Vesa Huuskonen

Palvelimen suojaaminen DDoS-hyökkäyksiltä

Tietotekniikan Pro gradu-tutkielma

14. kesäkuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Vesa Huuskonen

Yhteystiedot: vesa.p.huuskonen@student.jyu.fi

Ohjaaja: Timo Hämäläinen

Työn nimi: Palvelimen suojaaminen DDoS-hyökkäyksiltä

Title in English: Protecting server against DDoS attacks

Työ: Pro gradu-tutkielma

Opintosuunta: Tietoliikenne

Sivumäärä: 78+7

Tiivistelmä: Tutkielmassa käytiin läpi palvelunestohyökkäyksiin liittyviä protokollia sekä eräitä tunnettuja hyökkäysmenetelmiä. Teoriaosuudessa perehdyttiin myös aiheesta jo tehtyihin tutkimuksiin. Hyökkäyksien havaitsemismenetelmiä käytiin läpi niin teoriatasolla kuin käytännössäkin. Hyökkäyksiltä suojautumista tutkittiin pilvipalveluiden osalta kirjallisuuskatsauksena käymällä läpi palveluiden materiaalia ja käytännön osuudessa pystyttämällä oma virtuaaliympäristö. Ympäristö koostui kahdesta komentorivipohjaisesta Ubuntupalvelimesta, Ubuntun työpöytäversiota käyttävistä asiakaskoneista ja kolmesta Kali Linuxia käyttävästä hyökkäävästä koneesta. Puolustus- ja havaitsemisjärjestelminä testauksessa käytettiin Linuxin iptables palomuuria sekä Snort tunkeutumisen havaitsemisjärjestelmää. Snortia testattiin sekä tunkeutumisen havaitsemistilassa että tunkeutumisen estotilassa. Tutkimuksessa havaittiin, että mahdollisuudet paikalliseen puolustautumiseen riippuvat paljon hyökkäyksen tyypistä ja etenkin volumetristen hyökkäysten osalta ulkopuolisten pilvipohjaisten ratkaisujen käyttö tai Internetoperaattorin apu ovat välttämättömiä. Toisaalta hitaita sovelluskerrosten hyökkäyksiä vastaan puolustautuminen onnistui yllättävän hyvin avoimen lähdekoodin ohjelmistoilla.

Avainsanat: bottiverkko, DDoS, palvelunestohyökkäys

Abstract: This study reviewed the protocols related to denial of service attacks as well as some known attack methods. The theoretical part also examined the research already done on

the topic. Attack detection methods were reviewed both at the theoretical level and in practice. Protection against attacks was studied in the case of cloud services as a literature review by reviewing the material of the services and in the practical part by setting up its own virtual environment. The environment consisted of two command-line-based Ubuntu servers, client computers running the desktop version of Ubuntu, and three attack machines running Kali Linux. The Linux iptables firewall and the Snort intrusion detection system were used as defense and detection systems in the testing. Snort was tested in both the intrusion detection mode and the intrusion prevention mode. The study found that the potential for local defense depends largely on the type of attack, and especially for volumetric attacks, the use of external cloud-based solutions or the assistance of an Internet operator is essential. On the other hand, defending against slow application layer attacks was surprisingly successful with open source software.

Keywords: botnet, DDoS, denial of service attack

Termiluettelo

AED	Arbor Edge Defence
APS	Availability Protection System
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
Captcha	Completely Automated Public Turing test to tell Computers and Humans Apart
CLDAP	Connection-less Lightweight Directory Access Protocol
DDoS	Distributed Denial of Service, hajautettu palvelunestohyökkäys
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System, Internetin nimipalvelujärjestelmä
GRE	Generic Routing Encapsulation
Http	Hypertext Transfer Protocol
Https	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IoC	Indicator of Compromise, esimerkiksi virustunniste, haittaohjelman MD5 tiiviste tai botnetin URL tai domain-nimi
IoT	Internet of Things, esineiden Internet
IP	Internet Protocol
NAT	Network Address Translation
OSI	Open System Interconnection
PoD	Ping of Death
SDN	Software-Defined Networking
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSDP	Simple Service Discovery Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol

TTL	Time To Live
MAC	Media Access Control
mDNS	Multicast Domain Name System
MSSP	Managed Security Service Provider
NTP	Network Time Protocol
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
WAF	Web Application Firewall
WLAN	Wireless Local Area Network

Kuviot

Taulukot

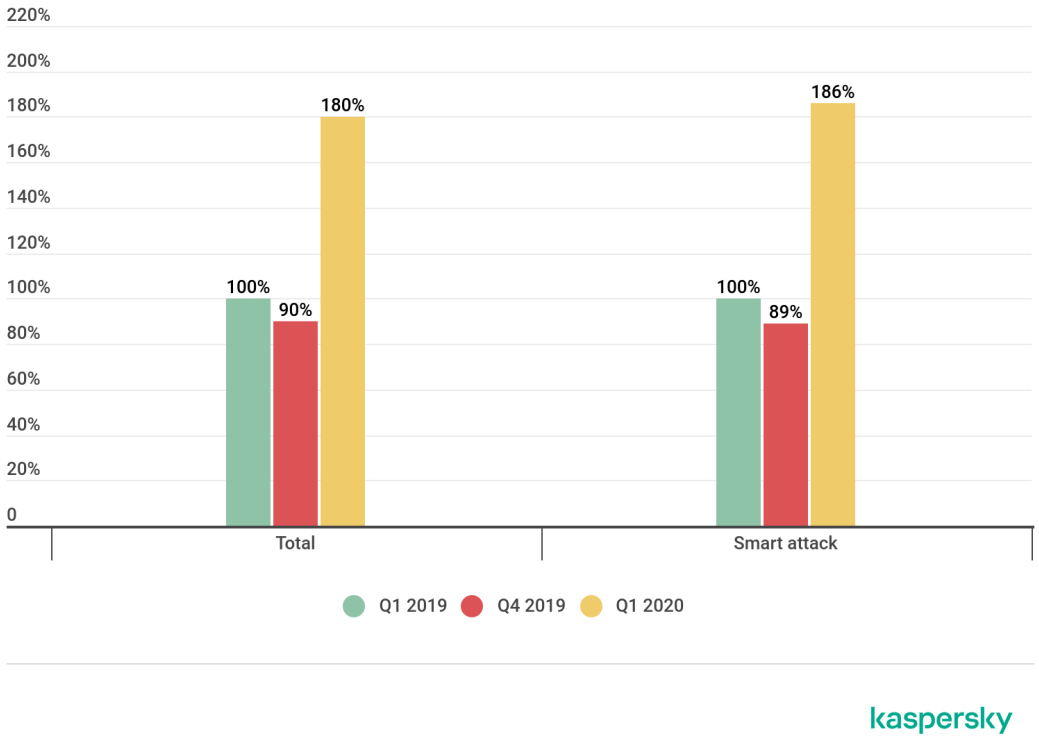
Sisältö

1 Johdanto

Yhteiskunnan riippuvuuden Internetistä kasvaessa ja palvelujen siirtyessä yhä enemmän ainoastaan verkkopohjaisiksi myös haavoittuvuus verkon katkoksille lisääntyy. Internetiin liitettyjen laitteiden määrä kasvaa jatkuvasti eikä tavallisilla kuluttajilla ole aina osaamista kodin laitteiden suojaamisesta. Nämä laitteet voivat joutua rikollisiin tarkoituksiin käytettävän bottiverkon jäseniksi. Bottiverkkoon voidaan kaapata perinteisten tietokoneiden lisäksi esimerkiksi televisiot, jääkaapit, WLAN-tukiasemat, reitittimet, valvontakamerat ja periaatteessa melkeinpä mikä tahansa verkkoon liitetty suojaamaton tai oletussalanalla oleva IoT-laite. Bottiverkko voi sisältää helposti satoja tuhansia laitteita joilla kohteelle voidaan aiheuttaa volumetrisissä hyökkäyksissä jopa satojen gigabittien liikenne sekunnissa. Lyhytkin katkos esimerkiksi pankkien palveluissa estää maksamisen kortilla ja käteisen käytön vähyyden vuoksi käytännössä lamauttaa päivittäistavarakaupan ja palveluja tarjoavien yritysten toiminnan. Yrityksille jo pienikin katkos verkkopalvelujen toiminnassa voi aiheuttaa vähintäänkin imagohaittaa tai merkittäviäkin taloudellisia menetyksiä esimerkiksi tuotannon ohjauksen lamaantuessa tai verkkokaupan kaatuessa ja asiakkaiden siirtyessä kilpailevien yritysten sivuille.

Hyökkäysten määrä vaihtelee ja esimerkiksi poliittiset tilanteet, vaalit, yritysten toiminta ja monet muut asiat vaikuttavat hyökkäysten määrään ja kohteisiin. Alkuvuonna 2020 COVID-2019 pandemia on Kasperskyn mukaan näkynyt niin Internetin käytössä yleisesti kuin myös hyökkäysten määrässä ja kohteissa. Eniten hyökkäysten kohteeksi ovat vuoden ensimmäisen neljänneksen aikana joutuneet lääketieteelliset organisaatiot, toimituspalvelut ja peli- ja koulutusalustat. Maaliskuun puolivälissä hyökkääjät yrittivät lamauttaa Yhdysvaltain sosiaali- ja terveysministeriön web-sivut. Tavoitteena oli estää ihmisiä saamasta oikeaa tietoa pandemiasta. Samalla tuntemattomat toimijat levittivät väärää tietoa sosiaalisessa mediassa ja sähköpostilla. Pariisissa hyökkäyksen kohteeksi joutui sairaalaryhmän järjestelmät. Hyökkäyksen takia etätyöntekijät eivät päässeet käsiksi sairaalan sovelluksiin ja sähköpostiin. Saksassa ja Hollannissa ruokaa toimittavat yritykset joutuivat hyökkäyksen kohteeksi. Ne pystyivät vastaanottamaan tilauksia mutta eivät käsittelemään niitä, joten ne joutuivat palauttamaan rahat asiakkaille. Lisäksi rikolliset vaativat toiselta yritykseltä lunnaita kaksi Bitcoi-

nia hyökkäyksen pysäyttämiseksi. Saksalainen etäopiskelualusta Mebis joutui myös hyökkäyksen kohteeksi jo ensimmäisenä etäkoulupäivänä. Järjestelmä oli alhaalla useita tunteja. (“Kaspersky. DDoS Attacks in Q1 2020.” 2020)

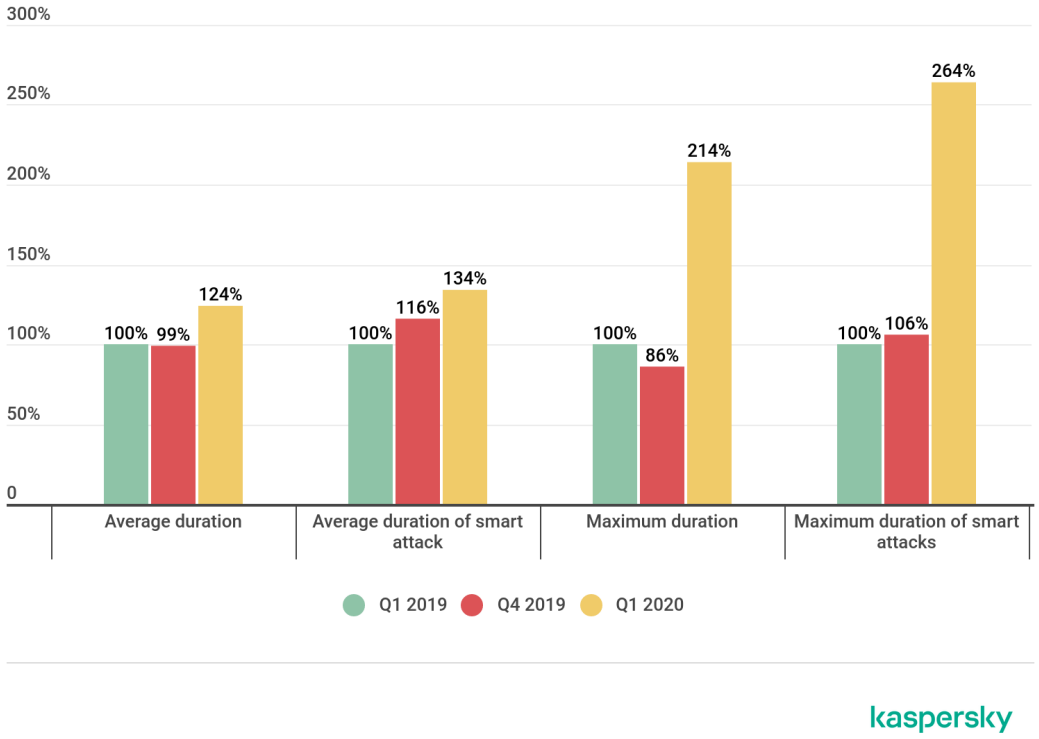


Kuvio 1. DDoS-hyökkäysten määrä Q1 2020 ja Q1 sekä Q4 2019

Kuviosta 1 näemme DDoS-hyökkäysten määrän muutoksen Q1 2020 ja Q1 sekä Q4 2019. Q1 2019 määrää on käytetty vertailuarvona (100%). Kuvasta nähdään, että hyökkäysten määrä kasvoi edelliseen neljännekseen verrattuna kaksinkertaiseksi.

Kuviosta 2 näemme hyökkäysten keston kehityksen Q1 2020 verrattuna Q1 ja Q4 2019. Q1 2019 arvoa on käytetty vertailuarvona 100%.

Palvelunestohyökkäyksien tekeminen on nykyään varsin helppoa. Hyökkäyksen tekevä taho ei välttämättä tarvitse teknistä osaamista vaan hyökkäys voidaan ostaa palveluna. Hyökkäyksissä motivaationa voi olla taloudelliset tai ideologiset syyt, poliittinen vaikuttaminen, oman



Kuvio 2. Hyökkäysten kesto Q1 2020 ja Q1 sekä Q4 2019

osaamisen todistaminen tai “hyökkäys” voi olla jopa vahinko. Esimerkiksi mielenkiintoisen linkin julkaiseminen suositulla sivustolla voi johtaa kyseisen palvelun ylikuormittumiseen. Tässä työssä on tarkoituksena tutkia menetelmiä hyökkäysten havaitsemiseen ja vaikutusten minimointiin.

1.1 Tutkimusongelma

Tässä pro gradu-tutkielmassa perehdytään palvelunestohyökkäyksiin ja erityisesti palvelimen suojaamiseen palvelunestohyökkäyksiä vastaan. Työssä käsitellään hyökkäysmenetelmiin liittyvät protokollat, hyökkäysmenetelmien tyypit ja niiden väliset erot sekä niiden estämiseen tai lieventämiseen liittyvät menetelmät.

Tutkielmassa on tarkoituksena myös simuloida hyökkäyksia ja testata löydettyjen suojausmenetelmien tehoa hyökkäyksien lieventämiseen ja torjumiseen. Pilvipohjaisten palveluiden

ja erillisten verkon reunalle asennettavien laitteistojen testaaminen on resurssien puutteen vuoksi jätetty pois ja niiltä osin palvelut käydään läpi kirjallisuuskatsauksena.

2 Protokollat

Protokolla on yhteyskäytäntö, jonka avulla laitteet tai ohjelmistot pystyvät keskustelemaan keskenään. Se varmistaa tietoliikenteen yhteensopivuuden. Yleisin nykyään käytettävä protokollaperhe on TCP/IP. Se jaetaan lähteestä riippuen kolmesta seitsemään eri kerrokseen, joita ovat nelikerroksisen TCP/IP mallin mukaisesti 1. Sovelluskerros, 2. Siirtokerros (TCP), 3. Internet-kerros (IP) ja 4. Verkkokerros. Toinen tunnettu malli on OSI-mallin mukainen jako seitsemään eri kerrokseen: 1. Fyysinen kerros, 2. Siirtokerros, 3. Verkkokerros, 4. Kuljetuskerros, 5. Istuntokerros, 6. Esitystapakerros ja 7. Sovelluskerros. Seuraavassa käydään läpi kerroksittain verkkoliikenteeseen ja palvelunestohyökkäyksiin liittyvät tärkeimmät protokollat.

Seuraavassa taulukossa esitetään TCP/IP ja OSI mallien vastaavuudet:

TCP/IP malli	OSI malli
Sovellus	Sovellus Esitystapa Istunto
Siirto	Siirto
Internet	Verkkokerros
Verkkokerros	Siirtokerros Fyysinen kerros

Taulukko 1. TCP/IP ja OSI vastaavuudet

(“Study-ccna.com” 2019)

2.1 ARP

ARP (Address Resolution Protocol) on protokolla, jolla Ethernet-verkoissa selvitetään loogista (IP-osoite) vastaava fyysinen osoite (MAC-osoite).

(“RFC826” 1982)

ARP-protokollaa voidaan hyödyntää DDoS-hyökkäyksissä esimerkiksi aiheuttamalla ARP-myrsky. Tämä onnistuu, jos merkittävä osa verkon koneista saadaan saastutettua madolla. Hyökkäyksessä DDoS-agenttina toimiva madon saastuttama tietokone lähettää jatkuvasti ARP-kyselyjä verkon yhdyskäytävälle tai toisille koneille kuluttaen niiden resursseja ja estäen normaalia verkkoliikennettä.

(Kumar 2005)

2.2 BGP

BGP (Border Gateway Protocol) on Internetin reititysprotokolla joka reitittää liikennettä autonomisten järjestelmien välillä mahdollisimman tehokkaasti. Autonominen järjestelmä (Autonomous system, AS) on tyypillisesti yksittäisen suuren organisaation (ISP, yliopistot, valtion laitokset, tieteelliset instituutiot) useista reitittimistä koostuva verkko. Internet puolestaan koostuu näistä yhteenliitetyistä verkoista joiden välistä liikennettä BGP reitittää tehokkainta mahdollista reittiä pitkin. DDoS-suojauksessa BGP-protokollaa voidaan käyttää reitityksen muuttamiseen niin, että normaalisti palvelimen verkkoon menevä liikenne ohjataan suojapalvelua tarjoavan yrityksen verkkoon.

(“RFC1105” 1989) (“BGP Routing Explained” 2020)

2.3 CLDAP

CLDAP (Connection-less Lightweight Directory Access Protocol) on 1995 julkaistu protokolla. Se oli suunnattu sovelluksiin joiden tarvitsi hakea pieniä määriä hakemistoissa säilytettyä dataa. Protokolla oli suunniteltu nimensä mukaisesti kevyeksi eikä se aiheuttanut yhteyden avaamisesta tai sulkemisesta ylimääräistä kuormaa. Protokolla ei kuitenkaan yleistynyt eikä edennyt varsinaisen standardin asteelle. Osittain syynä oli rajallinen toiminnallisuus ja tietoturvan sekä datan eheyden varmistamisen puutteet. Protokollan alun perin määritellyt RFC1798 on määritelty statuksella Historic. Protokolla käyttää porttia UDP 389.

(“RFC3352” 2003)

2.4 GRE

GRE (Generic Routing Encapsulation) on Ciscon kehittämä tunnelointiprotokolla. Sitä käytetään muodostamaan kahden verkkolaitteen välisiä yhteyksiä. Protokollaa käytetään osana pilvipohjaista suojausta muodostamaan tunneloitu yhteys palveluntarjoajan verkon ja suojattavan palvelimen tai verkon välillä.

(“RFC2784” 2000) (“RFC2890” 2000)

2.5 Http

Http (Hypertext Transfer Protocol) on asiakas-palvelin protokolla jonka avulla asiakas voi pyytää web-sivua palvelimelta. Asiakas, tyypillisesti internetselain, lähettää HTTP Request-viestin palvelimelle. Palvelin vastaa pyydetyllä web-sivulla. Http on käytännössä jäänyt pois käytöstä julkisessa Internetissä sijaitsevista palveluista, koska salaamaton liikenne ei täytä nykyisiä tietoturvavaatimuksia eikä anna tapoja varmistaa käytetyn palvelun aitoutta. Oletuksena Http käyttää TCP-porttia 80. Http-protokolla on haavoittuva erityisesti hitaille hyökkäyksille, kuten esimerkiksi Slowloris ja Http Flood.

(“RFC2616” 1999) (“RFC7231” 2014)

2.6 Https

Https (Hypertext Transfer Protocol Secure) on salattu versio Http-protokollasta. Https käyttää nykyään TLS-protokollaa (Transport Layer Security) liikenteen salaamiseen. Aiemmissa versioissa käytössä oli SSL-salaus (Secure Sockets Layer). Https käyttää oletuksena TCP-porttia 443. Salaus ei kuitenkaan suojaa DDoS-hyökkäyksiltä, vaan myös https-protokolla on haavoittuva erilaisille sovelluskerroksen hyökkäyksille.

(“RFC2818” 2000)

2.7 DNS

DNS (Domain Name System) on Internetin nimipalvelujärjestelmä joka mahdollistaa ihmiselle helpommin muistettavien nimien käyttämisen IP-osoitteiden sijaan. Protokollan avulla verkkotunnukset siis muutetaan IP-osoitteiksi. Nimipalvelua käytetään myös sähköpostin reititykseen. DNS käyttää UDP-porttia 53. DNS-protokollaa voidaan käyttää välineenä heijastushyökkäyksissä tai suoraan DNS-palvelimia kohtaan suoritetulla hyökkäyksellä voidaan lamauttaa suuria osia verkosta.

(“RFC1034” 1987) (“RFC1035” 1987)

DNS-palvelimille voidaan myös injektoida väärennettyjä tietueita ja näin saada käyttäjät ohjattua joko olemattomalle tai väärennetylle ja mahdollisia haittaohjelmia sisältäville sivustoille (DNS Spoofing). DNS Spoofing voidaan toteuttaa kahdella tavalla, joko manipuloimalla DNS-palvelimen tietokantaa suoraan (Planted attack) tai väärentämällä DNS-palvelimen lähettämä vastauspaketti. Paketin väärentämisessä käytetään hyväksi DNS-protokollan sisältämää pakettinumeroa. Vastauspaketin numero väärennetään vastaamaan uhrin lähettäämää DNS-kyselyä ja pidetään huoli siitä, että uhri saa väärennetyn paketin ensimmäisenä.

(Zhang 2013)

2.8 DHCP

Dynamic Host Configuration Protocol on protokolla, jolla verkkoon liitetty laite pyytää verkkoyhteyteen tarvittavia parametrejä DHCP-palvelimelta. Näitä ovat esimerkiksi IP-osoite, aliverkon peite, yhdyskäytävä, domainin nimi ja DNS-palvelimen osoite. Palvelin voi olla erillinen laite tai yhdistetty esimerkiksi palomuuriin. DHCP käyttää palvelimella UDP-porttia 67 ja asiakaskoneella UDP-porttia 68.

(“RFC2132” 1997)

2.9 mDNS

mDNS (multicast DNS) protokolla selvittää isäntänimiä IP-osoitteiksi pienissä verkoissa, joissa ei ole paikallista nimipalvelinta. Protokolla käyttää UDP-porttia 5353.

(“RFC6762” 2013)

2.10 NetBIOS over TCP/IP

NetBIOS (Network Basic Input/Output System) over TCP/IP on protokolla, joka mahdollistaa NetBIOS API:n käyttämisen TCP/IP-verkossa. NetBIOS tarjoaa kolme erillistä palvelua: Nimipalvelu tietokoneen nimen rekisteröimiseen ja selvittämiseen (portit 137 UDP ja TCP), yhteydettömän kommunikaaton (portti UDP 138) ja yhteydellisen kommunikaatiopalvelun (portti TCP 139).

(“RFC1001” 1987) (“RFC1002” 1987)

2.11 NTP

NTP (Network Time Protocol) on sovelluskerroksen protokolla jota käytetään tietokoneen tai verkkolaitteen kellon synkronointiin TCP/IP-verkossa. Jo viiden minuutin ero kellonajassa aiheuttaa ongelmia AD-domainiin kirjautumisessa. NTP käyttää hierarkista järjestelmää jossa ylimpänä on tarkat atomi- tai GPS-kellot, stratum 0 palvelimet. Stratum 1 palvelimet ovat suoraan linkitetty startum 0 palvelimiin ja niin edelleen. NTP käyttää UDP-porttia 123.

(“RFC5905” 2007)

2.12 SMTP

SMTP (Simple Mail Transfer Protocol) on protokolla, jota käytetään viestien välittämiseen sähköpostipalvelimien välillä. SMTP käyttää TCP-porttia 25. Portti on yleensä nykyään suljettu domainin ulkopuolisilta käyttäjiltä roskapostin vähentämiseksi. Turvattoman SMTP-protokollan seuraajaksi on kehitetty parannettu versio ESMTP, joka lisää protokollaan muun-

muassa salauksen ja autentikoinnin.

(“RFC821” 1982)

2.13 ESMTP

ESMTP (Extended/Enhanced Simple Mail Transfer Protocol) on laajennus SMTP-protokollaan. Ensimmäisen kerran ESMTP määriteltiin vuonna 1995 RFC:ssä 1869. Protokollassa on tuki autentikoinnille ja salaukselle, jotka alkuperäisestä SMTP-protokollasta puuttuivat.

(“RFC1869” 1995) (“RFC5321” 2008)

2.14 SNMP

SNMP (Simple Network Management Protocol) on alunperin vuonna 1988 määritelty verkkojen hallintaan käytettävä tietoliikenneprotokolla. Protokollan avulla voidaan kysellä verkossa olevien laitteiden tiloja tai laitteet voivat itsenäisesti antaa hälytyksiä. Protokollasta on kolme versiota, alkuperäinen SNMP, versio 2 joka ei yleistynyt ja SNMPv3 jossa tietoturvaan on kiinnitetty hieman enemmän huomiota. Siinä on esimerkiksi salaus sisäänrakennettuna. SNMP käyttää UDP-porttia 161 kyselyihin ja UDP-porttia 162 hälytyksiin.

(“RFC1157” 1990) (“RFC3410” 2002)

2.15 SOAP

SOAP (Simple Object Access Protocol) on World Wide Web Consortiumin ylläpitämä http:n päällä toimiva sovelluskerroksen protokolla, jonka avulla sovellukset voivat viestiä keskenään. SOAP-viesti on tavallinen XML-dokumentti sisältäen seuraavat elementit:
Envelope: Identifioi XML dokumentin SOAP-viestiksi
Header: Sisältää otsikkotiedot
Body: Sisältää kutsu- ja vastaustiedon
Fault: Sisältää status- ja virhetiedot
SOAP-viestejä käytetään esimerkiksi SSDP heijastushyökkäyksen suorittamisessa.

(“SOAP” 2020)

2.16 SSDP

SSDP (Simple Service Discovery Protocol) protokollaa käytetään verkosta löytyvien palveluiden mainostamiseen ja etsimiseen joko hyvin vähäisellä staattisella konfiguroinnilla tai kokonaan automaattisesti. Protokolla on UPnP-protokollan perusta, ja tarkoitettu käytettäväksi pienissä koti- tai toimistoverkoissa. IPv4 verkoissa SSDP käyttää multicast-osoitetta 239.255.255.250 ja UDP-porttia 1900. IPv6-verkoissa osoitealuetta FF0X::C jossa X on mikä tahansa validi arvo.

(“IPv6 Multicast Address Space Registry” 2020).

SSDP:tä käytetään vahvistetussa heijastushyökkäyksessä väärentämällä uhrin IP-osoite ja lähettämällä muokattu SOAP-pyyntö, jolla UPnP-laite saadaan avaamaan yhteys Internettiin. Laite lähettää vastauksensa hyökkäyksen kohteena olevaan IP-osoitteeseen. SOAP-pyynnöstä riippuen vahvistuskerroin voi olla jopa 30.

(“Guide to DDoS Attacks November 2017” 2020).

2.17 SSH

SSH (Secure Shell) on verkkoprotokolla, jota käytetään laitteiden etäyhteyksiin ja -hallintaan. SSH käyttää julkiseen avaimeen perustuvaa salausta siirrettävän tiedon suojaamiseen. Koneessa johon yhteys otetaan täytyy olla SSH-palvelinohjelma asennettuna. Oletusportti on TCP 22.

(“RFC3410” 2002)

2.18 Telnet

Telnet mahdollistaa laitteen etähallinnan. Telnetissä ei ole minkäänlaista salausta vaan kaikki tieto, mukaanlukien käyttäjätunnukset ja salasanat siirretään verkon yli selväkielisenä. Proto-

kollaa ei tule käyttää julkisissa verkoissa vaan se tulee korvata esimerkiksi SSH-protokollalla. Oletuksena Telnet käyttää TCP-porttia 23.

(“RFC854” 1983)

2.19 TCP

TCP (Transmission Control Protocol) on kuljetuskerroksen protokolla joka tarjoaa luotettavan yhteyden sovellusten välille. TCP on yhteydellinen protokolla. Ennen tiedon siirtoa yhteys muodostetaan kolmiosaisella kättelyllä. Tämän jälkeen tiedon siirto alkaa ja sen päätyttyä yhteys katkaistaan. TCP käyttää kehysten numerointia ja kuittausta, joten lähettäjä voi uudelleenlähettää mahdollisesti hävinneet paketit. Kuittaukset ja otsikkotiedot lisäävät siirrettävän datan määrää joten tiedon siirron kannalta TCP ei ole tehokkain mahdollinen protokolla. Niinpä sitä käytetään sovelluksissa joissa tiedonsiirron luotettavuus on tärkeää.

(“RFC793” 1981)

TCP-portit numeroidaan väliltä 0-65535 ja alue jakaantuu seuraavasti: Järjestelmäportit eli tunnetut portit 0-1023 (IANAn määrittelemät), käyttäjäportit eli rekisteröidyt portit 1024-49151 (Iana) ja dynaamiset portit 49152-65535.

(“RFC6335” 2011)

2.20 MC-SQLR

MC-SQLR (SQL Server Resolution Protocol) on sovelluskerroksella toimiva yksinkertainen request/response-protokolla, jolla asiakas pystyy kysymään tietokantapalvelimelta palvelimeen ja tietokantoihin liittyviä ominaisuuksia. Protokolla ei sisällä mitään autentikointiin, tiedon suojaukseen tai luotettavuuteen liittyviä ominaisuuksia. Protokolla käyttää UDP-porttia 1434. Protokollaa käytetään MS SQL vahvistetun heijatushyökkäyksen suorittamiseen.

(“SQL Server Resolution Protocol” 2019)

2.21 UDP

UDP (User Datagram Protocol) on kuljetuskerroksen yhteydetön protokolla jota käytetään tiedonsiirtoon sovellusten välillä. Toisin kuin TCP, UDP ei takaa toimitettavan datan perille menoa tai järjestystä mutta on vastaavasti nopeampi ja kuormittaa verkkoa vähemmän. Käytetään yleensä sovelluksilla joilla pakettien uudelleenlähetys ei olisi järkevää (esim. VoIP, videon striimaus) tai joissa tiedonsiirron varmistus tapahtuu sovelluskerroksella.

UDP-porttien numerointi ja jako vastaa TCP-porttien jakoa. Sovellus voi käyttää joko UDP- tai TCP-porttia tai molempia.

(“RFC768” 1980)

2.22 UPnP

UPnp-protokolla (Universal Plug and Play) käyttää SSDP-protokollaan perustuvaa etsintäprotokollaa etsiäkseen muita verkossa olevia UPnp-laitteita. Protokolla käyttää UDP-porttia 1900.

(“RFC6970” 2013)

2.23 IP

IP (Internet Protocol) on verkkokerroksen protokolla. Verkon reitittimet välittävät IP-paketteja otsikkokentän perusteella ja ylemmän tason protokollat on kapseloitu IP-paketin dataosioon.

(“RFC791” 1981)

2.24 ICMP

ICMP (Internet Control Message Protocol) on verkkokerroksen protokolla jota käytetään IP-pakettien virheiden raportointiin. Yleisimpiä ICMP-protokollaa käyttäviä työkaluja ovat ping ja traceroute. Yhteyttä testaava laite lähettää Echo Requestin ja kohteen pitäisi vastata siihen Echo Replyllä. ICMP-protokollaa hyödyntäviä DoS-hyökkäyksiä ovat esimerkiksi

Ping of Death, jossa väärin muotoillulla tai liian suurella ping-paketilla saadaan kohdepalvelin jumiutumaan tai kaatumaan sekä Ping Flood jossa suurella määrällä ping-paketteja kulutetaan verkkoresursseja.

(“RFC792” 1981)

3 Hyökkäysmenetelmät

DDoS hyökkäykset voidaan jakaa volumetrisiin, TCP State-Exhaustion ja sovelluskerroksen hyökkäyksiin.

Volumetrisissä hyökkäyksissä tavoitteena on yksinkertaisesti kuluttaa koko kohteen verkkoliitännän kapasiteetti aiheuttamalla enemmän liikennettä kuin liitäntä pystyy käsittelemään. Palvelun toiminta estyy kun kaistaa ei enää riitä normaaleille käyttäjille.

TCP State-Exhaustion kohdistuu verkon aktiivilaitteisiin, esimerkiksi kuorman tasaajiin, palomuureihin tai palvelimiin. Riittävällä kuormituksella saadaan suurikapasiteettisetkin laitteet ylikuormitettua.

Sovelluskerroksen hyökkäykset ovat Netscoutin mukaan kaikkein vaarallisimpia ja vaikeimpia havaita ja lieventää. Hyökkäys tapahtuu nimensä mukaisesti OSI-mallin kerroksella 7, eli sovelluskerroksella. Näin ollen ne muistuttavat normaalia käyttäjien aiheuttamaa liikennettä. Tämän hyökkäyksen toteuttamiseen ei myöskään tarvita suuria bottiverkkoja, vaan jopa yhdellä laitteella ja hyvin pienellä liikennemäärällä voidaan aiheuttaa merkittävää kuormitusta kohteelle. (“Netscout. What is DDoS?” 2019)

Heijastushyökkäyksellä (Reflection DDoS Attack) tarkoitetaan hyökkäystä, jossa hyökkääjä väärentää aiotun uhrin IP-osoitteen ja lähettää sitten sinänsä laillisia pyyntöjä laillisille palvelimille. Vastaukset näihin pyyntöihin kohdistuvat väärennetyn IP:n ansiosta uhrin koneelle.

Vahvistushyökkäyksellä (Amplification Attack) tarkoitetaan hyökkäyksiä, joissa pieni alkuperäinen kysely tuottaa huomattavasti suuremman vastauksen. Vastauksen koko saattaa olla kymmeniä tai satoja kertoja suurempi kuin alkuperäinen pyyntö. Näin hyökkääjä pystyy pienellä oman verkkoliikenteen määrällä aiheuttamaan uhrille erittäin suuria liikennemääriä. Vahvistus- ja heijastushyökkäyksiä käytetään usein yhdessä.

Vahvistuskertoimet voivat olla hyvinkin suuria. Rossowin testien mukaan BAF (Bandwidth Amplification Factor) voi olla esimerkiksi NTP-protokollalla jopa 4670-kertainen. PAF (Packet Amplification Factor) Voi olla yli 10. Rossow määrittelee kertoimen BAF jakamalla

	BAF			PAF	
Protocol	all	50%	10%	all	Scenario
SNMP v2	6.3	8.6	11.3	1.00	GetBulk request
NTP	556.9	1083.2	4670.0	10.61	Request “monlist” statistics
DNSNS	54.6	76.7	98.3	2.08	ANY lookup at author. NS
DNSOR	28.7	41.2	64.1	1.32	ANY lookup at open resolv.
NetBios	3.8	4.5	4.9	1.00	Name resolution
SSDP	30.8	40.4	75.9	9.92	SEARCH request
CharGen	358.8	n/a	n/a	1.00	Character generation request
QOTD	140.3	n/a	n/a	1.00	Quote request
BitTorrent	3.8	5.3	10.3	1.58	File search
Kad	16.3	21.5	22.7	1.00	Peer list exchange
Quake 3	63.9	74.9	82.8	1.01	Server info exchange
Steam	5.5	6.9	14.7	1.12	Server info exchange
ZAv2	36.0	36.6	41.1	1.02	Peer list and cmd exchange
Sality	37.3	37.9	38.4	1.00	URL list exchange
Gameover	45.4	45.9	46.2	5.39	Peer and proxy exchang

Taulukko 2. Protokollien vahvistuskertoimet

vahvistavan palvelimen uhrille lähettämän UDP-hyötykuorman pituus vastaavalla pyynnön UDP-hyötykuorman pituudella:

$$BAF = \frac{len(UDPpayload)amplifiertovictim}{len(UDPpayload)attackertoamplifier}$$

PAF puolestaan määritellään jakamalla vahvistavan palvelimen lähettämä IP-pakettimäärä hyökkääjän palvelimelle lähettämien pakettien määrällä:

$$PAF = \frac{numberofpacketsamplifiertovictim}{numberofpacketsattackertoamplifier}$$

(Rossow 2014)

3.1 ARP Storm

ARP Storm eli ARP-myrsky on hyökkääjän luoma tilanne, jossa verkkoon generoidaan suuria määriä ARP-paketteja. ARP-kyselyissä voidaan käyttää aliverkkoon kuuluvia tai aliverkon ulkopuolisia IP-osoitteita. Verkon kuormittamisen ja näin aikaansaadun palveluneston lisäksi hyökkäyksen tavoitteena voi olla myös koko aliverkon IP-osoitteiden kerääminen myöhempiä hyökkäyksiä varten. Esimerkiksi Ettercap (“Ettercap Home Page” 2019) on hyökkäystyökalu, joka käyttää tätä menetelmää muodostaakseen listan verkossa käytössä olevista IP-osoitteista. Varsinainen hyökkäys tapahtuu saastuttamalla merkittävä osa verkon koneista madolla. Code red on eräs tunnettu ARP-myrskyn aiheuttaja. Hyökkäyksessä DDoS-agenttina toimiva madon saastuttama tietokone lähettää jatkuvasti suuria määriä ARP-kyselyjä verkon yhdyskäytävälle tai toisille koneille kuluttaen niiden resursseja ja estäen normaalia verkkoliikennettä. Aina ARP-myrsky ei kuitenkaan ole hyökkäyksestä johtuva, vaan se voi johtua myös verkkolaitteen virheellisestä konfiguroinnista, viallisesta laitteesta tai muun kuin ARP-myrskyn luomiseen tarkoitetun haittaohjelman virheellisestä toiminnasta.

(S. Vidya 2011)

3.2 CLDAP Reflection Attack with Amplification

Vaikka CLDAP-protokolla ei yleistynytkään, hyökkäyksiä on silti viimevuosina havaittu yhä suurempia määriä. CLDAP-hyökkäykset suoritetaan käyttämällä hyökäysskriptejä. Skriptille annetaan seuraavat parametrit:
1. Target_ip and target_port. Parametrilla annetaan hyökkäyksen kohteen IP-osoite ja portti CLDAP-palvelimille. Palvelimet lähettävät vastauspaketin tähän osoitteeseen ja porttiin.
2. List of reflectors. Tämä parametri sisältää listan CLDAP-palvelimista joita voidaan käyttää hyökkäykseen.
3. Time limit. Tällä parametrilla voidaan määritellä aikaraja hyökkäyksen kestolle.
Kun skripti suoritetaan yllä olevilla parametreilla, listan palvelimille lähetetään 52 tavua pitkä kysely. Palvelimet vastaavat kyselyyn osoitteeseen, joka on määritelty parametrilla target_ip. Hyökkäyksen kohde joutuu käsittelemään serverien lähettämät CLDAP-vastaukset

joita se ei ole itse pyytänyt vahvistettuna.

(Suk-June Choi 2017)

3.3 TCP SYN Flooding

TCP SYN tulvahyökkäys keksittiin jo vuonna 1994. Menetelmä julkaistiin vuonna 1996 ja siitä lähtien hyökkäyksiä on havaittu verkossa. Hyökkäys perustuu palvelinkoneen resurssien kuluttamiseen, joten se vaatii oikein toteutettuna verrattain vähän kaistaa verkkoyhteydeltä. Normaali TCP-yhteyden avaus aloitetaan lähettämällä asiakaskoneelta SYN palvelinkoneella sijaitsevaan LISTEN-tilassa olevan TCP-porttiin. Palvelin vastaa tähän SYN-ACK-viestillä ja portin tila vaihtuu SYN-RECEIVED tilaan. Asiakaskone taas vastaa tähän ACK jolloin yhteys on muodostettu. Hyökkäyksessä käytetään hyväksi TCP-yhteyden muodostamisen tunnettua heikkoutta. Hyökkäyksessä lähetetään paljon SYN-pyyntöjä, mutta SYN-ACK-vastaukseen ei joko reagoida tai pyynnöt lähetetään väärennetystä IP-osoitteesta jolloin palvelinkoneen SYN-ACK vastaus menee tähän olemattomaan osoitteeseen. Kummassakin tapauksessa kohdekone jää odottamaan ACK-vastausta jokaiseen yhteyden avaukseen kunnes uusia yhteyksiä ei enää voida muodosta.

Palvelimen muistiresurssien suojaamiseksi kaiken yhteyden vaatiman informaation tallentavien tietorakenteiden (TCB, Transmission Control Block) määrää on yleensä rajoitettu käyttöjärjestelmän kernelissä. Järjestelmästä riippuen tämä raja voi olla globaali tai lokaali porttinumerokohtainen. Raja voi koskea täysin muodostettuja yhteyksiä tai vain SYN-RECEIVED-tilassa olevia. Yleensä järjestelmissä on implementoituna parametri listen() järjestelmäkutsuun, joka mahdollistaa sovellukselle arvon ehdottamista rajaksi. Tätä kutsutaan nimellä backlog. Kun backlogin raja saavutetaan, joko saapuvat SYN-viestit hylätään tai keskeneräisiä backlogissa olevia yhteyksiä aletaan korvata.

Hyökkäyksen onnistumisen kannalta on olemassa kolme tärkeää hyökkäysparametria:

IP-osoitteen valinta:

Väärennetyn IP-osoitteen tulee olla valittu niin, että osoite ei vastaa palvelimen lähettämään SYN-ACK viestiin. Jos käytetään osoitetta, joka on jonkun normaalin isäntäkoneen käytös-

sä, tämä vastaa tuntemattomaan SYN-ACK viestiin lähettämällä TCP resetin. Tämä puolestaan vapauttaa palvelinkoneen allokoimat resurssit välittömästi, eikä toivottua vaikutusta saavuteta.

Hyökkäyksen koko:

Hyökkäyksen koon täytyy olla riittävän suuri, jotta saavutetaan backlog. Optimitilanteessa hyökkäys ei ole kuitenkaan backlogia suurempi hyökkääjän verkkoliikenteen minimoimiseksi. Koska backlogin koko vaihtelee eri järjestelmissä hyökkääjä voi räätälöidä hyökkäyksen sopivan kokoiseksi kyseiselle järjestelmälle.

Hyökkäyksen taajuus:

Puoliavoimien yhteyksien elinaikaa on yleensä rajoitettu ja TCP implementaatiot vapauttavat puoliavoimien yhteyksien varaaman muistin tietyn ajan jälkeen. Jotta hyökkäys olisi tehokas hyökkääjän tulee lähettää uusi SYN-tulva kohteeseen juuri kun aikaisempien puoliavoimien yhteyksien varaamaa muistia aletaan vapauttaa. Tämä taajuus täytyy säätää kohteen kohteen ajastimen mukaiseksi. Liian suuri taajuus aiheuttaa liikaa liikennettä ja saattaa helpommin kiinnittää huomion hyökkääjään, liian pieni taajuus puolestaan heikentää hyökkäyksen tehoa ja sallii laillisten yhteyksien muodostamisen hyökkäyksen aikana.

Hyökkäyksen tehoa voidaan lieventää useilla eri menetelmillä. Kuten aiemmin todettiin, väärennetty IP-osoite on oleellinen tekijä onnistuneelle hyökkäykselle. Väärennettyjen osoitteiden suodattaminen saapuvasta liikenteestä on tehokas puolustusmenetelmä tähn kuten moniin muihinkin hyökkäyksiin. Backlogin kasvattaminen on myös eräs menetelmä, mutta sen osalta TCP-implementaatio saattaa skaalautua huonosti josta taas voi aiheutua suorituskykyongelmia. SYN-RECEIVED-ajastimen säätäminen poistaa hyökkäykseen liittyvät puoliavoimet yhteydet nopeammin backlogista, mutta saattaa liian lyhyeksi säädettynä aiheuttaa myös laillisten yhteyksien hylkäämistä. Puoliavoimien yhteyksien kierrättämistä vanhimmasta alkaen backlogin täytyttyä voidaan myös käyttää. Menetelmä toimii mikäli lailliset yhteydet pystytään muodostamaan kokonaan nopeammin kuin backlogin täyttyminen hyökkääjän paketeista kestää. Suurilla pakettitiheyksillä menetelmä ei ole tehokas. SYN Cache-menetelmässä minimoidaan tila jonka SYN allokoi. Menetelmässä tulevasta SYN-viestistä valitaan tietyt salaiset bitit joista muodostetaan tiivistesumma IP-osoitteen ja portin kanssa.

Tiivisteen arvo määrittelee minne keskeneräinen TCP tallennetaan globaalissa tiivistetaulussa. Jokaiselle tiivistearvolle on raja jonka jälkeen vanhin merkintä hylätään.

(“RFC4987” 2007)

3.4 DNS Reflection Attack with Amplification

DNS Reflection Attack with Amplification on epäsymmetrinen hyökkäysmenetelmä, jossa hyökkääjä lähettää DNS kyselyn väärennetyllä kohdeosoitteella. Tällä menetelmällä kohdekoneelle saadaan pienikokoisella kyselyllä vastauksena huomattavasti suurempikokoinen DNS-vastaus. Tavoitteena on saada kohteen verkkoliitännän kaistanleveys kulutettua vahvistusvaikutusta hyväksi käyttäen ja näin estää normaali verkkoliikenne.

(“Guide to DDoS Attacks November 2017” 2020)

3.5 DNS Flood

DNS Flood on symmetrinen palvelunestohyökkäys jolla pyritään kuluttamaan loppuun palvelimen resurssit, kuten muisti tai prosessorikapasiteetti. DNS-palvelimet käyttävät UDP-protokollaa, joten kyseessä on yksi UDP Flood-hyökkäyksen variantti. Hyökkäys voidaan toteuttaa skriptillä joka muodostaa ja lähettää väärin muotoiltuja DNS-kyselyjä palvelimelle. Osa tai kaikki paketin tiedot voivat olla väärennettyjä. Satunnaisen IP-osoitteen käyttäminen vaikeuttaa myös hyökkäyksen torjumista. Paketit näyttävät tulevan useista eri kohteista, joten IP-osoitteen perusteella tapahtuva suodatus ei toimi hyökkäykseltä suojautumiseen.

Toinen versio hyökkäyksestä on DNS NXDOMAIN Flood. Se perustuu suureen määrään DNS-kyselyjä joiden kohteena on olematon tai virheellinen osoite. DNS-palvelin käyttää paljon resursseja yrittäessään suorittaa kyselyn ja sen välimuisti täyttyy näistä pyynnöistä. Lopulta palvelin ei enää pysty vastaamaan oikeisiin DNS-kyselyihin.

(“DNSFlood” 2020) (“RFC3833” 2004)

3.6 Fork Bomb

Fork bomb tunnetaan myös nimellä rabbit virus. Unix ja Linux järjestelmissä on järjestelmäkutsu fork. Sillä muodostetaan olemassa olevasta prosessista uusi prosessi. Nämä prosessit voivat suorittaa omia tehtäviään toisistaan riippumatta. Fork bomb hyödyntää tätä ominaisuutta haaroittamalla prosessia rekursiivisesti kunnes järjestelmän prosessori- ja muistiresurssit on kulutettu loppuun. Tässä vaiheessa järjestelmä ei vastaa näppäimistökomentoihin ja usein ainoa tapa toipua hyökkäyksestä onkin pakottaa uudelleenkäynnistys katkaisemalla palvelimelta sähköt. Mikäli palvelimelle pystytään antamaan komentoja järjestelmän jumiuttaminen on erittäin helppoa. Esimerkiksi komentoriviltä :(){ :|: &};: aiheuttaa nopeasti prosessien määrän kasvamisen niin suureksi, että järjestelmä lakkaa vastaamasta. Myös useilla ohjelmointikielillä kyseessä on hyvin yksinkertainen operaatio, esimerkiksi

Python: #!/usr/bin/env python

import os while True: os.fork()

Ruby:

#!/usr/bin/env ruby loop { fork { bomb } }

C:

#include <unistd.h> int main(void) { for (;;) { fork(); } }

(“Fork bomb attack” 2020)

3.7 Http Flood

Http-tulvahyökkäyksessä hyökkääjä käyttää oikein muotoiltuja HTTP GET tai POST pyyntöjä kuormittaakseen palvelinta. Pyynnöt on muotoiltu niin, että ne kuormittavat mahdollisimman paljon palvelinta. Tässä menetelmässä ei käytetä väärin muotoiltuja paketteja, väärennettyjä osoitteita eikä heijastustekniikoita ja menetelmä tarvitsee kohtuullisen vähän kaistanleveyttä sivun tai palvelimen ylikuormittamiseen. (“Imperva DDoS Attacks” 2019)

3.8 IP Fragmention Attack

IP Fragmention Attack perustuu IP-verkkojen ominaisuuteen, jossa verkon suurinta lähetysyksikköä (Maximum Transmission Unit, MTU) suuremmat datagrammit pilkotaan pienemmiksi paketeiksi jotta lähettäminen on mahdollista. MTU on usein noin 1500 tavua, mutta se voi olla verkon ominaisuuksista riippuen myös suurempi tai pienempi. Vastaanottaja puolestaan kokoaa paketin takaisin alkuperäiseen muotoonsa. Hyökkäyksestä on useita eri variantteja.

(“IP Fragmentation Attack” 2020)

3.9 NTP Amplification

NTP-vahvistushyökkäyksessä hyökkääjä käyttää julkisesti saatavilla olevia aikapalvelimia kuormittaakseen hyökkäyksen kohteena olevaa palvelinta UDP-liikenteellä. Hyökkäyksen vahvistuskerroin voi olla 1:20-1:200 tai jopa enemmän. Tämä mahdollistaa helposti suuren liikennemäärän aiheuttamisen kohdekoneelle.

(“Imperva DDoS Attacks” 2019)

3.10 ICMP Flood, Ping Flood

Kohteena olevalle koneelle lähetetään suuri määrä ICMP Echo Request (ping) paketteja mahdollisimman nopealla tahdilla odottamatta vastausta. Tämä kuluttaa kaistaa kumpaankin suuntaan, sillä yleensä kohdekone pyrkii vastaamaan pingiin ICMP Echo Reply-paketilla.

(“Imperva DDoS Attacks” 2019)

3.11 Microsoft SQL Reflection Attack with Amplification

Hyökkäyksessä hyökkääjä väärentää kohteen IP-osoitteen ja lähettää muotoilemansa pyynnön julkisessa verkossa olevalle MS SQL-palvelimelle käyttäen MC-SQLR-protokollaa (MS-SQL Server Resolution Protocol). Tietokantapalvelimen vastaus sisältää tietoa niin tietokan-

nan käynnissä olevista instansseista kuin niiden kytkeytymisistä toisiinsa. Tietokantapalvelimen konfiguraatiosta ja tietokantojen instansseista riippuen pyynnölle voidaan saada jopa 25-kertainen vahvistus. Lähettämällä MC-SQLR-pyyntöjä suurelle määrälle tietokantapalvelimia ja väärentämällä niihin kohteen IP-osoite saadaan kohteen kaistanleveys käytettyä kokonaan ja estettyä näin oikea liikenne.

Microsoft SQL Server 2008 versiosta lähtien protokolla on ollut oletuksena pois käytöstä. Aiemmissa versioissa palvelu on pitänyt disabloida manuaalisesti. Mikäli palvelu on tarpeen pitää käytössä tulisi ottaa käyttöön suojatoimia joilla voidaan estää palvelimen käyttö hyökkäysvälineenä. Protokollan käyttämä portti UDP 1434 tulee estää palomuurilla ulkoverkosta tai sallia yhteys vain luotetuista IP-osoitteista. Myös käyttäjien autentikointia voidaan käyttää esimerkiksi SSH- tai VPN-protokollilla.

(“Guide to DDoS Attacks November 2017” 2020)

3.12 Ping of Death (PoD)

PoD on menetelmä, jossa hyökkääjä lähettää liian suuren tai tarkoituksellisesti väärin muotoillun ping-paketin tarkoituksena aiheuttaa vastaanottavan koneen jumiutuminen tai kaatuminen. IP-paketin maksimipituus on otsikkotiedot mukaanlukien 65535 tavua. Siirtoyhteyskerros tyypillisesti rajoittaa kehyksen kokoa esimerkiksi ethernet-verkossa 1500 tavuun. Tällöin suuret paketit pilkotaan useammaksi IP-paketiksi, fragmentiksi. Vastaanottaja kokoaa nämä taas yhdeksi kokonaiseksi paketiksi. PoD:n väärin muotoillusta paketista johtuen vastaanottaja saakin paketin kokoamisen jälkeen suuremman kuin 65535 tavun paketin. Tämä voi aiheuttaa puskurin ylivuodon ja jopa kaataa koneen.

(“Imperva DDoS Attacks” 2019)

3.13 Slowloris

Slowloris on sovelluskerroksen tarkasti kohdennettu hyökkäys, jossa tietty palvelin saadaan kuormitettua vaikuttamatta kohdeverkon muihin palvelimiin tai portteihin. Hyökkääjä avaa yhteyksiä ja pitää niitä auki mahdollisimman pitkään lähettämällä vain osittaisia pyyntöjä.

Kohde joutuu pitämään yhteyttä auki odottaessaan loppua pyyntöä jota ei koskaan tule. Tämä ylikuormittaa yhtäaikaisten yhteyksien määrän ja muuta käyttäjät eivät saa palveluun yhteyttä. Slowloris on tehokas lähinnä Apache-palvelimia vastaan johtuen Apachen MPM-modulin suunnitteluvirheestä jonka takia jokainen yhteys saa oman säikeen.

(“Imperva DDoS Attacks” 2019)

3.14 Smurf Attack

Smurf-hyökkäys muistuttaa ping-tulvahyökkäystä, mutta smurf on vahvistushyökkäys. Echo Request lähetetään väärennetyllä IP-osoitteella kaikille verkon koneille. Osoitteena käytetään kohteena olevan palvelimen osoitetta. Vastaukset pingiin ohjautuvat palvelimelle ja riittävän suurella määrällä ICMP-vastauksia palvelin saadaan alas.

(“Imperva DDoS Attacks” 2019)

3.15 SNMP Reflection

SNMP heijastushyökkäyksessä lähetetään suuri määrä SNMP-kyselyjä väärennetyllä IP-osoitteella. Osoitteena käytetään kohdekoneen osoitetta. Vahvistus saadaan aikaan epäsymmetrisellä kysely-vastausparilla, jolloin vastaus on merkittävästi kyselyä suurempi. Vahvistuskerroin voi olla jopa luokkaa 1700.

(“Imperva DDoS Attacks” 2019)

3.16 SSDP Reflection Attack with Amplification

SSDP vahvistettu heijastushyökkäys tapahtuu väärentämällä uhrin IP-osoite hyökkääjän muodostamaan SOAP-pyyntöön. Pyyntö lähetetään Internetissä oleville avoimille UPnP-laitteille. Pyynnön seurauksena laitteet lähettävät vastauksensa kohteena olevan uhrin IP-osoitteeseen. Pyynnön muotoilusta riippuen voidaan saavuttaa jopa 30-kertainen vahvistus. Lähettämällä pyyntö suurelle määrälle avoimia UPnP-laitteita uhrin kaistanleveys saadaan kulutettua loppuun ja estettyä normaali liikenne. Hyökkäys voidaan tunnistaa tarkkailemalla verkon lo-

kitietoja ja etsimällä merkintöjä portista UDP 1900 sisäänpäin tulevasta liikenteestä suuresta määrästä eri IP-osoitteita. Koska kyseessä on volumetrinen hyökkäys sen lieventäminen tai estäminen paikallisesti ilman operaattorin tai palveluntarjoajan apua on vaikeaa. Omien laitteiden käyttö hyökkäyksen apuvälineenä voidaan estää tarkistamalla omasta verkosta löytyvät haavoittuvat laitteet ja estämällä ulospäin lähtevä liikenne portista UDP 1900.

(“Guide to DDoS Attacks November 2017” 2020)

3.17 UDP Flood

UDP tulva on mikä tahansa DDos hyökkäys jossa kohteelle lähetetään suuri määrä UDP-paketteja satunnaisiin portteihin. Kohteena oleva kone joutuu jatkuvasti tarkastamaan onko portissa kuuntelevaa sovellusta, ja jos sovellusta ei löydy se vastaa ICMP-viestillä “Destination Unreachable”. Kumulatiivinen vaikutus kohteelle on se, että järjestelmä hukkuu näihin satunnaisiin paketteihin eikä pysty vastaamaan lailliseen liikenteeseen.

(“What is an UDP Flood Attack?” 2020)

3.18 Wordpress Pingback Reflection Attack with Amplification

Wordpress on suosittu sisällönhallintajärjestelmä jota käytetään web-sivujen ja blogien ylläpitoon. Sitä voidaan myös käyttää palvelunestohyökkäykseen hyödyntämällä pingback-ominaisuutta. Ominaisuus ilmoittaa WordPressin verkkosivuille siitä, että jollekin toiselle palvelun verkkosivulle on lisätty sinne johtava linkki. Oletuksena tämä ominaisuus lataa koko pingback-prosessin laukaisseen linkin sisältävän web-sivun. Hyökkääjä voi valita sopivan määrän WordPress sivustoja ja lähettää niille pingback pyynnön kohdesivuston URL-osoitteella varustettuna. Tämän johdosta jokainen verkkosivusto lähettää kohteelle pyynnön ladata kyseinen verkkosivu. Suuri määrä pyyntöjä ylikuormittaa kohdepalvelimen.

Hyökkäys voidaan tunnistaa tutkimalla verkon lokitietoja ja etsimällä suurta määrää sisään tulevaa liikennettä joka sisältää HTTP GET pyyntöjä satunnaisilla arvoilla. Satunnaisuuden on tarkoitus ohittaa välimuisti pakottamalla koko sivun uudelleenlataus. Sisääntulevaa hyökkäysliikennettä ei pystytä erottamaan normaalista liikenteestä, eikä sitä näin ollen pystytä

estämään. WordPress-sivustojen käyttö hyökkäysvälineenä sen sijaan pystytään estämään. WorsPress tarjoaa tähän työkalun, jolla pingback-ominaisuus voidaan kytkeä pois.

(“Guide to DDoS Attacks November 2017” 2020)

4 Muita aiheeseen liittyviä tutkimuksia

Suodatus- ja suojautumismekanismit voidaan luokitella yhteistoiminnan tai vasteajan perusteella. Yhteistoiminnan perusteella järjestelmät jaetaan yhteistoiminnalliseen suodatukseen (cooperative filtering) jossa laitteet tai solmut kommunikoivat keskenään sääntöihin ja suodatukseen liittyvissä asioissa. Suodatusmekanismi joka ei jaa tietoa muiden koneiden tai solmujen kanssa on yksittäin toimiva suodatus (individual filtering). Vasteajan perusteella menetelmät jaetaan proaktiivisiin, jolloin puolustusmekanismi on aktiivinen ennen hyökkäyksen alkamista ja reaktiivisiin, joissa hyökkäys käynnistää puolustusmekanismit. Kalkan et al. mukaan tehokkain menetelmä on proaktiivinen ja yhteistoiminnallinen mekanismi. Mikäli mekanismi olisi mahdollista ottaa käyttöön läpi verkon DDoS-hyökkäykset voidaan estää mahdollisimman lähellä hyökkäyksen lähdettä ennen kuin hyökkäys ehtii laajeta. Järjestelmästä olisi myös mahdollista tehdä yksittäisiä järjestelmiä tarkempi, koska sillä olisi laajempi näkyvyys ja tieto verkon toiminnasta.

(Kübra Kalkan 2016)

4.1 ScoreForCore

ScoreForCore on proaktiivinen ja yhteistoiminnallinen suodatusmenetelmä. Se suorittaa tilastollista analyysiä liikenteestä ja vertaa tallennettua profiilia verkon liikenteeseen. Näin se pystyy havaitsemaan myös uusia tuntemattomia hyökkäysmenetelmiä. Testeissä menetelmällä saavutettiin jopa 80 prosentin havaitsemistarkkuus ennalta tuntemattomia hyökkäyksiä vastaan ja lähes kaikki tunnetut hyökkäykset pystyttiin rajaamaan lähelle hyökkäyksen lähdettä.

(Kübra Kalkan 2016)

4.2 FireCol

FireCol on proaktiivinen, yhteistoiminnallinen ja skaalautuva ratkaisu DDoS-hyökkäyksien havaitsemiseen varhaisessa vaiheessa. Mekanismissa muodostetaan virtuaalisia suojarenkai-

ta suojattavan kohteen ympärille. Järjestelmän ydin koostuu Internetpalveluntarjoajien tasolla olevista hyökkäyksenestojärjestelmistä. Hyökkäysten torjunta tapahtuu mahdollisimman lähellä alkupistettä säästäen verkon resursseja. Testeissä menetelmä havaittiin suorituskykyiseksi ja toimintavarmaksi. Se myös tarvitsi vähän laskentatehoa eikä aiheuttanut paljoa ylimääräistä liikennettä.

(J. Francois 2012)

4.3 SOS

SOS (Secure Overlay Services) on myös proaktiivinen yhteistoiminnallinen menetelmä. Tässä mallissa käyttäjien paketit autentikoi erityinen Secure Overlay Access Point (SOAP). Varmistettuaan, että paketin lähettäjällä on oikeus käyttää kohteen palveluja ne lähettävät paketit majakkasolmuille (beacon nodes) jotka tuntevat salaisten solmujen, Secure Servletsien paikan ja osoitteen. Secure Servlet välittää paketin kohteeseen. Rakenteesta johtuen menetelmä toimii vain tunnettujen käyttäjien ja ennalta määritettyjen kohteiden välisessä liikenteessä.

(A. D. Keromytis ja Rubenstein 2004)

4.4 CBF

Confidence Based Filtering (CBF) on pilviympäristöihin tarkoitettu yksittäin toimiva proaktiivinen suodatusmekanismi. Siinä kerätään laillisia paketteja ennen hyökkäystä ja muodostetaan niistä attribuuttipareja profiileiksi. Attribuuttien laskemiseen käytetään IP-paketin otsikosta kenttiä kokonaispituus, TTL, protokollatyyppi ja lähde IP. TCP-otsikosta käytetään kenttiä lippu (Flag) ja kohdeportti. Pakettien laillisuus päätetään laskemalla luottamusarvoja attribuuttipareista. Menetelmä on nopea ja vaatii vähän tallennuskapasiteettia, mutta epärelevantit attribuuttiparit vaikuttavat hyökkäyksen tunnistuksen tarkkuuteen.

(Qi Chen ja Yu 2011)

4.5 PFS

PFS (Probabilistic Filter Scheduling) koostuu hyökkäyspolulla olevista suodattavista reitittimistä. Menetelmässä määritellään suodatustekniikan lisäksi myös ajastus ja kuinka löydetään parhaat sijainnit suodattimille. Suodattavat reitittimet käyttävät todennäköisyysmerkintöjä paketeille tunnistaakseen hyökkäyspolut ja levittääkseen suodattimia. Hyökkäyksen kohde kerää nämä merkinnät ja päättelee, mikä reititin on välittämässä eitoivottuja paketteja ja tämän perusteella pyytää tiettyä reititintä aktivoimaan suodatuksen. Suodattimien ajastus päättää mitkä suodattimet asennetaan ja mitkä hylätään perustuen suodattimien käytön aktiivisuuteen. Ajastuksen tarkoitus on pienentää suodatuspyyntöjen tulvaa reitittimille. Testeissä hyökkäyksen kohde pystyi vastaanottamaan jopa 45 prosenttia laillisesta liikenteestä ollessaan volumetrisen hyökkäyksen kohteena.

(Seo, Lee ja Perrig 2011)

4.6 StopIt

StopIt on suodatinperustainen DoS-puolustusjärjestelmä. Järjestelmä koostuu jokaiseen autonomiseen järjestelmään asennetusta StopIt-palvelimesta, reitittimistä ja suojattavista koneista. Hyökkäyksen tapahtuessa kohteenä oleva kone lähettää oman verkkonsa reitittimelle pyynnön estää liikenne hyökkäävn koneen IP-osoitteesta tietyksi ajaksi. Reititin varmistaa pyynnön tarkistamalla että hyökkäys on käynnissä. Se lähettää pyynnön edelleen oman autonomisen järjestelmänsä StopIt-palvelimelle. Palvelin välittää pyynnön hyökkäävän koneen verkon StopIt-palvelimelle. Palvelin puolestaan lähettää pyynnön hyökkäävän koneen lähimmälle reitittimelle joka asentaa suodattimen ja lähettää reititin-isäntä-pyynnön hyökkäävälle koneelle. StopIt-yhteensopivan koneen tulisi asentaa paikallinen suodatin liikenteen estämiseksi. Mikäli se ei niin tee, reititin ryhtyy tarvittaviin toimiin kyseistä konetta vastaan. StopIt ei ole tehokas sellaisia hyökkäyksiä vastaan jotka eivät saavuta kohdekonetta vaan tukkivat verkkoyhteyden. StopIt on myös perussuunnittelultaan haavoittuva IP-osoitteen väärentämiselle, reitittimien tai StopIt-palvelimen resurssit kuluttaville hyökkäyksille ja sitä voidaan myös käyttää hyökkäyksen välineenä estämään laillisten käyttäjien liikenne.

(Xin Liu 2008)

4.7 PacketScore

PacketScore on tilastolliseen menetelmään perustuva suodatusmekanismi jossa jokaisen paketin IP- ja TCP-otsikkotietojen attribuuttien arvot analysoidaan ja näiden perusteella lasketaan pisteytys. Paketti tulkitaan lailliseksi mikäli sen saama arvo on dynaamisen rajan yläpuolella. Dynaaminen raja-arvo määritellään Bayesilaiseen teoreemaan perustuen. Simulaatioissa laillisten ja hyökkäyspakettien pistejakaumista pystyttiin hyvin erottamaan hyökkäävä liikenne. PacketScore ei kuitenkaan pysty erottamaan FlashCrowd-tilannetta hyökkäyksestä.

(Yoohwan Kim ja Chao 2006)

4.8 Hop-Count Filtering

Hop-Count Filtering (HCF) on menetelmä IP-osoitteen väärentämistä käyttäviä DDoS- hyökkäyksiä vastaan. Menetelmässä käytetään IP-paketin hyppyjen määrää joka voidaan päätellä paketin otsikkotietojen TTL-kentästä sekä tietoa IP-osoitteiden sijainnista. Näitä tietoja vertaamalla palvelin voi päätellä onko paketin IP-osoite väärennetty vai ei. Testeissä päästiin lähes 90 prosentin tarkkuuteen väärennetyn osoitteen tunnistamisessa. Hyökkääjä voi kuitenkin käyttää satunnaista alkuarvoa TTL-kentässä, joka heikentää menetelmän tehoa merkittävästi. Myös NAT (Network Address Translation) voi vaikuttaa menetelmän toimivuuteen aiheuttamalla tietylle IP-osoitteelle useita valideja TTL-arvoja.

(Cheng Jin 2003)

4.9 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) ei ole alunperin tarkoitettu DDoS-hyökkäyksiltä suojautumiseen, mutta se antaa siihen uusia tapoja. Ohjelmistopohjainen liikenneanalyysi, keskitetty hallinta, globaali näkymä verkkoon ja edelleelähetyssääntöjen dynaaminen hallinta antavat uusia mahdollisuuksia DDoS-hyökkäyksien havaitsemiseen ja torjumiseen. Toisaalta mahdolliset SDN:n haavoittuvuudet voivat itsessään mahdollistaa hyökkäysten toteuttamisen. SDN on aiheena niin laaja ja kiinnostava, että siinä riittäisi aihetta kokonaiselle lop-

putyölle.Yan et al. esitteleekin työssään taulukossa 1. useita niin lähde-, verkko- kuin kohdeperusteisia SDN:ää käyttäviä mekanismeja hyökkäysten torjuntaan.

(Qiao Yan ja Li 2016)

4.10 Analyzing well-known countermeasures against distributed denial of service attacks

Tutkimuksessaan H. Beitollahi ja G. Deconinck käyvät varsin kattavasti läpi niin hyökkäysten tunnistamista, selviytymistekniikoita, proaktiivisia tekniikoita kuin reaktiivisiakin tekniikoita. Puolustustekniikoita on jaettu alueittain kolmeen eri ryhmään.

Source-end tarkoittaa lähellä hyökkäyksen lähdettä tapahtuvaa puolustamista, käytännössä hyökkääjän Internetpalveluntarjoajan reittimillä. Etuna tässä on haitallisen liikenteen rajoittaminen aikaisessa vaiheessa jolloin se haittaa mahdollisimman vähän laillista liikennettä. Haasteena puolestaan on haitallisen liikenteen tunnistamisen vaikeus ja suuri määrä vääriä positiivisia ja negatiivisia tulkintoja.

Core-end puolestaan tarkoittaa Internetin runkoverkossa tapahtuvaa puolustautumista. Mikä tahansa runkoverkon reititin voi yrittää tunnistaa ja rajoittaa haitallista liikennettä. Edelleen vaikeutena on haitallisen liikenteen tunnistaminen normaalin liikenteen seasta, sekä suuresta liikennemäärästä johtuen ei välttämättä ole mahdollista käyttää reitittimien tehoa tunnistamiseen ja suodattamiseen. Runkoreittimiä voidaan kuitenkin käyttää kokonaisliikeen määrän rajoittamiseen uhrin serverin suuntaan.

Victim-end tarkoittaa hyökkäyksen kohteessa olevaa puolustautumista. Suurien liikennemäärien rajoittaminen ei onnistu, mutta hyökkäysliikenteen tunnistaminen laillisesta liikenteestä onnistuu helpommin.

Myös tämän tutkimuksen mukaan hyökkäyksen tunnistamisen kannalta verkkokerroksen hyökkäykset ovat helpompia havaita. Hyökkäyspaketit sisältävät yleensä satunnaista dataa ja epäkelpoja järjestys- ja kuittausnumeroita. Hyökkäykseen käytettävät zombi/bottikoneet eivät edes yritä muodostaa TCP-yhteyttä vaan pyrkivät vain tukkimaan yhteyden satunnaisilla paketeilla. Serverin on näinollen suhteellisen helppoa erottaa hyökkäyspaketit lailli-

sesta liikenteestä jossa pyritään aina muodostamaan TCP-yhteys ja käytetään systemaattisia järjestysnumeroita ja korrektia hyötykuormaa. Sovelluskerroksen hyökkäyksien tunnistaminen on vaikeampaa, koska hyökkäyksissä matkitaan laillisen käyttäjän toimia. Haastavimmaksi tilanteeksi todetaankin niin sanottu flash crowd, jolla tarkoitetaan nopeaa kasvua laillisten käyttäjien yhteysmäärässä. Hyökkäysten havaitsemismenetelmistä artikkelissä käsitellään jaksottainen muutospisteen havainnointi, signaalin spektrin muutosten havainnointi, neuroverkkojen käyttö ja statistiset tekniikat.

Selviytymistekniikoista käsitellään perinteisiä menetelmiä, joihin kuuluvat resurssien lisääminen palvelimella, proxy-palvelimien lisääminen, backlogin jonon kasvattaminen, yhteyspyyntöjen aikakatkaisun säätäminen lyhyemmäksi sekä näiden yhdistelmät.

Proaktiivisina tekniikoina käsitellään sisääntulevan ja lähtevän liikenteen suodatusta ja sen laajentamista runkoreitittimille reittipohjaisena pakettisuodatuksena, D-WARDiksi kutsuttua palomuuria ja erilaisia autonomisten järjestelmien ja Internetpalveluntarjoajien yhteistoiminnassa toteutettuja menetelmiä. Näissä kaikissa on kuitenkin omat haasteensa, eivätkä ne sovellu erityisen hyvin yksittäisen palvelimen suojaamiseen siihen kohdistetulta hyökkäykseltä.

Reaktiivisina tekniikoina käsitellään PushBack, joka ei toimi hitaiden sovelluskerroksen hyökkäysten kanssa kuten ei myöskään K-MaxMin jos hyökkäys kasvaa hitaasti. Myös TTL-arvojen tarkkailuun perustuva HCF ja samantyyppinen Anti-DDos vaikuttavat tutkimuksen perusteella haittaavan merkittävästi myös laillisten käyttäjien yhteyksiä. Erilaiset Client-puzzle tai CAPTCHA-menetelmät voivat olla itsessään menetelmiä palvelunestohyökkäyksille, eivät suojaa volumetrisilta hyökkäyksiltä ja aiheuttavat lisätyötä laillisille käyttäjille. SYN cookies-menetelmää tutkimuksessa pidettiin lupaavimpana tekniikkana SYN- tulvahyökkäyksiä vastaan.

Johtopäätöksenä tutkimuksessa oli, että sekä proaktiivisissa että reaktiivisissa menetelmissä on paljon heikkouksia ja haasteita. Tästä syystä palvelimet joutuvat käyttämään selviytymistekniikoita ja ne palvelimet joilla selviytymistekniikat eivät toimi joutuvat DDoS-hyökkäysten uhriksi. Työssä myös uskotaan sovelluskerroksen hyökkäysten määrän kasvavan tulevaisuudessa. Tämä vahvistaa käsitystäni siitä, että tutkimusta yksittäisen palvelimen suojaamisme-

netelmistä kannattaa jatkaa.

(Hakem Beitollahi 2012)

4.11 DDoS flooding attack detection scheme based on F-divergence

Tutkimuksessa pyritään löytämään menetelmä erottee toisistaan flash crowd eli äkillinen laillisen liikenteen kasvu ja DDoS-hyökkäys. Tämä on erityisen haastavaa, kuten Yu Chen et al. tutkimuksessaan (Yu Chen 2006) havaitsi. Hyökkääjät voivat piilottaa paikallisia poikkeamia tai lähettää virheellisiä hyökkäyskuvioita tutkimuksessa kehitetylle Change-Aggregation Tree(CAT) palvelimelle ja näin rikkoa koko prosessin. H. Rahmani et al. pyrki tutkimuksessaan kehittämään IP-paketin otsikkotietoihin perustuvan statistisen menetelmän. Menetelmä pohjautuu M. Bassevillen julkaisemaan F-divergenssiin (Basseville 1988), menetelmään jossa mitataan todennäköisyysjakaumien etäisyyksiä. Menetelmän olisi tarkoitus korjata viat entropiaan perustuvissa DDoS-hyökkäysten tunnistusmenetelmissä.

Perinteisissä statistisissa menetelmissä hyökkäyksestä raportoidaan kun tietty ennalta asetettu raja-arvo liikenteen määrässä saavutetaan. Raja-arvoa on kuitenkin mahdotonta määritellä niin, että vääriä positiivisia tai negatiivisia tulkintoja ei tulisi eivätkä menetelmät pysty erottamaan normaalin liikenteen määrän kasvua DDoS-hyökkäyksestä. Tutkimuksessa esitetään vaihtoehdoksi kokonaisvaihteluetäisyyden (Total Variation Distance, TVD) määrittelyä liikennevirroille. Tämä menetelmä vaikuttaa tehokkaammalle kuin entropiaan perustuva menetelmä, etenkin suuremmilla pakettimäärillä.

(Hamza Rahmani 2012)

5 Hyökkäyksien havaitseminen

Hyökkäyksen havaitseminen normaalin liikenteen seasta voi olla hyvin haastavaa. Esimerkiksi linkin julkaiseminen suositulla keskustelupalstalla voi aiheuttaa palvelunestohyökkäyksen kaltaisen tilanteen kyseistä palvelua tarjoavalle palvelimelle. Tässä tilanteessa ei kuitenkaan ole tarkoituksenmukaista estää tarpeettomasti käyttäjien pääsyä palveluun. Yksinkertaisimmillaan palvelunestohyökkäys voidaankin toteuttaa pyytämällä suurta joukkoa ihmisiä vierailemaan sivuilla tiettynä aikana. Tämän tyyppisen "hyökkäyksen" havaitseminen on käytännössä mahdotonta, koska hyökkäystä ei voida tunnistaa minkään tallennetun sormenjäljen tai hyökkäyskuvion perusteella inhimillisestä tekijästä johtuen. Tämän normaalin ihmisten aiheuttaman liikenteen seasta pitäisi pystyä erottamaan tahalliset työkaluilla toteutetut hyökkäykset, jotta niiden vaikutusta pystyttäisi lieventämään. Hyökkäystä voidaan epäillä esimerkiksi havaittaessa liikennemäärän äkillinen kasvu, tietyn tyyppisen liikenteen epänormaalista määrästä tai vertaamalla liikennettä normaalitilanteessa tallennettuun sormenjälkeen ja havaitsemalla poikkeamia.

Jo vuonna 2003 Feinstein et al. luetteli vaatimuksia hyökkäysten havaitsemiseen ja niihin vastaamiseen. Heidän mukaansa yksittäisten hyökkäyspolulla olevien reitittimien pitäisi pystyä automaattisesti tunnistamaan, että verkko on hyökkäyksen kohteena ja säätämään liikennevirtaa vaikutuksen pienentämiseksi. Havaitsemis- ja vastaustekniikoiden pitäisi olla adaptiivisia verkkoympäristöstä riippuen ilman mainittavaa manuaalista säätämistä. Hyökkäyksen tunnistuksen pitäisi luonnollisesti olla mahdollisimman tarkka, jotta puolustusmekanismi ei aiheuttaisi palvelunestovaikutusta laillisille käyttäjille eikä oikeita hyökkäyksiä kuitenkaan jäisi tunnistamatta. Puolustusmenetelmän tulisi sisältää älykäs pakettien hylkäysmekanismi, jotta hyökkäyksen vaikutus verkolle minimoidaan ilman vaikutusta normaaliin verkkoliikenteeseen. Tunnistusmenetelmän tulisi olla tehokas ja havaita laaja valikoima erityyppisiä hyökkäyksiä sekä olla robusti tulevaisuudessa keksittäville tunnistuksen kiertomenetelmille. Nämä vaatimukset pätevät edelleen.

(Laura Feinstein ja Kindred 2003)

Hyökkäyksen tunnistamiseen voidaan käyttää myös tilastollista sovelluksen sormenjälkeä.

Julkaisussa Statistical Application Fingerprinting for DDoS Attack Mitigation M.E. Ahmed et al. pyrkivät tunnistamaan hyökkäysliikenteen luomalla kuljetuskerroksen ominaisuuksiin pohjautuvan uuden rakenteen, sovelluksen sormenjäljen. Tähän sormenjälkeen perustuen ehdotetaan uutta liikenteen luokittelurakennetta, jota puolestaan voidaan laajentaa havaitsemaan palvelunestohyökkäykset normaalin liikenteen seasta.

(Muhammad Ejaz Ahmed 2019)

Perinteisemmistä menetelmistä hyökkäyksen tunnistamiseen voidaan käyttää tunkeutumisen havaitsemiseen tarkoitettuja järjestelmiä (Intrusion Detection System, IDS). Eräs tähän käyttöön suunniteltu järjestelmä on avoimen lähdekoodin Snort. Kyseessä on ennalta määriteltyihin sääntöihin perustuva järjestelmä, joka on saatavilla lähdekoodin lisäksi valmiina pakettina Fedoralle, Centosille, FreeBSD:lle ja Windowsille sekä asennettavissa Linux-jakeluiden pakettienhallinnan avulla. Snortia voidaan käyttää pakettisnifferinä, pakettien lokittamiseen tai täysimittaisena tunkeutumisen estojärjestelmänä. Snortille löytyy paljon valmiita sääntöjä. Sääntöjä on kolmessa kategoriassa: Community ruleset sisältää käyttäjien tekemiä sääntöjä jotka ovat vapaasti kaikkien ladattavissa GPLv2 lisenssin mukaisesti. Registered Ruleset sisältää Talos Security Intelligence and Research Teamin kehittämiä, testaamia ja hyväksymiä sääntöjä. Sääntöjen lataaminen ja käyttö vaatii ilmaisen rekisteröitymisen ja Snort Subscriber Rules lisenssisopimuksen hyväksymisen. Näiden lisäksi on olemassa Subscriber Ruleset, joka vaatii maksullisen tilauksen. Nämä säännöt ovat itseasiassa täysin samoja kuin rekisteröidyn käyttäjän säännöt, mutta ilman maksullista tilausta rekisteröity käyttäjä saa uudet säännöt käyttöönsä 30 päivän viiveellä. Käyttäjä voi luonnollisesti tehdä myös omia sääntöjä tarpeen mukaan.

(“Snort FAQ” 2020)

Kuten kaikki sääntöihin perustuvat tunkeutumisen havaitsemisjärjestelmät, myös Snort tunnistaa vain hyökkäykset jotka täsmäävät johonkin käytössä olevista säännöistä. Snortiin on kyllä lisätty rate filter-toiminnallisuus, jonka on tarkoitus estää DDoS-hyökkäyksiä. Toiminnolla voidaan rajoittaa tietystä osoitteesta tulevia tai osoitteeseen meneviä paketteja aikayksikköä kohden. Tämä ominaisuus ei ole kuitenkaan kovin tehokas varsinkaan mikäli hyök-

kääjä käyttää satunnaisia väärennettyjä IP-osoitteita tai hyökkäys on voimakkaasti hajautettu eri koneille ja voi myös helposti johtaa vääriin positiivisiin tulkintoihin. Suorituskyky riippuu paljon laitteistosta, etenkin muistin määrästä. Testeissä A. Saboor et al. myös huomasi, että Snort ei hyödy useammasta prosessoriytimestä mutta yksittäisen ytimen kellotaajuus vaikuttaa merkittävästi siihen kuinka suuren pakettimäärän Snort pystyy käsittelemään.

(Amtul Saboor 2013)

6 Hyökkäyksiltä suojautuminen

Hyökkäyksiltä suojautuminen voidaan jakaa karkeasti kahteen kategoriaan, palvelimella tai sen lähiverkossa tapahtuvaan suojautumiseen ja pilvipohjaisiin ratkaisuihin.

6.1 Paikallinen suojautuminen

Paikallisesti palvelua voidaan suojata lähinnä hitaammilta hyökkäyksiltä, jotka eivät kuluta kaikkea kaistaa yhteydeltä. Volumetrisilta hajautetuilta hyökkäyksiltä suojautuminen paikallisesti on käytännössä hyvin hankalaa. Mikäli liikenne tulee pääsääntöisesti tietyistä IP-osoitteista, esimerkiksi iptablesia käyttäen voidaan kyseisistä osoitteista saapuvat paketit pudottaa.

(“Pusher. Per-IP rate limiting with iptables.” 2019)

Netscoutin AED (Arbor Edge Defense) on verkon reunalle ennen palomuuria sijoitettava tilattoman pakettien käsittelymoottorin sisältävä laite. Se pystyy tunnistamaan miljoonia IoC:tä ja pienentämään kuormaa seuraavana olevalta tilalliselta palomuurilta. Laite seuraa myös ulos lähtevää liikennettä ja pystyy suodattamaan sitä perustuen esimerkiksi tunnettuihin huonomaineisiin IP-osoitteisiin, domaineihin, URL-osoitteisiin tai vaikka geografiaan. AED pystyy havaitsemaan ja pysäyttämään sovelluskerroksen, TCP state exhaustion ja DDoS hyökkäyksiä jopa 40 Gbps liikennemääriin asti. Sitä suuremmista hyökkäyksistä Cloud signaling-toiminnallisuus automaattisesti uudelleenreitittää liikenteen joko Arborin pilveen tai MSSP:n (Managed Security Service Provider) pilvipohjaiseen keskukseen jossa hyökkäystä pystytään vaimentamaan.

(“Netscout AED datasheet” 2019)

6.2 Pilvipalvelut

Pilvipohjaiset ratkaisut pystyvät suojaamaan myös volumetrisiltä hyökkäyksiltä, jotka normaalisti tukkisivat palvelun tai jopa operaattorin internetyhteyden. Palvelut perustuvat maa-

ilmanlaajuisiin pesukeskuksista koostuviin verkkoihin, jotka pystyvät tyypillisesti käsittelemään useiden Tbps liikenteen. Pilvipohjaisten suojamenetelmien käyttötapoja on useita. Liikenne voidaan ohjata kulkemaan jatkuvasti pilvipalvelun kautta, jolloin on mahdollista myös piilottaa varsinaisen serverin osoite hyökkääjiltä. Toinen vaihtoehto on tunnistaa hyökkäys ja ohjata liikenne siinä vaiheessa BGP-reitityksellä pilveen josta puhdistettu liikenne ohjataan GRE-tunnelia pitkin alkuperäiselle palvelimelle. Suojattavan palvelimen lähiverkossa voi olla myös palveluntarjoajan laite joka suodattaa liikennemäärältään pienempiä hyökkäyksiä ja suuren volumetrisen hyökkäyksen sattuessa signaloi automaattisesti pilvipalvelua ohjaamaan liikenteen pilveen.

(“Imperva DDoS Protection” 2019)

6.3 Reverse proxy

Reverse proxy on tekniikka, jota käytetään kuorman tasaukseen, hyökkäyksiltä suojautumiseen, maailmanlaajuiseen kuorman tasaukseen, välimuistina ja SSL-salaukseen. Reverse proxy on sijoitettu palvelimien etupuolelle. Asiakaskoneet ottavat yhteyden reverse proxyyn joka katkaisee suoran yhteyden palvelimelle. Proxy välittää pyynnön palvelimelle, vastaanottaa vastauksen palvelimelta ja välittää sen alkuperäiselle koneelle. Asiakaskoneet eivät siis koskaan pääse kommunikoimaan suoraan palvelimien kanssa.

(“What Is A Reverse Proxy? | Proxy Servers Explained” 2020)

6.4 Verisign/Neustar

Verisign on siirtänyt turvapalveluiden asiakkaat Neustarille. Neustarilla on pitkä historiayrityspuolen asiakkaiden turvapalveluiden tuottamisessa, mukaanluettuna DNS ja DDoS-suojaukset. Verisign puolestaan jatkaa Internetin juuripalvelinten hallintaa sekä tiettyjen ylätason verkkotunnusten hallintaa. Neustar SiteProtect NG on suurin pelkästään dataliikenteen pesemiseen tarkoitettu verkko maailmassa, mahdollistaen jopa yli 11,8 Tbps DDoS-hyökkäyksen vaimentamisen. Pilvipohjaisen DDoS-suojauksen käyttöönotto voi tapahtua kahdella eri tavalla: Hyökkayksen tapahtuessa tai epäiltäessä sellaisen olevan valmisteilla tehdään yksin-

kertaisesti DNS-uudelleen ohjaus SiteProject NG pilveen vaihtamalla uhatun koneen tai palvelun osoite DNS-tietueessa Neustarin ilmoittamaan osoitteeseen. Liikenne alkaa ohjautua pilvipalveluun jossa haitallinen liikenne suodatetaan pois ja turvallinen liikenne päästetään ohjautumaan alkuperäiselle kohteelle. Suodatus ja vaimennus tapahtuu kustomoituna ammattilaisten toimesta, jotta riittävä vaimennus saadaan aikaan. Menetelmä on asiakkaalle yksinkertainen eikä vaadi muita erityisjärjestelyjä. Menetelmä soveltuu parhaiten pienehkölle määrälle isäntäkoneita tai websivuja.

Monimutkaisemmissa ympäristöissä voidaan käyttää BGP-uudelleenohjausta (Border Gateway Protocol).

Neustar WAF+ on pilvipohjainen, aina päällä oleva Web-sovelluspalomuuri sovelluskerroksen hyökkäysten vaimentamiseen.

(“Security Solutions” 2020)

6.5 Microsoft Azure

Azuren DDoS-suojaus tarjoaa kaksi palvelutasoa, Basic ja Standard. Näistä Basic on automaattisesti käytössä Azuressa. Se sisältää jatkuvasti käytössä olevan liikenteen monitoroinnin ja reaaliaikaisen vaimennuksen yleisimmille verkkokerroksen hyökkäyksille. Koko Azuren maailmanlaajuinen verkko on käytettävissä hyökkäysliikenteen hajauttamiseen ja vaimentamiseen. Suojaus on tarjolla julkisiin IPv4 ja IPv6 osoitteisiin.

Standard-palvelu tarjoaa lisäominaisuuksia lisähinnasta. Suojauskäytäntöjä säädetään dedikoidun liikenteenseurannan avulla käyttäen koneoppimisalgoritmeja. Suojauskäytännöt otetaan käyttöön virtuaalisiin verkkoihin , kuten Azure Load Balancer, Azure Application Gateway ja Azure Service Fabric asennettujen resurssien julkisiin IP-osoitteisiin.Tosiaikainen telemetria on käytettävissä hyökkäyksen aikana ja myös historiatiedoille. Sovelluskerroksen suojaus voidaan ottaa käyttöön esimerkiksi Azure Application Gateway Web Application Firewall palvelulla tai asentamalla kolmannen osapuolen palomuurisovellus Azuren kaupasta.

(“Azure DDoS Protection” 2019)

Feature	DDoS Protection Basic	DDoS Protection Standard
Active traffic monitoring and always on detection	Yes	Yes
Automatic attack mitigations	Yes	Yes
Availability guarantee	Azure Region	Application
Mitigation policies	Tuned for Azure traffic region volume	Tuned for application traffic volume
Metrics and alerts	No	Real time attack metrics and diagnostic logs via Azure monitor
Mitigation reports	No	Post attack mitigation reports
Mitigation flow logs	No	NRT log stream for SIEM integration
Migration policy customizations	No	Engage DDoS Experts
Support	Best effort	Access to DDoS Experts during an active attack
SLA	Azure Region	Application guarantee and cost protection
Pricing	Free	Monthly and usage based

Taulukko 3. Azuren DDoS suojaustasot

6.6 AWS Shield

AWS Shield on Amazonin tarjoama palvelu DDoS-suojaukseen. Palvelu seuraa jatkuvasti liikennettä ja havaitessaan palvelunestohyökkäyksen aloittaa hyökkäyksen lieventämistoimet automaattisesti. Tämä auttaa minimoimaan palvelun alhaalla olon ja viiveet. Palvelussa on kaksi tasoa, Standard ja Advanced. Standard palvelutaso suojaa yleisimpiä verkko- ja kuljetuskerroksen hyökkäyksiä vastaan. AWS Shield Advanced tarjoaa lisäsuojaa suuria ja hienostuneempia hyökkäyksiä vastaan. Palvelu vaatii käytännössä Amazonin pilvipalveluiden käyttöä, esimerkiksi Amazon Elastic Compute Cloud (EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator tai Amazon Route 53 palveluja. Hienostuneempien sovelluskerrosten hyökkäysten lieventämiseen tarvitaan käyttöön AWS WAF (Web Application Firewall).

(“AWS Shield” 2020)

6.7 Project Shield

Project Shield on Googlen tarjoama ilmainen suojapalvelu DDoS-hyökkäyksiä vastaan. Palvelu on tarjolla vain tarkkaan kohdennetulle ryhmälle, johon kuuluvat uutispalvelut ja lehtimiehet, ihmisoikeuksiin liittyvät organisaation, vaaleihin liittyvät tiedot ja tulokset sekä poliittiset organisaatiot. Kaupallisille toimijoille palvelua ei tarjota. Palvelu otetaan käyttöön yksinkertaisesti muuttamalla DNS-asetuksia. Project Shield suojaa sekä kerrosten 3/4 että kerroksen 7 hyökkäyksiä vastaan ja plvelua voidaan käyttää myös välimuistina alkuperäisen sivuston liikenteen pienentämiseksi. Monipuolisempaan puolustukseen palvelu tarjoaa kustomoitavia puolustusmenetelmiä. Tarjolla on myös reaaliaikaista analytiikkaa liikenteestä ja virheistä.

(“Project Shield” 2020)

6.8 Cloudflare

Cloudflaren verkko kattaa 200 kaupunkia yli 90 eri maassa. Tarjolla on neljä eri kokoisile organisaatioille tarkoitettua palvelupakettia, Free, Pro, Business ja Enterprise.

Yksityishenkilöille ja ei businesskriittisiin projekteihin on tarjolla nimensä mukaisesti ilmainen vaihtoehto Free. Se sisältää DDoS hyökkäyksien vaimennuksen ja tukipalvelun sähköpostin välityksellä. Ympäri maailmaa sijoitetut data centerit tuottavat samalla sijaintiin perustuvan pääsyn web-sivuille pienentäen latenssia ja parantaen suorituskykyä staattisen sisällön osalta. Välimuisti voidaan myös tyhjentää kokonaan tai vain yksittäisten tiedostojen osalta. Välimuistille voidaan myös asettaa TTL (Time To Live), joka ilmaisessa versiossa voi olla minimissään kaksi tuntia. Asiakaspuolen datan lataus palvelimen suuntaan on myös rajoitettu 100 megatavuun.

Seuraava taso Pro on tarkoitettu ammattimaisien web-sivujen tai blogien suojaamiseen. Lisänä edelliseen on lyhyempi välimuistin TTL-aika, konfiguroitavissa minimissään yhteen tuntiin, automaattinen TCP-asetusten valinta web-sivujen nopeuttamiseksi, häviöttömän kuvien optimoinnin sekä Captcha/JS Challenge bottien ja validien selaimien tarkastamiseen. Palvelun kuukausihinta on 20 USD.

Business tarjoaa lisäksi 24/7/365 tuen chat-palvelun välityksellä, 100% uptime SLA (Service Level Agreement), välimuistin ohituksen kekseille, mahdollisuuden omien SSL-sertifikaattien käyttöön, Regex palomuurisäännöt ja muokattavat WAF säännöt (rajoitettu 25 kappaleeseen). Välimuistin TTL on konfiguroitavissa 30 minuuttiin ja latauskoko palvelimelle on 200 megatavua. Palvelun hinta on 200 USD kuukaudessa.

Enterprise sisältää kehittyneemmän DDoS-hyökkäyksien vaimennuksen joka toimii kerrosten 3, 4 ja 7 hyökkäyksiä vastaan ja sisältää priorisoidun IP-osoitealueet ja reitityksen. Välimuistin TTL voidaan asettaa minimissään yhteen sekuntiin ja yli 500 megatavun lataukset sallitaan palvelimelle. Välimuistissa on mahdollista käyttää Cache-Tag otsikkotietoa joka tallennetaan välimuistissa olevan objektin metadataan. Sen avulla maailmanlaajuiset välimuistin tyhjennykset valittujen objektien osalta onnistuvat sekunneissa. Myös palvelinkohtainen alidomainin välimuistin tyhjennys on mahdollista. Palomuurisääntöjä ja user agentin estosääntöjä voi molempia olla käytössä 1000 kappaletta. WAF sääntöjen määrää ei ole rajoitettu.

(“Cloudflare plans and features” 2020)

6.9 Netscout

Netscout tarjoaa sekä on-premises että pilvipohjaista suojausta hyökkäyksiä vastaan. Myös näiden yhdistelmä on mahdollinen.

Arbor Cloud koostuu 14:stä eri puolilla maailmaa sijaitsevasta pesukeskuksesta (scrubbing center) jotka pystyvät käsittelemään yli 11 Tbps liikenteen. Pelkässä pilvipohjaisessa ratkaisussa haitallinen liikenne tunnistetaan pilvpohjaisesti ja liikenne uudelleenreititetään Arborin pilven pesukeskuksiin kun hyökkäys on tunnistettu.

Hybridiratkaisussa suojattavan verkon reunalla sijaitsee Arborin AED/APS on always-on-tyyppinen suojaus joka pystyy havaitsemaan ja vaimentamaan kaikentyyppisiä hyökkäyksiä. Tehokkaasti se pystyy vaimentamaan lähinnä sovelluskerroksen hyökkäyksiä. Suurissa volumetrisissä hyökkäyksissä AED/APS signaloi Arbor Cloudia uudelleenreitittämään liikenteen pilven pesukeskuksiin haitallisen liikenteen suodattamiseksi.

Arbor Cloudin operaatiokeskus sijaitsee Sterlingissä, Virginiassa. Pohjois-Amerikan pesukeskukset sijaitsevat New Yorkissa, Ashburnissa, San Josessa, Los Angelesissa ja Dallasissa. Euroopan keskukset puolestaan sijaitsevat Amsterdamissa, Frankfurtissa, Marseillessa, Lontoossa ja Tukholmassa. Aasiassa on kolme keskusta, Sydney, Tokio ja Singapore ja Etelä-Amerikassa yksi Sao Paulossa.

Kaikkien palveluiden lisensointi tapahtuu läpi menevän puhtaan liikenteen määrän perusteella. Lisenssivaihtoehtoja on 100 tai 500 Mbps sekä 1-6 Gbps.

(“Arbor Cloud DDoS Protection Services” 2020)

6.10 Imperva

Imperva tarjoaa DDoS-suojausta web-sivustoille, verkoille, yksittäisille IP-osoitteille ja DNS-palvelimille.

Verkon suojauksessa organisaation koko verkko aliverkkoineen suojataan kerrosten 3 ja 4 DDoS-hyökkäyksiltä. Verkko voi sisältää mitä tahansa IP-pohjaisia sovelluksia, esimerkiksi DNS-palvelimia, sähköpostipalvelimia tai web-palvelimia. Palvelu voidaan ottaa käyttöön

jatkuvasti päällä olevana tai pyynnöstä aktivoitavana palveluna. Palveluun voidaan lisätä Cloud Application Security-palveluja suojauksen laajentamiseksi. Suojattavan verkon reunareititin käyttää BGP-protokollaa ohjatakseen saapuvan liikenteen oman verkon sijaan Impervan pilveen. Varsinainen suojaus toteutetaan Impervan DDoS-pesusovelluksella nimeltään Behemoth, joka suorittaa kerrosten 3 ja 4 hyökkäysten suodatuksen. Tämän jälkeen puhdas liikenne ohjataan suojattavaan verkkoon GRE-tunnelia pitkin. Tunnelin käyttö on välttämätöntä, koska BGP-reititys ohjaa suojattavan verkon IP-osoitteisiin kohdistuvan liikenteen Impervan pilveen eikä IP-osoitteella reititetty liikenne näin koskaan saavuttaisi suojattavaa verkkoa ilman tunnelin käyttöä. Suojauksessa käytetään epäsymmetristä reititystä, jossa sisääntuleva liikenne ohjautuu Impervan pilven kautta ja ulospäin lähtevä liikenne reititetään organisaation ISP:n kautta.

(“Introduction: DDoS Protection for Networks” 2020)

7 Suojamenetelmien simulointi

Pilvipalveluiden osalta suojamenetelmien simulointi vaatisi kohtuuttomasti resursseja, joten näiltä osin työ tehdään kirjallisuuskatsauksena.

Palvelimella tapahtuvia suojamenetelmiä simuloidaan niin fyysisillä laitteilla kuin myös virtuaalikoneista koostuvassa verkossa. Työkaluina käytetään mm. Kali Linuxia sekä Python skriptejä etenkin hitaiden sovelluskerroksen hyökkäysten toteuttamiseen. Web-palvelimen suorituskyvyn ja hyökkäysten vaikutusten arviointiin käytetään ApacheBench työkalua. Työkalu ei sisälly Ubuntun työpöytäversion oletusasennukseen, mutta sen asentamiseen riittää apache2-utils-paketin asentaminen. Web-palvelimena käytetään Ubuntu serverille asennettua Apachea.

7.1 iptables

IPtables on Linuxin komentorivityökalu, jolla voidaan konfiguroida kerneliin sisältyvän Netfilter-pakettisuodattimen paketinsuodatussääntöjä. Sääntöihin (rule) määritellään ehdot, jotka määrittävät mihin paketteihin kyseisiä sääääntöjä sovelletaan. Säännöt tallennetaan ketjuihin (chain). Kun ketjuun tulee käsiteltäväksi paketti ketjun sääntöjä käydään järjestyksessä läpi yksi kerrallaan, kunnes löytyy sääntö joka täsmää pakettiin. Ketjut puolestaan tallennetaan tauluihin (table). Tällä hetkellä on olemassa kolme toisistaan riippumatonta taulua jotka ovat filter, nat ja mangle. Oletuksena on filter-taulu joka sisältää ketjut INPUT, FORWARD ja OUTPUT. Nat-taulu sisältää ketjut PREROUTING, OUTPUT ja POSTROUTING. Mangle puolestaan sisälsi kerneliin 2.4.17 asti ketjut PREROUTING ja OUTPUT sekä kernelistä 2.4.18 alkaen lisäksi ketjut INPUT, FORWARD ja POSTROUTING.

(“iptables(8) - Linux man page” 2020)

Liitteessä A on JavaPipe LLC:n suosittelemat IPtablesin asetukset DDoS-hyökkäysten estämiseksi. Liitteessä B on saman yrityksen asetukset Linuxin kernelille hyökkäysten tehon minimoimiseksi.

(“DDoS Protection With IPtables: The Ultimate Guide” 2020)

7.2 Snort

Snort on avoimen lähdekoodin tunkeutumisen havaitsemisjärjestelmä, jota voidaan inline-tilassa käyttää myös tunkeutumisen estojärjestelmänä. Tällöin Snort on käytännössä asennettava omana koneenaan jonka läpi suojattavan kohteen liikenne kulkee. Snortille on olemassa paljon valmiita sääntöjä ja myös omien sääntöjen kirjoittaminen on helppoa. Snort on myös A. Saboor et al. mukaan saavuttanut de-facto standardin aseman NIDS-järjestelmien joukossa joustavuuden, tuen ja kustannustehokkuuden ansiosta. Tämän takia Snort oli luonnollinen valinta yhdeksi testattavaksi järjestelmäksi.

(Amtul Saboor 2013)

7.3 Testiympäristö

Virtuaalisena testiympäristönä toimi Oracle VM Virtualbox Manager (“VirtualBox” 2020) johon oli asennettuna Kali-Linux-2019.1 (“KaliLinux” 2020), Ubuntu Server 18.04.3 LTS (“Ubuntu Server” 2020) jossa web-serverinä Apache 2.4.29 (“Apache” 2020) sekä Ubuntun Desktop 18.04.2 LTS (“Ubuntu Desktop” 2020). Hyökkäyksen vaikutusta testattiin ApacheBench-työkalulla, joka on suunniteltu Apache web-serverin suorituskyvyn mittaamiseen. Toisena testityökaluna käytettiin sitespeed.io ohjelmaa, joka toimii hieman eri tavalla käyttäen testaamiseen oikeita selainmoottoreita. (“sitespeed.io” 2020)

Apachen käyttöön testattavana web-palvelinohjelmistona vaikutti sen suosio. W3Techsin statistiikan mukaan (“Usage statistics of web servers” 2020) Apachea käyttää 38,6 % web-palvelimista, toisena tulevan Nginxin osuus on 32%. Lisäksi se on erityisen haavoittuva esimerkiksi Slowloris-hyökkäykselle ja näin hyökkäyksen ja suojaustoimien vaikutus on helposti havaittavissa.

7.4 Testi iptables

Normaalissa käyttötilanteessa ilman hyökkäystä testi ajettuna seuraavilla parametreilla:

ab -n 1000000 -c 100 -s 220 -r http://192.168.0.1/mlinvoice

antoi seuraavat tulokset:

Concurrency Level: 100

Time taken for tests: 335.492 seconds

Complete requests: 1000000

Failed requests: 0

Non-2xx responses: 1000000

Total transferred: 543000000 bytes

HTML transferred: 314000000 bytes

Requests per second: 2980.70 [#/sec] (mean)

Time per request: 33.549 [ms] (mean)

Time per request: 0.335 [ms] (mean, across all concurrent requests)

Transfer rate: 1580.58 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	1 3.8	0	101
Processing:	0	32 20.8	30	2432
Waiting:	0	27 13.5	29	2307
Total:	0	33 20.6	30	2432

Percentage of the requests served within a certain time (ms)

50% 30

66% 32

75% 34

80% 39

90% 51

95% 62

98% 73

99% 84

100% 2432 (longest request)

Testin mukaan lähes kaikkiin pyyntöihin (99%) vastattiin siis 84 millisekunnissa tai nopeammin ja 50 % vastauksista saatiin 30 millisekunnissa tai nopeammin.

Slowloris-hyökkäys yhdeltä koneelta suoritettuna parametreillä 1000 sokettia ja headerien välinen sleep time 5 muutti tilannetta mutta silti suurimpaan osaan pyynnöistä vastattiin kohtuullisessa ajassa:

Concurrency Level: 100
Time taken for tests: 4106.855 seconds
Complete requests: 1000000
Failed requests: 48
(Connect: 0, Receive: 16, Length: 16, Exceptions: 16)
Non-2xx responses: 999984
Total transferred: 542991312 bytes
HTML transferred: 313994976 bytes
Requests per second: 243.50 [#/sec] (mean)
Time per request: 410.686 [ms] (mean)
Time per request: 4.107 [ms] (mean, across all concurrent requests)
Transfer rate: 129.12 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	4 267.6	0	65602
Processing:	6	407 1604.9	32	131208
Waiting:	0	404 1528.8	32	52960
Total:	15	411 1637.4	33	131208

Percentage of the requests served within a certain time (ms)
50% 33
66% 36

75% 39
80% 41
90% 52
95% 3423
98% 7250
99% 7885
100% 131208 (longest request)

Ensimmäisten tuloksien perusteella virtuaalikoneilla tehty hyökkäystesti ei osoittautunut ainakaan ApacheBenchillä testattuna kovin menestyksekkääksi. Testin mukaan ainoastaan 5% vastauksista kesti yli 3,4 sekuntia. Tämä vaikuttaisi ristiriitaiselta verrattuna käytännössä selaimella testattuun toimintaan. Esimerkiksi testipalvelimelle asennettu MLInvoice laskutusohjelma oli kuitenkin lähes käyttökelvoton hyökkäyksen ollessa käynnissä.

Edellisen testin tuloksista johtuen testi suoritettiin myös toisella tavalla. Sitespeed.io on avoimen lähdekoodin työkalu jolla voi tarkkailla ja testata websivun suorituskykyä. Tämä testiohjelma avaa verkkosivun käyttäen oikeaa selainta. Testissä käytettäväksi voidaan konfiguroida mikä tahansa yleisesti käytössä olevista selaimista ja näin nähdä myös mahdolliset selaimen aiheuttamat erot sivun latausnopeudessa. Tämä testimenetelmä antoikin aivan erilaisia tuloksia niin ilman hyökkäystä kuin meneillään olevan hyökkäyksen kanssa suoritettuna. Testi suoritettuna komennolla
sitespeed.io http://192.168.0.1/mlinvoice -b chrome -n 20
antoi seuraavat tulokset:
backEndTime: 2.21s ($\pm$11.61*ms*)
firstPaint: 4.21s ($\pm$74.08*ms*)
DOMContentLoaded: 3.82s ($\pm$67.45*ms*)
Load: 3.95s ($\pm$68.70*ms*)
rumSpeedIndex: 4211 ($\pm$74.08)

Slowloris-hyökkäyksen ollessa käynnissä tulos muuttui merkittävästi, aivan kuten manuaalisesti selaimella testattaessa havaittiin:

backEndTime: 19.85s ($\pm$3.44s)

firstPaint: 30.77s ($\pm$4.01s)

DOMContentLoaded: 30.33s ($\pm$4.02s)

Load: 30.47s ($\pm$4.02s)

rumSpeedIndex: 30775 ($\pm$4008.35)

Tulosten erotessa merkittävästi ApacheBenchin vastaavista testi toistettiin lisäämällä MlInvoicen kirjautumissivu URLiin. Testi normaalitilanteessa komennolla:
ab -n 1000 -c 100 -s 220 -r http://192.168.0.1/mlinvoice/login.php

Concurrency Level: 100

Time taken for tests: 23.229 seconds

Complete requests: 1000

Failed requests: 0

Total transferred: 7480000 bytes

HTML transferred: 7015000 bytes

Requests per second: 43.05 [#/sec] (mean)

Time per request: 2322.928 [ms] (mean)

Time per request: 23.229 [ms] (mean, across all concurrent requests)

Transfer rate: 314.46 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	2 4.6	1	23
Processing:	2007	2263 134.6	2288	2671
Waiting:	2007	2247 128.0	2264	2670
Total:	2007	2266 136.3	2290	2690

Percentage of the requests served within a certain time (ms)

50% 2290

66% 2326
75% 2357
80% 2375
90% 2427
95% 2466
98% 2495
99% 2535
100% 2690 (longest request)

Tulokset eroavat merkittävästi ensimmäisestä testikerrasta. Tulos on varsin lähellä sitespeed.io:n backEndTimea

ApacheBenchmark samoilla parametreillä ajettuna hyökkäyksen aikana:

Concurrency Level: 100
Time taken for tests: 280.103 seconds
Complete requests: 1000
Failed requests: 117
(Connect: 0, Receive: 39, Length: 39, Exceptions: 39)
Total transferred: 7188280 bytes
HTML transferred: 6741415 bytes
Requests per second: 3.57 [#/sec] (mean)
Time per request: 28010.347 [ms] (mean)
Time per request: 280.103 [ms] (mean, across all concurrent requests)
Transfer rate: 25.06 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	1542 7198.2	1	65301
Processing:	2929	25319 24386.0	20752	131052
Waiting:	0	22205 14566.7	20593	62596
Total:	2930	26861 25156.5	21655	131052

Percentage of the requests served within a certain time (ms)

50% 21655

66% 23586

75% 24267

80% 25290

90% 58926

95% 73967

98% 130919

99% 130951

100% 131052 (longest request)

Myös hyökkäyksen aikana suoritetussa testissä lähes 70 % ApacheBenchmarkin avauspyynnöistä osuu samaan aikaikkunaan sitespeed.io:n backEndTimen kanssa. Testien konfiguraatioerot huomioonottaen tulokset näyttävät järkeville ja melko vertailukelpoisille. Huomionarvoista on, että ApacheBenchmark ei ilmeisesti anna luotettavaa tulosta mikäli testikomennossa annetaan pelkästään webserverillä oleva hakemisto. Sitespeed.io:n testaukseen käyttämä selain osaa kyseisessä tilanteessa uudelleenohjata lataamaan halutun sivun.

Nyt tuloksista voidaan havaita, että pisimmät latausajat venyivät yli 130 sekuntiin ja puoleen sivun latauspyynnöistä pystyttiin vastaamaan aikaisintaan 21,7 sekunnin kuluttua. Näin pitkä viive tekee sivustosta käytännössä käyttökelvottoman.

Slowloris-hyökkäys perustuu suureen määrään yhtäaikaisia yhteyksiä joita pidetään yllä jäljittelemällä erittäin hidasta asiakasyhteyttä ja hyökkäyksen toteutus on helppoa yksittäisellä koneella pienen kaistanleveysvaatimuksen takia. Looginen valinta hyökkäyksen vaimentamisen tutkimiseen oli näin ollen rajoittaa yhtäaikaisia samasta osoitteesta saapuvia yhteyk-

siä. Testaus aloitettiin yksinkertaisella palomuurisäännöllä:
/sbin/iptables -A INPUT -p tcp -m connlimit –connlimit-above 111 -j REJECT –reject-with tcp-reset (“DDoS Protection With IPtables: The Ultimate Guide” 2020)
Sääntö rajaa saapuvat TCP-yhteydet maksimissaan 111 yhteyteen ja hylkää yli menevät TCP-RESETillä. Raja on asetettu näin suureksi jotta testissä käytettävää sataa yhtäaikaista yhteyttä ei rajoiteta säännöllä. Tulokset vaikuttivat lupaavilta. Tulokset hyökkäyksen aikana ApacheBencmarkilla samoilla parametreillä ajettuna:

Concurrency Level: 100
Time taken for tests: 43.307 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 7480000 bytes
HTML transferred: 7015000 bytes
Requests per second: 23.09 [#/sec] (mean)
Time per request: 4330.722 [ms] (mean)
Time per request: 43.307 [ms] (mean, across all concurrent requests)
Transfer rate: 168.67 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	3 8.1	0	38
Processing:	2007	4131 1383.6	4076	9161
Waiting:	2007	4131 1383.9	4076	9161
Total:	2008	4134 1389.5	4077	9187

Percentage of the requests served within a certain time (ms)
50% 4077
66% 4158
75% 4326
80% 4382

90% 5208
95% 7165
98% 8293
99% 9117
100% 9187 (longest request)

80% pyynnöistä pystyttiin nyt myös hyökkäyksen aikana käsittelemään alle 4,4 sekunnissa, joten hyökkäyksen vaikutus sivupyyntöihin oli vain kaksi sekuntia. Yhden prosentin osalta vaikutus oli noin 6 sekuntia. Tämä oli kuitenkin merkittävä parannus ilman palomuurisääntöä suoritettuun testiin, jossa hyökkäyksen vaikutukset latausaikoihin olivat 23-128 sekuntia.

Testi toistettiin käyttämällä kolmea hyökkäävää konetta. Näinkin pieni hajautettu hyökkäys vaikutti erittäin tehokkaasti. Edellisessä testissä hyviä tuloksia tuottanut palomuurisääntö ei enää pystynytkään lieventämään hyökkäyksen vaikutuksia:

Concurrency Level: 100
Time taken for tests: 321.912 seconds
Complete requests: 1000
Failed requests: 423 (Connect: 0, Receive: 141, Length: 141, Exceptions: 141)
Total transferred: 6440280 bytes
HTML transferred: 6039915 bytes
Requests per second: 3.11 [#/sec] (mean)
Time per request: 32191.204 [ms] (mean)
Time per request: 321.912 [ms] (mean, across all concurrent requests)
Transfer rate: 19.54 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	630 2659.2	0	31528
Processing:	7970	28648 32219.7	16384	129435
Waiting:	0	13749 8072.1	14375	73438
Total:	7970	29278 32241.8	17039	129435

Percentage of the requests served within a certain time (ms)

50% 17039

66% 19088

75% 20415

80% 22797

90% 106870

95% 108541

98% 108544

99% 108545

100% 129435 (longest request)

Rajoittamalla palomuurisäännöllä samasta IP-osoitteesta tulevien yhteyksien määrää vielä pienemmäksi tilanne paranisi. Yhteyksien määrää ei kuitenkaan voi rajata kovin pieneksi, koska ei ole tarkoituksenmukaista rajoittaa esimerkiksi yritysten verkoista saman julkisen IP-osoitteen takaa tulevia yhteyksiä liikaa. Lisäksi jo pienikin bottiverkko pystyisi tuottamaan yhteyksiä sadoista eri osoitteista. Tämä menetelmä siis olisi käyttökelpoinen DoS-hyökkäykselle mutta ei DDoS-hyökkäykselle.

7.5 Testi Snort

Ensimmäiset testit Snortilla suoritettiin IDS-tilassa. Paketin malware-tools sisältämän säännön Sid: 1-28532 kuvauksessa mainitaan testissä käytetty Slowloris, mutta kyseisellä säännöllä ei kuitenkaan havaittu hyökkäystä. Tutkimalla Slowloris skriptiä havaittiin sen muodostavan yhteyden ylläpitämiseen tarvittavan keep-alive-viestin niin, että viestin alussa on aina merkit “X-a: “ ja sen jälkeen satunnaisluku väliltä 1-5000. Tämä varmistettiin tutki-

malla skriptin muodostamaa liikennettä Wiresharkilla. Hyökkäävän koneen ja palvelimen välistä liikennettä tallentamalla havaittiin, että viestin hyötykuorma alkoi tavuilla 58 2d 61 3a 20 jotka vastaavat ascii-merkkejä X-a: . Loppuosa koostui satunnaisesta luvusta. Luvun pituudesta riippuen viestin hyötykuorman pituus oli 10 tai 11 tavua. Nämä havainnot olivat yhtäpitäviä hyökkäyskoodia tutkimalla tehtyjen havaintojen kanssa. Tämän perusteella muodostettiin seuraava sääntö:
alert tcp any any -> any $HTTP_PORTS (msg:"Slowloris"; content:"X-a "; dsize:<12; detection_filter:track by_dst, count 5, seconds 60; clastype:denial-of-service; sid:1000001; rev:1;)

Sääntö toimi ja Snort hälytti hyökkäyksestä. Normaali käyttötilanne tai suorituskyvyn testaus ApacheBenchmarkilla ei aiheuttanut vääriä positiivisia tulkintoja. Testitulosten perusteella Snort ei hidastanut sivun latautumista. Tulosten vaikuttaessa lupaaville IDS-tilassa Snort asennettiin uudelle palvelimelle. Palvelimella otettiin käyttöön kolme verkkokorttia joista kaksi asennettiin promiscuous-tilassa ilman IP-osoitetta ja yksi hallintakäyttöön. Promiscuous-tila tarkoittaa, että verkkoliitäntä ottaa vastaan kaiken kuulemansa liikenteen toisin kuin normaalitilassa, jolloin vastaanotetaan vain osoitteen perusteella liitännälle kuuluva liikenne. Promiscuous-liitännöistä toinen oli saapuvan liikenteen puoleisessa verkossa ja toinen palvelimen puoleisessa verkossa joten liikenne palvelimelle kulki uuden palvelimen kautta. Snortin manuaalin mukaisesti snort.conf-tiedostossa otettiin käyttöön afpacket DAQ (Data Acquisition library), joka tarvitaan Snortin käyttämiseksi inline-tilassa. Konfiguraatiossa otettiin siis käyttöön seuraavat rivit:
config daq: afpacket
config daq_mode: inline

Sääntöä muokattiin hieman vaihtamalla alert käskyyn drop:
drop tcp any any -> any $HTTP_PORTS (msg:"Slowloris"; content:"X-a "; dsize:<12; detection_filter:track by_dst, count 5, seconds 60; clastype:denial-of-service; sid:1000001; rev:1;)
Tämän jälkeen Snort käynnistettiin komennolla:
snort -A console -Q -c /etc/snort/snort.conf -i enp0s8:enp0s3 -N

Menetelmä vaikuttikin toimivan. Varsinaiselle palvelimelle asennettu Snort IDS-tilassa ei hälyttänyt Slowloris-hyökkäyksestä. ApacheBenchillä suoritettu testi kuitenkin antoi säännöllisesti ensimmäisellä testiajolla paremmat tulokset kuin seuraavilla toistoilla.

Concurrency Level: 100
Time taken for tests: 273.093 seconds
Complete requests: 1000
Failed requests: 230
(Connect: 0, Receive: 110, Length: 10, Exceptions: 110)
Total transferred: 6694600 bytes
HTML transferred: 6278425 bytes
Requests per second: 3.66 [#/sec] (mean)
Time per request: 27309.253 [ms] (mean)
Time per request: 273.093 [ms] (mean, across all concurrent requests)
Transfer rate: 23.94 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	1079 5049.4	2	31586
Processing:	2014	23509 37257.3	9613	130182
Waiting:	0	9352 5302.9	9573	60361
Total:	9235	24588 37215.5	9659	130182

Percentage of the requests served within a certain time (ms)
50% 9659
66% 10306
75% 12238
80% 12615
90% 130173
95% 130177
98% 130180

99% 130181
100% 130182 (longest request)

Tulos oli parempi kuin ilman Snort-sääntöä, mutta hyökkäyksen vaikutus näkyi kuitenkin selvästi.

(Angela Orebaugh 2005)

7.6 Snort ja iptables yhdessä

Seuraavana testinä otettiin käyttöön sekä palomuurisäännöt että Snort inline-tilassa. Hyökkäys suoritettiin Slowloris-skriptillä käyttäen kolmea hyökkäävää konetta. Tulos oli yllättävän hyvä.

ab -n 1000 -c 100 -s 220 -r http://192.168.0.1/mlinvoice/login.php

Concurrency Level: 100
Time taken for tests: 38.656 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 7480000 bytes
HTML transferred: 7015000 bytes
Requests per second: 25.87 [#/sec] (mean)
Time per request: 3865.586 [ms] (mean)
Time per request: 38.656 [ms] (mean, across all concurrent requests)
Transfer rate: 188.97 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	4 10.8	2	271
Processing:	2052	3719 911.8	3368	6400
Waiting:	2052	3718 911.8	3366	6399
Total:	2061	3723 909.7	3375	6400

Percentage of the requests served within a certain time (ms)

50% 3375

66% 4222

75% 4271

80% 4323

90% 5161

95% 5306

98% 6050

99% 6153

100% 6400 (longest request)

Tulos suojautumisessa kolmea yhtäaikaa hyökkäävää konetta vastaan oli Snort ja iptables yhdistettynä merkittävästi parempi kuin kumpikaan yksinään edes yhtä hyökkäävää konetta vastaan. 50 prosentissa pyyntöjä hyökkäyksen vaikutus oli normaalitilanteeseenkin verrattuna vain sekunnin luokkaa.

8 Yhteenveto

Palvelunestohyökkäykset ovat nykyään yleinen ongelma. Ne voivat pahimmillaan kaataa tehokkaankin palvelimen minuuteissa ja hyökkäyksen toteuttaminen ei enää vaadi teknistä osaamista vaan se voidaan suorittaa valmiilla skripteillä tai jopa ostaa palveluna. Hyökkäysmenetelmät jakaantuvat karkeasti volumetrisiin hyökkäyksiin joiden tarkoitus on kuluttaa kohteen Internetyhteyden kaista hyökkäysliikenteellä, TCP State-Exhaustion hyökkäyksiin verkon aktiivilaitteita vastaan ja hitaisiin sovelluskerroksen hyökkäyksiin. Hyökkäyksien havaitseminen normaalin verkkoliikenteen seasta voi olla haastavaa, varsinkin jos kyseessä on hajautettu hidas hyökkäys. Testeissä käytetyllä Snort-ohjelmistolla onnistuttiin havaitsemaan testihyökkäys, mutta kuten kaikki sääntöihin perustuvat tunkeutumisen havaitsemisjärjestelmät myös Snort havaitsee vain hyökkäykset joille sillä on toimiva sääntö. Iptables on myös tehokas työkalu, mutta varsinkin hajautetun hyökkäyksen torjumiseen esimerkiksi liikenteen rajoittaminen IP-osoitekohtaisesti ei ole erityisen tehokas keino ellei osoitteita aseteta mustalle listalle. Tämä taas saattaa helposti aiheuttaa kohtuuttomasti ongelmia laillisillekin käyttäjille esimerkiksi väärennettyjen osoitteiden takia tai NAT-palvelun takana olevien käyttäjien osalta. Kahden menetelmän yhdistelmän, eli Snort inline-tilassa ja iptables havaittiin olevan tehokas menetelmä sovelluskerrosten hyökkäyksiltä suojaukseen. Säännöissä olisi vielä paljon kehittämisen varaa eri tyyppisiä hyökkäyksiä varten.

Torjuntamenetelmistä havaittiin myös, että paikallisesti puolustautuminen on mahdollista lähinnä hitaita hyökkäyksiä vastaan. Volumetrisissa hyökkäyksissä kohteen verkkoyhteys saturoituu hyökkäysliikenteestä, joten tehokas puolustautuminen vaatii liikenteen reitittämisen DNS- tai BGP-protokollaa käyttäen pilvipalveluina toteutettujen pesureiden kautta. Nämä suodattavat haitallisen hyökkäysliikenteen ja ohjaavat puhtaan liikenteen GRE-tunnelia pitkin alkuperäiseen kohteeseen. Vaihtoehtoja palveluntarjoajaksi on paljon, mutta osa niistä on rajattu tietyille käyttäjäryhmille tai käyttötarkoituksiin.

Torjuntamenetelmien aktivoiminen vasta silloin kun hyökkäyksen vaikutukset havaitaan voi olla liian myöhäistä. Vähintäänkin tulisi olla suunnitelma kuinka hyökkäyksen sattuessa toimitaan ja valmius puolustustoimien aktivoimiseen. Kriittisten palveluiden kohdalla myös mahdollisten varajärjestelmien toimivuus tulisi varmistaa.

Tehokkain menetelmä vaikuttaisi olevan pilvipalveluna toteutettu DDoS-suojaus, jonka kapasiteetti riittää suojaamaan myös volumetrisilta hyökkäyksiltä. Liikenteen pesurit suodattavat myös tehokkaasti sovelluskerroksen hyökkäyksiä. Liikenne voidaan ohjata pysyvästi palvelun kautta ja näin piilottaa palvelimen oma osoite näkyvistä. Tehokkaan suojauksen varjopuolena ovat kustannukset, mutta kohtuullisen suojaustason saa jo muutamalla kymmenellä eurolla kuukaudessa tehokkaampien suojauspakettien hinnan noustessa helposti jo vähintään satoihin euroihin.

Lähteet

A. D. Keromytis, V. Misra, ja D. Rubenstein. 2004. “SOS: an architecture for mitigating DDoS attacks”. *IEEE Journal on Selected Areas in Communications* 22 (1): 176–188. doi:10.1109/JSAC.2003.818807.

Amtul Saboor, Baber Aslam, Monis Akhlaq. 2013. “Experimental Evaluation of Snort against DDoS Attacks under Different Hardware Configurations”. *2013 2nd National Conference on Information Assurance (NCIA)* 1 (2): 31–37. doi:10.1109/NCIA.2013.6725321.

Angela Orebaugh, Jacob Babbin, Simon Biles. 2005. *Snort Cookbook: Solutions and Examples for Snort Administrators.* O’Reilly Media,Inc. ISBN: 9780596007911.

“Apache”. 2020. Viitattu 28. maaliskuuta 2020. https://httpd.apache.org/.

“Arbor Cloud DDoS Protection Services”. 2020. Viitattu 7. toukokuuta 2020. https://www.netscout.com/product/arbor-cloud.

“AWS Shield”. 2020. Viitattu 7. helmikuuta 2020. https://aws.amazon.com/shield/.

“Azure DDoS Protection”. 2019. Viitattu 7. helmikuuta 2020. https://azure.microsoft.com/en-us/services/ddos-protection/.

Basseville, M. 1988. “Distance measures for signal processing and pattern recognition”. *Signal Processing* 18 (4): 349–369. doi:10.1016/0165-1684(89)90079-0.

“BGP Routing Explained”. 2020. Viitattu 6. toukokuuta 2020. https://www.cloudflare.com/learning/security/glossary/what-is-bgp/.

Cheng Jin, Kang G. Shin, Haining Wang. 2003. “Hop-Count Filtering: An Effective Defense Against SpoofedDDoS Traffic”. *Proceedings of the 10th ACM conference on Computer and communications security:* 30–41. doi:10.1145/948109.948116.

“Cloudflare plans and features”. 2020. Viitattu 20. maaliskuuta 2019. https://www.cloudflare.com/plans/.

"DDoS Protection With IPtables: The Ultimate Guide". 2020. Viitattu 29. maaliskuuta 2020. https://javapipe.com/blog/iptables-ddos-protection/.

"DNSFlood". 2020. Viitattu 29. maaliskuuta 2020. https://www.imperva.com/learn/application-security/dns-flood/.

"Ettercap Home Page". 2019. Viitattu 2. kesäkuuta 2020. https://www.ettercap-project.org/.

"Fork bomb attack". 2020. Viitattu 14. kesäkuuta 2020. https://www.imperva.com/learn/application-security/fork-bomb/.

"Guide to DDoS Attacks November 2017". 2020. Viitattu 3. toukokuuta 2020. https://www.cisecurity.org/wp-content/uploads/2017/03/Guide-to-DDoS-Attacks-November-2017.pdf.

Hakem Beitollahi, Geert Deconinck. 2012. "Analyzing well-known countermeasures against distributed denial of service attacks". *Computer Communications* 35 (15): 1312–1332. doi:10.1016/j.comcom.2012.04.008.

Hamza Rahmani, Farouk Kamoun, Nabil Sahli. 2012. "DDoS flooding attack detection scheme based on F-divergence". *Computer Communications* 35 (11): 1380–1391. doi:10.1016/j.comcom.2012.04.002.

"Imperva DDoS Attacks". 2019. Viitattu 7. huhtikuuta 2019. https://www.imperva.com/learn/application-security/ddos-attacks/.

"Imperva DDoS Protection". 2019. Viitattu 7. huhtikuuta 2019. https://www.imperva.com/products/ddos-protection-services/.

"Introduction: DDoS Protection for Networks". 2020. Viitattu 17. toukokuuta 2020. https://docs.imperva.com/bundle/cloud-application-security/page/introducing/network-ddos-protection.htm.

"IP Fragmentation Attack". 2020. Viitattu 14. kesäkuuta 2020. https://www.imperva.com/learn/application-security/ip-fragmentation-attack-teardrop/.

"iptables(8) - Linux man page". 2020. Viitattu 29. maaliskuuta 2020. https://linux.die.net/man/8/iptables.

"IPv6 Multicast Address Space Registry". 2020. Viitattu 3. toukokuuta 2020. https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml.

J. Francois, R. Boutaba, I. Aib. 2012. "FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks". *IEEE/ACM Transactions on Networking* 20 (6): 1828–1841. doi:10.1109/TNET.2012.2194508.

"KaliLinux". 2020. Viitattu 28. maaliskuuta 2020. https://www.kali.org/.

"Kaspersky. DDoS Attacks in Q1 2020." 2020. Viitattu 25. toukokuuta 2020. https://securelist.com/ddos-attacks-in-q1-2020/96837/.

Kumar, Sanjeev. 2005. "Impact of Distributed Denial of Service (DDoS) Attack Due to ARP Storm". *Networking - ICN 2005* 1 (1): 997–1002. doi:10.1007/978-3-540-31957-3_113.

Kübra Kalkan, Fatih Alagöz. 2016. "A distributed filtering mechanism against DDoS attacks: ScoreForCore". *Computer Networks* 108:199–209. doi:10.1016/j.comnet.2016.08.023.

Laura Feinstein, Ravindra Balupari, Dan Schnackenberg, ja Darrell Kindred. 2003. "Statistical Approaches to DDoS Attack Detection and Response". *Proceedings DARPA Information Survivability Conference and Exposition* 1 (1): 11–49. doi:10.1109/DISCEX.2003.1194894.

Muhammad Ejaz Ahmed, Hyoungshick Kim, Saeed Ullah. 2019. "Statistical Application Fingerprinting for DDoS Attack Mitigation". *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY* 14 (6): 1471–1484. doi:10.1109/TIFS.2018.2879616.

"Netscout AED datasheet". 2019. Viitattu 11. elokuuta 2019. https://www.netscout.com/sites/default/files/2019-03/SECPDS_013_EN-1901%20-%20NETSCOUT%20Arbor%20Edge%20Defense.pdf.

"Netscout. What is DDoS?" 2019. Viitattu 7. huhtikuuta 2019. https://www.netscout.com/what-is-ddos.

"Project Shield". 2020. Viitattu 7. helmikuuta 2020. https://projectshield.withgoogle.com/landing.

"Pusher. Per-IP rate limiting with iptables." 2019. Viitattu 7. huhtikuuta 2019. https://making.pusher.com/per-ip-rate-limiting-with-iptables/.

Qi Chen, Wanchun Dou, Wenmin Lin, ja Shui Yu. 2011. "CBF: A Packet Filtering Method for DDoS Attack Defense in Cloud Environment". *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing:* 427–434. doi:10.1109/DASC.2011.86.

Qiao Yan, Qingxiang Gong, F. Richard Yu, ja Jianqiang Li. 2016. "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges". *IEEE Communications Surveys & Tutorials* 18 (1): 602–622. doi:10.1109/COMST.2015.2487361.

"RFC1001". 1987. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc1001.

"RFC1002". 1987. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc1002.

"RFC1034". 1987. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc1034.

"RFC1035". 1987. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc1035.

"RFC1105". 1989. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc1105.

"RFC1157". 1990. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc1157.

“RFC1869”. 1995. Viitattu 20. huhtikuuta 2020. https://tools.ietf.org/html/rfc1869.

“RFC2132”. 1997. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc2132.

“RFC2616”. 1999. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc2616.

“RFC2784”. 2000. Viitattu 7. toukokuuta 2020. https://tools.ietf.org/html/rfc2784.

“RFC2818”. 2000. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc2818.

“RFC2890”. 2000. Viitattu 7. toukokuuta 2020. https://tools.ietf.org/html/rfc2890.

“RFC3352”. 2003. Viitattu 3. toukokuuta 2020. https://tools.ietf.org/html/rfc3352.

“RFC3410”. 2002. Viitattu 17. maaliskuuta 2020. https://tools.ietf.org/html/rfc3410.

“RFC3833”. 2004. Viitattu 17. maaliskuuta 2020. https://tools.ietf.org/html/rfc3833.

“RFC4987”. 2007. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc4987.

“RFC5321”. 2008. Viitattu 20. huhtikuuta 2020. https://tools.ietf.org/html/rfc5321.

“RFC5905”. 2007. Viitattu 4. huhtikuuta 2020. https://tools.ietf.org/html/rfc5905.

“RFC6335”. 2011. Viitattu 5. toukokuuta 2020. https://tools.ietf.org/html/rfc6335.

"RFC6762". 2013. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc6762.

"RFC6970". 2013. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc6970.

"RFC7231". 2014. Viitattu 15. maaliskuuta 2020. https://tools.ietf.org/html/rfc7231.

"RFC768". 1980. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc768.

"RFC791". 1981. Viitattu 20. maaliskuuta 2020. https://tools.ietf.org/html/rfc791.

"RFC792". 1981. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc792.

"RFC793". 1981. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc793.

"RFC821". 1982. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc821.

"RFC826". 1982. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc826.

"RFC854". 1983. Viitattu 28. maaliskuuta 2020. https://tools.ietf.org/html/rfc854.

Rossow, Christian. 2014. "Amplification Hell: Revisiting Network Protocols for DDoS Abuse". Teoksessa *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium.* Helmikuu.

S. Vidya, R. Bhaskaran. 2011. "ARP Storm Detection and Prevention Measures". *IJCSI International Journal of Computer Science Issues* 8 (2): 456–460. ISSN: 1694-0814.

"Security Solutions". 2020. Viitattu 14. kesäkuuta 2020. https://www.home.neustar/security-solutions?opt-out=true.

Seo, Dongwon, Heejo Lee ja Adrian Perrig. 2011. “PFS: Probabilistic filter scheduling against distributed denial-of-service attacks”. *2011 IEEE 36th Conference on Local Computer Networks:* 9–17. doi:10.1109/LCN.2011.6114645.

“sitespeed.io”. 2020. Viitattu 6. toukokuuta 2020. https://www.sitespeed.io/.

“Snort FAQ”. 2020. Viitattu 7. kesäkuuta 2020. https://www.snort.org/faq.

“SOAP”. 2020. Viitattu 3. toukokuuta 2020. https://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

“SQL Server Resolution Protocol”. 2019. Viitattu 3. toukokuuta 2020. https://docs.microsoft.com/en-us/openspecs/windows_protocols/mc-sqlr/5d3c0525-bcfb-44ad-85b3-143cbeb9494f.

“Study-ccna.com”. 2019. Viitattu 26. toukokuuta 2019. https://study-ccna.com/osi-tcp-ip-models/.

Suk-June Choi, Jin Kwak. 2017. “A study on reduction of DDoS amplification attacks in the UDP-based CLDAP protocol”. *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT):* 1–4. doi:10.1109/CAIPT.2017.8320670.

“Ubuntu Desktop”. 2020. Viitattu 28. maaliskuuta 2020. https://ubuntu.com/download/desktop.

“Ubuntu Server”. 2020. Viitattu 28. maaliskuuta 2020. https://ubuntu.com/download/server.

“Usage statistics of web servers”. 2020. Viitattu 10. toukokuuta 2020. https://w3techs.com/technologies/overview/web_server.

“What Is A Reverse Proxy? | Proxy Servers Explained”. 2020. Viitattu 14. kesäkuuta 2020. https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/.

“What is an UDP Flood Attack?” 2020. Viitattu 14. kesäkuuta 2020. https://www.netscout.com/what-is-ddos/udp-flood.

"VirtualBox". 2020. Viitattu 29. maaliskuuta 2020. https://www.virtualbox.org/.

Xin Liu, Yanbin Lu, Xiaowei Yang. 2008. "To filter or to authorize: Network-layer DoS Defense against multimillion-node botnets". *ACM SIGCOMM COMPUTER COMMUNICATION REVIEW* 38 (4): 195–206. doi:10.1145/1402946.1402981.

Yoohwan Kim, Mooi Choo Chuah, Wing Cheong Lau, ja H.J Chao. 2006. "PacketScore: A statistics-based packet filtering scheme against distributed denial-of-service attacks". *IEEE Transactions on Dependable and Secure Computing* 3 (2): 141–155. doi:10.1109/TDSC.2006.25.

Yu Chen, Kai Hwang. 2006. "Collaborative Change Detection of DDoS Attacks on Community and ISP Networks". *International Symposium on Collaborative Technologies and Systems (CTS'06):* 401–410. doi:10.1109/CTS.2006.27.

Zhang, Gui Shu, Ya Lan; Xia. 2013. "The SSL MIMT Attack with DNS Spoofing". *Applied Mechanics and Materials; Zurich* 385–386:1647–1650. doi:10.4028/www.scientific.net/AMM.385-386.1647.

Liitteet

A

1: Drop invalid packets

/sbin/iptables -t mangle -A PREROUTING -m conntrack –ctstate INVALID -j DROP

2: Drop TCP packets that are new and are not SYN

/sbin/iptables -t mangle -A PREROUTING -p tcp ! –syn -m conntrack –ctstate NEW -j DROP

3: Drop SYN packets with suspicious MSS value

/sbin/iptables -t mangle -A PREROUTING -p tcp -m conntrack –ctstate NEW -m tcpmss ! –mss 536:65535 -j DROP

4: Block packets with bogus TCP flags

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags FIN,SYN FIN,SYN -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags SYN,RST SYN,RST -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags FIN,RST FIN,RST -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags FIN,ACK FIN -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ACK,URG URG -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ACK,FIN FIN -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ACK,PSH PSH -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ALL ALL -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ALL NONE -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ALL FIN,PSH,URG -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ALL SYN,FIN,PSH,URG -j DROP

/sbin/iptables -t mangle -A PREROUTING -p tcp –tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

5: Block spoofed packets

/sbin/iptables -t mangle -A PREROUTING -s 224.0.0.0/3 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 169.254.0.0/16 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 172.16.0.0/12 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 192.0.2.0/24 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 192.168.0.0/16 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 10.0.0.0/8 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 0.0.0.0/8 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 240.0.0.0/5 -j DROP

/sbin/iptables -t mangle -A PREROUTING -s 127.0.0.0/8 ! -i lo -j DROP

6: Drop ICMP (you usually don’t need this protocol)

/sbin/iptables -t mangle -A PREROUTING -p icmp -j DROP

7: Drop fragments in all chains

/sbin/iptables -t mangle -A PREROUTING -f -j DROP

8: Limit connections per source IP

/sbin/iptables -A INPUT -p tcp -m connlimit –connlimit-above 111 -j REJECT –reject-with

tcp-reset

9: Limit RST packets

/sbin/iptables -A INPUT -p tcp –tcp-flags RST RST -m limit –limit 2/s –limit-burst 2 -j ACCEPT

/sbin/iptables -A INPUT -p tcp –tcp-flags RST RST -j DROP

10: Limit new TCP connections per second per source IP

/sbin/iptables -A INPUT -p tcp -m conntrack –ctstate NEW -m limit –limit 60/s –limit-burst 20 -j ACCEPT

/sbin/iptables -A INPUT -p tcp -m conntrack –ctstate NEW -j DROP

11: Use SYNPROXY on all ports (disables connection limiting rule)

iptables -t raw -A PREROUTING -p tcp -m tcp –syn -j CT –notrack

iptables -A INPUT -p tcp -m tcp -m conntrack –ctstate INVALID,UNTRACKED -j SYN-PROXY –sack-perm –timestamp –wscale 7 –mss 1460

iptables -A INPUT -m conntrack –ctstate INVALID -j DROP

B

kernel.printk = 4 4 1 7

kernel.panic = 10

kernel.sysrq = 0

kernel.shmmax = 4294967296

kernel.shmall = 4194304

kernel.core_uses_pid = 1

kernel.msgmnb = 65536

kernel.msgmax = 65536

vm.swappiness = 20

vm.dirty_ratio = 80

vm.dirty_background_ratio = 5

fs.file-max = 2097152

net.core.netdev_max_backlog = 262144

net.core.rmem_default = 31457280

net.core.rmem_max = 67108864

net.core.wmem_default = 31457280

net.core.wmem_max = 67108864

net.core.somaxconn = 65535

net.core.optmem_max = 25165824

net.ipv4.neigh.default.gc_thresh1 = 4096

net.ipv4.neigh.default.gc_thresh2 = 8192

net.ipv4.neigh.default.gc_thresh3 = 16384

net.ipv4.neigh.default.gc_interval = 5

net.ipv4.neigh.default.gc_stale_time = 120

net.netfilter.nf_conntrack_max = 10000000

net.netfilter.nf_conntrack_tcp_loose = 0

net.netfilter.nf_conntrack_tcp_timeout_established = 1800

net.netfilter.nf_conntrack_tcp_timeout_close = 10

net.netfilter.nf_conntrack_tcp_timeout_close_wait = 10

net.netfilter.nf_conntrack_tcp_timeout_fin_wait = 20

net.netfilter.nf_conntrack_tcp_timeout_last_ack = 20

net.netfilter.nf_conntrack_tcp_timeout_syn_recv = 20

net.netfilter.nf_conntrack_tcp_timeout_syn_sent = 20

net.netfilter.nf_conntrack_tcp_timeout_time_wait = 10

net.ipv4.tcp_slow_start_after_idle = 0

net.ipv4.ip_local_port_range = 1024 65000

net.ipv4.ip_no_pmtu_disc = 1

net.ipv4.route.flush = 1

net.ipv4.route.max_size = 8048576

net.ipv4.icmp_echo_ignore_broadcasts = 1

net.ipv4.icmp_ignore_bogus_error_responses = 1

net.ipv4.tcp_congestion_control = htcp

net.ipv4.tcp_mem = 65536 131072 262144

net.ipv4.udp_mem = 65536 131072 262144

net.ipv4.tcp_rmem = 4096 87380 33554432

net.ipv4.udp_rmem_min = 16384

net.ipv4.tcp_wmem = 4096 87380 33554432

net.ipv4.udp_wmem_min = 16384

net.ipv4.tcp_max_tw_buckets = 1440000

net.ipv4.tcp_tw_recycle = 0

net.ipv4.tcp_tw_reuse = 1

net.ipv4.tcp_max_orphans = 400000

net.ipv4.tcp_window_scaling = 1

net.ipv4.tcp_rfc1337 = 1

net.ipv4.tcp_syncookies = 1

net.ipv4.tcp_synack_retries = 1

net.ipv4.tcp_syn_retries = 2

net.ipv4.tcp_max_syn_backlog = 16384

net.ipv4.tcp_timestamps = 1

net.ipv4.tcp_sack = 1

net.ipv4.tcp_fack = 1

net.ipv4.tcp_ecn = 2

net.ipv4.tcp_fin_timeout = 10

net.ipv4.tcp_keepalive_time = 600

net.ipv4.tcp_keepalive_intvl = 60

net.ipv4.tcp_keepalive_probes = 10

net.ipv4.tcp_no_metrics_save = 1

net.ipv4.ip_forward = 0

net.ipv4.conf.all.accept_redirects = 0

net.ipv4.conf.all.send_redirects = 0

net.ipv4.conf.all.accept_source_route = 0

net.ipv4.conf.all.rp_filter = 1