Minna Lehtomäki

Sovelluskerroksella tapahtuvat DDoS-hyökkäykset

Tietotekniikan pro gradu -tutkielma

29. toukokuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Minna Lehtomäki

Yhteystiedot: minna.j.lehtomaki@gmail.com

Ohjaaja(t): Timo Hämäläinen

Työn nimi: Sovelluskerroksella tapahtuvat DDoS-hyökkäykset

Title in English: Application layer DDoS-attacks

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 92+1

Tiivistelmä: Tutkielmassa esitellään sovelluskerroksella tapahtuvia hajautettuja palvelunestohyökkäyksiä eli DDoS-hyökkäyksiä, niiden havaitsemista ja niiltä suojautumista. Lisäksi perehdytään yksityiskohtaisemmin salatuissa yhteyksissä tapahtuvaan hyökkäysten havaitsemiseen. Tarkoituksena on tarjota kattava tietopaketti sovelluskerroksesta ja sen taustana olevista protokollista, palvelunestohyökkäyksistä, DDoS-hyökkäysten havaitsemiseen liittyvästä viimeaikaisesta tutkimuksesta sekä hyökkäyksiltä suojautumisesta.

Käytännönosiona simuloidaan erilaisia hitaita hyökkäyksiä (SlowLoris, RUDY, SlowRead), RangeAttack-hyökkäys, porttiskannauksia sekä laskennallinen SSL-hyökkäys salatuissa yhteyksissä ja yritetään havaita ne pelkistä otsakkeista saatavien tietojen perusteella. Havaitsemismenetelmä perustuu liikenteen vuon klusterointiin sekä neuroverkkoihin ja sillä pystyttiin testauksessa tunnistamaan kaikki tehdyt hyökkäystyypit.

Avainsanat: DoS, DDoS, palvelunestohyökkäys, hajautettu palvelunestohyökkäys, sovelluskerros, hyökkäyksen havaitseminen, hyökkäykseltä suojautuminen, salatut yhteydet, SSL DDoS

Abstract: Thesis is about application layer distributed denial of service attacks (DDoS-attacks), how to detect them and how to defend against them. The goal is also to get more detailed insight about detection of DDoS-attacks in encrypted traffic. Thesis offers a lot of basic knowledge about application layer and it's protocols, recent studies about application layer DDoS-detection and information about DDoS-protection mechanisms.

Practical part concerns simulation of DDoS-attack in SSL-encrypted traffic using attacks like SlowLoris, RUDY, SlowRead, RangeAttack, port scanning and SSL computational DoS. The idea is to detect attacks using only information extracted from the packet headers. Method is based on clustering of the traffic flow and use of neural networks. Used method successfully detected all different kinds of attack.

Keywords: DoS, DDoS, Denial of Service, Distributed Denial of Service, Application layer, layer 7, attack detection, DDoS-protection, encrypted traffic, SSL DDoS

Termiluettelo

ACL	Access Control List on lista, joka määrittelee järjestelmän sallitut toiminnot ja ketkä niitä saavat suorittaa.
AE	Auto-Encoder on neuroverkko, jonka tarkoitus on oppia datan koodaus siten, että se pystyy puristamaan tiedon kokoon ja purkamaan takaisin.
ARP	Address Resolution Protocol on verkkokerroksen protokolla, jonka avulla muutetaan verkko-osoitteita (esim. IP-osoite) fyysisiksi osoitteiksi (esim. MAC-osoite).
ASV	Adaptive Selective Verification on valitsevaan varmistamiseen perustuva DDoS-hyökkäysten torjuntamekanismi.
BOOTP	Bootstrap Protocol on protokolla, joka automaattisesti antaa laitteille IP-osoitteet konfigurointipalvelimelta.
Botti	Ohjelma, joka suorittaa automaattisesti skriptejä internetin yli.
C4.5-päätöspuu	On päätöspuiden luomiseen tarkoitettu algoritmi.
CAPTCHA	Completely Automated Public Turing Test to Tell Computers and Humans Apart on ohjelma, joka tuottaa sekä toteuttaa testejä ihmisten erottamiseksi tietokoneista.
CHARGEN	Character Generator Protocol on TCP/IP-viitemalliin kuuluva testaukseen ja debuggaukseen tarkoitettu palvelu, joka lähettää satunnaisia merkkejä TCP- tai UDP-yhteyden yli isäntäkoneelle.
COD	Continuous Outlier Detection on datavirtojen poikkeamien havaitsemisalgoritmi.
CPP	Client Puzzle Protocol on algoritmi, joka vaatii käyttäjältä matemaattisen pulman ratkaisua ennen yhdistämistä palvelimelle.

DRDoS	Distributed Reflection Denial of Service on väärennetyillä IP-osoitteilla tapahtuva heijastettu palvelunestohyökkäys.
DDoS	Distributed Denial of Service eli hajautettu palvelunesto-hyökkäys.
DHCP	Dynamic Host Configuration Protocol on protokolla, joka ja-kaa IP-osoitteita lähiverkkoon kytkeytyville laitteille.
DNS	Domain Name System eli nimipalvelu on sovelluskerroksen protokolla, joka muuntaa verkkotunnuksia IP-osoitteiksi.
DoS	Denial of Service eli palvelunestohyökkäys.
Dstream	Tiheyteen perustuva datavirojen klusterointialgoritmi.
E2E	End-to-End (security) on sähköpostin suojausperiaate, jossa suojataan koko reitti lähettäjältä vastaanottajalle.
ESS	Enhanced Security Services on turvallisuuslaajennus S/MIME-standardille.
FTP	File Transfer Protocol on internetin yli toimiva sovellusker-roksen tiedostonsiirtoprotokolla.
Gopher	TCP/IP-viitemallin sovelluskerroksen protokolla, jota käyte-tään tiedostojen jakamiseen, etsimiseen ja noutamiseen Inter-netissä. Gopher on HTTP-protokollan edeltäjä.
HTTP	Hypertext Transfer Protocol on internetin perustana oleva pro-tokolla.
HTTPS	HTTP-protokollan käyttöä yli SSL- tai TLS-salatun yhteyden.
ICMP	Internet Control Message Protocol on verkkokerroksen proto-kolla, joka mahdollistaa virheviestien lähetyksen verkon lait-teille.
IDMS	Intelligent DDoS Mitigation System eli älykäs hajautettujen palvelunestohyökkäysten lieventämisjärjestelmä.

IDS	Intrusion Detection System eli hyökkäyksen havaitsemisjärjestelmä on järjestelmä tai palvelu, joka valvoo verkon liikennettä ja ilmoittaa rikkomuksista hallinta-asemalle.
IEC	International Electrotechnical Commission on kansainvälinen sähköalan standardointijärjestö.
IETF	Internet Engineering Task Force on internetin protokollien standardointijärjestö.
ILOF	Incremental Local Outlier Factor on datavirtojen poikkeamien havaitsemisalgoritmi.
IMAP	Internet Message Access Protocol on sovelluskerroksen protokolla, jolla luetaan sähköposteja.
IP	Internet Protocol on protokolla, joka vastaa IP-tietoliikennepakettien toimittamisesta internetissä.
ISO	International Organization for Standardization on kansainvälinen standardoimisjärjestö.
ISP	Internet Service Provider eli internetin palveluntarjoaja tarjoaa asiakkailleen internet-yhteyden.
LAND	Local Area Network Denial on eräs palvelunestohyökkäys.
LFG	Lexical Functional Grammar on kieliopin viitekehys.
MAC	Message Authentication Code on kryptografiassa käytettävä tiedonpala, jolla varmistetaan viestin lähettäjä.
MAC-osoite	Media Access Control address on verkkoon liitetyn laitteen fyysinen osoite.
MD5	Message Digest algorithm 5 on kryptografinen salausalgoritmi.

MITM	Man-In-The-Middle eli mies-välissä hyökkäys on tietoturvahyökkäys, jossa hyökkääjä asettuu keskustelevien osapuolten väliin.
NFS	Network File System on protokolla, jonka avulla tiedostoihin pääsee käsiksi etäyhteyden avulla.
NMS	Network Management Station eli tietoverkon hallinta-asema hallitsee ja valvoo tietoverkon laitteita.
OSI-viitemalli	Open Systems Interconnection Reference Model on 7-kerroksinen malli, jolla kuvataan verkon rakennetta ja protokollia.
P2P	Peer-to-Peer (security) on sähköpostin suojausperiaate, jossa suojataan reitti lähettäjältä lähimmälle sähköpostipalvelimelle.
Ping	Packet InterNet Groper on verkonhallintatyökalu, jolla voidaan testata laitteen saavutettavuutta verkossa.
PCT	Private Communications Technology on Microsoftin vastine SSL-protokollalle.
POD-hyökkäys	Ping of Death on eräs palvelunestohyökkäys.
POP3	Post Office Protocol version 3 on sähköpostien noutamiseen tarkoitettu protokolla.
RBF-neuroverkko	Radial Basis Function neural network eli radiaalikantafunktioverkko on eräänlainen neuroverkko.
RR	Resource Records ovat DNS-tietokannassa olevia tietueita.
RUDY	R-U-Dead-Yet on eräs palvelunestohyökkäys.
SAP	Service Access Point on eri OSI-viitemallin kerrosten välillä sijaitseva palvelupiste tai rajapinta.
SIP	Session Initiation Protocol on multimediaa välittävissä kommunikointiyhteyksissä käytettävä tietoliikenneprotokolla.

S/MIME	Secure/Multipurpose Internet Mail Extensions on standardi, jonka avulla salataan ja allekirjoitetaan sähköpostit julkisen avaimen salausta käyttämällä.
SMTP	Simple Mail Transfer Protocol on sovelluskerroksen protokolla sähköpostien lähettämiseen palvelinten välillä.
SNMP	Simple Network Management Protocol on verkonhallintaan tarkoitettu tietoliikenneprotokolla.
SOAP	Simple Object Access Protocol on web-palveluiden käyttämä protokolla jäsennellyn tiedon välittämiseen verkossa.
SOCKS	Socket Secure on OSI-viitemallin istuntokerroksen protokolla, joka kuljettaa verkon paketteja palvelimen ja asiakkaan välillä välityspalvelimen läpi.
Spoofaus/Spoofing	Tarkoittaa toisena laitteena esiintymistä dataa väärentämällä.
SSH	Secure Shell on kryptografinen verkkoprotokolla, jota käytetään liikenteen salaamiseen.
SSL	Secure Sockets Layer on tietoturvateknologia, jolla voidaan muodostaa salattu yhteys.
STLP	Secure Transport Layer Protocol on Microsoftin vastine SSL-protokollalle.
TCP	Transmission Control Protocol on tietoliikenneprotokolla, jolla luodaan yhteys laitteiden välille internetin yli tiedonsiirtoa varten.
TCP/IP-viitemalli	On 4-kerroksinen malli, jolla kuvataan verkon rakennetta ja protokollia.
Telnet	Telecommunications Network Protocol on yhteysprotokolla, jolla luodaan virtuaalisia pääteyhteyksiä laitteiden välille internetin yli.

TFTP	Trivial File Transfer Protocol on yksinkertainen protokolla, jolla voidaan siirtää tiedostoja etäyhteyden yli.
TLS	Transport Layer Security on SSL-teknologian seuraaja.
TMH	Trust Management Helmet eli luottamuksenhallintamenetelmä on eräs tapa havaita palvelunestohyökkäyksiä.
UDP	User Datagram Protocol on yhteydetön protokolla eli se mahdollistaa tiedon siirron laitteiden välillä internetin yli ilman yhteyttä.
URI	Uniform Resource Identifier on jono merkkejä, joiden avulla voidaan ilmaista jonkin tiedon paikka internetissä.
URL	Uniform Resource Locator eli epävirallisesti nettiosoite määrittää tiedon paikan internetissä. Se on eräs URI:n tyyppi.
VoIP	Voice over Internet Protocol on protokolla, jolla voidaan suorittaa puhe- tai multimediaistuntoja verkon yli.
VPN	Virtual Private Network eli virtuaalinen erillisverkko on teknologia, jolla yhdistetään yksityisiä verkkoja julkisen verkon yli.
WAF	Web Application Firewall on palomuuri, joka valvoo web-palvelimelle saapuvaa ja siltä lähtevää HTTP-liikennettä.
XML	Extensive Markup Language on merkintäkieli, joka määrittelee tai kuvailee kahden osapuolen välillä vaihdettavaa dataa.

Kuviot

Taulukot

Sisältö

1 Johdanto

Palvelunestohyökkäykset eli DoS-hyökkäykset ovat hyökkäyksiä, joissa uhrina olevan palvelun tai koneen toiminta estetään ylikuormittamalla se pyynnöillä. Nykyään palvelunestohyökkäykset toteutetaan usein niin sanotulla bottiverkolla, jonka koko voi olla muutamasta botista kymmeniin tuhansiin botteihin. Botit ovat yleensä tavallisten internetin käyttäjien koneita, jotka on valjastettu bottiverkon käyttöön haittaohjelmien avulla. Hyökkäyksen aikana jokainen bottiverkon botti lähettää pyyntöjä samanaikaisesti kohdelaitteelle, jolloin liikenne tulee yhden lähteen sijasta hajautetusti monesta lähteestä. Tästä syystä bottiverkoilla toteutettua palvelunestohyökkäystä kutsutaan hajautetuksi palvelunestohyökkäykseksi eli DDoS-hyökkäykseksi.

Palvelunestohyökkäykset voivat tapahtua sekä TCP/IP-mallin verkko- että sovelluskerroksella, mutta toteutustapa vaihtelee kerroksesta riippuen. Kuljetuskerroksen hyökkäykset perustuvat yleensä TCP- ja UDP-protokolliin, jolloin kohteena olevaa palvelua kuormitetaan suurella määrällä kyseisten protokollien yhteyspyyntöjä. Tätä kutsutaan tulvahyökkäykseksi. Sovelluskerroksella tapahtuvat palvelunestohyökkäykset perustuvat myös kyseisen kerroksen protokollien, kuten HTTP-protokollan, hyväksikäyttöön. Sovelluskerroksella hyökkäykset on kuitenkin hankalampi havaita kuljetuskerrokseen verrattuna, koska edistyneet hyökkäykset perustuvat hyökkäysliikenteen naamioimiseen siten, että se muistuttaa mahdollisimman paljon aitoa liikennettä. Lisäksi yhteyden salaaminen asettaa omat haasteensa sovelluskerroksella tapahtuvien hyökkäysten havaitsemiselle, koska salattu yhteys ei kuitenkaan itsessään suojaa palvelunestohyökkäyksiltä.

1.1 Hyökkäysten yleisyys ja vaikutukset

Tietoturva- ja verkonhallintaohjelmia myyvän Arbor Networksin tekemän tutkimuksen mukaan hajautetut palvelunestohyökkäykset ovat yleisin uhka palveluntarjoajille. Suurin raportoitu hyökkäys oli vuonna 2015 500 Gbps ja viimeiset vuodet trendinä on ollut, että hyökkäykset ovat aina vain suurempia. Toinen trendi on pilvipalveluihin kohdistuvien hyökkäysten kasvu. Palveluntarjoajien asiakkaat ovat hyökkäyksen yleisin kohde ja kaksi kolmasosaa

hyökkäyksistä kohdistuukin heihin. Moniulotteisten hyökkäysten eli useita eri hyökkäystapoja yhdistävien hyökkäysten määrä on myös kasvanut ja tällä hetkellä reilu puolet hyökkäyksistä on tällaisia. Yleisimmät palvelut, joihin sovelluskerroksen palvelunestohyökkäykset kohdistuvat ovat DNS, HTTP ja HTTPS. (Arbor Networks 2016).

Tietoturvayhtiö Incapsula puolestaan havaitsi, että vuoden 2015 viimeisellä kvartaalilla sovelluskerroksella toistuvien palvelunestohyökkäysten määrä kasvoi 15 % edelliseen kvartaaliin nähden. Toistuvissa palvelunestohyökkäyksissä samaan uhriin kohdistuu monta peräkkäistä hyökkäystä, joista jokainen kestää vain vähän aikaa. Tarkoituksena tällaisella hyökkäystyylillä on kohdistaa uhriin jatkuvaa painetta sekä hyväksikäyttää niiden suojautumismenetelmien heikkoutta, joiden uudelleen aktivoimiseen jokaisen hyökkäyksen jälkeen kuluu aikaa. Sama ilmiö on tuotu esille Arbor Networksin tutkimuksessa (Arbor Networks 2016). (Incapsula 2016).

Incapsulan mukaan vuoden 2015 neljännellä kvartaalilla noin 45 % uhreista koki hyökkäyksen useammin kuin kerran. Sellaisten uhrien määrä, joita vastaan hyökättiin yli 10 kertaa, tuplaantui reilusta 5 prosentista melkein 11 prosenttiin. Aiemmin vuonna 2015 suurin osa sovelluskerroksen hyökkäyksistä kesti useita tunteja kerrallaan ja yli 24 tunnin hyökkäykset olivat harvinaisia. Vuoden 2015 lopussa puolestaan hyökkäysten kesto oli 58 prosentissa tapauksista alle tunnin ja 23 prosentissa kesto oli yhdestä kolmeen tuntia. Pisin sovelluskerroksen hyökkäys kesti 101 päivää ja suurimmassa hyökkäyksessä päästiin yli 161 000 pyyntöön sekunnissa. (Incapsula 2016). Arbor Networksin tutkimuksessa noin 35 % hyökkäyksistä kesti 1-6 tuntia ja noin 18 % kesti alle tunnin (Arbor Networks 2016).

Incapsulan mukaan eniten hyökkäyksiä tekivät Kiina (39,8 %), Etelä-Korea (12,6 %), USA (11,7 %) ja Vietnam (5,8 %). Hyökkäysten kohteena olivat eniten USA (47,6 %), Iso-Britannia (23,2 %) ja Japani (8,6 %). Näistä Iso-Britannia ja Japani kokivat selvän kasvun edelliseen kvartaaliin verrattuna, kun Iso-Britanniassa hyökkäysten määrä nousi 20,7 prosenttiyksiköllä ja Japanissa hyökkäysten määrä nousi 7,5 prosenttiyksiköllä. (Incapsula 2016).

Arbor Networks puolestaan listasi, että eniten hyökkäyksiä tekivät USA (12,8 %), Etelä-Korea (7,1 %) ja Kanada (5,6 %). Kummassakin tutkimuksessa siis USA ja Etelä-Korea pääsivät kolmen suurimman joukkoon hyökkäysten tekemisessä. Arbor Networksin mukaan

suosituimpia hyökkäyksen kohteita olivat USA (32,2 %), Kiina (10,5 %) ja Ranska (6,4 %). (Arbor Networks 2016). Ainoa yhteinen kohteena ollut maa on täten molemmissa tutkimuksissa USA.

Incapsula on tutkinut hyökkäysten vaikutuksia yrityksille vuonna 2014. Tutkimus koostuu Pohjois-Amerikkalaisista yrityksistä ja sisältää tiedon siitä, mitä hajautetut palvelunestohyökkäykset maksavat yrityksille. Melkein puolet (45 %) vastaajista oli kokenut palvelunestohyökkäyksen ja näistä kaksi kolmasosaa (70 %) oli kokenut useita hyökkäyksiä. Sellaiset yritykset, joissa on yli 500 työntekijää, ovat todennäköisimpiä kohteita. Tällöin myös hyökkäyksen aiheuttamat kustannukset yritykselle ovat pieniä yrityksiä suuremmat ja hyökkäykseltä puolustautuminen vaatii enemmän henkilöstöä. Puolet hajautetuista palvelunestohyökkäyksistä kestää 6-24 tuntia ja keskimääräinen tuntikustannus on yritykselle 40 000$. Tällöin yhden hyökkäyksen keskimääräinen kustannus on noin 500 000$. Kustannus koostuu IT-yksikön ja esimerkiksi myyntitulojen menetyksien aiheuttamista kustannuksista. Lisäksi hyökkäys aiheuttaa kustannuksia asiakkaiden luottamuskadon ja asiakastietojen menetyksen muodossa, aineettoman omaisuuden menetyksiä sekä laitteiston ja ohjelmistojen korvaamisesta aiheutuvia kustannuksia. (Incapsula 2014).

1.2 Tutkimusongelma

Tässä pro gradu -tutkielmassa perehdytään sovelluskerroksella tapahtuvien DDoS-hyökkäysten havaitsemiseen ja niiltä suojautumiseen. Tarkoituksena on tarjota aiheeseen riittävät taustatiedot sekä avata lukijalle mitä sovelluskerroksella tapahtuvat hajautetut palvelunestohyökkäykset ovat ja millaisia vaikutuksia niillä on, miten niitä tehdään, minkälaisia motiiveja hyökkääjillä on, kuinka hyökkäyksiä voidaan havaita ja torjua sekä kuinka hyökkäyksiltä voidaan suojautua. Päämääränä on näiltä osin koota mahdollisimman kattava ja ajankohtainen tietopaketti käyttäen erilaisia tieteellisiä artikkeleita sekä kirjallisuutta.

Lisäksi tutkielman tarkoituksena on ollut perehtyä yksityiskohtaisemmin salatuissa yhteyksissä tapahtuvien DDoS-hyökkäyksien havaitsemiseen. Tämä on tapahtunut suorittamalla käytännön simulaatio Jyväskylän yliopiston tietoliikennelaboratoriossa, jossa on käytössä

ympäristö palvelunestohyökkäysten simuloimiseksi. Salattujen yhteyksien palvelunestohyökkäysten havaitsemismenetelmä perustuu liikenteen ominaisuuksien klusterointiin sekä syväoppimiseen (*deep learning*) kuuluvien pinottujen AE:iden käyttöön. Tarkoituksena simuloinnissa on ollut havaita muun muassa erilaisia sovelluskerroksen hitaita hyökkäyksiä sekä porttiskannauksia.

1.3 Tutkielman rakenne

Tutkielma koostuu kahdesta osiosta. Ensin on kattava tietopaketti sovelluskerrosten palvelunestohyökkäyksistä sekä olennaisista asiaan liittyvistä arkkitehtuureista ja protokollista. Toisena osiona on salattujen palvelunestohyökkäysten havaitsemiseen liittyvä käytännönosio.

Palvelunestohyökkäysten tietopaketti sisältää luvut 2-5. Luvussa 2 esitellään internetin sovelluskerroksen rakennetta, sen protokollia ja yhteyden salaamista SSL/TLS-protokollien avulla. Sovelluskerroksen protokollia tarkasteltaessa otetaan huomioon myös tietoturvanäkökulmat. Luvussa 3 puolestaan perehdytään tarkemmin siihen, mitä DoS- ja DDoS-hyökkäykset ovat sekä minkä tyyppisiä hyökkäyksiä on. Lisäksi esitellään muutamia hyökkäysten simulointiin käytettäviä työkaluja, eritellään vähän motiiveja hyökkäysten takana sekä annetaan muutamia esimerkkejä viimeaikaisista palvelunestohyökkäyksistä. Luvussa 4 tehdään katsaus sovelluskerroksen palvelunestohyökkäysten havaitsemis- ja tunnistamismenetelmiin uusimpien tutkimusartikkelien kautta (julkaistut menetelmät noin vuodesta 2010 eteenpäin). Luvussa 5 esitellään mahdollisia suojautumiskeinoja palvelunestohyökkäyksiä vastaan sekä listataan joitakin tarjolla olevia, suojautumiseen tarkoitettuja palveluita.

Toisena osiona oleva salattujen yhteyksien palvelunestohyökkäysten havaitsemiseen liittyvä käytännönosio, joka kattaa luvun 6. Siinä kerrotaan yksityiskohtaisesti hyökkäysten havaitsemiseen käytetty menetelmä sekä esitellään simulaatiossa käytetty ympäristö ja ohjelmat. Lisäksi luvussa 6 käydään läpi simulaation kulkua ja tuloksia sekä niihin liittyvä pohdintaa. Lopuksi luku 7 sisältää yhteenvedon koko tutkielmasta.

2 Internetin sovelluskerros

Internetin sovelluskerroksella tarkoitetaan lyhyesti kuvattuna OSI-viitemallin tai TCP/IP-viitemallin mukaista internet-arkkitehtuuria, jossa sovellusten keskinäinen kommunikointi verkon yli tapahtuu. Tämän luvun tarkoituksena on antaa yleiskuva siitä, mikä on internetin sovelluskerros. Luvussa 2.1 perehdytään OSI-viitemalliin sekä TCP/IP-viitemalliin. Luvussa 2.2 puolestaan käydään läpi sovelluskerroksen protokollia ja luvussa 2.3 esitellään sovellusten internetin yli tapahtuvan viestinnän salausmenetelmiä.

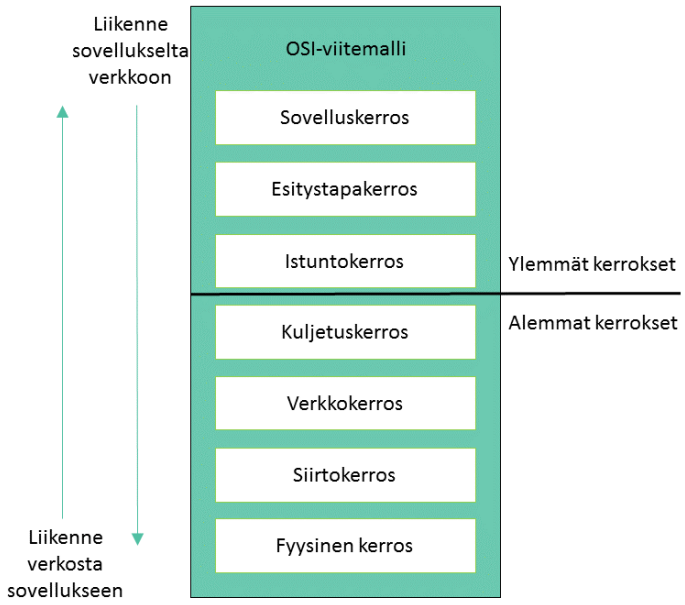
2.1 Internetin arkkitehtuurin kuvaaminen viitemalleilla

2.1.1 OSI-viitemalli

OSI-viitemalli (lyhyemmin OSI-malli) on ISO:n ja IEC:n kehittämä, alun perin vuonna 1984 julkaistu standardi, jossa määritellään kehykset järjestelmien välisille yhteyksille verkon yli. OSI-malli on kerrostettu arkkitehtuurimalli, mikä tarkoittaa sitä, että järjestelmä koostuu loogisiin kokonaisuuksiin jaetuista kerroksista, jotka kommunikoivat niin sisäisesti kuin keskenäänkin. (ISO7498-1).

Samalla kerroksella toimivat entiteetit voivat kommunikoida keskenään joko yhteydellisesti (*connection mode*) tai yhteydettömästi (*connectionless mode*). Yhteydellinen kommunikointi tapahtuu kolmessa vaiheessa: ensin muodostetaan yhteys, sitten siirretään data ja lopuksi yhteys vapautetaan muiden käyttöön. Tämän tiedonvälityksen palvelut tarjoaa aina alempi viitemallin kerros. Yhteydettömässä kommunikoinnissa puolestaan lähetetään dataa pala kerrallaan lähteenä toimivan entiteetin palvelupisteen eli SAP:n kautta kohde-entiteetille, muodostamatta kuitenkaan yhteyttä näiden kahden entiteetin välille. Tällöin jokainen tiedonpala voidaan reitittää kerroksella muista paloista riippumatta. Yhteydettömässä tilassa kerrokset tarjoavat yleensä vain osan yhteydellisen tilan palveluista. Tässä luvussa eri kerrosten palveluita esitellessä kyseessä ovatkin, ellei toisin mainita, nimenomaan yhteydellisessä tilassa tarjotut palvelut. (ISO7498-1).

OSI-viitemalli koostuu seitsemästä kerroksesta, jotka ovat fyysinen kerros (*Physical Layer*), siirtokerros (*Data Link Layer*), verkkokerros (*Network Layer*), kuljetuskerros (*Transport Layer*), istuntokerros (*Session Layer*), esitystapakerros (*Presentation Layer*) ja sovelluskerros (*Application layer*). Kuviossa 1 näkyy OSI-viitemallin rakenne ja viestien liikkumisen suunta. Liikenne sovellukselta verkkoon tapahtuu aina ylemmistä kerroksista kohti alempia ja liikenne verkosta sovellukseen päin tapahtuu alemmista kerroksista kohti ylempiä kerroksia.



Kuvio 1. OSI-viitemallin kerrokset.

Sovelluskerros on mallin ylin kerros, jossa sovellukset viestivät toisilleen käyttäen sovellusprotokollia ja esitystapakerroksen palveluita. Sovelluskerros sisältää ne viestintään liittyvät toiminnot, joita ei ole alemmilla kerroksilla suoritettu. Nämä toiminnot voivat olla joko ihmisten tai ohjelmien suorittamia. Tiedon siirtämisen lisäksi sovelluskerroksen palveluihin voi kuulua viestintäkumppaneiden tunnistaminen (esimerkiksi nimeltä, osoitteelta tai kuvaukselta), palveluiden laadusta päättäminen (esimerkiksi vastausaika tai siedettävä virheiden määrä), yhteistyössä olevien sovellusten synkronointi, virheistä toipumisen vastuullisuudesta sopiminen, tietoturvanäkökulmista (esimerkiksi autentikointi, kulunvalvonta tai

datan eheys) sopiminen, vuoropuhelun käytäntöjen valinta sekä abstraktien syntaksien tunnistaminen. (ISO7498-1).

Sovelluskerroksen alapuolella on esitystapakerros, jonka tarkoituksena on tarjota sovellus-entiteeteille tiedon esitystapa, jota entiteetit voivat käyttää kommunikoinnissa tai johon ne voivat viitata kommunikoinnissaan. Esitystapakerros täten varmistaa, että sovelluskerroksen datan sisältö säilyy siirtämisen aikana. Tällöin sovellus-entiteettien ei tarvitse huolehtia siitä, käyttävätkö ne yhteistä tiedonesitystapaa. Esitystapakerroksen sovelluskerrokselle tarjoamia palveluita ovat siirrossa käytettävien syntaksien tunnistaminen, siirrossa käytettävien syntaksien valitseminen ja pääsyn tarjoaminen istuntokerroksen palveluihin. (ISO7498-1).

Seuraava alempi kerros on istuntokerros, joka tarjoaa esitystapa-entiteeteille mahdollisuuden järjestää ja synkronisoida vuoropuhelunsa sekä hallita datan vaihtoa. Tähän istuntokerros käyttää istunto-yhteyttä, joka huolehtii datan vaihtoa varten tarvitusta vuorovaikutuksesta sekä yhteyden vapauttamisesta. Yhteydettömässä tilassa tapahtuvalle viestinnälle istuntokerros tarjoaa ainoastaan siirto-osoitteiden yhdistämistä (*mapping*) vastaaviin istunto-osoitteisiin ja poikkeuksista tiedottamista. Ylemmälle kerrokselle tarjottuja palveluita yhteydellisessä tilassa ovat tässä kerroksessa istuntoyhteyden muodostaminen, sen synkronointi ja sen vapauttaminen, datan siirto, tunnisteiden (*token*) hallinta, poikkeuksista raportointi, toiminnallisuuden hallinta sekä uudelleensynkronointi. (ISO7498-1).

Keskimmäinen kerros on kuljetuskerros, joka tarjoaa istuntokerrokselle luotettavaa ja tehokasta datan siirtoa istunto-entiteettien välillä. Se siis huolehtii pakettien perille saapumisesta oikeassa järjestyksessä. Kuljetuskerroksen ylemmälle kerrokselle tarjoamia palveluita ovat tiedonsiirtoyhteyden muodostaminen ja sen vapauttaminen, datan siirto ja toimintojen pysäyttäminen. (ISO7498-1).

Kuljetuskerroksen alapuolella on verkkokerros, joka mahdollistaa verkkoyhteyksien luomisen, ylläpitämisen ja päättämisen kommunikoivien sovellusten välillä sekä tarjoaa toiminnot ja prosessit verkkopalveluiden datapakettien lähettämiseen verkkoyhteyden yli. Verkkokerros hoitaa myös reitittämisen sekä välityksen kuljetuskerroksen puolesta. Verkkokerros siis huolehtii internetin yli tapahtuvasta reitityksestä ja viestinnästä. Verkkokerroksen tarjoamia toimintoja ovat verkko-osoitteiden ja verkkoyhteyksien tarjoaminen sekä verkkoyhteyden

päätepisteiden tunnisteiden tarjoaminen, verkkopalveluiden data-pakettien siirto, palveluiden laatuparametrien tarjoaminen, virheistä huomauttaminen, verkkopalveluiden datapakettien nollaus, verkkoyhteyden vapauttaminen ja datan vastaanottamisen kuittaus. Jotkin näistä toiminnoista ovat valinnaisia, jolloin käyttäjän täytyy niitä erikseen pyytää. Tällöin verkon palveluntarjoaja eli ISP voi joko toteuttaa pyynnön tai ilmoittaa, että palvelua ei ole saatavilla. (ISO7498-1).

Siirtokerros on viitemallin toiseksi alimmainen kerros, joka tarjoaa toiminnot ja prosessit yhteydettömään ja yhteydelliseen tilaan verkkoentiteettien välillä sekä siirtopalveluiden datapakettien kuljettamisen. Lisäksi se havaitsee ja mahdollisesti korjaa fyysisellä kerroksella tapahtuneita virheitä. Siirtoyhteys rakennetaan yhden tai useamman fyysisen yhteyden päälle. Siirtokerros siis huolehtii tiedonkulusta lähiverkon laitteiden välillä. Siirtokerros tarjoaa seuraavia toimintoja: siirto-osoitteiden, siirtoyhteyksien sekä siirtoyhteyksien päätepisteiden tunnisteiden tarjoaminen, siirtopalveluiden datapakettien kuljettaminen, virheistä huomauttaminen, palvelun laatuparametreista vastaaminen ja nollaus tiettyyn tilaan. (ISO7498-1).

Fyysinen kerros on OSI-viitemallin alin kerros, joka nimensä mukaisesti koostuu arkkitehtuurin pohjalla olevista laitteista. Fyysisen kerroksen tiedonsiirto koostuu mekaanisesta ja sähköisestä tiedonsiirrosta ja tiedonsiirto tapahtuu fyysisten yhteyksien yli. (ISO7498-1).

2.1.2 TCP/IP-viitemalli

TCP/IP on internetin käytetyin protokolla, joka tosiasiassa koostuu kahdesta erillisestä protokollasta, TCP- ja IP-protokollasta. TCP/IP-viitemallissa viestintä verkon yli tapahtuu nimensä mukaisesti pääasiassa TCP- ja IP-protokollien mukaisesti, vaikka viitemalliin sisältyy muitakin protokollia.

TCP-protokolla on kuljetusprotokolla eli se huolehtii viestien luotettavasta välityksestä päätelaitteiden välillä. Tämä tapahtuu jakamalla viestit pienempiin osiin eli paketteihin, huolehtimalla tiedon kulun sujuvuudesta (sopeutetaan pakettien koko ja kuljetusnopeus sopivaksi) sekä varmistamalla pakettien järjestys ja virheettömyys vastaanottopäässä. TCP on yhteydellinen protokolla, jolloin se muodostaa yhteyden viestin lähettäjän ja vastaanottajan välille

ja lähettää viestin tämän yhteyden kautta. Tämä yhteys pysyy yllä, kunnes kaikki paketit on lähetetty. Protokolla myös takaa pakettien perille saapumisen kuittaamalla ACK-viestillä (*Acknowledgment*), että paketti on saapunut perille. Vasta tämän viestin saavuttua lähetetään seuraava paketti matkaan. Pakettien järjestämiseen ja lähetyksen perille saapumisen varmistamiseen käytetään TCP-otsaketta (*TCP-header*), joka sisältää sekvenssinumerot, ACL-numerot, osoitteet sekä muut yhteyden kannalta oleelliset tiedot. (Blank 2002).

IP-protokolla puolestaan mahdollistaa viestin välityksen kahden laitteen välillä tarjoamalla uniikin osoitteen jokaiselle laitteelle. Tätä osoitetta kutsutaan IP-osoitteeksi ja tällä hetkellä ovat käytössä protokollan versiot 4 (IPv4) ja 6 (IPv6). (Edwards & Bramante 2009).

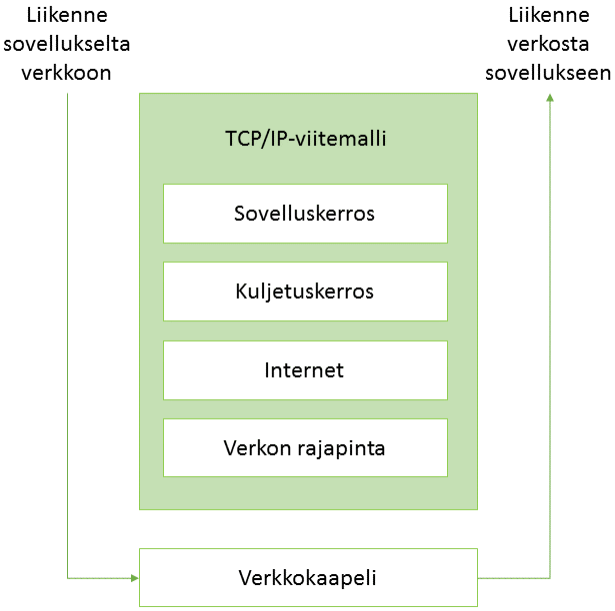
IPv4 käyttää 32 bittistä (eli neljä tavuista) osoitteistoa, joka koostuu verkon numerosta (*network number*) ja paikallisesta eli lokaalista osoitteesta (*local address*). Osoitteet jaetaan vielä kolmeen luokkaan, joissa lokaalille osoitteelle on varattu joko 24, 16 tai 8 viimeistä bittiä. IPv4 voidaan esittää desimaalimuodossa, esimerkiksi 168.192.1.1. (RFC-791).

IPv6 puolestaan käyttää 128 bittistä osoitteistoa, mikä tarkoittaa, että osoitteita voi olla aiempaa enemmän, osoitteille saa enemmän hierarkiatasoja ja osoitteita on helpompi konfiguroida automaattisesti. IPv6 osoite esitetään muodossa, jossa on neljä heksadesimaalilukua ryhmässään ja peräkkäisiä ryhmiä on kahdeksan kaksoispisteillä eroteltuna, esimerkiksi FE80:0000:0000:0000:0202:B3FF:FE1E:8329. Tarvittaessa peräkkäiset nollasarjat voidaan korvata myös kahdella kaksoispisteellä, jolloin sama osoite voidaan ilmaista muodossa FE80::0202:B3FF:FE1E:8329. (RFC-1883).

Syitä TCP/IP-protokollan käyttöön ovat muun muassa reititys, osoitteisto, nimipalvelu (*name resolution*), kyky toimia monella eri kerroksella sekä avoimet standardit. TCP/IP-protokolla on riippumaton solmusta (*node*), sen tekijästä, käyttöjärjestelmästä sekä sijainnista. Tämän vuoksi internetin mahdollisuudet ovat melkein rajattomat. (Edwards & Bramante 2009).

TCP/IP-viitemallin standardit ja prosessit voidaan jakaa neljään kerrokseen, jotka ovat sovelluskerros (*Application layer*), kuljetuskerros (*Transport layer*), Internet-kerros (*Internet layer*) ja verkon rajapintakerros (*Network Interface layer*). Näistä sovelluskerros on ainoa, joka ei varsinaisesti osallistu tiedon liikuttamiseen. Kuten kuviosta 2 voidaan nähdä, paketit

liikkuvat kerrosten läpi sovelluskerrokselta alempiin kerroksiin, jonka jälkeen ne kulkevat pitkin verkkokaapelia kohteen verkon rajapintaan ja matkaavat jälleen kerrosten läpi itse sovellukselle. Ylempien kerrosten protokollat toimivat aina omalla kerroksellaan, joten niiden ei tarvitse huolehtia alempien kerrosten toiminnoista. (Edwards & Bramante 2009).



Kuvio 2. TCP/IP-viitemallin kerrokset.

Verkon rajapintakerrosta kutsutaan usein myös linkkikerrokseksi (*Link layer*) tai data-linkkikerrokseksi (*Data Link layer*). Sen vastuulla ovat niiden laitteen ajurien ja laitteiston rajapintojen hallinta, jotka yhdistävät solmun verkkokaapeliin. (Edwards & Bramante 2009).

Internet-kerros, tai joskus myös verkkokerros (*Networking layer*), vastaa pakettien kuljetuksesta verkon läpi. Tälle kerrokselle kuuluvat siis kaikki reitittämiseen liittyvät protokollat, kuten ARP ja IP. Solmut, jotka toimivat tällä kerroksella, vastaanottavat datagrammeja, selvittävät minne datagrammit lähetetään ja lähettävät ne määränpäähänsä. Lisäksi tällä kerroksella päätetään menetelmä, jolla kyseiselle solmulle kuuluvien pakettien tiedot lähetetään eteenpäin. Internet-kerros myös sisältää protokollat virheviestien vastaanottamiseen ja lähettämiseen sekä viestien hallintaan. (Edwards & Bramante 2009).

Internet-kerroksella toimii ICMP-protokolla, jota käytetään lähinnä virheviestien välittämiseen, vianmäärityksen tekemiseen sekä datavirran hallintaan. Reitittimet antavat koneen lähettää dataa mahdollisimman nopeasti. Silloin, kun reitittimellä on liikaa liikennettä, lähetetään ICMP-paketti, jossa pyydetään hidastamaan lähetystahtia. Yleisesti käytetty ICMP-protokollan palvelu on yhdistettävyyden tutkimiseen käytetty Ping. Ping-työkalulla voidaan lähettää pienen määrän dataa sisältäviä ICMP-kaiutuspaketteja kohdekoneelle ja pyytää kohdetta lähettämään nämä paketit takaisin. Mikäli paketit palautuvat onnistuneesti takaisin, ei yhteysongelmaa kohteeseen ole. (Blank 2002).

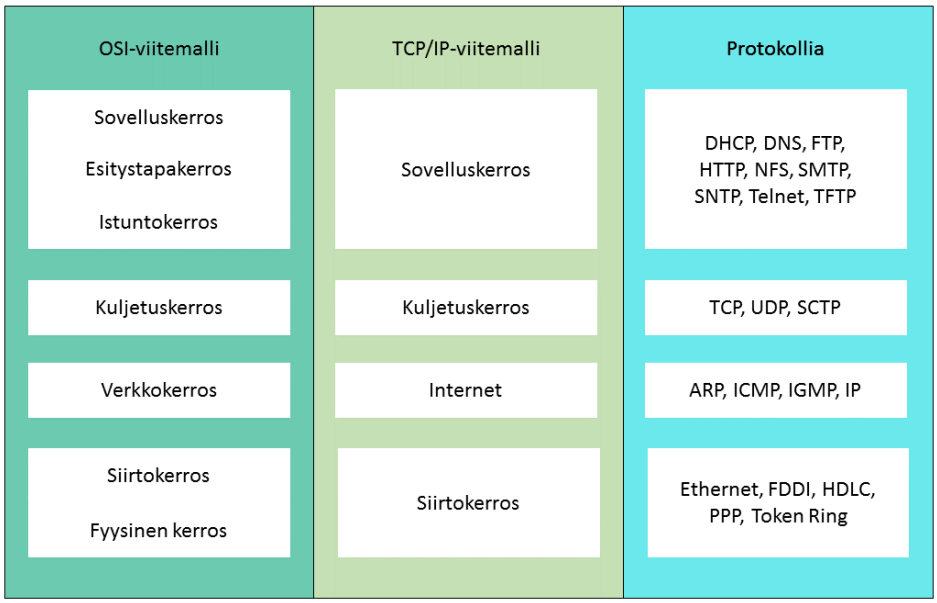
Kuljetuskerroksen pääasialliset protokollat ovat aiemmin esitelty TCP sekä lisäksi UDP. UDP eroaa TCP:stä siinä, että se on yhteydetön protokolla. Myöskään pakettien perille saapumista ei varmisteta ACK-viesteillä. Vastaanottaja ei siis mitenkään voi tietää, mikäli jokin paketti hukkuu matkalla. UDP on kuitenkin paljon nopeampi ja yksinkertaisempi kuin TCP ja sitä käytetäänkin esimerkiksi videoiden suoratoistoon (s*treaming*). Myös UDP-paketissa on otsake, mutta se sisältää vain tiedon porttinumerosta sekä tarkistussumman, jonka avulla vastaanottaja määrittää, onko vastaanotettu paketti koskematon. Mikäli tarkistussumma ei ole oikea, paketti hylätään ja heitetään pois. (Blank 2002).

Ylimpänä TCP/IP-viitemallissa, kuten myös OSI-viitemallissa, on sovelluskerros. Tällä kerroksella tapahtuu käyttäjän vuorovaikutus ohjelman kanssa. Käyttäjä aloittaa prosessin, jossa ohjelma muodostaa yhteyden verkon palveluihin. Yhdessä kuljetuskerroksen protokollien kanssa ohjelma lähettää tietoa halutussa muodossa, käyttämällä jotakin kuljetuskerroksen protokollaa. Sovelluskerroksen vastuulla on lähinnä itse ohjelma ja sen vaatimat prosessit eikä niinkään tiedon kuljetus. (Edwards & Bramante 2009).

2.1.3 Viitemallien erot ja yhtäläisyydet

TCP/IP-viitemalli vastaa OSI-viitemallia, mutta TCP/IP-viitemallin standardit ja prosessit voidaan jakaa neljään kerrokseen seitsemän sijaan. Kuten kuviosta 3 nähdään, TCP/IP-viitemallin sovelluskerros vastaa OSI viitemallin kolmea ylintä kerrosta (sovelluskerros, esitystapakerros sekä istuntokerros). Kummassakin viitemallissa kuljetuskerrokset vastaavat

toisiaan, kuten myös verkko- ja internet-kerros. OSI-viitemallin siirtokerros sekä fyysinen kerros puolestaan vastaavat TCP/IP-viitemallin siirtokerrosta.



Kuvio 3. OSI-viitemallin ja TCP/IP-viitemallin vastaavuudet sekä joitakin eri kerrosten protokollia.

Nykyisin TCP/IP-viitemalli on käytetympi kuin OSI-viitemalli. Teknisiä syitä tähän ovat TCP/IP-viitemallin protokollien suoraviivaisuus (OSI-viitemallin protokollat ovat monimutkaisempia) sekä protokollien nopeus verrattuna OSI-viitemallin protokolliin. Lisäksi OSI-viitemallissa on useampia toteutusmahdollisuuksia ja eri osia voidaan jättää pois, josta seuraa yhteensopivuusongelmia. OSI-viitemallin teknisiä etuja puolestaan ovat OSI-viitemallin tietoturvallisuus (turvallisempi kuin TCP/IP-viitemalli) sekä suuntautuneisuus yhteentoimivuuteen tulevaisuudessa. TCP/IP-viitemallin huonona puolena on myös sen osoite- ja reititysjärjestelmien mahdollisuus ylikuormittua. (Maathuis & Smit 2003).

Taloudellisia syitä TCP/IP-viitemallin puolesta ovat protokollien ilmaisuus (OSI-viitemallissa ne taas maksavat), protokollien käytettävissä olo ajallaan sekä laaja-alaisen teknisen

ammattitaidon olemassaolo (johtuen laajasta käytöstä). OSI-viitemallin taloudellinen etu verrattuna TCP/IP-viitemalliin on ratkaisujen pitkäikäisyys. (Maathuis & Smit 2003).

2.2 Sovelluskerroksen protokollat

2.2.1 Tiedostojen siirtoon ja multimediayhteyksiin käytettävät protokollat

FTP on sovellusprotokolla, jota käytetään tiedostojen siirrossa TCP/IP-pohjaisissa tietokoneverkoissa. Se on standardoitu ensimmäisen kerran vuonna 1971 (RFC-114). FTP ei myöskään ole tietoturvallinen, joten sitä käyttävätkin yleensä anonyymit palvelimet, joilla ei ole korkeita turvallisuusvaatimuksia tiedostojen siirrolle. Mikäli tiedostojen siirtoon tarvitaan salausta, käytetään siihen yleensä SSH-protokollaa käyttävää asiakasohjelmaa. (Dostálek & Kabelová 2006).

TFTP on yksinkertaistettu muoto FTP:stä. Erona TCP-protokollaa käyttävään FTP-protokollaan on, että TFTP käyttää UDP-protokollaa tiedostojen siirtoon. TFTP ei myöskään sisällä kaikkia FTP:n toiminnoista. TFTP noudattaa asiakas-palvelin arkkitehtuuria, jossa TFTP-palvelin allokoi eri portteja tukemaan monia TFTP-asiakkaita millä tahansa hetkellä. Protokollaa käytetään pääasiallisesti yhdessä BOOTP:n kanssa siirtämään konfigurointitiedostoja laitteille, joilla ei ole kovalevyä tallennukseen. TFTP:tä käytetään myös siirtämään tiedostoja tietoverkon laitteiden välillä, kun tarkoituksena on esimerkiksi vikojen etsintä, asetusten muuttaminen ja päivittäminen. Tietoturvallisuus on protokollassa rajoitettua. Tämä tarkoittaa sitä, että järjestelmän ylläpitäjä voi antaa käyttäjälle oikeuden tiettyihin hakemistoihin, mutta näitä istuntoja pitäisi valvoa ja ylläpitää tietoturvan säilyttämiseksi. (Edwards & Bramante 2009).

SIP on tekstipohjainen protokolla, jonka avulla voidaan aloittaa, muokata sekä lopettaa internetin yli tapahtuvia multimediaistuntoja. Tällaisia ovat esimerkiksi ääni- ja videopuheluiden soittaminen sekä pikaviestintä (*Instant Messaging*). SIP sisältää elementtejä HTTP- ja SMTP-protokollista: HTTP-protokollasta on lainattu asiakas-palvelin arkkitehtuuri sekä URL:ien käyttö ja SMTP-protokollasta on lainattu tekstien salaustapa sekä otsakkeiden tyyli. SIP protokollaa käytetään hyväksi erityisesti VoIP-puheluissa. (Johnston 2001).

SIP-protokolla suunniteltiin tekemään kommunikaatioyhteys standardiksi ja avoimeksi kaikille SIP-protokollaa noudattaville järjestelmille, joten siinä on monia tietoturvaheikkouksia. Se on altis esimerkiksi MITM-hyökkäyksille REGISTER-viestin kautta, IP-spoofaukselle, RTP- tai INVITE-tulvahyökkäyksille sekä lisäksi sen autentikointi on heikkoa (käytössä MD5-algoritmi). (Porter ym. 2006).

2.2.2 Etäyhteyteen ja laitteiden valvontaan liittyvät protokollat

NFS-protokolla sallii käyttäjälle pääsyn etätiedostoihin, kuin ne olisivat tallennettuna käyttäjän omalle koneelle. Aiemmin NFS käytti UDP-protokollaa siirtoprotokollana, mutta vuoden 1995 versiosta 3 (NFSv3) lähtien kuljetus on hoidettu käyttämällä TCP-protokollaa. Protokollan avulla käyttäjä voi katsella, varastoida, päivittää sekä hallinnoida tiedostoja etäpalvelimella. Toimiakseen protokolla vaatii NFS-asiakasohjelman käyttäjän koneelle ja NFS-palvelinohjelman etäkoneelle. (Edwards & Bramante 2009). NFS-protokolla ei itsessään sisällä päästä päähän suojausta, joten se ei ole tietoturvallinen. Protokollan yhteydessä voidaan kuitenkin käyttää jotakin kryptografista salausmenetelmää. (RFC-7530).

Telnet on yksi internetin vanhimmista sovellusprotokollista. Ensimmäisenä lyhennettä Telnet käytettiin vuonna 1969 ja se standardoitiin vuonna 1980 (RFC-764). Kolme vuotta myöhemmin, vuonna 1983, standardi korvattiin uudemmalla versiolla (RFC-854). Telnet protokollaa käytetään emuloimaan perinteistä terminaalia (yleensä näppäimistö ja näyttö) TCP/IP-pohjaisissa tietokoneverkoissa. Toisin sanoen Telnet protokollan avulla käyttäjä voi käyttää esimerkiksi palvelinta oman koneensa komentoriviltä ja palvelimelle toiminta näyttäisi samalta kuin siihen olisi fyysisesti kytketty perinteinen terminaali. Telnet itsessään ei ole tietoturvallinen, koska se ei käytä mitään salausta, mutta Telnetiä voidaan käyttää yhdessä SSL/TLS-salauksen kanssa. Tämä ei kuitenkaan ole yleistä vaan yleisemmin käytetään SSH-protokollaa. (Dostálek & Kabelová 2006).

SNMP on protokolla, jota käytetään tietoverkossa olevien laitteiden valvontaan sekä niiden asetusten muuttamiseen. Tämä tapahtuu niin kutsutuilla SNMP-agenteilla. Ne ovat ohjelmia, jotka toimivat yhteistyössä tietoverkon hallinta-aseman (NMS) kanssa. Tukihenkilöstö pystyy valvomaan tietoverkkoa hallinta-aseman kautta ja SNMP-agentti vastaa hallinta-aseman

kyselyihin (*queries*). (Edwards & Bramante 2009). SNMP-protokollasta on kolme pääversiota, joista kahdessa ensimmäisessä on huomattavia tietoturvavikoja. Ne lähettävät datan salaamattomana ja lisäksi niiden rajoitettu autentikointikyky toimii string-muuttujilla. Kolmas versio lieventää näitä ongelmia, mutta se ei välttämättä ole kovin laajasti tuettu. Paras keino tehdä SNMP turvalliseksi on käyttää käyttöoikeuslistoja (*access lists*) sekä jotakin salausta. (Alder ym. 2007).

2.2.3 IP-osoitteiden hallinta, nimipalvelut ja HTTP-protokolla

DHCP on sovellusprotokolla, joka vastaa IP-osoitteiden dynaamisesta jakamisesta verkon laitteille. Protokolla perustuu vanhempiin protokolliin, kuten ARP ja BOOTP. DHCP-palvelin pystyy tarjoamaan muun muassa IP-osoitteet, aliverkon maskit sekä tiedot yhdyskäytävästä (*gateway*). Kun laite yhdistyy tietoverkkoon, DCHP-asiakas pyytää DHCP-palvelimelta tietoa, jonka saatuaan asiakas voi yhdistää tietoverkkoon ja toimia siellä. (Edwards & Bramante 2009).

DHCP-protokollaan kohdistuvat uhkat tulevat yleensä sisäisistä asiakkaista. Tämä johtuu siitä, että DHCP-palvelinta ei pitäisi pystyä saavuttamaan ulkopuolelta, kunhan ulkopuolisia aliverkkoja ei käytetä tai ulkopuolisesta lähteestä tulevia paketteja ei välitetä. Yleisimmät DHCP-protokollaan kohdistuvat tietoturvauhat ovat palvelunestohyökkäykset, validin IP-osoitteen ja konfiguraation luvaton hankkiminen päästäkseen verkkoon sekä MITM-hyökkäys käyttämällä apuna väärennettyä DHCP-palvelinta. (Rooney 2010).

Jokaiselle laitteelle tietoverkossa on siis oma IP-osoite, jolla sen voi tunnistaa ja jonka avulla laitteeseen voi ottaa yhteyden. IP-osoitteiden muistaminen on kuitenkin hankalaa ihmisille, joten jokaiselle IP-osoitteelle on määritelty oma verkkotunnus (*domain name*). Näiden verkkotunnusten ja niitä vastaavien IP-osoitteiden suhteet määritellään maailmanlaajuisessa jaossa olevassa DNS-tietokannassa. Tämä tietokanta sisältää yksittäisiä tietueita, nimeltään RR, jotka sisältävät verkkotunnukset, niiden IP-osoitteet sekä muuta DNS:n kautta jaettua tietoa. DNS-protokolla vastaa pääasiassa tietueiden hakemisesta DNS-tietokannasta, mutta myös esimerkiksi muutoksien tiedottamisesta sekä tietueiden päivittämisestä. DNS-proto-

kolla toimii kysely/vastaus-periaatteella eli asiakas tekee kyselyn palvelimelle, johon palvelin vastaa. DNS-protokolla myös pakkaa DNS-paketit mahdollisimman pieniksi, mutta itse pakettien siirrosta vastaa jokin kuljetusprotokolla. DNS-protokolla voi käyttää sekä UDP:tä että TCP:tä, mutta jokainen kysely/vastaus-pari suoritetaan käyttämällä samaa kuljetusprotokollaa. (Dostálek & Kabelová 2006).

DNS-protokolla on altis tietueiden kyselyihin liittyville hyökkäyksille, kuten spoofaus, ID-otsakkeen arvaus (*ID quessing, query prediction*) ja DNS-myrkytys (*cache poisoning*). Palvelimeen ja konfiguraatioon kohdistuvia hyökkäyksiä ovat muun muassa dynaamisen päivityksen väärinkäyttö, vyöhykkeen (*zone*) siirto, DNS-palvelimen uudelleenkonfigurointi luvattomasti, etähallinnan väärinkäyttö sekä ylivuodon tai käyttöjärjestelmän virheiden hyödyntäminen. Lisäksi DHCP-palvelin on muiden verkon palveluiden tapaan altis palvelunestohyökkäyksille. (Rooney 2010).

HTTP-protokolla on internetin käytetyin protokolla. Se on peräisin vuodelta 1990 ja sen edeltäjä on nykyisin melkein unohduksissa oleva Gopher-protokolla. HTTP-protokollaa käytetään tiedon etsimiseen internetissä. HTTP tukee useita autentikointimenetelmiä (esimerkiksi käyttäjätunnus ja salasana sekä Kerberos), mutta yleisesti salaukseen käytetään SSL- tai TLS-protokollaa. SSL- tai TLS-protokollalla salatusta HTTP-protokollasta käytetään nimitystä HTTPS. Käyttäjien autentikointi voidaan suorittaa joko HTTP-protokollan toimesta tai SSL/TLS-kerroksen toimesta. (Dostálek & Kabelová 2006).

HTTP-protokollassa kommunikointi perustuu asiakas-palvelin arkkitehtuuriin, jossa muodostetaan TCP-yhteys asiakkaan ja palvelimen välille. Kommunikointi asiakaskoneen ja palvelimen välillä käynnistyy, kun käyttäjä kirjoittaa selaimeen URI:n, josta asiakaskone erottaa palvelimen nimen ja kääntää sen DNS:n avulla IP-osoitteeksi. Tämän jälkeen asiakaskone muodostaa TCP-yhteyden saatuun palvelimen IP-osoitteeseen. Selain lähettää HTTP-pyynnön (*HTTP request*) yhteyden läpi ja palvelin lähettää HTTP-vastauksen (*HTTP response*) samaa yhteyttä käyttäen. Selain puolestaan näyttää vastauksen käyttäjälle. Yleensä nettisivut sisältävät monia objekteja, joista jokainen täytyy ladata erillisellä HTTP-pyynnöllä. Protokollan vanhoissa versioissa jokaista objektia varten tarvittiin uusi TCP-yhteys,

josta aiheutui piikki lähetyskanaville (*transmission channels*). Protokollan versiossa 1.1 oletuksena on, että koko nettisivun lataamista varten muodostetaan vain yksi TCP-yhteys. Tällöin asiakkaan ei tarvitse odottaa vastausta jokaiseen pyyntöön vaan pyyntöjä voi tehdä monta peräkkäin. Tätä prosessia kutsutaan putkikuljetukseksi (*pipelining*). (Dostálek & Kabelová 2006).

HTTP-protokollassa on joitakin rajoitteita. Näitä on esimerkiksi keskustelun rajoittuminen yhteen pyyntö/vastaus-sykliin. Tämä on ongelma esimerkiksi verkkokaupoissa, koska kahden eri tavaran lisääminen ostoskoriin anonyyminä tapahtuu kahden eri viestisyklin aikana. Toinen rajoite on itse asiakas/palvelin-arkkitehtuuri, koska se estää palvelinta lähettämästä asynkronisia tapahtumia asiakkaalle. Tämä on ongelma silloin, kun haluttaisiin palvelimen lähettävän tietoa asiakkaalle palvelimen päivittyessä. (Dostálek & Kabelová 2006).

SOAP on yksinkertainen ja kielestä sekä alustasta riippumaton protokolla, joka mahdollistaa palveluiden kommunikoinnin strukturoidun tiedonvaihdon kautta. SOAP-protokollaa käytetään XML-pohjaisten viestien vaihtamiseen verkon yli, yleensä HTTP-protokollaa käyttäen. SOAP-protokollan viestin muoto koostuu niin sanotusta SOAP-kirjekuoresta (*SOAP Envelope*), joka sisältää vapaaehtoisia otsakkeita sekä leipätekstin (*body*). Otsakkeet sisältävät kontekstiin liittyvää tietoa (esimerkiksi tietoturvaan liittyviä tietoja) ja leipäteksti itse hyötykuorman tai sovelluksen dataa. (Balani & Hathi 2009). Alkuperäinen SOAP-protokollan määrittely keskittyi laajennettavuuteen ja tietoturvallisuus oli toissijaista. Tämän vuoksi protokollan tietoturva riippuu sen alla toimivista kuljetusprotokollista ja niiden tietoturvasta. (McGovern ym. 2003).

2.2.4 Sähköpostiprotokollat

SMTP on sähköpostissa käytetty sovellusprotokolla, joka määrää sähköpostien formaatin. Lähetetyt viestit tallennetaan viestijonoon, jota SMTP-asiakas käy läpi säännöllisesti. SMTP-asiakas myös muodostaa yhteyden SMTP-palvelimeen, jolle viesti toimitetaan. SMTP-palvelin hyväksyy sähköpostin ja tarkastaa, onko se tarkoitettu paikalliselle käyttäjälle. Paikallisille käyttäjille tarkoitetut viestit palvelin säilöö suoraan vastaanottajan postilaatikkoon ja muille menevät viestit laitetaan sähköpostijonoon, josta koko prosessi toistuu.

SMTP toimii yhdessä POP3- ja IMAP-protokollien kanssa, jotka vastaavat viestien vastaanottamisesta. (Dostálek & Kabelová 2006).

Sähköposteja voidaan suojata kahdella eri periaatteella, jotka ovat E2E ja P2P. E2E-menetelmässä lähettäjä turvaa sähköpostin koko reitin lähettäjältä vastaanottajalle. Tämä tapahtuu elektronisella allekirjoituksella ja/tai elektronisella kirjekuorella. E2E-menetelmä perustuu S/MIME-protokollaan ja sen ESS-laajennukseen. P2P-menetelmässä puolestaan turvataan reitti loppukäyttäjältä lähimmälle sähköpostipalvelimelle estämällä sähköpostien luvaton tutkiminen sähköpostilaatikossa. Tässä menetelmässä tietoturvallisuus saadaan aikaan SSL/TLS-protokollalla. (Dostálek & Kabelová 2006).

SMTP-protokolla on altis tahallisille ja tahattomille palvelunestohyökkäyksille, koska se ylikuormittuu helposti jo pienestä määrästä sähköposteja. SMTP-protokolla kuitenkin sisältää vastakeinoja palvelunestohyökkäyksille. Esimerkiksi, jos kuormitus on liian suurta, palvelin voi lopettaa sähköpostien vastaanottamisen lähettämällä väliaikaisia virheviestejä tai kieltäytymällä yhteydenmuodostuksesta. Koska SMTP-protokolla sietää viivettä, on tietyn sähköpostin lähetys mahdollista myöhemmin. (Bencsáth & Rónai 2007).

2.3 SSL- ja TLS-protokollat sekä HTTP-protokollan käyttö niiden kanssa

Tässä luvussa perehdytään kuljetuskerroksella käytettäviin SSL- ja TLS- salausprotokolliin sekä HTTP-yhteyden salaamiseen niiden avulla. Luvussa 2.3.1 on protokollien historiaa ja luvuissa 2.3.2 ja 2.3.3 perehdytään itse protokolliin tarkemmin. Luvussa 2.3.4 kerrotaan hieman TLS-salatusta HTTP-yhteydestä eli HTTPS-yhteydestä.

2.3.1 Yleistä

SSL- ja TLS-protokollat toimivat TCP/IP-viitemallin kuljetuskerroksella, jossa niiden tarkoituksena on tehostaa TCP:n suojausta käyttämällä kryptografisia tekniikoita ja tarjoamalla tietoturvapalveluita verkossa tapahtuviin maksuihin. (Oppliger 2009).

SSL-protokolla on Netscape Communications-yhtiön vuonna 1994 kehittämä tietoverkkosalausprotokolla, jonka ensimmäinen versio ilmestyi vain 8 kuukautta ensimmäisen suositun web-selaimen julkaisun jälkeen. SSL-protokollalle myönnettiin patentti vuonna 1997, jonka tarkoituksena oli suojella älyllistä pääomaa muilta vastaavilta projekteilta eikä niinkään rahallisen edun tavoittelu. Tästä syystä patentti onkin luovutettu kaikkien vapaaseen käyttöön ilmaiseksi. SSL-protokollan kolmannen version (SSL 3.0) julkaisun jälkeen tietoturvayhteisö oli sekaisin, koska liikkeellä oli useampi samankaltainen protokolla: SSL sekä Microsoftin PCT- ja STLP-protokollat. Niinpä vuonna 1997 protokollat yhdistettiin ja standardoitiin yhdeksi protokollaksi nimeltä TLS. (Oppliger 2009). Ensimmäinen versio IETF:n TLS-protokollasta julkaistiin vuonna 1999 ja viimeisin versio eli 1.2 on vuodelta 2008 (RFC-5246).

2.3.2 SSL-protokolla

SSL-protokolla on asiakas-palvelin arkkitehtuuriin perustuva tietoturvaprotokolla, joka tarjoaa perustietoturvapalveluita internetissä tapahtuvaan kommunikaatioon. Nämä palvelut liittyvät autentikointiin, yhteyden luottamuksellisuuteen ja yhteyden eheyteen. (Oppliger 2009).

SSL-protokolla koostuu kahdesta kerroksesta, joista alempi toimii kuljetuskerroksen protokollan päällä ja vastaa ylemmän kerroksen datan kapseloinnista. Tämän tekee niin sanottu SSL-tietueprotokolla (*SSL Record Protocol*), joka paloittelee, pakkaa ja kryptografisesti suojaa datan. Ylemmällä kerroksella sijaitsevat neljä aliprotokollaa, jotka ovat kättelyprotokolla (*SSL Handshake Protocol*), salauksen vaihto-protokolla (*SSL Change Cipher Spec Protocol*), hälytysprotokolla (*SSL Alert Protocol*) ja dataprotokolla (*SSL Application Data Protocol*). Kättelyprotokolla vastaa vertaisten (*peers*) autentikoinnista toisilleen, salausmenetelmän neuvottelusta sekä kommunikoinnin pakkausmenetelmästä. Salauksen vaihto-protokolla puolestaan vastaa salausmenetelmän toteutuksesta ja hälytysprotokollan tarkoituksena on välittää virheviestejä sekä tietoa mahdollisista ongelmista. Dataprotokollan tarkoituksena on välittää dataa alemman kerroksen tietueprotokollalle. (Oppliger 2009).

SSL-protokollalla pystytään muodostamaan turvallinen (autentikoitu ja luottamuksellinen) kommunikaatioyhteys, mutta sillä pystytään myös turvallisesti kuljettamaan tietoa lähettäjän ja vastaanottajan välillä. Yhteyden muodostamiseen käytetään SSL-yhteyksiä ja SSL-istuntoja. SSL-istunto määrittää kryptografiset parametrit, joita SSL-yhteydet käyttävät suojaamaan ja pakkaamaan tietoa. SSL-yhteys puolestaan välittää tietoa tyypillisesti asiakkaan ja palvelimen välillä. Yhden SSL-istunnon aikana voidaan muodostaa monta SSL-yhteyttä. Tiedon turvallinen kuljettaminen puolestaan tapahtuu jakamalla tieto palasiin (*fragments*), jotka jokainen käsitellään erikseen. Jokainen palanen pakataan, autentikoidaan MAC:n avulla, salataan, alkuun lisätään otsake ja lopuksi lähetetään vastaanottajalle. Prosessi suoritetaan toisin päin vastaanottajan puolella. (Oppliger 2009).

SSL-protokolla käyttää salainen avain-menetelmää (s*ecret key cryptography*) eli symmetristä salausta viestin autentikoimiseen ja salaukseen. Menetelmässä viestin lähettäjällä ja vastaanottajalla on sama salainen avain, jota voidaan käyttää viestin salaamiseen ja salauksen purkamiseen. SSL-protokolla käyttää vertaisten autentikointiin ja avaimen luomiseen niin sanottua julkisen avaimen salausmenetelmää (*public key cryptography*), jota kutsutaan myös epäsymmetriseksi salaukseksi. Menetelmässä salausavain jaetaan kahteen osaan: yksityiseen ja julkiseen avaimeen. Julkinen avain voidaan turvallisesti jakaa kenelle tahansa ja sen avulla ihmiset voivat salata viestejä ja vahvistaa allekirjoituksia. Yksityinen avain puolestaan on yleensä laitekohtainen, joten sitä ei saa luovuttaa kenellekään. Yksityisellä avaimella on mahdollista purkaa salaus viesteistä ja luoda allekirjoituksia. (St Denis 2006).

SSL-protokollan etuna on, että se on riippumaton sovelluskerroksen protokollista. Se on ainoastaan riippuvainen TCP-protokollasta, jolloin se voi tarjota turvallisuuspalveluita mille tahansa TCP-pohjaiselle protokollalle. (Oppliger 2009).

2.3.3 TLS-protokolla

Kuten luvussa 2.3.1 kerrottiin, TLS-protokolla on SSL-protokollan seuraaja. Tästä syystä protokollan rakenne on identtinen SSL-protokollan kanssa. Tietoturvaprotokolla (tässä siis TLS) on asiakas-palvelin-arkkitehtuurin mukainen ja se toimii jonkin kuljetuskerroksen pro-

tokollan päällä. Myös TLS-protokollassa on kaksi kerrosta, joissa toimivat vastaavat protokollat kuin SSL-protokollassa. Alemmalla kerroksella on tietueprotokolla (*TLS Record Protocol*) ja ylemmällä kerroksella kättelyprotokolla (*TLS Handshake Protocol*), salauksen vaihto-protokolla (*TLS Change Cipher Spec Protocol*), hälytysprotokolla (*TLS Alert Protocol*) ja dataprotokolla (*TLS Application Data Protocol*). Lisäksi myös TLS-protokollalla on erilliset TLS-istunnot ja TLS-yhteydet kuten SSL-protokollalla. (Oppliger 2009).

TLS-protokollaa käytetään useimpien yleisten verkkoprotokollien, kuten HTTP, FTP ja SMTP, salaamiseen. Lisäksi se on osana VoIP-protokollaa sekä VPN-protokollaa. Kaikkein yleisintä TLS-protokollan käyttö on HTTP-protokollan kanssa, jolloin puhutaan HTTPS-protokollasta. (Husák ym. 2015). SSL- ja TLS-protokollat eroavat toisistaan salausavaimien luontitavassa, tuetuissa salausmenetelmissä (*cipher suites*), viestien autentikoinnissa, sertifikaattien hallinnassa sekä hälytysviestien (*alert messages*) tyypeissä (Oppliger 2009).

2.3.4 SSL/TLS-salattu HTTP eli HTTPS

HTTPS-protokolla tarkoittaa HTTP-yhteyttä yhdistettynä SSL/TLS-salaukseen. Kun yhteys salataan SSL/TLS-protokollalla, isäntäkoneiden täytyy ensin sopia salausmenetelmästä ja sen parametreista. (Husák ym. 2015). Tässä tapauksessa kerrotaan TLS-salauksen käyttämisestä, mutta prosessi on samankaltainen SSL-salausta käytettäessä.

HTTPS-asiakas on samanaikaisesti sekä HTTP-asiakas että TLS-asiakas. Se aloittaa yhteyden palvelimen kanssa ja lähettää ClientHello-viestin porttiin 443 aloittaakseen TLS-kättelyn (RFC-2818). Tämä viesti sisältää tiedon käytetystä protokollaversiosta, listan salauspalveluista (*cipher suite list*) ja laajennuksista. Seuraavissa viesteissä autentikoidaan vertaiset X.509 sertifikaateilla ja muodostetaan salattu yhteys annettujen parametrien perusteella. Osa viesteistä on salaamattomia ennen kuin jaetut avaimet on julkaistu ja vahvistettu Finished-viesteillä. (Husák ym. 2015). Tämän jälkeen kättely on ohitse ja asiakas voi alkaa tekemään HTTP-pyyntöjä. Kaikki viestit HTTPS-yhteyden yli lähetetään TLS-sovellustiedostoina (*TLS application data*). TLS tarjoaa myös yhteyden turvallisen sulkemisen, joka varmistaa, että tietoa ei enää lähetetä yhteyden yli. (RFC-2818).

3 DoS- ja DDoS-hyökkäykset

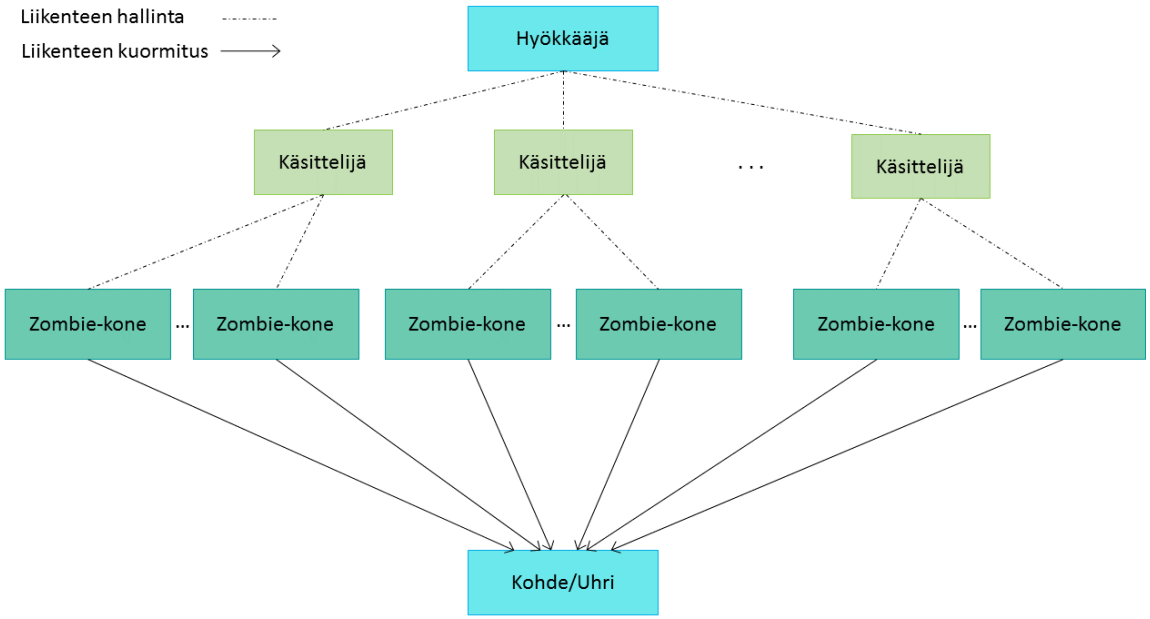
Tässä luvussa perehdytään tarkemmin palvelunestohyökkäyksiin, mitä ne ovat, miten niitä tehdään ja millaisia vaikuttimia hyökkäysten takaa löytyy. Luvuissa 3.1 ja 3.2 esitellään yleisesti palvelunestohyökkäyksiä ja jaotellaan hyökkäyksiä eri tyyppeihin. Luvussa 3.3. esitellään lyhyesti netistä löytyviä työkaluja, joita voidaan käyttää palvelunestohyökkäysten tekemiseen. Luvussa 3.4 puolestaan läpikäydään hyökkäyksen mahdollisia motiiveja ja luvussa 3.5 on esimerkkejä viimeaikaisista palvelunestohyökkäyksistä.

3.1 Yleistä

Palvelunestohyökkäys eli DoS-hyökkäys tapahtuu internetissä ja sen tarkoituksena on lamaannuttaa tietoverkon kyky tarjota palveluita normaalisti. Tarkoituksena on häiritä palveluita rajoittamalla tavallisten asiakkaiden pääsyä esimerkiksi palvelimelle. Yleensä tämä saavutetaan joko hyökkäämällä verkon kaistanleveyttä tai yhdistettävyyttä vastaan. Kaistanleveyttä vastaan tehdyt hyökkäykset kuormittavat tietoverkkoa aiheuttamalla niin paljon liikennettä, ettei tavalliselle käyttäjälle jää resursseja käytettäväksi. Yhdistettävyyttä vastaan aiheutetut hyökkäykset puolestaan toteutetaan lähettämällä palvelulle niin paljon yhteyspyyntöjä, että palvelu ei pysty enää vastaamaan tavallisen käyttäjän yhteydenottoihin. (Douligeris & Aikaterini 2003).

Hajautettu palvelunestohyökkäys eli DDoS-hyökkäys vastaa DoS-hyökkäystä, mutta siinä palveluiden normaalia käyttöä estetään hyökkäämällä samanaikaisesti monesta eri lähteestä vain yhden lähteen sijaan. Tällaisten hyökkäysten estäminen ja lieventäminen on tavallista DoS-hyökkäystä vaikeampaa, koska hyökkäyksen lähdettä on vaikea erottaa todellisesta käyttäjästä. Hyökkäyksen vaikutukset ovat myös suhteessa vakavampia kuin tavallisessa palvelunestohyökkäyksessä, koska hyökkäyksen suuri volyymi saattaa aiheuttaa esimerkiksi järjestelmän sammumista, tiedostojen tuhoutumista sekä osittaista tai kokonaista palveluiden menettämistä. DDoS-hyökkäys käyttää hyväkseen internetin järjestelmän rakennetta, joka perustuu siihen, että palvelut voi saavuttaa kuka tahansa ilman rajoitteita. (Douligeris & Aikaterini 2003).

Kuviossa 4 näkyy hajautetun palvelunestohyökkäyksen arkkitehtuuri. Ylimpänä on hyökkääjä, joka organisoi koko hyökkäyksen ja hallitsee käsittelijöiden (*handlers/masters*) liikennettä. Käsittelijät ovat hyökkääjän saastuttamia koneita, joissa on erityinen ohjelma pyörimässä ja jotka pystyvät hallitsemaan kerralla useita zombie-koneita eli botteja (*attack daemon agents/zombie hosts/bot*). Zombie-kone on käsittelijän saastuttama kone, jossa myös pyörii erityinen ohjelma. Zombie-koneet luovat ja lähettävät paketteja, joilla ne kuormittavat uhria.



Kuvio 4. DDoS-hyökkäyksen arkkitehtuuri.

Hyökkäyksen edetessä hyökkääjä valitsee jotain skannaintyökalua käyttämällä ne zombie-koneet, jotka suorittavat hyökkäyksen. Zombie-koneissa täytyy olla jokin haavoittuvuus, jonka avulla hyökkääjä pystyy pääsemään koneeseen käsiksi. Lisäksi zombie-koneella täytyy olla riittävästi resursseja hyökkäyksen toteuttamiseen. Koneet valittuaan hyökkääjä piilottaa hyökkäyskoodin jokaiseen zombie-koneeseen. Tarkoituksena on, että zombie-koneen käyttäjä ei huomaa olevansa osa palvelunestohyökkäystä. Tämä pyritään varmistamaan piilottamalla koodi sekä käyttämällä mahdollisimman vähän zombie-koneen resursseja eli muistia ja kaistanleveyttä. Hyökkääjä kommunikoi käsittelijöiden kanssa ajoittaakseen

hyökkäyksen sekä selvittääkseen toiminnassa olevat zombie-koneet ja niiden päivitystarpeen. Kaikkeen kommunikointiin voidaan käyttää joko TCP-, UDP- tai ICMP-protokollaa. Itse hyökkäyksen aikana hyökkääjä voi hallita hyökkäyksen yksityiskohtia (esimerkiksi kohde tai hyökkäyksen kesto) ja ominaisuuksia (esimerkiksi tyyppiä, pituutta tai porttinumeroa). (Douligeris & Aikaterini 2003).

Hajautettu palvelunestohyökkäys voi tapahtua joko verkkokerroksella, kuljetuskerroksella tai sovelluskerroksella. Lisäksi se voidaan toteuttaa monen eri protokollan haavoittuvuuksia hyväksikäyttämällä. Sovelluskerroksella tapahtuvien hyökkäysten määrä on kasvussa, koska perinteiset palvelunestohyökkäyksiä torjuvat työkalut eivät juurikaan valvo sovelluskerroksen liikennettä. Tämä tarkoittaa sitä, että sovelluskerroksella palvelunestohyökkäyksellä on vastassaan vähemmän suojauskerroksia verrattuna verkko- ja kuljetuskerrokseen. Tästä seuraa, että sovelluskerroksella palvelunestohyökkäykset onnistuvat todennäköisemmin kuin muilla kerroksilla. (Stevanovic & Vlajic 2014).

3.2 DDoS-hyökkäystyyppejä

DDoS-hyökkäykset jaetaan tässä tutkielmassa tulvahyökkäyksiin, hitaisiin hyökkäyksiin, nimipalveluihin, IP-osoitteisiin ja sähköpostiin liittyviin hyökkäyksiin, web-palveluiden datansiirtoon liittyviin hyökkäyksiin sekä muihin protokolliin liittyviin hyökkäyksiin. Suurimmalta osin tässä tutkielmassa keskitytään sovelluskerroksen protokolliin kohdistuviin hyökkäyksiin, mutta lyhyesti esitellään myös esimerkiksi kuljetuskerroksella tapahtuvia hyökkäyksiä. Tyypillisesti edistynyt palvelunestohyökkäys nimittäin sisältää useita erilaisia hyökkäyksiä sekä eri kerroksilla tapahtuvia hyökkäyksiä (moniulotteinen hyökkäys), koska niiden torjunta on hankalampaa kuin yhdellä tavalla tapahtuvien hyökkäysten torjuminen.

3.2.1 Tulvahyökkäykset

Tulvahyökkäykset ovat palvelunestohyökkäyksiä, joissa uhria kuormitetaan suurella määrällä paketteja. Tämä pakettien tulva aiheuttaa sen, että aidoille käyttäjille ei riitä yhteyksiä ja palvelu on saavuttamattomissa. Tulvahyökkäykset ovat yksinkertaisia toteuttaa ja niillä voi aiheuttaa palvelunestohyökkäyksen useisiin eri sovelluskerroksen protokollilla toimiviin

palvelimiin. Tulvahyökkäykset voidaan jakaa kahteen tyyppiin, jotka ovat istuntotulva (*session flooding*) ja pyyntötulva (*request flooding*). Istuntotulvassa lähetetään paljon istuntoon liittyviä yhteyspyyntöjä, kun taas pyyntötulvassa lähetetään suuria määriä muita pyyntöjä (Yadav & Selvakumar 2015). Tulvahyökkäyksiä ovat kuljetusprotokollia hyväksikäyttävät UDP-tulva ja SYN-tulva, ping-protokollaa hyväksikäyttävä ICMP-tulva sekä sovelluskerroksella tapahtuva HTTP-tulva. Näiden lisäksi tulvahyökkäyksiin lukeutuvat ns. heijastetut hyökkäykset, jotka voidaan toteuttaa usean eri protokollan avulla.

UDP-tulva (*UDP Flood*) on UDP-kaikuun (*echo*) perustuva hyökkäys, jossa hyökkääjä väärentää UDP-paketteja. Näiden väärennettyjen pakettien avulla hyökkääjä yhdistää yhden koneen kaiutuspalvelun (*echo service*) toisen koneen CHARGEN-protokollaan. Tästä seuraa, että nämä kaksi palvelua käyttävät kaiken kaistanleveytensä yrittäessään vaihtaa merkkejä itsensä kanssa. (Lau ym. 2000).

SYN-tulva (*SYN Flood/TCP SYN Flood*) hyväksikäyttää TCP-protokollan kolmivaiheista kättelyä, jossa vaihdetaan kolme pakettia ennekuin asiakas voi käyttää palvelua. Palvelin lähettää SYN/ACK-paketin asiakkaalle vastaanotettuaan tältä yhteydenottopyynnön SYN-paketissa. Tämän jälkeen palvelin jää odottamaan asiakkaalta viimeistä ACK-pakettia, jotta yhteys voidaan muodostaa. SYN-tulvassa tämä ACK-paketti jätetään kuitenkin lähettämättä, jolloin palvelin jää turhaan odottamaan. Kun yhteydenavauspyyntöjä tulee tarpeeksi, palvelimen uusien yhteyksien puskurijono täyttyy, jonka jälkeen palvelin ei voi enää käsitellä uusia yhteydenavauspyyntöjä. (Lau ym. 2000).

ICMP-tulva (*ICMP Flood, Ping Flood, Smurf Attack*) on UDP-tulvan kaltainen hyökkäys, mutta ICMP kaiutus-paketeilla toteutettuna. ICMP-tulvassa hyökkääjä lähettää suuren määrän ICMP-kaiutuspyyntöjä tietoverkon lähetysosoitteisiin (*broadcast address*), jotka puolestaan välittävät pyynnön kaikille kyseisessä verkossa oleville koneille. Lähetysosoitteita kutsutaan vahvistimiksi (*amplifiers*), koska ne tavallaan vahvistavat alkuperäistä pyyntöä monistamalla sen kaikille verkon koneille. Kaiutuspyyntöihin laitetaan lähettäjäksi uhrina olevan osoite, jolloin kaikki kaiutuspyynnön saavat koneet alkavat lähettää vastauspaketteja uhrille. ICMP-tulvasta kärsivät sekä uhri että välikätenä toimivat vahvistimet. Hyökkäys saattaa myös aiheuttaa koko tietoverkon ylikuormittumisen. (Lau ym. 2000).

HTTP-tulvahyökkäyksessä (*HTTP Flood*) uhrin web-palvelin kaadetaan lähettämällä sille suuri määrä aidonoloisia HTTP GET-pyyntöjä. Uhrina oleva palvelin ei pysty erottamaa haitallisia HTTP GET-pyyntöjä aidoista, vaan joutuu vastaamaan jokaiseen pyyntöön. Tämä johtaa palvelimen kuormittumiseen, jolloin oikeat pyynnöt eivät pääse enää läpi. Hyökkäys voidaan toteuttaa esimerkiksi virusten, bottien tai DoS-työkalujen avulla. Erityistapaus HTTP-tulvasta on niin kutsuttu F5-hyökkäys, jossa hyökkääjä kuormittaa palvelimen käyttämällä selaimen uudelleenlataustoimintoa (*reload*). Uudelleenlatauksen pikanäppäin on F5, joten siitä tulee hyökkäyksen nimi. (Yagatai & Isohara & Sasase 2007).

Heijastettu hyökkäys (*Reflected attack*) eli DRDoS on tulvahyökkäys, jossa tulva aiheutetaan lähettämällä väärennettyjä pyyntöjä heijastinkoneiden tai heijastinpalveluiden (*reflectors*) kautta. Hyökkääjä vaihtaa pyyntöjen lähettäjäksi uhrin IP-osoitteen, jolloin kaikki vastauspaketit ohjautuvat uhrille. ICMP-tulva on eräs heijastetun hyökkäyksen muoto, mutta heijastettuja hyökkäyksiä voi aiheuttaa muillakin protokollilla, kuten IP-, TCP-, UDP-, DNS-, SNMP- ja HTTP-protokollilla. (Paxson 2001).

3.2.2 Hitaat hyökkäykset

Hitaat HTTP-hyökkäykset (*Slow HTTP attack*) hyväksikäyttävät sovelluskerroksen HTTP-protokollaa. Mikäli pyyntö ei ole kokonainen tai siirto tapahtuu todella hitaasti, palvelimen resurssit ehtyvät sen jäädessä odottamaan loppuja tietoja. Hitaat HTTP-hyökkäykset voidaan jakaa tekniikaltaan kahteen kategoriaan: ensimmäinen tekniikka on lähettää HTTP-pyynnöt hitaasti ja toinen tekniikka on lukea HTTP-vastaus hitaasti. Ensimmäistä tyyppiä ovat SlowLoris- ja hidas HTTP POST-hyökkäys. Toista tyyppiä puolestaan edustaa Slow Read-hyökkäys. (Park ym. 2014).

SlowLoris-hyökkäys (*Slow headers, Slow HTTP GET*) pyrkii estämään pääsyn web-palvelimelle tai web-sovelluksen palomuurille (WAF) käyttämällä kaikki vapaana olevat yhteydet (soketit) kohdepalvelimella. Tarkoituksena on ylläpitää HTTP-yhteys kohdepalvelimeen yhdistettynä mahdollisimman kauan, kunnes juuri ennen yhteyden aikakatkaisua yhteys katkaistaan hetkeksi ja muodostetaan uusi yhteys samaan koneeseen. Yhteyden katkaisun tarkoituksena on saada liikenne näyttämään aidolta palvelimelle. Oikeat käyttäjät eivät pääse

muodostamaan yhteyttä palvelimelle, koska hyökkääjän käytössä ovat kaikki palvelimen yhteydet. Jotta yhteys pysyisi päällä, hyökkäyksen aikana lähetetään palvelimelle osittaisia otsakkeita sekä tekaistua dataa. (Moustis & Kotzanikolaou 2013).

SlowLoris-hyökkäys voidaan toteuttaa myös käyttämällä PRAGMA-otsaketta, jota selain käyttää kertoakseen ohjelmalle ja välimuisteille, että se haluaa uuden version aiemmin pyydetystä resurssista. Tällöin yhteyden aikakatkaisu nollaantuu ja yhteys pysyy pidempään auki. (Dantas, Nigam & Fonseca 2014).

SlowLoris-hyökkäyksen tapaan hidas HTTP POST-hyökkäys perustuu HTTP-protokollan tapaan odottaa, kunnes pyyntö on kokonainen. Erona näiden kahden välillä on, että HTTP POST-hyökkäys käyttää hyväkseen POST-metodia ja SlowLoris käyttää hyväkseen GET-metodia ja otsakkeita. (Park ym. 2014).

SlowRead-hyökkäyksessä hyökkääjä hyväksikäyttää TCP-protokollan vuonvalvontaa lähettämällä ensin aidon pyynnön kolminkertaisen kättelyn suoritettuaan. Tämän jälkeen hyökkääjä ilmoittaa ikkunan koon olevan normaalia pienempi, pakottaen näin HTTP-vastausoperaation hidastamaan nopeuttaan. Kun hyökkääjä ilmoittaa ikkunan kooksi nolla, palvelin ei enää lähetä tietoja, mutta pitää kuitenkin yhteyden auki. (Park ym. 2014).

RUDY-hyökkäys käyttää myös hyväkseen HTTP-protokollan haavoittuvuuksia. Normaalisti HTTP POST-pyynnön yhteydessä lähetetään yksi tai kaksi datapakettia palvelimelle, jonka jälkeen palvelin sulkee yhteyden ja siirtyy palvelemaan seuraavaa käyttäjää. RUDY-hyökkäyksessä datapaketit on rikottu useiksi paketeiksi, joissa jokaisessa on vain yhden tavun verran dataa. Nämä paketit lähetetään satunnaisilla aikaväleillä, jolloin palvelin ei voi sulkea yhteyttä vaan se odottaa, kunnes pyyntö on kokonainen. Hyökkäyksen aikana luodaan tuhansia pyyntöjä usean minuutin aikana, jolloin palvelin ei enää pysty käsittelemään oikeita pyyntöjä. RUDY-hyökkäyksen voi tehdä sellaisiin web-palveluihin, jotka sisältävät esimerkiksi sisäänkirjautumista, palautelomakkeita tai muita POST-menetelmää käyttäviä palveluita. (Shetty & Prasad & Nalini 2015).

3.2.3 DNS-nimipalveluihin, IP-osoitteiden jakoon ja sähköpostipalvelimiin liittyvät hyökkäykset

DNS-palvelimet vastaavat DNS-kyselyihin etsimällä nimitietoja välimuistista. Välimuisti koostuu joukosta nimitietoja, joilla jokaisella on välimuistista poistamisesta kertova TTL-arvo (*Time-To-Live*). Pyynnön saadessaan DNS-palvelin, etsii välimuististaan pyyntöä vastaavaa nimitietoa. Jos sellainen löytyy, palvelin vastaa kyselyyn antamalla löydetyt tiedot. Mikäli tietoa ei löydy välimuistista, palvelin etsii aluehierarkiasta lähimmän alueen, joka sisältää kysellyn tiedon, ja tallentaa tämän välimuistiin. Etsintä päättyy siihen, että oikea alue löytyy vastaamaan kyselyyn tai tapahtuu jokin virhe (esimerkiksi relevanteilta alueilta ei saada vastausta). (Li & Chen & Lei 2010).

DNS-palvelimeen kohdistuvat palvelunestohyökkäykset häiritsevät palvelimen nimipalveluita, mukaan lukien kaikki vyöhykkeelle ja alivyöhykkeille kuuluvat verkkonimet. Kaikki hyökkäyksen kohteena olevaa palvelinta välittävät DNS-palvelimet eivät myöskään pysty vastaamaan nimikyselyihin, joista ei ole välimuistikopiota. Ne eivät myöskään pysty selvittämään verkkonimiä, jos tiedot häviävät yhteyden aikakatkaisun vuoksi. Näin ollen yhden DNS-palvelimen toimimattomuus johtaa siihen yhteydessä olevien DNS-palvelimien toimimattomuuteen. (Li & Chen & Lei 2010).

DNS-vahvistettu hyökkäys (*DNS Reflection Amplification attack*) on palvelunestohyökkäys, jossa uhrin resurssit, kuten CPU, kaistanleveys tai muisti, ehdytetään. Hyökkäys tapahtuu ohjaamalla DNS-palvelimen liikenne uhrin IP-osoitteeseen. Hyökkäys alkaa suhteellisen pienillä pyynnöillä palvelimelle, mutta hyökkäyksen edetessä vastauspakettien kokoa kasvatetaan tekemällä esimerkiksi ANY-, NS- tai RRSIG-kyselyitä palvelimelle. Hyökkäystä helpottaa DNS-palvelimien käyttämä UDP-protokolla, joka ei vaadi asiakkaan ja palvelimen välille kättelyitä. DNS-vahvistettuja hyökkäyksiä on kolmea tyyppiä: ensimmäisenä ovat toistuvat kyselyt, joissa samaan kyselyyn vastataan monta kertaa. Seuraavana ovat vaihtelevat kyselyt, joissa kysellään erilaisia domain-verkkotunnuksia. Kolmantena on hajautettu hyökkäys, jossa käytetään useita eri DNS-palvelimia ja useita alue-tiedostoja. (Jose & Binu 2014).

DNS-tunnelointi (*DNS tunneling*) on hyökkäystekniikka, jossa tietoa piilotetaan DNS-pyyntöjen ja DNS-vastausten sisälle. Tämä tapahtuu käyttämällä käyttäjän koneella kustomoitua asiakasta ja salaista DNS-palvelinta kohteena olevalla toimialueella (*domain*) sijaitsevan yrityksen ulkopuolella. Tunneloinnin periaate on helppo: tunnelointityökalu sekä sulauttaa dataa DNS-kyselyyn että kuljettaa DNS-pyyntöjä ja DNS-vastauksia tunneloidun asiakkaan ja etäällä olevan DNS-hyökkäyspalvelimen välillä. Tämän jälkeen hyökkäyspalvelin lähettää saadun datan oikeaan kohteeseen DNS-dataan piilotettuna. (Aiello & Mongelli & Papaleo 2013).

DHCP-ehdytyshyökkäyksellä (*DHCP starvation attack*) estetään IP-osoitteiden jako todellisille käyttäjille. Hyökkäys tapahtuu lähettämällä DHCP-palvelimelle DHCPDISCOVER-viestejä väärennettyjä MAC-osoitteita käyttäen. Tästä seuraa, että käytettävissä olevat IP-osoitteet loppuvat kesken, jolloin todellisille käyttäjille ei pystytä enää jakamaan IP-osoitteita. (Yaibuates & Chaisricharoen 2014).

SMTP-palvelimiin kohdistuvat hyökkäykset tapahtuvat kuormittamalla palvelinta tuhansilla samanaikaisesti lähetetyillä sähköposteilla. Tämä voi tapahtua myös tahattomasti silloin, kun käyttäjät lähettävät liian monta sähköpostia samanaikaisesti (esimerkiksi sähköisiä uutiskirjeitä). Myös esimerkiksi spämmääjät voivat vahingossa aiheuttaa palvelunestohyökkäyksen kerätessään sähköpostiosoitteita tiedostonkeräyshyökkäyksellä (*Directory Harvest Attack*). Sähköpostipalvelimeen kohdistuvasta hyökkäyksestä aiheutuu tuntien viivettä sähköpostien toimittamiseen ja lisäksi koko palvelin saattaa kaatua. (Bencsáth & Rónai 2007).

3.2.4 SOAP-protokollan ja XML:n avulla tapahtuvaan datansiirtoon liittyvät hyökkäykset

XML-hyökkäys kohdistuu sellaisiin palvelimiin, jotka käyttävät XML-dokumentteja kommunikointiin (esimerkiksi SOAP-protokollaa käyttävät palvelimet). Hyökkääjä tekee väärin muotoillun XML-pyynnön palvelimelle, jolloin palvelin kuormittuu yrittäessään käsitellä pyyntöä. Lähetetty XML-dokumentti voi olla esimerkiksi liian suuri (useita megatavuja) ja lisäksi se voi sisältää liian suurikokoisia elementtejä, attribuutteja tai nimiavaruuksia (*namespaces*). Hyökkäys voi myös hyödyntää prosessorin (CPU) jäsentelyä, jota kuormitetaan

esimerkiksi jatkuvilla avoimilla tageilla, sisäkkäisillä XML-rakenteilla tai käyttämällä monia määreitä (*declarations*) nimiavaruuksille. (Vissers ym. 2014).

SOAP-protokollan haavoittuvuuksiin liittyviä hyökkäyksiä ovat ylisuuriin salauksiin (*oversized encryption*) perustuvat hyökkäykset sekä WS-osoitteistuksen väärentäminen (*Web services addressing spoofing*). Ylisuurta salausta voidaan käyttää hyökkäykseen, koska SOAP-protokollassa saatetaan sijoittaa koko pyyntö muistiin salauksen purkamista ja allekirjoituksen vahvistamista varten (esimerkiksi Axis2:n turvallisuusmoduuli Rampart tekee näin). Tästä aiheutuu herkästi muistin ehtyminen. WS-osoitteistuksen väärentämisessä hyökkääjä puolestaan käyttää hyväkseen SOAP-protokollan otsakkeessa olevia ReplyTo- ja FaultTo-attribuutteja. Niitä muokataan siten, että palvelin lähettää vastaus- tai virheviestejä uhrin koneelle hyökkääjän sijasta. Kyse on näin ollen heijastetusta (*reflected*) hyökkäyksestä. (Vissers ym. 2014).

3.2.5 Muut protokollien haavoittuvuuksia hyväksikäyttävät hyökkäykset

Asymmetrinen hyökkäys (*asymmetric attack*) tarkoittaa yleisesti sellaista palvelunestohyökkäystä, jossa kohdetta kuormitetaan lähettämällä paljon suurta työmäärää vaativia pyyntöjä. Tällaisia pyyntöjä ovat esimerkiksi tietokantapyynnöt tai monimutkaiset scriptipyynnöt. Pyyntöjen määrän ei asymmetrisessä hyökkäyksessä tarvitse erota normaalin käyttäjän lähettämien pyyntöjen määrästä, koska yhden pyynnön kuormittavuus on niin suuri. (Chuan ym. 2014).

Teardrop-hyökkäys käyttää hyväkseen haavoittuvuutta paloiteltujen TCP/IP-pakettien uudelleenkokoamisprosessissa. Hyökkääjä lähettää liian suuren tai päällekkäisen, paloitellun TCP/IP-paketin uhrille. Kun uhrin kone yrittää kasata TCP/IP-paketin takaisin kokoon, kone kaatuu ja kone on pakotettu uudelleenkäynnistymään. (Raghavan & Dawson 2011).

POD-hyökkäys (*Ping of Death*) on ICMP-protokollaan perustuva palvelunestohyökkäys, jossa hyökkääjä lähettää viallisia ping-paketteja uhrille. Hyökkäyksessä käytettävät ping-paketit ovat kooltaan liian suuria (65 353 tavua), kun normaalin ping-paketin koko on vain 64 tavua. Kun uhrin kone vastaanottaa ylisuuren ping-paketin, se kaatuu. Hyökkäys onnistui

aikanaan useimmissa käyttöjärjestelmissä, mutta ongelma on korjattu vuoden 1997 jälkeen julkaistuihin versioihin. (Raghavan & Dawson 2011).

LAND-hyökkäyksessä hyökkäyksen kohteena on käyttöjärjestelmän TCP/IP-pino. Hyökkääjä lähettää väärennetyn TCP SYN-paketin, jonka lähettäjän ja kohdekoneen IP-osoite on sama. Tämä aiheuttaa uhrin koneessa sen, että kone yrittää keskustella itsensä kanssa, kunnes se lopulta kaatuu. (Raghavan & Dawson 2011).

P2P-hyökkäys (*Peer-to-Peer attack, index poisoning attack*) on palvelunestohyökkäys, joka tapahtuu vertais- eli P2P-verkossa. P2P-verkossa jokaisella vertaisella (*peer*) on taulukko tiedosto-osoittimista, jotka sisältävät tunnistetiedot kyseisen tiedoston mahdollisista jakajista. Hyökkääjä voi kuitenkin levittää väärennettyjä tiedosto-osoittimia suojaamattomille vertaisille, jolloin verkon liikenne ohjautuu kohteena olevalle uhrille. Tämä uhri voi olla joko P2P-verkkoon kuuluva tai kokonaan sen ulkopuolella. (Lou & Hwang & Hu 2009).

RangeAttack-hyökkäys (*Apache Range Header Attack*) on Apache-palvelimiin kohdistuva hyökkäys, joka tapahtuu lähettämällä palvelimelle suuri, joukoista koostuva otsake (*range header*). Tästä seuraa, että palvelin yrittää tehdä useita kopioita pyydetystä palvelusta. Tämä syö muistin resursseja, jonka johdosta palvelin alkaa sivuttaa tietoa (siirtää dataa swap –alueelle, *swapping*) ja lopulta kaatuu. (InfosecStuff 2011).

0Day-hyökkäys (*Zero-day attack*) tarkoittaa sellaista hyökkäystä, joka on vielä tuntematon ja moniulotteisella hyökkäyksellä (*Multi-vector attack*) puolestaan tarkoitetaan sellaista hyökkäystä, jossa käytetään samanaikaisesti useita eri hyökkäystekniikoita.

3.3 Hyökkäykseen käytettäviä työkaluja

Internet on täynnä erilaisia hajautettujen palvelunestohyökkäysten tekemiseen tarkoitettuja ohjelmia ja työkaluja, jotka on helppo asiasta kiinnostuneen löytää. Tähän on koottu esittelymielessä joitakin yleisesti tunnettuja DDoS-työkaluja. Jokaisesta työkalusta on lyhyt kuvaus sekä linkki, josta ohjelmiston on voinut tutkielman tekohetkellä löytää.

DAVOSET on komentorivityökalu, jolla voidaan suorittaa hajautettuja palvelunestohyökkäyksiä käyttämällä hyväksi vastapuolen toiminnallisuuden (*Abuse of Functionality*) ja

XML-entiteettien (*XML External Entities*) haavoittuvuuksia. Sellaisia sivustoja, jotka sallivat pyyntöjen tekemisen toisille sivustoille (toiminnallisuuden haavoittuvuus), voidaan käyttää hyväksi DoS-hyökkäysten tekemisessä työkalun avulla. Työkalun voi löytää täältä: https://github.com/MustLive/DAVOSET. (Viitattu 25.5.2016).

DDOSIM on sovelluskerroksen palvelunestohyökkäyssimulaattori, joka simuloi useita Zombie-koneita satunnaisilla IP-osoitteilla. Nämä Zombie-koneet muodostavat kohdepalvelimelle TCP-yhteyden ja tämän jälkeen DDOSIM aloittaa keskustelun esimerkiksi HTTP-palvelimen kanssa. Se sisältää seuraavat toiminnallisuudet: HTTP DDoS käyttäen aitoja tai väärennettyjä pyyntöjä, SMTP DDoS sekä TCP-yhteystulva satunnaiseen porttiin. Simulaattori on löydettävissä täältä: https://sourceforge.net/projects/ddosim/. (Viitattu 25.5.2016).

GoldenEye on python-sovelluksena toimiva HTTP-palvelunestohyökkäyksen simulointityökalu. Se käyttää hyväkseen HTTP-protokollan yhteyden ylläpidosta huolehtivaa Keep-Alive-otsaketta sekä välimuistikopiot estävää no-cache-arvoa otsakkeessa. Sovellus löytyy osoitteesta https://github.com/jseidl/GoldenEye. (Viitattu 25.5.2016).

HOIC (*High Orbit Ion Cannon*) on avoimen lähdekoodin palvelunestohyökkäystyökalu, jolla pystyy hyökkäämään yhtäaikaisesti jopa 256 eri sivustolle. Hyökkäysmenetelmänä on monisäikeinen HTTP-tulva. Työkalu löytyy osoitteesta https://sourceforge.net/projects/highorbitioncannon/. (Viitattu 25.5.2016).

HULK (*HTTP Unbearable Load King*) on python-sovellus, jolla voidaan simuloida palvelunestohyökkäystä HTTP-protokollaa hyväksikäyttäen. Ohjelma toimii siten, että se generoi peräkkäisiä, uniikkeja HTTP-pyyntöjä vaikuttaen näin suoraan palvelimen itsensä kuormitukseen. Työkalu löytyy osoitteesta http://www.sectorix.com/2012/05/17/hulk-web-server-dos-tool/. (Viitattu 25.5.2016).

LOIC (*Low Orbit Ion Cannon*) on avoimen lähdekoodin työkalu, joka generoi vääriä UDP-, TCP- tai HTTP GET-paketteja. Työkalu on löydettävissä täältä: https://sourceforge.net/projects/loic/. (Viitattu 25.5.2016).

OWASP Switchblade on työkalu, jolla on tarkoitus testata ohjelmistojen saavutettavuutta, suorituskykyä sekä kapasiteettia. Se tarjoaa 3 palvelunestohyökkäystyyppiä, joita voidaan

testata: puolikkaita SSL-yhteyksiä avaava hyökkäys, HTTP POST-hyökkäys sekä SlowLoris-hyökkäys. Työkalu on saatavilla täällä: https://www.owasp.org/index.php/OWASP_HTTP_Post_Tool. (Viitattu 25.5.2016).

PyLoris on työkalu, jolla voidaan testata palvelinta yhteyksien ehtymistä aiheuttavien palvelunestohyökkäyksien varalta. Se pystyy käyttämään SOCKS-välityspalvelimia ja SSL-yhteyksiä ja käyttämään hyökkäykseen HTTP-, FTP-, SMTP-, IMAP- ja Telnet-protokollia. Työkalu löytyy täältä: https://sourceforge.net/projects/pyloris/. (Viitattu 25.5.2016).

Tor's Hammer on pythonilla toteutettu hitaita hyökkäyksiä simuloiva testaustyökalu. Menetelmänä se käyttää Slow POST-hyökkäystä. Työkalu on löydettävissä täältä: https://sourceforge.net/projects/torshammer/. (Viitattu 25.5.2016).

XOIC on DoS-hyökkäysten tekemiseen tarkoitettu työkalu, jolla voi simuloida HTTP-, TCP-, UDP- ja ICMP-protokollaa hyväksikäyttäviä hyökkäyksiä. Työkalu löytyy osoitteesta https://sourceforge.net/projects/xoic/. (Viitattu 25.5.2016).

3.4 Hyökkäyksen motiivit

DDoS-hyökkäysten tarkoitus ei yleensä ole aiheuttaa vahinkoa itse järjestelmälle tai rikkoa sitä, vaan tarkoitus on aiheuttaa vahinkoa uhrille henkilökohtaisista syistä, saada materialistista etua tai saavuttaa suosiota (Douligeris & Aikaterini 2003). Vuonna 2015 Arbor Networksin tekemän tutkimuksen mukaan yleisin hajautetun palvelunestohyökkäyksen motiivi oli halu esitellä hyökkäysten kapasiteettia. Muita yleisiä motiiveja olivat online-peleihin ja online-vedonlyöntiin liittyvät motiivit sekä kiristäminen ja vandalismi. (Arbor Networks 2016).

Hajautettua palvelunestohyökkäystä käytetään, koska se on halpa, hankala havaita ja todella tehokas. DDoS-hyökkäys on halpa, koska se hyötyy madoilla tai muilla automaattisilla menetelmillä kaapatuista tuhansien zombie-koneiden hajautetuista verkoista. Käyttöoikeuksia tällaisiin bottiverkkoihin (*botnet*) on mahdollista hankkia mustan pörssin markkinoilta, esimerkiksi alle sadalla dollarilla voi ostaa käyttöoikeuden tulvahyökkäykseen. On myös mahdollista tehdä esimerkiksi tuntipohjainen sopimus viiden dollarin tuntihinnalla. Hajautetun

palvelunestohyökkäyksen huono havaittavuus johtuu sen tavasta matkia normaalia liikennettä ja yhteyksiä, jolloin todelliset käyttäjät eivät erotu zombie-koneista. (Malecki 2012).

Tietoturva- ja verkonhallintaohjelmia myyvä Arbor Networks seuraa yli 1000 suurta DDoS-hyökkäystä päivittäin. Heidän mukaansa hyökkäykset kohdistuvat pääosin huomiota herättämättömiin kohteisiin kuten tavallisiin kotikäyttäjiin tai pieniin verkkosivuihin. Nämä hyökkäykset ovat kooltaan merkityksettömiä ja verkon operaattorit voivat helposti estää ne. Yhtiö onkin sitä mieltä, että tällöin motivaationa on viha tai halveksunta tiettyä uhria kohtaan. (Nazario 2008). Toinen yleinen motiivi on kuuluisuuden saavuttaminen sekä suositun web-palvelimen kaatamisesta saatava kunnioitus muiden hakkereiden silmissä (Mirkovic & Reiher 2004).

Myös kostaminen on tavallinen DDoS-hyökkäyksen motiivi Arbor Networksin mukaan. Esimerkiksi spämmääjät (*spammers*) ja verkkourkintaa (*phishing*) harjoittavat ryhmät hyökkäävät kilpailevia spämmääjä- ja verkkourkintaryhmiä vastaan. Lisäksi he hyökkäävät spämmäystä ja verkkourkintaa vastustavia organisaatioita vastaan. Yhtiön mukaan vain pientä osaa DDoS-hyökkäyksistä motivoi taloudellinen hyöty. Tavoitteena saattaa olla esimerkiksi kilpailijan resurssien vahingoittaminen tai yhtiöiden kiristys (Mirkovic & Reiher 2004). Rahallista hyötyä tavoittelevien kohteena ovat yleensä pornosivut sekä netin uhkapelisivustot, mutta myös pankit, mainostoimistot sekä ISP:t. (Nazario 2008).

Osa palvelunestohyökkäyksistä on Arbor Networksin mukaan poliittisesti motivoituneita. Tällöin uhri on luultavasti toiminut väärin hyökkääjän puolella olevia kohtaan. Hyökkäyksen kohteena on tällöin esimerkiksi hallitus, kansallinen infrastruktuuri tai jokin poliittinen puolue. (Nazario 2008). Esimerkiksi sotatilassa oleva maa saattaa iskeä vastustajan kriittisiin resursseihin käyttämällä merkittävän osan koko maan tietokoneiden laskentatehosta hyökkäyksessä (Mirkovic & Reiher 2004). Hallituksen laukaisemissa palvelunestohyökkäyksissä kohteena ovat yleensä eri hallitusten sivustot (.gov-loppuiset osoitteet), avainasemassa olevaa infrastruktuuria, kommunikointia sekä kuljetusta tarjoavat palvelut ja lisäksi olennaisten yhtiöiden palvelimet tai rahaliikennettä ohjaavat palvelimet (Malecki 2012).

DDoS-hyökkäykset voivat olla myös ideologisia, jolloin niiden takana ovat haktivistit. Heillä on tapana hakea julkisuutta ottamalla kohteekseen esimerkiksi korkean profiilin organisaatioita (esimerkiksi Facebook), eroavaa näkemystä poliittista näkemystä edustavia sivustoja tai erilaisten suurtapahtumien (urheilutapahtumat, vaalit) sivuja. Tällä hetkellä yksi tunnetuimpia haktivistiryhmiä on Anonymous. (Malecki 2012). Joissain tapauksissa puolestaan oikea uhri ei välttämättä ole edes hyökkäyspakettien kohde. Tällöin kohteena voivat olla muut palvelut, jotka luottavat uhrin oikeaan toimintaan. (Mirkovic & Reiher 2004).

Uskotaan, että seuraavien muutaman vuoden aikana hyökkäykset kehittyvät enemmän ihmismäistä käytöstä imitoiviksi. Tällöin HTTP-pyyntöjen sarja valitaan siten, että se matkii mahdollisimman hyvin tavallisen ihmiskäyttäjän selauskäytöstä. Tämä tarkoittaa sitä, että yhtä web-sivu pyyntöä seuraa siihen suoraan linkittynyt toinen pyyntö, joka on yleensä kontekstuaalisesti suhteessa aikaisempaan pyydettyyn sivuun. Tämä vaatii sitä, että hyökkääjän on ennen hyökkäyksen tekemistä tutkittava uhrin sivusto ja tehtävä valistuneita arvauksia todennäköisimmästä ihmisen käyttäytymisestä sivustolla. (Stevanovic & Vlajic 2014).

3.5 Uutisia viimeaikaisista hyökkäyksistä

Tämän vuoden alussa Hacker News uutisoi, että Britannian yleisradioyhtiön BBC:n nettisivuille tehtiin vuodenvaihteessa 2015-2016 historian suurin DDoS-hyökkäys. Hakkeriryhmä New World Hacking kaatoi BBC:n palvelimet kolmeksi tunniksi käyttäen omaa hyökkäystyökaluaan ja laukaisivat jopa 602 Gbps saavuttaneen hyökkäyksen. Se ohitti siis reilusti aiemman ennätyksen, joka oli vuonna 2015 mitattu 334 Gbps. (Khandelwal 2016).

Tammikuussa 2016 Security Week uutisoi, että yksi maailman suurimpia pankki- ja finanssipalveluita tarjoavista firmoista, HSBC, joutui massiivisen DDoS-hyökkäyksen kohteeksi. Huhujen mukaan myös muut maan pankit olivat samoihin aikoihin joutuneet palvelunestohyökkäysten kohteeksi, mutta kukaan ei ole ottanut iskuista vastuuta. HSBC kertoi onnistuneesti puolustautuneensa hyökkäykseltä, mutta järjestelmät eivät olleet toiminnassa asiakkaille useisiin tunteihin. (Lennon 2016).

Tivi ja Kauppalehti uutisoivat maaliskuussa 2016, kuinka suomalainen hakkeriryhmä FinnSec Security on tehnyt palvelunestohyökkäyksen eduskunnan julkiseen verkkopalveluun. Samalla viikolla hakkeriryhmä teki iskut Kelaa ja puolustusministeriötä vastaan. Motiivikseen he ilmoittivat muun muassa Suomen kehnon tietoturvan ja kyberpuolustuksen, joiden tulisi olla parempia. (Vänskä 2016; Jouslehto 2016). Toukokuussa puolestaan valtion verkkosivut joutuivat Kouvolan sanomien uutisen mukaan palvelunestohyökkäyksen kohteeksi. Hyökkäys kohdistui valtion yhteiseen julkaisualustaan, Valtoriin, mutta hyökkäyksen tekijä tai motiivi ei ole selvillä. (Puhakka 2016).

Huhtikuussa 2016 pelimaailma koki kolauksen, kun peliyhtiö Blizzardin palvelimia vastaan tehtiin useita palvelunestohyökkäyksiä, jolloin pelaajat eivät päässeet kirjautumaan peleihin useaan tuntiin. Palvelunestohyökkäykset ovat pelimaailmassa tavallisia ja kaaosta haluavien on helppo tehdä hyökkäys pelien autentikointipalvelimia vastaan. (Schreier 2016).

Toukokuun 2016 lopussa NS1, suuri DNS-palveluntarjoaja, joutui hajautetun palvelunestohyökkäyksen kohteeksi. Hyökkäyksen lähde vaihteli viikon aikana Euroopan, Venäjän, Kiinan ja USA:n bottiverkkojen välillä, volyymin noustessa 30-50 Gbps välille. Hyökkääjät lähettivät aidonoloisia DNS-pyyntöjä jopa 50-60 miljoonan paketin sekuntivauhtia. NS1 torjui hyökkäystä suodattamalla liikennettä käyttäen käyttäytymiseen perustuvia sääntöjä. Lisäksi NS1:n kumppanit joutuivat hyökkäyksen kohteeksi, jolloin häirittiin yhtiön nettisivuja sekä yhtiön muita DNS-palveluihin liittymättömiä palveluita. Tästä voidaan päätellä, että NS1 nimenomaan oli kohde eikä sen asiakkaat. Vielä ei kuitenkaan ole tietoa hyökkääjästä tai hyökkäyksen motivaatiosta. (Gallagher 2016).

4 DDoS-hyökkäyksen havaitseminen ja tunnistaminen

Tässä luvussa käsitellään yleisesti hajautettujen palvelunestohyökkäysten havaitsemista sovelluskerroksella, esitellään lyhyesti erästä uutta alempien kerrosten hyökkäysten havaitsemismenetelmää sekä esitellään erilaisia tieteellisiin artikkeleihin pohjautuvia sovelluskerroksen DDoS-hyökkäysten havaitsemismenetelmiä.

4.1 Taustaa

Palvelunestohyökkäykset, jotka tapahtuvat sovelluskerroksella, ovat hankalia havaita ja torjua, koska suurin osa pystyy onnistuneesti muodostamaan TCP-yhteyden ja täten välttämään havaitsemisen. Myös hyökkäyspaketit on yleensä taitavasti naamioitu aidon näköiseksi liikenteeksi ja aito liikenne myös suojaa niitä havaitsemiselta. Ne myös hämäävät väärien IP-pakettien tarkkailemiseen perustuvia menetelmiä, koska TCP-yhteyden luodakseen täytyy käyttää aitoja IP-osoitteita ja -paketteja. Lisäksi sovelluskerroksella tapahtuvat DDoS-hyökkäykset käyttävät vain vähän kaistanleveyttä ja lähettävät joskus vähemmän paketteja verrattuna alempien kerrosten hyökkäyksiin, minkä seurauksena uhri saattaa epäillä kyseessä olevan konevika tai toimintavirhe. Hyökkäys ei myöskään vaadi hyökkääjältä kovin paljoa vaivaa. (Ndibwile ym. 2015).

Vanhempiin hajautettujen palvelunestohyökkäysten havaitsemismenetelmiin lukeutuvat CPP, Ingress/Egress-suodatus, tunkeutumisen havaitsemisjärjestelmät eli IDS:t sekä resursseihin perustuvat kynnysarvot. Asiakaspulma-protokolla eli CPP on algoritmi, jossa jokaisen yhteyden muodostavan asiakkaan täytyy ennen yhteyden muodostusta ratkaista matemaattinen pulma. Pulma on yksinkertainen ja helppo ratkaista, mutta vaatii laskentatehoa asiakkaan laitteelta. (Durcekova & Schwartz & Shahmehri 2012).

Tunkeutumisen havaitsemisjärjestelmät tarkkailevat järjestelmässä tai verkossa esiintyviä tapahtumia ja analysoivat, mitkä niistä ovat mahdollisesti haitallisia tietoturvapolitiikan mukaan. Haitallisia tapahtumia ovat esimerkiksi haittaohjelmat, salasanahyökkäykset, käyttöoikeuksien väärinkäytökset tai väärennetyt paketit. (Durcekova & Schwartz & Shahmehri

2012). IDS havaitsee verkkokerroksella tapahtuvat hyökkäykset helpommin kuin sovelluskerroksella tapahtuvat, koska verkkokerroksella hyökkäykset perustuvat yleensä virheellisten pakettien lähettämiseen, jotka IDS:n on helppo erottaa aidoista paketeista. Sovelluskerroksella puolestaan palvelunestohyökkäys perustuu aidonnäköisten pyyntöjen lähettämiseen, jolloin aitoa yhteydenottopyyntöä ei voi erottaa zombie-koneen lähettämästä yhteydenottopyynnöstä tarkastelemalla pyyntöä itseään. (Beitollahi & Deconinck 2012b).

Ingress/Egress-suodatus on tekniikka, jossa valvotaan, että tulevilla tai lähtevillä paketeilla ei ole väärennettyä IP-osoitetta otsakkeessaan. IP-osoitteen avulla muut koneet tunnistavat, mistä paketti on lähtöisin, jotta se voidaan esimerkiksi lähettää takaisin. Mikäli IP-osoite on väärennetty, ei voida tietää, mistä paketti on oikeasti lähtöisin. Ingress/Egress-suodatuksen tarkoituksena on siis estää spoofaus eli esiintyminen toisena koneena. Resursseihin perustuvat kynnysarvot puolestaan määrittävät, minkä verran pyyntöjä voidaan käsitellä ilman resurssien ehdyttämistä. Yleensä se määritellään tiettynä prosenttiosuutena maksimikäsittelymäärästä. (Durcekova & Schwartz & Shahmehri 2012).

Uudempia sovelluskerroksella tapahtuvien palvelunestohyökkäysten havaitsemiseen perustuvia menetelmiä on kahdenlaisia: Allekirjoitukseen (*signature*) perustuvat menetelmät ja poikkeamiin (*anomaly*) perustuvat menetelmät. Allekirjoitukseen perustuvat menetelmät pohjautuvat tilastollisten muutosten valvontaan. Ensimmäisenä askeleena on valita saapuvalle liikenteelle jokin parametri ja mallintaa sitä satunnaisena sarjana normaalioloissa. Kaikki tällaiset menetelmät perustuvat oletuksiin, esimerkiksi oletetaan saapuvien ja lähtevien pakettien tahti on suhteessa toisiinsa, joka ei päde esimerkiksi videoiden streamauksessa. Allekirjoitukseen perustuvat menetelmät voivat tunnistaa hyökkäyksen, jos se täsmää tunnettuihin haitallisesta toiminnasta kertoviin piirteisiin. Näiden menetelmien heikkoutena on vaikeus luoda täsmällisiä allekirjoituksia, kun hyökkääjän puolestaan on helppo vaihdella hyökkäysliikenteen tyyppiä ja sisältöä. Allekirjoituksiin perustuvia menetelmiä voidaan käyttää myös hyökkääjän ja zombie-koneiden välisen kommunikoinnin havaitsemiseen, mutta yleensä kommunikoinnin salaus tekee menetelmät tehottomiksi. (Durcekova & Schwartz & Shahmehri 2012).

Poikkeamiin perustuvat menetelmät puolestaan tunnistavat hyökkäyksen, mikäli valvottu liikenne ei täsmää normaalin liikenteen profiiliin. Nämä menetelmät ovat tällä hetkellä tutkimuksen fokuksena, koska niillä pystytään havaitsemaan uudenlaisia hyökkäyksiä. Normaalin liikenteen profiilien rakentaminen on yleensä ensimmäinen vaihe. Profiilien rakentaminen tapahtuu käyttämällä harjoitusaineistoa ja tilastollista mallintamista. Tilastollisella poikkeamien tunnistuksella löydetään ensin parametrit samankaltaisuusmittojen (*similarity measures*), esimerkiksi IP paketin pituus, luomiseen, jonka jälkeen tätä mittaa verrataan uuteen liikenteeseen. Jos valvottu liikenne eroaa normaalin liikenteen kynnysarvosta, palvelunestohyökkäys on havaittu. Menetelmän ongelmana on löytää harjoitusaineisto, jolla saadaan luotua kaikkea normaalia liikennettä vastaavat profiilit. Tästä seuraa helposti aitojen käyttäjien luokitteleminen hyökkääjiksi. Tämän minimoimiseksi voidaan parametrien määrää lisätä, mutta tämä puolestaan hidastaa havaitsemisnopeutta. (Durcekova & Schwartz & Shahmehri 2012).

4.2 Alempien kerrosten havainnointi

Alempien kerrosten havainnointiin löytyy paljon erilaisia menetelmiä, mutta tämä tutkielma keskittyy kuitenkin pääasiassa sovelluskerrosten hyökkäysten havaitsemiseen. Kuitenkin modernit palvelunestohyökkäykset yhdistävät monia erilaisia hyökkäyksiä monella eri kerroksella, erityisesti kuljetuskerroksella. Tästä syystä olen poiminut tähän lukuun tänä vuonna julkaistun artikkelin (Saied & Overill & Radzik 2016), jossa pyritään havaitsemaan tunnetut ja vielä tuntemattomat (0Day-hyökkäykset) TCP-, UDP- ja ICMP-protokollaa käyttävät hajautetut palvelunestohyökkäykset.

Saied, Overill ja Radzik käyttävät hyökkäysten havaitsemiseen valvottuja neuroverkkoja (*supervised artificial neural networks*), jotka ovat eteenpäin syöttäviä (*feed forward*) ja käyttävät back-propagation-algoritmia (*Error back propagation*) sekä sigmoidifunktiota aktivaatiofunktionaan (*Sigmoid activation function*). Neuroverkon kouluttamiseen he käyttävät pakettien otsakkeita, kuten lähtöosoitteita, ID-numeroa sekä sekvenssinumerosta ja kohdeportista muodostettua paria. (Saied & Overill & Radzik 2016).

Neuroverkko koostuu yleensä syötekerroksesta (*input layer*), piilotetusta kerroksesta (*hidden layer*) ja ulostulokerroksesta (*output layer*). Saied, Overill ja Radzik kertovat artikkelissaan, että heillä on käytössä kolme havaitsijaa (*detector*), joista jokainen sisältää eri protokollille tarkoitetun kolmikerroksisen neuroverkon. Järjestelmä valvoo jatkuvasti liikennettä ja havaitsijat voivat keskustella toistensa kanssa sekä tarjota toisilleen apua tarvittaessa käyttämällä salattuja viestejä. Kun väärennetty paketti havaitaan, suojamekanismi pudottaa väärennetyt paketit ja päästää aidot paketit läpi. Estetyt paketit puolestaan päästetään jatkamaan, kun järjestelmä merkkaa ne normaaleiksi. (Saied & Overill & Radzik 2016).

Algoritmi pystyy tunnistamaan tuntemattomat hyökkäykset, mikäli ne ovat samankaltaisia kuin sille jo opetetut hyökkäykset. Jos neuroverkon opettamiseen kuitenkin käytetään vain vanhaa dataa, niin järjestelmä ei havaitse kaikkia hyökkäyksiä. Vain ajantasaisella datalla koulutetut neuroverkot pystyvät havaitsemaan sekä tunnetut että tuntemattomat hyökkäykset. Tarkkuus on tällä menetelmällä heidän artikkelissaan 98 %. (Saied & Overill & Radzik 2016).

4.3 CAPTCHA-testit

DDoS-hyökkäyksiä voidaan yrittää estää tarkastelemalla, onko palvelimen käyttäjä ihminen vai kone. Tähän käytetään ns. CAPTCHA-testejä (joskus puhutaan myös pelkästä Turingin testistä), jotka ovat jonkin ohjelman automaattisesti generoimia testejä (yleensä vääristynyttä tekstiä sisältäviä kuvia, mutta myös äänitestejä sekä loogisia- tai semanttisia testejä), jotka ihminen pystyy ratkaisemaan, mutta kone ei. CAPTCHA-testillä pyritään varmistamaan, että käyttäjä on oikea ihminen eikä botti, ennen kuin hänelle annetaan oikeus suorittaa joitakin asioita palvelimella (esimerkiksi kommentin lähettäminen keskustelupalstalla). CAPTCHA-testi torjuu tehokkaasti ja halvalla spämmääjiä, spoilaajia, hakukoneiden ryömijöitä (*crawlers*) ja käytännössä kaikkia automaattisia ohjelmia, jotka yrittävät käyttää jotakin palvelua tai sivustoa (Guo ym. 2011).

CAPTCHA-testiä käytetään palvelunestohyökkäysten torjumiseen esimerkiksi Moreinin ym. (2003) artikkelissa esitellyssä WebSOS-arkkitehtuurissa. Arkkitehtuurin tarkoituksena on sallia todellisten käyttäjien pääsy web-palvelimelle palvelunestohyökkäyksen aikana ja

se yhdistelee CAPTCHA-testejä, kryptografisia protokollia datan alkupisteen selvittämiseen, pakettien suodattamista, kerrostettuja verkkoja sekä johdonmukaista hajautusta (*hashing*) tämän tavoitteen saavuttamiseen. (Morein ym. 2003).

Myös Singh ja De ovat artikkelissaan (2015) ehdottaneet CAPTCHA:n käyttöä hajautettujen palvelunestohyökkäysten torjunnassa. Heidän menetelmänsä perustuu kolmeen vaiheeseen, joista jokaisella on mahdollisuus suodattaa hyökkääjät pois palvelimelta. Ensimmäisessä vaiheessa rajoitetaan pääsyä palvelimelle käyttämällä ”musta lista”-menetelmää (*blacklisting*) eli estetään mustalla listalla olevien IP-osoitteiden pääsy palvelimelle. Toisessa vaiheessa rajoitetaan samalta IP-osoitteelta tulevien HTTP-pyyntöjen tahtia. Mikäli pyyntöjen tahdin lukumäärä on ennakkoon määriteltyä kynnysarvoa pienempi, käyttäjä siirtyy normaaliin sisäänpääsyyn. Mikäli taas pyyntöjen tahdin lukumäärä ylittää kynnysarvon, käyttäjä siirretään kolmanteen vaiheeseen, jossa käyttäjän tulee suorittaa CAPTCHA-testi. Mikäli CAPTCHA-testiä ei pääse läpi, päivitetään IP-osoite mustalle listalle. (Singh & De 2015).

Chen ym. ehdottavat artikkelissaan (2012) CAPTCHA-testien käyttöä yhdessä neuroverkkojen kanssa, jotta DDoS-hyökkäykset voitaisiin havaita ja torjua tehokkaasti. Järjestelmä käyttää RBF-neuroverkkoa, joka pyrkii tunnistamaan oikeat käyttäjät ja sallimaan heidän pääsynsä palveluun, samalla estäen luvattomien käyttäjien tekemät DDoS-hyökkäykset. Parametreina ovat IP-osoitteen näkymiseen kuluva keskimääräinen aika, pakettien lähteen hajautusaste (varianssi IP-osoitteen näkymiselle kuluvalle keskimääräiselle ajalle) sekä verkkokorttien myyjien hajautusaste (MAC-osoitteen kolme ensimmäistä numeroa). Mikäli käyttäjä tunnistetaan tietokannan avulla luotetuksi, on hänellä pääsy sivustolle suoraan. Mikäli käyttäjään ei luoteta, tarkastellaan, voidaanko hänen tunnistetietonsa verifioida ja ovatko tunnistetiedot aitoja. Tarvittaessa joko suoritetaan CAPTCHA-testi, joka sisältää satunnaisia kysymyksiä, käyttäjän kanssa kommunikointia, merkkien tunnistuskyvyn arviointia, kielen tunnistusta, tiedon käsittelyä sekä ongelmien analysointia tai vaihtoehtoisesti evätään käyttäjältä pääsy palveluun. (Chen ym. 2012).

CAPTCHA-testejä on kritisoitu siitä, että ne eivät välttämättä huomioi vammaisia tai näkövammaisia ja lisäksi ne vaativat suuren kuvakirjaston, palvelimen sekä ohjelmiston CAPTCHA-testien luomiseen. CAPTCHA-testin luomiseen, näyttämiseen ja arvosteluun

kuluva aika lisää palvelimen kuormaa sekä aiheuttaa viivettä käyttäjälle. CAPTCHA-testi on myös mahdollista kiertää arvaamalla, kuten sanakirjahyökkäyksessä (*dictionary attack*). Tämä onnistuu erityisesti silloin, kun CAPTCHA-testejä on vain muutama täysin uniikki tai ne ovat staattisia. Toinen mahdollisuus CAPTCHA-testin kiertämiseen on kirjoittaa kuvanprosessointiohjelma, joka yrittää suorittaa ihmisen osan testissä käyttämällä esimerkiksi hahmontunnistusalgoritmeja. Kolmas tapa CAPTCHA-testin kiertämiseen on huijata ihmisiä ratkaisemaan testit botin puolesta. Tämä onnistuu esimerkiksi näyttämällä hyökkääjän oman palvelimen tai sivuston käyttäjille toisen palvelimen CAPTCHA-testejä. Kun liikennettä on tarpeeksi, hyökkääjä voi saada CAPTCHA-testiin ratkaisun ajoissa välittääkseen sen kohdesivustolle. Toinen vaihtoehto on välittämää vastaantulevat testit ihmisoperaattoreille, jotka ratkaisevat bottien kohtaamat testit. Tämän jälkeen botit voivat jatkaa omaa toimintaansa normaalisti. (Guo ym. 2011).

4.4 Koneoppimisen ja hahmontunnistuksen avulla tapahtuva havainnointi

Liao ym. ovat artikkelissaan (2014) luokitelleet tämän hetken suosittuja sovelluskerroksen palvelunestohyökkäysmalleja ja jakaneet ne neljään kategoriaan. Nämä kategoriat ovat yhden URL:in toistohyökkäys, monen URL:in toistohyökkäys, satunnaisvalintahyökkäys ja istunnon toistohyökkäys. Yhden URL:in toistohyökkäyksessä toistuvia pyyntöjä lähetetään joko kiinnitetyllä tai satunnaisella nopeudella, kun taas monen URL:in toistohyökkäyksessä käytetään eri URL:ja. Satunnaisvalintahyökkäyksessä puolestaan skannataan uhrin web-sivusto ja valitaan tämän jälkeen satunnaisesti kohteen URL:ja. Istunnon toistohyökkäyksessä simuloidaan normaalien käyttäjien käyttäytymistä ja lähetetään toistuvasti hyökkäyspyyntöjä. (Liao ym. 2014).

Liao ym. ehdottavat artikkelissaan (2014) palvelunestohyökkäyksen havaitsemiseen piirteenirroitukseen (*feature extraction*) ja käyttäjien käyttäytymisen mallintamiseen perustuvaa menetelmää. He käyttivät piirteenirroitusta palvelimen lokitiedostoon, josta he saivat yhdeksän eri piirrettä mallintamista varten. Nämä piirteet ovat lähdeosoite (sourceAddress), pyyn-

töajat (requestTimes), eri pyyntöjen ajat (diffReqTimes), onnistuneiden pyyntöjen määrä (timesOfCode200), pyyntöjen kokonaispituus (totalLength), istunnon kesto (sessionDuration), URL:in tasojen läpikäyntijärjestys (sequenceOfUrlLevel), pyyntöjen määrä aikajaksossa (sequenceOfRequestFrequency) sekä pyyntöjen välit (sequenceOfRequestInterval). Paremman testeistä suoriutumisen takia he muodostivat lisäksi 8 muuttujaa, joista viittä lopulta käytettiin. (Liao ym. 2014).

Hyökkäyksen havaitsemiseen käytettiin kolmea tiedonlouhinnan luokittelualgoritmia, jotka olivat naiivi bayes, RBF-neuroverkko sekä C4.5-päätöspuu. Tuloksia arvioitiin kolmen tunnusluvun avulla, jotka olivat tarkkuus (oikein ennustetut suhteessa kaikkiin), havaitsemisaste (hyökkäyksiksi oikein luokitellut suhteessa kaikkiin oikein luokiteltuihin) sekä väärien positiivisten tulosten aste (väärin hyökkäyksiksi luokiteltujen suhde kaikkiin normaaleihin). Tulokseksi he saivat, että heidän valitsemillaan piirteillä ja kaikilla kolmella luokittelualgoritmilla eri hyökkäystyyppien (yhden URL:in toistohyökkäys, monen URL:in toistohyökkäys, satunnaisvalintahyökkäys ja istunnon toistohyökkäys) luokittelun tarkkuus oli lähes 100 %, havaitsemisaste 1 % ja väärien positiivisten aste lähellä nollaa. (Liao ym. 2014).

Yadav ja Selvakumar esittelevät artikkelissaan (2015) poikkeamiin (*anomalies*) perustuvan tilastollisen hahmontunnistusmenetelmän. He ovat muodostaneet harjoitusdatan esikäsittelemällä web-palvelimen lokitiedostosta saatua normaalin liikenteen ja hyökkäysliikenteen aikaista dataa. Tämä esikäsittelyprosessi sisältää piirteenirroitusta, josta on saatu 8 piirrettä sekä piirteiden muodostamista, josta on saatu vielä 9 piirrettä lisää (yhteensä siis 17 piirrettä). Näin saadusta datasta on pääkomponenttianalyysia käyttäen etsitty sellaiset parametrit ja kynnysarvo, joilla voidaan erotella toisistaan normaali liikenne ja hyökkäysliikenne. Näihin parametreihin ja kynnysarvoon perustuen menetelmässä luokitellaan logistisen regression avulla hyökkääjiin ja normaaliin liikenteeseen. Heidän menetelmällään pyyntötulvahyökkäysten, istuntotulvahyökkäysten ja asymmetristen hyökkäysten havaintoaste oli noin 99 %, kokonaistarkkuus noin 98,5 % ja väärien positiivisten aste oli noin 1,5 %. (Yadav & Selvakumar 2015).

Stevanovic ja Vlajic ehdottavat artikkelissaan (2014) 3-vaiheista HTTP-pohjaisten DDoS-hyökkäysten havaitsemis- ja suojautumismenetelmää. Ensimmäisessä vaiheessa seulotaan

kaikki triviaalit hyökkäykset, joissa hyökkäys tapahtuu lähettämällä yksi tai rajallinen määrä toisiinsa liittymättömiä HTTP-pyyntöjä sivustolle. Nämä hyökkäykset voidaan havaita analysoimalla liikennettä pyyntö kerrallaan. Haitallisiksi havaitut istunnot estetään eivätkä ne siirry seuraavaan vaiheeseen. (Stevanovic & Vlajic 2014).

Toisessa vaiheessa seulotaan liikenteestä keskitason hyökkäykset. Keskitason hyökkäykset pyrkivät sekoittumaan normaaliin liikenteeseen tekemällä jatkuvasti ennalta määriteltyjä ”ihmismäisiä” pyyntöjonoja. Tällaiset hyökkäykset voidaan havaita analysoimalla istuntojen kronologista selausjärjestystä eli sekvenssiä. Havaittuihin sekvensseihin Stevanovic ja Vlajic käyttävät datavirta-algoritmeja, kuten COD, ILOF ja DStream, joilla päivitetään tyypillisten/sallittujen sekvenssiprofiilien joukkoa ja samalla seurataan uusia sekvenssiprofiileja. Vertailumittana käytetään pisimmän yhteisen alisekvenssin normalisoitua pituutta. Sellaiset istunnot, jotka merkitään vaiheessa 2 poikkeaviksi, eivät jatka vaiheeseen 3 vaan niille esitetään CAPTCHA-testi. Mikäli käyttäjä läpäisee CAPTCHA-testin, heidän sekvenssiprofiilinsa lisätään tyypillisten selausprofiilien joukkoon. Muussa tapauksessa pääsy sivustolle estetään. (Stevanovic & Vlajic 2014).

Kolmannessa vaiheessa pyritään havaitsemaan edistyneet hyökkäykset, joissa pyritään matkimaan tavallisten käyttäjien pyyntöjen sekvenssiä. Näiden havaitseminen perustuu käyttäjien selauskäyttäytymisen aikatasojen (*time domain*) analysointiin, jolloin aiemmin mainitut algoritmit seuraavat tyypillisiä sivun katseluaikoja sekä tunnistavat poikkeavat katseluajat. Jos poikkeamien määrä ylittää ennalta määrätyn kynnysarvon, luokitellaan istunto mahdollisesti vaaralliseksi ja pyydetään käyttäjää suorittamaan CAPTCHA-testi. (Stevanovic & Vlajic 2014).

Tämä menetelmä onnistui heidän testeissään tunnistamaan 92 % istunnoista oikein haitallisiksi. Väärin haitallisiksi luokiteltiin 14–27 % hyökkäyksen tyypistä riippuen ja heidät siirrettiin CAPTCHA-testiin. (Stevanovic & Vlajic 2014).

Choi ym. esittävät artikkelissaan (2011) aikaikkunan valvontamallin (*Timeslot Monitoring Model, TMM*), jonka tavoitteena on mallintaa käyttäjän pyyntöjen sarjan jatkuvuutta tietyn valvonta-ajan aikana. Malli luo normaaleille käyttäjille ja hyökkääjille omat profiilit, joiden

perusteella päätetään tukivektorikoneen (*support vector machine, SVM*) avulla onko kyseessä hyökkäys vai ei. (Choi ym. 2011).

Malli perustuu HTTP-protokollan tyypillisiin pyyntöjen ja vastausten kuvioihin ja erityisesti pyyntöjen välille jäävään satunnaiseen käyttäjän reaktioaikaan. Hyökkäysliikenne eroaa tavallisesta liikenteestä, sillä ihmisten ja palvelimen välillä on vuorovaikutusta ja ihminen tutkii vastauksena saatua sivua, kun taas botit yleensä lähettävät pyyntöjä jatkuvasti ja toistuvasti palvelimen kanssa ilman varsinaista vuorovaikutusta. (Choi ym. 2011).

Palvelun pyyntökäytös (*service request behavior, SRB*) on jokaiselle palvelulle erikseen määriteltävä määre, esimerkiksi paketti, pakettien sarja tai avainsana, joka edustaa palvelun sovelluskerroksen pyyntöä. HTTP-protokollan tapauksessa pyyntökäytöstä voisi edustaa GET-paketti. Valvonta-aika (*monitoring period, MP*) puolestaan tarkoittaa ennalta määrätyistä aikayksiköistä (*timeslot, TS*) koostuvaa ajanjaksoa, jonka aikana pyyntökäytöksen piirteitä voidaan havaita. Valvonta-aika alkaa, kun uusi pyyntökäytös ilmenee ja loppuu, kun ennalta määritelty aikaväli kuluu umpeen. Näin ollen yhden pyyntökäytösilmentymän aikana voi kulua useampi valvonta-aikaväli, mikäli valvonta-aika on määritelty pyyntökäytökseen kuluvaa aikaa pienemmäksi. Avainpiirteet (*key features, KF*) irroitetaan aina valvonta-ajan päätyttyä ja niitä ovat palvelun pyytämien aikayksiköiden kokonaismäärä valvonta-aikana, enimmäismäärä jatkuvia palvelun pyytämiä aikayksiköitä valvonta-aikana sekä enimmäismäärä palvelun ei-pyytämiä aikayksiköitä valvonta-aikana. Näiden kolmen avainpiirteen avulla luodaan tukivektorikone, jolla erotetaan hyökkääjät ja normaali liikenne toisistaan. (Choi ym. 2011).

Choi ym. saivat tulokseksi, että heidän menetelmänsä havaitsee yhden valvonta-aikavälin aikana 99,4 % hyökkääjistä ja väärien positiivisten määrä oli 2,4 %. Kasvattamalla aikavälien määrää kolmeen saatiin havaitsemistasoksi 100 % ja väärien positiivisten määräksi 0 %. (Choi ym. 2011).

4.5 Välityspalvelimia käyttävät havaitsemismenetelmät

Ndibwile ym. ehdottavat artikkelissaan (2015), että yhden palvelimen sijaan käytössä olisikin kolme palvelinta: oikea palvelin (*Real Web Server*), harhautuspalvelin (*Decoy Web Server*) ja syöttipalvelin (*Bait Web Server*). Aito palvelin ottaa vastaan seulotun oikean liikenteen, syöttipalvelin hoitaa käyttäjien tunnistuksen joko aidoiksi käyttäjiksi tai hyökkääjiksi ja harhautuspalvelin vastaanottaa haitalliseksi luokitellun liikenteen. Syöttipalvelin sisältää vain etusivun ja JavaScriptillä luodun popUp-ikkunan, joka toivottaa tervetulleeksi sivustolle. PopUp-ikkunan sulkeminen erottelee aidot käyttäjät boteista, jonka jälkeen käyttäjä ohjautuu joko oikealle palvelimelle tai harhautuspalvelimelle. Harhautuspalvelin vaikuttaa oikealta palvelimelta, jolloin hyökkääjät eivät erota sitä oikeasta ja samalla väärin luokitellut käyttäjät voidaan seuloa ilman, että heidän käyttökokemuksensa häiriintyy. (Ndibwile ym. 2015).

Ndibwile ym. käyttävät tunkeutujien havaitsemiseen IDS:ää, joka seuloo kaikesta liikenteestä epäilyttävän liikenteen käyttäen koneoppimisalgoritmia ja lähettää sen suoraan harhautuspalvelimelle. IDS:n normaaliksi luokittelema liikenne puolestaan ohjataan syöttipalvelimelle. Koneoppimisalgoritmia opetettiin etukäteen merkityllä datalla (*labeled datasets*) ja luokittelumalliksi valittiin satunnaiset päätöspuut (*random tree algorithms*). Luokittelumalli rakennettiin Snort NIPS:in päälle. Oppimisvaiheessa normaaliin liikenteeseen sekoitettiin eri DDoS-työkaluilla (esimerkiksi LOIC, HOIC ja SlowLoris) luotua merkittyä dataa, jolloin harjoitteluvaiheen jälkeen suurin osa käytetyillä työkaluilla luodusta liikenteestä havaitaan ja luokitellaan oikein. (Ndibwile ym. 2015).

Devi ja Yogest esittävät artikkelissaan (2012) palvelunestohyökkäysten havaitsemiseen ja torjumiseen menetelmän, joka perustuu käyttäjien istuntotietojen analysoimiseen ja niiden perusteella annettavaan epäilyttävyyspistemäärään. Menetelmä toimii siten, että ensiksi web-palvelimen logeista lasketaan sisäänpääsymatriisi (*access matrix*). Tämä saadaan laskettua HTTP-pyyntöjen tahdista, HTTP-istunnon tahdista, palvelimen käytetyistä dokumenteista sekä käytön kestosta. Koska matriisi on moniulotteinen, käytetään sen yksinkertaistamiseen singulaariarvohajotelmaa (*singular value decomposition, SVD*) ja riippumattomien

komponenttien analyysiä (*independent component analysis, ICA*). Palvelunestohyökkäyksen vastamekanismi sijaitsee heidän menetelmässään käänteisellä välityspalvelimella (*reverse proxy*). Vastamekanismi tutkii jokaisen pyynnön jokaisesta HTTP-istunnosta, jäsentää pyynnön tyypin URL:sta sekä ylläpitää työmäärää ja istunnon pyyntöjen saapumishistoriaa. Vastamekanismi asettaa jokaiselle saapuvalle istunnolle epäilyttävyyspistemäärän, joka lasketaan vertaamalla saapuvan istunnon parametreja yksinkertaistetun sisäänpääsymatriisin vastaaviin arvoihin. Mikäli hajonta on suurta, on istunto epäilyttävä. Tämän jälkeen pyyntö ohjataan oikealle palvelimelle taakantasauskäytännön ja järjestelijän päätöksen perusteella. Järjestelijä analysoi epäilyttävyyspistemäärän ja palvelimen senhetkisen kuorman, jonka jälkeen se päättää pääseekö istunto eteenpäin vai pudotetaanko se pois. (Devi & Yogesh 2012).

4.6 Ruuhkapiikkien erottaminen palvelunestohyökkäyksistä

Ruuhkapiikki (*Flash crowd, Flash event*) ja hajautettu palvelunestohyökkäys ovat ilmiöinä samanlaisia, koska kummassakin palvelimelle tulee lyhyen ajan sisällä paljon pyyntöjä. Nämä voidaan kuitenkin erottaa toisistaan kahden asian perusteella. Ensimmäiseksi palvelunestohyökkäys on yleensä lähtöisin pieneltä joukolta käyttäjiä, kun ruuhkapiikki johtuu suuresta kävijämäärästä. Toisekseen palvelunestohyökkäyksen tekijät ovat yleensä uusista asiakasklustereista, kun taas ruuhkapiikin käyttäjät ovat lähtöisin aiemmin tunnetuista asiakasklustereista. (Devi & Yogesh 2012).

Zhou ym. (2014) ovat kehittäneet internetin runkoon (*backbone*) sopivan puolustautumisarkkitehtuurin, joka soveltuu neljänlaisten hyökkäystyyppien havaitsemiseen. Nämä ovat toistuvat pyynnöt, rekursiiviset pyynnöt, toistuva työmäärä sekä ruuhkapiikit. Toistuvien pyyntöjen hyökkäyksessä sekä hyökkäys että sen lähtöpiste keskittyvät tiettyyn pisteeseen. Rekursiivisten pyyntöjen hyökkäyksessä puolestaan hyökkäys lähtee tietystä paikasta, mutta hajautuu useaan kohteeseen. Toistuvien työmäärien hyökkäyksissä lähetetään jatkuvasti suuritöisiä pyyntöjä (esim. tietokantakyselyt) ja ruuhkapiikeissä on kyse siitä, että suuri määrä ihmisiä vierailee yllättäen samalla sivulla (esimerkiksi ohjelmistotarjousten takia). (Zhou ym. 2014).

Arkkitehtuuri koostuu kolmesta eri moduulista, jotka ovat epänormaalin liikenteen havaintomoduuli (*abnormal traffic detection module, ATDM*), hyökkäyksen havaintomoduuli (*application layer DDoS attack detection module, DADM*) sekä suodatusmoduuli (*filter module, FM*). Liikenteen havaintomoduuli analysoi reaaliajassa normaalista HTTP GET-liikenteestä poikkeavia piirteitä ja ilmoittaa niistä hyökkäyksen havaintomoduulille. Tarkoituksena on tutkia liikenteen intensiteettiä (vastaanotettujen pakettien kokonaismäärä) vakioilla aikaväleillä ja ennustaa liikenteen intensiteettiä autoregressiivisen mallin (*auto-regressive model, AR*) avulla. Koska normaalinkin liikenteen intensiteetissä on nousuja ja laskuja, käyttävät he Kalman-suodinta (*Kalman filter*) arvojen tasoittamiseen. Mikäli havaittujen ja ennustettujen arvojen välinen hajonta ylittää kynnysarvon, lähetetään hyökkäyksen havaintomoduulille viesti epänormaalista liikenteestä. Hyökkäyksen havaintomoduuli puolestaan toimii aina vain silloin, kun se saa viestin liikenteen havaintomoduulilta epänormaalista liikenteestä ja lopettaa toimintansa, kun liikenteen havaintomoduuli ilmoittaa liikenteen normalisoituneen. (Zhou ym. 2014).

Hyökkäyksen havaintomoduulissa on hyökkäyksenhavaitsemisjärjestelmä, joka lähettää hyökkäyksen lähtöpisteiden IP-osoitteet suodatinmoduulille. Havaintomoduuli mallintaa liikennettä frekvenssivektoreiden, liikenteen mallien, dynaamisten mallien sekä liikenteen lähtöpisteiden entropian perusteella. Liikenteen reaaliaikainen prosessointi onnistuu reaaliaikaisen frekvenssivektorin (r*eal-time frequency vector, RFV*) avulla, joka sisältää jokaisen sivuston resurssin keskimääräisen vierailumäärän. Liikenteen mallien määrittelemiseksi käytetään Pearsonin korrelaatiokerrointa, jonka tulosten perusteella frekvenssivektoreita luokitellaan joko samaan tai eri malliin kuuluviksi. Liikenteen mallien lisäksi käytetään päätöksentekoon dynaamisia malleja, jotka koostuvat aikaan sidonnaisista käyttäytymismalleista, tässä tapauksessa käyttäjien vierailuista sivulla. Lisäksi tarkastellaan liikenteen lähtöpisteiden entropiaa, jotta pystytään erottamaan toisistaan ruuhkapiikit ja palvelunestohyökkäykset. Suodatinmoduuli puolestaan joko hyväksyy tai hylkää pyynnöt perustuen IP-osoitteen epäilyttävyyteen. Tämä tapahtuu Bloom-suodattimen avulla (*Bloom filter*). (Zhou ym. 2014).

Sachdeva, Kumar ja Singh esittävät artikkelissaan (2016) liikenteen klusterien entropiaan perustuvan menetelmän palvelunestohyökkäysten erottamiseen ruuhkapiikeistä. Menetelmä

perustuu ajatukseen, että vaikka jokainen yksittäinen hyökkäyspaketti on täysin aito, niin yhdessä ne vaikuttavat liikenteen normaaliin jakautumiseen vähentäen sen satunnaisuutta. He käyttävät Shannonin entropiaa tarkastellakseen lähteiden IP-osoitteiden satunnaista jakautumista sekä liikenteen klusterien (samasta verkosta lähtöisin olevien pakettien) jakautumista. IP-osoitteiden entropia ja liikenteen klusterien entropia lasketaan tietyn ajan välein normaalissa tilassa, jolloin saadaan vertailukohtana käytetty keskiarvo näistä kahdesta entropiasta. Nykyistä liikennettä valvotaan jatkuvasti ja laskettuja nykyisiä entropioita verrataan keskiarvoihin. Ilman hyökkäystä IP-osoitteiden entropia ja liikenteen klusterien entropia vaihtelee vain vähän. Näin ollen hyökkäys havaitaan, kun nykyiset entropiat eroavat huomattavasti keskiarvoista. (Sachdeva & Kumar & Singh 2016).

Menetelmällä voidaan havaita viisi erilaista tapahtumaa tarkastelemalla eri loogisia yhdistelmiä entropioiden suuruudesta. Nämä tapahtuvat ovat DDoS-hyökkäys (kumpikin entropia keskiarvoa suurempi), ruuhkapiikki (IP-osoitteiden entropia keskiarvoa suurempi), DDoS-spoofing-hyökkäys (IP-osoitteiden entropia keskiarvoa suurempi ja liikenteen klusterien entropia keskiarvoa pienempi), suuren volyymin DDoS-hyökkäys (IP-osoitteiden entropia keskiarvoa pienempi) sekä monimutkainen DDoS-hyökkäys (liikenteen klusterien entropia keskiarvoa suurempi). IP-osoitteiden kynnysarvon valitsemiseen he käyttävät kuuden sigman menetelmää (*six sigma method*) ja liikenteen klusterien valitsemiseen taas ROC-käyrää (*receiver operating characteristic curve*). (Sachdeva & Kumar & Singh 2016).

Sirisha ym. esittelevät artikkelissaan (2015) osittaiseen Markovin piilomalliin (*Hidden semi-Markov Model, HsMM*) perustuvan menetelmän, jolla voidaan havaita ruuhkapiikin spatio-temporaaliset mallit (*spatial-temporal patterns*) ja havaita sovelluskerroksen palvelunesto-hyökkäyksiä. Menetelmä soveltuu vain aikaisessa vaiheessa tapahtuvaan hyökkäysten havainnointiin. Lisäksi se soveltuu hyökkäyksiin, joissa pyynnöt tulevat tasaisella tahdilla (*Constant Rate Attack*), kasvavalla tahdilla (*Increasing Rate Attack*) tai pyrähdyksittäin (*Stochastic pulsing attacks*). (Sirisha ym. 2015).

Menetelmässä on kolme vaihetta, jotka ovat datan esikäsittelyvaihe, harjoitusvaihe ja valvontavaihe. Datan esikäsittelyvaiheessa määritellään sisäänpääsymatriisi (*access matrix*),

jonka avulla luodaan normaalin liikenteen spatio-temporaaliset mallit ja tarkkailla palvelunestohyökkäyksiä ruuhkapiikin aikana. Harjoitusvaiheessa sisäänpääsymatriisille tehdään pääkomponenttianalyysi (*Principal Component Analysis, PCA*) ja riippumattomienkomponenttien analyysi (*Independent Component Analysis, ICA*). Lopuksi käytetään osittaista Markovin piilomallia ja lasketaan entropia ja kynnysarvo valvontavaihetta varten. Valvontavaiheessa tarkoituksena on havaita dynaamiset siirtymät normaalissa liikenteessä entropian ja kynnysarvon avulla, jotta voidaan erottaa palvelunestohyökkäys ruuhkapiikin aikana. (Sirisha ym. 2015).

4.7 Hyökkäyksen havaitseminen pilvipalveluissa

Huang, Huang ja Chiang esittelevät artikkelissaan (2013) moduuleihin perustuvan hyökkäyksenhavaitsemisjärjestelmän, jossa haitalliset IP-osoitteet laitetaan estolistalle, mikäli käyttäjä ei läpäise Turingin testiä. Järjestelmä koostuu neljästä moduulista, jotka ovat lähteiden tarkastus- ja laskentamoduuli (*Source Checking and Counting module*), Turing Test-moduuli, havaintomoduuli (*Attack Detection module*) ja kysymyksenluontimoduuli (*Question Generation module*). Jokaisella saapuvalla paketilla on jokin neljästä tilasta, jotka ovat musta, estettävä, valkoinen tai tuntematon tila. Ensimmäisenä saapuvat paketit ohjataan lähteiden tarkastus- ja laskentamoduuliin, jossa mustalle listalle kuuluvat paketit pudotetaan ja estettävälle listalle kuuluvat paketit lähetetään Turing Test-moduuliin. Muiden pakettien lähtöosoitteet, kohdeosoitteet ja saapumisajat tallennetaan laskentamoduulilla ennen kuin ne lähetetään havaintomoduuliin. (Huang & Huang & Chiang 2013).

Turing Test-moduuliin ohjataan kaikki estetty liikenne. Moduuli valitsee satunnaisesti tekstipohjaisen kysymyksen, johon käyttäjän täytyy vastata oikein päästäkseen jatkamaan kohdeosoitteeseen. Kysymyksen luontimoduuli puolestaan luo satunnaisia tekstipohjaisia kysymyksiä Turing Test-moduulia varten käyttäen LFG-teoriaa. (Huang & Huang & Chiang 2013).

Havaintomoduulin tarkoituksena puolestaan on löytää haitalliset lähdeosoitteet ja lisätä ne estettävien listalle. Sen algoritmissa on neljä tasoa, joista ensimmäisessä moduuli on tark-

kailutilassa. Tällöin se kerää ja tallentaa tietoa virtuaaliklusterien (palvelua tarjoavien virtuaalikoneiden klusterien) käyttäytymisestä. Tällaista tietoa ovat yhteyksien enimmäismäärä ja keskiarvoinen määrä sekunnissa, saapuvat paketit sekunnissa ja saapuvat tavut sekunnissa. Kun tarpeeksi dataa on kerätty, niin siirrytään seuraavalle tasolle. Toisessa tasossa tämänhetkistä tietoa verrataan maksimiarvoihin. Jos arvot ovat suuremmat, niin siirrytään kolmannelle tasolle ja aktivoidaan laskentamoduuli laskemaan virtuaaliklusterin liikennettä. Tasolla kolme etsitään kynnysarvojen, kuten yhteyskertojen määrän, avulla sellaisia IP-osoitteita, jotka siirretään estettyjen listalle. Kun kuluu tietty aika, jolloin IP-osoitteita ei ole siirretty yhtäkään estettyjen listalle, niin siirrytään takaisin tasolle 2 ja kytketään laskentamoduuli pois päältä. Neljännellä tasolla liikenne virtuaaliklusteriin tai siitä poispäin on erittäin korkea (se vie esimerkiksi 90 % kaistanleveydestä). Tasolla 4 analyysit veisivät liikaa resursseja, joten silloin kaikki tulevat yhteydet estetään lisäämällä virtuaaliklusterin julkinen IP-osoite estolistalle lyhyeksi ajaksi. Tällä tavoin liikenteen määrä vähenee, koska uudet yhteydet ohjataan suoraan Turing Test-moduuliin. Virtuaaliklusterin IP-osoite lisätään toistuvasti estolistalle, kunnes liikenteen määrä vähenee tai resursseja kasvatetaan. (Huang & Huang & Chiang 2013).

Vissers ym. ehdottavat artikkelissaan (2014) suodatinmenetelmää HTTP- ja XML-hyökkäyksiltä puolustautumiseen. Tarkoituksena on käyttää suodatinta pyyntöjen käsittelemiseen ennen kuin välittäjä (*cloud broker*) yrittää jakaa käyttäjien pyynnöt palveluiden tarjoajille (*cloud providers*). Suodattimen tarkoituksena on normaalijakaumaan perustuvien mallien (*Gaussian models*) avulla päättää, mikä on jokaisen palvelun profiilin normaalia käyttöä. Aluksi järjestelmä rajoittaa jokaisen käyttäjän pyyntöjen määrää tietyn ajan sisällä, jotta tulvahyökkäys saadaan estettyä. Tämän jälkeen jokaisen pyynnön täytyy läpikäydä ensin HTTP-otsakkeen tarkistus ja vasta sitten XML-dokumentin sisältö jäsennellään ja tarkastetaan. Tässä yhteydessä tarkastetaan myös, että otsake ja XML-dokumentin sisältö vastaavat toisiaan. (Vissers ym. 2014).

4.8 Klusterointiin, entropiaan ja luokitteluun perustuvat havaitsemismenetelmät

Ye, Zheng ja She esittelevät artikkelissaan (2012) klusterointiin (ryhmittelyyn) perustuvan menetelmän sovelluskerroksen palvelunestohyökkäysten havaitsemiseen. He tarkastelevat neljää istunnon piirrettä, joiden perusteella he ryhmittelevät palvelimen istunnot. Nämä piirteet ovat istunnon objektien keskimääräinen koko (*average size of all objects in the session*), istunnon pyyntöjen tahti (*request rate of the session*), istunnon objektien keskimääräinen suosio (*average popularity of all objects in the session*) sekä istunnon objektien keskimääräinen siirtymistodennäköisyys (*average transition probability of objects in the session*). Istunnon objektien keskimääräinen suosio valittiin piirteeksi, koska yleensä sivustoilla tietyt objektit ovat toisia suositumpia, mutta botit eivät tätä tiedä vaan luultavasti ne lähettävät pyyntöjä satunnaisista objekteista. Siirtymistodennäköisyys puolestaan valittiin piirteeksi, koska tilastotieteen näkökulmasta sillä on merkitystä, että joistakin objekteista siirrytään toisiin todennäköisemmin kuin muihin. (Ye & Zheng & She 2012).

Nämä piirteet Ye, Zheng ja She (2012) standardoivat (*normalize*) ja ryhmittelevät käyttäen hierarkkista klusterointia. Etäisyyden mittana ryhmittelyssä käytetään euklidista etäisyyttä ja linkkisääntönä (*linkage rule*) Wardin menetelmää. Artikkelissaan he saavat menetelmällään tulokseksi noin 90 prosentin havaitsemisasteen. (Ye & Zheng & She 2012).

Chwalinski, Belavkin ja Cheng esittelevät artikkelissaan (2013) entropiaan ja suurimman uskottavuuden menetelmään perustuvan klusterointimenetelmän, jolla pyritään luokittelemaan oikein käyttäjien aikeita. He ottavat huomioon kahdenlaiset hyökkäystilanteet: toisessa hyökkääjä vaeltaa satunnaisesti sivuston objektien välillä ja toisessa hyökkääjä valikoi linkit satunnaisesti suosituimpien objektien joukosta. Ensin sivuston objektien määrää vähennettiin yhdistelemällä niitä Hammingin etäisyyden (*Hamming distance*) avulla suosioon perustuviksi kategorioiksi. Saapuvien pyyntöjen sekvenssit luokitellaan klusteroimalla ne siten, että klusterien keskimääräinen entropia on mahdollisimman pieni. Jotta menetelmän havainnointi paranisi, klusteroinnissa muodostuneet epäsuositut yhdistelmät allokoidaan uudestaan olemassa oleviin luokkiin, jolloin luokkien määrä vähenee ja havaitseminen on nopeampaa

ja oikeampaa. Itse havaitsemisjärjestelmä koostuu kahdesta osasta, joista ensimmäisessä havaitaan hyökkäyksen olevan meneillään tarkastelemalla pyyntöjen sekvenssin jakaumaa. Toisessa osassa puolestaan määritetään, mitkä pyynnöistä ovat aitoja ja mitkä liittyvät hyökkäykseen suurimman uskottavuuden menetelmästä (*Maximum Likelihood*) saatavan arvon perusteella. (Chwalinski & Belavkin & Cheng 2013).

Wang, Yang ja Long käyttävät artikkelissaan (2010) palvelunestohyökkäyksen havaitsemiseen suhteellista entropiaa (*relative entropy*), jolla mitataan kahden sekvenssin samankaltaisuutta. Näin he pyrkivät erottamaan suosittuja sivuja käyttävät aidot käyttäjät satunnaisia pyyntöjä tekevistä boteista. Menetelmä koostuu oppimisvaiheesta ja havaintovaiheesta. Pyyntösuhde (*click ratio*) eli kyseisen objektin pyyntökertojen lukumäärää käytetään klusteroinnin pohjana oppimisvaiheessa, jolloin sivut jakautuvat eri klustereihin riippuen niiden suosiosta. Havaintovaiheessa nämä klusterit ja suhteellinen entropia yhdistetään ja saatua suuretta verrataan kynnysarvoon. Kynnysarvoa suuremmat arvot saaneet istunnot luetaan epänormaaleiksi ja ne suodatetaan pois. Normaalit istunnot puolestaan asetetaan palvelujonoon. (Wang & Yang & Long 2010).

Lee, Kim ja Kim ehdottavat artikkelissaan (2011) sekvenssijärjestyksestä riippumatonta (*sequence-order-independent*) ja pääkomponenttianalyysiin (*PCA*) perustuvaa menetelmää. Tässä menetelmässä käyttäjien nettikäyttäytymistä mallinnetaan käyttäen sekvenssijärjestyksestä riippumattomia attribuutteja sivujen pyyntösekvenssien sijasta. Nämä attribuutit ovat käyttäjän pyyntöjen ja pyyntöjen keskiarvon suhde, tiettyjen sivujen osuus kakista käyttäjän pyytämistä sivuista, käyttäjän pyytämien sivujen osuus kaikista palvelimen sivuista sekä suosituimpien sivujen pyyntömäärä suhteessa kokonaispyyntömäärään. Nämä attribuutit muodostavat käyttäjän aktiivisuutta, kiinnostuksenkohteita sekä kiinnostuksen laajuutta ja intensiivisyyttä kuvaavan matriisin. Tästä matriisista johdetaan käyttäytymistä kuvaava malli jakamalla data ensin *k*-keskiarvon klusterointimenetelmällä (*k-means clustering*) ryhmiin ja käyttämällä jokaiselle ryhmälle erikseen pääkomponenttianalyysiä. Parhaimmillaan heidän menetelmällään saavutettiin artikkelissa 95,4 % havaitsemisaste, jolloin väärien positiivisten aste oli 20,9 %. Keskimääräinen havaintoaste oli 86,7 % ja keskimääräinen väärien positiivisten aste oli 4,5 %. (Lee & Kim & Kim 2011).

4.9 Hyökkäyksen havaitseminen SSL/TLS-salatuissa yhteyksissä

Zolotukhin ym. esittelevät artikkelissaan (2015) salatuissa yhteyksissä tapahtuvien palvelunestohyökkäysten havaitsemiseen kaksivaiheista algoritmia, jossa hyökkäysten havaitseminen perustuu paketin otsakkeista saataviin tietoihin (aikaleima, lähteen IP-osoite ja portti, kohteen IP-osoite ja portti, protokolla, paketin ja ikkunan koko, sekvenssinumero, ACK-kuittausnumero, elinaika sekä TCP-kuittaus). (Zolotukhin ym. 2015).

Menetelmän ensimmäisessä vaiheessa käytetään poikkeamien havaitsemisalgoritmia hyökkäysliikennettä sisältävien aikavälien löytämiseen. Jokaisesta aikavälistä lasketaan otosentropia (*sample entropy*) lähteen IP-osoitteelle, lähteen portille, kohteen IP-osoitteelle ja kohteen portille. Lisäksi lasketaan kaikkien voitten (*flow*) kokonaismäärä, keskimääräinen vuon kesto, pakettien keskimääräinen määrä vuossa, pakettien keskimääräinen koko sekä TCP-ikkunan keskimääräinen koko. Näistä lasketaan χ^2-arvot, jotka jaetaan kahteen klusteriin yhden sidoksen klusterointimenetelmällä (*single-linkage clustering*). Toisessa klusterissa ovat tällöin normaalit arvot ja toisessa klusterissa poikkeavat arvot (*outliers*). Poikkeavat arvot poistetaan ja normaaleja arvoja käytetään hyökkäysten havaitsemiseen, jolloin normaaliklusterista χ^2-arvon perusteella liian kaukaiset aikavälit ovat hyökkäysliikennettä sisältäviä. (Zolotukhin ym. 2015).

Toisessa vaiheessa puolestaan saadut aikavälit analysoidaan, jotta havaitaan hyökkäykseen yhteydessä olevat vuot. Tätä varten otsakkeista saatavat tiedot standardoidaan ja normaalin käyttäytymisen malli muodostetaan käyttämällä tiheyteen perustuvaa DBSCAN-klusterointimenetelmää (*density-based spatial clustering of applications with noise*). Artikkelin perusteella menetelmän oikeiden positiivisten aste oli 100 %, väärien positiivisten aste 0,07 % ja tarkkuus melkein sata prosenttia (99,9993 %). (Zolotukhin ym. 2015).

4.10 Markov-prosesseihin ja satunnaiskulkuun perustuvat havaitsemismenetelmät

Chuan ym. esittelevät artikkelissaan (2014) satunnaiskulkuun (*random walk*) perustuvan palvelunestohyökkäyksen havaitsemismenetelmän, joka on tarkoitettu asymmetristen hyökkäysten havaitsemiseen. Menetelmä toimii siten, että ensimmäisenä tunnistetaan jokaisen käyttäjän pyynnöt kokonaisliikenteestä, jotta saadaan muodostettua pyyntösekvenssi eli peräkkäisten sivupyyntöjen sarja. Seuraavaksi muodostetaan yhdistetty sivujen siirtymämatriisi, jolle opetetaan kaikki hyväksytyt käyttäjien pyyntösekvenssit. Tämän jälkeen jokaiselle käyttäjälle rakennetaan satunnaiskulkumalli, joka koostuu sivujen pyyntösekvenssistä tarkkailun aikana. Satunnaiskulkua kuvataan suunnattuna graafina, joka koostuu itse sivuista (solmuista) ja tilasiirtymistä sivujen välillä (poluista). Satunnaiskulkumallia toistetaan eli opetetaan niin kauan kunnes se konvergoi, jonka jälkeen sitä voi käyttää yhdessä siirtymämatriisin kanssa ennustamaan käyttäjän seuraavia pyyntöjä. Lopuksi lasketaan ennustetun ja havaitun pyyntösekvenssin välinen samankaltaisuus, jotta voidaan arvioida, onko kyseessä hyökkäys vai normaali käyttäjä. Samankaltaisuuden mittana käytetään Jacobin kerrointa (*Jacobi coefficient*) eli kahden sekvenssin samojen sivupyyntöjen määrää suhteessa sivujen kokonaismäärään. Mikäli kerroin on pienempi kuin ennalta määritelty kynnysarvo, on kyseessä hyökkääjä. Näin ollen valittu kynnysarvo vaikuttaa menetelmän tehokkuuteen. Parhaimmillaan he saavuttivat menetelmällään artikkelissaan 96 % havaitsemissuhteen, jolloin väärien negatiivisten suhde oli 2.1 %. (Chuan ym. 2014).

Limkar ja Jha esittelevät artikkelissaan (2012) Markovin piilomallia (*Hidden Markov Model, HMM*) käyttävän menetelmän sovelluskerroksen DDoS-hyökkäysten havaitsemiseen. Menetelmä koostuu harjoitusvaiheesta ja havaintovaiheesta. Harjoitusvaiheessa rakennetaan ja opetetaan malli käyttämällä harjoitusdataa. Havaintovaiheessa puolestaan käyttäjien pyyntösekvenssejä verrataan malliin ja epänormaalit käyttäjät laitetaan CAPTCHA-testiin. Markovin piilomallia käytetään käyttäjän käyttäytymisprofiilin muodostamiseen viimeisimmän kuukauden aikaisen käytöksen perusteella. Tämän jälkeen saman käyttäjän saapuvaa pyyntöä ja viimeisintä pyyntöä verrataan profiileihin, jolloin saadaan prosenttiluku kuvaamaan

tiettyyn malliin kuulumisen todennäköisyyttä. Jokaiselle erilaiselle käyttäjäryhmälle rakennetaan riippumattomat Markovin piilomallit, jotta tehokkuus lisääntyy ja saadaan kuvattua käyttäjien käytös mahdollisimman monipuolisesti. (Limkar & Jha 2012).

Wang, Yang ja Long esittelevät artikkelissaan (2011) kaksi käyttäjien web-käyttäytymistä mallintavaa mallia, joista toinen perustuu klikkausten suhteeseen (*click-ratio*) ja toinen Markov-prosesseihin. Näihin malleihin verrataan palvelun käyttäjien käytöstä käyttämällä suurten hajontojen teoriaa (*large deviation theory*), josta saatava suuren hajonnan todennäköisyys on suurempi normaaleilla käyttäjillä kuin hyökkääjillä. Heidän artikkelissaan klikkausten suhteeseen perustuva malli pystyi havaitsemaan kaikki hyökkääjät väärien positiivisten asteen ollessa 2 %. Vastaavasti Markov-prosesseihin perustuva malli pystyi havaitsemaan noin 61 % hyökkääjistä väärien positiivisten asteen ollessa 2%. (Wang & Yang & Long 2011).

4.11 Internetin vuon analysointi

Giralte ym. esittelevät artikkelissaan (2013) internetin vuon (*flow*) analysointiin perustuvan menetelmän, jolla pystytään havaitsemaan HTTP-palvelimeen kohdistuvat tietynlaiset palvelunestohyökkäykset. Näihin kuuluvat hyökkäykset, joissa pyydetään satunnaisesti eri sivuja (*random walking*), pyydetään toistuvasti sisäänpääsyä tietokantaan (*login access*), käytetään hakutoimintoa suorittaakseen suuria määriä tietoa hakevia komentosarjoja (*search access*) tai joissa käytetään hyväkseen POST-toimintoa. Järjestelmän tarkoituksena on havaita palvelunestohyökkäys käyttämällä internetin vuohon kolmea peräkkäistä analyysiä, jotka ovat tilastoanalyysi, HTTP-graafianalyysi (*HTTP graph cache analysis*) ja HTTP-polkuvälimuistien analyysi (*HTTP path cache analysis*). (Giralte ym. 2013).

Järjestelmä koostuu päivitystilasta (*update mode*) ja havainnointitilasta (*detection mode*). Päivitystilassa päivitetään analyyseihin tarvittavat tilastotiedot, polkugraafit sekä välimuistit ja tähän tilaan pääsevät vain tietyt IP-osoitteet. Näiden IP-osoitteiden on sijaittava samassa verkossa palvelimen kanssa, jotta saadaan mitattua nopeimmat vasteet ja pienimmät viiveet. Havainnointitilassa suoritetaan kaikelle liikenteelle analyysit, joiden avulla päätetään, onko kyseessä normaali vai epäilyttävä käyttäjä. (Giralte ym. 2013).

Ensimmäisenä jokaiselle käyttäjälle suoritetaan tilastoanalyysi käyttäen sovelluskerroksen yleisiä mittoja, kuten pakettien määrää vuossa, voitten määrää per käyttäjä tai pyyntöjen määrää per käyttäjä. Näitä mittoja verrataan keskiarvoon keskihajonta huomioiden. Mikäli käyttäjä eroaa keskiarvosta, merkataan hänet epäilyttäväksi. (Giralte ym. 2013).

Seuraavaksi tehdään HTTP-graafianalyysi. Tässä menetelmässä analysoidaan HTTP-vuo, joista jokaiselle järjestelmä rekisteröi eri pyynnöt ja niiden välisen ajan. Jokaiselle sallitulle HTTP-polulle (*path*) tallennetaan välimuistiin kahden peräkkäisen URL-pyynnön välinen minimiaika, johon vuota verrataan. Jos pyyntöjen sekvenssi ei vastaa mitään tallennettua polkua tai pyyntöjen välinen aika on minimiä pienempi, merkitään käyttäjä epäilyttäväksi. (Giralte ym. 2013).

Lopuksi tehdään HTTP-polkuvälimuistien analyysi, jossa lasketaan yhden HTTP-polun toistot yhtä vuota tai yhtä käyttäjää kohden. Jos polku toistuu ennalta määrättyä kynnysarvoa useammin, epäillään kyseessä olevan botti ja käyttäjä merkataan epäilyttäväksi. Jos jokin näistä kolmesta analyysistä havaitsee epätavallista käytöstä, käyttäjä merkitään epäilyttäväksi ja hänestä voidaan tarvittaessa raportoida ulkopuoliselle torjuntajärjestelmälle. Tällöin myös muut jäljellä olevat analyysit jätetään kyseisen käyttäjän tapauksessa käyttämättä. (Giralte ym. 2013).

Zhang, Zhang ja Fan esittelevät artikkelissaan (2012) datan yhdistämiseen voiksi (*flows*) ja näiden voitten arvioimiseen perustuvan sovelluskerroksen DDoS-hyökkäyksen havaitsemismenetelmän. Data yhdistetään vuoksi käyttämällä hyväksi pintapiirteitä (*surface characteristics*), keskimääräistä skannausaikaa (*scan time*) sekä sivujen pyyntösekvenssejä. Tämän jälkeen lasketaan Dempsterin-Shaferin teoriaa (DST) käyttäen, kuinka todennäköisesti vuo on hyökkäys ja verrataan todennäköisyyttä ennalta määrättyyn kynnysarvoon. Menetelmällä he saavuttivat artikkelissaan väärien positiivisten asteeksi 4,5 % ja väärien negatiivisten asteeksi 5,7 %. (Zhang & Zhang & Fan 2012).

4.12 Selailukäytökseen, siirtymätodennäköisyyteen ja luottamukseen perustuvat menetelmät

Beitollahi ja Deconinck esittelevät artikkelissaan (2012b) kehittämänsä ConnectionScore-tekniikan. Menetelmä perustuu palvelimen käyttäjien normaalia käyttäytymistä ja piirteitä esittäviin tilastollisiin attribuutteihin, joita palvelin mittaa normaaliolosuhteissa käyttäjistä sekä heidän liikenteestään. Attribuutit ovat sivustosta riippuvaisia, jolloin hyökkääjä ei voi niitä etukäteen tietää. (Beitollahi & Deconinck 2012b).

Beitollahi ja Deconinck antavat artikkelissaan esimerkkejä mahdollisista attribuuteista, joita ovat pyyntöjen tahti (*request rate*), latausten tahti (*download rate*), yhteysaika (*uptime*), yhteydetön aika (*downtime*), selailukäytös (*browsing behavior*), IP-osoitteiden lähteiden jakauma (*source IP address distribution*) sekä käyttäjien saapumistahdin jakauma (*arrival distribution rate of users*). Selailukäytöstä kuvataan luokittelemalla sivut eri kategorioihin ja mittaamalla, millaisella tahdilla käyttäjät eri kategorioita selaavat. Tämän lisäksi selailukäytöstä tarkastellaan mittaamalla sivujen käyttöastetta ja suosiota, koska noin 90 % käyttäjistä käyttää vain noin 10 % kaikista sivuston sivuista. Lisäksi voidaan tarkkailla, mitä osaa hyperlinkeistä pyydetään ja montaako peräkkäistä hyperlinkkiä pyydetään (puhutaan syvyydestä, *hyperlink depth*). IP-osoitteiden puolestaan oletetaan jakautuvan tasaisesti normaalin liikenteen tapauksessa ja keskittyvän klustereiksi hyökkäyksen aikana ja käyttäjien saapumistahdin oletetaan noudattavan Poisson-jakaumaa. (Beitollahi & Deconinck 2012b).

Palvelin käyttää mitattuja attribuutteja vertailuprofiilina hyökkäyksen aikana ja pisteyttää yhteydet sen perusteella. Mitä pienemmät pisteet yhteys saa, sitä todennäköisemmin kyseessä on hyökkääjän muodostama yhteys. Kun yhteydet on pisteytetty, palvelin voi ottaa uudelleenkäyttöön pullonkaulan aiheuttavia resursseja alimmat pisteet saaneilta yhteyksiltä, kunnes kuormitus on saatu laskettua kynnystasolle. (Beitollahi & Deconinck 2012b).

Ye ja Zheng esittelevät artikkelissaan (2011) menetelmän aitojen käyttäjien erottelemiseen satunnaisia pyyntöjä tekevistä boteista. Heidän oletuksenaan on, että normaalit käyttäjät valitsevat kiinnostavia sivuja ja objekteja, jolloin heidän pyynnöistään muodostuu peräkkäinen sarja, jota he kutsuvat siirtymätodennäköisyydeksi. Botit puolestaan pyytävät sivuja satunnaisesti, jolloin niiden pyynnöt eivät juurikaan korreloi keskenään. He käyttävät käyttäjien

erottelemiseen kynnysarvoa, joka vaihtelee riippuen käytetystä intervallista. Vertailu tapahtuu tunnusluvun avulla, joka koostuu lukumäärävektorista (*frequency vector*), siirtymätodennäköisyysmatriisista (*transition probability matrix*) ja isäntäkoneen pyyntösarjan todennäköisyydestä (*host request sequence probability*). Lukumäärävektori kuvaa kaikkien objektien suosiota ja siirtymätodennäköisyyksien matriisi sisältää tiedon sivujen siirtymätodennäköisyyksistä. Isäntäkoneen pyyntösarjan todennäköisyys puolestaan kuvaa pyyntöjen sarjan siirtymätodennäköisyyden keskimääräistä todennäköisyyttä. Mikäli tunnusluku alittaa kynnysarvon, käyttäjä merkitään botiksi. Havaitut botit estetään, jolloin palvelinta suojellaan ylikuormittumiselta. (Ye & Zheng 2011).

Yu ym. ehdottavat artikkelissaan (2010) istuntotulvahyökkäysten havaitsemiseen ja niiltä suojautumiseen luottamuksenhallintamenetelmää eli TMH:ta. Aitoja käyttäjiä mallinnetaan tiheysjakaumalla (*density distribution*), joka muodostuu pyyntöjen välisestä ajasta ja todennäköisyydestä vierailla uudestaan tietyn ajan kuluessa. Mallia päivitetään aidoilta käyttäjiltä kerättyjen tietojen perusteella. (Yu ym. 2010).

Menetelmässä jokaiselle luodulle yhteydelle tallennetaan neljä luottamuksen puolta, jotka ovat lyhytaikainen luottamus (*short-term trust*), pitkäaikainen luottamus (*long-term trust*), negatiivinen luottamus (*negative trust*) ja väärinkäytetty luottamus (*misusing trust*). Lyhytaikainen luottamus mittaa käyttäjän viimeaikaista käytöstä ja sen avulla tunnistetaan hyökkäyksen aikana paljon pyyntöjä lähettävät käyttäjät. Pitkäaikainen luottamus puolestaan mittaa pitkäaikaista käytöstä ja erottaa toisistaan ne käyttäjät, joilla on normaali vierailuhistoria ja joilla on epänormaali vierailuhistoria. Negatiivinen luottamus on kumuloituvaa epäluottamusta käyttäjää kohtaan, jota kertyy, kun käyttäjän kokonaisluottamus putoaa tietyn kynnysarvon alle. Väärinkäytetty luottamus taas mittaa sellaisen käyttäjän epäilyttävää käytöstä, joka väärinkäyttää kumuloitunutta mainettaan. (Yu ym. 2010).

Näitä neljää suuretta käytetään laskemaan kokonaisluottamus, jonka perusteella päätetään, hyväksytäänkö käyttäjän seuraava yhteyspyyntö. Nämä neljä puolta säilötään evästeenä käyttäjälle lisenssiksi, jonka asiakas liittää yhteyspyyntöön aina palatessaan uudelleen sivustolle. TMH laskee lisenssin arvojen perusteella kokonaisluottamuksen, päivittää lisenssin ja

tekee päätöksen käyttäjän pyynnön hyväksymisestä. Lisenssi on suunniteltu siten, että sen väärentäminen on hankalaa, mutta sen avulla käyttäjä on helppo tunnistaa. (Yu ym. 2010).

5 DDoS-hyökkäyksen torjuminen

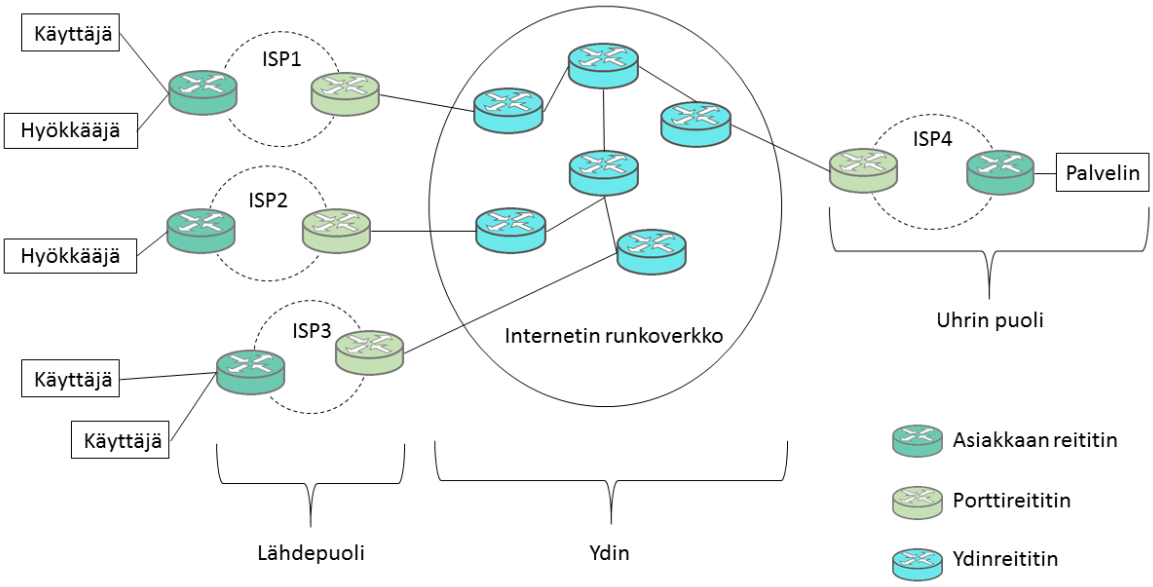
Tässä luvussa tarkastellaan yleisesti, miten hajautetuilta palvelunestohyökkäyksiltä voidaan suojautua sekä esitellään lyhyesti muutamia ohjelmia ja palveluita hyökkäysten torjumiseen.

5.1 Yleisiä ohjeita

Beitollahi ja Deconinck analysoivat artikkelissaan (2012a) erilaisia puolustautumiskeinoja hajautettuja palvelunestohyökkäyksiä vastaan. Heidän mukaansa puolustautumismenetelmää suunnitellessa, käyttöönottaessa tai arvioidessa täytyisi käsitellä seuraavat viisi kohtaa:

1) Hyökkäyksen havaitsemispaikka: hyökkäys voidaan havaita kahdessa paikassa, jotka ovat uhrin palvelin sekä liikenteen lähtöpistettä, ydinreitittimiä (*core routers*) tai uhria lähellä olevat reitittimet.
2) Hyökkäyksen havaitsemismenetelmä
3) Vastamekanismi: Vastamekanismina voi olla joko liikenteen suodattaminen tai liikenteen tahdin rajoittaminen. Liikenteen suodattaminen tapahtuu joko suodattamalla kaikki hyökkäysliikenteeksi laskettu liikenne kokonaan tai päästämällä läpi kaikki hyväksytyksi liikenteeksi laskettu liikenne ja estämällä muu liikenne kokonaan. Liikenteen tahdin rajoittaminen puolestaan voi tapahtua joko rajoittamalla hyökkäysliikennettä tai kaikkea uhrille päin kulkevaa liikennettä.
4) Vastamekanismin käyttöpaikka: Internetin rakenne sallii vastamekanismeja käytettävän neljässä eri pisteessä, jotka ovat uhrin puoli, ydin, lähdepuoli tai hajautetusti. Kuviosta 5 nähdään, että lähdepuoleen kuuluvat asiakkaiden reitittimet (*customer edge routers*) ja palveluntarjoajien porttireitittimet (*gateway routers*), jotka sijaitsevat lähellä liikenteen lähtöpistettä. Ydinreitittimet (*core routers*) puolestaan viittaavat internetin runkoverkon (*Internet backbone*) korkean suorituskyvyn reitittimiin. Uhrin puoleen puolestaan kuuluvat ne pisteet, jotka ovat lähellä uhria. Näitä ovat uhrin reititin, uhrin palvelin sekä näihin yhteydessä oleva porttireititin. Vastamekanismi on hajautettu silloin, kun se sijaitsee useammassa näistä kolmesta pisteestä.
5) Valvontakeskuksen sijainti: Valvontakeskus voi sijaita joko lähdepuolella, ytimessä tai uhrin puolella. Valvontakeskuksella tarkoitetaan niitä pisteitä, joissa määritellään

suodattimen säännöt ja tahdin rajoitusten määrät sekä aloitetaan tai pysäytetään puolustusmekanismit.



Kuvio 5. Vastamekanismin käyttöpaikkana voi olla lähdepuoli, ydin, uhrin puoli tai jokin näiden yhdistelmä.

Lähdepuolella sijaitsevissa pisteissä on hyvä käyttää puolustusmekanismeja, jotka rajoittavat tahtia tai suodattavat hyökkäysliikennettä. Ytimessä sijaitsevissa pisteissä puolestaan kannattaa puolustuskeinona vain rajoittaa kaikkea liikennettä. Uhrin puolen pisteissä taas on parasta havaita ja erotella hyökkäysliikenne normaalista liikenteestä. Paras suoja palvelunestohyökkäyksiä vastaan saadaan hajauttamalla vastamekanismit lähdepuolen, ytimen ja uhrin puolen pisteisiin. Tällöin uhrin puolella voidaan havaita hyökkäysliikenne, erotella hyökkäysliikenne normaalista liikenteestä ja pyytää ylempiä reitittimiä (esimerkiksi lähdepuolen reitittimet) rajoittamaan tahtia tai suodattamaan liikennettä. Tällainen ratkaisu vaatii yhteistyötä useilta palveluntarjoajilta ja domain-verkkotunnusten haltijoilta. Tämän lisäksi täytyy varmistaa kommunikointikanavien tietoturva ja autentikointi. (Beitollahi & Deconinck 2012a).

Palvelunestohyökkäysten puolustustekniikat voidaan luokitella reaktioajan perusteella selviytymistekniikoihin (*survival*), ennakoiviin tekniikoihin (*proactive*) sekä reaktiivisiin tekniikoihin (*reactive*). Selviytymistekniikat ehdottavat, että uhrin palvelin torjuu palvelunestohyökkäykset suurentamalla pullonkaulana olevia resursseja. Ennakoivat tekniikat puolestaan pyrkivät estämään palvelunestohyökkäysten tekemisen muokkaamalla esimerkiksi porttireitittimiä tai ydinreitittimiä. Reaktiiviset tekniikat taas taistelevat palvelunestohyökkäyksiä vastaan, kun hyökkäys on jo tapahtunut. (Beitollahi & Deconinck 2012a).

Selviytymistekniikat ovat uhrin puolen puolustautumismenetelmiä, joissa uhri vastaa DDoS-hyökkäykseen suurentamalla resursseja (esimerkiksi kaistanleveys, muisti, prosessorin teho tai TCP puskurit) joko dynaamisesti tai staattisesti. Resurssien kasvattamisesta seuraa, että uhri voi käsitellä suurempaa määrää liikennettä ilman, että normaalikäyttäjien liikenne häiriintyy. Ennakoivissa tekniikoissa puolestaan palvelunestohyökkäys estetään ennen kuin hyökkäys vahingoittaa uhria. Tällöin uhri ei huomaa olleensa DDoS-hyökkäyksen kohteena. Ennakoivissa tekniikoissa reitittimien tehtävänä on havaita ja suodattaa haitallista liikennettä. Reaktiivisissa tekniikoissa uhri havaitsee hyökkäyksen ja vastaa siihen sopivalla tavalla. Suurin osa DDoS-hyökkäysten puolustautumismenetelmistä kuuluu tähän kategoriaan. (Beitollahi & Deconinck 2012a).

Malecki on artikkelissaan (2012) esitellyt yksinkertaisia keinoja, joilla IT-osastot voivat varautua palvelunestohyökkäyksiin. IT-osaston pitäisi esimerkiksi tuntea palveluntarjoajansa ja tehdä heidän kanssaan yhteistyössä tehokas vastaussuunnitelma (*response plan*), koska palveluntarjoajat saattavat olla etulinjassa palvelunestohyökkäyksiltä suojauduttaessa. IT-osaston tulisi hänen mukaansa myös tunnistaa järjestelmän pullonkaulat eli ne kohdat, jotka todennäköisimmin ylikuormittuvat. Näitä ovat esimerkiksi palomuuri, tunkeutujien esto- tai havaitsemisjärjestelmät, kuorman tasaajat (*load balancers*) tai palvelimet. Lisäksi IT:n tulee valvoa näitä pullonkaulakohtia ja tarvittaessa päivittää ja optimoida suorituskykyään (*performance*) ja kestävyyttään (*resilience*). IT-osaston tulisi myös skannata ja valvoa sekä sisään- että ulospäin suuntautuvaa (myös sovelluskerroksen) liikennettä, jotta epätavalliset käyttäjämäärät tai käyttäytymismallit voitaisiin tunnistaa. (Malecki 2012).

Seuraavilla keinoilla voi yrittää valmistautua palvelunestohyökkäyksiin ja niiden torjumiseen (Malecki 2012):

- Sopivien vastatoimina toimivien tuotteiden ja palveluiden arviointi ja käyttöönotto (esimerkiksi seuraavan sukupolven palomuurit, joihin on integroitu tunnettujen palvelunestohyökkäysten havainnointiin ja torjuntaan liittyvät toiminnot ja jotka voidaan automaattisesti päivittää).
- Sellaisen palomuurin käyttäminen, joka syväluotaa (*deep scan*) sekä sisään tulevan että lähtevän liikenteen ja mahdollistaa näkyvyyden itse ohjelmistoon sekä lisäksi valvoo ja ilmoittaa epäilyttävästä liikenteestä. Palomuurin tulisi myös mahdollistaa palvelunestohyökkäyksen korjaus estämällä, suodattamalla tai uudelleenohjaamalla liikennettä havaittujen mallien, määrien tai piirteiden perusteella.
- Lisäämällä liikenteen vuon analysointiin käytettäviä ohjelmia, jotka voivat tutkia ohjelmien tai käyttäjien datankäyttöä, tarkastella dataa eri aikaväleillä sekä verrata eri lähteiden dataa.
- Uusien teknologioiden ja tekniikoiden seuraaminen ja niiden käyttöönotto.
- Ylimääräisten, hyvällä suorituskyvyllä varustettujen komponenttien lisääminen.
- Firman käytäntöihin perustuvan kaistanleveyden valvonnan lisääminen.

5.2 Ennaltaehkäisyyn ja suojautumiseen käytettäviä menetelmiä

Tässä luvussa esitellään ACL ja IDMS, jotka ovat tunnettuja suojausmenetelmiä hajautetulta palvelunestohyökkäykseltä. Näiden kahden menetelmän lisäksi esitellään ennakoivaan puolustautumistekniikkaan perustuva SeVen-järjestelmä.

ACL:t ovat kaikkiin tulokohtiin (*entry point*) yksityisen verkon ja internetin välillä sijoitettuja tietoturvakomponentteja. Niiden tarkoituksena on tarkastaa jokaisen sisään tulevan ja ulosmenevän paketin tietokentät ja noudattaa niiden suhteen verkon ennalta määrättyjä sääntöjä. ACL:n jokainen sääntö sanelee, mihin paketin otsakkeen arvoista sääntö liittyy ja mitä säännön täyttävälle paketille tehdään. Tyypillisesti ACL:t ovat neljädimensioisia koostuen

lähteen IP-osoitteesta, kohteen IP-osoitteesta, kohdeportista ja protokollatyypistä. Myös viisidimensioiset ACL:t ovat yleisiä ja niiden viides dimensio on lähteen porttinumero. Kun paketti saapuu ACL:ään, verkkolaite etsii ensimmäisen eli suurimman prioriteetin säännön, johon paketti täsmää ja toteuttaa sen. (Liu & Torng & Meiners 2011).

IDMS on älykäs hajautettujen palvelunestohyökkäysten lieventämiseen tarkoitettu järjestelmä, joka pystyy puolustautumaan sekä sovelluskerrosten palvelunestohyökkäyksiltä että alempien kerrosten volumetrisiltä (*volumetric*) hyökkäyksiltä (yleensä TCP-, UDP- tai ICMP-tulva). IDMS:n tulee olla tilaton eli se ei saa seurata kaikkien yhteyksien tilaa, lisäksi sen on oltava joustava (tukea eri konfiguraatioita) sekä helposti laajennettavissa. Lisäksi sen täytyy olla integroitu ja tukea hajautettuja havaintamenetelmiä sekä mahdollistaa useiden vastatoimien käyttö. (Networks Training: Intelligent DDoS Mitigation Systemns-IDMS). Arbor Networksin tekemän tutkimuksen mukaan suurin osa palveluntarjoajista käyttää IDMS-järjestelmää palvelunestohyökkäyksen lieventämiseen ennemmin kuin ACL:ää. (Arbor Networks 2016).

Dantas, Nigam ja Fonseca esittelevät artikkelissaan (2014) hyökkäykseltä puolustautumiseen SeVen-järjestelmän, joka perustuu ASV-menetelmään. Järjestelmä koostuu kahdesta puskurista, joista toisessa on osittain prosessoidut pyynnöt ja toisessa vastaanotetut sekä prosessointia odottavat pyynnöt. Mikäli puskuri täyttyy, täytyy valita, mitkä pyynnöt tiputetaan pois. Vaihtoehtoina on pitää puskurissa olevat pyynnöt (uudet pyynnöt tiputetaan) tai ottaa vastaan uusi pyyntö ja tiputtaa puskurissa oleva pyyntö. Uuden pyynnön vastaanottaminen päätetään generoimalla satunnaisluku ja vertaamalla sitä prosessointia odottavien pyyntöjen lukumäärän avulla laskettuun todennäköisyyteen. Mikäli uusi pyyntö otetaan vastaan, päätetään tasajakauman avulla, mikä pyyntö puskurissa pudotetaan pois. (Dantas, Nigam & Fonseca 2014).

5.3 Tarjolla olevia ohjelmia ja palveluita DDoS-hyökkäyksiltä suojautumiseen

Internetistä löytää paljon eri yhtiöitä, jotka tarjoavat suojautumispalveluita tai ohjelmistoja yrityksille. Tähän on koottu esittelymielessä joitakin tunnetuimpien tietoturvayhtiöiden järjestelmiä, jotka on tarkoitettu sovelluskerroksella tapahtuvilta palvelunestohyökkäyksiltä suojautumiseen. Jokaisesta palvelusta on lyhyt kuvaus sekä linkki yhtiön kotisivuille, josta lisätietoa on voinut tutkielman tekohetkellä löytää.

Snort on Ciscon kehittämä, sääntöihin perustuva, avoimen lähdekoodin tunkeutumisenestojärjestelmä, jolla voidaan analysoida liikennettä ja kirjata paketteja reaaliaikaisesti. Se yhdistää allekirjoitukseen, protokolliin ja poikkeuksiin perustuvat havaitsemismenetelmät. Snort on kaikkein laajimmin käytössä oleva tunkeutumisenestojärjestelmä maailmassa. Sitä on ladattu 4 miljoonaa kertaa ja sillä on satoja tuhansia rekisteröityneitä käyttäjiä. Lisätietoa löytyy Ciscon kotisivuilta: http://www.cisco.com/c/en/us/products/collateral/security/brief_c17-733286.html. (Viitattu 14.5.2016).

Incapsula DDoS Protection on Impervan kehittämä palvelu, jolla pystytään torjumaan suurin osa eri tyyppisistä palvelunestohyökkäyksistä, kuten verkko- ja sovelluskerroksella tapahtuvat sekä DNS-palveluihin kohdistuvat hyökkäykset. Se koostuu maailmanlaajuisesti sijoiteltujen datakeskuksien verkostosta ja jatkuvasti saatavilla olevasta valvontatiimistä. Palvelu jakautuu kolmeen osaan, jotka ovat verkkosivujen suojaus, infrastruktuurin suojaus sekä nimipalvelimien suojaus. Incapsula käyttää välityspalvelimia suojauksessaan. Tämä tarkoittaa sitä, että pyynnöt kulkevat aina ensin Incapsulan palvelimen kautta. Lisätietoa löytyy Incapsulan kotisivuilta: https://www.incapsula.com/ddos-protection-services.html. (Viitattu 14.5.2016).

Kona Site Defender on Akamain kehittämä monikerroksinen palvelunestohyökkäysten torjuntajärjestelmä, joka toimii sekä verkkokerroksen että sovelluskerrosten palvelunestohyökkäyksiä vastaan. Torjunta perustuu niin sanottuihin Kona sääntöihin (*Kona rules*), joita Akamai päivittää säännöllisesti. Lisätietoa löytyy Akamain kotisivuilta: https://www.akamai.com/us/en/resources/ddos-mitigation.jsp. (Viitattu 14.5.2016).

SiteProtect on Neustarin kehittämä pilviperustainen palvelunestohyökkäyspalvelu, jonka voi saada kahdentyyppisenä: tarvittaessa tai hybridinä. Suojaus voidaan toteuttaa käyttämällä hyväksi pelkästään pilveä, jonka kautta liikenne seulotaan hyökkäysten varalta käyttäen erilaisia teknologioita, kuten DNS-uudelleenohjaus ja BGP-uudelleenohjaus. Palvelu aktivoidaan tällöin tarvittaessa. Vaihtoehtona on hybridisuojaus, jossa pilvipalveluun yhdistetään aina aktiivisena oleva lokaali laitteisto. Laitteiston kapasiteetin ylittyessä käyttöön otetaan myös pilvipalvelu. Lisätietoa löytyy Neustarin kotisivuilta: https://www.neustar.biz/services/ddos-protection (Viitattu 14.5.2016).

Peakflow on Arbor Networksin ohjelma, jolla internetin toimijat voivat suojautua palvelunestohyökkäyksiltä. Peakflow kerää, kokoaa yhteen ja analysoi paketteja, NetFlow:ta, SNMP-protokollaa ja BGP-reititystä. Peakflow estää ja poistaa hyökkäysliikenteen pakettivirrasta erillisen valvontamoduulin avulla sekä lähettää aidon liikenteen oikeaan kohteeseen. Lisätietoa löytyy Arbor Networksin kotisivuilta: https://www.arbornetworks.com/images/DS_PeakflowSolution_EN2015.pdf. (Viitattu 14.5.2016).

APS on myös Arbor Networksin kehittämä sovelluskerrosten palvelunestohyökkäysten torjuntaan tarkoitettu ohjelma, joka pystyy tarkastamaan myös SSL-salatun liikenteen hyökkäysten varalta. He tarjoavat myös ohjelman hallintapalvelua mAPS, jolloin valvonta hoituu heidän yrityksensä kautta. Lisätietoa löytyy Arbor Networksin kotisivuilta: https://www.arbornetworks.com/ddos-protection-products/arbor-aps. (Viitattu 14.5.2016).

Arbor Networksillä on myös **Cloud DDoS Service**-niminen pilvipalvelu, joka tarjoaa monikerroksista suojaa palvelunestohyökkäyksiä vastaan. Monikerroksisuus syntyy siitä, että osa hyökkäysten torjumisesta tapahtuu pilvessä (esimerkiksi tulvahyökkäykset) ja osa paikan päällä (esimerkiksi sovelluskerroksen hyökkäykset). Ohjelma perustuu reaaliaikaiseen valvontaan ja epäilyttävästä liikenteestä ilmoittamiseen ja sen käyttämiä tekniikoita ovat muun muassa tahdin rajoitus sekä IP-osoitteiden estäminen. Lisätietoa löytyy Arbor Networksin kotisivuilta: https://www.arbornetworks.com/ddos-protection-products/arbor-cloud. (Viitattu 14.5.2016).

6 SSL DDoS-hyökkäyksen simulointi

Tässä luvussa esitellään salatun palvelunestohyökkäyksen (SSL DDoS-hyökkäys) simulointi laboratorioympäristössä sekä analysoidaan sen havaitsemista pelkästään otsakkeista saatavien tietojen avulla.

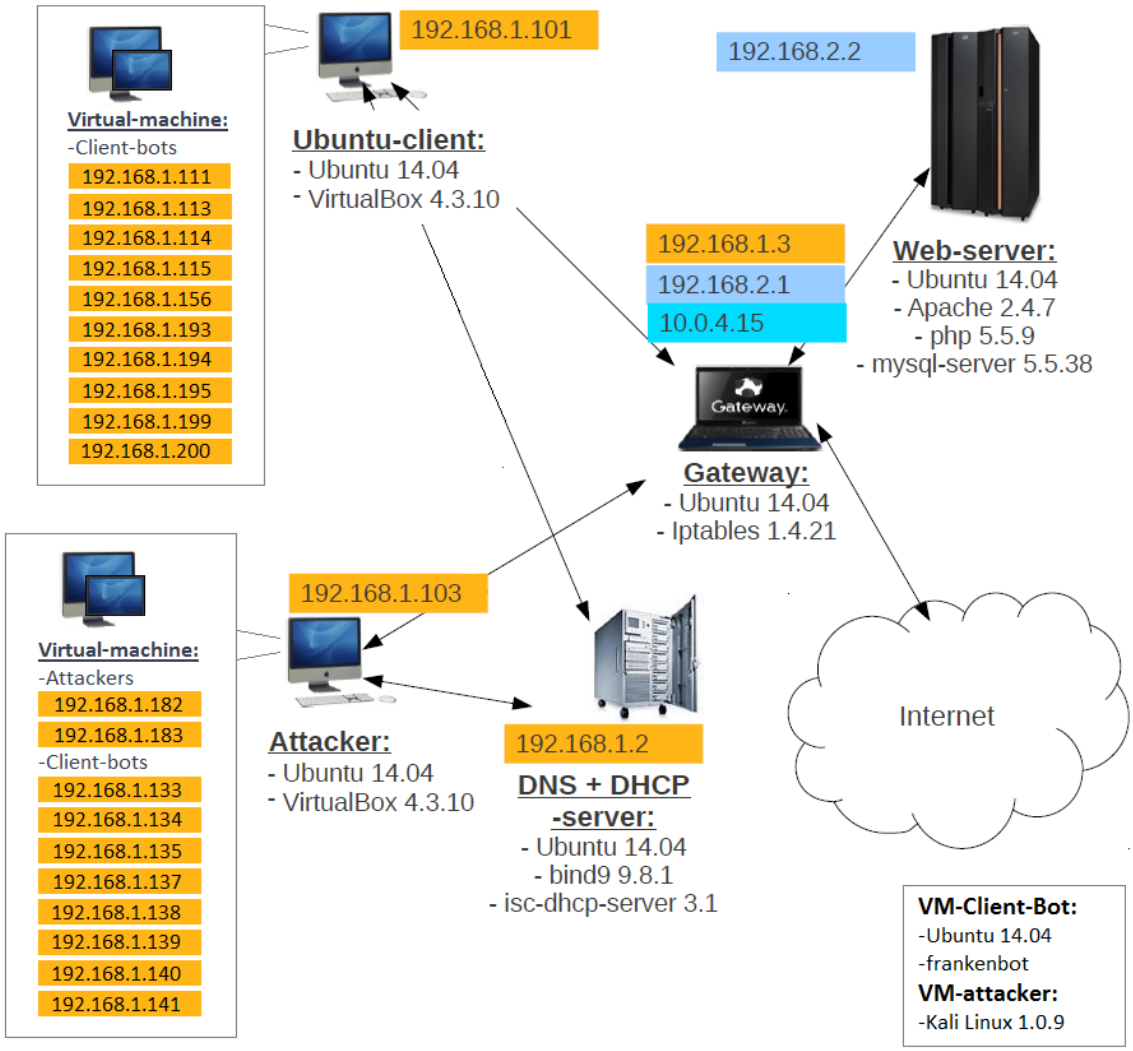
6.1 Ympäristö ja käytetyt ohjelmat

Hajautetun palvelunestohyökkäyksen simulointi suoritettiin Jyväskylän yliopiston tietoliikennelaboratoriossa, jossa käytössä oli 5 Linux-käyttöjärjestelmällä toimivaa konetta. Nämä koneet olivat asiakaskone (Ubuntu-Client), hyökkääjäkone (Attacker), yhdyskäytävä (Gateway), web-palvelin (Web-server) ja DNS/DHCP-palvelin (DNS+DHCP-server). Ympäristön rakenne ja oleellisimpia käytettyjä ohjelmia on havainnollistettu kuviossa 6. Asiakaskoneella oli 10 virtuaalikoneella (VirtualBox 4.3.10) luotua Ubuntu-asiakasta, jotka generoivat aidonoloista käyttäjäliikennettä frankenbot-sovelluksen avulla. Hyökkääjäkoneella oli lisäksi 8 vastaavanlaista asiakasta ja lisäksi kaksi virtuaalikoneella (VirtualBox 4.3.10) luotua Kali Linux-hyökkääjää. Hyökkääjää IP-osoitteella 192.168.1.182 kutsutaan jatkossa nimellä Hyökkääjä1 ja hyökkääjää IP-osoitteella 192.168.1.183 kutsutaan nimellä Hyökkääjä2.

Kuviossa 7 puolestaan näkyy komentokeskus (Command & Control-center), jonka kautta hallittiin kaikkia virtuaalikoneilla toimivia 18 asiakasbottia. Komentokeskuksessa näkyy botin tunnus, status, IP-osoite, käyttöjärjestelmä, luontipäivä, viimeisin päivitysaika, liikenteen tyyppi sekä istunnon pyyntöjen lähetystahti. Liikenteen tyyppiä vaihtamalla vaihtoehdosta ”normal” vaihtoehtoon ”slow” saatiin asiakasbotti luomaan normaalin liikenteen sijasta hidasta hyökkäysliikennettä (SlowLoris-hyökkäys).

Hyökkääjä2:lla käytettiin SlowHTTPTest-ohjelmaa (Versio 1.6), jolla pystyttiin tekemään SlowLoris-, SlowRead-, RUDY- ja RangeAttack-hyökkäykset. Nämä hyökkäykset on esitelty tarkemmin luvussa 3. Hyökkääjä1:llä puolestaan oli käytössä SSLsqueeze-ohjelma (versio 1.0), jolla pystytään kuormittamaan salattua yhteyttä ja luomaan laskennallinen SSL-

palvelunestohyökkäys. Hyökkäys kuormittaa paitsi palvelimen salauksenpurkua, joka tapahtuu SSL-yhteyden kättelyn aikana, niin myös SSL-protokollan uudelleenneuvottelua, joka antaa samalle TCP-yhteydelle mahdollisuuden lähettää satoja kättelyitä. Tästä seuraa se, että palvelimen laskentateho ei enää riitä salauksien purkamiseen. Näiden ohjelmien lisäksi kummallakin hyökkääjällä tehtiin porttiskannauksia Nmap-työkalulla. Porttiskannauksen tarkoituksena on skannata kaikki kohdekoneen portit, jolloin on mahdollista myös löytää koneen heikot kohdat hyökkäystä varten.



Kuvio 6. Käytetyn ympäristön rakenne ja koneilla toimineet ohjelmat.

Command & Control Center

Current list of bots

botID	Status	Hostname	OS information	hotIps	sourceIP	Created	LastUpdated	trafficType	sessionRate
82748927488b1588b9001e57b489fb5f	command	192.168.1.114	Linux 3.16.0-30-generic amd64	192.168.1.114|	192.168.1.114 proxy:n/a	2016-04-11 12:27:26	2016-04-11 13:57:39	normal	19
065050832cf0610f3f21b6c7b56935f4	command	192.168.1.199	Linux 3.16.0-30-generic amd64	192.168.1.199|	192.168.1.199 proxy:n/a	2016-04-11 12:27:26	2016-04-11 13:57:39	normal	22
68cbf04cb496736ccf7e052f0d34af26	command	192.168.1.113	Linux 3.16.0-30-generic amd64	192.168.1.113|	192.168.1.113 proxy:n/a	2016-04-11 12:27:27	2016-04-11 13:57:39	normal	17
4ac2cd21f3e9272ab21a5c1fd4053ed9	command	192.168.1.200	Linux 3.16.0-30-generic amd64	192.168.1.200|	192.168.1.200 proxy:n/a	2016-04-11 12:27:27	2016-04-11 13:57:39	normal	15
ebbd3778f83a6b3f0a31a82f00213749	command	192.168.1.193	Linux 3.16.0-30-generic amd64	192.168.1.193|	192.168.1.193 proxy:n/a	2016-04-11 12:27:27	2016-04-11 13:57:39	normal	13
d3de6722e034c4fa0879c62e656201a8	command	192.168.1.115	Linux 3.16.0-30-generic amd64	192.168.1.115|	192.168.1.115 proxy:n/a	2016-04-11 12:27:27	2016-04-11 13:57:39	normal	13
d2d99a4b3a6f7aa3fbdd067d6767c463	command	192.168.1.156	Linux 3.16.0-30-generic amd64	192.168.1.156|	192.168.1.156 proxy:n/a	2016-04-11 12:27:28	2016-04-11 13:57:39	normal	17
7f4059492761b86a112f8f5b5cdbd94c	command	192.168.1.194	Linux 3.16.0-30-generic amd64	192.168.1.194|	192.168.1.194 proxy:n/a	2016-04-11 12:27:28	2016-04-11 13:57:39	normal	21
520a87d91ba71f8dc9a905424b548a7d	command	192.168.1.111	Linux 3.16.0-30-generic amd64	192.168.1.111|	192.168.1.111 proxy:n/a	2016-04-11 12:27:29	2016-04-11 13:57:39	normal	18
85c32d79d6bd5deb628dc182a2377fa4	command	192.168.1.195	Linux 3.16.0-30-generic amd64	192.168.1.195|	192.168.1.195 proxy:n/a	2016-04-11 12:27:30	2016-04-11 13:57:39	normal	15
d29e68d7cce3f61def1ec1e691079826	command	192.168.1.137	Linux 3.16.0-30-generic amd64	192.168.1.137|	192.168.1.137 proxy:n/a	2016-04-11 12:37:56	2016-04-11 13:57:39	normal	83
4cff4b3272bc11d62f43c59ff0ca2711	command	192.168.1.140	Linux 3.16.0-30-generic amd64	192.168.1.140|	192.168.1.140 proxy:n/a	2016-04-11 12:38:05	2016-04-11 13:57:39	normal	83
87a0f703b82af06edd1edefe0fdcf36d	command	192.168.1.138	Linux 3.16.0-30-generic amd64	192.168.1.138|	192.168.1.138 proxy:n/a	2016-04-11 12:41:45	2016-04-11 13:57:40	normal	17
a1f87c6c38ca16bf05b3a9e006bd0c99	command	192.168.1.139	Linux 3.16.0-30-generic amd64	192.168.1.139|	192.168.1.139 proxy:n/a	2016-04-11 12:41:50	2016-04-11 13:57:39	normal	20
bf9153e05ed5f2346f3f6269fbe6b2fc	command	192.168.1.133	Linux 3.16.0-30-generic amd64	192.168.1.133|	192.168.1.133 proxy:n/a	2016-04-11 12:41:54	2016-04-11 13:57:39	normal	16
e54d9e03ac67b5a8f1aeb6f06ddb7bf8	command	192.168.1.135	Linux 3.16.0-30-generic amd64	192.168.1.135|	192.168.1.135 proxy:n/a	2016-04-11 12:41:57	2016-04-11 13:57:39	normal	19
dbf1ea6049583fa5f5c9db897422526c	command	192.168.1.134	Linux 3.16.0-30-generic amd64	192.168.1.134|	192.168.1.134 proxy:n/a	2016-04-11 12:42:00	2016-04-11 13:57:39	normal	16
92162e3dbe2a0fb9e1f6f12d93b1c7dc	command	192.168.1.141	Linux 3.16.0-30-generic amd64	192.168.1.141|	192.168.1.141 proxy:n/a	2016-04-11 12:42:03	2016-04-11 13:57:39	normal	17

Current C&C DB parameter values

General directive: sleep

Initial response: sleep

Kuvio 7. Komentokeskus, jossa näkyy kaikkien bottien tila sekä botteihin liittyvät tiedot.

6.2 Havaitsemiseen käytettävä menetelmä

Hajautettujen palvelunestomenetelmien havaitsemiseen SSL/TLS-salatussa yhteydessä käytetään tässä tutkielmassa menetelmää, joka on esitelty artikkelissa Zolotukhin ym. 2016. Kuten myös luvusta 4 voidaan huomata, monet HTTP-pohjaisten hyökkäysten nykyiset havaitsemismenetelmät perustuvat pakettien sisällön, kuten pyydetyn palvelun, HTTP-pyynnön menetelmän tai istunnon tunnisteen (ID), analysoimiseen. Koska tässä tapauksessa yhteys on salattu, niin pakettien sisältöön ei päästä käsiksi ilman salauksen purkamista, joka taas rikkoo tietosuojalakeja ja –säännöksiä. Näin ollen hyökkäyksen havaitsemisen täytyy perustua lähinnä pakettien otsakkeista saataviin tai niistä johdettuihin tietoihin. (Zolotukhin ym. 2016).

Piirteenirroitus ja standardointi tapahtuu tarkastelemalla sovelluskerroksen HTTP- ja HTTPS-protokollia tietyillä aikaväleillä. Palvelimen liikenne (sekä sisään- että ulospäin) kaapataan aikavälillä $[T_s, T_e]$, joka ei oletuksen mukaan sisällä yhtään hyökkäystä. Tästä kaapatusta liikenteestä muodostetaan normaalin käyttäjän käyttäytymismallit. Tämän jälkeen hyökkäys voidaan havaita reaaliajassa jakamalla aikaväli $[T_s, T_e]$ samanmittaisiksi ja ei-päällekkäin oleviksi intervalleiksi, joiden pituus on ΔT. Jotta havainnointi tapahtuu reaaliajassa, on aikavälin pituus oltava riittävän pieni. (Zolotukhin ym. 2016).

Hyökkäyksen havaitseminen perustuu verkon liikenteen vuon analysoimiseen. Vuo koostuu samoja ominaisuuksia (lähde- ja kohdekoneen IP-osoite sekä portti) sisältävien IP-pakettien ryhmästä, joka ohittaa valvontapisteen aikavälin $[T_s, T_e]$ aikana. Kun tarkastellaan liikenteen vuota aikavälillä $[T_s + i\Delta T, T_s + (i+1)\Delta T]$, otetaan näin ollen huomioon kaikki tämän vuon paketit, jotka on lähetetty aiempien aikavälien $[T_s + (i-1)\Delta T, T_s + i\Delta T]$ ja $[T_s + (i-2)\Delta T, T_s + (i-1)\Delta T]$ aikana. Sellaiset vuot, joiden lähdesoketti (*source socket*) ja kohdesoketti (*destination socket*) vastaavat toisiaan yhdistetään ja tätä sanotaan yhdeksi keskusteluksi asiakkaan ja palvelimen välillä. Keskustelua voidaan kuvata neljällä piirteellä, jotka ovat lähteen IP-osoite, lähdeportti, kohteen IP osoite ja kohdeportti. Jokaiselle keskustelulle jokaisella aikavälillä voidaan irroittaa seuraavat piirteet:

1) keskustelun kesto
2) lähetettyjen pakettien määrä sekunnissa

3) lähetettyjen tavujen määrä sekunnissa
4) minimi- ja maksimipakettikoko sekä keskimääräinen pakettikoko
5) TCP-ikkunan minimi- ja maksimikoko sekä keskimääräinen koko
6) minimi- ja maksimielinaika (*TTL, time to live*) sekä keskimääräinen elinaika
7) eri TCP-kuittauksen (*TCP flag*) sisältävien pakettien prosenttiosuudet (FIN, SYN, RST, PSH, ACK ja URG)
8) erilaiset ominaisuudet (kättely, hälytys jne.) sisältävien salattujen pakettien prosenttiosuudet.

Piirteet 2-8 erotetaan erikseen asiakkaalta palvelimelle ja palvelimelta asiakkaalle lähetetyille paketeille. Koska piirteiden arvojen mitta-asteikko saattaa vaihdella, piirrevektorit standardoidaan max-min-normalisoinnilla (*max-min normalization*). (Zolotukhin ym. 2016).

Itse havaitsemisalgoritmi koostuu kolmesta vaiheesta, jotka ovat keskustelujen klusterointi, triviaalien DoS-hyökkäysten havaitseminen ja keskitason DoS-hyökkäysten havaitseminen. Klusterointivaiheessa mallinnetaan normaalin käyttäjän käyttäytymistä muodostamalla piirteiltään samanlaisista keskusteluista ryhmiä. Tätä ryhmiin jakamista kutsutaan klusteroinniksi ja se on mahdollista tehdä eri klusterointialgoritmeilla. Käytetyimpiä algoritmeja ovat hierarkkiset algoritmit (*hierarchical clustering*), keskipisteeseen perustuvat algoritmit (*centroid-based clustering*) sekä tiheyteen perustuvat algoritmit (*density-based clustering*). Jokainen laskettu klusteri sisältää tietyn osan verkon liikenteestä, esimerkiksi kaikki keskustelut palvelimen ja asiakkaan välillä, joissa pyydetään samaa palvelua. (Zolotukhin ym. 2016).

Triviaalien hyökkäysten havaitsemisvaiheessa pyritään erottamaan sellaiset hyökkäykset, joissa hyökkäykseen osallistuvat botit tekevät yhden tai rajoitetun määrän toisistaan riippumattomia HTTP-hyökkäyksiä kohdesivustolle (esimerkiksi SlowRead- ja SlowPost-hyökkäykset). Nämä hyökkäykset huomataan vertaamalla uuden keskustelun piirteitä normaaleista käyttäjistä muodostettuihin klustereihin. Mikäli keskustelu ei kuulu mihinkään klusteriin, merkitään se haitalliseksi. Hierarkkisessa klusteroinnissa klusteriin kuuluminen määräytyy määrittämällä uuden vektorin pienin etäisyys klusterissa oleviin vektoreihin. Mikäli etäisyys ylittää klusterin kynnysarvon T, kyseinen vektori ei kuulu klusteriin. Kynnysarvo

lasketaan kaavalla $T = \mu^n + \gamma\sigma^n$, missä μ^n on kahden vierekkäisen vektorin keskimääräinen etäisyys klusterissa ja σ^n on näiden etäisyyksien keskihajonta. Keskipisteeseen perustuvassa klusteroinnissa puolestaan tarkastellaan vektorin etäisyyttä klusterin keskipisteeseen. Periaate on sama kuin hierarkkisessa klusteroinnissa, mutta kynnysarvo T lasketaan kaavalla $T = \mu^c + \alpha\sigma^c$, missä μ^c on keskustan ja klusterissa olevien vektoreiden keskimääräinen etäisyys ja σ^c on etäisyyksien keskihajonta. (Zolotukhin ym. 2016).

Tiheyteen perustuvassa klusteroinnissa vektorin kuuluminen klusteriin määritetään saavutettavuuden (*density-reachability*) kautta. Vektori z on saavutettavissa pisteestä y suhteessa parametreihin ε ja N_{min}, jos on olemassa pisteiden ketju $y_1, y_2, \dots, y_m$, missä $y_1 = y$ ja $y_m = z$ s.e. $\forall i \in \{1,2, \dots, m-1\}$, pisteiden y_i ja y_{i+1} välinen etäisyys $d^E(y_i, y_{i+1}) \leq \varepsilon$ ja $N_\varepsilon(y_i) \geq N_{min}$. Parametrit ε ja N_{min} voidaan valita harjoitusdataan perustuen s.e. ε on keskimääräinen etäisyys klusterin vektoreiden välillä ja N_{min} on pienin määrä pisteitä, joista klusterin jokainen piste on saavutettavissa toisistaan. (Zolotukhin ym. 2016).

Keskitason DoS-hyökkäysten havaitsemisvaiheessa puolestaan pyritään erottamaan sellaiset hyökkäykset, joissa botit generoivat satunnaisia pyyntösekvenssejä, jotka matkivat normaalien käyttäjien tuottamaa nettiliikennettä. Tätä varten kaikki saman lähteen IP-osoitteen, kohteen IP-osoitteen ja kohteen portin sisältävät keskustelut ryhmitellään yhteen ja jokaista ryhmää tarkastellaan erikseen lyhyillä aikaväleillä. Jokaiselle ryhmälle lasketaan eri klustereissa olevien piirrevektorien prosenttiosuudet, jotka säilötään histogrammivektoriin $h^{it} = (h_1^{it}, h_2^{it}, \dots, h_{n_C}^{it})$. Tässä n_C on paljastuneiden klusterien määrä ja h_j^{it} on klusteriin j kuuluvat i:nnen ryhmän piirrevektorit t:nnellä aikavälillä jaettuna i.nnen ryhmän vektorien kokonaismäärällä kyseisellä aikavälillä. Kun histogrammivektori on laskettu kaikille keskusteluryhmille kaikilla aikaväleillä, muodostetaan näistä vektoreista matriisi H^i, jossa t:nnes rivi $H^{it} = (H_1^{it}, H_2^{it}, \dots, H_{n_C}^{it})$ vastaa histogrammivektoria h^{it}. (Zolotukhin ym. 2016).

Saadut histogrammit analysoidaan pinotulla AE:lla, jolloin korkean tason piirteet voidaan oppia alemman tason piirteistä ja muodostaa kunnolliset piirteet mallien muodostamista varten. Tavallisessa AE:ssa on kolme kerrosta, jotka ovat näkyvä syötekerros (*visible input layer*), piilotettu kerros (*hidden layer*) ja rekonstruktio-/uudelleenrakennuskerros (*reconst-*

ruction layer). AE:n päämääränä on oppia, miten syötevektorit (*input vectors*) saadaan koodaamisen (*encoding*) ja koodinpurkamisen (*decoding*) kautta tuotettua uudelleen ulostulovektoreiksi (*output vectors*). Vektorista h koodatun vektorin h^e i:nnes arvo saadaan kaavalla $h_i^e = f(\sum_{j=1}^{n^{\text{input}}} w_{ij}^e h_j + b_i^e)$, missä f on aktivointifunktio, n^{input} on syötekerroksen neuronien määrä, w_{ij}^e on piilotetun kerroksen i:nnen neuronin ja syötekerroksen j:nnen neuronin välisen yhteyden paino ja b_i^e on syötekerroksen harhavektorin i:nnes arvo. Vastaavasti uudelleenrakennetun vektorin i:nnes arvo saadaan kaavasta $h_i^d = f(\sum_{j=1}^{n^{\text{hidden}}} w_{ij}^d h_j + b_i^d)$, missä n^{hidden} on piilotetun kerroksen neuronien määrä, w_{ij}^d on tuloskerroksen i:nnen neuronin ja piilotetun kerroksen j:nnen neuronin välisen yhteyden paino ja b_i^d on piilotetun kerroksen harhavektorin i:nnes arvo. (Zolotukhin ym. 2016).

Tarkoituksena on minimoida syötevektorin ja uudelleenrakennetun vektorin välinen keskineliövirhe $E_h = \frac{1}{2}\sum_{i=1}^{n}(h_i - h_i^d)^2$. AE käyttää back-propagation-algoritmia, jolloin yhteyden paino muuttuu jokaisen piirrevektorin prosessoinnin jälkeen perustuen tuloksen keskineliövirheeseen. Muutokset painoissa Δw_{ij}^l ja harha Δb_i^l ($l \in \{e, d\}$) saadaan kaavoista $\Delta w_{ij}^l = -\eta h_j^e \delta_i^l$ ja $\Delta b_i^l = -\eta \delta_i^l$. Parametrit $\delta_i^d = (h_i^d - h_i) f'$ ja $\delta_i^d = \left(\sum_{j=1}^{n^{\text{hidden}}} w_{ij}^d \delta_i^d\right) f'$ mittaavat, kuinka suurta virhettä i:nnes neuroni aiheuttaa tulokseen. Jos AE rakentaa uudelleen alkuperäisen syötteen täydellisesti, piilotettu kerros sisältää riittävästi informaatiota syötteestä. (Zolotukhin ym. 2016).

Pinotun AE:n tapauksessa harjoittelu aloitetaan syötevektoreilla, jonka jälkeen seuraava kerros koulutetaan niillä vektoreilla, jotka aktivoivat edellisen kerroksen piilotetut neuronit. Näin toimitaan, kunnes kaikki kerrokset on käyty läpi. Tämän jälkeen voidaan back-propagation-algoritmilla parantaa kaikkien kerrosten tuloksia samanaikaisesti. Pinottu AE käyttää histogrammimatriisin H^{it} vektoreita kouluttamiseen, jonka jälkeen lasketaan uudelleenrakennusvirhe E jokaiselle histogrammimatriisin rivivektorille. Virhe saadaan laskettua t:nnelle riville kaavalla $E^{it} = \sqrt{\sum_{j=1}^{n_C}(H_j^{it} - \widehat{H}_j^{it})^2}$, missä $\widehat{H}^{it}$ on rivin H^{it} pinotulla AE:lla uudelleenrakennettu vastine. Näitä uudelleenrakennusvirheitä käytetään laskettaessa kyn-

nysarvo $T^E = \mu_i^E + \omega\sigma_i^E$, missä μ_i^E on i:nnen piirrevektorien ryhmän uudelleenrakennusvirheen keskiarvo ja σ_i^E on virheiden keskihajonta. Jos uudelleenrakennusvirhe on suurempi kuin vastaavalle ryhmälle harjoitusvaiheessa laskettu kynnysarvo, merkitään vektori h poikkeavaksi ja kyseisen asiakkaan yhteydet tulkitaan hyökkäykseksi. (Zolotukhin ym. 2016).

6.3 Testauksen kulku ja tulokset

Havainnointialgoritmi oli koulutettu valmiiksi, joten erillistä harjoitteludataa ei tarvinnut generoida, vaan testaus aloitettiin suoraan. Testaus alkoi 11.04.2016 kello 13:09 ja päättyi kello 13:41, jolloin kokonaiskesto oli 32 minuuttia. Taulukossa 1 näkyy aikajana testauksen aikana tapahtuneista hyökkäyksistä. Taulukkoon on listattu aloitus- ja lopetusaika, kesto, hyökkäyksen tehnyt kone sekä hyökkäyksen tyyppi. SlowLoris-hyökkäyksiä oli yhteensä 4, joista jokainen tapahtui eri koneella. Asiakasbotit 192.168.1.137 ja 192.168.1.140 generoivat ensin hidasta liikettä 24 minuuttia, jonka jälkeen niiden liikenne vaihdettiin normaaliksi. Tämä vastaa SlowLoris-hyökkäystä. Hyökkääjä2 teki kolmen minuutin SlowLoris-hyökkäyksen alkaen kello 13:11 ja asiakasbotti 192.168.1.141 laitettiin kolmeksi minuutiksi generoimaan hidasta liikennettä kello 13:16 alkaen.

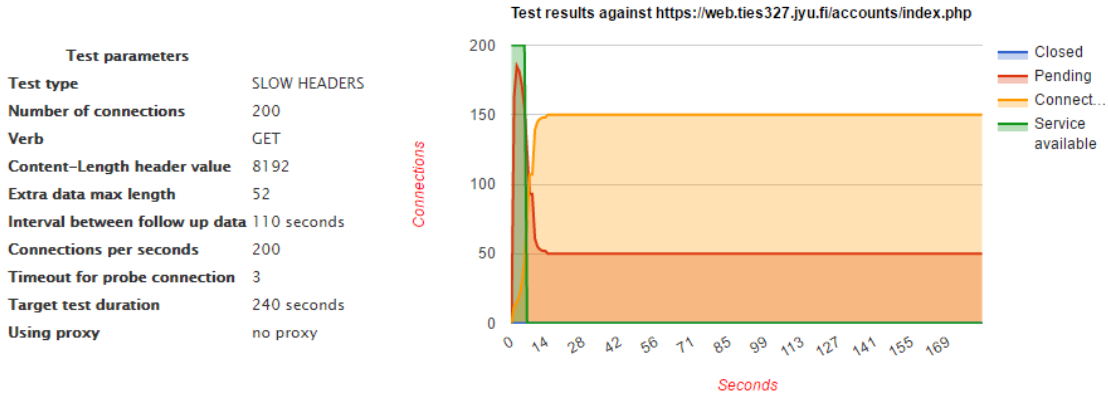
SSLsqueeze-hyökkäyksiä tehtiin 2, joista molemmat tapahtuivat samalla koneella. Hyökkääjä1 teki kello 13:12 alkaen kaksi minuuttia kestäneen SSLsqueeze-hyökkäyksen ja toisen kaksi minuuttia kestäneen SSLsqueeze- hyökkäyksen kello 13:34 alkaen. Porttiskannauksia tehtiin puolestaan 3 kappaletta kahdella eri koneella. Ensimmäisen porttiskannauksen teki hyökkääjä1 kello 13:19 ja kaksi seuraavaa tapahtuivat kello 13:27 ja kello 13:39 hyökkääjä2:n toimesta.

RUDY-, SlowRead- ja RangeAttack-hyökkäyksiä tapahtui kaikkia 1 ja ne suoritti hyökkääjä2. RUDY-hyökkäys alkoi kello 13:23 ja sen kesto oli 4 minuuttia. SlowRead-hyökkäys puolestaan alkoi kello 13:31 ja kestona oli 5 minuuttia. RangeAttack-hyökkäys aloitettiin kello 13:39 ja se kesti vain 9 sekuntia.

Taulukko 1. Testauksen kulun aikajana, joka sisältää testauksen aikana tehdyt hyökkäykset ja hyökkäykset suorittaneet koneet.

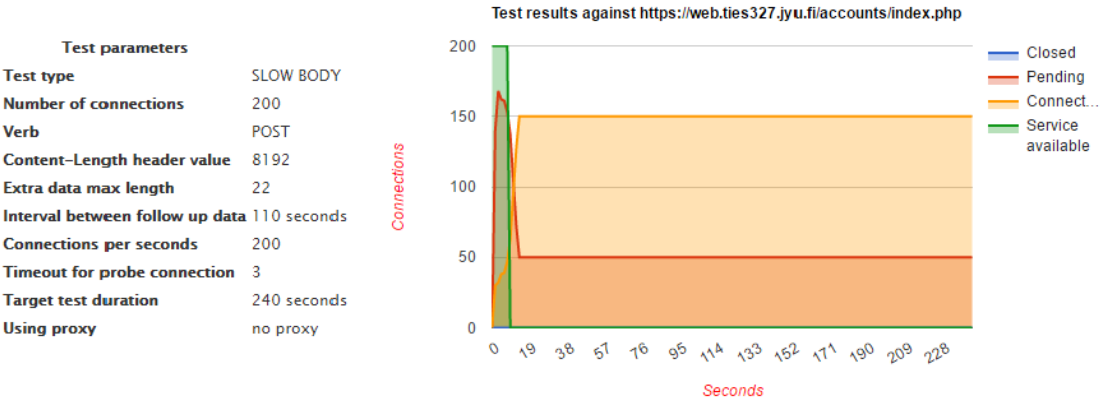
Aloitusaika	**Lopetusaika**	**Kesto**	**Kone**	**Hyökkäys**
13:09	13:33	24 min	Asiakasbotti 192.168.1.137	SlowLoris
13:09	13:33	24 min	Asiakasbotti 192.168.1.140	SlowLoris
13:11	13:14	3 min	Hyökkääjä2	SlowLoris
13:12	13:14	2 min	Hyökkääjä1	SSLsqueeze
13:16	13:19	3 min	Asiakasbotti 192.168.1.141	SlowLoris
13:19	13:19	3 s	Hyökkääjä1	Porttiskannaus
13:23	13:27	4 min	Hyökkääjä2	RUDY
13:27	13:27	3 s	Hyökkääjä2	Porttiskannaus
13:31	13:16	5 min	Hyökkääjä2	SlowRead
13:34	13:36	2 min	Hyökkääjä1	SSLsqueeze
13:38	13:38	9 s	Hyökkääjä2	RangeAttack
13:39	13:39	3 s	Hyökkääjä2	Porttiskannaus

Kuvioista 8-11 löytyvät SlowHTTPTest-ohjelman tulosteet, jotka sisältävät hyökkäyksessä käytetyt parametrit sekä kuvaajan palvelun saavutettavuudesta. Kuviossa 8 on SlowLoris-hyökkäyksen tiedot. Siitä näkyy, että palvelu on ollut saavutettavissa noin ensimmäiset 10 sekuntia, jonka jälkeen palvelu on ollut saavuttamattomissa. Hyökkäys siis onnistuneesti esti palvelun käytön.



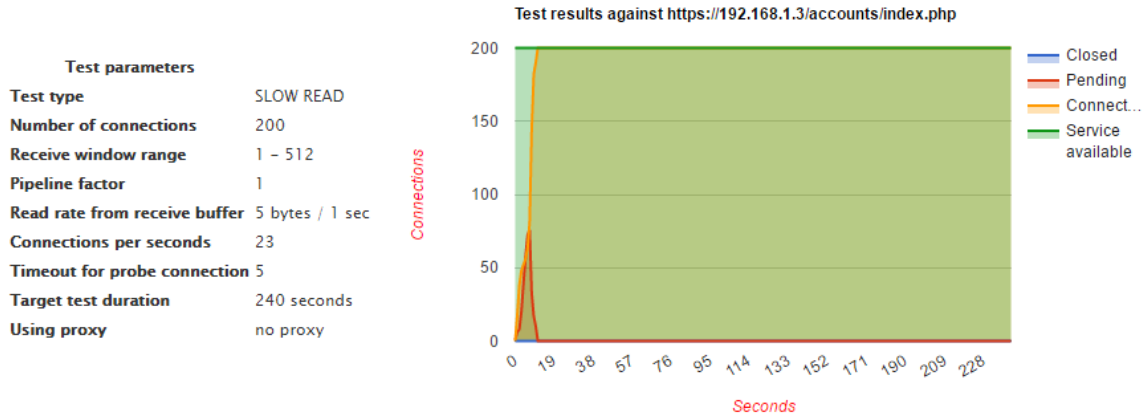
Kuvio 8.SlowLoris-hyökkäyksen parametrit ja palvelun saavutettavuus hyökkäyksen aikana.

Kuviossa 9 näkyy puolestaan RUDY-hyökkäyksen tiedot. Myös RUDY-hyökkäys pystyi estämään palvelun käytön ja kuviosta 9 näkyy, että palvelu on ollut käytettävissä vain ensimmäiset 15 sekuntia hyökkäyksen alkamisesta.



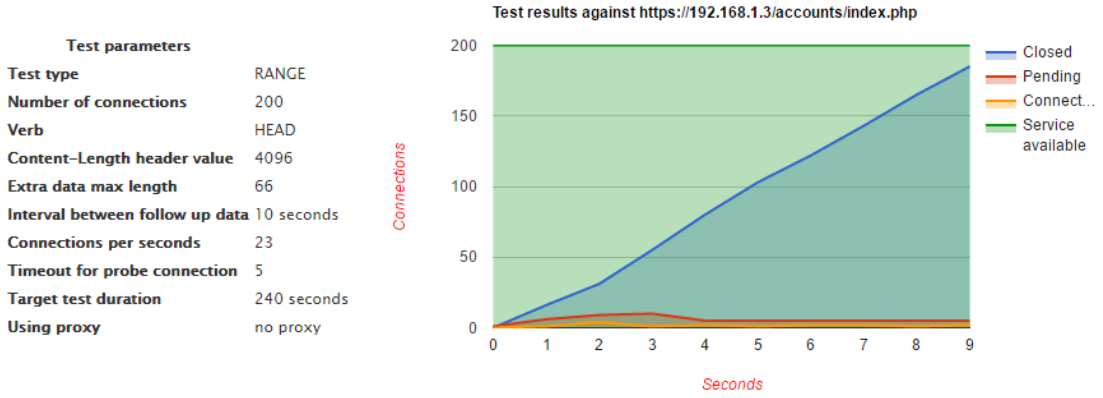
Kuvio 9. RUDY-hyökkäyksen parametrit ja palvelun saavutettavuus hyökkäyksen aikana.

Kuviossa 10 näkyy SlowRead-hyökkäyksen tiedot. SlowRead-hyökkäys ei pystynyt tässä tapauksessa estämään palvelun käyttöä vaan palvelu oli saavutettavissa koko hyökkäyksen ajan.



Kuvio 10. SlowRead-hyökkäyksen parametrit ja palvelun saavutettavuus hyökkäyksen aikana.

Kuviossa 11 näkyy RangeAttack-hyökkäyksen tiedot. Hyökkäys kesti vain vähän aikaa, noin 9 sekuntia, joten se ei ehtinyt vaikuttamaan palvelun käyttöön ollenkaan.



Kuvio 11. RangeAttack-hyökkäyksen parametrit ja palvelun saavutettavuus hyökkäyksen aikana.

Havaitsemisjärjestelmän antamasta tulosteesta voidaan taulukosta 1 löytyvän aikajanan avulla tarkastaa, huomattiinko kaikki tehdyt hyökkäykset vai ei. Koko tuloste sisälsi 401 800 hyökkäykseksi luokiteltua keskustelua. Näistä 509 oli niin sanottuja vääriä positiivisia eli normaaleja keskusteluja, jotka tulkittiin hyökkäyksiksi. Tällöin vääriä positiivisia oli 0,13 % ja aitoja positiivisia 99,87 % kaikista hyökkäyksiksi luokitelluista keskusteluista. Taulukosta 2 löytyy hyökkäyksen havaitsemisjärjestelmän antamasta tulosteesta osa. Tulostetta on karsittu suuren kokonsa vuoksi siten, että jokaisesta havaitusta hyökkäyksestä on jätetty korkeintaan kolme erillistä keskustelua ja myös väliin jääneitä virheellisiä havaintoja on korkeintaan kolme kappaletta. Taulukkoon on kirjattu aika, lähtöosoite, kohdeosoite sekä hyökkäyksen tyyppi. Hyökkäyksen tyyppi lukee aina vain ensimmäisen keskustelun kohdalla. Taulukon 1 ja 2 värit vastaavat toisiaan eli tietty hyökkäys näkyy samanvärisellä pohjalla kummassakin taulukossa. Taulukossa 2 valkoisella pohjalla olevat keskustelut ovat virheellisesti hyökkäyksiksi tulkittuja keskusteluja.

Taulukko 2. Hyökkäyksen havaitsemisjärjestelmän antama tuloste, johon on poimittu havaitut hyökkäykset ja osa vääristä positiivisista.

Aika	**Lähtöosoite**	**Kohdeosoite**	**Hyökkäys**
13:09:53.213522	192.168.1.140:x	192.168.1.3:443	SlowLoris
13:09:53.213522	192.168.1.140:34675	192.168.1.3:443	
13:09:53.213522	192.168.1.140:34675	192.168.1.3:443	
13:10:09.128381	192.168.1.137:x	192.168.1.3:443	SlowLoris
13:10:09.128381	192.168.1.137:42385	192.168.1.3:443	
13:10:09.155900	192.168.1.137:42386	192.168.1.3:443	
13:10:21.034569	192.168.1.115:x	192.168.1.3:443	
13:10:34.885142	192.168.1.193:x	192.168.1.3:443	
13:10:42.634192	192.168.1.115:x	192.168.1.3:443	
13:11:21.017700	192.168.1.183:50705	192.168.1.3:443	SlowLoris
13:11:21.029619	192.168.1.183:50706	192.168.1.3:443	
13:11:21.029619	192.168.1.183:x	192.168.1.3:443	

(jatkuu)

Taulukko 2. (jatkuu).

Aika	Lähtöosoite	Kohdeosoite	Hyökkäys
13:11:22.962040	192.168.1.138:x	192.168.1.3:443	
13:13:14.772535	192.168.1.182:38823	192.168.1.3:443	SSLsqueeze
13:13:14.772584	192.168.1.182:38824	192.168.1.3:443	
13:13:14.812841	192.168.1.182:38825	192.168.1.3:443	
13:13:26.781164	192.168.1.193:x	192.168.1.3:443	
13:16:28.829282	192.168.1.115:x	192.168.1.3:443	
13:16:52.792039	192.168.1.141:58206	192.168.1.3:443	SlowLoris
13:16:52.792039	192.168.1.141:58206	192.168.1.3:443	
13:16:52.792039	192.168.1.141:x	192.168.1.3:443	
13:19:30.828104	192.168.1.138:x	192.168.1.3:443	
13:19:45.825171	192.168.1.194:x	192.168.1.3:443	
13:20:00.373716	192.168.1.182:38081	192.168.1.3:443	Porttiskannaus
13:20:00.373716	192.168.1.182:x	192.168.1.3:443	
13:20:05.647819	192.168.1.193:x	192.168.1.3:443	
13:24:01.741108	192.168.1.113:x	192.168.1.3:443	
13:24:12.380271	192.168.1.111:x	192.168.1.3:443	
13:24:13.553216	192.168.1.183:50956	192.168.1.3:443	RUDY
13:24:13.562026	192.168.1.183:50957	192.168.1.3:443	
13:24:13.562026	192.168.1.183:50957	192.168.1.3:443	
13:24:16.978585	192.168.1.194:51690	192.168.1.3:443	
13:27:41.988267	192.168.1.156:x	192.168.1.3:443	
13:27:52.970653	192.168.1.182:33804	192.168.1.3:443	Porttiskannaus
13:27:52.970653	192.168.1.182:x	192.168.1.3:443	
13:27:55.948072	192.168.1.193:x	192.168.1.3:443	
13:31:25.022380	192.168.1.156:x	192.168.1.3:443	
13:31:28.885295	192.168.1.133:x	192.168.1.3:443	

(jatkuu)

Taulukko 2. (jatkuu).

13:31:37.757305	192.168.1.183:51205	192.168.1.3:443	SlowRead
13:31:37.757305	192.168.1.183:x	192.168.1.3:443	
13:31:37.757555	192.168.1.183:51206	192.168.1.3:443	
13:31:39.029370	192.168.1.113:x	192.168.1.3:443	
13:34:37.014670	192.168.1.199:x	192.168.1.3:443	
13:34:45.853859	192.168.1.182:51596	192.168.1.3:443	SSLsqueeze
13:34:45.854119	192.168.1.182:51597	192.168.1.3:443	
13:34:45.854134	192.168.1.182:51598	192.168.1.3:443	
13:38:35.675394	192.168.1.140:52159	192.168.1.3:443	
13:38:40.353320	192.168.1.183:51454	192.168.1.3:443	RangeAttack
13:38:40.353320	192.168.1.183:x	192.168.1.3:443	
13:38:40.353914	192.168.1.183:51455	192.168.1.3:443	
13:38:41.014149	192.168.1.199:x	192.168.1.3:443	
13:39:33.953082	192.168.1.156:x	192.168.1.3:443	

Taulukosta 2 nähdään, että kaikki muut hyökkäykset, paitsi viimeinen porttiskannaus, havaittiin järjestelmän toimesta. Koska aiemmatkin porttiskannaukset havaittiin oikein, epäilisin viimeisen puuttumisen johtuvan siitä, että järjestelmä ei ehtinyt tulostaa sitä tulosteeseen ennen järjestelmän sammuttamista. SlowHTTPTest-ohjelman antamien tulosteiden (kuviot 8-11) valossa havaitsemismenetelmän toimivuuden kannalta ei ollut merkitystä, oliko hyökkäys onnistunut vai ei. Menetelmä pystyi havaitsemaan myös epäonnistuneet hyökkäykset, eli RangeAttack- ja SlowRead-hyökkäykset, vaikka ne eivät itse palvelun käyttöä missään vaiheessa estäneet. Näin ollen menetelmällä pystyy havaitsemaan hyökkäykset ennen kuin ne ehtivät aiheuttaa vahinkoa uhrille.

7 Yhteenveto

Tässä tutkielmassa on tarkasteltu OSI-viitemallin seitsemännen kerroksen (tai vastaavasti TCP/IP-mallin neljännen kerroksen) eli sovelluskerroksen palvelunestohyökkäyksiä. Tutkielman voidaan ajatella jakautuvan kahteen osaan, joista ensimmäinen käsittelee mahdollisimman kattavasti sovelluskerroksella tapahtuvia palvelunestohyökkäyksiä sekä sovelluskerroksen taustalla olevia protokollia. Tutkielman toinen osio puolestaan sisältää käytännöntyön salatuissa yhteyksissä tapahtuvista palvelunestohyökkäyksistä ja niiden havaitsemisesta.

Tutkielman alkuosan tarkoitus on ollut tarjota sekä kattava katsaus sovelluskerroksen palvelunestohyökkäyksiin että riittävät taustatiedot internetin rakenteesta ja relevanteista protokollista. Internetin rakenteen kuvaus on sisältänyt OSI-viitemallin sekä TCP/IP-mallin esittelyt. Lisäksi on esitelty tarkemmin sovelluskerroksen protokollia ja niiden tietoturvaheikkouksia. Palvelunestohyökkäyksistä on käyty läpi mitä ne ovat, miten niitä tehdään, miksi niitä tehdään, miten ne voidaan havaita sekä miten niiltä pystyy suojautumaan. Tutkielman alkuosan lukemalla pitäisi saada hyvä käsitys sovelluskerroksen hajautetuista palvelunestohyökkäyksistä ilman, että lukijan juurikaan tarvitsee turvautua ulkopuolisiin lähteisiin. Palvelunestohyökkäysten havaitsemisesta on esitelty tämän hetken uusimmat menetelmät, joten lukijalle pitäisi jäädä hyvä käsitys siitä, minkälaiset menetelmät ovat tällä hetkellä tutkimuksen kohteena. Lisäksi on tarjottu yleisiä ohjeita, menetelmiä sekä joitakin palveluja palvelunestohyökkäyksiltä suojautumiseen.

Tutkielman toisen osan tarkoituksena on ollut perehtyä käytännön kautta tarkemmin salatuissa yhteyksissä tapahtuviin palvelunestohyökkäyksiin eli SSL DDoS-hyökkäyksiin. Tämä on valikoitunut kiinnostuksen aiheeksi, koska tällä hetkellä ei juurikaan ole saatavissa tutkimustietoa SSL DDoS-hyökkäysten havaitsemisesta. Kuitenkin internetin palvelut ovat menossa enemmän siihen suuntaan, että kommunikaatio tapahtuu salatuissa yhteyksissä. Näin ollen SSL DDoS-hyökkäysten voisi olettaa lisääntyvän tulevaisuudessa.

Käytännönosio on suoritettu laboratorioympäristössä simuloimalla erilaisia hitaita hyökkäyksiä, RangeAttack-hyökkäys, porttiskannauksia sekä laskennallinen SSL-hyökkäys. Käytetty havaitsemismenetelmä on perustunut pelkkien otsakkeista saatavien tietojen avulla

tapahtuvaan luokitteluun sekä neuroverkkoihin kuuluvien AE:iden käyttöön. Kyseisellä menetelmällä on tutkielmassa pystytty havaitsemaan paitsi useita erilaisia hitaita hyökkäyksiä salatussa yhteydessä niin myös RangeAttack-hyökkäys, porttiskannaus ja laskennallinen hyökkäys. Myöskään hyökkäyksen onnistumisella ei ole ollut vaikutusta havaitsemiseen, vaan menetelmällä on pystytty havaitsemaan hyökkäykset myös silloin, kun ne eivät ole haitanneet palvelun käyttöä. Nämä asiat indikoivat sitä, että mallia kehittämällä olisi mahdollista laajentaa hyökkäysten havaitseminen niin sanottuihin edistyneisiin hyökkäyksiin eli ihmismäistä selauskäytöstä imitoiviin hyökkäyksiin. Mahdollisia kehityssuuntia olisivat esimerkiksi useiden erilaisten havaitsemismenetelmien yhdisteleminen moniulotteiseksi järjestelmäksi, jolloin saataisiin paremmin havaittua erityyppisiä palvelunestohyökkäyksiä salatuissa verkoissa. Väärien positiivisten vähentäminen voisi tapahtua analysoimalla dataa eri vaiheissa siten, että väärin seulotut käyttäjät pystytään luokittelemaan myöhemmin uudelleen oikeaan ryhmään. Lisäksi epävarmoihin tapauksiin voisi soveltaa esimerkiksi CAPTCHA-testin käyttöä.

Tutkimuksen kehityssuunta on selvästi se, että pääpaino alkaa suuntautua edistyneiden hyökkäysten tunnistamiseen, mikä onkin hyvä, sillä tällaisten hyökkäysten määrä on ollut viime vuosina kasvussa. Lisäksi pilvipalveluissa tapahtuvat hyökkäykset ovat olleet tutkimuksissa kiinnostuksen kohteena, varmasti johtuen pilvipalveluiden viime vuosien suosiosta. Mielenkiintoinen aspekti on myös se, kuinka ruuhkapiikit ja palvelunestohyökkäykset pystytään erottamaan toisistaan, vaikka ilmiöt ovat huomattavan samanlaisia. Näitä aiheita käsitteleviä tutkimuksia tullaankin varmasti julkaisemaan lisää. Kuitenkin tutkimusten joukosta puuttuu itseäni kiinnostava tutkimussuunta eli tekoälyn yhdistäminen hyökkäysten havaitsemiseen sekä itsenäisesti oppivat havaitsemis- ja puolustusjärjestelmät. Jääkin nähtäväksi, tulevatko tekoälysovellukset jossain vaiheessa laajenemaan myös tietoturvapuolelle.

Lähteet

[ISO7498-1]. ISO/IEC IS 7498-1:1994(E): Information technology – Open Systems Interconnetion – Basic Reference Model: The Basic Model. International Organization for Standardization, Geneve, Switzerland. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=20269

[RFC-791]. Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981. URL: https://tools.ietf.org/html/rfc791

[RFC-1883]. Deering, S. & Hinden, R. "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, DOI 10.17487/RFC1883, December 1995. URL: https://tools.ietf.org/html/rfc1883

[RFC-2818]. Rescorla, E. "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000. URL: https://tools.ietf.org/html/rfc2818

[RFC-5246]. Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, URL: http://www.rfc-editor.org/info/rfc5246

[RFC-7530]. Haynes, T., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, URL: http://www.rfc-editor.org/info/rfc7530

Aiello, M. & Mongelli, M. & Papaleo, G. (2013). Basic Classifiers for DNS Tunneling Detection. *2013 IEEE Symposium on Computers and Communications (ISCC)*, s.880-885. DOI: 10.1109/ISCC.2013.6755060

Alder, R., Burke, J., Keefer, C., Orebaugh, A., Pesce, L & Seagren E.S. (2007). *How to cheat at configuring Open Source Security tools.* Syngress Publishing, Inc. Massachussetts.

Arbor Networks (2016). Worldwide Infrastructure Security Report, Vol. XI. URL: https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf (Viitattu 15.5.2016)

Balani, N. & Hathi, R. (2009). *Apache CXF Web Service Development: Develop and Deploy SOAP and RESTful Web Services.* Packt Publishing, Birmingham.

Beitollahi, H. & Deconinck, G. (2012a). Analyzing well-known countermeasures against distributed denial of service attacks. *Computer Communications,* vol.35 (2012), s.1312-1332. DOI: 10.1016/j.comcom.2012.04.008

Beitollahi, H. & Deconinck, G. (2012b). Tackling Applications-layer DDoS Attacks. *Procedia Computer Science,* vol.10 (2012), s.432-441. DOI: 10.1016/j.procs.2012.06.056

Bencsáth, B. & Rónai, M.A. (2007). Empirical analysis of Denial of Service attack against SMTP servers. *International Symposium on Collaborative Technologies and Systems (CTS 2007),* s.72-79. DOI: 10.1109/CTS.2007.4621740

Blank, A. (2002). *TCP/IP JumpStart: Internet Protocol Basics.* Sybex, California.

Chen, J.-H., Zhong, M., Chen, F.-J. & Zhang, A.-D. (2012). DDoS Defense System with Turing test and Neural Network. *IEEE International Conference on Granular Computing (GrC),* p.38-43. DOI: 10.1109/GrC.2012.6468680

Choi, Y.S., Oh, J.T., Jang, J.S. & Kim, I.K. (2011). Timeslot Monitoring Model for Application Layer DDoS Attack Detection. *6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT),* s.677-679.

Chuan, X., Guofeng, Z., Gaogang, X. & Shui, Y. (2014). Detection on Application Layer DDoS using Random Walk Model. *International Conference on Communications (ICC),* s.707-712. DOI: 10.1109/ICC.2014.6883402

Chwalinski P. & Belavkin, R. & Cheng, X. (2013). Detection od Application Layer DDoS Attack with Clustering and Likelihood Analysis. *IEEE Globecom Workshops, 1st International workshop on Security and Privacy and Big Data,* s.217-222. DOI: 10.1109/GLOCOMW.2013.6824989

Dantas, Y.G., Nigam, V. & Fonseca, I.E. (2014). A Selective Defense for Application Layer DDoS Attacks. *2014 IEEE Joint Intelligence and Security Informatics Conference (JISIC),* s.75-82. DOI: 10.1109/JISIC.2014.21

Devi, S. R. & Yogesh, P. (2012). An effective approach to counter application layer DDoS attacks. *3rd International Conference on Computing Communication & Networking Technologies (ICCNT)*, s.1-4. DOI: 10.1109/ICCCNT.2012.6395941

Dostálek, L. & Kabelová, A. (2006). *Understanding TCP/IP: A clear and comprehensive guide to TCP/IP protocols.* Packt Publishing, Birmingham.

Douligeris, C. & Aikaterini, M. (2003). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, vol.44 (2004), s.643-666. DOI: 10.1016/j.comnet.2003.10.003

Durcekova, V. & Schwartz, L. & Shahmehri, N. (2012). Sophisticated Denial of Service Attacks Aimed at Application layer. *ELEKTRO, 2012*, s.55-60. DOI: 10.1109/ELEKTRO.2012.6225571

Edwards, J. & Bramante, R. (2009). *Networking self-teaching guide: OSI, TCP/IP, LANs, MANs, WANs, Implementation, Management, and Maintenance.* Wiley, Indiana.

Gallagher, S. (2016). Majos DNS provider hit by mysterious, focused DDoS attack. *Ars Technica.* URL: http://arstechnica.com/information-technology/2016/05/major-dns-provider-hit-by-mysterious-focused-ddos-attack/ (Viitattu 27.5.2016)

Giralte, L.C., Conde C., de Diego, I.M. & Cabello, E. (2013). Detecting denial of service by modelling web-server behaviour. *Computers & Electrical Engineering*, vol.39, no.7, s. 2252-2262. DOI: 10.1016/j.compeleceng.2012.07.004

Guo, F., Wang, M., Li, Y. & Yan, H. (2011). The vulnerabilities and status of CAPTCHAs. *IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, s.448-452. DOI: 10.1109/ICCSN.2011.6014761

Huang, V. & Huang, R. & Chiang, M. (2013). A DDoS Mitigation System with Multi-Stage Detection and Text-Based Turing Testing in Cloud Computing. *27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, s.655-662. DOI: 10.1109/WAINA.2013.94

Husák, M., Čermák, M., Jirsík, T. & Čeleda, P. (2015). Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting. *10th International Conference on Availability, Reliability and Security (ARES)*, s.389-396. DOI: 10.1109/ARES.2015.35

Incapsula (2014). Matthews, T. Incapsula Survey: What DDoS Attacks Really Cost Businesses. URL: http://lp.incapsula.com/rs/incapsulainc/images/eBook%20-%20DDoS%20Impact%20Survey.pdf (Viitattu 15.5.2016)

Incapsula (2016). Global DDoS Threat Landscape Q4 2015. https://www.incapsula.com/ddos-report/ddos-report-q4-2015.html (Viitattu 8.2.2016)

InfosecStuff (2011), Baldwin, M. Mitigating the Apache Range Header DoS Vulnerability. http://infosecstuff.com/mitigating-the-apache-range-header-dos-vulnerability/ (Viitattu 14.5.2016)

Johnston A. (2001). *SIP:Understanding the Session Initiation Protocol.* Artech House, Massachusetts.

Jose, A.S. & Binu, A. (2014). Automatic Detection and Rectification of DNS Reflection Amplification Attacks with Hadoop MapReduce and Chukwa. *4th International Conference on Advances in Computing and Communications (ICACC),* s.195-198. DOI: 10.1109/ICACC.2014.54

Jouslehto, M. (2016). Eduskuntaan on tehty palvelunestohyökkyäys. *Kauppalehti.* URL: http://www.kauppalehti.fi/uutiset/eduskuntaan-on-tehty-palvelunestohyokkays/mFbQTLdz (Viitattu 27.5.2016)

Khandelwal, S. (2016). 602 Gbps! This may have been the largest DDoS attack in history. *The hacker News.* URL: http://thehackernews.com/2016/01/biggest-ddos-attack.html (Viitattu 27.5.2016)

Lau, F., Rubin S., Smith M. & Trajković L. (2000). Distributed denial of service attacks. *IEEE International Conference on Systems, Man and Cybernetics,* vol.3, s.2275-2280. DOI: 10.1109/ICSMC.2000.886455

Lee, S. & Kim, G. & Kim, S. Sequence-order-independent network profiling for detecting application layer DDoS attacks. *EURASIP Journal on Wireless Communications and Networking,* 2011:50. DOI: 10.1186/1687-1499-2011-50

Lennon, M. (2016). Britain's HSBC recovers from massive DDoS attack. *Security Week.* URL: http://www.securityweek.com/britains-hsbc-recovers-massive-ddos-attack (Viitattu 27.5.2016)

Li, W. & Chen, L. & Lei, Z. (2010). Alleviating the Impact of DNS DDoS Attacks. *2nd International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC),* vol. 1, s.240-243. DOI: 10.1109/NSWCTC.2010.63

Liao, Q., Li, H., Kang, S. & Liu, C. (2014). Feature Extraction and Construction of Application Layer DDoS Attack Based on User Behavior. *33rd Chinese Control Conference (CCC),* s.5492-5497. DOI: 10.1109/ChiCC.2014.6895878

Limkar, S. & Jha, R.K. (2012). An Effective Defence Mechanism for Detection of DDoS Attack on Application Layer Based on Hidden Markov Model. *Proceedings of the International Conference on Information Systems Design and Intelligent Applications: Advances in Intelligent and Soft Computing,* vol.132, s.943-950. DOI: 10.1007/978-3-642-27443-5_108

Liu, A.X. & Torng, E. & Meiners C.R. (2011). Compressing Network Access Control Lists. *IEEE Transactions on Parallel and Distributed Systems,* vol.22, no.12, s.1969-1977. DOI: 10.1109/TPDS.2011.114

Lou, X. & Hwang, K. & Hu, Y. (2009). Accountable File Indexing against DDos Attacks in Peer-to-Peer Networks. *IEEE Global Telecommunications Conference (GLOBECOM 2009),* s.1-6. DOI: 10.1109/GLOCOM.2009.5425979

Maathuis, I. & Smit, W. (2003). The battle between standards: TCP/IP vs OSI victory through path dependency or by quality. *The 3rd Conference on Standardization and Innovation in Information Technology, 2003,* s.161-176. DOI: 10.1109/SIIT.2003.1251205

Malecki, F. (2012). Simple ways to dodge the DDoS bullet. *Network Security,* vol.2012, no.8, s.18-20. DOI: 10.1016/S1353-4858(12)70075-2

McGovern, J., Tyagi, S., Stevens, M. & Mathew, S. (2003). *Java web services architecture.* Morgan Kaufmann Publishers, California.

Mirkovic, J. & Reiher, P. (2004). A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communications Review,* vol.34, no.2, April 2004, s.39-54.

Morein W.G., Stavrou A. Cook, D.L., Keromytis, A.D., Misra, V. & Rubenstein, D. (2003). Using Graphic Turing Tests to counter automated DDoS attacks against Web servers. *Proceedings of the 10th ACM International Conference on Computer and Communications Security (CSS).* s.8-19. DOI: 10.1145/948109.948114

Moustis, D. & Kotzanikolaou, P. (2013). Evaluating security controls against HTTP-based DDoS attacks. *Fourth International Conference on Information, Intelligence, Systems and Applications (IISA),* s.1-6. DOI: 10.1109/IISA.2013.6623707

Nazario, J. (2008). DDoS attack Evolution. *Network Security,* July 2008, s.7-10. DOI: 10.1016/S1353-4858(08)70086-2

Ndibwile, J.D., Govardhan, A., Okada, K. & Kadobayashi, Y. (2015). Web Server Protection against Application Layer DDoD Attacks using Machine learning and Traffic Authentication. *IEEE 39th Annual Computer Software and Applications Conference (COMPSAC),* vol.3, s.261-267. DOI: 10.1109/COMPSAC.2015.240

Networks Training: Intelligent DDoS Mitigation Systems-IDMS URL: http://www.networkstraining.com/intelligent-ddos-mitigation-system-idms/ (Viitattu 25.5.2016)

Oppliger, R. (2009). *SSL and TLS: Theory and Practice.* Artech House, Massachusetts.

Park, J., Iwai, K., Tanaka, H., Kurokawa, T. (2014). Analysis of Slow Read DoS attack. *International Symposium on Information Theory and its Applications (ISITA 2014).* s.60-64.

Paxson, V. (2001). *An Analysis of Using Reflectors for Distributed Denial of Service Attacks.* ICIR.org. URL: http://www.icir.org/vern/papers/reflectors.CCR.01/reflectors.html (Viitattu 25.11.2016)

Porter, T., Kanclirz, J., Zmolek, A., Rosela, A., Cross, M., Chaffin, L., Baskin, B. & Shim, C. (2006). *Practical VoIP Security.* Syngress Publishing Inc. Massachusetts.

Puhakka, K. (2016). Valtion verkkosivuilla palvelunestohyökkäys. *Kouvolan sanomat.* URL: http://www.kouvolansanomat.fi/Online/2016/05/11/Valtion%20verkkosivuilla%20palvelunestohy%C3%B6kk%C3%A4ys/2016848/4 (Viitattu 27.5.2016)

Raghavan S. & Dawson, E. (2011). *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks.* Springer India, New Delhi.

Rooney, T. (2010). *Introduction to IP address Management.* Wiley-IEEE Press, New Jersey.

Sachdeva, M. & Kumar, K. & Singh, G. (2016). A comprehensive approach to discriminate DDoS attacks from flash events. *Journal of Information Security and Applications,* vol.26, s.8-22. DOI: 10.1016/j.jisa.2015.11.001

Saied, A. & Overill, R.E. & Radzik, T. (2016). Detection of known and unknown DDoS attacks using Artificial Neural Networks. *Neurocomputing,* vol.172, s.385-393. DOI: 10.1016/j.neucom.2015.04.101

Schreier, J. (2016). Blizzard hit by multiple DDoS attacks. *Kotaku.* URL: http://kotaku.com/blizzard-hit-by-multiple-ddos-attacks-1770920820 (Viitattu 27.5.2016)

Shetty, N. & Prasad, N. & Nalini, N. (2015). *Emerginf Research in Computing, Information, Communication and Applications: ERCICA 2015,* vol.1. Springer India, New Delhi.

Singh, K.J. & De, T. (2015). DDoS Attack detection and mitigation technique based on HTTP count and verification using CAPTCHA. *International Conference on Computational Intelligencec and Networks (CINE),* s.196-197. DOI: 10.1109/CINE.2015.47

Sirisha, N., Durga Sri, B., Divya, P. & Durga, K.B.K.S. (2015). Detection and Protection of Application-Layer DDoS Attacks for Websites. *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 5, no. 5, s.1315-1320.

St Denis, T. (2006). *Cryptography for developers.* Syngress Publishing, Massachusetts.

Stevanovic D. & Vlajic, N. (2014). Next generation application-layer DDoS defences: Applying the concepts of outlier detectioning data streams with concept drift. *13th International Conference on Machine Learning and Applications (ICMLA)*, s.456-462. DOI: 10.1109/ICMLA.2014.80

Vissers, T., Somasundaram, T.S., Pieters, L., Govindarajan, K. & Hellinckx, P. (2014). DDoS defense system for web services in a cloud environment. *Future Generation Computer Systems,* vol.37, s.37-45. DOI: 10.1016/j.future.2014.03.003

Vänskä, O. (2016). Eduskunta myönsi palvelunestohyökkäyksen – hakkerit pilkkaavat kyberpuolustusta. *Tivi.* URL: http://www.tivi.fi/Kaikki_uutiset/eduskunta-myonsi-palvelunestohyokkayksen-hakkerit-pilkkaavat-kyberpuolustusta-6535304 (Viitattu 27.5.2016)

Wang, J. & Yang, X. & Long, K. (2010). A New Relative Entropy Based App-DDoS Detection Method. *IEEE Symposium on Computers and Communications (ISCC),* s.966-968. DOI: 10.1109/ISCC.2010.5546587

Wang, J. & Yang, X. & Long, K. (2011). Web DDoS Detection Schemes Based on Measuring User's Access Behaviour with Large Deviation. *IEEE Global Telecommunications Conference (GLOBECOM 2011),* s.1-5. DOI: 10.1109/GLOCOM.2011.6133798

Yadav, S. & Selvakumar S. (2015). Detection of application layer DDoS attack by modeling user behavior using logistic regression. *4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, s.1-6. DOI: 10.1109/ICRITO.2015.7359289

Yagatai, T. & Isohara, T. & Sasase, I. (2007). Detection of HTTP-GET flood Attack Based on Analysis of Page Access Behavior. *IEEE Pasific Rim Conference on Communications, Computers and Signal Processing (PacRim 2007),* s.232-235. DOI: 10.1109/PACRIM.2007.4313218

Yaibuates, M & Chaisricharoen, R. (2014). ICMP based Malicious Attack Identification Method for DHCP. *4th Joint International Conference on Information and Communication Technology, Electronic and Electrical Engineering (JICTEE),* s.1-5. DOI: 10.1109/JICTEE.2014.6804073

Ye, C. & Zheng, K. (2011). Detection of Application Layer Distributed Denial of Service. *International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 1, s.310-314. DOI: 10.1109/ICCSNT.2011.6181964

Ye, C. & Zheng, K. & She, C. (2012). Application layer DDoS detection using clustering analysis. *2nd International Conference on Computer Science and Network Technology (ICCSNT),* s.1038-1041. DOI: 10.1109/ICCSNT.2012.6526103

Yu, J., Fang, C., Lu, L. & Li, Z. (2010). Mitigating application layer distributed denial of service attacks via effective trust management. *IET Communications,* vol.4, no.16, s.1951-1962. DOI: 10.1049/iet-com.2009.0809

Zhang, M. & Zhang, W. & Fan, K. (2012). Application Layer DDoS Detection Model Based on Data Flow Aggregation and Evaluation. *International Conference on Communications and Information Processing (ICCIP),* s.37-45. DOI: 10.1007/978-3-642-31968-6_5

Zhou, W., Jia, W., Wen, S., Xiang, Y. & Zhou, W. (2014). Detection and defense of application-layer DDoS attacks in backbone web traffic. *Future Generation Computer Systems,* vol.38, s.36-46. DOI: 10.1016/j.future.2013.08.002

Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Niemelä, A. & Siltanen, J. (2015). Data Mining Approach for Detection of DDoS Attacks Utilizing SSL/TLS Protocol. *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: Lecture Notes in Computer Science*, vol.9247, s.274-285. DOI: 10.1007/978-3-319-23126-6_25

Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Siltanen, J. & Niemelä, A. (2016). Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. *23rd International Conference on Telecommunications (ICT 2016),* Thessaloniki, Greece, 16-18 May 2016.

Liitteet

A Käytetyt ohjelmakoodit

SlowHTTPTest-komennot:

SlowRead:
lowhttptest -c 1000 -X -g -o slowread_stats2 -i 110 -r 1000 -w 10 -y20 -n 5 -x 32 -u https://web.ties327.jyu.fi/accounts/index.php -p 5 -l 350

SlowLoris:
slowhttptest -c 200 -H -g -o slowloris_stats2 -i 110 -r 200 -s 8192 -t GET -u https://web.ties327.jyu.fi/accounts/index.php -x 24 -p 3

RangeAttack:
slowhttptest -c 200 -R -g -o rangeattack_stats -i 110 -r 23 -s 4096 -u https://192.168.1.3/accounts/index.php

RUDY:
slowhttptest -c 200 -B -g -o rudy_stats2 -i 110 -r 200 -s 8192 -t POST -u https://web.ties327.jyu.fi/accounts/index.php -x 10 -p 3

Nmap:

nmap 192.168.1.3

SSLsqueeze:

./sslsqueeze 192.168.1.3 443 2048 23