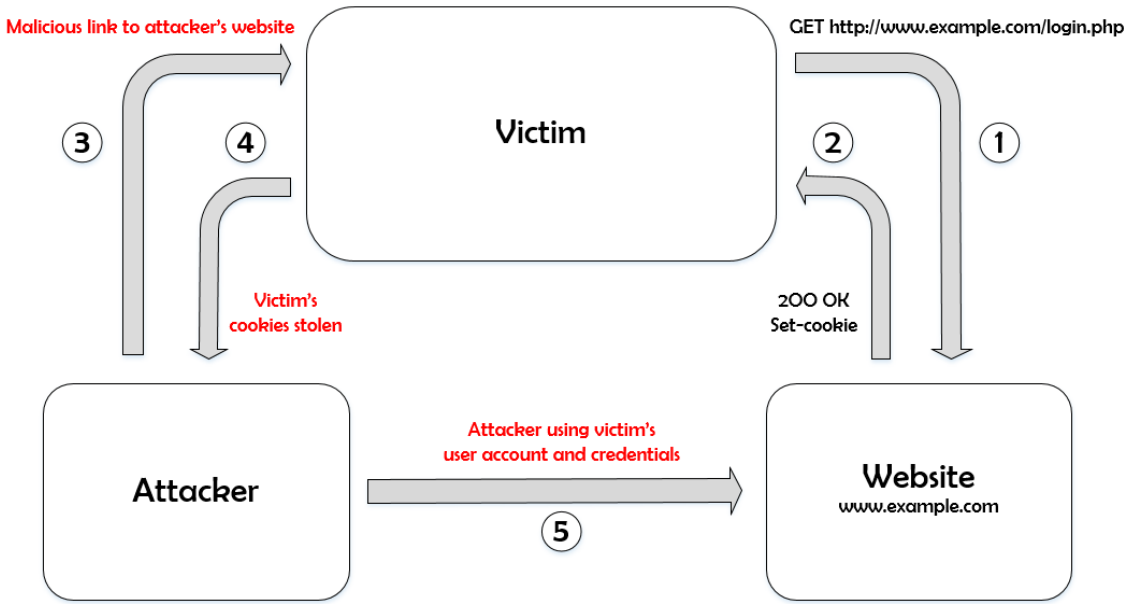
Etätiedoston solutus (engl. *remote file inclusion*) hyökkäysmetodi vastaa toiminnaltaan läheisesti LFI-hyökkäystä, mutta tässä kohdepalvelimeen on

esimerkiksi mahdollista siirtää paikallisia tiedostoja tai kohdepalvelin on mahdollista ohjata suorittamaan tiedosto toiselta palvelimelta. Tällaiset hyökkäykset ovat kuitenkin nykyään harvinaisia, sillä useimmat nykyaikaiset PHP-ympäristöt kieltävät suorittamasta koodia, joka sijaitsee eri palvelimessa (Karvi, 53).
Esimerkkihyökkäyksessä haavoittuva *picture*-parametri mahdollistaa hyökkääjälle paikallisen haitallisen tiedoston siirtämisen kohdepalvelimelle.

```
http://example.com/index.php?picture=C:\\ftp\upload\exploit
```

3.3.6 Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) -hyökkäyksissä hyökkääjän tavoitteena on saada suoritettua koodia tietyssä sivustossa uhrin puolesta. CSRF-hyökkäykset muistuttavat jossakin määrin XSS-hyökkäyksiä, mutta tässä koodia on tarkoitus suorittaa uhrin puolesta. CSRF-hyökkäyksessä käytetään hyväksi selaimen ja kohdepalvelun välistä tilatietoa, joka voi olla esimerkiksi selaimen evästeessä oleva istuntotunniste. Useiden web-sovellusten toteutuksissa käyttäjät yksilöidään ainoastaan palvelinevästeiden ja istuntotunnisteiden avulla. Tällöin hyökkääjälle riittää pääsy näihin tietoihin, jolloin sivustolla voidaan suorittaa koodia uhrin identiteetillä. Joissain sivustoissa myös aikakatkaisut on jätetty määrittelemättä tai ne on määritelty hyvin suuriksi, jopa viikoiksi, jolloin hyökkääjällä on rutkasti aikaa suorittaa sivustolla omaa haitallista toimintaansa. Kuviossa 11 on havainnollistettu tilannetta, jossa hyökkääjä saa varastettua uhrin ja web-palvelimen välisen istuntotunnisteen.



Kuvio 11. CSRF-hyökkäyksen toimintaperiaate

Varastetun istuntotunnisteen avulla hyökkääjä voi käytännössä suorittaa kohdesivustolla mielivaltaisia toimintoja uhrin puolesta.

3.4 Hyökkäystyyppien kehittyminen

Nykypäivän hyökkäystyypit ja niihin käytetyt menetelmät eivät pysy muuttumattomina, vaan ne kehittyvät jatkuvasti. Tämä tekee niistä entistä vaikeammin havaittavia ja torjuttavia jo olemassa olevilla järjestelmillä. Lisäksi web-sovellusten yhteydessä käytettävät uudet teknologiat avaavat aina mahdollisuuksia uusille hyökkäysvektoreille. Suhteellisen nuoret *websocket*-protokollaa hyödyntävät toteutukset ovat yksi esimerkki tapauksista, jotka mahdollistavat uusien hyökkäysvektoreiden kehittämisen ja niiden käytännön implementoinnin (Fagerlund 2014, 39).

Web-sovellusten tietoturvan yleisen kehittymisen myötä kustomoidut, piilotetut hyökkäyslausekkeet ovat kasvattaneet suosiotaan. Näissä hyökkäyksissä hyökkäyslausekkeita pyritään piilottamaan esimerkiksi koodaamalla ne aluksi johonkin toiseen muotoon. Menetelmällä pystytään kiertämään joitakin web-sovelluksen suojausjärjestelmiä. Koodaus on myös mahdollista tehdä kahteen kertaan, jolloin niiden havaitsemisesta tulee entistä vaikeampaa (Enrique 2013).

Esimerkkinä yksinkertainen JavaScript-koodia hyödyntävä XSS-lauseke, joka on muotoa:

```
<script>alert('XSS Attack')</script>
```

Kuitenkin useissa tapauksissa merkit: “<”, “>” ja “/” ovat kiellettyjä web-sovelluksen lomakkeen kentässä ja tällöin näitä merkkejä käyttävät hyökkäyslausekkeet suodatetaan automaattisesti. Joissakin tilanteissa rajaus voidaan kuitenkin kiertää käyttämällä kyseisten merkkien koodattuja muotoja. Näissä tapauksissa kyseiset merkit ovat niiden heksakoodatussa muodossa, joka riittää jossain tapauksissa kiertämään sovelluksen oman suojauksen. Esimerkkinä XSS-lauseke, jossa edellä mainitut erikoismerkit ovat niiden koodatussa muodossa.

```
%3Cscript%3Ealert('XSS Attack')%3C%2Fscript%3C
```

Suhteellisen uusi haavoittuvuus on niin sanottu *shellshock*-haavoittuvuus. Kyseisen haavoittuvuuden avulla hyökkääjä voi mahdollisesti ottaa haltuunsa päivittämättömän Linux Bash -komentotulkin avulla hyökkäyksen kohteen käyttämän järjestelmän kokonaan. Kyseinen haavoittuvuus on tietojen perusteella ollut olemassa jo yli 25 vuotta, mutta se löydettiin vasta muutama vuosi sitten. (Bash-komentotulkin Shellshock-haavoittuvuus mahdollistaa laajan hyväksikäytön 2014.)

Vuoden 2016 Black Hat -konferenssissa keskusteltiin uudesta HEIST-hyökkäyksestä (HTTPS Encrypted Information can be Stolen through TCP-windows). HEIST-hyökkäyksessä pystytään pääsemään SSL/TLS-salatun liikenteen väliin ja näin ollen käsiksi mahdollisesti arkaluontoiseen tietoon, kuten luottokorttinumeroihin ja henkilötietoihin. Tässä monivaiheisessa hyökkäyksessä HTTPS-suojatulle sivustolle asetetaan JavaScript-tiedosto, jonka avulla saadaan tietoon salattujen tiedostojen tarkka koko. (Masters 2016.)

4 Web-sovelluksien tietoturva

4.1 Yleistä

Web-sovellusten tietoturvan osalta on pitkään vallinnut oletus, että tietoverkkojen jo olemassa oleva suojaus sekä ohjelmistopuolen oma sisäinen suojaus olisivat täysin riittäviä myös modernien ja kehittyneiden sovellustasolla tapahtuvien uhkien torjumiseen. Oletus on kuitenkin väärä. Uudet kehittyneet hyökkäystyypit ja -menetelmät vaativat jatkuvasti uusien suojausmenetelmien kehittämistä ja niiden jatkuvaa ylläpitoa. Yleisimpien tietoturvariskien tunnistaminen ja niiden estämiseksi suoritettavat toimenpiteet eivät enää ole riittävä keino kattavan suojauksen toteuttamiseen. On pystyttävä näkemään pintaa syvemmälle ja ennakoimaan myös tulevia uusia, kustomoituja ja piilotettuja uhkia. Web-sovellusten tietoturvan kehittämisen suurimmiksi haasteiksi ja ongelmakohdiksi määritellään usein tietoturvan toteuttamisen monimutkaisuus, kalleus ja siitä saatavien suoranaisten hyötyjen vähäisyys. Tietoturvan ei voida kovinkaan usein katsoa tuovan suoria voittoja ja hyötyjä organisaatiolle. Tietoturva tulisikin siksi nähdä ennemminkin välttämättömänä organisaation ydinliiketoimintaa tukevana toimintona.

Web-sovellusten tietoturva kattaa monta erilaista osakokonaisuutta. Tyypillisesti uskotaan, että ainoa web-sovellukseen liittyvä haavoittuva komponentti on sovelluksen pohjalla toimiva lähdekoodi. Tämä ei kuitenkaan pidä paikkaansa. Haavoittuvia osia voivat olla myös esimerkiksi selaimen ja palvelimen välinen yhteys, verkko- tai alustapuolen haavoittuvuus tai selaimen tai sen liitännäisen tietoturva-aukko**.** Tällöin onkin ehdottoman tärkeää huomioida tietoturva jokaisessa web-sovellukseen liittyvässä osakomponentissa.

Organisaatioissa salatun liikenteen määrä on jatkuvassa kasvussa. Salattua liikennettä ei kuitenkaan voida aina pitää tietoturvan kannalta optimitilanteena. Salattu liikenne asettaa aina omat haasteensa tietoturvan huomioimiselle. Sen johdosta liikenteeseen menetetään se näkyvyys, jota vaaditaan tietoturvauhkien tehokkaaseen havainnointiin ja torjuntaan. Tietoturvayhtiö Gartnerin mukaan vuoteen 2017 mennessä jopa yli 50 prosenttia kaikista kyberhyökkäyksistä käyttää hyödykseen jollain tapaa salattua tietoliikennettä (Vijayan 2016). Salatun liikenteen

osalta on pystyttävä löytämään tarvittavat keinot, joilla salattu liikenne pystytään purkamaan ja käsittelemään hallitusti verkon keskitetyssä solmukohdassa.

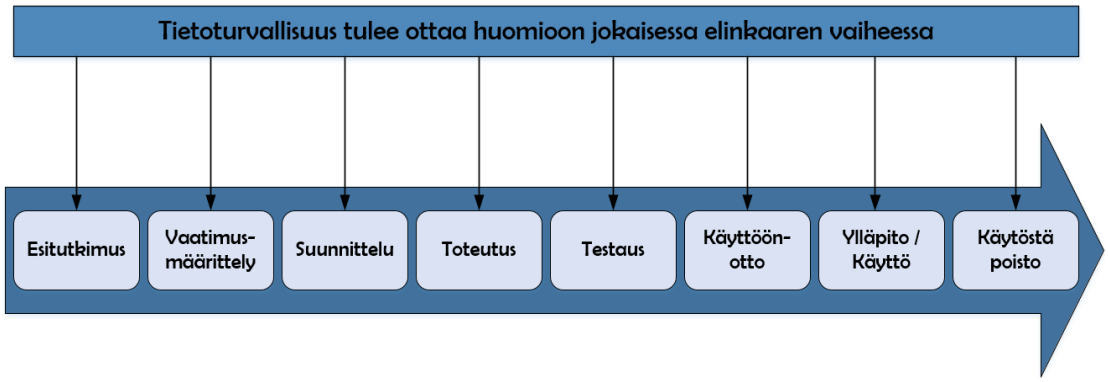
4.2 Web-sovelluskehitys ja tietoturvastrategia

Tietoturvan huomioiminen web-sovelluskehityksessä on organisaatioissa lähes poikkeuksetta ainakin osittain puutteellisella tasolla. Suurimpia yksittäisiä syitä tälle ovat web-sovelluksien suunnittelu- ja kehitysprojektien tiukat aikataulut ja niiden rajallinen budjetti. Lisäksi ongelmia aiheuttaa usein epäselvyys siitä, kuka itse asiassa on vastuussa web-sovellusten tietoturvasta. On hyvin tavallista, että vastuu tietoturvasta jätetään ohjelmistokehittäjille tai jollekin muulle organisaation osalle. Todellisuudessa kyse on yhteispelistä. Web-sovellusten tietoturvassa on kyettävä omaksumaan laajempi näkökulma ja jakamaan vastuuta useille eri osa-alueille. Sama periaate on heijastettavissa myös pilvipalveluihin. Usein uskotaan, että pilvipalveluiden tarjoaja on yksin vastuussa asiakkaan pilvessä olevien palveluiden tietoturvasta. Todellisuudessa vastuu on myös osittain pilvipalvelut hankkineella asiakkaalla. Tietoturvallisuuden kannalta organisaatioilla on yleensä käytössä erillinen tietoturvan asiantuntijaryhmä, joka tarjoaa konsultoivaa apua kaikissa tietoturvaan liittyvissä asioissa (VAHTI 1/2013, 16).

Ohjelmistokehityksen prosesseissa tietoturvan kannalta yhdeksi merkittävimmistä kohdista mielletään usein ohjelmiston testaus ennen sen varsinaista käyttöönottoa. Testauksen tavoitteena on usein arvioida ohjelmiston laatua, ominaisuuksia ja suorituskykyä sekä pyrkiä löytämään siitä mahdolliset ongelmakohdat ja haavoittuvuudet. Testien perusteella sovelluksen kehittäjille laaditaan erillinen raportti, jonka pohjalta he tekevät sovellukseen tarvittavat suorituskykyä ja tietoturvaa parantavat muutokset. On hyvin tyypillistä, että nämä ohjelmistokehityksessä suoritettavat tietoturvatestit jätetään usein projektien viime metreille, jolloin siitä kärsivät sekä suoritettava testaus että sen vaikutuksena itse sovellus. Testit ovat usein myös kertaluontoisia testauksia, jolloin sovelluksen muuttuessa sen elinkaarensa aikana testeistä saadun informaation merkittävyys laskee huomattavasti. Lisäksi haavoittuvuuksien osalta kertaluontoisten tietoturvatestien tuloksena tehdyt korjaukset suojaavat vain yleisesti tunnetuilta haavoittuvuuksilta. Uudet vielä tuntemattomat hyökkäystyypit eli niin sanotut

nollapäivähaavoittuvuudet ovat yhä olemassa sovelluksessa. Tässä suhteessa testien säännöllinen tiheä toistettavuus parantaisi huomattavasti mahdollisuuksia löytää myös nopeasti testien välillä ilmenneet uudet haavoittuvuudet. Säännölliset tietoturvatestit ovat kuitenkin kalliita toteuttaa ja tämän vuoksi niitä yleensä vielä vierastetaankin eri yrityksissä ja organisaatioissa.

Valtiohallinnon tieto- ja kyberturvallisuuden johtoryhmä (VAHTI) on suosittelut tietoturvan huomioimista web-sovelluskehityksen jokaisessa elinkaaren vaiheessa aina esitutkimuksesta sovelluksen käytöstä poistoon. Tähän tavoitteeseen on kuitenkin hyvin vaikea päästä johtuen rajallisesta ajasta, budjetista ja tietotaidosta. Kuviossa 12 on kuvattu VAHTI-ohjeen sovelluskehityksen tietoturvaohjeen tietoturvakehystä, joka painottaa tietoturvan huomioimista sovelluskehityksen jokaisen elinkaarivaiheen aikana.



Kuvio 12. Sovelluskehityksen tietoturvakehys (VAHTI 1/2013, 43, muokattu)

Tietoturvastrategiassa on tärkeää laatia erillinen toimintasuunnitelma, jonka perusteella osataan toimia oikein tietoturvauhkien tapahtuessa. Tämän tueksi tarvitaan henkilöstöllisiä resursseja tietoturvauhkien havainnoimiseksi sekä järjestelmiä niiden havainnoimisen helpottamiseksi. Web-sovellukseen kohdistuvien hyökkäysten havainnoimisen jälkeen on osattava tunnistaa komponentti, jossa hyökkäys tapahtui sekä hyökkäyksen mahdolliset vaikutukset. Selkeät toimintaprosessit uhkien varalle ovat näin ollen olennainen osa niiden ennaltaehkäisyä sekä toiminnan loogisuutta hyökkäysten aikana ja niiden jälkeen.

4.3 Open Web Application Security Project

OWASP (Open Web Application Security Project) on voittoa tavoittelematon järjestö, jonka tarkoituksena on edesauttaa tietoturvallisten web-sovellusten suunnittelua, kehittämistä ja ylläpitoa. Järjestö on pannut liikkeelle useita erilaisia projekteja. Näistä projekteista yleisesti tunnetuin lienee OWASP Top 10 -lista, johon listataan kymmenen web-sovellusten yleisintä haavoittuvuutta. Lista julkaistaan kolmen vuoden välein ja sen sisältämä informaatio on kerätty monipuolisesti eri yritysten ja organisaatioiden omista tilastoista. (OWASP Top 10 2013.) Tällä hetkellä OWASP:in uusin Top 10 -listaus on vuodelta 2013, ja uusi listaus on tarkoitus julkaista hieman aikataulusta jäljessä vuoden 2017 alkupuolella. OWASP on noussut suhteellisen nopealla tahdilla yhdeksi web-sovellusten tietoturvan kehittämisen kulmakivistä, ja se tunnistetaankin jo yleisesti yrityksissä ja organisaatiossa.

Varsinainen OWASP Top 10 -listaus tarjoaa jokaisesta yleisimmästä haavoittuvuudesta sen kuvauksen, käytännön esimerkin haavoittuvuuden hyväksikäytöstä sekä ohjeen, kuinka haavoittuvuudelta voidaan suojautua tehokkaasti. Taulukossa 3 on lueteltuna vuoden 2013 OWASP Top 10 haavoittuvuudet.

Taulukko 3. OWASP Top 10 2013 haavoittuvuudet

LUOKKA	HAAVOITTUVUUS
A1	Injektio
A2	Rikkoutunut autentikointi ja istunnonhallinta
A3	Cross-Site Scripting (XSS)
A4	Turvaton suora objektiviittaus
A5	Virheellinen konfigurointi
A6	Arkaluontoisen tiedon paljastuminen
A7	Puuttuva funktiotason pääsynvalvonta
A8	Cross-Site Request Forgery (CSRF)
A9	Tunnettuja haavoittuvuuksia sisältävien komponenttien käyttö
A10	Varmentamattomat uudelleenohjaukset ja välitykset

Verrattuna tätä edeltäneeseen OWASP-listaukseen vuodelta 2010, on lista pystynyt pääosin samana. Ainoastaan muutama listauksen haavoittuvuus on vaihtanut

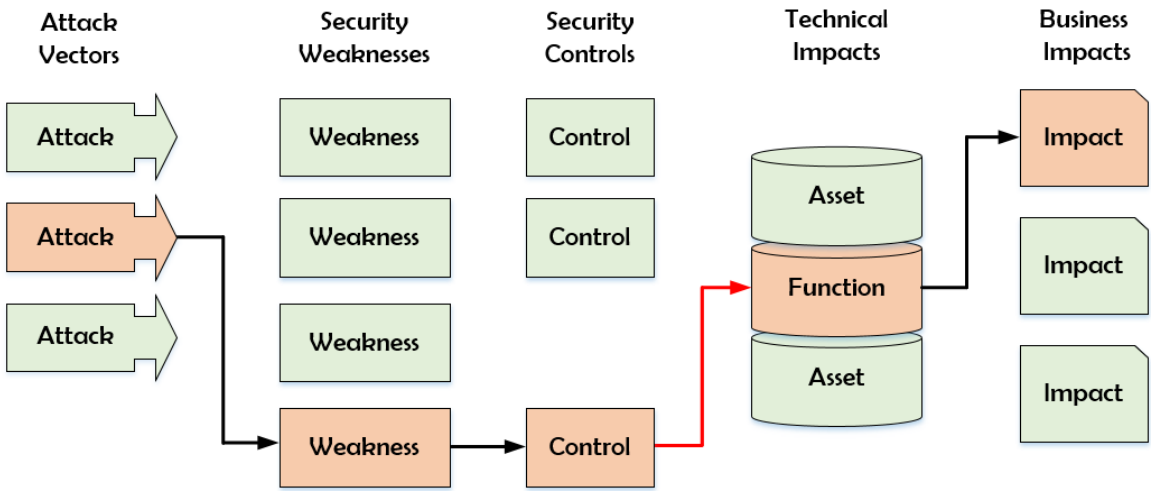
paikkaa ja osa vuoden 2010 listauksen haavoittuvuuksista on 2013 listauksissa yhdistetty yhdeksi haavoittuvuudeksi.

OWASP:in Top 10 listauksen ohella yksi sen tärkeimmistä projekteista on web-sovelluksiin kohdistuvien riskienhallinta. OWASP käyttää web-sovelluksiin kohdistuvien riskien luokitteluun eräänlaista luokittelutaulukkoa. Sen avulla jokaisesta web-sovellukseen kohdistuvasta riskistä voidaan muodostaa kokonaiskuva, joka muodostuu useasta erillisestä riskin osatekijästä. Tekijöitä ovat muun muassa riskin levinneisyys, sen hyödynnettävyys, havainnollistettavuus sekä sen teknisistä- ja liiketoiminnallisista vaikutuksista. Taulukossa 4 on listattuna OWASP-säätiön riskien luokitteluun käytettävät kategoriat.

Taulukko 4. OWASP riskien luokittelu (OWASP Top 10 - 2013, muokattu)

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Techincal Impacts	Business Impacts
App Specific	Easy	Widespread	Easy	Severe	App / Business Specific
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

OWASP painottaa riskien yhteydessä niiden toteuttamisen mahdollistavia erilaisia polkuja. Kuvion 13 kaaviossa on havainnollistettu tätä kyseistä periaatetta.



Kuvio 13. OWASP riskien vaikutukset (OWASP Risk 2013, muokattu)

Erilaiset hyökkäysvektorit mahdollistavat kohdejärjestelmän haavoittuvuuksien hyödyntämisen ja hyväksikäytön. Mikäli nämä hyökkäykset läpäisevät myös tietyt

tietoturvakontrollit, saavat ne aikaan kohdejärjestelmässä tiettyjä vaikutuksia. Vaikutukset on luokiteltu kahteen kategoriaan: teknisiin ja liiketoiminnallisiin vaikutuksiin. Usein tekniset vaikutukset johtavat epäsuorasti liiketoiminnallisiin vaikutuksiin. (OWASP Risk 2013.)

OWASP:n tarjoamien parhaiden käytänteiden ja yhtiön tarjoamien avoimeen lähdekoodiin perustuvien työkalujen käyttöönotto osaksi ohjelmistosuunnittelun kehitys- ja ylläpitoprosesseja on viime vuosina nostanut asemaansa suomalaisissa ja globaaleissa ohjelmistokehitysprosesseissa. OWASP tunnistetaan tällä hetkellä hyvin myös valtiotasolla. Valtiohallinnon tieto- ja kyberturvallisuuden johtoryhmän sovelluskehityksen tietoturvaohjeessa mainitaan seuraava:

Tietoturvaohjeistus (OSK-002): Organisaation tulee toteuttaa ja ylläpitää listaa dokumenteista, web-osoitteista jne. jotka tarjoavat tietoturvaohjeistusta organisaation käyttä- mille teknologioille. Resurssit voivat olla sisäistä ohjeistusta tai esimerkiksi OWASP tai MSohjeistuksia. Uusien, sovelluskehityksestä vastuussa olevien työntekijöiden tulee tutustua kyseiseen ohjeistukseen perehdytyksen aikana. Listaa ylläpidetään ja päivitetään käytettyjen teknologioiden muuttuessa ja päivittyessä. Lista on organisaatiokohtainen, joten muiden organisaatioiden määrittelemät listat eivät sovellu tähän tarkoitukseen. (VAHTI 1/2013, 37.)

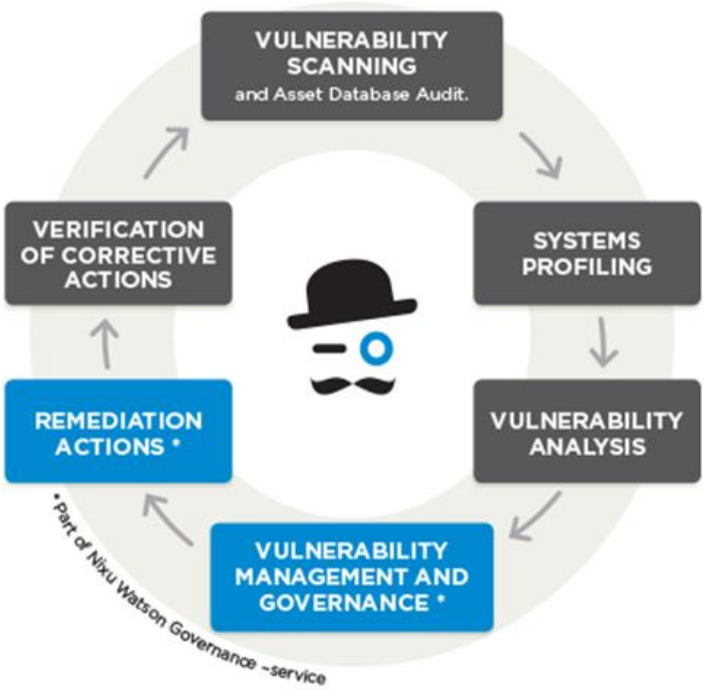
OWASP on saanut sen ylläpitämästä Top 10 -listauksesta myös kritiikkiä. Yksi OWASP:n ylläpitämän Top 10 listan kritiikeistä kohdistuu sen hitaaseen päivitysväliin. Kritiikin esittäjien mukaan listausta ei voida pitää ajantasaisena, mikäli siinä esiintyvä tieto on useamman vuoden vanhaa. Hyökkäystyypit ja -tekniikat kehittyvät jatkuvasti, jolloin ajantasaisen listan ylläpitäminen vaatisi nykyistä tiheämpää päivitysväliä. Toinen erityistä kritiikkiä saanut kohta on turvallisuusuhkien luokittelu. Esimerkiksi vuoden 2013 listassa A2-haavoittuvuus (rikkoutunut autentikointi ja istunnonhallinta) sisältää käytännössä useita erilaisia haavoittuvuuksia, eikä sitä näin ollen tulisi luokitella vain yhdeksi haavoittuvuudeksi. Kritiikkiä on saanut osakseen myös puutteellinen tarkistuslista. (Sethi 2013.)

Useat yritykset ja organisaatiot arvioivat listauksen kymmentä kohtaa vain muutamalla yleisimmällä mittarilla. Tällöin heille jää usein puutteellinen näkemys kokonaiskuvasta. OWASP on kuitenkin tiedostanut ongelman ja lähestynyt sitä

käynnistämällä sen pohjalta oman projektin, jonka tuloksena julkaistiin Application Security Verification Standard -listaus (ASVS). OWASP myös painottaa, ettei sen tarjoama Top 10 listaus ole yksin riittävä tietolähde web-sovellusprojektien tietoturvan ohjenuoraksi.

4.4 Teknisten haavoittuvuuksien hallinta

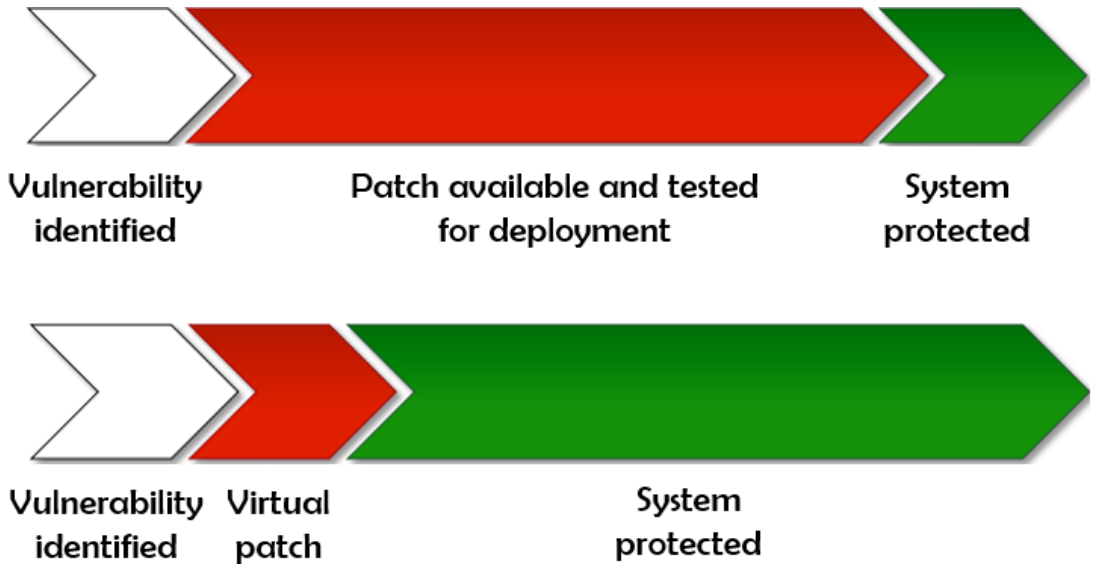
Haavoittuvuuksien hallinnalla tarkoitetaan prosesseja eli toimintatapoja, joiden avulla on tarkoitus selvittää organisaation tietojärjestelmien haavoittuvuudet, analysoida ne ja suorittaa niille tarvittavat toimenpiteet. Tietoturvakonsultointiyhtiö Nixun *Watson*-haavoittuvuuksienhallintaprosessia on kuvattu kuviossa 14.



Kuvio 14. Haavoittuvuuksienhallintaprosessi (Nixu Watson N.d)

Prosessi lähtee liikkeelle kohdejärjestelmän skannauksesta, jonka avulla on tarkoitus paikallistaa siinä mahdolliset olevat haavoittuvuudet. Mikäli haavoittuvuuksia havaitaan, on tätä ensimmäistä vaihetta seuraava toimenpide haavoittuvuuksien profilointi ja analysointi. Tässä kohdassa arvioidaan haavoittuvuuden hyödynnettävyyttä ja vakavuutta. Analyysin pohjalta tehdään tarvittavat hallinnolliset päätökset, jonka perusteella suoritetaan tarvittavat korjaustoimenpiteet. Korjaustoimenpiteiden jälkeen varmistetaan niiden toimivuus.

Kuvion 14 haavoittuvuuksienhallintaprosessin korjaustoimenpiteiden suorittamiseen on käytössä web-sovellusten osalta useita erilaisia mahdollisuuksia. Mikäli haavoittuvuus havaitaan tuotantokäytössä olevasta sovelluksesta, yksi mahdollisuus on poistaa koko palvelu. Vaihtoehtoisesti sen haavoittuva komponentti voidaan poistaa käytöstä tuotantoympäristössä siihen asti, kunnes sen haavoittuvuus on korjattu ohjelmistotasolla. Kuitenkin etenkin kaupallisissa tuotteissa haavoittuvuuden korjaaminen voi viedä useita kuukausia, jonka ajan organisaation on selvittävä ilman kyseistä web-palvelua. Tästä aiheutuvat taloudelliset menetykset voivat kuitenkin olla estetekijä edellä mainitun ratkaisun toteuttamiselle. Toinen vaihtoehto on yksinkertaisesti jatkaa haavoittuvan ohjelmiston käyttöä ja ottaa tiedostettu riski mahdollisen hyökkäyksen aiheuttamista haittavaikutuksista. Haavoittuvuus tulee tietenkin korjata heti, kun siihen on mahdollisuus. Myös tässä tapauksessa ongelmaksi nousee korjaamiseen kuluva aika, mutta myös mahdollisesta onnistuneesta hyökkäyksestä aiheutuvat haitat organisaatiolle. Kolmas tapa sovelluksen haavoittuvuuden korjaamiseen on virtuaalinen paikkaus. (Ks. kuvio 15.)



Kuvio 15. Virtuaalinen paikkaus (Anders N.d, 14, muokattu)

Virtuaalisessa paikkauksessa estetään web-sovelluksen haavoittuvassa komponentissa tapahtuva haitallinen toiminta erillisellä suojausratkaisulla. Virtuaalisella paikkauksella voidaan luoda esimerkiksi väliaikainen suojausmekanismi uudelle haavoittuvuudelle, johon ei ole vielä saatavilla korjaavaa tietoturvapäivitystä. Virtuaalisen paikkauksen ehdottomiin etuihin kuuluvat sen nopea käyttöönotto,

tehokkuus ja joustavuus. Web-sovellus itsessään ei vaadi lainkaan muutoksia. Suojaus suoritetaan ainoastaan loogisesti tai fyysisesti sen edessä toimivalla, yleensä erillisellä palomuuriratkaisulla. Virtuaalisen paikkauksen suurin haittapuoli on se, että web-sovelluksessa oleva haavoittuvuus ei varsinaisesti korjaannu kyseisellä menetelmällä. Haavoittuvuuden hyväksikäyttö ainoastaan estetään lähteen ja kohteen väliin asetettavalla erillisellä järjestelmällä tai suojausmekanismilla.

4.5 Tietoturvamallit

Erilaisten tietoturvapolitiikkojen rakentamisessa puhutaan yleensä erillisistä tietoturvamalleista. Pääasiallisesti esille nousevat positiivinen ja negatiivinen tietoturvamalli. Uusimpana tulokkaana edellä mainittujen rinnalle on tullut myös niin sanottu hybridimalli, joka pyrkii yhdistelemään positiivista ja negatiivista tietoturvamallia.

Positiivisella tietoturvamallilla tarkoitetaan tietoturvan toteuttamistapaa, joka perustuu ennalta määritellyn, hyväksi havaitun liikenteen sallimiseen. Positiivisen tietoturvan yhteydessä puhutaan usein myös valkoisesta listasta (engl. *whitelist*). Valkoiseen listaan määritellään kaikki sallittu liikenne, kun vastaavasti kaikki muu liikenne kielletään (OWASP – Positive Security Model 2015). Positiivisen tietoturvamallin etuihin kuuluu sen tarjoama suoja. Kun lähtökohtana on sallia vain se, mitä halutaan sallia, on pohjalla vankka tietoturvapolitiikka. Positiivinen tietoturvamalli suojaakin näin ollen erinomaisesti muun muassa uusilta nollapäivähaavoittuvuuksilta. Positiivisen tietoturvamallin heikkouksiin kuuluu yleensä sen työläs ylläpitoprosessi. Kaikki uudet sallituksi haluttavat kohteet on lisättävä mallin määritellylle valkoiselle listalle.

Negatiivisella tietoturvamallilla tarkoitetaan vastaavasti tilannetta, jossa etukäteen määritelty, yleensä haitalliseksi katsottu liikenne estetään. Negatiivisen mallin yhteydessä puhutaan usein myös mustasta listasta (engl. *blacklist*). Mustalle listalle määritellään kaikki sellainen liikenne, mitä ei haluta sallia. Vastaavasti kaikki liikenne mitä ei kielletä, on sallittua (OWASP – Positive Security Model 2015). Negatiivisen tietoturvamallin vahvuuksiin kuuluu sen helppo ylläpidettävyys. Mallin apuna voidaan käyttää kolmannen osapuolen etukäteen luomia mustia listoja ja niitä

voidaan tarvittaessa täydentää itsenäisesti. Negatiivisen tietoturvamallin heikkoudet liittyvät uusien, vielä tuntemattomien uhkien torjumiseen. Koska tässä toteutusmallissa kaikki sallitaan, jota ei olla erikseen kielletty, voi aika ajoin esiintyä uusia haitallisia tunnisteita, joita ei olla vielä ehditty lisäämään tämän toteutusmallin mustalle listalle.

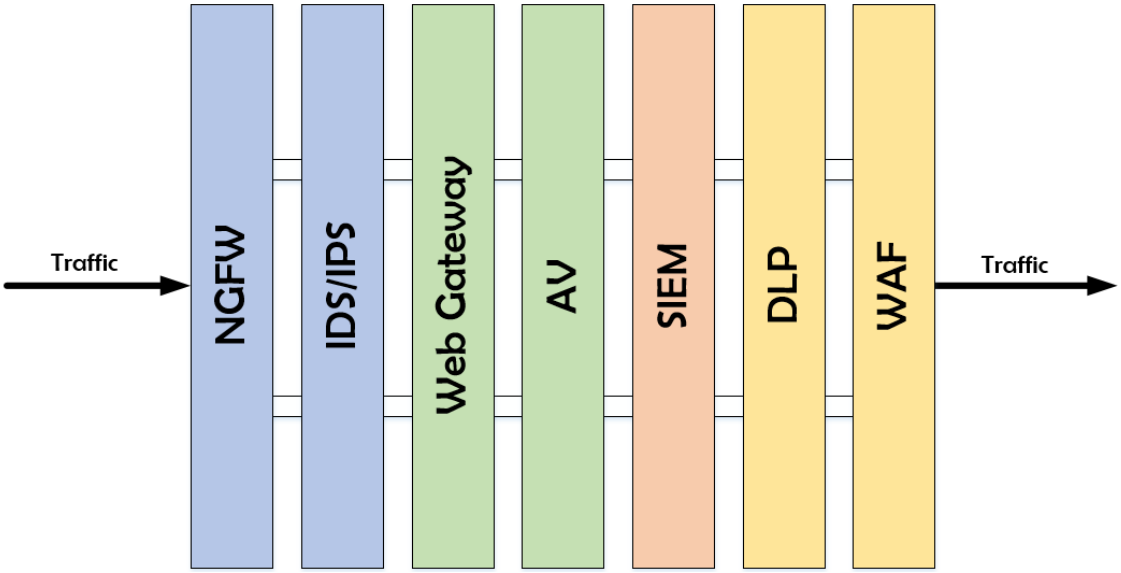
Positiivisesta ja negatiivisesta tietoturvamallista on lisäksi kehitelty erillinen hybridimalli. Hybridimallilla tarkoitetaan sellaista tietoturvamallia, jossa hyödynnetään sekä positiivista että negatiivista tietoturvamallia. Tässä toimintatavassa pyritään hyödyntämään molempien tietoturvamallien parhaita puolia ja vastaavasti paikkaamaan niiden heikkouksia. Hybridimalli lisää myös tietoturvamallin muiden mallien kaipaamaa joustavuutta.

4.6 Kerroksellinen suojautuminen

Kerroksellinen suojautuminen ei ole tietoturvan saralla uusi käsite, mutta siitä on vasta viime vuosina tullut olennainen osa organisaation kokonaisvaltaista tietoturvan toteuttamista. Tietoverkkojen suojauksessa luotettiin pitkään yhteen suojaukselliseen tietoturvatasoon. Tämä suojauksellinen taso oli yleensä perinteinen palomuuri. Tietojärjestelmien kriittisyys on osaltaan johtanut tilanteeseen, jossa kaikkea ei voida laskea enää yhden kortin varaan. Tämä on ilmennyt sekä korkean käytettävyyden ja erillisten tietoturvatasojen implementoinneilla.

Kerroksellisen suojautumisarkkitehtuurin toteutuksen pääperiaatteena on luoda organisaation sisälle suojausmenetelmä, joka koostuu useista erillisistä suojaustasoista. Näiden suojaustasojen tarkoituksena on muun muassa suojella organisaation dataa sekä organisaation tarjoamia ulkoisia ja sisäisiä palveluja. Hyvä esimerkki kerroksellisesta suojautumisesta on niin kutsuttu Sveitsin juusto -malli (engl. *swiss cheese model*), joka mainitaan usein riskianalyysien ja riskienhallinnan yhteyksissä. Kyseisessä mallissa usea reikäinen juustoviipale on sijoitettuna jonoon peräkkäin pystysuunnassa. Jotta juustoviipaleista rakennettu jono on mahdollista läpäistä, pitää reiän osua samalle kohdalle jokaisen viipaleen kohdalla. Kerroksellisella mallilla eliminoidaan yhden tason heikkous. Tällä tarkoitetaan sitä, että vaikka yksi juustoviipaleista olisi muita reikäisempi, ei se automaattisesti tarkoita

koko juustojonon läpäisyä. Samaa periaatetta on mahdollista hyödyntää myös tietoturvassa. Jokainen erillinen suojaustaso lisää omalta osaltaan tietoturvaa. Tietoverkoissa kerroksellisuuteen voidaan laskea kuuluvaksi muun muassa verkkoliikenteen välissä fyysisesti tai loogisesti toimivat erilaiset palomuurit, välityspalvelimet, IDS- ja IPS-järjestelmät, erilaiset haittaohjelmantorjuntajärjestelmät, DLP-järjestelmät, SIEM-järjestelmät, SSL-liikenteen analysoijat sekä sovelluksien omat sisäiset suojausmekanismit. Kuviossa 16 on kuvattu näitä tyypillisiä kerroksellisen suojautumisarkkitehtuuriin kuuluvia osatekijöitä.



Kuvio 16. Kerroksellisen suojautumisen komponentteja

Kerroksellisen suojauksen toteutustapana voidaan käyttää perinteistä, kuvion 16 mukaista arkkitehtuuria, jossa liikenne kulkee järjestyksessä erikseen jokaisen tietoturvatason läpi. Tietyn tyyppiselle liikenteelle voidaan myös määritellä tarkemmin, minkä suojaustasojen läpi kyseinen liikenne välitetään. Perinteisen mallin lisäksi voidaan käyttää erillistä orkestraattoria, joka ohjailee luodun säännöstön perusteella tietyt ehdot täyttävän liikenteen tietyille tietoturvatasoille. Kerroksellisen suojauksen tuomat edut ja hyödyt ovat sen tarjoamassa suojauksen monipuolisuudessa ja tehokkuudessa. Kerroksellista suojautumista koskeva kritiikki liittyy erityisesti kyseisen järjestelmän toteuttamisen kustannuksiin, ylläpitoon, kasvavaan viiveeseen sekä sen rakenteen monimutkaisuuteen.

5 Sovellustason palomuurit

5.1 Yleistä

Yhtenä ratkaisuna web-sovellusten suojaamiseen ovat web-sovellustasolla toimivat palomuurit. Ne toimivat nimensä mukaisesti sovellustasolla eli OSI-mallin ylimmillä kerroksilla. Sovellustasolla toimivat palomuurit voivat olla fyysisiä laitteita, lisäosia tai erillisiä filttereitä, joiden avulla luodaan tietynlainen säännöstö HTTP-protokollan avulla tapahtuvaan liikennöintiin. Sovellustason palomuurin yleisimpiä käyttötarkoituksia ovat web-sovelluksien suojaaminen tyypillisimmiltä sovellustason hyökkäyksiltä, kuten XSS-hyökkäyksiltä ja erilaisilta injektioilta. Lisäksi web-sovellustason palomuurilla pystytään muun muassa valvomaan muodostettujen sessioiden evästeitä ja varmentamaan web-sovellukseen syötettyjen käyttäjäsyötteiden oikeellisuus. Toteutustavasta riippuen niiden avulla on myös mahdollista suojautua vielä toistaiseksi tuntemattomilta sovelluksen haavoittuvuuksilta eli niin sanotuilta nollapäivähaavoittuvuuksilta. Web-sovellustason palomuurien tarkoituksena ei ole korvata perinteisiä palomuurijärjestelmiä. Niiden toiminta rajoittuukin lähinnä ainoastaan HTTP-protokollan avulla tapahtumaan viestintään, jolloin ne eivät ymmärrä eivätkä käsittele lainkaan alemman tason protokollia.

Web-sovellustason palomuurit ovat olleet markkinoilla jo pidemmän aikaa, mutta niiden varhaisten versioiden toteutus on jäänyt ainakin vielä osittain puutteelliselle tasolle. Vasta viime vuosina kyseiset palomuurituotteet ovat kehittyneet huomattavasti parempaan suuntaan ja ovatkn tätä kautta saaneet hiljalleen jalansijaa myös organisaatioiden tietoverkoista. Niiden kasvun odotetaankin lisääntyvän tulevina vuosina entisestään. Kasvua tulevat edesauttamaan web-palvelujen entistä laajempi ja monipuolinen käyttö, tiukentuva lainsäädäntö sekä jatkuvasti kasvava tarve suojata liiketoimintakriittisiä web-sovelluksia. Gartnerin vuonna 2014 tekemän arvion mukaan vuoteen 2020 mennessä yli 50 prosenttia julkiseen internetiin kohdistetuista web-palveluista on suojattu jonkin tyyppisellä web-sovellustason palomuurilla (D’Hoinne & Hils & Young & Feiman 2014).

Web-sovellustasolla toimivat palomuurit tunnistetaan nykyään myös yleisissä auditointikriteeristöissä niin kansallisella kuin myös globaalilla tasolla. Kansallisen turvallisuusauditointikriteeristön (KATAKARI) vuonna 2015 julkaiseman ohjeistuksen 13 kohdassa sovellusten suojauksen osalta mainitaan seuraava:

KATAKRI I 13 ”Monitasoinen suojaaminen koko elinkaaren ajan – Ohjelmistoilla toteutettavat pääsynhallintatoteutukset”. ”Sovellusten suodatustoiminnallisuutta voidaan tukea ja/tai toteuttaa myös esimerkiksi sovelluspalomuureilla (WAF, web application firewall)”. (KATAKRI 2015, 51.)

Valtiohallinnon tieto- ja kyberturvallisuuden johtoryhmän (VAHTI) vuonna 2013 julkaisemassa sovelluskehityksen tietoturvaohjeessa sanotaan ylläpitokohdan korkeasta tasosta seuraavaa:

Verkkopohjainen turvamekanismi (YLP-020): Tuotantoympäristössä tulee olla käytössä mekanismi, jolla voidaan reagoida nopeasti yllättäviin tietoturvatilanteisiin ja estää siten hyökkäys. Tällainen mekanismi voidaan rakentaa esimerkiksi sovelluspalomuurilla. (VAHTI 1/2013, 97.)

Kyseisellä VAHTI-listauksen kohdalla viitataan selvästi virtuaaliseen paikkaukseen

Web-sovellustason palomuureista on olemassa useita erilaisia toteutuksia useilta eri laitevalmistajilta. Suosittuja kaupallisia sovellustason palomuureja ovat F5 BIG-IP ASM, Citrix NetScaler AppFirewall ja Fortinet FortiWeb. Vastaavasti suosittuja avoimeen lähdekoodiin perustuvia ratkaisuja ovat ModSecurity, AQTRONIX WebKnight ja ESAPI WAF.

5.2 Toimintaperiaate

Perinteiset OSI-mallin kolmannella ja neljännellä kerroksella toimivat palomuurit tarkkailevat liikennettä perustuen verkkokerroksen informaatioon, kuten lähde- ja kohdeosoitteisiin sekä kuljetuskerroksen porttinumeroihin. Tällöin ne eivät tee paketeille lainkaan niin sanottua syvää tarkistusta (engl. *deep inspection*). Toisin sanoen perinteiset palomuurit eivät kykene tutkimaan niiden läpi kulkevaa liikennettä OSI-mallin ylimmillä kerroksilla. Tämä tunnistettu heikkous oli pitkään iso

ongelmakohta ja sai hyökkääjät kohdistamaan toimintansa juuri OSI-mallin ylimmille kerroksille.

Viime vuosina suuressa nosteessa ovat olleet niin sanotut seuraavan sukupolven palomuurituotteet (engl. *next-generation firewall*). Niiden ominaisuuksiin kuuluvat perinteisten palomuurien pakettisuodatuksen ohella muun muassa syvällinen pakettien tutkiminen sekä sisäinen IPS-järjestelmä, applikaation tunnistus, virus-suojaus ja URL-filtteröinti. Lisäksi yksi niiden yleensä tarjoamista ominaisuuksista on suojauksen ylettäminen jollakin asteella OSI-mallin seitsemännelle kerrokselle asti.

Web-sovellustason palomuurit eli tyypillisemmin WAF-tuotteet (engl. *web application firewall)* ovat olleet markkinoilla jo pidemmän aikaa, mutta vasta viime vuosina niiden tärkeys on alettu tunnistamaan myös eri yrityksissä ja organisaatiossa. WAF-tuotteiden toimintaperiaatteena on tutkia HTTP- ja HTTPS-verkkoliikennettä perinteisiä palomuureja syvemmälle ja yrittää löytää siitä haitalliseksi luokiteltuja tunnisteita ja poikkeamia erillisestä säännöstöstä. Tarkemmin ottaen WAF tarkkailee sen läpi kulkevan HTTP-liikenteen kehyksien otsakkeita sekä HTTP:n hyötykuormaa. WAF-palomuureja ei ole suunniteltu korvaamaan perinteisiä tai seuraavan sukupolven palomuureja. Niiden tehtävänä on ainoastaan tarjota suojaa web-sovelluksille sovellustasolla tapahtuvia uhkia vastaan.

Nykypäivän WAF-tuotteet voidaan luokitella niiden ominaisuuksien perusteella kahteen erilliseen luokkaan: perinteisiin WAF-tuoteisiin ja kehittyneisiin WAF-tuotteisiin. Taulukossa 5 on lueteltuna näiden kahden WAF-kategorian tyypillisiä ominaisuuksia.

Taulukko 5. WAF-tuotteiden ominaisuudet niiden ominaisuuksien perusteella

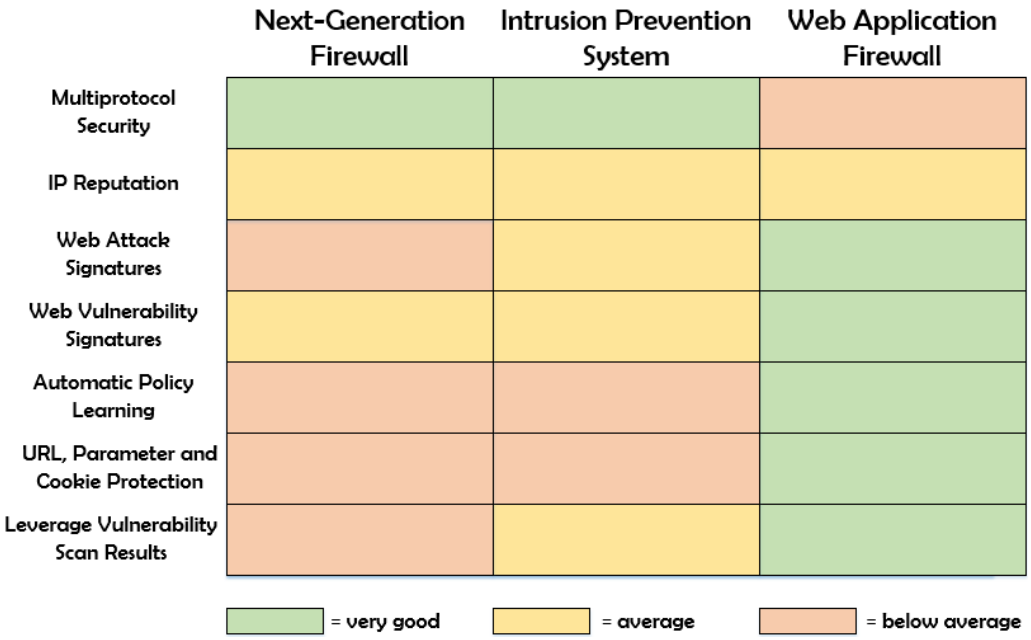
Features	Traditional WAF	Advanced WAF
Signature database	YES	YES
Site Learning	YES	YES
Data leakage prevention	YES	YES
Understanding web site's structures	YES	YES
Cookie protection	YES	YES
User tracking	NO	YES
Brute Force mitigation	NO	YES
Bot detection	NO	YES
Heavy URL protection	NO	YES
L7 DDoS protection	NO	YES
SIEM integration	NO	YES
Client fingerprints	NO	YES

Perinteisten WAF-tuotteiden ominaisuuksiien kuuluvat muun muassa tunnistepohjainen tietokanta, liikenteen oppiminen, tietovuotojen estäminen ja web-sovelluksen käyttämien rakenteellisten elementtien ymmärtäminen. Kehittyneemmät WAF-tuotteen pystyvät kaikkien näiden perinteisten WAF-järjestelmien toimintojen lisäksi muun muassa havainnoimaan ja estämään haitallisia botteja, estämään *brute force* –kirjautumisia, suojaamaan web-sovellusta sovellustasolla tapahtuvilta palvelunestohyökkäyksiltä ja muodostamaan asiakaslaitteen yksityiskohtaisista attribuuteista eräänlaisen sormenjäljen. (McHenry 2016, 34.)

WAF-tuotteet sekoitetaan usein tunkeilijan havaitsemis- ja estojärjestelmiin. Osittain niiden toiminta onkin samankaltaista, mutta eroavaisuudet ovat yhtäläisyyksiä suuremmat. IDS- ja IPS-järjestelmät tarkkailevat kaikkea niiden läpi kulkevaa tai niihin erikseen kohdistettua liikennettä ja yrittävät löytää siitä haitallisia tunnisteita ja poikkeamia. Usein niissä käytetään järjestelmän sisäistä tunnistetietokantaa, johon on etukäteen määriteltynä kaikki haitalliset tunnisteet. IDS- ja IPS-järjestelmien heikkoudet tulevat esille siinä, ettei niitä ole suunniteltu täysin HTTP-protokollan avulla tapahtuvan liikenteen suojaamiseen. Kyseiset suojausjärjestelmät eivät pysty ymmärtämään täysin web-liikenteen loogista käyttäytymistä. IDS- ja IPS-järjestelmien yhteydessä pystyy myös harvoin käyttämään liikenteen oppimista. WAF-tuotteet hyödyntävät usein ainakin osittain toiminnassaan hyväksi havaitun liikenteen oppimista. WAF-järjestelmät on optimoitu täysin HTTP- ja HTTPS-liikenteelle. Usein ne pystyvät tarkkailemaan myös XML-dokumentteja sekä SOAP-liikennettä. WAF:it

ymmärtävät lisäksi HTTP-liikenteen loogisen toimintamallin ja osaavat näin ollen ennakoida liikenteen käyttäytymistä. (McMillan 2009.)

IDS- ja IPS-järjestelmiä ei kuitenkaan voida suoranaisesti korvata WAF-tuotteilla. Siinä missä WAF-tuotteiden tarjoama suojaus on rajattuna yleensä HTTP- ja HTTPS-protokolliin, pystyvät IDS- ja IPS-järjestelmät tarkkailemaan myös muita verkon protokollia. Samaa voidaan todeta myös seuraavan sukupolven palomuureista. Vaikka seuraavan sukupolven palomuurit ovatkin optimoitu pääsääntöisesti lähtevälle liikenteelle, on niillä keinoja myös sovellustason uhkien torjunnassa. Yhdessä toimiessaan kaikki edellä mainitut järjestelmät tukevat hyvin toisiaan ja paikkaavat vastapuolten tunnistettuja heikkouksia. Yhteispelillä saadaan luotua vankkarakenteinen web-sovelluksen suojaamiseen käytettävä kerroksellinen suojausarkkitehtuuri. Kuviossa 17 on havainnollistettu Brocaden suorittamaa yleispätevää vertailua WAF-, IPS- ja NGFW-tuotteiden välillä.



Kuvio 17. Erilaisten suojausjärjestelmien vertailu (Sims 2016, 17, muokattu)

Vertailun perusteella voidaan todeta seuraavan sukupolven palomuurien ja IPS-järjestelmien heikkoudet sovellustason uhkien torjumisessa. Kriittisesti arvioituna vertailu on kuitenkin hyvin pintapuolinen eikä siinä keskitytä lainkaan eri laitevalmistajien tarjoamiin erikoisominaisuuksiin.

5.3 Toteutustavat ja käyttöönotto

WAF-järjestelmien käyttöönottomahdollisuuksia on useita, ja jokaisella niistä on omat vahvuutensa ja heikkoutensa. Toteutustavan tulisikin aina olla tapauskohtainen, jossa huomioidaan tarkasti järjestelmälle etukäteen asetetut vaatimukset. Mahdollinen toteutustapa on myös osaltaan riippuvainen WAF-tuotteen tyypistä.

WAF-järjestelmän yksinkertaisin toteutus verkossa on sen toimiminen siirtoyhteyskerroksen siltana (engl. *L2-bridge*). Toteutustavassa WAF-järjestelmä on sijoitettuna päätelaitteiden ja suojattavan web-palvelimien väliin. Päätelaitteet keskustelevat suoraan palvelinten kanssa ja WAF-järjestelmä on niille täysin läpinäkyvä. Toteutustavan hyödyt ovat läpinäkyvyys, korkea suorituskyky ja järjestelmän integroimisen helppous. Toinen mahdollinen toteutustapa on läpinäkyvänä välityspalvelimena toimiminen (engl. *transparent proxy*). Tällöin siirtoyhteyskerroksen sillan tavoin päätelaitteet keskustelevat suoraan palvelinten kanssa ja WAF-järjestelmä on niin ikään päätelaitteille täysin läpinäkyvä. Ero aikaisempaan toteutustapaan on kuitenkin se, että tässä toteutustavassa WAF-järjestelmä reitittää liikennettä verkkotasolla. Toteutustavan hyötynä on L2-sillan hyötyjen lisäksi myös mahdollisuus verkkotason ominaisuuksien hyödyntämiseen. Kolmas toteutustapa on järjestelmän toimiminen käänteisenä välityspalvelimena (engl. *reverse proxy*). Tällöin päätelaitteet keskustelevat palvelimien sijasta WAF-järjestelmän kanssa, joka reitittää pyynnöt edelleen kohti palvelimia. Toteutustavan suurin hyöty on palvelinten piilottaminen loppukäyttäjiltä, jolloin saadaan muodostettua ylimääräinen tietoturvataso. Neljäs toteutustapa on sijoittaa WAF-järjestelmä täysin erilleen normaalista liikennevirrasta ja ainoastaan peilata tarvittava liikenne WAF-järjestelmälle analysoitavaksi. Toteutustavan hyötyinä ovat käyttöönoton yksinkertaisuus ja viiveiden pysyminen samana. Viidennessä toteutustavassa WAF-järjestelmä integroidaan erillisenä lisäosana suoraan suojattavaan palvelimeen. Toteutustavassa ongelmaksi nousevat yleensä kuitenkin palvelimen resurssit, skaalautuvuus, keskitetty hallinta ja joidenkin tärkeiden ominaisuuksien puuttuminen. (Anders N.d, 31-35.)

WAF-tuotteen käyttöönottoympäristö on suunniteltava tarkkaan ennen käyttöönottoa. Pahimmassa tapauksessa huonosti konfiguroitu ja käyttöönotettu järjestelmä aiheuttaa vääriä positiivisia hälytyksiä, joilla voi olla suora vaikutus suojattavan web-sovelluksen toimivuudelle. WAF:in käyttöönotto tulisikin integroida web-sovelluksen normaaliin kehityselinkaareen, jolloin sitä voidaan testata sovelluksen normaalissa testausvaiheessa. Lähtökohtina erilaisilla WAF:in implementoinneilla tulisi aina olla se, minkälaista suojausta web-sovellus tarvitsee: riittääkö sovellukselle perustason suojaus yleisimpiä sovellustason hyökkäyksiä vastaan vai tarvitaanko sovellukselle kenties erittäin tarkkaa ja monipuolista suojausta?

WAF:in tehokas hyödyntäminen vaatii sen ylläpitäjältä tietoa suojattavasta web-sovelluksesta ja sen käyttämistä rakenteista. Ylläpitäjän on osattava tulkita WAF-järjestelmän tuottamien lokitietojen pohjalta mahdolliset väärät positiiviset ja oikeat positiiviset hälytykset. Joskus vastaan voi myös tulla tilanteita, joissa vaaditaan läheistä yhteistyötä sovelluksen varsinaisten kehittäjien kanssa.

5.4 Järjestelmien suojauskyvyn arviointi

Web-sovellusten suojaamisen toteutuksen arvioimiseen on käytössä useita erilaisia arviointikriteeristöjä. Yleinen hyvä mittaristo on OWASP-säätiön vuonna 2015 käynnistämä Benchmark-projekti. OWASP Benchmark on ilmainen, avoimeen lähdekoodiin perustuva testausjärjestelmä, jonka avulla voidaan arvioida web-sovelluksen suojausjärjestelmän suorituskykyä, kattavuutta ja tarkkuutta ohjelmistossa olevien haavoittuvuuksien tunnistamiseksi. Projekti keskittyy lähinnä web-sovellusten testauksessa käytettävien haavoittuvuusskannerien tulosten analysointiin, mutta sitä voidaan hyödyntää myös web-sovelluksen suojauksen arvioimisessa. Ilman kattavaa järjestelmien arviointia niiden suojauskyvyn vertailu on haastavaa. OWASP Benchmarkia voidaan käyttää yhdessä SAST- (Static Application Security Testing) ja DAST (Dynamic Application Security Testing) -järjestelmien kanssa. (OWASP Benchmark Project 2016.)

OWASP Benchmarkissa suojaustehon arviointiin käytetään neljää erillistä mittaria. Taulukossa 6 on havainnollistettu yleisesti web-sovellusten suojauksen arviointiin

käytettävää matriisia, joka sisältää neljä OWASP Benchmarkin käyttämää arviointimäärettä.

Taulukko 6. Suojauskyvykkyyden arvioinnin matriisi

	TRUE	FALSE
POSITIVE	True Positive (TP)	False Positive (FP)
NEGATIVE	True Negative (TN)	False Negative (FN)

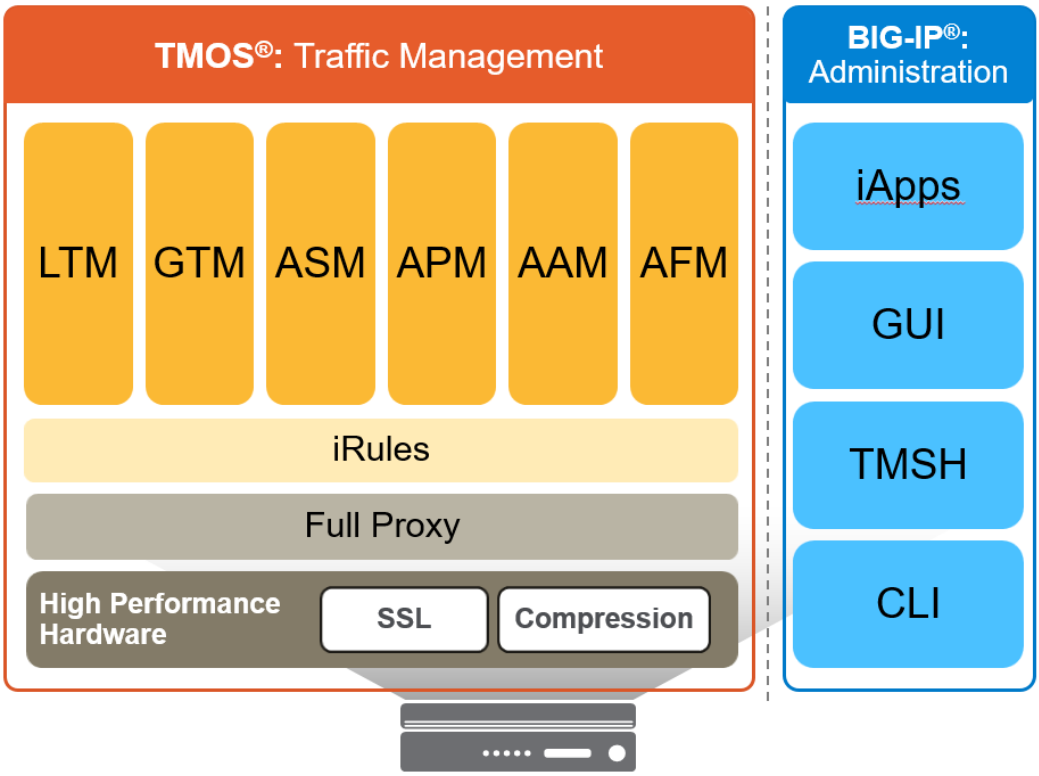
Todellinen positiivinen (engl. *true positive*) kuvaa tilannetta, jossa suojausjärjestelmä havaitsee web-sovelluksessa oikean haavoittuvuuden. Väärä negatiivinen (engl. *false negative*) kuvaa tilannetta, jossa järjestelmä ei löydä sovelluksesta tunnettua haavoittuvuutta. Näiden kahden mittarin ideaalitilanteena voidaan pitää luonnollisesti sitä, että web-sovelluksesta löydetään kaikki haavoittuvuudet eli todelliset positiiviset, ja vastaavasti yhtään tunnettua haavoittuvuutta ei jää huomaamatta. Todellinen negatiivinen (engl. *true negative*) -tilanteessa järjestelmä ei havaitse sovelluksessa haavoittuvuutta, joka kuvaa myös todellista tilannetta. Väärä positiivinen (engl. *false positive*) -tilanteessa järjestelmä löytää sovelluksesta haavoittuvuuden, joka ei kuitenkaan todellisuudessa ole haavoittuvuus. Väärillä positiivisilla voidaan pahimmassa tapauksessa haitata palvelun saatavuutta ja sitä kautta web-sovelluksen toimivuutta. (Mts.)

6 F5 BIG-IP ASM

6.1 Yleistä

F5 Networksin BIG-IP -tuote on ADC-järjestelmä (Application Delivery Controller), jonka tarkoituksena on varmistaa palveluiden korkea käytettävyys, kuormanjako, skaalautuvuus ja parantaa sen yhteydessä toimivien palveluiden kokonaisvaltaista tietoturvaa. F5 Networks on ollut ADC-järjestelmien markkinajohtaja jo yli kymmenen vuotta. Se on myös alan suurin suunnannäyttäjä. (F5 Networks Inc 2016.)

BIG-IP-tuote perustuu nykyään erilliseen TMOS-arkkitehtuuriin (Traffic Management Operation System). Kuviossa 18 on havainnollistettu F5:n TMOS-arkkitehtuuria ja sen rakennetta.



Kuvio 18. TMOS-arkkitehtuuri (F5 Solutions and Technology 2013)

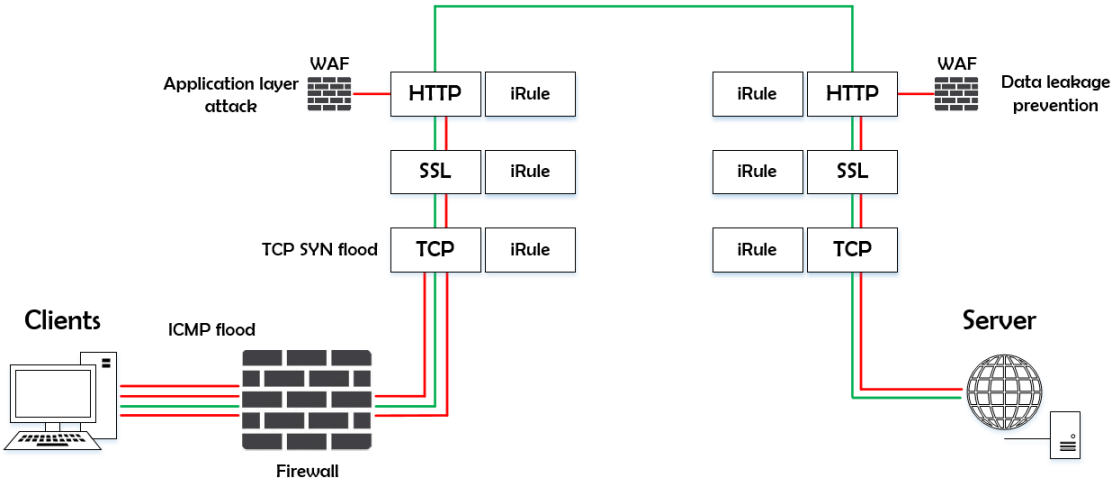
Järjestelmän fyysisen raudan päällä toimii puhdas välityspalvelin -arkkitehtuuri (engl. *full proxy*), jossa pystytään hyödyntämään kaikkia sen toiminnan kannalta parhaita ominaisuuksia. Välityspalvelinarkkitehtuurin päällä toimivat BIG-IP:n moduulit. Moduulien normaaliin toimintaan voidaan vaikuttaa erillisillä iRule-säännöillä. iRule-

säännöillä on siis mahdollisuus yliajaa moduulien yleinen toimintaperiaate ja niiden käyttämät omat yleissäännöt. BIG-IP järjestelmän hallintaan voidaan käyttää muun muassa graafista web-pohjaista käyttöliittymää, perinteistä komentoriviä tai erillisiä iAppseja.

BIG-IP-järjestelmä koostuu useista erillisistä moduuleista, joita voidaan käyttää joko itsenäisesti tai yhdessä. ASM (Application Security Manager) on yksi BIG-IP:n moduuleista ja se on OSI-mallin seitsemännellä kerroksella eli sovelluskerroksella toimiva monipuolinen kehittynyt web-sovelluksien suojaamiseen käytettävä palomuuri. Moduuli kuuluu BIG-IP:n tietoturvamoduuleihin. Muita BIG-IP:n tietoturvamoduuleita ovat AFM (Advanced Firewall Module) ja APM (Access Policy Manager).

ASM:n kolme toiminnallista pääaspektia ovat haitallisten tunnisteiden paikantaminen, normaalista toiminnasta poikkeavien toimintojen havaitseminen liikennevirrasta sekä sovellustason palvelunestohyökkäyksiltä suojautuminen. ASM:n avulla web-sovellusta pystytään suojaamaan muun muassa OWASP Top 10:n uhkilta, uusilta nollapäivähaavoittuvuuksilta sekä sovelluksen omilta haavoittuvuuksilta. ASM on myös täysin yhteensopiva tunnetun maksukorttiturvallisuusstandardin PCI DSS:n kanssa. Yksi ASM:n ehdottomista vahvuuksista verrattuna muihin vastaaviin palomuurituotetoteutuksiin on sen kyky torjua uusia nollapäivähaavoittuvuuksia sen jo olemassa olevilla versioilla. Torjunta mahdollistetaan käyttämällä politiikkojen rakentamisessa hyödyksi positiivista tietoturvamallia. Tämä tapahtuu oppimalla suojattavan sovelluksen käyttämiä rakenteita ja estämällä tämän kautta kaikki siitä poikkeava toiminta. (Wagnon 2016a.)

ASM on suunniteltu toimimaan asiakkaan ja kohdepalvelimen välillä OSI-mallin ylimmällä kerroksella. Kuviossa 19 on havainnollistettu ASM-moduulin sijoittumista kerrokselliseen liikennemalliin.



Kuvio 19. F5 BIG-IP osana kerroksellista suojausta (Agility 2016, muokattu)

ASM toimii osana tätä toimintamallia liikenteen molempiin suuntiin. Se pystyy havaitsemaan ja torjumaan HTTP-pyynnöistä haitallisia tunnisteita, johon perinteinen palomuuri ei kykene. Vastaavasti palvelimen lähettämien HTTP-vastausten osalta ASM pystyy estämään tyypillisiä tietovuotoja. Kaikkien BIG-IP:n kerroksellisten suojaustasojen oletustoimintaan pystytään vaikuttamaan erillisten iRule-sääntöjen avulla.

6.2 Toimintaperiaate

BIG-IP:n ASM-moduulin yhteydessä yksi keskeisimmistä käsitteistä on tietoturvapolitiikat (engl. *security policies*). ASM:n avulla suojattavalle web-sovellukselle luodaan yksilöllisiä tietoturvapolitiikoita, jotka sidotaan sovelluksen käyttämään F5-virtuaalipalveluun. Tietoturvapolitiikalla tarkoitetaan asetusten ja määritysten kokonaisuutta, jonka avulla määritellään, minkälainen liikenne sallitaan ja kielletään kohti suojattavaa web-sovellusta ja sieltä ulospäin. Tietoturvapolitiikat voidaan rakentaa pääsääntöisesti kolmella eri tavalla. Ne voidaan rakentaa automaattisesti, manuaalisesti tai käyttäen kolmannen osapuolen haavoittuvuusskannausta. Lisäksi on mahdollisuus rakentaa täysin erillinen tietoturvapolitiikka XML-pohjaisille sovelluksille.

ASM hyödyntää toimintatavassaan hybridi-tietoturvamallia. Positiivista tietoturvamallia ASM käyttää oppiessaan hyväksi havaittuja liikennevirtoja, jotka

kulkevat järjestelmän läpi. Negatiivisena tietoturvana toimii ASM:n kattava haitallisia tunnisteita sisältävä hyökkäystunnistetietokanta. Yhdistelemällä positiivista ja negatiivista tietoturvamallia saavutetaan tehokas ja laaja-alainen tietoturvapolitiikka. Laillisen liikenteen oppiminen toimii päätekijänä tietoturvapolitiikan rakentamisessa ja sen suojausta tukemassa on erillinen haitallisista tunnisteista koostuva tietokanta.

ASM:n hyökkäystunnistetietokanta sisältää tällä hetkellä yli 2300 erilaista haitallista tunnistetta. Hyökkäystunnistetietokannan avulla järjestelmä pystyy tunnistamaan ja estämään web-sovellukseen kohdistuvia hyökkäysvektoreita. Valmiiden tunnisteiden lisäksi hyökkäystunnisteita on mahdollista ladata lisää manuaalisten tai automaattisten päivityspakettien avulla. F5 Networks julkaisee tiheällä aikavälillä päivityspaketteja, joiden avulla hyökkäystunnistetietokantaa on mahdollista täydentää. Omia tunnisteita voidaan myös vaihtoehtoisesti rakentaa itse. Mikäli ASM havaitsee liikennevirrasta haitallisen tunnisteen, koko tunnisteen sisältämä HTTP-pyyntö estetään tai siitä generoidaan hälytys. Kyseinen toimintamalli on riippuvainen tietoturvapolitiikassa käytettävistä yksilöllisistä asetuksista. (Wagnon 2016c.)

ASM:n virtuaalipalvelukohtainen yksiköllinen tietoturvapolitiikka voidaan asettaa globaalisti kahteen erilaiseen tilaan. Tiloja ovat estotila (engl. *blocking*) ja tarkkailutila (engl. *transparent*). Tarkkailutilassa oleva politiikka ei koskaan estä mitään siihen kohdistuvaa liikennettä, vaan se generoi ainoastaan hälytyksiä ja oppimisehdotuksia politiikassa käytettävistä asetuksista riippuen. Myöskään estotilassa oleva tietoturvapolitiikka ei automaattisesti estä mitään haitallista liikennettä. Haitallista liikennettä estetään tässä tapauksessa ainoastaan silloin, kun myös tietyt muut ehdot täyttyvät.

Tietoturvapolitiikka rakentuu web-sovelluksen käyttämistä komponenteista, joita kutsutaan elementeiksi. ASM:n yhteydessä on käytössä neljä erillistä loogista elementtiryhmää. Elementtiryhmiä ovat parametrit, URL-resurssit, tiedostotyypit ja evästeet. Kaikki web-sovelluksen rakenteet kategorisoidaan näiden elementtiryhmien alle. Jokainen näistä yksittäisistä elementeistä sisältää omat yksityiskohtaiset asetukset ja attribuutit. Globaalin politiikan tavoin elementtiryhmän alainen yksittäinen elementti voidaan asettaa kahteen erilaiseen tilaan. Tiloja ovat tarkkailutila (engl. *staging*) ja estotila (engl. *enforce*). Tarkkailutilassa olevassa elementissä tapahtuvaa ASM:n haitalliseksi katsomaa toimintaa tai ei estetä, vaikka

politiikka olisi asetettuna globaalisti estotilaan. ASM:n toimintalogiikkana on estää haitallista liikennettä ainoastaan silloin, kun sekä politiikka ja elementti, jossa hälytys tapahtui ovat molemmat estotilassa. Sama toiminta on käytössä myös hyökkäystunnisteilla. Hyökkäystunniste voi olla elementtien tapaan joko tarkkailu- tai estotilassa. Mikäli ASM havaitsee liikenteestä hyökkäystunnisteen, joka on tarkkailutilassa, ei hyökkäystunnisteen sisältänyttä HTTP-pyyntöä estetä vaan siitä generoidaan ainoastaan lokitieto. Taulukossa 7 on havainnollistettu kokonaisuudessaan ASM:n liikenteen käsittelytapoja riippuen politiikassa käytetyistä asetuksista.

Taulukko 7. Liikenteen estyminen politiikan asetusten mukaan

	POLICY ENFORCEMENT MODE	
EXPLICIT ENTITY MODE	TRANSPARENT	BLOCKING
STAGING	ALLOW	ALLOW
ENFORCED	ALLOW	BLOCK

Toimintamallin taustalla on tarkoitus antaa järjestelmän ylläpitäjälle aikaa hienosäätää politiikan elementtejä ja hyökkäystunnisteita siten, että osa elementeistä ja hyökkäystunnisteista on estotilassa ja osa tarkkailutilassa. Tälle toimintatavalle voi syntyä tarvetta esimerkiksi silloin, kun suojattavan web-sovelluksen rakenne muuttuu osittain. Web-sovellukseen voidaan esimerkiksi tuoda ohjelmistokehittäjien toimesta uusia ominaisuuksia, jotka vaativat ASM:n tietoturvapolitiikan oppimisen. Tällöin uudet, vasta opitut elementit asetetaan aluksi tarkkailutilaan, jolloin järjestelmällä ja sen ylläpitäjällä on aikaa tarkkailla liikennettä ja säätää sen perusteella elementtien attribuutteja. Kokonaissuojauksessa ei silti menetetä mitään, sillä politiikka on jatkuvasti estotilassa vanhojen rakenteellisten elementtien osalta.

Tietoturvapolitiikkojen yhteydessä käytetään termiä tarkkailuajanjakso (engl. *enforcement readiness period*), kun puhutaan ajanjaksosta, jonka ajan uudet opitut elementit ja hyökkäystunnisteet ovat tarkkailutilassa ennen kuin järjestelmä ehdottaa niiden asettamista estotilaan. Oletusasetuksena kaikissa politiikkatyypeissä tämä ajanjakso on määritelty seitsemän päivän mittaiseksi. Mikäli uudessa opitussa elementissä tai järjestelmän hyökkäystunnisteessa ei tapahdu tämän ajanjakson

aikana hälytystä eikä elementin attribuuteissa tapahdu muutoksia, katsoo ASM niiden olevan tarpeeksi vakaita estotilaan asettamista varten.

Joissain toteutuksissa yksittäisten elementtien oppiminen politiikkaan voi olla työlästä ja haastavaa. Esimerkiksi kaikkien web-sovelluksen käyttämien tarkkojen URL-osoitteiden oppiminen voi monimutkaisessa web-sovelluksessa olla lähes mahdotonta. Tällaisissa tilanteissa elementtiryhmissä voidaan myös hyödyntää villi kortti -elementtiä (engl. *wildcard*). Villi kortti toimii yksittäisen elementin tavoin. Sillä on täysin samat elementtiryhmää vastaavat attribuutit kuin yksittäisillä tarkoilla elementeillä. Villin kortin tarkoituksena on kuitenkin olla yksi elementti, joka täsmää useisiin yksittäisiin tarkkoihin elementteihin. Villin kortin avulla elementtiryhmään lisättäviä uniikkien elementtien määrää voidaan karsia huomattavasti ja korvata ne yhdellä yleispätevällä elementillä. Mikäli elementtiryhmässä on sekä villi kortti että yksittäisiä tarkkoja elementtejä, katsotaan osumaa aina ensin yksittäisistä tarkoista elementeistä. Yksinkertaisimmillaan villi kortti on tähti (*) -elementti, joka täsmää jokaiseen elementtiryhmän alle sisältyvään elementtiin. (Configuration Guide for BIG-IP Application Security Manager 2013 ,4-1.) Taulukossa 8 on listattuna ASM:n yhteydessä käytettävän villin kortin mahdolliset arvot ja niiden selitykset.

Taulukko 8. Villin kortin mahdolliset arvot ja niiden kuvaukset

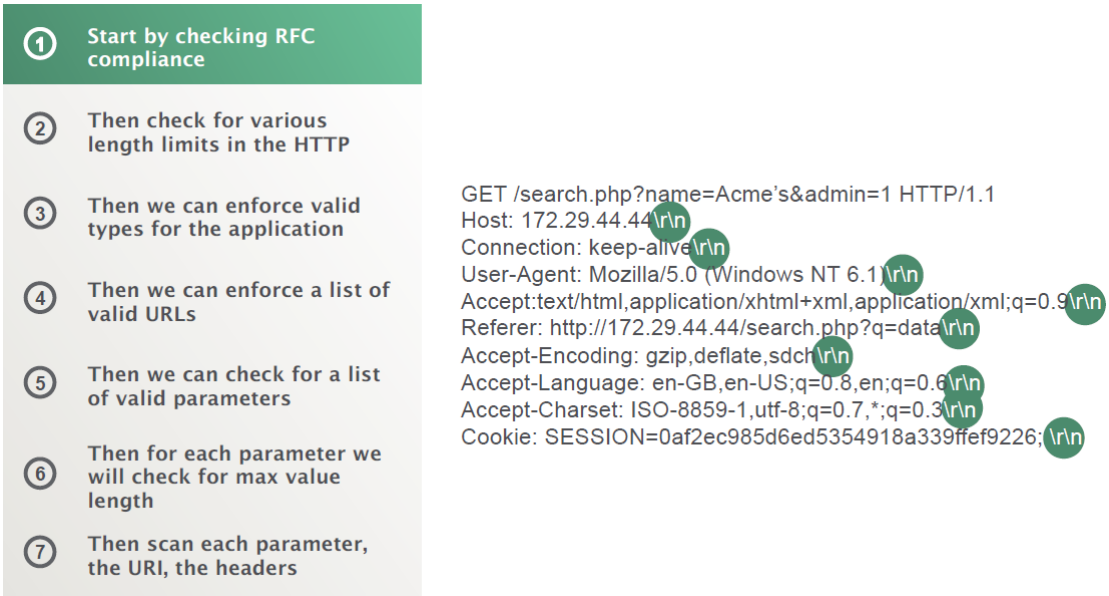
WILDCARD VALUE	DESCRIPTION
*	Match all characters
?	Match any single character
[abcde]	Match any character that is in the specified sequence
[!abcde]	Match any character that is not in the specified sequence
[a-e]	Excactly one character in the range
[!a-e]	Any character not in the range

Villin kortin yhteydessä ei ole mahdollista käyttää säännöllisiä lausekkeita (engl. *regular expressions*) vaan sille on taulukon mukainen oma syntaksi.

Yksi ASM:n kehittyneistä ominaisuuksista on siihen integroitu automaattinen politiikanrakentaja (engl. *real traffic policy builder*). Automaattisen politiikanrakentajan avulla järjestelmä pystyy rakentamaan politiikkaa automaattisesti perustuen politiikassa käytettyihin ylläpitäjän määrittelemiin raja-arvoihin. Kun automaattinen politiikanrakentaja on käytössä, uudet opitut elementit lisätään politiikkaan automaattisesti, kun tietyt ennalta määritellyt raja-arvot

täyttyvät. Automaattisen politiikanrakentajan tärkeimpänä tehtävänä on helpottaa järjestelmän ylläpitoa ja pienentää siihen kuluvaa ylläpitoaikaa ja sitä kautta työmäärää. Rakentajan avulla järjestelmän ylläpitäjältä ei enää vaadita kattavaa tietämystä itse suojattavasta sovelluksesta. Ilman automaattista politiikanrakentajaa järjestelmän ylläpitäjän on manuaalisesti hyväksyttävä tai hylättävä jokainen ASM-järjestelmän generoima oppimisehdotus. ASM:n tietoturvapolitiikka ei koskaan siirry automaattisesti tarkkailutilasta estotilaan eikä vastaavasti myöskään yksittäisiä elementtejä siirretä estotilaan ilman järjestelmän ylläpitäjän hyväksyntää. Tällä toimintamallilla vältetään tilanne, jossa järjestelmä alkaa estämään liikennettä ilman sen ylläpitäjän lupaa.

ASM-moduuliin on pyritty rakentamaan sisäistä kerroksellisuutta ja tätä kautta erilaisia tietoturvatarkistustasoja. Tällöin järjestelmän läpi kulkevat HTTP-viestit kulkevat aina läpi tietyt säännönmukaisuustarkastukset. Kuviossa 20 on havainnollistettu ASM:n yhteydessä oletuksena suoritettavia erilaisia tarkastusprosesseja.



Kuvio 20. ASM-moduulin liikenteelle suorittamat tarkastukset (Dries 2013)

Prosessi aloitetaan tarkastamalla järjestelmän läpikulkevan HTTP-viestin RFC-standardinmukaisuus. Tällä tarkoitetaan HTTP-otsakkeen tarkistuksia, kuten esimerkiksi pyynnön alkamista standardin mukaisella metodilla ja viestin

yleisrakenteen standardinmukaisuutta. RFC-tarkistuksen jälkeen viestille suoritetaan erilaisia pituustarkistuksia ja niitä verrataan politiikassa määritettyihin minimi- ja maksimipituuksiin. Kyseisten tarkastusten jälkeen suoritetaan käytettävästä politiikkatyypistä riippuen erikseen URL-resurssien, tiedostotyyppien ja parametrien tarkistus. Jokaiselle yksittäiselle parametrille, tiedostotyypille tai URL-resurssille voidaan lisäksi suorittaa niiden tarkempiin attribuutteihin, kuten arvoihin ja pituuksiin liittyviä tarkistuksia. Ennen viestin välittämistä kohdepalvelimille suoritetaan vielä skannaus erilaisten haitallisten hyökkäystunnisteiden varalta.

ASM-moduuli suorittaa säännönmukaisuustarkistusten yhteydessä normalisoinnin sellaisille HTTP-kyselylle, jotka se vastaanottaa koodatussa muodossa. Normalisoinnilla tarkoitetaan prosessia, jossa koodatut pyynnöt dekoodataan ASM:n toimesta niiden alkuperäiseen muotoon. Mikäli järjestelmä ei pysty onnistuneesti dekoodamaan sille kohdistettua koodattua pyyntöä, siitä generoidaan tietoturvapolitiikan asetuksista riippuen ”*Evasion technique detected*” hälytys ja pyyntö joko lokitetaan tai se estetään. (K7929 2016.)

Esimerkkinä URL-koodatun hakukentän dekoodaus:

```
http://www.example.com/search?=testi%20testi

http://www.example.com/search?=testi testi
```

Esimerkissä sovelluksen hakukentän arvo ”*testi testi*” on URL-koodattu muotoon: ”*testi%20testi*”. ASM suorittaa arvolle dekoodauksen, jolloin se palautetaan sen alkuperäiseen muotoonsa: ”*testi testi*”. Tämän jälkeen ASM suorittaa kyselylle normaalit tarkistukset. Kyseisen toimintamallin tarkoituksena on estää järjestelmän kiertoyritykset, joissa hyökkäyslausekkeisiin käytettäviä komponentteja on tarkoitus piilottaa erilaisilla koodaustekniikoilla. (Mt.)

6.3 Tietoturvapolitiikat

6.3.1 Automaattinen tietoturvapolitiikka

Automaattinen tietoturvapolitiikka on F5 Networksin suosittelema tapa toteuttaa tietoturvapolitiikkoja. Sen avulla saavutetaan tehokas, ja asetuksista riippuen tarvittaessa myös erittäin helposti ylläpidettävä tietoturvapolitiikka. Toteutustavassa

on mahdollista hyödyntää järjestelmän tarjoamaa automaattista politiikanrakentajaa.

Automaattisen tietoturvapolitiikan luominen aloitetaan nimeämällä uusi politiikka ja sitomalla se F5-virtuaalipalveluun. Politiikka voidaan sitoa jo olemassa olevaan virtuaalipalveluun tai vastaavasti politiikan luomisen yhteydessä voidaan luoda uusi virtuaalipalvelu. Politiikka voidaan myös tarvittaessa luoda asennusvaiheessa niin, ettei sitä sidota mihinkään virtuaalipalveluun. Muista aloitusvaiheen pakollisista asetuksista määritellään web-sovelluksen käyttämä merkistöstandardi. Seuraavassa vaiheessa valitaan politiikan yhteyteen sidottavat hyökkäystunnisteet. Hyökkäystunnisteet ovat loogisesti luokiteltuina erilaisiin kokonaisuuksiin, joka helpottaa niiden käyttöönottoa. Hyökkäystunnisteista on suositeltavaa valita ainoastaan ne, jotka ovat jollain tavalla yhteydessä suojattavaan web-sovellukseen. Ylimääräisten tunnisteiden käyttöönotto ei sinällään ole väärin, mutta niistä saatava hyöty on olematon. Lisäksi turhaan käyttöönotettavat tunnisteet voivat aiheuttaa ylläpidollisia ongelmia, kuten vääriä positiivisia hälytyksiä ja ylimääräistä resurssien käyttöä. Politiikan rakentaminen viimeistellään valitsemalla politiikan tyyppi, joka voi olla perustavanlaatuinen, tehostettu tai kattava. Kyseisillä politiikan tyypeillä määritellään se, miten tarkkaa politiikkaa ollaan rakentamassa. Politiikan rakentamisen yksityiskohtaisuus on kuitenkin suoraan verrannollinen sen ylläpitoon ja rakentamiseen kuluvaan aikaan. Lisäksi tarkoilla politiikoilla on suurin mahdollisuus tuottaa vääriä positiivisia hälytyksiä. (Eames 2013, 15-19.)

6.3.2 Manuaalinen tietoturvapolitiikka

Manuaalinen tietoturvapolitiikka on toinen vaihtoehtoinen tapa toteuttaa ASM:lle tietoturvapolitiikkoja. Manuaalisessa tavassa ylläpitäjälle annetaan enemmän näkyvyyttä ja kontrollia politiikkaan etenkin sen rakennusvaiheessa. Manuaalisessa tavassa ASM-järjestelmän ylläpitäjältä vaaditaan myös ymmärrystä suojattavasta web-sovelluksesta ja sen rakenteellisista komponenteista. Toteutustavassa järjestelmän ylläpitäjän on itse määriteltävä politiikkaan mahdollisesti lisättävät, suojattavassa sovelluksessa käytettävät yksittäiset elementit ja niiden attribuutit. Manuaalista politiikan rakentamista on kuitenkin helpotettu alustamalla sen käyttöön kaksi valmista helppokäyttöistä pohjaa.

Politiikan rakentamiseen voidaan käyttää ASM:n valmiita sovellusvalmiita pohjia (engl. *application ready security template*) tai politiikka voidaan rakentaa käyttämällä nopean käyttöönoton politiikkaa (engl. *rapid deployment security policy*).

Sovellusvalmiit pohjat ovat ASM:lle valmiiksi tuotuja tietoturvapolitiikoita tunnetuille yleisesti käytössä oleville web-sovelluksille. Niiden käyttöönotto on helppoa, sillä järjestelmän ylläpitäjän tarvitsee ainoastaan valita käytettävä sovellus, ja ASM huolehtii kaikesta muusta. Se tietää jo etukäteen sovelluksen tarvitsemat hyökkäystunnisteet ja osaa tätä kautta rakentaa tietoturvapolitiikan täysin itsenäisesti. Tällöin ylläpitäjän tehtäväksi jää käytännössä mahdollisten väärien hälytysten karsiminen ja mahdolliset politiikkaan tehtävät muutokset. Tällä hetkellä ASM:ssä on mukana noin parikymmentä erillistä valmispohjaa, ja lisää tuodaan jatkuvasti. Pohjia on tehty muun muassa Microsoftin Sharepointille ja Exchangelle sekä Oraclen web-toteutuksille. (Wagnon 2016a.)

Nopean käyttöönoton politiikassa tarkoituksena on rakentaa nopeasti käyttöönotettava tietoturvapolitiikka minimaalisella ylläpidollisella työllä. Taulukossa 9 on lueteltuna kaikki nopean käyttöönoton tietoturvapolitiikassa oletuksena suoritettavat tarkastukset.

Taulukko 9. Nopean käyttöönoton politiikan suorittamat tarkastukset

RAPID DEPLOYMENT SECURITY POLICY CHECKS
Performs HTTP compliance checks
Stops information leakages
Prevents illegal HTTP methods from being used in a request
Checks response codes
Enforces cookie RFC compliance
Applies attack signatures to requests and responses

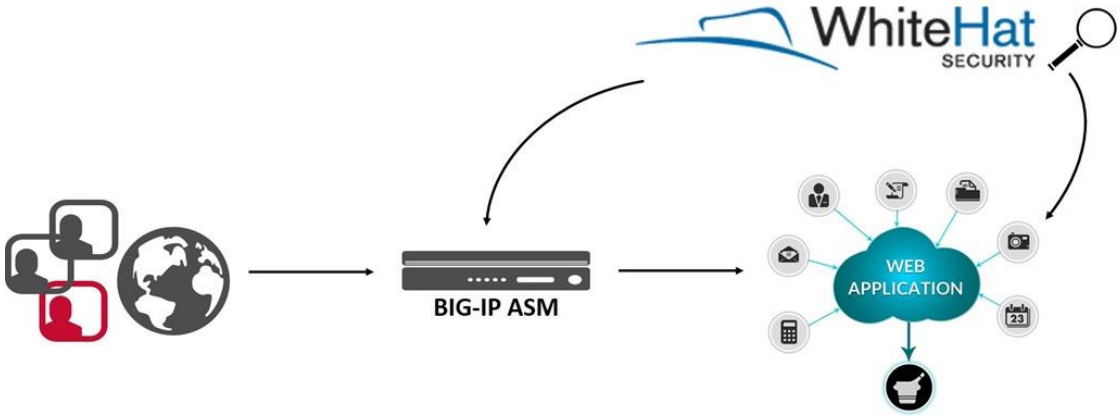
Oletuksena nopean käyttöönoton politiikassa käytetään käytännössä ainoastaan ASM:n hyökkäystunnistetietokantaa negatiivisen tietoturvamallin tavoin sekä suoritetaan HTTP-pyynnöille erilaisia säännönmukaisuustarkastuksia. Lisäksi HTTP-otsakkeen evästeille suoritetaan RFC-säännönmukaisuustarkistukset.

Nopean käyttöönoton politiikassa ei ole tarkoituksena suoranaisesti oppia mitään suojattavasta web-sovelluksesta. Kaikkien elementtiryhmien osalta käytössä ovat

ainoastaan villit kortit, joiden attribuutit ovat määriteltyinä äärettömiksi. Nopean käyttöönoton politiikassa on kuitenkin käytössä useita automaattisen tietoturvapolitiikan käyttämiä ominaisuuksia, kuten tarkkailuajanjakso, jonka tarkoituksena on tässä tapauksessa antaa järjestelmän ylläpitäjälle mahdollisuus karsia hyökkäystunnisteista johtuvat mahdolliset väärät positiiviset hälytykset.

6.3.3 Politiikka käyttäen kolmannen osapuolen haavoittuvuusskannausta

Kolmas tapa rakentaa ASM:n tietoturvapolitiikka on käyttää siihen kolmannen osapuolen haavoittuvuusskannausta. Kyseisessä toteutustavassa hyödynnetään proaktiivista virtuaalipaikkausta. Kuviossa 21 on havainnollistettu politiikan rakentamista kolmannen osapuolen haavoittuvuusskannerin avulla.



Kuvio 21. ASM-politiikka käyttäen haavoittuvuusskanneria (Wagnon 2016b)

Havainnekuvassa suojattava web-sovellus skannataan haavoittuvuuksien varalta käyttäen siihen yhtä ASM:n tukemaa haavoittuvuusskanneria. Tässä esimerkissä skannaukseen käytetään WhiteHatin Sentinel-haavoittuvuusskanneria. Skannerin havaitsemat haavoittuvuudet raportoidaan ja integroidaan ASM-järjestelmään. Tämän jälkeen ASM tekee itsenäisesti raportin pohjalta tarvittavat web-sovellusta suojaavat korjaukset. Tällä hetkellä ASM-järjestelmään on mahdollista integroida WhiteHat Sentinelin lisäksi myös neljä muuta kolmannen osapuolen haavoittuvuusskanneria: IBM Security AppScan Scanner, Cenzic Hailstorm Scanner, QualysGuard WAS ja HP WebInspect. (Eames 2016, 88.)

6.4 Elementtiryhmät ja niiden tasot

6.4.1 Parametrit

Parametrit ovat yksi tietoturvapolitiikan tärkeimmistä yksittäisistä tekijöistä. Parametrit-elementtiryhmän avulla voidaan sallia tai kieltää tiettyjä web-sovelluksen käyttämiä rakenteellisia parametreja. Web-sovelluksen haavoittuvuuden vuoksi tietty lähetetty parametri ja sen arvo saattavat laukaista web-sovelluksessa tietyn toiminteen, johon loppukäyttäjällä ei tulisi olla pääsyä. Esimerkkinä tästä ovat parametrit ”*mode*” ja ”*debug*” sekä niiden arvot ”*admin*” ja ”*on*”.

```
www.example.com/index.html?mode=admin
www.example.com/index.html?debug=on
```

Parametrit-elementtiryhmässä jokaisella yksittäisellä parametrilla on kaikkien muiden ASM:n elementtiryhmien tapaan yksittäiset tarkat attribuutit. Parametrit-elementtiryhmän attribuutteja ovat muun muassa parametrin arvon tyyppi ja sen maksimipituus. Jokaiselle parametrille on myös mahdollista määritellä yksityiskohtaisesti sen yhteydessä sallitut ja kielletyt merkit. Jos siis esimerkiksi web-sovelluksen tietyn parametrin arvossa ei tulisi koskaan esiintyä ”<” -merkkiä, voidaan se kieltää ainoastaan tämän kyseisen parametrin osalta. Parametrien osalta käytössä ovat myös sensitiiviset parametrit, jotka eivät ole nähtävissä elementtiryhmän normaalissa parametritaulukossa. Sensitiiviseksi parametriksi voidaan asettaa esimerkiksi: ”*password*” parametri.

Parametrit-elementtiryhmän erikoisuutena on se, että sen sisältämien elementtien taso voi olla joko globaali taso tai URL-taso. Käytettäessä globaalia tasoa, parametri opitaan tietoturvapolitiikkaan globaalisti. URL-tasoa käytettäessä parametri opitaan sen sijaan ainoastaan määritellyssä tai opitussa URL-resurssissa. Parametrin tason avulla politiikasta saadaan tarvittaessa rakennettua parametrien osalta tiukempi, mikäli käytetään URL-tasoa. Parametrien yhteydessä on mahdollista käyttää myös säännöllisiä lausekkeita. (Configuration Guide For BIG-IP Application Security Manager 2013, 2-18.)

6.4.2 URL-resurssit

URL-elementtiryhmän avulla tietoturvapolitiikkaan voidaan lisätä sallittuja ja kiellettyjä URL-osoitteita. URL-resurssit on mahdollista jakaa kahteen erilliseen alakategoriaan: HTTP URL-resursseihin ja Websocket URL-resursseihin. HTTP URL-resurssit vastaavat normaaleja yleisesti käytössä olevia URL-resursseja. Websocket URLit puolestaan esiltetiin ASM:n osalta BIG-IP:n 12.0 –versiossa ja niiden tehtävänä on suojata URL-elementtiryhmää *websocket*-liikenteeltä.

URL-elementtiryhmän tarkoituksena on rajata sovelluksen käyttäjä pysymään sivuston käyttämien laillisten URL-resurssien piirissä. Ilman URL-resurssien rajaamista sovelluksen käyttäjillä on mahdollisuus päästä käsiksi konfigurointivirheen tai haavoittuvuuden johdosta sivuston arkaluontoisiin URL-osoitteisiin, kuten esimerkiksi joiden sivustojen ylläpitoon käytettävälle *"/admin.asp*" -sivulle. URL-osoitteiden rajaamisen lisäksi niille määriteltäviä tarkkoja attribuutteja ovat muun muassa niiden minimi- ja maksimipituudet. Ilman URL-osoitteiden pituustarkastuksia mahdollinen hyökkääjä voi helposti lähettää web-palvelimelle ylipitkiä URL-osoitteita, mikä voi viedä palvelimelta huomattavan määrän tärkeitä resursseja. (Wagnon 2016d.)

6.4.3 Tiedostotyypit

Tiedostotyypit-elementtiryhmän avulla tietoturvapolitiikassa voidaan sallia tai kieltää tiettyjen tiedostotyyppien esiintyminen HTTP-pyynnöissä ja -vastauksissa. Tämän avulla web-sovelluksen loppukäyttäjä voidaan rajoittaa käyttämään vain sivuston yleisesti sallimia tiedostotyyppejä. Tiedostotyypit-elementtiryhmän avulla voidaan myös estää palvelun loppukäyttäjää pääsemästä mahdollisten haavoittuvuuksien ja sivuston rakenteellisten virheiden kautta web-sovelluksen käyttämän alustan omiin tiedostoihin, asennusvaiheessa käytettyihin tiedostoihin ja muihin kriittisiin tiedostoihin, joihin normaalilla käyttäjällä ei tulisi olla pääsyä. Tiedostotyypit-elementtiryhmän elementeille määriteltäviä tarkkoja attribuutteja ovat muun muassa maksimipituus, POST-kyselyn maksimipituus sekä tiedostotyypin yhteydessä käytettävän kyselyketjun (engl. *query string*) maksimipituus. (Wagnon 2016d.)

6.4.4 Evästeet

Evästeet-elementtiryhmä on neljäs ASM:n yhteydessä käytettävä ryhmä. ASM suorittaa evästeille erilaisia eheystarkistuksia, jotta voidaan varmistua siitä, ettei evästeitä ole muokattu normaalin liikennöinnin aikana. Evästeiden tarkistuksessa järjestelmä generoi aluksi jokaiselle käytössä olevalle tietoturvapolitiikalle satunnaisen turvallisuusavaimen (engl. *security key*). Kyseistä avainta käytetään jatkossa yhdessä erillisen salausalgoritmin kanssa.

ASM käyttää tarkistuksissa kahta erilaista evästetyyppiä. Pääevästettä (engl. *main cookie*) ja kehysevästettä (engl. *frame cookie*). Pääevästeiden perimmäisenä tarkoituksena on varmentaa palvelimen ja asiakkaiden välisten evästeiden eheys. ASM tarkkailee palvelimelta tulevien HTTP-vastauksien *Set-Cookie*-kenttää. Mikäli se havaitsee kyseisen kentän, järjestelmä ottaa kentässä esiintyvästä evästeestä tiivistesumman. Evästeestä otettu tiivistesumma lisätään palvelimen alkuperäiseen evästeeseen ja välitetään *reverse proxy* -periaatteella asiakkaalle. ASM tarkistaa jatkossa asiakkaalta tulevat evästeet ja mikäli ne ovat muuttuneet, generoidaan tapahtumasta *ASM Modified Cookie* -hälytys. Jos taas eväste on pysynyt muuttumattomana, pyyntö välitetään kohdepalvelimelle. ASM ei koskaan generoi hälytystä palvelimen asiakkaalle lähettämän evästeen muuttuessa. Toinen pääevästeiden tärkeä tehtävä on valvoa muodostettujen sessioiden elinikää. Oletuksena pääevästeiden elinaika on 600 sekuntia. Pääevästeen avulla verifioidaan myös ASM:n kehysevästeiden muuttumattomuus eli eheys. (SOL6850 2016.)

ASM:n pääevästeet ovat aina muotoa:

```
TSxxxxxxxx={eväste}
```

Pääeväste alkaa *TS*-merkistöllä, jota seuraa kahdeksan heksadesimaalin rivistö. Heksadesimaalit identifioivat käytössä olevan tietoturvapolitiikan

Kehysevästeet tallentavat viittaavan objektin ja käsittelevät dynaamista tietoa. Ne voidaan jakaa kahteen erilliseen alaluokkaan, joita ovat *flow frame cookie* ja *extraction frame cookie*. Kehysevästeiden tarkoituksena on tallentaa viitattu objekti ja mahdollistaa sen sallittu muuttuminen (SOL6850 2016). Esimerkkinä *flow frame cookiesta* on tilanne, jossa suojattavan web-sovelluksen toimintalogiikkaan kuuluu

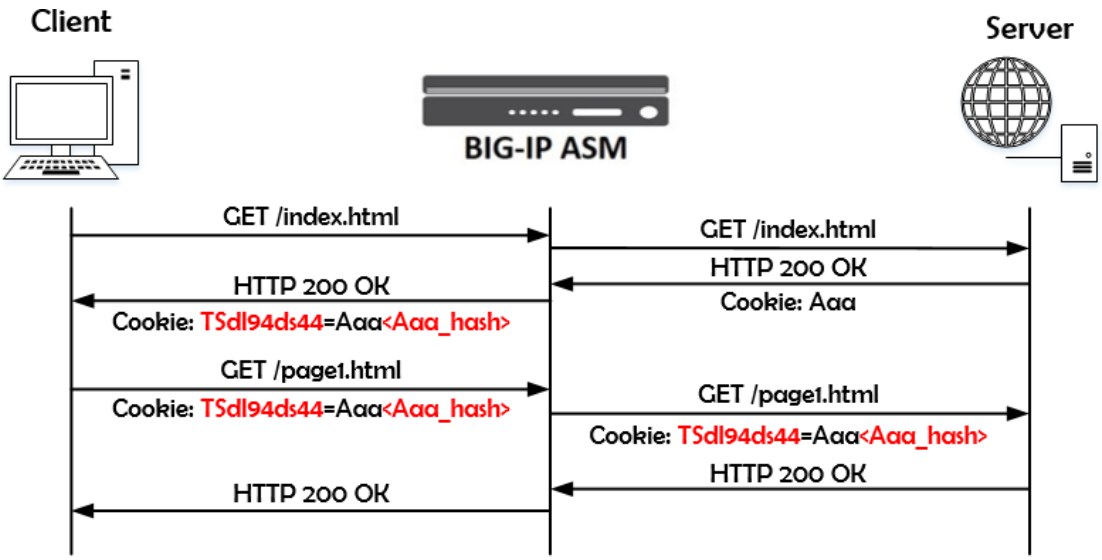
se, että sivuston käyttäjän on vierailtava aluksi sivuston tietyssä URI-resurssissa, ennen siirtymistä seuraavaan. Kyseisessä varmennuksessa hyödynnetään ASM:n *flow frame* -evästettä.

ASM:n käyttämät kehysevästeet ovat aina muotoa:

```
TSxxxxxxxx_d={eväste}
```

Kehyseväste alkaa pääevästeen tavoin *TS*-merkistöllä, jota seuraa kahdeksan heksadesimaalin rivistö. Pääevästeestä poiketen ASM:n evästeeseen lisäämä kenttä päättyy aina ”*_d*”-merkkeihin.

Muista ASM:n elementtiryhmistä poiketen evästeiden yhteydessä voidaan käyttää ainoastaan *Never (wildcard only)* tai *Selective* -oppimistasoa. Oppimistasot on selitetty tarkemmin kappaleessa 6.5.1. Villiä korttia käytettäessä evästeistä tarkistetaan käytännössä ainoastaan niiden muuttumattomuus eli eheys ja niissä mahdollisesti esiintyvät hyökkäystunnisteet. Tarkistukset suoritetaan ainoastaan asiakaslaitteelta web-sovellukselle tuleville HTTP-kyselyille, koska suojattavaa web-sovellusta itsessään voidaan pitää luotettuna. Tällä toimintamallilla eliminoidaan myös turhista tarkistuksista aiheutuva ylimääräinen kuorma ja mahdolliset väärät positiiviset hälytykset. Kuviossa 22 on havainnollistettu evästeiden käsittelyä asiakkaan ja palvelimen välillä, kun käytössä on ASM-moduuli.



Kuvio 22. ASM-moduulin pääevästeiden käsittely

Havainnekuvassa ASM toimii käänteisen välityspalvelimen periaatteella**.** ASM ei automaattisesti poista evästeestä sen lisäämiä kenttiä, vaan ehtojen täytyttyä eväste välitetään palvelimelle sellaisenaan. Tämä saattaa joskus aiheuttaa ongelmia palvelinpuolen evästeiden käsittelyssä. Tällöin F5:lle on mahdollista luoda erillinen iRule-sääntö, jonka yhteydessä käytetään säännöllistä lauseketta. Säännön avulla ehtojen täytyttyä palvelinevästeestä poistetaan automaattisesti ASM:n lisäämät ylimääräiset lisäkentät ennen viestin välittämistä kohdepalvelimille. Säännön haittapuolena on se, että se lisää F5-laitteen kuormitusta ja voi aiheuttaa suurilla liikennemäärillä viivettä asiakkaan ja palvelimen välillä. (K13693 2016.)

6.5 Oppimisprosessit ja politiikan kehittäminen

6.5.1 Automaattisen politiikan oppimistasot, metodit ja raja-arvot

Automaattisen tietoturvapolitiikan luomisen yhteydessä sille määritellään erityinen oppimistaso. Oppimistasoja on kolmea eri tyyppiä ja ne on rakennettu siten, että järjestelmän ylläpitäjä voi valita niistä omaan tilanteeseensa sopivimman vaihtoehdon. Perustavanlaatuinen oppimistasotaso (engl. *fundamental*) tarjoaa yleisen, helposti ylläpidettävän ja moneen tilanteeseen sopivan peruspolitiikan. Tehostettu taso (engl. *enhanced*) tarjoaa perustavanlaatuista tasoa monipuolisemman ja tarkemman politiikan, jonka ylläpito on silti pyritty pitämään mahdollisimman yksinkertaisena. Kattava taso (engl. *comprehensive*) on tasoista monipuolisin ja kattavin sillä sen avulla on mahdollista rakentaa erittäin tarkka ja yksityiskohtainen tietoturvapolitiikka. Kattavan tason tietoturvapolitiikan rakentaminen on kuitenkin hidasta ja sen ylläpitäminen on työlästä. Sillä on myös kaikista oppimistasoista suurin riski tuottaa vääriä positiivisia hälytyksiä. (Configuration Guide for BIG-IP Application Security Manager 2013, 2-2.)

ASM:ssä käytettävällä oppimistasolla on suorat vaikutukset eri elementtiryhmien osalta käytettäviin oppimismetodeihin. ASM:ssä oppimismetodeja on kolmea tyyppiä. Ensimmäinen näistä on *Add All Entities*, jonka tarkoituksena on oppia kaikki web-sovelluksessa käytettävät asetuksen alaisen elementtiryhmän yksittäiset elementit ja niiden tarkat attribuutit. Politiikan rakentamisvaiheessa *Add All Entities* -asetuksella olevassa elementtiryhmässä käytetään villiä korttia. Tällöin ASM oppii

kaikki villiin korttiin osuvat yksittäiset elementit, ja ne joko lisätään automaattisesti politiikkaan tai ASM generoi niistä oppimisehdotuksia. Samalla ASM myös oppii yksittäisten elementtien käyttämiä tarkkoja attribuutteja. Kun ASM on lopulta oppinut kaikki web-sovelluksen käyttämät elementit kyseisen elementtiryhmän osalta, voidaan tämä ryhmä siirtää estotilaan. Tällöin villi kortti poistetaan, jolloin ryhmän sisälle jää ainoastaan kaikki opitut tarkat elementit. Tämän myötä jatkossa kaikki liikenteestä havaitut tuntemattomat elementit sekä tunnetut elementit, joiden attribuutit poikkeavat opituista, aiheuttavat hälytyksen. *Add All Entities* -oppimismetodi takaa parhaimman tietoturvatason, mutta vaatii suuren määrän ylläpitoa ja kehittämistä. Sillä on myös oppimismetodeista suurin mahdollisuus tuottaa vääriä hälytyksiä, tämän vuoksi sitä tulisi käyttää harkitusti tarkkaan valituissa elementtiryhmissä. (Mts, 2-2.)

Never (wildcard only) on toinen ASM:n käyttämistä oppimismetodeista. Se poikkeaa täysin *Add All Entities* -tavasta, sillä sen alaisessa elementtiryhmässä käytetään sen nimen mukaisesti ainoastaan villiä korttia. Yksittäisiä tarkkoja elementtejä ei tällöin lisätä politiikkaan lainkaan. Mikäli käytössä on automaattinen politiikanrakentaja, säätää se villin kortin attribuutteja automaattisesti perustuen nähtyyn liikenteeseen. Villiä korttia ei poisteta elementtiryhmästä missään politiikan elinkaaren vaiheessa. *Never (wildcard only)* -oppimismetodi tarjoaa perustason suojauksen minimaalisella ylläpidollisella työllä. Tämän oppimistavan yhteydessä on myös mahdollista käyttää eräänlaista täydentävää mallia, jossa muutamasta web-sovelluksen kriittisimmästä elementistä luodaan politiikkaan yksittäinen tarkka elementti ja niille säädetään omat tarkat attribuutit. Tällöin politiikkaan saadaan helposti yhdistettyä monipuolisuus ja ylläpidon helppous. (Mts. 2-2.)

Kolmas oppimismetodi on valikoiva oppiminen (engl. *selective*), jonka tarkoituksena on tarjota eräänlainen välimuoto *Add All Entities* ja *Never (wildcard only)* -oppimismetodeille. *Selective*-oppimismetodissa elementtiryhmään sisällytetään villi kortti. Villille kortille määritellään omat attribuutit, jotka sopivat yleisesti suojattavan web-sovelluksen käyttämiin arvoihin. Oppimisvaiheessa elementtiryhmään ei lisätä yksittäisiä tarkkoja elementtejä, mikäli ne osuvat attribuuttiensa puolesta ryhmän villiin korttiin. Jos järjestelmä kuitenkin havaitsee liikennevirrassa elementin, jonka attribuutit poikkeavat villistä kortista, siitä generoidaan ASM:n toimintamallin

perusteella oppimisehdotus. Mikäli oppimisehdotus päätetään hyväksyä, ei tässä tapauksessa löysätä villin kortin attribuutteja vaan politiikkaan lisätään tämä yksittäinen tarkka elementti, jossa hälytys tapahtui. Samalla kyseessä olevalle elementille säädetään (omat) tarkat yksilölliset attribuutit. *Selective*-oppimismetodissa elementtiryhmän villiä korttia ei poisteta missään politiikan elinkaaren vaiheessa. Tavoitetilassa politiikka sisältää villin kortin lisäksi useita yksittäisiä tarkkoja elementtejä. *Selective*-oppimismetodi takaa hyvän tasapainon ylläpidettävyyden ja tietoturvan välillä. (Mts. 2-2.)

Taulukossa 10 on listattuna ASM:n käyttämät kolme oppimistasoa ja niistä seuraavat oppimismetodit eri elementtiryhmiin.

Taulukko 10. Oppimistasojen vaikutukset elementtiryhmien oppimismetodeihin

SECURITY POLICY GROUP	FUNDAMENTAL	ENHANCED	COMPREHENSIVE
File Types	Add All Entities	Add All Entities	Add All Entities
URLs	Never (Wildcard only)	Selective	Add All Entities
Parameters	Selective	Selective	Add All Entities
Cookies	Never (Wildcard only)	Selective	Selective

Käytössä olevalla oppimistasolla on myös suoria vaikutuksia automaattisessa politiikassa oletuksena liikenteelle suoritettaviin tarkastuksiin. Taulukossa 11 on listattuna kaikki ASM:n suorittamat tarkastukset ja niiden käyttöönotto politiikan eri oppimistyypeillä.

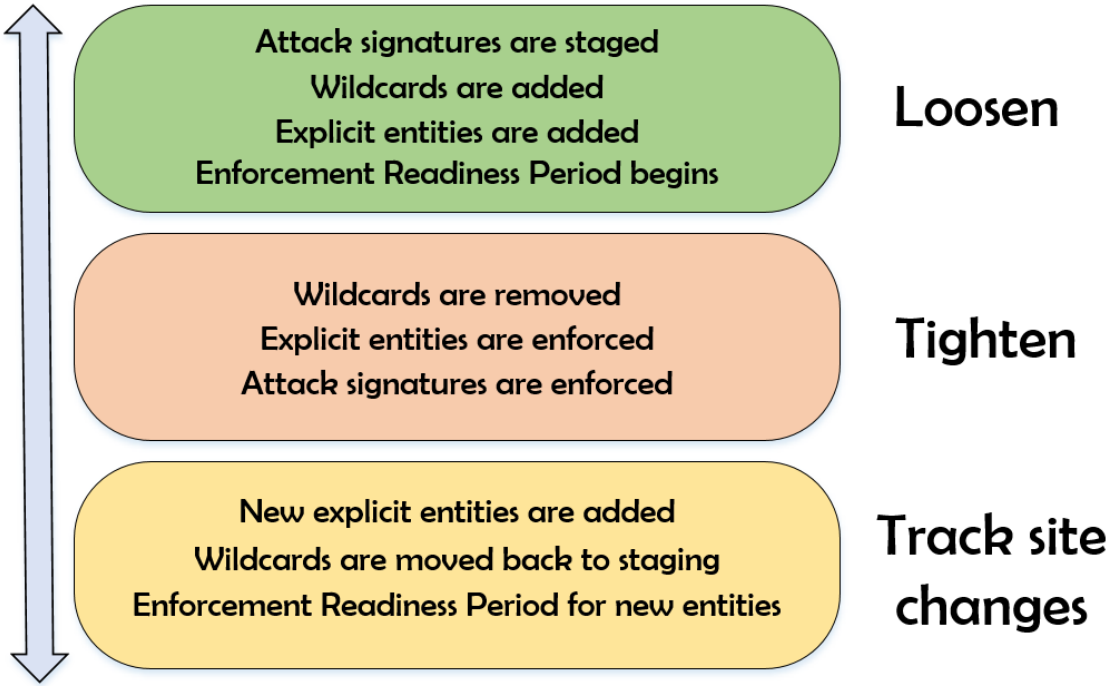
Taulukko 11. Oppimistasojen vaikutukset liikenteelle suoritettaviin tarkastuksiin

SECURITY POLICY CHECK	FUNDAMENTAL	ENHANCED	COMPREHENSIVE
HTTP Protocol Compliance	YES	YES	YES
Evasion Techniques Detected	YES	YES	YES
File Type Lengths	YES	YES	YES
Attack Signatures	YES	YES	YES
URL Meta Characters	NO	YES	YES
Parameter Level	YES (Global level)	YES (Global level)	YES (Parameter level)
Parameter Name Meta Chracters	NO	YES	YES
Parameter Value Lengths	NO	YES	YES
Value Meta Characters	NO	NO	YES
Allowed Methods	NO	YES	YES
Request Length exceedes buffer size	YES	YES	YES
Header Length	YES	YES	YES
Cookie Length	YES	YES	YES
Failed to Convert Character	YES	YES	YES
Content Profiles	NO	YES	YES
Detect advanced protocols	NO	NO (YES, IF XML/JSON)	NO (YES, IF XML/JSON)
Host Names	YES	YES	YES
CSRF URLs	NO	NO	NO

Taulukossa mainitut tarkistustyypit ovat ainoastaan politiikan oppistasojen oletusasetuksia. Kaikkia näitä yksittäisiä asetuksia voidaan poistaa tai ottaa käyttöön erikseen riippumatta käytettävästä politiikan oppimistasosta.

6.5.2 Tietoturvapolitiikan elinkaaren vaiheet

ASM:n tietoturvapolitiikan rakentamisen yhteydessä puhutaan yleensä politiikan elinkaaresta. Politiikan elinkaareen kuuluu kolme erilaista vaihetta. Jokainen näistä vaiheista sisältää omat spesifit sääntönsä ja näiden sääntöjen avulla määritellään se, milloin politiikan elementtiryhmän alainen yksittäinen elementti on valmis siirrettäväksi elinkaaren seuraavalle tasolle. Politiikan elinkaaren vaiheita on havainnollistettu kuviossa 23.



Kuvio 23. Automaattisen tietoturvapolitiikan elinkaaren vaiheet

Ensimmäinen politiikan elinkaaren vaihe on liikenteen oppiminen. Tässä vaiheessa politiikkaa löysätään (engl. *loosen*). Kyseisessä politiikan rakennusvaiheessa elementtiryhmiin lisätään niiden asetuksista riippuen uusia yksittäisiä elementtejä ja lisättyjen elementtien attribuutteja säädetään oikeanlaisiksi. Politiikan löysäämisvaihe voidaan kuvitella eräänlaiseksi hyvän liikenteen lisäämiseksi valkoiselle listalle. Alussa kaikki liikenne on estettyä ja ASM:n oppiman liikenteen

perusteella hyväksi havaittu liikenne web-sovellukselle sallitaan. Liikenteen oppimisnopeus riippuu politiikassa käytettävästä oppimisnopeudesta ja siitä, tuleeko liikenne luotettavaksi luokitelluista lähde-IP-osoitteista. Käytettäessä automaattista politiikanrakentajaa, luotetusta IP-osoitteesta tuleva liikenne opitaan oletuksena välittömästi, kun taas tuntemattomista IP-osoitteista tuleva liikenne vaatii useita eri sessioita ja lähde-IP-osoitteita tietyltä aika-ikkunalta ennen liikenteen oppimista ja lisäämistä ASM:n tietoturvapolitiikkaan. Taulukossa 12 on lueteltuna ja kuvattuna lyhyesti ASM:n automaattisessa politiikassa käytettävät oppimisnopeudet.

Taulukko 12. Liikenteen oppimisnopeudet ja niiden kuvaukset

LEARNING SPEED	DESCRIPTION
HIGH	Policy Builder learns traffic from a small number of requests
MEDIUM	Policy Builder learns traffic from a medium number of requests
LOW	Policy Builder learns traffic from a high number of requests

Automaattisen tietoturvapolitiikan oppimisnopeuksista suositeltavimpia ovat keskitaso ja hidas. Tällöin politiikalla on pienempi riski tuottaa vääriä positiivisia hälytyksiä. Nopeaa oppimisnopeutta on suositeltavaa käyttää ainoastaan järjestelmän testauksen yhteydessä. Oppimisnopeuden valinta on kuitenkin paljon riippuvainen suojattavan web-sovelluksen laajuudesta ja siihen kohdistuvasta liikenteen määrästä.

Toinen vaihe on politiikan kiristäminen ja sen vakauttaminen (engl. *tighten*). Politiikan vakauttamisen yhteydessä elementtiryhmistä poistetaan villejä kortteja sekä yksittäisiä elementtejä ja hyökkäystunnisteita asetetaan estotilaan. Järjestelmä ehdottaa yksittäisten elementtien siirtämistä estotilaan, kun elementin attribuutit ovat pysyneet muuttumattomina eivätkä elementit ole aiheuttaneet hälytyksiä tarkkailuajanjakson aikana. Kolmannessa ja viimeisessä vaiheessa valvotaan sovelluksessa tapahtuvia muutoksia politiikan vakauttamisen jälkeen (engl. *track site changes*). Mikäli politiikan rakentamisen jälkeen jatketaan automaattinen politiikanrakentajan käyttöä ja järjestelmä havaitsee sovelluksessa muutoksia, generoidaan muutoksista lokitieto ja politiikkaa löysätään tilapäisesti. Löysäämisen tarkoituksena on varmistaa, että laillista liikennettä kohti web-sovellusta ei esty. Löysäämisen aikana järjestelmä oppii sovellukseen tulleet muutokset nähdyn

liikenteen perusteella. Kun ASM on havainnut lopulta tarpeeksi liikennettä, se ehdottaa jälleen politiikan kiristämistä ja vakauttamista.

6.5.3 Tietoturvapolitiikan ylläpitäminen

Varsinaisen politiikan luomisen jälkeen alkaa sen kehitys ja ylläpito. Tähän kuluva ylläpidollinen työ on yleensä suoraan riippuvainen siitä, kuinka tarkka politiikka luotiin sen rakennusvaiheessa, ja miten web-sovellus muuttuu sen elinkaarensa aikana. Yksi sovellustason palomuurien suurimmista kompastuskivistä onkin sovelluksen muuttuminen sen elinkaaren aikana. ASM osaa käsitellä myös muuttuvaa web-sovellusta. Mikäli web-sovelluksen tiedetään muuttuvan säännöllisesti, on suositeltavaa jatkaa automaattisen politiikanrakentajan käyttöä ja pitää oppimislippu päällä politiikan estoasetuksissa. Tällöin web-sovelluksen mahdolliset uudet elementit lisätään lopulta politiikkaan niiden ylitettyä ennalta määritellyt kynnysarvot. Uudet opitut elementit ovat aina aluksi tarkkailutilassa. Tällöin ASM saa aikaa uusien elementtien tarkkailuun ja niiden asetuksien säätämiseen. (K11914 2016.)

Mikäli kuitenkin tiedetään etukäteen, ettei suojattava web-sovellus tule muuttumaan lainkaan sen elinkaaren aikana tai muuttuminen on vähäistä ja muutoksiin osataan valmistua etukäteen, on syytä ottaa toinen lähestymistapa. Tällöin automaattista politiikanrakentajaa ei politiikan valmistumisen jälkeen enää tarvita. Se voidaan ottaa käyttöön vain tilanteissa, jolloin web-sovelluksen tiedetään muuttuneen.

6.6 Integroitavat lisätoiminnallisuudet

F5 Networks tarjoaa BIG-IP ASM-tuotteeseen myös useita muita tietoturvalaajennuksia, joiden käyttöönotto on kuitenkin valinnaista. Osaa näistä laajennuksista voidaan myös käyttää täysin erillään, eikä niitä ole siten sidottu mitenkään ASM-järjestelmään.

Cross-Site Request Forgery -suojauksen avulla voidaan suojautua tyypillisimmiltä CSRF-hyökkäyksiltä. Suojauksen toimintaperiaatteena on lisätä jokaisen suojatun URL-resurssin perään erillinen JavaScript-koodi. Esimerkkitapauksessa ”*index.html*” URI:n perään lisätään JavaScript-koodi seuraavasti:

```
http://www.example.com/index.html?CSRT=14540554957304674023
```

CSRF -suojauksen toteuttamisella on oma varjopuolensa. Joissakin toteutuksissa sillä on suuri mahdollisuus estää laillista liikennettä tai rikkoa web-sovelluksen normaali toimintalogiikka. Tämän johdosta CSRF-suojaus tulisi aina testata hyvin tarkkaan erillisessä testiympäristössä ennen sen ottamista varsinaiseen tuotantokäyttöön. CSRF-suojaus on ASM:ssä erillinen komponentti, eikä sitä ole sidottu mitenkään ASM:n varsinaisiin tietoturvapolitiikoihin. CSRF-suojauksen konfigurointi tapahtuu ASM:ssä aina URL-resurssi kohtaisesti. (K11930 2016.)

IP Intelligence on yksi F5:n tarjoamista lisätoiminnallisuuksista, jonka avulla voidaan suojata haluttuja tietojärjestelmiä haitalliselta toiminnalta. IP Intelligencen toimintamallina on kerätä tietoa sen hetkisistä haitallisista IP-osoitteista F5 Networksin omasta Webroot BrightCloud -pilvestä. Haitallisiksi IP-osoitteiksi lasketetaan esimerkiksi erilaisten bottiverkkojen ja epäilyttävien välityspalvelimien käyttämät IP-osoitteet. Pilven sisältämä tieto päivittyy jatkuvasti ja se synkronoidaan F5-laitteen kanssa oletuksena viiden minuutin välein. Vuonna 2014 pilvi sisälsi tiedot yli 230 miljoonasta eri IP-osoitteesta ja sen tietokanta jatkaa edelleen kasvuaan (Wagnon 2014b). Järjestelmän ylläpitäjällä on myös mahdollisuus lisätä IP Intelligencen piiriin itsellään tiedossa olevia haitallisia IP-osoitteita tai yliajaa BrightCloud-pilven rajoituksia.

Sovellustason DoS-profiilien avulla web-sovellusta voidaan suojata tyypillisimmiltä sovelluskerroksella tapahtuvilta palvelunestohyökkäyksiltä. Profiilin avulla liikenteelle määritellään tietyt raja-arvot, joiden ylittyessä järjestelmä luokittelee tulevan liikenteen palvelunestohyökkäykseksi ja suorittaa sille sääntöjen avulla määritellyt toimenpiteet. Sovellustason DoS-profiilit voivat toimia normaalin ASM-politiikan tavoin erikseen tarkkailu- tai estotilassa.

Datansuojaus-toiminteen (engl. *data guard*) avulla voidaan estää sensitiivisen ja arkaluontoisen datan paljastumista. ASM:ään integroituna kyseisellä ominaisuudella voidaan esimerkiksi estää sovelluksen käyttäjien erehdyksissä syöttämien luottokorttinumeroiden ja henkilötunnuksien paljastumista muille käyttäjille. Halutut käyttäjäsyötteet voidaan erikseen peittää tai estää riippuen politiikassa käytettävistä asetuksista. Oletuksena järjestelmästä löytyy suojaukset luottokorttinumeroille ja

Yhdysvaltojen sosiaaliturvatunnuksille. Omia suojauksia voidaan lisäksi rakentaa käyttämällä niihin säännöllisiä lausekkeita. Kaikki datansuojaus-toiminteen toiminteet voidaan saavuttaa myös käyttämällä BIG-IP:n iRule-sääntöjä. Datansuojaus on kuitenkin ASM:n sisäänrakennettu ominaisuus ja sen tarkoituksena on alentaa ominaisuuden käyttöönottokynnystä. (Wagnon 2015.)

Erillisten maarajoitusten avulla (engl. *geolocational enforcement*) ASM:n tietoturvapolitiikkaan on mahdollista integroida pääsy suojattavaan web-sovellukseen ainoastaan tietyistä maista. Maarajoitusten yhteydessä mahdollisuutena on toteutustavasta riippuen käyttää erillisiä white- ja blacklistauksia.

6.7 Lokitus ja raportointi

ASM:n yhteydessä ensisijaisena lokitietona käytetään sen sisäistä lokitusta. Oletuksena ASM lokittaa politiikoista ainoastaan kaikki hälytyksen aiheuttaneet tapahtumat. Tapahtumien lokitus on hyvin monipuolista, ja niistä saatava tieto sisältää muun muassa hyökkäyksen tyypin, hyökkäyksen merkittävyysasteen, aikaleiman, lähde-IP-osoitteen sekä web-sovelluksen elementin, johon hyökkäys kohdistui. Oletuksena hälytyksen aiheuttaneista tapahtumista lokitetaan myös HTTP-pyyntö kokonaisuudessaan. Tämä helpottaa mahdollisen hyökkäystunnisteen identifioimista ja antaa järjestelmän ylläpitäjälle mahdollisuuden selvittää, missä HTTP-kehyksen kentässä hyökkäys tapahtui.

Useissa tapauksissa ASM:n tuottama lokitieto on järkevää keskittää erilliselle palvelimelle tai järjestelmälle. Näitä voivat olla esimerkiksi erillinen syslog-palvelin tai SIEM-järjestelmä. Lokitietojen suuren määrän säilöminen fyysiselle BIG-IP-laitteelle voi aiheuttaa levytilaongelmia etenkin suurissa ympäristöissä, joissa käytetään samanaikaisesti useita eri tietoturvapolitiikoita. Lokitietojen säilömiseen erilliselle palvelimelle on käytettävissä useita erilaisia vaihtoehtoja. Järjestelmästä voidaan esimerkiksi yksityiskohtaisesti määritellä mitä lokitietoja palvelimelle generoidaan.

Tietoturvapolitiikkojen keskitetty raportointi on toteutettu F5:n *Analytics*-toiminteen avulla. Raportointi on saatavilla ASM:ssä *Reports*-välilehden alla. ASM sisältää useita valmiita raportointipohjia, joita pystytään myös tarpeen vaatiessa kustomoimaan,

jotta ne vastaisivat paremmin raporttien käyttötarkoitusta. Raportointipohjat sisältävät erilaisia kuvaajia ja taulukoita tietyiltä ennalta määritellyiltä aika-ikkunoilta. Niitä voidaan myös luoda erikseen politiikka- tai järjestelmäkohtaisesti. Raportointitaulukot on myös mahdollista viedä ajastetusti järjestelmän ylläpitäjän sähköpostiin. Toiminteen avulla voidaan luoda esimerkiksi viikoittainen selkeä ja nopea tilannekuva eri politiikojen hälytysasteista, niihin kohdistuneista hyökkäystyypeistä sekä hyökkäysten kokonaismäärästä.

7 Toteutus ja suojauskykytestit

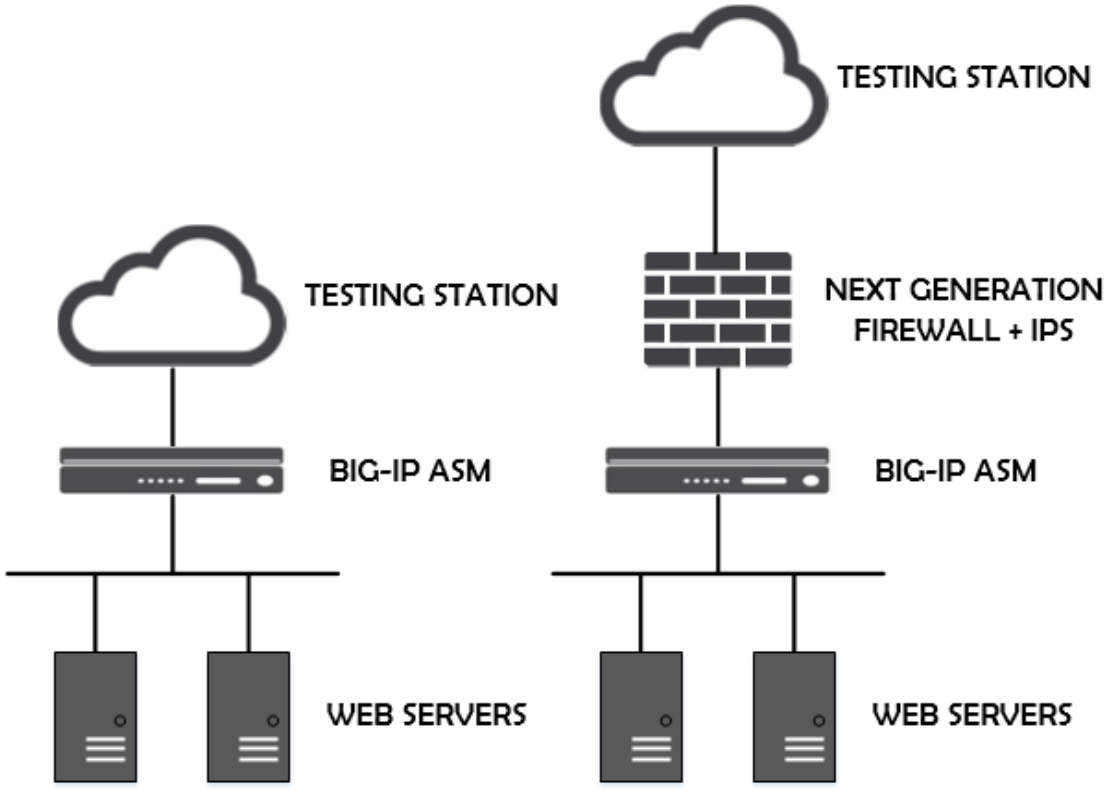
7.1 Toteutusympäristö ja toteutuksen vaiheet

Toteutus- ja testivaihe on jaettu kolmeen erilliseen osakokonaisuuteen. Toteutuksen ensimmäisessä vaiheessa luodaan kaksi erilaista ASM:n tietoturvapolitiikkaa. Ensimmäinen politikka luodaan automaattisesti käyttämällä sen rakentamiseen automaattista politiikanrakentajaa. Kyseisen politiikan oppimistasona käytetään *Fundamental*-oppimistasoa. Toinen politiikka luodaan vastaavasti manuaalisesti käyttämällä nopean käyttöönoton politiikkaa. Politiikkojen rakennusvaiheessa seurataan niiden yleistä toimintaa ja automaattisen politiikan osalta sen käyttämiä oppimisprosesseja. Molemmista politiikkatyypeistä pyritään löytämään niiden vahvuudet ja heikkoudet sekä niiden mahdollinen soveltuminen tietynlaisiin tilanteisiin.

Toteutuksen toisessa vaiheessa testataan edellisessä vaiheessa luotujen politiikkojen toimivuutta suorittamalla niiden suojaamiin web-palveluihin tyypillisiä sovellustason hyökkäyksiä. Testeihin on tarkoitus sisällyttää niin perinteisiä, yleisiä hyökkäystyyppejä, kuten OWASP Top 10 -listauksen hyökkäyksiä sekä uusia, vasta viime vuosina yleiseen tietoon tulleita hyökkäystyyppejä. Testeihin sisällytetään myös erilaisia manipulointeja HTTP-kehyksen kentissä, hyökkäyksiä koodatussa muodossa sekä hyökkäyksiä erilaisilla merkistökokotasoilla. Näiden testien avulla on tarkoitus selvittää, pystyykö ASM-politiikka suojaamaan web-sovellusta myös tyypillisimmiltä järjestelmän suojauksen kiertoyrityksiltä. Toisessa toteutusvaiheessa

pyritään myös tarkastelemaan aiemmin luotujen politiikoiden riskejä tuottaa vääriä positiivisia hälytyksiä. Testauksesta saatuja kokonaistuloksia verrataan lopuksi keskenään ja niistä tehdään tarvittavia johtopäätöksiä.

Toteutuksen kolmannessa vaiheessa tarkoituksena on testata ASM-moduulin käyttöä osana kerroksellista suojausmallia. Tässä vaiheessa verrataan kahta erilaista toteutusratkaisua: web-sovellusta, jonka suojana toimii ainoastaan seuraavan sukupolven moderni palomuuri ja siihen integroitu IPS-järjestelmä sekä web-sovellusta, jonka suojana on sekä seuraavan sukupolven palomuuri varustettuna IPS-järjestelmällä ja ASM-järjestelmä. Tarkoituksena on erillisten skriptien avulla kohdistaa näihin kahteen toteutus-skenaarioon satoja erilaisia hyökkäystunnisteita. Tämän jälkeen tarkastellaan, kuinka moni tunniste läpäisi ensimmäisen turvallisuuskerroksen (seuraavan sukupolven palomuuri) ja vastaavasti myös sitä, kuinka moni tunniste läpäisi seuraavan turvallisuuskerroksen (ASM-moduuli). Kyseisten testien avulla voidaan tehdä karkea johtopäätös kerroksellisen suojautumisen merkityksellisyydestä ja seuraavan sukupolven palomuurin kyvystä torjua sovellustason hyökkäyksiä. Kuviossa 24 on havainnollistettu testeissä käytettävien ympäristöjen topologioita.



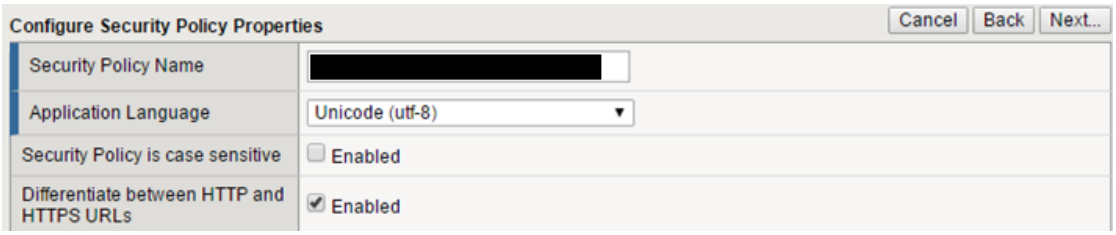
Kuvio 24. Toteutusvaiheessa käytettävä looginen verkkotopologia

Tietoturvapolitiikkojen rakentamisessa ja niiden testaamisessa sovellustason hyökkäyksiä vastaan käytetään kuvion oikeanpuoleista topologiaa. Topologiassa web-sovelluksen ja testausaseman välissä toimii ainoastaan ASM-moduuli. Vasemmanpuoleista topologiaa käytetään osittain kerroksellisuustestien suorittamiseen, jolloin tarkoituksena on selvittää seuraavan sukupolven palomuurin ja IPS-järjestelmän kyvyä torjua sovellustason uhkia.

7.2 Tietoturvapolitiikan rakentaminen

7.2.1 Automaattisesti

Automaattisen tietoturvapolitiikan luominen aloitetaan valitsemalla käyttöön politiikan asennusikkunassa automaattinen tietoturvapolitiikka, nimeämällä politiikka asiaankuuluvalla tavalla ja sitomalla se suojattavan sovelluksen F5-virtuaalipalveluun. Kuviossa 25 on havainnollistettu politiikan luomisen aloitusvalikkoa.

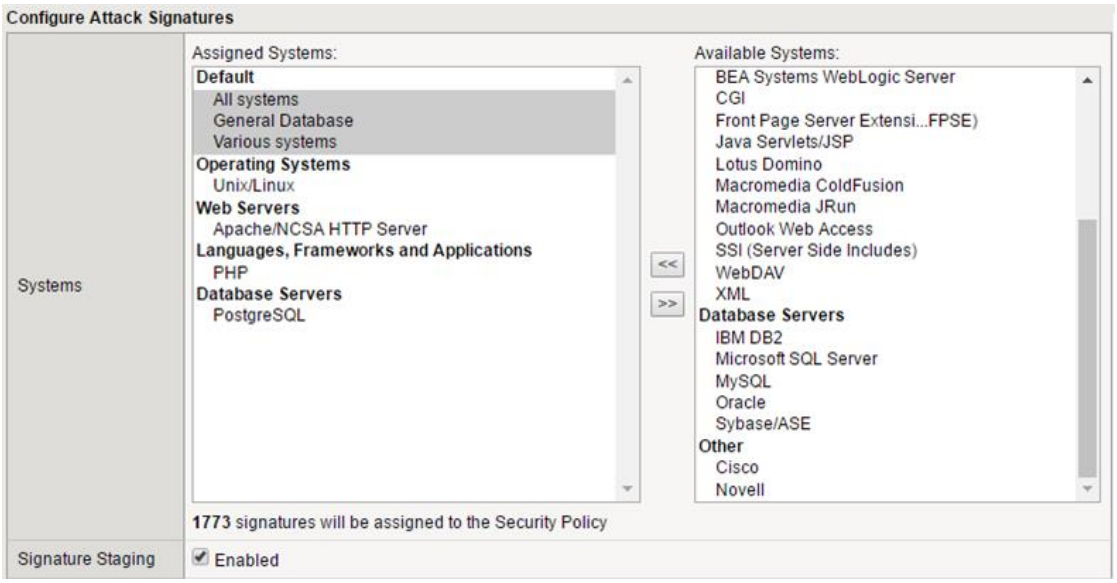


Kuvio 25. Automaattisen politiikan nimeäminen ja alkuasetuksien määritys.

Politiikan luomisvaiheessa määritellään myös muun muassa sovelluksessa käytettävä merkistöstandardi ja sovelluksen merkistökokoriippuvuus. Sovelluksen käyttämä kieli ja sen mahdollinen merkistökokoriippuvuus ovat molemmat aina syytä varmistaa erikseen sovelluksen kehittäjiltä. Tarvittaessa sovelluksen kieli voidaan määritellä myös automaatisesti, jolloin järjestelmä osaa tunnistaa käytettävän sovelluskielen sen läpi kulkevasta liikenteestä.

Politiikan rakentamisen seuraavassa vaiheessa määritellään sen yhteydessä käytettävät hyökkäystunnisteet. Hyökkäystunnisteet ovat lajiteltuna erilaisiin loogisiin ryhmiin, jotta niiden käyttöönotto olisi mahdollisimman yksinkertaista.

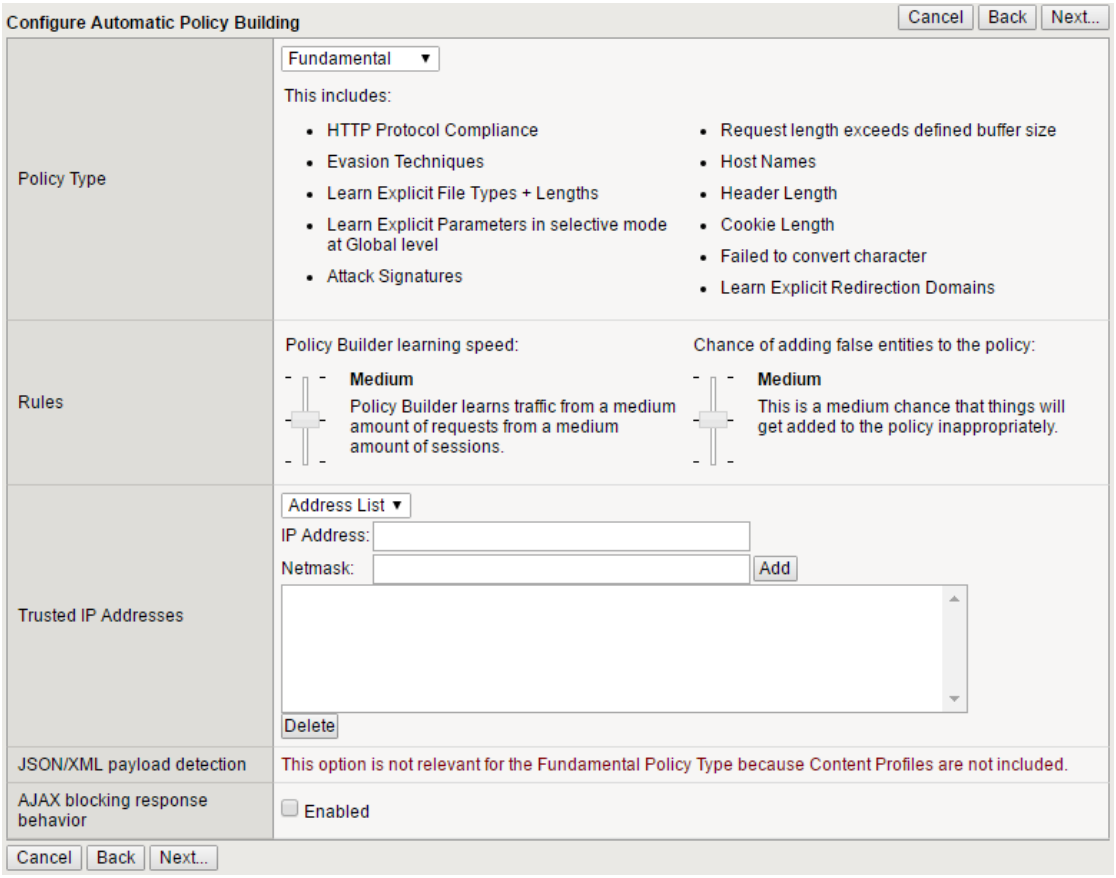
Kuviossa 26 on havainnollistettu politiikan yhteydessä käyttöönotettavia hyökkäystunnisteita.



Kuvio 26. Hyökkäystunnisteiden lisääminen automaattiseen politiikkaan

Tässä tapauksessa tunnisteista otetaan käyttöön perustunnisteet sekä web-sovelluksen alustan perusteella erilliset yksityiskohtaiset tunnistepaketit. Hyökkäystunnisteet asetetaan aluksi tarkkailutilaan, jonka aikana pystytään karsimaan niiden aiheuttamat mahdolliset väärät positiiviset hälytykset.

Automaattisen politiikan luomisen loppuvaiheessa määritellään järjestelmän käyttämät oppimisasetukset. Oppimisasetusten määrittelyvalikkoa on havainnollistettu tarkemmin kuviossa 27.



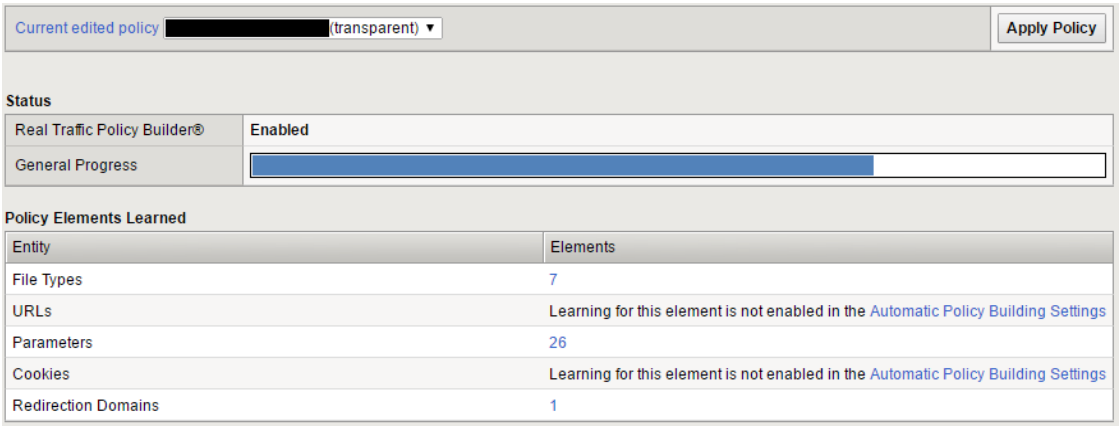
Kuvio 27. Oppimisasetuksien määrittely automaattiseen politiikkaan

Politiikan oppimistasoksi valitaan tässä tapauksessa *Fundamental*-oppimistaso. Automaattisen politiikanrakentajan oppimisnopeudeksi määritellään oletuksena käytettävä keskitaso. Samassa yhteydessä voitaisiin määritellä myös tarkemmat oppimisasetukset, kuten tuntemattomalla liikenteelle määritettävät raja-arvot. Oppimisnopeudet ja niihin liittyvät asetukset ovat määriteltävissä erikseen ASM:n politiikan elinkaaren jokaiselle vaiheelle. Lisäksi oppimisnopeudet voidaan säätää erikseen luotetulle ja epäluotetulle liikenteelle. Kuvion 27 valikon yhteydessä voitaisiin määritellä myös politiikan yhteydessä käytettäviä luotettuja lähde-IP-osoitteita tai verkkoja. Tässä tapauksessa jätämme kuitenkin kyseisen kohdan täyttämättä ja lisäämme luotetut lähde-IP-osoitteet politiikkaan myöhemmin.

Tietoturvapolitiikka on nyt luotu ja se on siirtynyt elinkaarensa ensimmäiseen vaiheeseen: politiikan löysäämiseen. Politiikkaa ajetaan aluksi globaalisti tarkkailutilassa. Myös kaikkia politiikan oppimia uusia yksittäisiä elementtejä ja käytettäviä hyökkäystunnisteita ajetaan aluksi tarkkailutilassa. Seuraavien päivien ja

viikkojen aikana sovellukselle tulee kohdistaa mahdollisimman paljon monipuolista, luotettua liikennettä, joka mahdollistaa ASM-järjestelmän tehokkaan oppimisen. Liikenteen oppimisessa helpottavana tekijänä ovat luotetut IP-osoitteet.

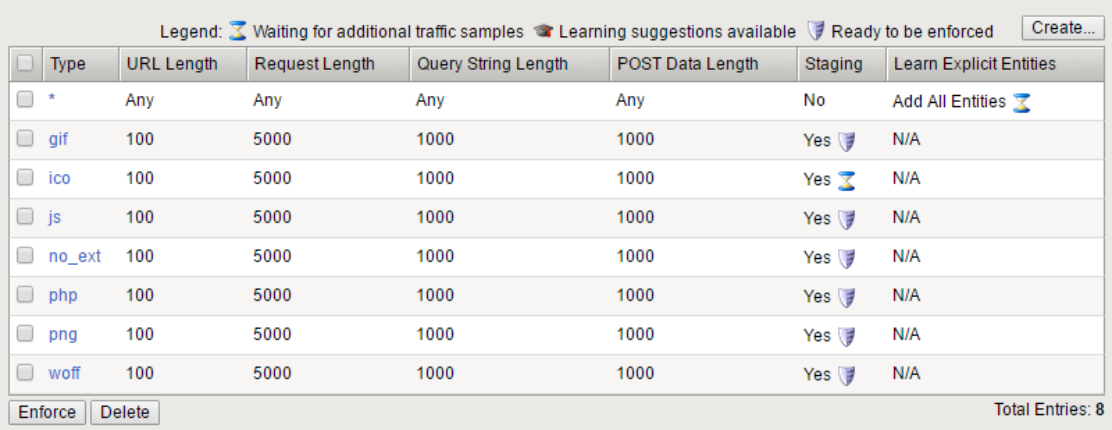
Kuviossa 28 on havainnollistettu automaattisen tietoturvapolitiikan rakentamisen edistymisen päänäkymää noin viikon kuluttua politiikan luomisesta.



Kuvio 28. Automaattisen politiikan ylläpidon päänäkymä

Päänäkymä kertoo automaattisen politiikanrakentajan tilasta ja politiikan rakentumisen kokonaisedistymisestä. Edistymistä kuvataan prosenttiosuudella ja sen tarkoituksena on havainnollistaa, kuinka pitkällä politiikan rakentaminen on. Aloitusvalikossa on myös nähtävissä politiikkaan lisättyjen yksittäisten elementtien määrä jokaisen elementtiryhmän osalta. Koska politiikkatyyppinä on käytössä perustavanlaatuinen politiikka, elementtien oppiminen on poistettu käytöstä URL- ja evästeet-elementtiryhmien osalta. Edellä mainitut kaksi elementtiryhmää sisältävät ainoastaan villi kortti -elementit. Automaattisen politiikan rakentamisen yhteydessä ei ole välttämätöntä käyttää automaattista politiikanrakentajaa. Mikäli automaattinen politiikanrakentaja on poistettu käytöstä, politiikkaan ei lisätä eikä siihen tehdä mitään muutoksia automaattisesti ilman järjestelmän ylläpitäjän hyväksyntää. Tällöin generoidut oppimisehdotukset hyväksytään politiikkaan manuaalisesti ”*Manual Traffic Learning*” välilehdeltä.

Tarkempaa katselmusta elementtiryhmien osalta voidaan tehdä navigoimalla kyseisen elementtiryhmän välilehden alle. Kuviossa 29 on havainnollistettu tiedostotyypit-elementtiryhmän sisältöä.



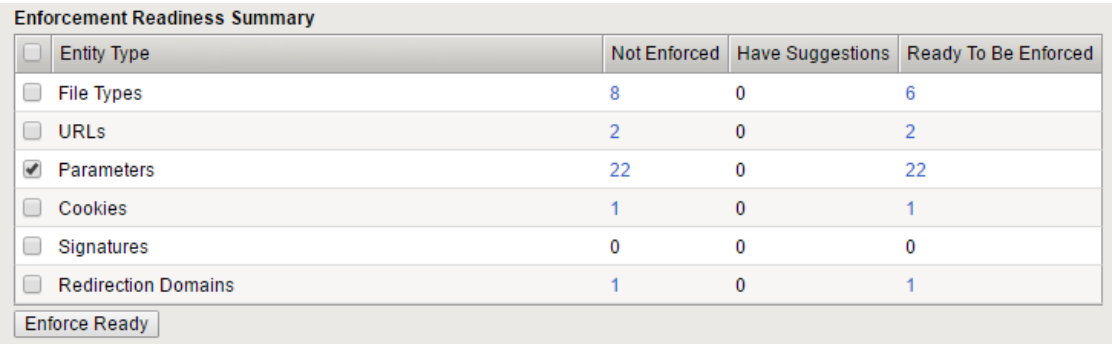
Legend: Waiting for additional traffic samples | Learning suggestions available | Ready to be enforced | Create...

Type	URL Length	Request Length	Query String Length	POST Data Length	Staging	Learn Explicit Entities
*	Any	Any	Any	Any	No	Add All Entities
gif	100	5000	1000	1000	Yes	N/A
ico	100	5000	1000	1000	Yes	N/A
js	100	5000	1000	1000	Yes	N/A
no_ext	100	5000	1000	1000	Yes	N/A
php	100	5000	1000	1000	Yes	N/A
png	100	5000	1000	1000	Yes	N/A
woff	100	5000	1000	1000	Yes	N/A

Enforce | Delete | Total Entries: 8

Kuvio 29. Elementtiryhmän opitut elementit sekä niiden ominaisuudet ja tilat

Kyseisen elementtiryhmän oppimismetodin (*Add All Entities*) perusteella siihen lisätään kaikki järjestelmän liikennevirrasta havaitsemat web-sovelluksen käyttämät tiedostotyypit, jotka osuvat elementtiryhmän alaiseen villiin korttiin. Tässä tapauksessa villinä korttina toimii tähti (*), joten politiikkaan lisätään yksinkertaisesti kaikki ASM-järjestelmän liikenteestä havaitsemat tiedostotyypit. Kaikki opitut tiedostotyypit ovat tällä hetkellä tarkkailutilassa, mutta suurin osa niistä on valmiita asetettavaksi estotilaan. Tästä ilmaisee seuraava lippu: *Ready to be enforced*. Villissä kortissa ja *ico*-tiedostotyypissä on sen sijaan tapahtunut muutoksia tarkkailuajanjakson aikana, jolloin tämä kyseinen seitsemän päivän tarkkailuajanjakso ei ole niiden osalta vielä ehtinyt päättymään. Tarkkailuajanjakson päättymistä eri elementtiryhmien osalta voidaan seurata myös erilliseltä *Enforcement Readiness Summary* -välilehdeltä. (Ks. kuvio 30.)



Enforcement Readiness Summary

Entity Type	Not Enforced	Have Suggestions	Ready To Be Enforced
File Types	8	0	6
URLs	2	0	2
Parameters	22	0	22
Cookies	1	0	1
Signatures	0	0	0
Redirection Domains	1	0	1

Enforce Ready

Kuvio 30. Elementtiryhmien elementtien tilat ja niille suositellut toimenpiteet

Välilehden taulukko kertoo jokaisen elementtiryhmän, hyökkäystunnisteiden ja uudelleenohjaus-toimialueiden osalta tarkkailutilassa olevien yksittäisten elementtien määrän. Lisäksi taulukon avulla pystytään näkemään elementtien määrän, joissa on ASM:n tarjoamia oppimisehdotuksia sekä niiden elementtien määrän, jotka ovat läpäisseet ennalta määritellyn tarkkailuajanjakson ja ovat näin ollen valmiita asetettavaksi estotilaan.

Kun ASM on lopulta oppinut tarpeeksi liikennettä ja kaikkien elementtiryhmien yksittäiset elementit on asetettu estotilaan, aloitetaan järjestelmän tuottamien lokitietojen seuraaminen. Politiikka on tällöin edelleen globaalisti tarkkailutilassa, jolloin mitään liikennettä ei estetä. Lokitietojen avulla voidaan päätellä järjestelmän tuottamat mahdolliset väärät positiiviset hälytykset tarkkailuajanjakson jälkeen. Kuviossa 31 on havainnollistettu esimerkkitapausta lokitiedosta, joka aiheutui puutteellisesta oppimisprosessista.

Violations

Violation	Severity	Learn	Alarm	Block
Illegal parameter data type Learn	Error	Yes	Yes	No

General Details

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	11767309951146397684
Time	2016-11-30 11:04:29
Request Status	
Severity	Error
Response Status Code	200
Attack Types	Parameter Tampering
Username	N/A
Session ID	bb9126fad1996e83
Source IP Address	[redacted] IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 31. Lokitieto ASM:n generoimasta hälytyksestä

Hälytys generoitiin suojattavan web-sovelluksen hakukentässä esiintyvän laittoman parametrin datatyypin vuoksi. Lokitietoa tarkasteltaessa huomattiin, että hälytyksen

aiheutti hakukentän yhteydessä toimivan parametrin *Integer*-datatyyppi. Kyseinen datatyyppi parametrissa sallii sen arvona ainoastaan numerot. Hakukentän on kuitenkin toimittava myös kirjaimilla ja muilla tarvittavilla erikoismerkeillä. Oppimalla väärä positiivinen hälytys *Learn*-lipun avulla, parametrin datatyyppi vaihdettiin *Integer*-arvosta *Alpha-Numeric*-arvoon. Muutoksen avulla kyseisessä parametrissa tapahtuvat väärät positiiviset hälytykset saatiin karsittua pois.

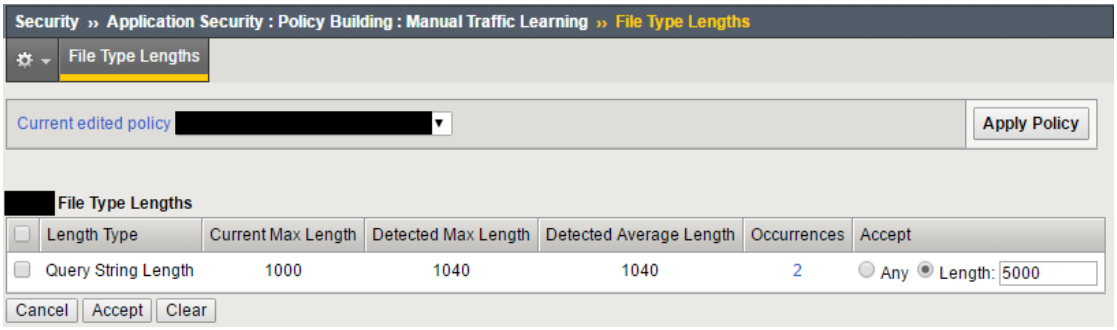
Yksittäisille parametreille voidaan tehdä myös tarkempaa säätöä. Mikäli vastaan tulee tilanne, jossa tietty erikoismerkki halutaan sallia tietyn parametrin yhteydessä, mutta ei globaalisti, voidaan se tehdä yksittäisen parametrin attribuutteja säätämällä. Kuviossa 32 on havainnollistettu esimerkkitapausta, jossa sivuston hakukentässä halutaan sallia "?"-erikoismerkki.

Kuvio 32. Erikoismerkin salliminen yksittäisessä parametrissa

Esimerkissä kyseinen merkki halutaan sallia nimenomaan vain hakukentässä käytettävän parametrin yhteydessä. ”?”-merkki on estettynä politiikassa globaalisti, mutta poikkeuksen tekemällä se saadaan sallituksi ainoastaan tietyssä yksittäisessä parametrissa. Oletuksena kyseinen tarkastustyyppi on käytössä ainoastaan *Enhanced* ja *Comprehensive* –oppistasoilla.

Kun käytössä oleva politiikka ei enää tuota vääriä positiivisia hälytyksiä, se voidaan asettaa globaalisti estotilaan. Tämän jälkeen kaikki hälytyksen aiheuttaneet tapahtumat, jotka tapahtuvat sellaisessa elementissä tai hyökkäystunnisteessa, joka ei ole tarkkailutilassa, estetään automaattisesti. Näistä hälytyksen aiheuttaneista tapahtumista generoidaan aina erikseen myös lokitieto, jonka perusteella ne voidaan hyväksyä politiikkaan, mikäli ne johtuivat väärästä positiivisesta hälytyksestä.

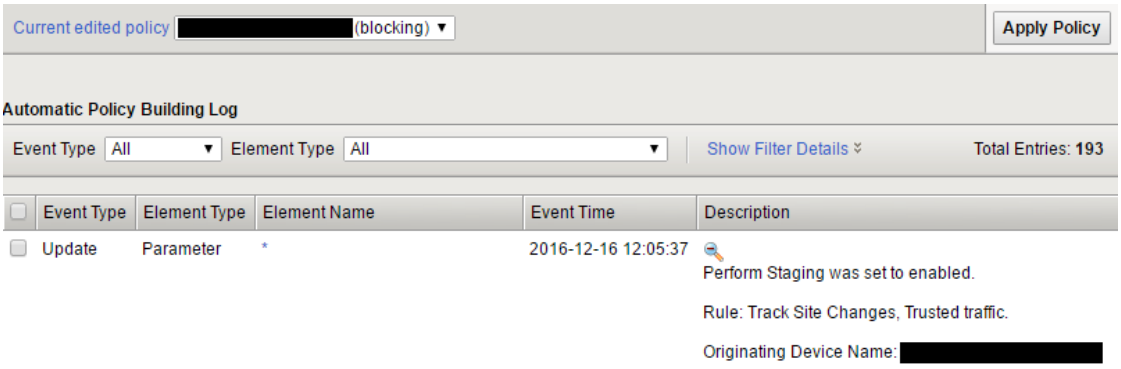
Kuviossa 33 on havainnollistettu tilannetta, jossa politiikan tiedostotyypin kyselypituus aiheutti väärän positiivisen hälytyksen politiikan ollessa globaalisti estotilassa.



Kuvio 33. Väärä positiivinen hälytys tiedostotyypissä

Politiikassa määritelty arvo kyselypituudelle kyseisen tiedostotyypin osalta oli asetettuna 1000 tavuun. ASM oli sen sijaan havainnut liikenteestä kaksi HTTP-pyyntöä, joiden kyselypituus tämän tiedostotyypin osalta ylitti politiikkaan määritellyn raja-arvon. Automaattinen politiikanrakentaja ehdotti kyselypituuden kasvattamista 5000 tavuun, joka päätettiin hyväksyä.

Kaikkien tai lähes kaikkien järjestelmän tuottamien väärien positiivisten hälytysten karsimisen jälkeen politiikan voidaan sanoa olevan vakaa. Web-sovellusten osalta tulee kuitenkin usein eteen tilanteita, jolloin itse sovellukseen tehdään ohjelmistokehittäjien toimesta muutoksia. Nämä muutokset voivat olla esimerkiksi sovelluksen uusia ominaisuuksia tai sovelluksen ulkoasun muutoksia. Sovelluksen muuttuessa on myös ASM:n tietoturvapolitiikan osattava reagoida muutoksiin asiaankuuluvalla tavalla. Sivuston muutosten valvonnan (engl. *track site changes*) avulla estotilassa olevaa politiikkaa voidaan tilapäisesti löysätä automaattisesti, jotta järjestelmä ehtii oppimaan web-sovelluksen käyttämät uudet rakenteelliset elementit. Löysäämisellä tarkoitetaan tässä tapauksessa uusien elementtien lisäämistä politiikkaan ja niiden asettamista tarkkailutilaan. Kuviossa 34 on todennettu automaattisen politiikanrakentajan lokia sen jälkeen, kun sovellukseen tehtiin muutos.

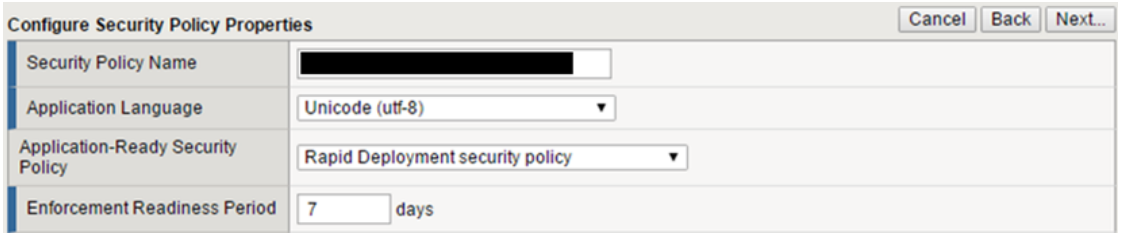


Kuvio 34. Sovelluksen muuttuminen ja automaattinen politiikanrakentaja

Muutos sovelluksen käyttämissä parametreissa johti parametrit-elementtiryhmän villin kortin asettamiseen takaisin tarkkailutilaan. Koska kyseisessä elementtiryhmässä on käytössä valikoiva oppiminen, antaa toimenpide ryhmälle aikaa lisätä siihen uusia tarkkoja elementtejä sekä säätää näiden yksittäisiä attribuutteja. Kun ASM näkee sovelluksen olevan jälleen vakaa ja tarkkailuajanjakso on umpeutunut uusien vastaopittujen elementtien osalta, järjestelmä ehdottaa jälleen politiikan vakauttamista ja kiristämistä.

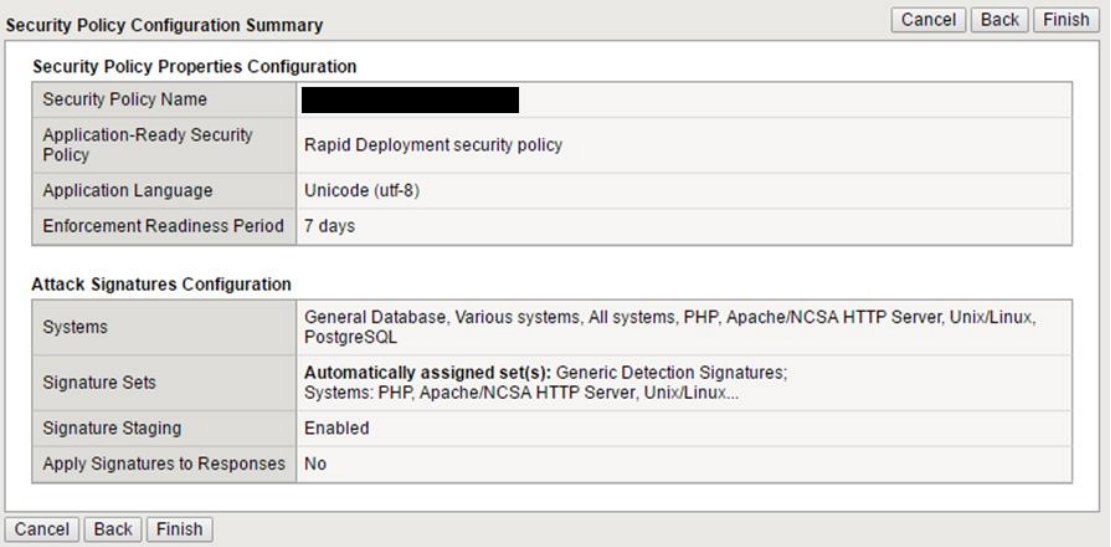
7.2.2 Manuaalisesti

Manuaalisen politiikan rakentamiseen käytetään nopean käyttöönoton tietoturvapolitiikkaa. Toinen vaihtoehtoinen toteutustapa on käyttää sovellusvalmiita tietoturvapolitiikkapohjia. Manuaalisen tietoturvapolitiikan luomisen yhteydessä tehtävät asetukset ja määritykset vastaavat lähes täysin automaattisen politiikan yhteydessä tehtäviä asetuksia. Kuviossa 35 on havainnollistettu manuaalisen politiikan luomisen yhteydessä määriteltäviä pohja-asetuksia.



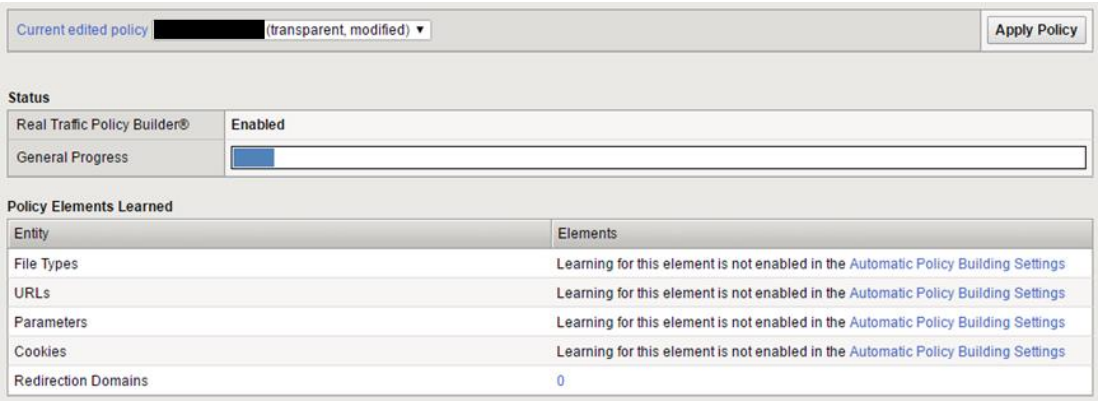
Kuvio 35. Manuaalisen politiikan luonti nopealla käyttöönotolla

Automaattisen tietoturvapolitiikan tapaan manuaaliselle politiikalle määritellään tarkkailuajanjakso ja politiikan yhteydessä käyttöönotettavat hyökkäystunnisteet. Kuviossa 36 on havainnollistettu yhteenvetoa luodusta politiikasta.



Kuvio 36. Manuaalisen politiikan asetukset

Hyökkäystunnisteiden merkitys korostuu nopean käyttöönoton politiikassa, sillä tässä politiikkatyypissä ei ole tarkoituksena suoranaisesti oppia mitään web-sovelluksen käyttämiä rakenteita. Politiikan tarkoituksena on sen sijaan toimia ainoastaan negatiivisen tietoturvamallin periaatteella, jonka yhteydessä HTTP-viestien otsakekentille suoritetaan erilaisia säännönmukaisuustarkistuksia. Nopean käyttöönoton politiikassa on kuitenkin mahdollisuus käyttää automaattista politiikanrakentajaa, ja siihen on myös mahdollista lisätä manuaalisesti web-sovelluksen käyttämiä rakenteellisia elementtejä. Kyse on siis ainoastaan manuaalisen politiikan oletustoiminnasta. Kuviossa 37 on havainnollistettu manuaalisen turvallisuuspolitiikan ylläpidon päänäkymää.



Kuvio 37. Manuaalisen politiikan ylläpidon päänäkymä

Oletuksena oppiminen on poistettu käytöstä jokaisen elementtiryhmän osalta. Automaattisen politiikan tavoin manuaalinen politiikka sisältää tarkkailuajanjakson, estotilan, tarkkailutilan ja mahdollisuuden käyttää automaattista politiikanrakentajaa (kuten kuviossa 37). Manuaaliseen politiikkaan voidaan sen nimensä mukaisesti lisätä manuaalisesti uusia yksittäisiä elementtejä ja säätä niiden attribuutteja. Tämä vaatii kuitenkin järjestelmän ylläpitäjältä kattavaa tietämystä suojattavasta web-sovelluksesta sekä sen käyttämistä rakenteista.

Tarkkailuajanjakso toimii manuaalisessa tietoturvapolitiikassa samalla periaatteella kuin automaattisessa politiikassa. Nopean käyttöönoton politiikassa tarkkailujakso rajoittuu kuitenkin lähinnä hyökkäystunnisteista aiheutuvien mahdollisten väärien positiivisten hälytysten karsimiseen. Tarkkailuajanjakson päättymisen jälkeen hyökkäystunnisteet voidaan asettaa estotilaan. Tämän jälkeen politiikalle voidaan suorittaa vielä automaattisen politiikan tavoin viimeistelytoimenpiteet, joilla vältetään mahdolliset väärät positiiviset hälytykset ennen globaaliin estotilaan siirtymistä.

7.3 Politiikoiden suojauskyky sovellustason hyökkäyksiä vastaan

Toteutuksen toisessa vaiheessa tarkoituksena on testata kahden aiemmin luodun ASM:n tietoturvapolitiikan suojauskykyä, kun niiden suojaamiin web-sovelluksiin kohdistetaan erilaisia hyökkäysvektoreita. Koska molemmat käyttöönotetut politiikat hyödyntävät toiminnassaan negatiivisen tietoturvamallin pohjalta ASM:n hyökkäystunnistetietokantaa, pyritään testeissä löytämään näyttöä myös positiivista

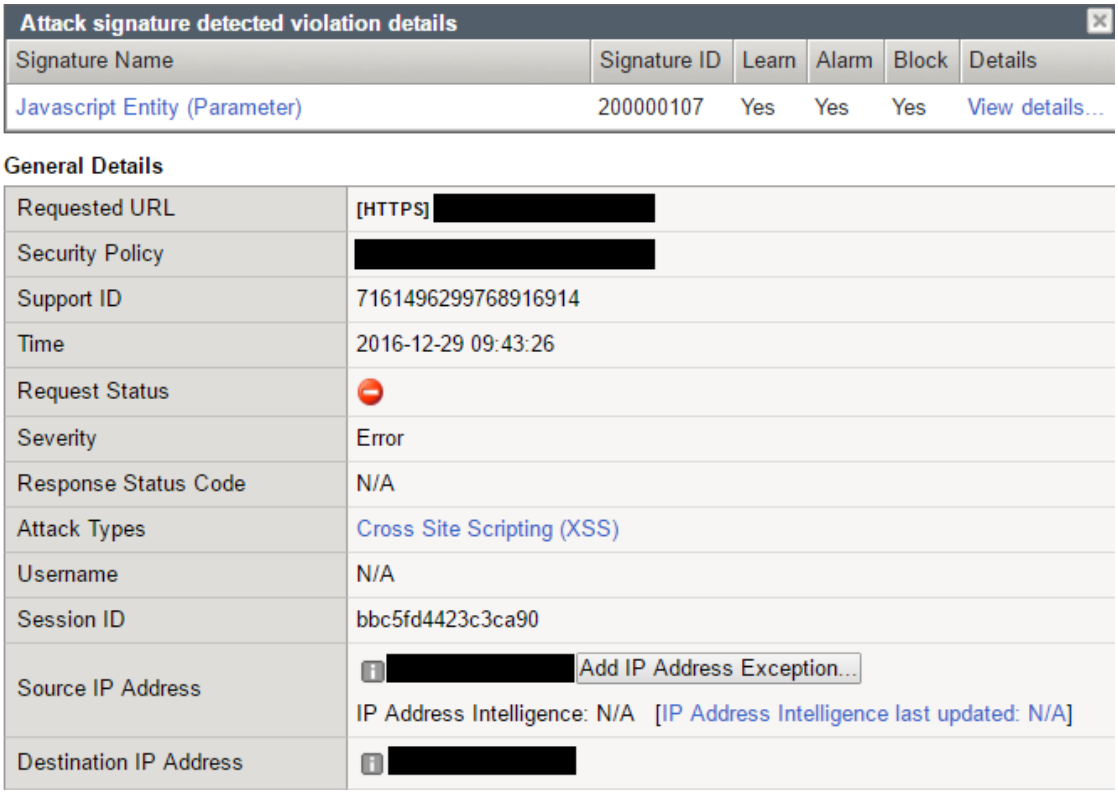
tietoturvamallia hyödyntävän automaattisen tietoturvapolitiikan eduista. Hyökkäysten kohteena käytettiin erillisessä testiympäristössä yksinkertaista PHP-pohjaista web-sovellusta ja myös osittain DVWA-sovellusta (Damn Vulnerable Web Application). DVWA-sovellus on nimensä mukaisesti erittäin haavoittuvainen avoimeen lähdekoodiin perustuva testikäyttöön suunniteltu web-sovellus.

Testausvaiheen suorittamisessa hyödynnettiin osittain NSS Labs Inc:n web-sovellustason palomuurin testausohjetta. Ohjeen mukaan web-sovellustason palomuuri tulee testata kattavasti erilaisten sovellustason hyökkäysten varalta, käyttäen myös modifioituja hyökkäyslausekkeita. Testaukseen tulee sisällyttää muun muassa OWASP Top 10 -listauksen uhkia. Lisäksi ohjeessa määritellään järjestelmän testaaminen tyypillisimmiltä web-sovellustason palomuurin kiertoyrityksiltä, virtuaalisen paikkauksen testaaminen sekä omien, kustomoitujen ja tapauskohtaisten torjuntamekanismien rakentaminen ja niiden kattava testaaminen. Testien perusteella tulee kyetä arvioimaan järjestelmän toimivuutta myös kriittisesti testaamalla sen tuottamia mahdollisia vääriä negatiivisia ja vääriä positiivisia hälytyksiä. (Test Methodology v2.0 N.d.)

Varsinaiset testit aloitetaan kohdistamalla järjestelmään muutamia erilaisia yksinkertaisia XSS- ja injektio-lausekkeita. Lausekkeita sijoitetaan sovelluksen käyttämien parametrien arvojen lisäksi evästeisiin ja HTTP-kehyksen eri kenttiin. Niitä kokeillaan syöttää myös erilaisilla merkistön kokojen vaihteluilla. Aluksi testattiin haitallisen tunnisteen sijoittamista web-sovelluksen käyttämän yksittäisen parametrin arvoksi. Tässä yhteydessä tunnisteissa testattiin muun muassa niiden merkistökokoriippuvuuden vaikutusta ja syötteiden välissä olevia tyhjiä kenttiä:

```
<script>alert(1)</script>
<IMG SRC=JaVAsCrIpt:alert(‘XSS’)>
<IMG SRC=jav    ascript:alert(‘XSS’)>
'';!--"<XSS>=&{()}
<b onmouseover=alert(’dong!’)>click this!</b>
```

Kuvion 38 todennuksessa on lokitieto ASM:n tietoturvapolitiikan torjumasta yksittäisestä XSS-hyökkäyksestä.



Attack signature detected violation details

Signature Name	Signature ID	Learn	Alarm	Block	Details
Javascript Entity (Parameter)	200000107	Yes	Yes	Yes	View details...

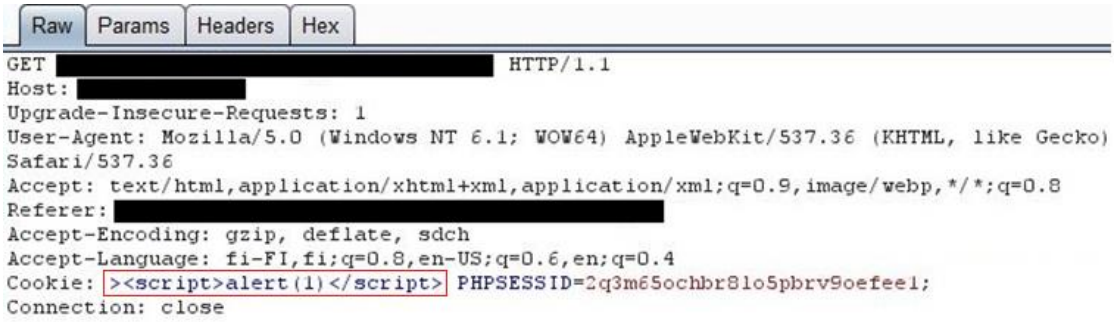
General Details

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	7161496299768916914
Time	2016-12-29 09:43:26
Request Status	
Severity	Error
Response Status Code	N/A
Attack Types	Cross Site Scripting (XSS)
Username	N/A
Session ID	bbc5fd4423c3ca90
Source IP Address	[redacted] Add IP Address Exception... IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 38. Lokitieto XSS-hyökkäyksen torjumisesta

Kyseisessä lokitiedossa tunniste estettiin sen sisältämän ”*<script>*"-merkistön perusteella. Järjestelmä esti myös kaikki muut siihen kohdistetut XSS-lausekkeet oman hyökkäystunnistetietokantansa avulla. Näihin estettyihin tunnisteisiin kuuluivat myös erilaisilla merkistökokovaihteluilla ja syötteen välissä olevalla tyhjällä tilalla olevat tunnisteet. XSS-tunnisteiden ohella sovelluksen lomakkeen kentässä kokeiltiin myös erilaisten yksinkertaisten injektiotunnisteiden lähettämistä, mutta myös näiden osalta tulokset olivat samankaltaiset. Molemmat käytetyt politiikkatyypit estivät kaikki niihin kohdistetut injektio-lausekkeet.

Haitallisia tunnisteita sijoitettiin seuraavaksi HTTP-kyselyn otsakekenttiin. Toimenpiteen suorittamiseen käytettiin apuna BurpSuiten *proxy*-työkalua. Tunnisteita kokeiltiin sijoitettavaksi muun muassa evästeisiin ja *user-agent*-kenttään. Kuviossa 39 HTTP-kehyksen evästekenttään sijoitetaan BurpSuiten *proxy*-työkalun avulla yksinkertainen XSS-tunniste.



Kuvio 39. XSS-lauseke HTTP-kehyksen evästeessä

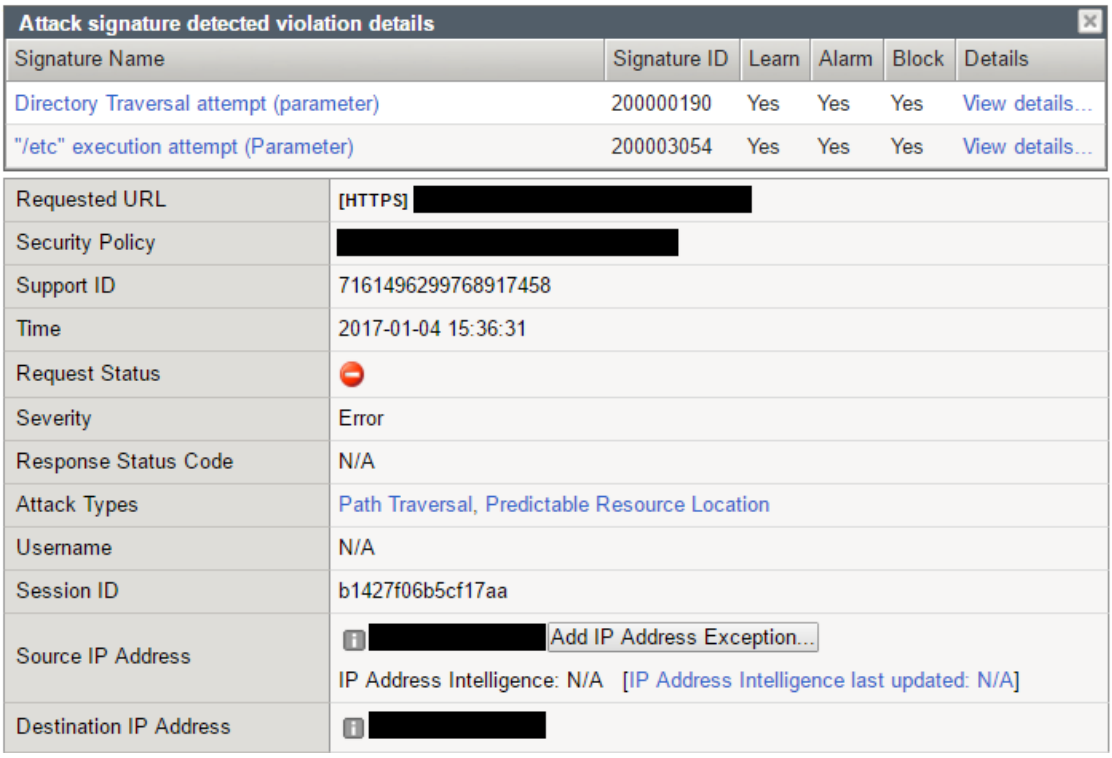
Evästeeseen ja *user-agent* –kenttään sijoitetut tunnisteet estettiin ASM:n toimesta molempien politiikkatyyppien osalta. Tässä testitapauksessa eroavaisuuksia järjestelmän toiminnassa ei havaittu, vaikka haitallinen tunniste lähetettäisiin HTTP-kehyksen otsakkeessa.

Paikallisen tiedon solutushyökkäyksissä yritettiin kahta yksinkertaista hyökkäyslausetta. Molemmissa tapauksissa tarkoituksena oli päästä hyödyntämään web-sovelluksen olemassa olevaa haavoittuvuutta ja päästä tätä kautta käsiksi web-sovelluksen pohjalla toimivan alustan käyttämiin tietojärjestelmiin.

```
../../../../../../../etc/passwd%00
```

```
<iframe width="420" height="315" src="../apache/logs/ac-
cess.log" frameborder="0" allowfullscreen></iframe>
```

Kyseiset lausekkeet eivät kuitenkaan koskaan päässeet kohdejärjestelmälle asti, sillä ASM-politiikat torjuivat nämä hyökkäykset. Kuviossa 40 on todennus ASM:n havaitsemasta ja estämästä paikallisen tiedon solutushyökkäyksestä.



Attack signature detected violation details

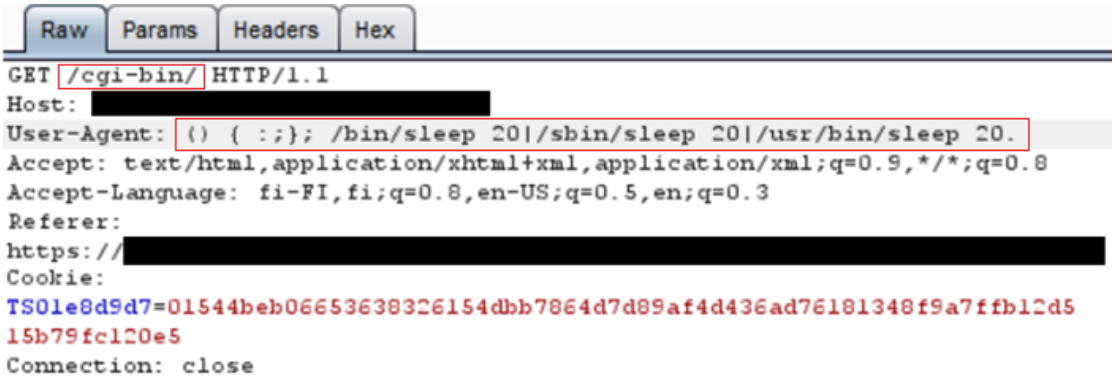
Signature Name	Signature ID	Learn	Alarm	Block	Details
Directory Traversal attempt (parameter)	200000190	Yes	Yes	Yes	View details...
"/etc" execution attempt (Parameter)	200003054	Yes	Yes	Yes	View details...

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	7161496299768917458
Time	2017-01-04 15:36:31
Request Status	
Severity	Error
Response Status Code	N/A
Attack Types	Path Traversal, Predictable Resource Location
Username	N/A
Session ID	b1427f06b5cf17aa
Source IP Address	[redacted] Add IP Address Exception... IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 40. ASM-politiikan estämä paikallisen tiedoston solutus

Tässä varsinainen estotapahtuma aiheutui parametrin kielletystä *"/etc"* –arvosta sekä hakemistorakenteessa siirtymisestä.

Molempien politiikkatyyppien osalta niiden suojaamaan web-sovellukseen kohdistettiin *shellshock*-haavoittuvuutta hyödyntävä hyökkäyslauseke BurpSuiten avulla. (Ks. kuvio 41.)



Raw | Params | Headers | Hex

```
GET /cgi-bin/ HTTP/1.1
Host: [redacted]
User-Agent: () { :;}; /bin/sleep 20|/sbin/sleep 20|/usr/bin/sleep 20.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fi-FI,fi;q=0.8,en-US;q=0.5,en;q=0.3
Referer:
https://[redacted]
Cookie:
TS01e8d9d7=01544beb066536380261546dbb7864d7d89af4d436ad76181348f9a7ffb12d5
15b79fc120e5
Connection: close
```

Kuvio 41. Shellshock-hyökkäys

Hyökkäyksen tavoitteena on päästä käsiksi kohdejärjestelmään komentotulkkiin. ASM:n tietoturvapolitiikoiden torjuma *shellshock*-hyökkäys on todennettu kuviossa 42.

Attack signature detected violation details

Signature Name	Signature ID	Learn	Alarm	Block	Details
Bash Shellshock execution attempt (Parameter)	200003168	Yes	Yes	Yes	View details...

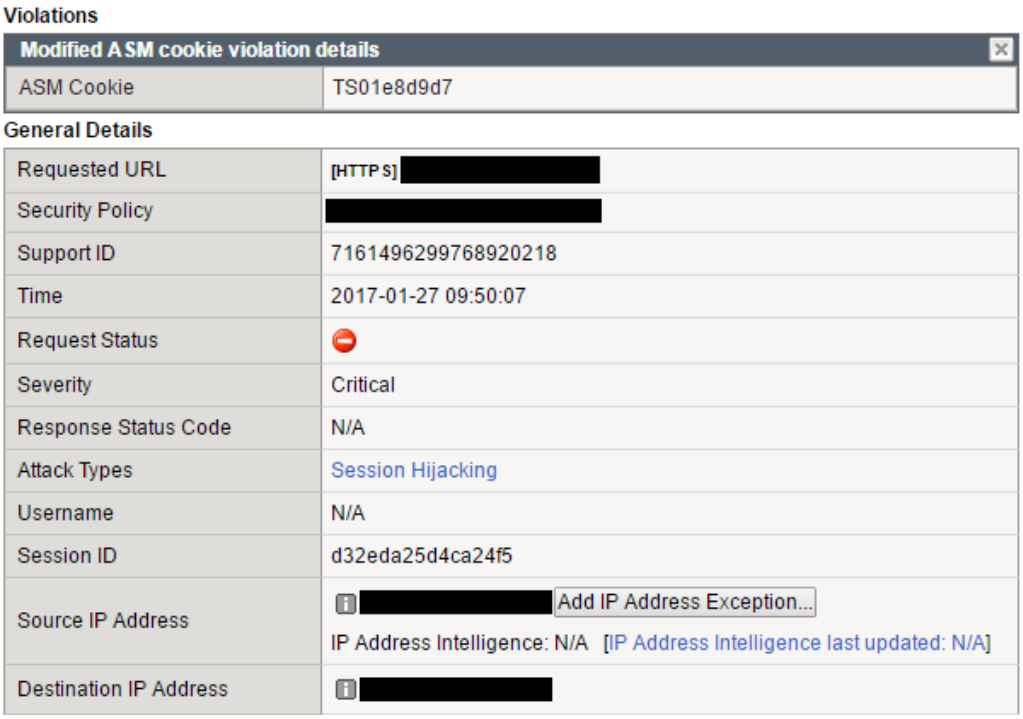
General Details

Requested URL	[HTTP S] [redacted]
Security Policy	[redacted]
Support ID	7161496299768917216
Time	2016-12-29 14:10:15
Request Status	
Severity	Error
Response Status Code	N/A
Attack Types	Command Execution
Username	N/A
Session ID	bbc5fd4423c3ca90
Source IP Address	[redacted] IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 42. Lokitieto ASM-politiikan estämästä shellshock-hyökkäyksestä

Järjestelmä havaitsi ja esti onnistuneesti myös suhteellisen uuden *shellshock*-haavoittuvuutta hyödyntävän hyökkäystyypin. Myös tämä hyökkäys estettiin ASM:n hyökkäystunnistetietokannan perusteella.

Sivuston käyttämien evästeiden muokkaamiseen käytettiin BurpSuiten *Intruder*-työkalua. Kaikki evästeeseen tehdyt muutokset, kuten väärennetyllä evästeellä tehdyt kyselyt jäivät kuitenkin välittömästi kiinni molempien ASM-politiikkojen suojaukseen. (Ks. kuvio 43.)



Violations

Modified ASM cookie violation details	
ASM Cookie	TS01e8d9d7

General Details

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	7161496299768920218
Time	2017-01-27 09:50:07
Request Status	
Severity	Critical
Response Status Code	N/A
Attack Types	Session Hijacking
Username	N/A
Session ID	d32eda25d4ca24f5
Source IP Address	[redacted] Add IP Address Exception... IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 43. Lokitieto evästemanipulaation estosta

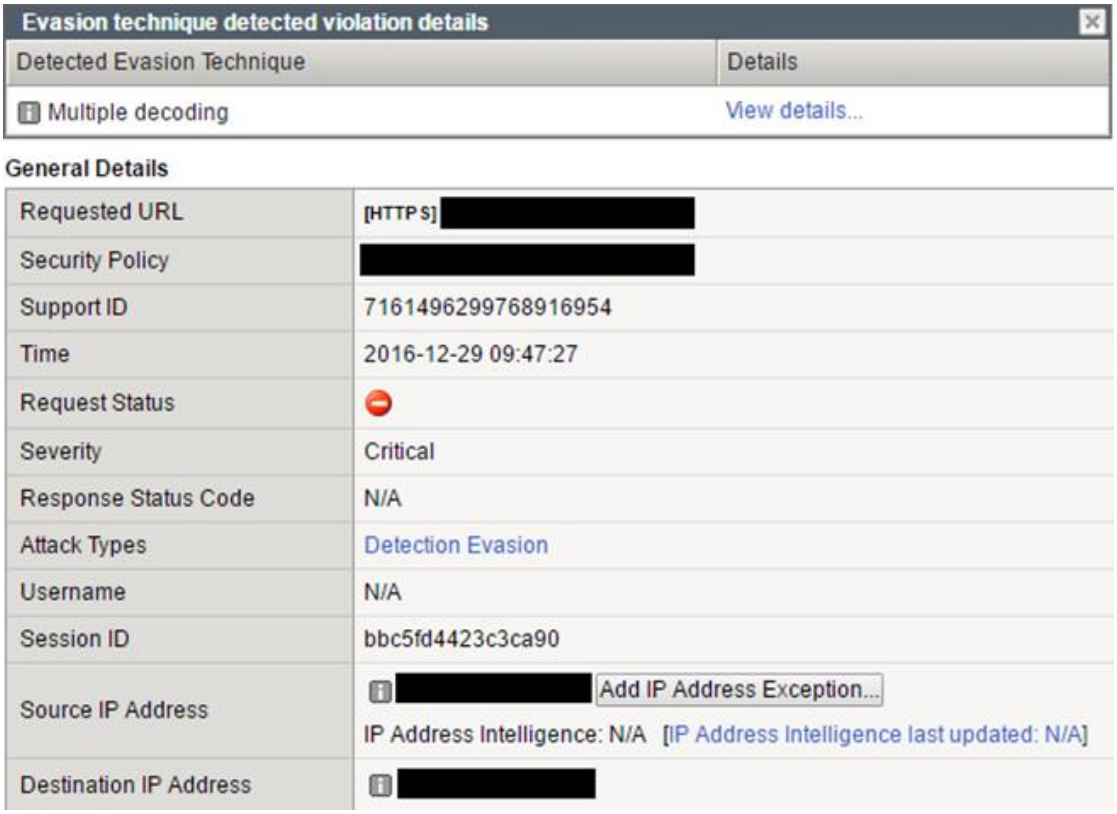
Evästeen muokkaamisen lisäksi kokeiltiin muun muassa väärennetyn evästeen käyttöä. Myös tämä yritys jäi kiinni ASM:n evästeille suorittamiin eheystarkistuksiin.

Web-sovellustason palomuurin suojauksen kiertämiseen on myös olemassa muutamia yleisesti käytettyjä keinoja. Lähes kaikkien kiertotekniikoiden pohjana on hyökkäyslausekkeiden jonkinasteinen piilottaminen. Kiertoyrityksistä ASM:n tietoturvapolitiikoita vastaan yritettiin kahta erillistä piilotettua hyökkäyslauseketta: koodattua paikallisen tiedon solutusta sekä XSS-lausekkeen kaksinkertaista URL-koodausta, jossa lausekkeen käyttämät ”>”-, ”<”- ja ”/”-merkit on korvattu niiden kaksinkertaisesti koodatuilla muodoilla:

```
..2f..2f..2f..2f..2fetc2fpasswd%00

%253Cscript%253Ealert(‘document.cookie’)%253C%252Fscript%253E
<IMG SRC=j&#X41vascript:alert(‘testing’)>
```

Molemmat ASM:n tietoturvapolitiikat havaitsivat onnistuneesti nämä järjestelmän kiertoyritykset ja niissä käytettävän koodauksen. Kuvion 44 todennuksessa on lokitieto ASM:n onnistuneesti estämästä kaksinkertaisesti heksakoodatusta XSS-hyökkäyksestä.



Kuvio 44. Moninaisesti koodatun XSS-hyökkäyksen esto

Palomuurin injektio-hyökkäyksen havainnointikykyä pyrittiin kiertämään käyttämällä lausekkeissa hyväksi kommenttikenttiä:

```
id=1/**/union/*,*/select/*,*/pwd/*,*/from/*,*/users

/?id=1/**/union/*&id=*/select/*&id=*/pwd/*&id=*/from/*&id=*/us-
ers
```

Kommenttikenttien hyödyntämisen avulla ei saatu läpäistyä kummankaan tietoturvapolitiikan suojausta. Kuviossa 45 on todennettu lokitietoa estetyistä tunnisteista.

Violations

Violation		Severity	Learn	Alarm	Block
Attack signature detected	Learn	Error	Yes	Yes	Yes
Evasion technique detected	Learn	Critical	Yes	Yes	Yes

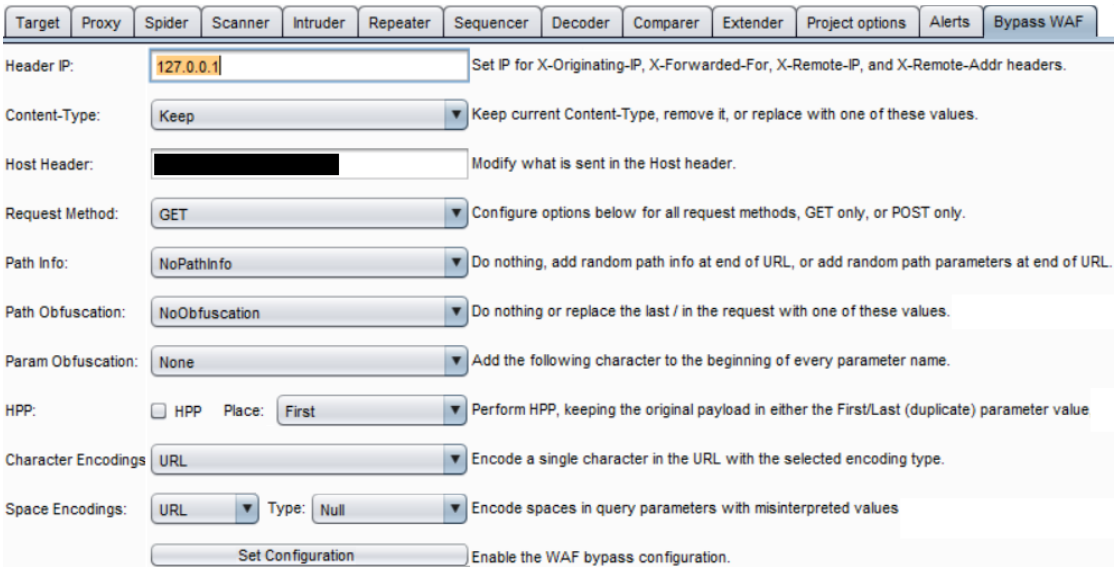
General Details

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	7161496299768913499
Time	2017-01-27 10:37:08
Request Status	
Severity	Critical
Response Status Code	N/A
Attack Types	Detection Evasion
Username	N/A
Session ID	405d1bcbe7cf405a
Source IP Address	[redacted] Add IP Address Exception... IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A
Destination IP Address	[redacted]

Kuvio 45. Kommenttikenttien avulla kierretyn injektio-hyökkäyksen esto

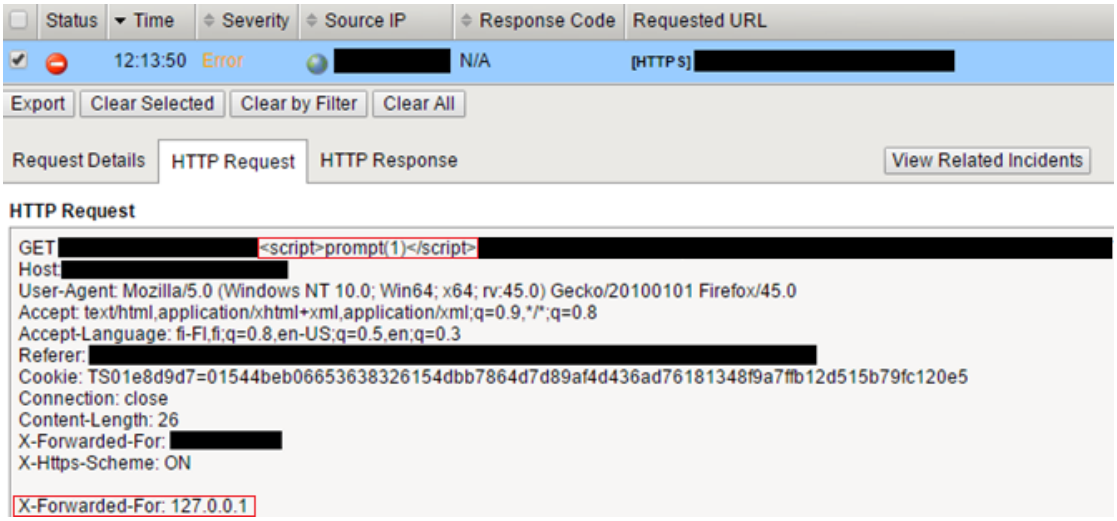
Järjestelmä havaitsi syötteissä sen käyttämän tunnistetietokannan perusteella hyökkäystunnisteen ja SQL-lausekkeen kommenttikentän avulla tehdyn kiertoyrityksen.

Kiertoyrityksistä viimeisenä kokeiltiin HTTP-kehyksen *X-Forwarded-For*-kentän manipulointia. BurpSuiten avulla HTTP-kehyksen *X-Forwaded-For*-kentän arvoksi määriteltiin *localhost*-osoite (127.0.0.1), jonka avulla pyrittiin saamaan järjestelmä luulemaan HTTP-pyynnön generoituneen alun perin siltä itseltään. Järjestelmän uskottiin tämän avulla voivan mahdollisesti päästää läpi kehyksen toisessa kentässä sijaitsevan haitallisen tunnisteen. Kiertoyritys suoritettiin BurpSuiten *Bypass WAF* -työkalun avulla. Kuviossa 46 on todennus BurpSuiten *Bypass WAF* -valikosta ja siihen määritellyistä kentistä.



Kuvio 46. WAF:n kiertoyritys

Asetuksen avulla *localhost*-osoite asetettiin myös HTTP-kehyksen *X-Originating-IP, X-Remote-IP* ja *X-Remote-Addr* -kentille. Järjestelmän kiertoyritys ei kuitenkaan tuottanut tulosta tässäkään kokeilutapauksessa. (Ks. kuvio 47.)



Kuvio 47. WAF:n kiertoyrityksen estäminen

Manipulaation yhteydessä syötetty XSS-tunniste jäi kiinni järjestelmän suojaukseen myös kehyksen *X-Forwarded-For*-kentän ollessa *localhost*-osoite. Todennuksessa on nähtävissä myös virtuaalipalvelun HTTP-profiilin lisäämä alkuperäinen *X-Forwarded-For* -kenttä.

Automaattisen ja manuaalisen tietoturvapolitiikan eroavaisuuksia erilaisten hyökkäysten torjumisessa on kohtuullisen vaikea demonstroida suorittamalla perinteisiä tunnisteisiin perustuvia hyökkäyksiä, sillä automaattinen politiikka suojaa web-sovellusta pääasiassa tuntemattomilta ja kustomoiduilta hyökkäyksiltä. Automaattisen politiikan hyötyjä voidaan kuitenkin demonstroida tilanteella, jossa web-sovellukseen yritetään soluttaa oma paikallinen tekstitiedosto. Kyseistä toimintatapaa voidaan käyttää hyödyksi esimerkiksi syöttämällä tähän sovelluksessa olevaan txt-tiedostoon sivustolla vierailevien käyttäjien evästeitä. Kuviossa 48 on havainnollistettu automaattisen tietoturvapolitiikan torjumaa tekstitiedoston solutusta.

Violations

Violation		Severity	Learn	Alarm	Block
Illegal URL length	Learn	Warning	Yes	Yes	Yes
Illegal file type	Learn	Critical	Yes	Yes	Yes
Illegal request length	Learn	Warning	Yes	Yes	Yes

General Details

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	7161496299768916450
Time	2016-12-16 09:27:41
Request Status	
Severity	Critical
Response Status Code	N/A
Attack Types	Buffer Overflow, Forceful Browsing
Username	N/A
Session ID	cd3b4d673dbcbb6
Source IP Address	[redacted] IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 48. Automaattisen tietoturvapolitiikan estämä tiedoston solutus

Sovellukseen yritettiin soluttaa .txt-päätteinen tekstitiedosto, jota ei oltu opittu tietoturvapolitiikkaan sen rakennusvaiheessa. Automaattinen tietoturvapolitiikka esti tämän epäilyttävän toiminnan laittoman tiedostotyypin, URL-pituuden ja kyselypituuden perusteella.

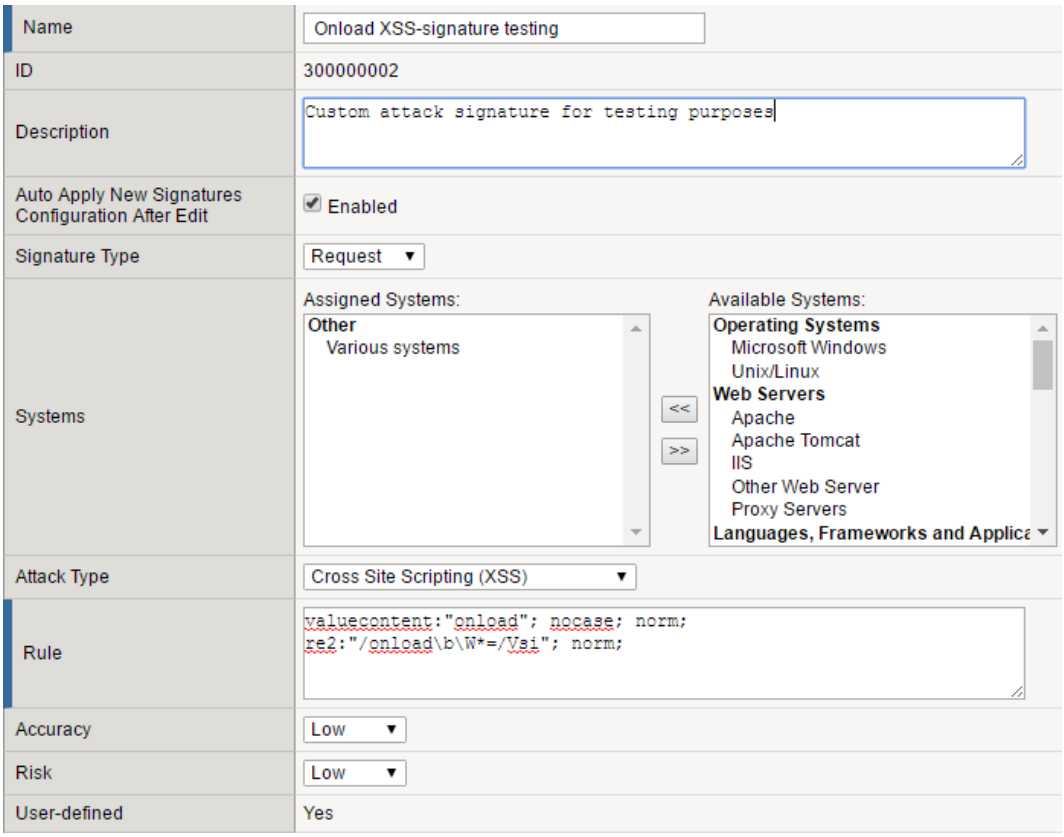
Kustomoituja omia hyökkäystunnisteita voidaan rakentaa tilanteissa, joissa web-sovelluksesta on paljastunut haavoittuvuus, jolle ei ole vielä olemassa erillistä F5:n omaa hyökkäystunnistetta. Kustomoidun hyökkäystunnisteen rakentamisessa voidaan käyttää apuna säännöllisiä lausekkeita. Tästä esimerkkinä luodaan hyökkäystunniste web-sovelluksen onload-parametria hyödyntäville XSS-hyökkäyksille. Tunnisteen tarkoituksena on hakea *onload*-parametria ilman merkistökokoriippuvuutta (nocase). *Norm*-vivun ansiosta pyynnölle suoritetaan ASM:n toimesta normalisointi, jolloin arvoa haetaan myös erilaisten koodaustekniikoiden takaa.

```
valuecontent:"onload"; nocase; norm;
```

Täsmällisen sanan hakua varten rakennetaan erillinen säännöllinen lauseke, jota käytetään tunnisteen yhteydessä. Säännöllisessä lausekkeessa käytetään re2-ohjelmistokirjastoa.

```
re2:"/onload\b\W*=/Vsi"; norm;
```

Kuviossa 49 on havainnollistettu kustomoidun hyökkäystunnisteen rakentamista ASM:n käyttöliittymästä käsin.



Kuvio 49. Kustomoitu säännöllisellä lausekkeella rakennettu hyökkäystunniste

Tunnistetta luotaessa sille määritellään muun muassa luokka, kategoria ja tyyppi. Lisäksi voidaan vielä erikseen määritellä tunnisteen riskitaso. Säännön yhteyteen voidaan sijoittaa myös tietynlaisia referenssejä, joiden avulla kyseisen hyökkäystunnisteen estämät HTTP-kyselyt voidaan identifioida kuuluvaksi tiettyyn luokkaan. Yhtenä hyvänä käytäntönä on määritellä kustomoidun tunnisteen referenssiksi tunnisteen torjuman hyökkäyksen CVE-numero (Common Vulnerabilities and Exposures).

Luodun tunnisteen testaamiseksi käytetään seuraavaa XSS-tunnistetta, joka sisältää *onload*-parametrin.

```
&#34;&#62;<svg><style>{-o-link-source&colon;'<body/onload=con-
firm(1)>'
```

Kuvion 50 todennuksessa on lokitieto politiikan estämästä XSS-tunnisteesta. Tunniste estettiin perustuen aikaisemmin luotuun kustomoituun hyökkäystunnisteeseen.

Violations

Violation	Severity	Learn	Alarm	Block
Attack signature detected Learn	Error	Yes	Yes	Yes

Details for Attack Signature 300000002

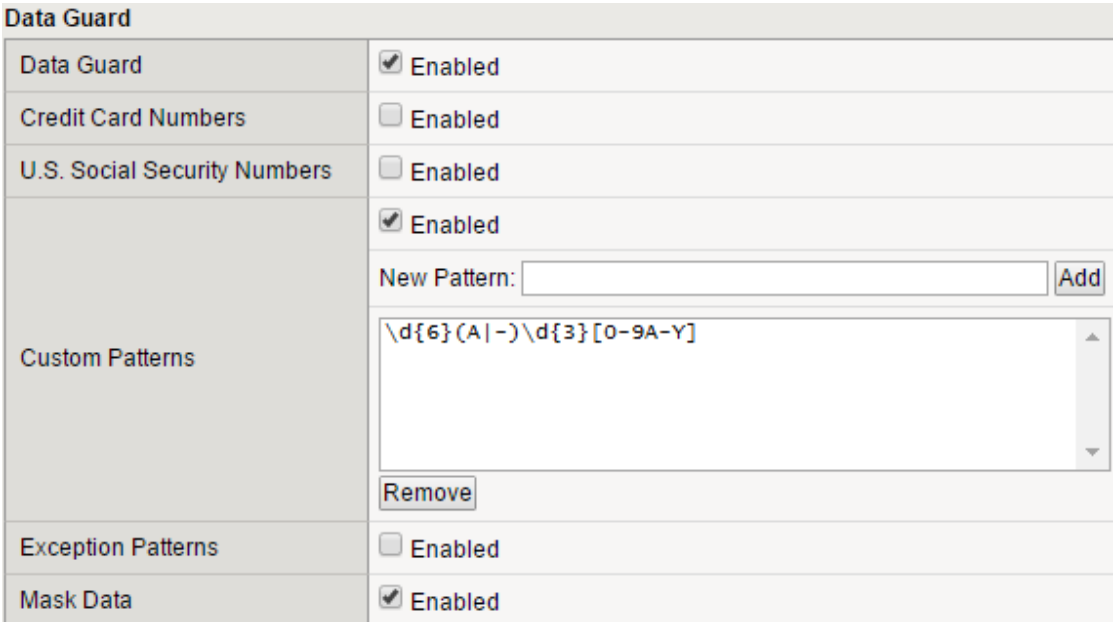
Name	Onload XSS-signature testing
ID	300000002
Signature Type	Request
Signature Scope	Parameter/Cookie, XML, JSON, GWT
Systems	Various systems
Attack Type	Cross Site Scripting (XSS)
Accuracy	Low
Risk	Low
User-defined	Yes
Rule	valuecontent:"onload"; nocase; norm; re2:"/onload\b\W*=/Vsi"; norm;
Revision	4
Last Updated	01/03/2017
Description	Custom attack signature for testing purposes

Kuvio 50. XSS-hyökkäyksen torjuminen kustomoidulla hyökkäystunnisteella

Omien hyökkäystunnisteiden rakentaminen ei ole erityisen suositeltavaa, sillä niillä voi joissain tapauksissa olla suuri riski tuottaa vääriä positiivisia hälytyksiä. *Onload*-hyökkäystunniste löytyy myös ASM:n hyökkäystunnistetietokannasta jo oletuksena, joten tällä tunnisteella ei saavuteta tietoturvapolitiikoihin mitään konkreettisia

hyötyjä. Esimerkin avulla tarkoituksena oli ainoastaan havainnollistaa mahdollisuutta rakentaa ASM:n omia kustomoituja hyökkäystunnisteita. Ainoat tilanteet, joissa kustomoidun hyökkäystunnisteen rakentamisesta voi olla apua, ovat uudet hyökkäykset ja niihin käytettyjen uusien hyökkäysmekanismien estäminen. Haavoittuvuuksien listaamiseen käytettävän CVE:n perusteella voidaan havaita uusi hyökkäys ja siihen käytettävä hyökkäysalgoritmi, joka halutaan estää. Tämän perusteella voidaan rakentaa kustomoitu tunniste. F5 Networks julkaisee kuitenkin järjestelmäänsä hyökkäystunnistepäivityksiä hyvin nopealla syklillä, joten edellä mainittu ratkaisu olisi joka tapauksessa hyvin lyhytaikainen.

ASM:n datansuojaus-toiminteen avulla rakennettiin sääntö, joka sensuroi automaattisesti sovelluksen käyttäjien erehdyksissä syöttämät Suomen henkilötunnukset. Toiminne korvaa kyseiset merkkijonot tähti (*) -merkillä. Ominaisuus voi olla tarpeellinen esimerkiksi erilaisissa chat-keskusteluissa ja keskustelufoorumeissa. Suomen 11-merkkiä pitkiä henkilötunnuksia varten järjestelmään rakennettiin oma kustomoitu säännöllinen lauseke. (Ks. kuvio 51.)



Kuvio 51. Säännöllinen lauseke Suomen henkilötunnusten sensurointiin

Datansuojaus-toiminne asetettiin tietoturvapolitiikassa erikseen tarkkailutilaan. Tällöin säännölliseen lausekkeeseen osuvat merkkijonot ainoastaan peitetään ja tapahtumasta generoidaan lokitieto. Ominaisuuden hyödyntäminen estotilassa ei ole

suositeltavaa, sillä tällöin säännölliseen lausekkeeseen osuva data estää koko sen alaisen sivusto-resurssin. Kuviossa 52 on ASM:n generoima lokitieto datansuojaus-toiminteen toimivuudesta.

Data Guard: Information leakage detected violation details

Detected Pattern	Context	Enforcement URL
Custom Pattern	ords=*****&	[redacted]
Custom Pattern	rong>*****</str	[redacted]

Requested URL	[HTTPS] [redacted]
Security Policy	[redacted]
Support ID	7161496299768928110
Time	2017-01-27 14:37:03
Request Status	
Severity	Error
Response Status Code	200
Attack Types	Information Leakage
Username	N/A
Session ID	405d1bcbe7cf405a
Source IP Address	[redacted] Add IP Address Exception... IP Address Intelligence: N/A [IP Address Intelligence last updated: N/A]
Destination IP Address	[redacted]

Kuvio 52. Lokitieto säännölliseen lausekkeeseen osuvasta HTTP-vastauksesta

ASM:n datansuojaus-ominaisuuden käyttö etenkin omia säännöllisiä lausekkeita käytettäessä tulee olla hyvin testattua. Jos esimerkiksi kuviossa 52 esiintyvään säännölliseen lausekkeeseen lisättäisiin vivun ”(A|-)”-yhteyteen "?"-merkki, osuisivat lausekkeeseen myös henkilötunnukset ilman väliosaa. Tässä tapauksessa 10-merkkiä pitkän, 9 numeroa ja numeron tai A-Y-kirjaimen sisältämän lausekkeen sisälle osuisivat esimerkiksi Suomessa käytettävät yleiset puhelinnumerot.

Kahdelle luodulle ASM-politiikalle suoritettiin myös testi, jonka perusteella oli tarkoitus selvittää politiikkojen aiheuttamat väärät positiiviset hälytykset suhteessa niihin kohdistettuihin kokonaisliikennemääriin. Testit suoritettiin molempien politiikoiden tarkkailuajanjakson päättymisen jälkeen. Testeissä yhdisteltiin suojattavan web-sovelluksen normaalia käyttöä syöttämällä sovelluksen lomakkeen kenttään erillinen sanakirja, joka koostui syötteistä, joiden uskottiin aiheuttavan mahdollisesti vääriä positiivisia hälytyksiä. Lista koostui muun muassa erikoisista

nimistä, erikoismerkkejä sisältävistä lauseista ja sanoista sekä lauseista, jotka sisälsivät sanoja, joita käytetään tyypillisesti sovellustason hyökkäyksissä. Luodussa sanakirjassa oli yhteensä 250 erilaista sanaa, merkkijonoa tai lausetta, joiden uskottiin voivan aiheuttaa järjestelmässä aiheettoman liikenteen estämisen. Liitteessä 4 on listattuna osa tässä testissä käytetyistä syötteistä. Väärien positiivisten hälytysten testien tuloksia analysoidessa tulee ottaa huomioon testeihin käytetty yksinkertainen web-sovellus, suhteellisen lyhyt seurantajakso sekä automaattisen politiikan yhteydessä käytetty oppimismetodi. Taulukossa 13 on esitelty testien tuloksia.

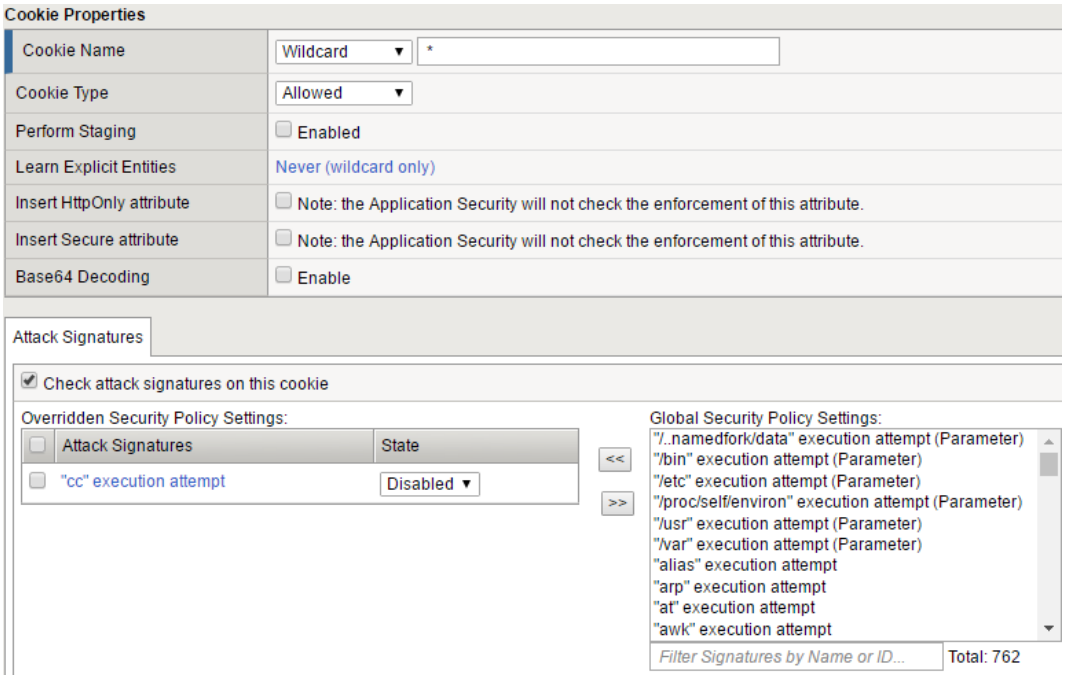
Taulukko 13. Väärät positiiviset hälytykset suhteessa liikennemäärään

POLICY TYPE	TOTAL REQUESTS	FALSE POSITIVES	FALSE POSITIVE RATE
Manual Policy (Rapid Deployment)	10000	1	0.01%
Automatic Policy (Fundamental)	10000	1	0.01%

Molempiin politiikkatyyppeihin kohdistettiin normaalia selainliikennettä noin 10 000 HTTP-kyselyn verran. Näistä kyselyistä vääriä positiivisia hälytyksiä havaittiin molemmissa politiikkatyypeissä ainoastaan yksi. Prosenttiosuudella mitattuna väärien positiivisten hälytysten määrä suhteessa generoituun liikennemäärään oli siis ainoastaan 0,01 prosenttia. Molemmissa politiikkatyypeissä väärän positiivisen hälytyksen aiheutti evästeessä havaittu hyökkäystunniste, jonka avulla kohdejärjestelmässä on mahdollista suorittaa mielivaltaisia komentoja (”cc” execution attempt).

```
TS01e8e9d6=015ac5beb%050dj50:04+2%dk50%3F%3Bcc%3A53662a43
%3Bcc = ;cc
```

Kun evästeessä havaittulle merkistölle: *”%3Bcc*” suoritetaan ASM:n toimesta normalisointi, se muuntuu arvoon ”*;cc*”, joka tulkitaan ASM:n toimesta hyökkäytunnisteeksi. Väärä positiivinen hälytys kitkettiin molemmista politiikkatyypeistä tekemällä evästeet-elementtiryhmän alaiseen villiin korttiin kyseisen hyökkäystunnisteen yliajo. (Ks. kuvio 53.)



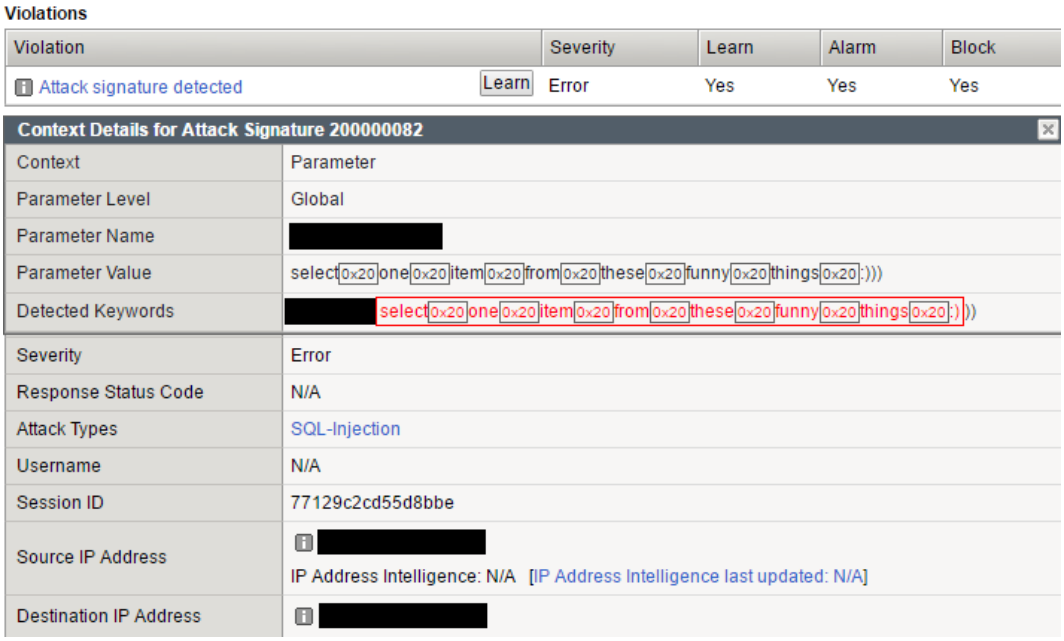
Kuvio 53. Evästeessä havaitun hyökkäystunnisteen ohitus

Väärien positiivisten hälytysten toisessa testausvaiheessa käytettiin ennalta rakennettua sanakirjaa, johon listattiin tyypillisiä (mahdollisia) vääriä positiivisia hälytyksiä aiheuttavia merkkejä, sanoja ja lauseita. Taulukossa 14 on kuvattu tuloksia näiden testien osalta.

Taulukko 14. Väärät positiiviset hälytykset sanakirjan avulla.

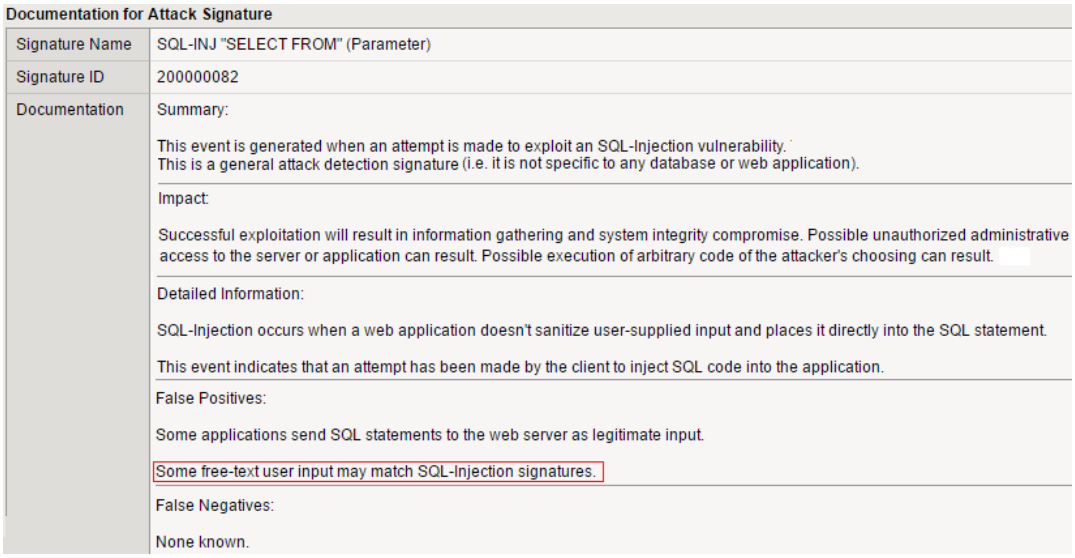
POLICY TYPE	DICTIONARY (words)	FALSE POSITIVES	FALSE POSITIVE RATE
Manual Policy (Rapid Deployment)	250	3	1,20 %
Automatic Policy (Fundamental)	250	3	1,20 %

Molemmissa aiemmin luoduissa politiikkatyypeissä havaittiin kolmea väärää positiivista hälytystä. Kuviossa 54 on kuvattu yhtä näistä molemmissa politiikkatyypeissä esiintynyttä väärää positiivista hälytystä.



Kuvio 54. Väärä positiivinen hälytys web-sovelluksen parametrin arvossa

Kyseessä oleva väärä positiivinen hälytys aiheutui englanninkielisestä käyttäjäsyötteestä: ”*Select one item from these funny things :))”.* ASM tulkitsi syötteen SQL-injektioksi sen sisältämien SQL-komentosyötteiden sanojen ”*select*” ja ”*from*” perusteella. Tarkempaa analysointia tehtäessä huomattiin, että ASM ei estä syötettä ainoastaan näiden kahden sanan perusteella, vaan estotapahtumaan vaaditaan myös tietty määrä välejä (koodaattuna kuviossa 0x20 muotoon) ja lisäksi muita sanoja. Näiden tehtyjen havaintojen perusteella voidaankin suoraan todeta, että kyseisen väärän positiivisen hälytyksen esiintyminen tietoturvapolitiikassa on harvinaista. Kuviossa 55 on kyseisen väärän positiivisen hälytyksen aiheuttanut hyökkäystunniste.



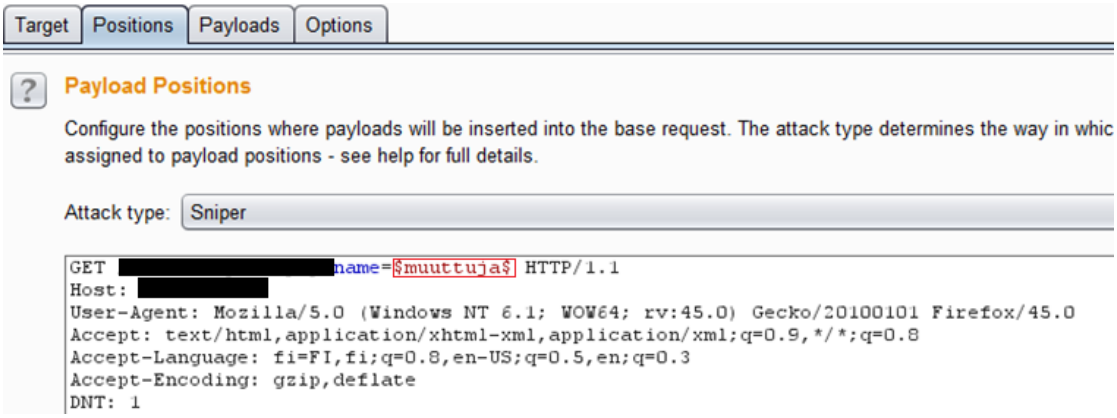
Documentation for Attack Signature	
Signature Name	SQL-INJ "SELECT FROM" (Parameter)
Signature ID	200000082
Documentation	Summary: This event is generated when an attempt is made to exploit an SQL-Injection vulnerability. This is a general attack detection signature (i.e. it is not specific to any database or web application). Impact: Successful exploitation will result in information gathering and system integrity compromise. Possible unauthorized administrative access to the server or application can result. Possible execution of arbitrary code of the attacker's choosing can result. Detailed Information: SQL-Injection occurs when a web application doesn't sanitize user-supplied input and places it directly into the SQL statement. This event indicates that an attempt has been made by the client to inject SQL code into the application. False Positives: Some applications send SQL statements to the web server as legitimate input. Some free-text user input may match SQL-Injection signatures. False Negatives: None known.

Kuvio 55. Hyökkäystunniste, josta aiheutui väärä positiivinen hälytys

Hyökkäystunnisteen yksilöllisissä tiedoissa mainitaan, että jotkin lailliset normaalit käyttäjäsyötteet voivat täsmätä tähän hyökkäystunnisteeseen. Suomenkielisissä web-sovelluksissa kyseistä hälytystä voidaan pitää suhteellisen harmittomana, mutta esimerkiksi englanninkielisellä keskustelufoorumilla edellä mainittu väärä positiivinen hälytys voi joissain tapauksissa aiheuttaa ongelmia.

7.4 Tietoturvapolitiikka osana kerroksellista suojausta

Kerroksellisuustestien tarkoituksena on tutkia eri skenaarioiden avulla, kuinka monta lähetettyä haitallista tunnistetta mikäkin suojaustaso saa kiinni. Käytettävät suojaustasot ovat seuraavan sukupolven palomuuri ja siihen integroitu IPS-järjestelmä, ASM:n nopean käyttöönoton tietoturvapolitiikka ja ASM:n automaattisesti oppimalla rakennettu politiikka. Jokaiseen näistä kolmesta suojaustasosta kohdistetaan erillisten skriptien avulla useita satoja haitallisia tunnisteita. Testit suoritettiin käyttämällä BurpSuiten *intruder*-työkalua. XSS-tunnisteet sijoitettiin tässä testausvaiheessa sovelluksen *name*-parametrin muuttujaan. (Ks. kuvio 56.)



Kuvio 56. Testauksessa käytettävä *name*-parametrin muuttuja

Kuviossa 57 on todennettu BurpSuiten avulla web-sovellukseen kohdistettujen XSS-tunnisteiden kuormaa.

Kuvio 57. BurpSuitella sovellukseen kohdistettujen XSS-tunnisteiden kuorma

Syötettäviä tunnisteita on yhteensä 332 kappaletta. Syötteiden valinnassa pyrittiin mahdollisimman suureen monipuolisuuteen ja kattavuuteen.

Ennen kyseistä testiä ASM-moduuli määritetään lokittamaan kaikki siihen kohdistuva liikenne. Tällöin lokitietoja seuraamalla saadaan helposti selville myös järjestelmän suojauksen mahdollisesti läpäisseet yksittäiset tunnisteet. Seuraavan sukupolven palomuuri ja siihen integroitu IPS-järjestelmä määritellään estämään sen oman luokittelun perusteella korkean ja kriittiset tason uhkat ja vastaavasti ainoastaan lokittamaan pienen ja keskitason vakavuuden uhkat. ASM sen sijaan on määritelty estämään kaikki sen luokittelemat haitalliset tapahtumat. Taulukossa 15 on listattuna kerroksellisuustestien tuloksia XSS-tunnisteiden osalta kokonaisuudessaan.

Taulukko 15. Suojaustasojen havainnointi- ja estokyky XSS-tunnisteilta

PROTECTION LAYER	SPOTTED ILLEGAL REQUESTS	BLOCKED ILLEGAL REQUESTS
Next Generation Firewall + IPS	180/332	51/332
Manual Policy (Rapid Deployment)	331/332	331/332
Automatic Policy (Fundamental)	331/332	331/332

332 lähetetystä XSS-tunnisteesta palomuuri ja siihen integroitu IPS-järjestelmä havaitsivat liikennevirrasta 180 erilaista XSS-tunnistetta ja estivät niistä 51. Sen sijaan ASM:n molempien tietoturvapolitiikkojen tehokkuus kävi ilmi niiden päästäessä ainoastaan yhden tunnisteen läpi. Kyseinen tunniste oli *”<HTML><BODY>”*, jota voidaan kuitenkin pitää lähes harmittomana. Sama testaus suoritettiin myös käyttämällä muuttujan kuormana erilaisia injektiotunnisteita. Tässä testissä mukana oli 312 erilaista haitallista injektiotunnistetta. Taulukossa 16 on listattu tämän testausvaiheen tuloksia.

Taulukko 16. Suojaustasojen havainnointi- ja estokyky injektio-tunnisteilta

PROTECTION LAYER	SPOTTED ILLEGAL REQUESTS	BLOCKED ILLEGAL REQUESTS
Next Generation Firewall + IPS	147/312	0/332
Manual Policy (Rapid Deployment)	312/312	312/312
Automatic Policy (Fundamental)	312/312	312/312

Palomuuri ja siihen integroitu IPS-järjestelmä havaitsivat liikennevirrasta 147 erilaista haitallista tunnistetta, mutta ne eivät estäneet niistä ainuttakaan. Kaikki niiden havaitsemat injektiotunnisteet luokiteltiin joko pieneksi tai keskitason uhkaksi. Vastaavasti 165 tunnistetta läpäisi tämän suojaustason ilman mitään sen tekemää havaintoa. ASM:n tietoturvapolitiikat sen sijaan osoittivat jälleen vahvuutensa.

Molemmat politiikkatyypit havaitsivat ja estivät liikennevirrasta kaikki niihin kohdistetut 312 injektiotunnistetta.

Yksi suurista tietoturvan ongelmakohdista liittyy SSL/TLS-salattuun liikenteeseen. Useat web-sovellukset käyttävät asiakkaan ja palvelimen välillä yhteysprotokollana salattua HTTPS-protokollaa, jolloin SSL/TLS-salaus puretaan vasta palvelinpäässä. Tällöin mikään liikenteen välissä toimiva palomuuriratkaisu ei kykene havaitsemaan salatusta liikenteestä haitallisia tunnisteita ilman palomuurilaitteessa tapahtuvaa salauksen purkua tai muussa verkon pisteessä puretun salaamattoman liikenteen ohjaamista uudelleen palomuurin läpi. Luonnollisesti myöskään ASM ei pysty havaitsemaan haitallisia tunnisteita HTTPS-salatusta liikenteestä. Tyypillisesti F5:n BIG-IP ADC -järjestelmää käytetään kuitenkin verkon keskitettynä SSL-salauksen purkusolmuna. Kyseiseen laitteeseen integroituna ASM nostaa entisestään sen arvoa sovellustason uhkien torjunnassa, kun asiakkaan ja palvelimen väliseen liikennöintiin käytetään salattua HTTPS-protokollaa. Taulukossa 17 on havainnollistettu suojaustasojen havainnointikykyä, kun SSL/TLS-salaus puretaan F5-laitteella.

Taulukko 17. Suojaus HTTPS-liikenteelle, kun SSL-purku tapahtuu F5-laitteella

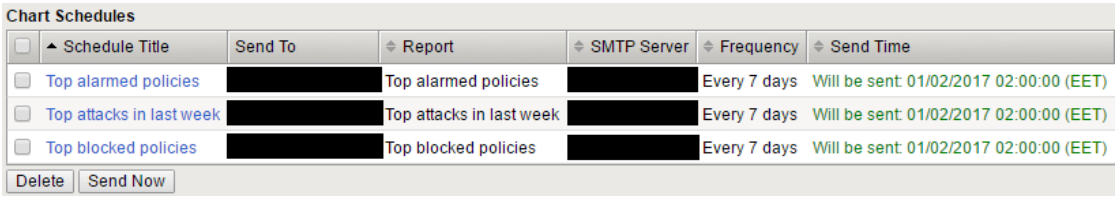
PROTECTION LAYER	SPOTTED ILLEGAL REQUESTS	BLOCKED ILLEGAL REQUESTS
Next Generation Firewall + IPS	0/664	0/664
Manual Policy (Rapid Deployment)	663/664	663/664
Automatic Policy (Fundamental)	663/664	663/664

Tällöin palomuuri ei loogisesti pysty havaitsemaan mitään haitallista liikennettä liikenteen ollessa HTTPS-salattua. Mikäli liikenne halutaan viedä salattuna aina kohdepalvelimille asti, voidaan salattu liikenne purkaa ja tämän jälkeen salata se uudelleen F5-laitteella.

7.5 Tietoturvapolitiikkojen keskitetty raportointi

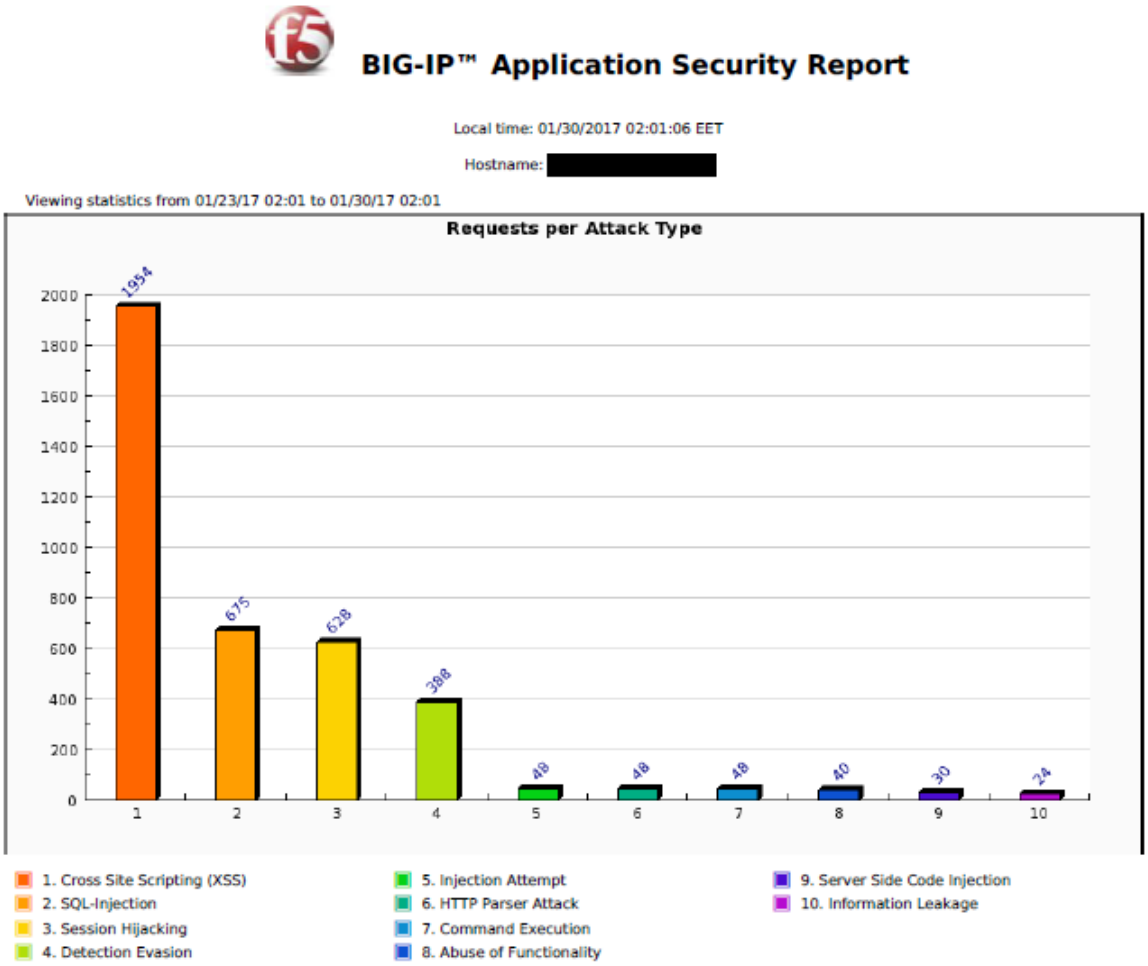
Tietoturvapolitiikkojen seurannan parantamiseksi otettiin käyttöön tietoturvapolitiikkojen moninainen keskitetty raportointi. Raportoinnin avulla voidaan muodostaa muun muassa nopea kokonaiskuva järjestelmän eri tietoturvapolitiikkoihin kohdistuneista hyökkäyksistä ja niiden määristä. Raportoinnin pohjalta voidaan myös helposti seurata hälytyksen aiheuttaneita lähde-IP-osoitteita ja tehdä tämän pohjalta tarkempaa analysointia.

ASM:ssä on luotuna valmiiksi useita raportointipohjia, joiden tulokset voidaan generoida manuaalisesti järjestelmästä käsin. Raportoinnin analysoinnin helpottamiseksi se voidaan kuitenkin osittain automatisoida. Tämä tapahtuu lähettämällä automaattisesti generoidut raportit järjestelmän ylläpitäjän sähköpostiin sähköpostipalvelimen ja SMTP-protokollan välityksellä. Kuviossa 58 on todennus raportoinnin konfiguroinnista, jossa sähköpostiin lähetetään kolme eri järjestelmän tuottamaa raporttia.



Kuvio 58. Ajastettu raportointi järjestelmän ylläpitäjän sähköpostiin

Raporteina käytettiin valmiita raportointipohjia *Top alarmed policies*, *Top attacks in last week* ja *Top blocked policies*. Raportit määriteltiin lähetettäviksi ajastetusti viikon välein. Kuviossa 59 on osatodennus ASM-järjestelmän generoimasta ylläpitäjän sähköpostiin lähetetystä tietoturvapolitiikoiden koostetusta raportista.



Kuvio 59. Ote tietoturvapolitiikoiden keskitetystä raportointigraafista

Kyseessä olevan yksittäisen raportin avulla pystytään saamaan nopea ja pintapuolinen katsaus kaikista järjestelmän estämistä hyökkäystyypeistä ja niiden kokonaismääristä. Hyökkäykset on lisäksi luokiteltu erillisiin alakategorioihin, mikä helpottaa eri uhkien tunnistamista.

Mikäli koostettujen raporttien perusteella havaitaan jotakin poikkeuksellista toimintaa, kuten erittäin suuria liikenteen estomääriä useista eri lähdeosoitteista, voidaan olettaa hälytyksen olevan mahdollisesti väärä positiivinen hälytys. Raportin perusteella voidaan tällöin tehdä manuaalista, tarkempaa analysointia ASM:n käyttöliittymästä käsin. Vastaavasti raporttien perusteella voidaan myös havaita tyypillisiä ”koputteluyrityksiä”, joissa tietystä lähde-IP-osoitteesta on havaittu paljon estettyä liikennettä.

8 Tietoturvapolitiikoiden vertailu ja johtopäätökset

8.1 Tulosten vertailu

Toteutusvaiheessa rakennettujen ASM:n tietoturvapolitiikkojen vertailu on kattavan toteutusvaiheen perusteella suhteellisen helppoa. Automaattisen ja manuaalisen tietoturvapolitiikan konfigurointitavat eivät juurikaan eroa toisistaan ja niiden molempien luominen onkin hyvin suoraviivainen prosessi. Automaattisen tietoturvapolitiikan rakentaminen ja sen saaminen toimintakuntoon on manuaalista tapaa hitaampi. Toteutusvaiheessa käytettyjen luotettujen IP-osoitteiden ja yksinkertaisimman *Fundamental*-oppimistason ansioista oppimisprosessit olivat kuitenkin tehokkaita ja politiikka saatiin toimintakuntoon käytännössä heti viikon tarkkailuajanjakson jälkeen. Tarkkailuajanjakson jälkeen ei myöskään esiintynyt lainkaan sellaisia web-sovelluksen käyttämiä rakenteita, joita järjestelmä ei olisi opetteluvaiheen aikana jo oppinut. Tämä oli yllättävää, sillä tarkkailuajanjakson aikana luotetusta IP-osoitteista generoitu liikenne testiympäristössä oli kuitenkin varsin vähäistä. Oppimisvaiheen nopeus ja tehokkuus ovat kuitenkin riippuvaisia muun muassa sovelluksen monimutkaisuudesta, siihen kohdistuvasta liikenteen määrästä ja käytetystä oppimistyypistä. Toteutusvaiheessa keskityttiin oppimistasoista lähinnä vain *Fundamental*-oppimistasoon. Kaikista tarkinta ja yksityiskohtaisinta tietoturvapolitiikkaa rakentavaa *Comprehensive* -oppimistasoa käytettäessä on suuri mahdollisuus, että joitakin yksittäisiä web-sovelluksen käyttämiä rakenteita jää oppimisprosessin ulkopuolelle. Tällöin seurauksena on politiikan estotilaan siirtymisen jälkeen mahdollisesti suuri määrä vääriä positiivisia hälytyksiä. Toteutusvaiheen loppupuolella testattiin myös lyhyesti edellä mainittua *Comprehensive*-oppimistasoa. Tämän yhteydessä havaittiin, että paljon vääriä positiivisia hälytyksiä aiheutti politiikan *Illegal Meta Character in Value* -tarkastuskenttä. Testien yhteydessä havaittiin, että *Comprehensive*-oppimistasoa käytettäessä tämä tarkastuskenttä voidaan jättää politiikasta pois. Kyseisen tarkastustyypin poistaminen politiikasta voi kuitenkin aiheuttaa yleisen suojaustehon alenemisen, sillä jotkin merkit kuten "<" ja " ’ " voivat tietyissä tapauksissa olla riskejä web-sovelluksen tietoturvalle. Tähän tuo kuitenkin apua ASM:n

hyökkäystunnistetietokanta, joka testien perusteella esti lähes kaikki näitä merkkejä hyödyntävät erilaiset hyökkäyslausekkeet.

Automaattinen politiikanrakentaja suoriutui testeissä kohtalaisen hyvin, mutta sen toiminnassa huomattiin myös pieniä ongelmia. Tämä ilmeni alussa esimerkiksi järjestelmän oppiessa web-sovelluksen parametrien vääriä datatyyppejä. Automaattinen politiikanrakentaja toimii hyvin tilanteissa, joissa järjestelmän ylläpitäjällä ei ole kattavaa tietämystä ASM-järjestelmän ylläpidosta, eikä vastaavasti myöskään itse suojattavasta web-sovelluksesta. Oppimisehdotusten hyväksyminen politiikkaan manuaalisesti on aina varmin ja paras tapa toteuttaa ASM:n tietoturvapolitiikkoja. Tällöin tiedetään varmasti, mitä uusia elementtejä politiikkaan ollaan lisäämässä ja vastaavasti, mitä hienosäätöä politiikan elementtiryhmien elementteihin ollaan tekemässä.

Manuaalisen tietoturvapolitiikan, eli toteutusvaiheessa käytetyn nopean käyttöönoton tietoturvapolitiikan nimestä huolimatta sen rakentaminen on myös aikaa vievä prosessi. Tässä politiikkatyypissä tarkkailuajanjakson tehtävänä on lähinnä varmistaa se, ettei sen yhteydessä käytetty hyökkäystunnistetietokanta aiheuta käyttäjille vääriä positiivisia hälytyksiä. Tarkkailuajanjakso voidaan kuitenkin jättää tätä politiikkatyyppiä käytettäessä pois, mutta tällöin järjestelmän ylläpitäjä ottaa tiedostetun riskin hyökkäystunnisteista aiheutuvien mahdollisten väärien positiivisten hälytysten vuoksi. Vaikka nopean käyttöönoton tietoturvapolitiikan nimi ei varsinaisesti kuvaakkaan sen todellista luonnetta, on kyseinen politiikkatyypin käyttöönotto silti automaattista tietoturvapolitiikkaa nopeampi.

Jo suojauskykytestien aikaisessa vaiheessa kävi selväksi, että ASM:n sisäinen hyökkäystunnistetietokanta oli varsin tehokas eri hyökkäyslausekkeita torjuttaessa. Toteutusvaiheessa suoritettujen testien perusteella automaattisen politiikan tarjoamat hyödyt verrattuna manuaaliseen politiikkaan voidaankin jopa hieman kyseenalaistaa. Toisaalta automaattisen politiikan suurimpiin hyötyihin kuuluvat sen tarjoama suoja uusia nollapäivähaavoittuvuuksia ja kustomoituja hyökkäyslausekkeita vastaan. Näitä hyötyjä ei kuitenkaan saatu täysin tuotua esille toteutusvaiheen aikana. Perusteltuja automaattisen tietoturvapolitiikan hyötyjä ovat kuitenkin esimerkiksi sen HTTP-kyselyille suorittamat pituustarkastukset. Mikäli näitä pituuksia ei olla määritelty tietoturvapolitiikassa eikä ohjelmistotasolla,

mahdollistetaan hyökkääjälle web-sovelluksen lomakkeissa ylipitkien POST-pyyntöjen lähettäminen. Ylipitkien viestien avulla voidaan esimerkiksi tehdä monimutkaisia hyökkäyksiä, joilla voidaan mahdollisesti kiertää järjestelmän tunnistetietokanta. Lisäksi ylipitkien HTTP-kyselyiden lähettäminen voi viedä web-sovelluksen käyttämältä alustalta tärkeitä resursseja ja aiheuttaa sovellukselle pahimmassa tapauksessa osittaisen palvelun estymisen tilan (DoS). Tämän perusteella voidaankin ajatella niin sanotun hybridipolitiikan käyttöä. Tässä mallissa käytössä on manuaalinen tietoturvapolitiikka, jonka lisäksi politiikan alaisiin elementtiryhmiin määritellään muutamia web-sovelluksen käyttämiä kriittisimpiä yksittäisiä elementtejä. Näille yksittäisille elementeille voidaan tämän jälkeen määritellä omia tarkkoja attribuutteja, kuten minimi- ja maksimipituuksia.

Toteutusvaiheen väärien positiivisten hälytysten testin tulos yllätti positiivisesti. Kun molempiin politiikkatyyppeihin kohdistettiin normaalia web-liikennettä 10 000 kyselyn verran, oli tuloksena molempien politiikkatyyppien osalta vain yksi väärä positiivinen hälytys. Etukäteen luodun sanakirjan avulla kohdistettujen syötteiden tuloksena tuli esille vain lähinnä muutama mahdollinen ongelmakohta, jotka voidaan kuitenkin helposti karsia poistamalla kyseiset ASM:n suorittamat tarkastukset politiikasta kokonaan.

Kerroksellisuustestien tulokset olivat suurilta osin odotusten mukaiset. Seuraavan sukupolven palomuuri ja siihen integroitu IPS-järjestelmä havaitsivat molemmista käytetyistä tunnistekuormista noin puolet. Toisaalta tulos on palomuurille ja IPS-järjestelmälle hyvä, mutta toisaalta se tuo esille palomuurien ja IPS-järjestelmien rajatut mahdollisuudet torjua sovellustason uhkia. Kerroksellisuustesteissä kahden ASM-politiikkatyypin välillä ei havaittu käytännössä lainkaan eroavaisuuksia. Tämä taas johtuu siitä, että kaikki syötetyt tunnisteet estettiin ASM:n hyökkäystunnistetietokannan perusteella, joka on käytössä molemmissa politiikkatyypeissä.

8.2 Johtopäätökset

Toteutusvaiheen tuloksien pohjalta saadut johtopäätökset vastaavat hyvin opinnäytetyön alkuperäistä tutkimusongelmaa. Tutkimusongelmana oli selvittää

minkälaisen lisäarvon web-sovellustason palomuuri tuo organisaatiolle jo olemassa olevien suojausjärjestelmien tueksi. Web-sovellustason palomuurin efektiivisyys kävi hyvin ilmi monipuolisten testien perusteella. Järjestelmän tehokkuus todistettiin sekä yksinkertaisia ja monimutkaisia hyökkäyslausekkeita sekä erilaisia kiertoyrityksiä käyttäen. Lisäksi testien avulla saatiin konkreettisten esimerkkien kautta selville seuraavan sukupolven palomuurin ja IPS-järjestelmän ongelmakohdat web-sovellustason uhkien torjunnassa.

Väärien positiivisten hälytysten testien tuloksiin on suhtauduttava hyvin kriittisesti. Testeissä oli käytössä ainoastaan yksi web-sovellus ja siihen kohdistettujen kyselyiden kokonaismäärä oli varsin pieni (10 000). Myös oman sanakirjan pohjalta suoritettujen testien perusteella on hankalaa lähteä arvioimaan todellista väärien positiivisten hälytysten määrää. NSS Labsin suorittamien testien perusteella ASM:n tietoturvapolitiikan keskimääräinen väärien positiivisten hälytysten esiintymismäärä oli 0,124 prosenttia (Web Application Firewall Product Analysis N.d). Raportista ei kuitenkaan käy ilmi, miten ja minkälaisilla testausmenetelmillä kyseiseen tulokseen ollaan päädytty. Lisäksi testien suoritusajankohtaa ei ole saatavilla. Näiden tietojen valossa NSS Labsin testien tuloksena saatuun prosenttiosuuteen täytyy suhtautua varauksella.

Toteutusvaiheen kerroksellisuustestien perusteella voidaan helposti vetää johtopäätös web-sovellustason palomuurin merkityksellisyydestä ja tarpeellisuudesta organisaation web-palvelujen suojaamisessa. Merkityksellisyyttä tulee ennestään kasvattamaan jatkuvasti lisääntyvä web-sovelluksia kohtaan suoritettava epäilyttävä haitallinen toiminta. Web-sovellustason palomuurin käyttöönotto organisaatiossa vaatii kuitenkin selkeät implementointi-, ylläpito- ja toimintaprosessit, jotta järjestelmän käyttöönotto olisi mahdollisimman mutkatonta ja sujuvaa. Lisäksi on määriteltävä toimintatavat väärien positiivisten hälytysten esiintymisen varalle. Opinnäytetyön kappaleessa 8.3 on käsiteltynä tarkemmin toteutusvaiheen tulosten ja niistä saatujen johtopäätösten pohjalta kehitettyjen käyttöönottoprosessien toimintamalleja.

8.3 Tietoturvapolitiikkojen rakentaminen erilaisissa tilanteissa

8.3.1 Automaattinen tietoturvapolitiikka kehitysympäristössä

Tietoturvapolitiikan rakentaminen kehitysympäristössä on kaikista toteutustavoista varmin ja tehokkain. Kehitysympäristössä tietoturvapolitiikkaa voidaan hioa yleensä määrittelemättömän ajanjakson ajan, jolloin sen yhteydessä tapahtuvat väärät positiiviset hälytykset saadaan karsittua minimiin. Kehitysympäristössä merkittävään rooliin nousevat myös ASM-politiikan yhteydessä määriteltävät luotetut IP-osoitteet. Verkko, josta web-sovellusta ja samalla ASM-politiikkaa testataan, voidaan määrittää luotetuksi, jolloin kaikki web-sovelluksen rakenteelliset elementit opitaan niiden ensiesiintymisellä. Tämä nopeuttaa politiikan rakentamista huomattavasti normaalista raja-arvoihin perustuvasta mallista. Liitteessä 1 on kuvattu automaattisen tietoturvapolitiikan rakentamisen prosessia kehitysympäristössä ja sen tuomista varsinaiseen tuotantoympäristöön.

8.3.2 Automaattinen tietoturvapolitiikka tuontatoympäristössä

Tietoturvapolitiikan rakentaminen suoraan tuotantoympäristöön on vaihtoehtoinen tapa rakentaa tietoturvapolitiikka. Politiikan rakentamisessa suoraan tuotantoympäristöön ongelmaksi nousevat automaattisessa politiikassa liikenteen oppiminen ja mahdolliset väärät positiiviset hälytykset. Web-sovelluksen ja sen tarjoaman palvelun ollessa tuotantokäytössä on automaattisen politiikan yleensä opittava liikennettä osittain epäluotetuista lähteistä. Tällöin ei voida olettaa, että käyttäjien syöttämät tiedot tulisi oppia automaattisesti politiikkaan. Mikäli tietoturvapolitiikka rakennetaankin suoraan tuotantoympäristöön, on yleensä suotavaa, että oppimisehdotukset hyväksytään tällöin politiikkaan manuaalisesti järjestelmän ylläpitäjän toimesta. Tällöin politiikkaan ei opita ja lisätä automaattisen politiikanrakentajan toimesta mitään automaattiesti.

Tuotantoympäristössä politiikan oppimisessa voidaan myös hyödyntää ASM:n politiikalle asetettavia raja-arvoja, joiden avulla liikennettä opitaan vasta silloin, kun ennalta määritellyt rajat täyttyvät. Rajojen asettaminen on yksilöllinen prosessi ja riippuvainen monesta yksittäisestä tekijästä, kuten web-sovelluksen rakenteen monimutkaisuudesta ja käyttäjien määrästä. Nyrkkisääntönä voidaan kuitenkin pitää

sitä, että mitä korkeammaksi rajat asetetaan, sitä varmemmin ASM ei opi asioita, joita sen ei tulisi oppia. Oppimisraja-arvojen korkeat raja-arvot ovat osittain verrannollisia politiikan rakentamiseen kuluvaan aikaan.

Tuotantoympäristössä tietoturvapolitiikkaa on myös luonnollisesti ajettava aluksi välttämättömästi tarkkailutilassa, jossa palvelun normaali käyttö ei esty missään vaiheessa. Tarkkailutilaa ajetaan niin pitkään, kunnes ASM on oppinut suojattavan web-sovelluksen käyttämät rakenteet täydellisesti, eikä se enää aiheuta vääriä positiivisia hälytyksiä. Liitteessä 2 on kuvattu automaattisen tietoturvapolitiikan rakentamisen prosessikaaviota tuotantoympäristössä.

8.3.3 Manuaalinen tietoturvapolitiikka kehitysympäristössä

Manuaalisen tietoturvapolitiikan rakentaminen kehitysympäristössä ei juurikaan eroa automaattisesta toteutustavasta, mutta on silti edellä mainittua hieman yksinkertaisempi. Kaikissa manuaalisissa toteutustavoissa jätetään pois web-sovelluksen oppimisprosessi. Tällöin politiikan valmisteluvaihe sisältää lähinnä mahdollisten väärien positiivisten hälytysten karsimisen. Liitteessä 3 on kuvattu manuaalisen tietoturvapolitiikan rakentamisen prosessia kehitysympäristössä ja sen tuomista varsinaiseen tuotantoympäristöön.

9 Yhteenveto

9.1 Pohdinta

Opinnäytetyön tavoitteena oli tutkia tyypillisiä sovellustason uhkia ja web-sovellustason palomuurin käyttöä niiden torjumiseen. Työssä haettiin perusteluja sille, mitä hyötyjä web-sovellustason palomuurilla saavutetaan perinteisiin suojausmenetelmiin verrattuna. Yhtenä opinnäytetyön pääkohtana oli kaupallisen F5 BIG-IP ASM -sovellustason palomuurituotteen ominaisuuksiin ja sen tarjoamiin suojauskykyominaisuuksiin perehtyminen.

Testien yhteydessä ei opinnäytetyön perimmäisen aiheen ja laajuuden vuoksi keskitytty lainkaan sovelluspuolen omiin suojausmekanismeihin. Lähes kaikki testeissä käytetyt yksinkertaiset tunnisteet ovat torjuttavissa ohjelmistotasolla

hyvien tietoturvallisten käytänteiden mukaisilla ohjelmiston toteutustavoilla. Testien yhteydessä tarkoituksena oli ainoastaan tutkia ASM:n erilaisten tietoturvapolitiikoiden kyvykkyyttä torjua näitä yksinkertaisia sovellustasonhyökkäyksiä.

Web-sovellustason palomuurin tehokas ylläpitäminen vaatii ohjelmistopuolen osaamista. Osaamista vaaditaan etenkin politiikan rakennusvaiheessa ja mahdollisten väärien positiivisten hälytysten analysoinnissa. ASM:n yhteydessä järjestelmään kattavasti perehtymättömän henkilön puutetta on pyritty paikkaamaan automaattisella politiikanrakentajalla, jolloin järjestelmän ylläpitäminen on mahdollista myös vähäisellä ohjelmistokokemuksella. Todellisuudessa automaattinen politiikanrakentaja ei kuitenkaan paikkaa asiaa kokonaan, vaan web-sovellustason palomuurin tehokas ylläpitäminen vaatii edelleen aina jonkinasteista suojattavan web-sovelluksen tuntemusta.

Väärien negatiivisten ja positiivisten hälytysten todellisia määriä ei voida arvioida luotettavasti yhden web-sovelluksen ja siihen kohdistettujen testien perusteella. Laajempi kuva saavutetaan vasta, kun järjestelmä on käytössä useissa erilaisissa sovelluksissa ja liikenteen määrä ja laatu vastaavat sovellukseen normaalitilanteessa odotettua liikennettä.

Web-sovellustason palomuuria ei koskaan voida pitää täydellisenä suojausmekanismina, vaan se tulee nähdä pikemminkin lisäturvan tuojana. Parhaiten kyseinen palomuurituote toimii osana kerroksellista suojauskokonaisuutta. Virtuaalinen paikkaus on ehdottomasti yksi WAF-tuotteiden parhaista ominaisuuksista ja sitä tulisikin käyttää rohkeasti laajemmin väliaikaisena ratkaisuna erilaisten tietoturvauhkien torjunnassa.

9.2 Jatkokehitys

Opinnäytetyössä keskityttiin positiivista tietoturvamallia hyödyntävistä ASM:n politiikkatyypeistä yksityiskohtaisesti ainoastaan *Fundamental*-politiikkatyyppiin. Myös *Comprehensive*-politiikkatyyppiä testattiin lyhyen aikaa, mutta testaus tämän politiikkatyypin osalta jäi hyvin pintapuoliseksi. Jatkokehitystä ajatellen olisi mielenkiintoista vertailla kattavasti jokaista kolmea automaattisen politiikan

politiikkatyyppiä keskenään ja tehdä vertailun pohjalta johtopäätöksiä niiden eduista ja haitoista. Rajoitetusta aikataulusta johtuen testien ulkopuolelle jätettiin myös tietoturvapolitiikan rakentaminen kolmannen osapuolen haavoittuvuusskanneria käyttäen. Opinnäytetyön jatkokehityksen kohteeksi sopisikin erinomaisesti tämän politiikkatyypin toimivuuden vertailu suhteessa automaattiseen- ja manuaaliseen politiikkatyyppiin.

Toteutusvaiheessa suoritettuja, väärien positiivisten hälytysten testien laajuutta on tulevaisuudessa pyrittävä monipuolistamaan. Kattava ja monipuolinen testaus on kuitenkin erittäin hankala toteuttaa. Jokainen web-sovellus on yksilöllinen eikä kattavaa kokonaiskuvaa tavoiteta suppeilla testeillä. Luultavasti parhain kokonaiskuva saavutetaan web-sovelluskohtaisesti vasta järjestelmän ollessa tuotantokäytössä ja globaalisti tarkkailutilassa.

Yhtenä mielenkiintoisena lisäyksenä ASM:lle olisi eräänlainen CVE-numeron hakukenttä. Tämän avulla tiettyä CVE-numeroa vasten voitaisiin ASM:n hyökkäystunnistetietokannasta hakea sitä vastaavaa mahdollista hyökkäystunnistetta. Tämän perusteella voitaisiin helposti ja nopeasti selvittää löytyykö tietokannasta jo valmiina tunniste tietylle haavoittuvuudelle.

ASM:stä löytyvät myös lähes kaikki kehittyneen WAF-tuotteen ominaisuudet. Näistä ominaisuuksista kiinnostavia ominaisuuksia ovat etenkin haitallisten bottien torjumiseen käytettävät työkalut sekä asiakkaan laitteesta muodostettava sormenjälki ja sen luokittelu. Opinnäytetyön laajuuden rajaamiseksi näihin ominaisuuksiin ei kuitenkaan tässä opinnäytetyössä paneuduttu juuri lainkaan, mutta jatkokehityksenä näiden ominaisuuksien hyödyntäminen tulee varmasti ajankohtaiseksi.

Opinnäytetyön ulkopuolelle jätettiin myös järjestelmän suorituskykytestit. Ennen järjestelmän laajaa tuotantokäyttöön ottamista onkin erityisen tärkeää tehdä analysointia ASM-moduulin prosessorin- ja muistinkäytöstä, jotta resurssien riittävyys BIG-IP –järjestelmän normaaliin toimintaan saadaan varmistettua. Tämän yhteydessä on tarkoitus toteuttaa myös vertailut viiveissä lähteen ja kohteen välissä ASM-moduulin kanssa ja ilman.

Lähteet

Anders, D. N.d. Introduction to Web Application Firewalls. Viitattu 20.1.2017. https://www.owasp.org/images/d/d1/Intro-WebApplicationFirewalls.pdf

Agility. 2016. Solutions for an application security. F5 Networks. Viitattu 26.11.2016.

Bash-komentotulkin Shellshock-haavoittuvuus mahdollistaa laajan hyväksikäytön. 2014. Viestintävirasto kyberturvallisuus. Viitattu 24.11.2016. https://www.viestintavirasto.fi/kyberturvallisuus/varoitukset/2014/varoitus-2014-02.html

Benji, C. 2015. Question 93 : What's the difference between stored and reflected XSS?. Viitattu 15.12.2016. https://infoseceye.wordpress.com/2015/09/30/question-93-whats-the-difference-between-stored-and-reflected-xss/

Configuration Guide for BIG-IP Application Security Manager. 2013. Viitattu 1.12.2016. https://support.f5.com/content/kb/en-us/products/big-ip_asm/manuals/product/asm-config-11-4-0/_jcr_content/pdfAttach/download/file.res/Configuration_Guide_for_BIG-IP_Application_Security_Manager.pdf

D'Hoinne, J. & Hils, A. & Young, G & Feiman, J. 2014. Magic Quadrant for Web Application Firewall. Gartner. Viitattu 14.12.2016. http://www.ab-sol.net/wp-content/uploads/2014/11/Magic_Quadrant_for_Web_Application_Firewalls.pdf

Dries, L. 2013. F5 Application Security Manager. Viitattu 14.11.2016. https://www.motiv.nl/documenten/presentaties/f5-application-security-manager.pdf

Eames, J. 2016. BIG-IP Application Security Manager Operations Guide. Viitattu 8.10.2016. https://support.f5.com/content/kb/en-us/products/big-ip_asm/manuals/product/f5-asm-operations-guide/_jcr_content/pdfAttach/download/file.res/f5-asm-operations-guide.pdf

Enrique. 2013. Double Encoding – One Of The Biggest Enemies While Fixing Cross-Site Scripting (XSS). Viitattu 27.12.2016. https://www.question-defense.com/2013/01/10/double-encoding-one-of-the-biggest-enemies-while-fixing-cross-site-scripting-xss

Fagerlund, T. 2014. Websocket-protokollan tietoturva selainsovelluksissa. Pro gradu –tutkielma. Tampereen yliopisto. Tietojenkäsittelyopin tiedekunta. Viitattu 15.1.2017. https://tampub.uta.fi/bitstream/handle/10024/95854/gradu07209.pdf?sequence=3

F5 Networks Named a Leader in Gartner Magic Quadrant for Application Delivery Controllers for 10th Consecutive Year. 2016. F5 Networks Inc. Viitattu 9.10.2016. https://f5.com/about-us/news/press-releases/f5-networks-named-a-leader-in-gartner-magic-quadrant-for-application-delivery-controllers-for-10th-consecutive-year

Green, E. 2016. CIT 383: Administrative Scripting. Viitattu 13.11.2016. http://slideplayer.com/slide/9570294/

Halminen, L. 2016. Suomessa käytettävissä verkkolaitteissa vakava haavoittuvuus – tietoturvayhtiö syyttää valmistajia piittaamattomuudesta. Helsingin Sanomat. Verkkolehti. Viitattu 23.12.2016. http://www.hs.fi/talous/art-2000002918954.html

Humphrys, M. N.d. Socket class – Show rat HTTP request and response. Viitattu 19.11.2016. http://www.computing.dcu.ie/~humphrys/Notes/Networks/java.html

Ihantola, P. 2016. Welcome to web software development. Luentomateriaali. Aalto yliopisto. Viitattu 26.11.2016. https://www.cs.tut.fi/~seitti/2016/lectures/01_intro_to_wsd.html#/1

Jokinen, O. 2012. Keskitetty tunnistautumispalvelu web-sovellusarkkitehtuureissa. Pro gradu – tutkielma. Helsingin yliopisto. Tietojenkäsittelytieteen laitos. Viitattu 11.11.2016. https://helda.helsinki.fi/bitstream/handle/10138/37666/pro_gradu_olli_jokinen.pdf?sequence=2

Karvi, T. 2014. Luku VII: Selainten tietoturva. Helsingin yliopisto. Luentomateriaali. Viitattu 27.12.2016. https://www.cs.helsinki.fi/u/karvi/perusteet-luku7_14.pdf

KATAKRI. 2015. Tietoturvallisuuden auditointityökalu viranomaisille. Viitattu 23.12.2016. http://www.defmin.fi/files/3165/Katakri_2015_Tietoturvallisuuden_auditointityokalu_viranomaisille.pdf

K11914: Updating a BIG-IP ASM Security Policy when your website changes. 2016. AskF5. Viitattu 12.12.2016. https://support.f5.com/csp/article/K11914

K11930. Overview of the ASM CSRF protection feature. 2016. AskF5. Viitattu 11.11.2016. https://support.f5.com/csp/article/K11930

K13693: BIG-IP ASM cookies are passed to origin web servers. 2016. AskF5. Viitattu 12.12.2016. https://support.f5.com/csp/article/K13693

Lehtinen, M. 2015. OWASP Top 10 -haavoittuvuuksien korjaaminen TIM-järjestelmästä. Pro gradu -tutkielma. Jyväskylän yliopisto. Tietotekniikan tiedekunta. Viitattu 29.10.2016. https://jyx.jyu.fi/dspace/bitstream/handle/123456789/47871/URN%3ANBN%3Afi%3Ajyu-201511303863.pdf?sequence=1

Lehtonen, J. & Siljander, K. 2010. Web-palveluiden tietoturva. Jyväskylän yliopisto. Luentomateriaali. Viitattu 24.10.2016. https://koppa.jyu.fi/kurssit/96852/luento/web-tietoturva

Masters, G. 2016. HEIST attack breaches HTTPS in the browser. Viitattu 3.1.2017. http://www.itnews.com.au/news/heist-attack-breaches-https-in-the-browser-432667

McHenry, B A. 2016. Taking the Fear out of WAF. Viitattu 27.1.2017. http://www.slideshare.net/bamchenry/taking-the-fear-out-of-waf?qid=e8595d13-68af-4539-8be6-3f2ed7e75cfc&v=&b=&from_search=7

McMillan, J. 2009. What is the Difference Between an IPS and a Web Application Firewall. SANS. Viitattu 23.11.2016. https://www.sans.org/security-

resources/idfaq/what-is-the-difference-between-an-ips-and-a-web-application-firewall/1/25

Nixu Watson. N.d. Viitattu 12.12.2016. https://www.nixu.com/fi/watson

Organisaatio. 2016. Kansaneläkelaitos. Viitattu 16.10.2016. http://www.kela.fi/organisaatio

RFC 2616. 1999. Hypertext Transfer Protocol – HTTP/1.1. Viitattu 18.11.2016. https://tools.ietf.org/html/rfc2616

Saarinen, J. 2014. OWASP Top 10:n suositusten huomioiminen ohjelmistokehityksessä. Pro gradu –tutkielma. Helsingin yliopisto. Tietojenkäsittelytieteen tiedekunta. Viitattu 25.9.2016. https://helda.helsinki.fi/bitstream/handle/10138/136109/gradu_Jussi_Saarinen_011929604.pdf?sequence=2

Saarelma, H. 2014. Verkkosovellusten tietoturvatestaus osana verkkohotellipalvelua. Diplomityö. Tampereen yliopisto. Automaatiotekniikan tiedekunta. Viitattu 23.12.2016. http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/22538/Saarelma.pdf?sequence=1

Sethi, R. 2013. Why You Shouldn't Use the OWASP Top 10 as a List of Software Security Requirements. Viitattu 19.12.2016. http://www.infosecisland.com/blogview/22951-Why-You-Shouldnt-Use-the-OWASP-Top-10-as-a-List-of-Software-Security-Requirements.html

Sims, S. 2016. Brocade vADC Portfolio Overview 2016. Viitattu 24.11.2016. http://www.slideshare.net/ScottSims5/brocade-vadc-portfolio-overview-2016

Solutions and Technology. 2013. F5 Networks. Viitattu 9.10.2016.

SOL6850. 2016. Overview of BIG-IP ASM cookies. F5 Support. Viitattu 30.11.2016. https://support.f5.com/kb/en-us/solutions/public/6000/800/sol6850.html

Stenberg, D. 2015. HTTP2. Viitattu 13.11.2016. https://daniel.haxx.se/http2/http2-v1.12.pdf

Test Methodology v2.0. N.d. NSS Labs. Viitattu 14.12.2016. https://www.nsslabs.com/linkservid/A8536E6E-5056-9046-933F63321654E299/

Tietoturvallisuuden auditointityökalu viranomaisille. 2015. Kansallinen turvallisuusauditointikriteeristö. Viitattu 12.11.2016. http://www.defmin.fi/files/3165/Katakri_2015_Tietoturvallisuuden_auditointityokalu_viranomaisille.pdf

Toiminta. 2016. Kansaneläkelaitos. Viitattu 16.10.2016. http://www.kela.fi/toiminta

VAHTI 1/2013. 2013. Sovelluskehityksen tietoturvaohje. Viitattu 16.10.2016. https://www.vahtiohje.fi/web/guest/vahti-1/2013-sovelluskehityksen-tietoturvaohje

Vijayan, J. 2016. When Encryption Becomes The Enemy's Best Friend. Viitattu 12.1.2017. http://www.darkreading.com/attacks-breaches/when-encryption-becomes-the-enemys-best-friend/d/d-id/1324580

Wagnon, J. 2015. The BIG-IP Application Security Manager Part 8: Data Guard. F5 DevCentral. Viitattu 29.1.2017. https://devcentral.f5.com/articles/the-big-ip-application-security-manager-part-8-data-guard

Wagnon, J. 2016a. The BIG-IP Application Security Manager Part 1: What is the ASM. F5 DevCentral. Viitattu 4.10.2016. https://devcentral.f5.com/articles/the-big-ip-application-security-manager-part-1-what-is-the-asm

Wagnon, J. 2016b. Integrating WhiteHat Scans With BIG-IP ASM. F5 DevCentral. Viitattu 24.11.2016. https://devcentral.f5.com/articles/integrating-whitehat-scans-with-big-ip-asm

Wagnon, J. 2016c. The BIG-IP Application Security Manager Part 4: Attack Signatures. F5 DevCentral. Viitattu 1.12.2016. https://devcentral.f5.com/articles/the-big-ip-application-security-manager-part-4-attack-signatures

Wagnon, J. 2016d. The BIG-IP Application Security Manager Part 3: The Importance of File Types, Parameters and URLs. F5 DevCentral. Viitattu 22.12.2016. https://devcentral.f5.com/articles/the-big-ip-application-security-manager-part-3-the-importance-of-file-types-parameters-and-urls#.UkSeHrEo7IV

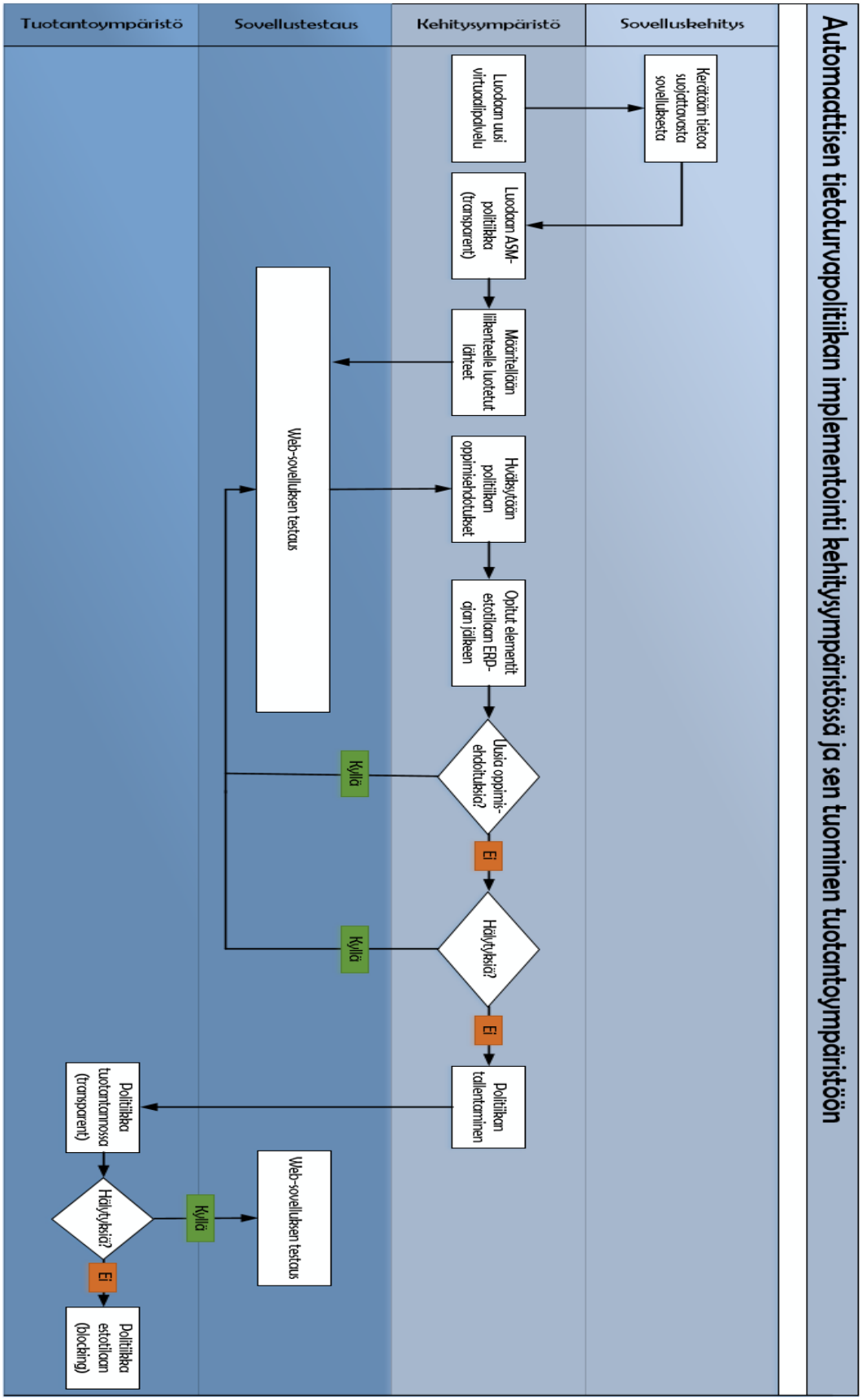
Web Application Firewall Product Analysis. N.d. NSS Labs. Viitattu 13.1.2017. https://interact.f5.com/rs/f5/images/Web%20Application%20Firewall%20Product%20Analysis%20-%20F5%20Big-IP%20ASM%2010200.pdf

Web Applications Security Statistics Report. 2016. WhiteHat Security. Viitattu 11.11.2016. https://www.whitehatsec.com/resources-category/threat-reports/

Web Application Vulnerability Statistics. 2013. Positive Technologies. Viitattu 11.11.2016. https://www.ptsecurity.com/upload/iblock/478/4780da188f459678353ab7ae007ed415.pdf

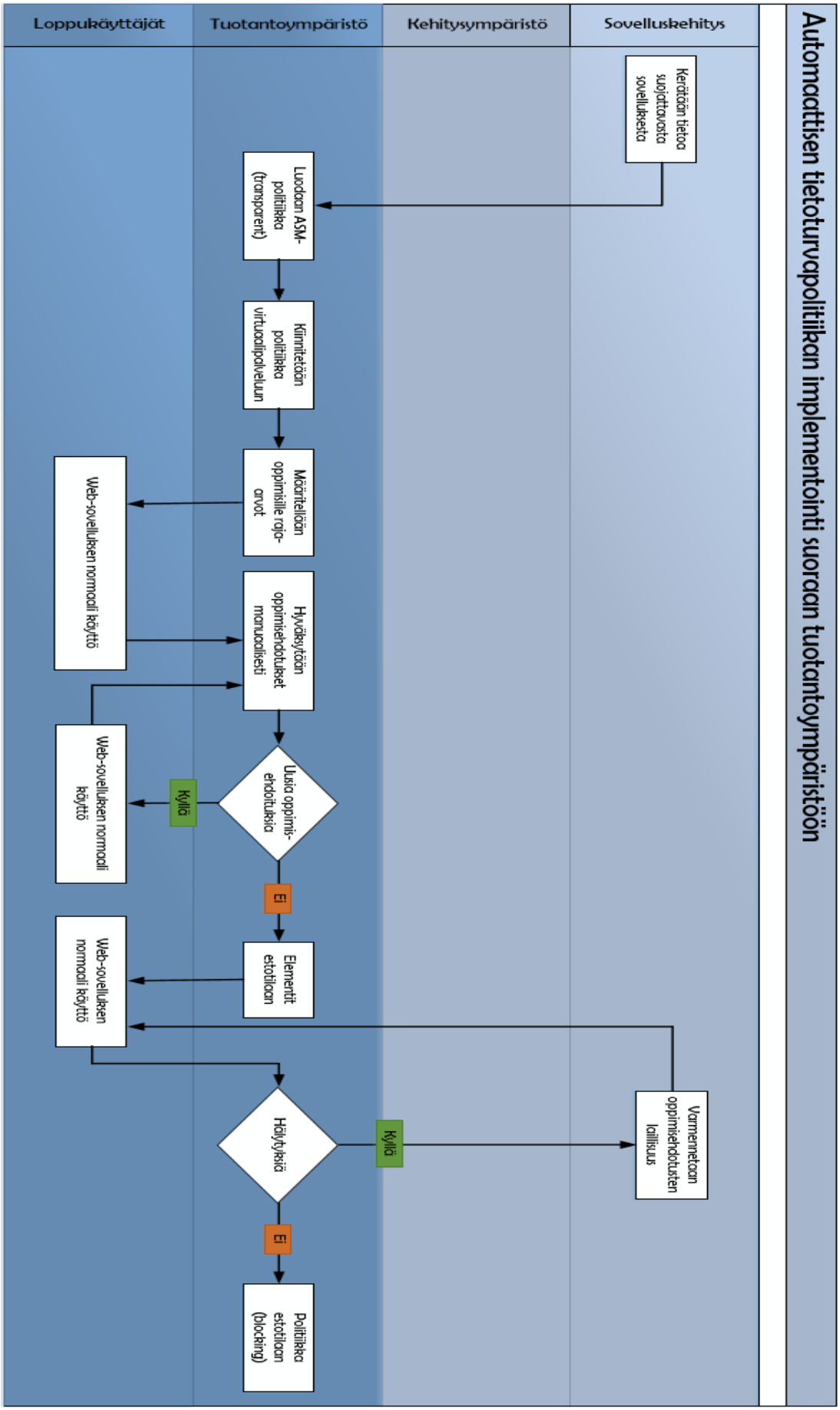
Liitteet

Liite 1. Automaattisen politiikan rakentaminen kehitysympäristössä



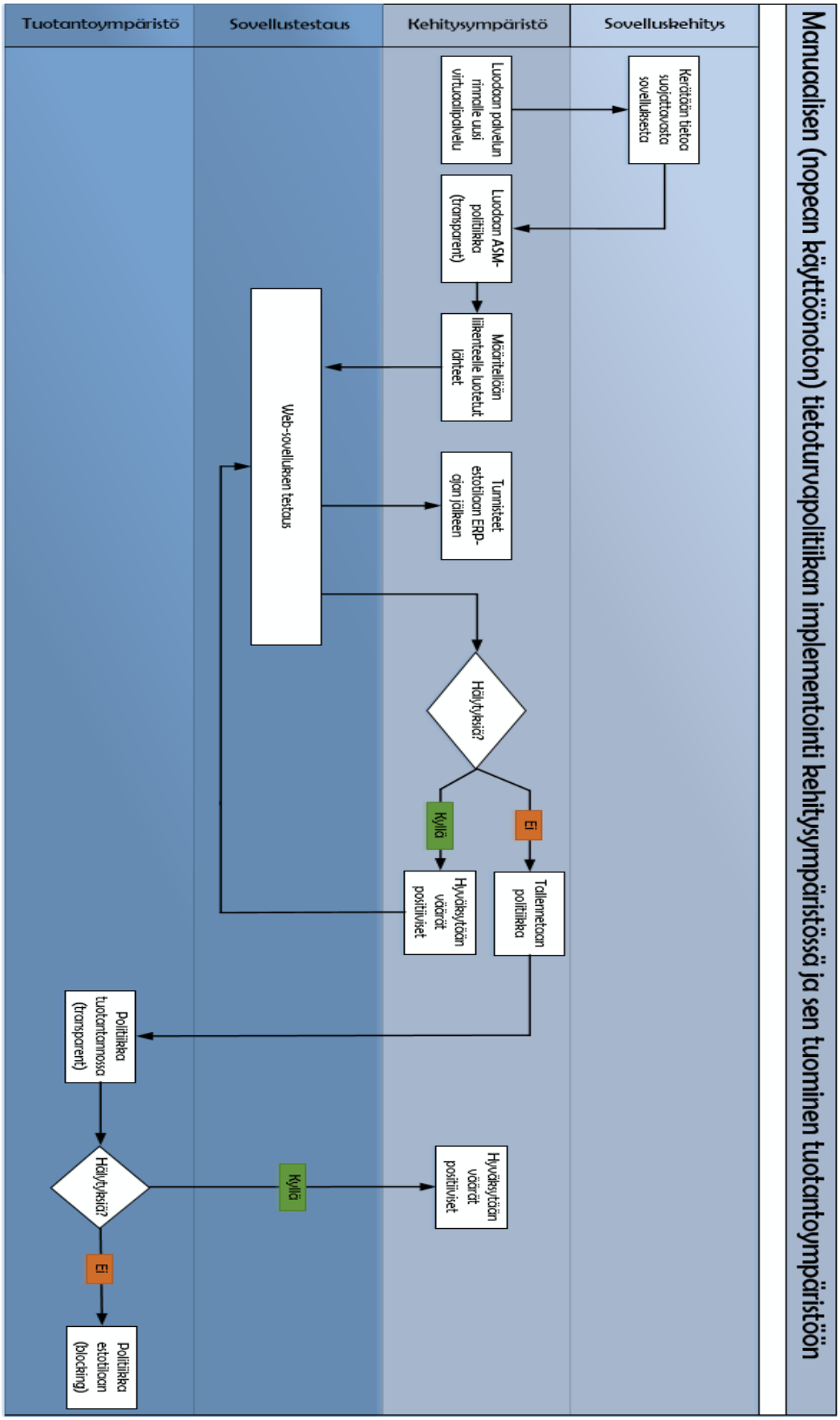
1. Luodaan suojattavalle web-sovellukselle F5-virtuaalipalvelu.
2. Kerätään tarvittavat tiedot suojattavasta web-sovelluksesta.
3. Luodaan automaattinen tilanteeseen sopiva politiikka (tarkkailutila).
4. Määritellään testauksessa käytettävät luotettavat IP-osoitteet.
5. Testataan web-sovellusta luotetuista IP-osoitteista.
6. Seurataan automaattisen politiikanrakentajan generoimia oppimisehdotuksia ja hyväksytään niitä politiikkaan manuaalisesti.
7. Hyväksytään politiikkaan manuaalisesti elementtien attribuuteissa tapahtuvia muutoksia.
8. Siirretään yksittäiset elementit manuaalisesti estotilaan, niiden pysyessä muuttumattomana koko tarkkailuajanjakson ajan.
9. Karsitaan politiikasta turhat väärät positiiviset hälytykset.
10. Testataan web-sovellusta, kunnes politiikassa ei ilmene enää vääriä positiivisia hälytyksiä.
11. Tallennetaan valmis testiympäristön politiikka.
12. Viedään politiikka tuotantoympäristöön (tarkkailutila).
13. Seurataan politiikkaa tarkkailutilassa vielä muutama viikko tarkkailutilassa.
14. Mikäli vääriä positiivisia hälytyksiä havaitaan edelleen, palautetaan politiikka testiympäristöön ja jatketaan sovelluksen testausta.
15. Mikäli vääriä positiivisa hälytyksiä ei havaita, asetetaan politiikka estotilaan tuotantoympäristössä.

Liite 2. Automaattisen politiikan rakentaminen tuotantoympäristössä



1. Kerätään tarvittavat tiedot suojattavasta web-sovelluksesta.
2. Luodaan politiikka jo olemassa olevalle F5-virtuaalipalvelulle (tarkkailutila).
3. Määritellään automaattisen politiikanrakentajan liikenteen oppimiseen käytettävät raja-arvot.
4. (Määritellään luotetut IP-osoitteet)
5. (Testataan web-sovellusta luotetuista IP-osoitteista)
6. Politiikka ja web-sovellus tuotantokäytössä. Politiikkaan generoidaan automaattisesti oppimisehdotuksia web-sovelluksen rakenteesta, kun ennalta määritellyt raja-arvot täyttyvät.
7. Seurataan politiikan lokitietoja ja hyväksytään politiikkaan sen generoimia oppimisehdotuksia tapauskohtaisesti.
8. Hyväksytään tapauskohtaisesti politiikan generoimat muutosehdotukset sen yksittäisissä elementeissä.
9. Siirretään politiikan yksittäisiä elementtejä estotilaan, kun ne ovat läpäisseet tarkkailuajanjakson ilman muutoksia.
10. Kaikkien elementtien ollessa estotilassa, karsitaan järjestelmät tuottamat väärät positiiviset hälytykset. Koska liikenne tulee epäluotetusta lähteestä, on jokainen mahdollinen hälytys analysoitava tarkkaan ja pääteltävä onko kyseessä väärä positiivinen- vai oikea positiivinen hälytys.
11. Kun politiikan tuottamat väärät hälytykset on saatu karsittua, siirretään politiikka globaalisti estotilaan, jolloin se alkaa estämään haitallista liikennettä.

Liite 3. Manuaalisen politiikan rakentaminen kehitysympäristössä



1. Luodaan suojattavalle web-sovellukselle F5-virtuaalipalvelu.
2. Kerätään tarvittavat tiedot suojattavasta web-sovelluksesta.
3. Luodaan manuaalinen politiikka käyttäen, joko sovellusvalmiita pohjia tai nopean käyttöönoton politiikkaa (tarkkailutila)
4. (Määritellään politiikkaan manuaalisesti mahdolliset web-sovelluksen käyttämät rakenteelliset kriittiset elementit ja määritellään niille tarkat attribuutit)
5. Määritetään testauksessa käytettävät luotetut IP-osoitteet.
6. Testataan web-sovellusta luotetuista IP-osoitteista
7. Seurataan lokitietoja ja karsitaan politiikan hyökkäystunnisteiden mahdollisesti aiheuttamat väärät positiiviset hälytykset.
8. Asetetaan hyökkäystunnisteet estotilaan, kun ne eivät enää tuota vääriä positiivisia hälytyksiä.
9. Tallennetaan politiikka ja viedään se tuotantoympäristöön (tarkkailutila)
10. Ajetaan politiikkaa tuotantoympäristössä tarkkailutilassa muutama viikko ja seurataan sen tuottamia lokitietoja ja mahdollisia uusia vääriä positiivisia hälytyksiä.
11. Asetetaan politiikka globaalisti estotilaan.

Liite 4. Ote sanakirjasta väärien positiivisten hälytysten testissä

this or this or --> this
select one thing from these funny things :)))
selection of ascii art works
i prefer delete this item from the file
-2.00--3.00
6490/10*4*1=
:')
:((
:'(
:'''''(
20%
%20
pro%entti
16:05:01
TESTI_6.4.2245.52.525.18
1.55€€€
&%"!%&¤
t: script-kiddie
%2F
%2
%2RTVR5
%2Testi
testi</>
--testi/-testi
’testi*’
testi
Testi testi etc...
Avasin .exe tiedoston
Avasin .bat tiedoston
Liitteenä .jpg tiedosto
Liitteenä .pdf tiedosto
Adèle
O'Neill
Aliénor
Gøran
pass%word
Testi//Testi
Testi||Testi
this or 1234567