Markus Neero

SOVELLUSKERROKSEN PALVELUNESTOHYÖKKÄYKSET JA NIILTÄ SUOJAUTUMINEN

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2023

TIIVISTELMÄ

Neero, Markus
Sovelluskerroksen palvelunestohyökkäykset ja niiltä suojautuminen
Jyväskylä: Jyväskylän yliopisto, 2023, 55 s.
Kyberturvallisuus, pro gradu -tutkielma
Ohjaaja(t): Hämäläinen, Timo

Palvelunestohyökkäykset ovat olleet vakava uhka tietoverkoille jo vuosikymmenten ajan. Palvelunestohyökkäysten kehittyminen on johtanut uuden tyyppisten hyökkäysten kehittymiseen, joita kutsutaan sovelluskerroksen palvelunestohyökkäyksiksi. Sovelluskerroksen palvelunestohyökkäykset eivät vaadi aikaisempiin hyökkäyksiin verraten niin paljoa resursseja, ja ne ovat usein myös vaikeampia havaita. Tässä tutkielmassa on esitelty erilaisia sovelluskerroksen protokollia, niitä vastaan kehitettyjä hyökkäyksiä, sekä puolustuskeinoja näitä hyökkäyksiä vastaan. Lisäksi tutkielman tutkimusosiossa testattiin SlowHTTPTest -työkalun avulla erilaisia palvelunestohyökkäyksiä Apachen uusinta palvelinohjelmistoa vastaan ja analysoitiin, miten hyökkäykset vaikuttivat palvelimen kykyyn vastata pyyntöihin. Testauksen tuloksena voidaan todeta, että Apachen palvelinohjelmisto on vakioasetuksillaan edelleen altis sovelluskerroksen palvelunestohyökkäyksille.

Asiasanat: palvelunestohyökkäys, sovelluskerros, Apache

ABSTRACT

Neero, Markus
Sovelluskerroksen palvelunestohyökkäykset ja niiltä suojautuminen
Jyväskylä: University of Jyväskylä, 2023, 55 pp.
Cyber Security, Master's Thesis
Supervisor(s): Hämäläinen, Timo

Denial-of-Service attacks have been a serious threat to computer networks for decades. The development of these attacks has led to the emergence of a new type known as application layer Denial-of-Service. Unlike previous attacks, these application layer attacks do not require as many resources and are often more difficult to detect. This study examines different application layer protocols, the attacks that have been developed against them, and the defense mechanisms employed to mitigate such attacks. Furthermore, the research conducted in this study involved testing various Denial-of-Service attacks using the Slow-HTTPTest tool against the latest Apache server software. The analysis focused on understanding the impact of these attacks on the server's ability to respond to requests. The results demonstrate that, with its default settings, the Apache server software remains vulnerable to application layer Denial-of-Service attacks.

Keywords: Denial-of-Service, Application layer, Apache

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Palvelunestohyökkäykset ovat olleet vakava uhka tietoverkoille jo vuosikymmenten ajan (Zargar, Joshi & Tipper, 2013). Palvelunestohyökkäykset ovat kehittyneet jatkuvasti, mutta myös uusia puolustuskeinoja kehitetään niiden torjumiseksi. Puolustuskeinojen kehittyminen on johtanut uudentyyppisten, sovelluskerroksen palvelunestohyökkäysten, kehittymiseen. (Gonzalez ym., 2014).

Palvelunestohyökkäyksellä tarkoitetaan hyökkäystä, jossa eri keinoja käyttämällä pyritään estämään järjestelmien oikeutettuja käyttäjiä käyttämästä palveluja. (Carl ym., 2006) Sovelluskerroksen palvelunestohyökkäykset ovat palvelunestohyökkäyksiä, jotka käyttävät hyväkseen sovelluskerroksen protokollien vikoja ja haavoittuvuuksia. (Tripathi & Hubballi, 2021)

Jotta sovelluskerroksen palvelunestohyökkäyksiltä voitaisiin puolustautua, on tärkeää ymmärtää, minkälaisia hyökkäyksiä on olemassa, miten niitä vastaan voidaan puolustautua ja minkälaiset teknologiat vaikuttavat hyökkäysten taustalla. Tämä tutkielma tarjoaa yleiskatsauksen sovelluskerroksen palvelunestohyökkäyksiin ja niiden taustalla vaikuttaviin sovelluskerroksen protokolliin.

Sovelluskerroksen palvelunestohyökkäyksistä on ennenkin tehty tutkimuksia, joissa eri hyökkäyksiä kootaan yhteen (esim. Tripathi & Hubballi, 2021 ja Mantas ym., 2015) Mantasin ym. (2015) käsittelee hyökkäyksiä melko monipuolisesti, mutta ei esittele mitään puolustusmekanismeja hyökkäyksiä vastaan. Artikkelissa ei myöskään esitellä protokollien toimintaperiaatteita, jotka auttaisivat ymmärtämään hyökkäysten toimintaa. Tripathin ja Hubballin (2021) tutkimus on todennäköisesti tähän mennessä kattavin esitys sovelluskerroksen palvelunestohyökkäyksistä ja niiden torjunnasta. Tutkimuksessa hyökkäyksiä käsitellään kuitenkin melko pintapuolisesti, eikä taustalla vaikuttavien protokollien toimintaa avata lukijalle. Lukijan voidaan olettaa ymmärtävän sovelluskerroksen protokollien toiminnan perustasolla jo etukäteen, mutta tämä tutkielma antaa paremman kokonaiskuvan sovelluskerroksen hyökkäysten toiminnasta. Tripathin ja Hubballin (2021) tutkimus käsittelee myös tulvahyökkäyksiä, jotka yleensä katsotaan kuuluvaksi hajautettujen palvelunestohyökkäysten kategoriaan, koska niiden toteuttaminen pienillä resursseilla on hyvin vaikeaa. Tämän Tripathi ja

Hubballi (2021) toteavat myös omassa tutkimuksessaan. (Tripathi & Hubballi, 2021).

Pro Gradun tutkimusosiossa testataan yleisiä Hypertext Transfer Protocol-protokollaa (jatkossa HTTP-protokolla) hyödyntäviä sovelluskerroksen palvelunestohyökkäyksiä uusinta Apachen palvelinohjelmistoa vastaan ja pyritään selvittämään, onko ohjelmiston uusin versio vakioasetuksillaan edelleen haavoittuva kyseisille hyökkäyksille. Tutkimusosiossa testataan puhtaasti palvelinohjelmiston kykyä selvitä sovelluskerroksen palvelunestohyökkäyksistä, eikä käytössä ole erityisiä suojausmekanismeja. Apachen palvelinohjelmiston aikaisempien versioiden tiedetään olevan alttiita erilaisille sovelluskerroksen palvelunestohyökkäyksille ja tämä uhka on tiedostettu tutkimuskirjallisuudessa jo pidemmän aikaa (esim. Catillo, Pecchia & Villano, 2022 ja Suroto, 2017). Tutkielman tekijän tiedossa ei kirjoitushetkellä ole yhtään tutkimusta, jossa hitaita palvelunestohyökkäyksiä olisi testattu Apachen palvelinohjelmiston uusinta versiota vastaan. Tässä tutkielmassa pyritäänkin selvittämään, onko aikaisemmassa tutkimuksessa esitelty tieto johtanut sellaisiin käytännön muutoksiaan Apachen palvelinohjelmiston uusimmassa versiossa, jotka merkittävästi tai kokonaan suojaisivat palvelinohjelmistoa hitailta HTTP-protokollaan perustuvilta sovelluskerroksen palvelunestohyökkäyksiltä. Tämä tutkielma pyrkii vastaamaan seuraaviin tutkimuskysymyksiin:

- Minkälaisia protokollia käytetään sovelluskerroksella ja minkälainen on niiden perustoiminta?
- Minkälaisia sovelluskerroksen palvelunestohyökkäyksiä on esitelty aikaisemmassa tutkimuskirjallisuudessa ja miten niiltä voidaan puolustautua?
- Onko Apachen uusin palvelinohjelmisto altis sovelluskerroksen palvelunestohyökkäyksille, jos käytössä ei ole muita suojausmekanismeja?

Tutkielman kirjallisuuskatsausosiossa lähteinä on käytetty muun muassa alan kirjallisuutta, konferenssijulkaisuja sekä erilaisia internetiä koskevia standardeja. Kirjallisuuden hakuun on käytetty erilaisia tietokantoja, kuten Google Scholar ja IEEE Xplorer.

Tutkielman tuloksena voidaan sanoa, että Apachen uusin palvelinohjelmisto on yksinään vielä erittäin haavoittuvainen yleisille hitaille HTTP-protokollaan perustuville palvelunestohyökkäyksille. Käytännössä hyökkäyksiltä puolustautuminen vaatii lisäksi muita puolustusmekanismeja ja ymmärrystä siitä, miten sovelluskerroksen palvelunestohyökkäykset hyödyntävät sovelluskerroksen protokollia hyökkäysten toteuttamiseen.

Luvussa 2 esitellään kaksi mallia, joita voidaan käyttää tietoverkkojen kuvaamiseen kerroksittain. Nämä mallit ovat OSI-malli, sekä TCP/IP-malli. Tutkielman luvussa 3 kerrotaan palvelunestohyökkäyksistä, sekä siitä, miten sovelluskerroksen palvelunestohyökkäykset eroavat perinteisistä

palvelunestohyökkäyksistä. Lisäksi luvussa kerrotaan minkälaisia motiiveja hyökkäysten takana voi olla. Luvussa 4 esitellään neljä erilaista sovelluskerroksen protokollaa: Dynamic Host Configuration Protocol (jatkossa DHCP), HTTP, Domain Name System (jatkossa DNS) ja Network Time Protocol (jatkossa NTP). Luvussa viisi esitellään protokollittain erilaisia aikaisemmassa tutkimuskirjallisuudessa esiteltyjä sovelluskerroksen palvelunestohyökkäyksiä ja esitellään niihin ehdotettuja suojauskeinoja. Luvussa kuusi on tutkielman tutkimusosio, jossa testataan yleisiä hitaita palvelunestohyökkäyksiä Apachen palvelinohjelmistoa vastaan ja analysoidaan testien tuloksia. Luku seitsemän on tutkielman yhteenveto.

2 TIETOVERKKOJEN KUVAUS KERROKSITTAIN OSI-MALLIN JA TCP-IP-MALLIN AVULLA

Tietoverkot voivat olla hyvinkin monimutkaisia järjestelmiä. Tietoverkkojen jakaminen kerroksittain loogisiin kokonaisuuksiin, auttaa ymmärtämään paremmin niiden toimintaa ja helpottaa esimerkiksi protokollien kehitystä ja ongelmien selvittämistä. Tässä kappaleessa esitellään kaksi tunnettua mallia, jotka käsittelevät tietoverkkoja kerroksellisina arkkitehtuureina. Nämä mallit ovat Open Systems Interconnection Model -malli (jatkossa OSI-malli), sekä Transmission Control Protocol/Internet Protocol -malli (jatkossa TCP/IP-malli).

2.1 OSI-malli

OSI-malli on the International Organization for Standardizationin (jatkossa ISO) ja the International Electrotechinal Commissionin (jatkossa IEC) alun perin vuonna 1984 julkaisema standardi, jonka tarkoituksena on toimia viitekehyksenä järjestelmien väliselle viestinnälle. Nimessä esiintyvällä Open -sanalla (suom. avoin) viitataan siihen, että standardia noudattavat järjestelmät pystyisivät viestimään toistensa kanssa maailmanlaajuisesti. (Day & Zimmermann, 1983) Avoimuus ei kuitenkaan tarkoita mitään tiettyä teknologiaa tai toteutusta, vaan viittaa sovellettavien standardien vastavuoroiseen tunnustamiseen ja tukemiseen. OSI-mallin tarkoitus ei myöskään ole tarjota toteutusohjeita protokollille tai palveluille tai toimia lähtökohtana todellisten toteutusten yhteensopivuuden arviointiin. ISO-malli on konseptuaalinen viitekehys, jota asiantuntijat voivat käyttää hyväkseen tietoverkkojen ja standardien kehityksessä. (ISO/IEC 7498-1, 1994) OSI-malli ei myöskään ole tarkka kuvaus tietoverkkojen toiminnasta oikeassa elämässä. Käytännön elämässä jokin yksi protokolla saattaa suorittaa tehtäviä, jotka OSI-mallin mukaisesti kuuluisivat eri kerroksille. Joidenkin nykypäivänä käytössä olevien protokollien sijoittaminen OSI-malliin voikin olla muun muassa tästä syystä hankalaa. (Noergaard, 2005, s. 54)

OSI-mallissa tietoverkot jaetaan seitsemään eri kerrokseen. Kerros (N) käyttää sen alapuolella sijaitsevan kerroksen palveluita (N-1) ja tarjoaa palveluita sen yläpuolella sijaitsevalle kerrokselle (N+1). Näin tietoverkko saadaan jaettua loogisesti pienempiin osiin, joita voidaan käsitellä suhteellisen itsenäisesti. OSI-mallin kerrokset alhaalta ylös ovat: 1. fyysinen kerros, 2. siirtoyhteyskerros, 3. verkkokerros, 4. kuljetuskerros, 5. istuntokerros, 6. esitystapakerros ja 7. sovelluskerros. (Day & Zimmermann, 1983)

Ensimmäinen kerros on nimeltään fyysinen kerros. Nimensä mukaisesti fyysinen kerros sisältää verkon sähköiset ja mekaaniset ominaisuudet, jotka mahdollistavat fyysisten yhteyksien aktivoinnin, sulkemisen, ylläpidon, sekä bittien siirtämisen tietoverkoissa. Fyysiseen kerrokseen kuuluvat muun muassa kaapelit ja liittimet. (ISO/IEC 7498-1, 1994)

Toinen kerros on nimeltään siirtoyhteyskerros. Siirtoyhteys voidaan muodostaa yhden tai useamman fyysisen yhteyden päälle. Siirtoyhteyskerroksen tärkein tehtävä on varmistaa luotettava tiedonsiirto lähiverkon solmujen välillä. (Conard, 1983) Siirtoyhteyskerros muodostaa fyysiseltä kerrokselta saamistaan biteistä ryhmiä eli kehyksiä. Siirtoyhteyskerros pyrkii muun muassa varmistamaan, että kehykset on vastaanotettu kokonaan, niissä ei ole virheitä, ja että ne ovat tulleet oikealle laitteelle. Mikäli data on tullut oikeaan osoitteeseen, siirtoyhteyskerros poistaa kehyksestä siiroyhteyskerroksen otsakkeet ja siirtää loput tiedot, eli datagrammin, seuraavalla kerrokselle. (Noergaard, 2005, s.59)

Kolmas kerros on nimeltään verkkokerros. Verkkokerros mahdollistaa verkkoyhteyden muodostamisen, ylläpitämisen ja päättämisen järjestelmien välillä. (ISO/IEC 7498-1, 1994) Verkkokerros huolehtii datagrammien reitittämisestä laitteiden välillä hyödyntäen verkko-osoitteita ja fyysisiä osoitteita, sekä mahdollistaa verkko-osoitteiden muuntamisen niitä vastaaviksi fyysisiksi osoitteiksi. Verkkokerros poistaa verkkokerros -otsakkeet siirtoyhteyskerrokselta saamastaan datagrammista ja siirtää sen seuraavalla kerrokselle, eli kuljetuskerrokselle. Kuljetuskerrokselle lähetettävää sisältöä kutsutaan paketiksi. (Noergaard, 2005, s. 60)

Neljäs kerros on nimeltään kuljetuskerros. Kuljetuskerroksen protokollien tehtävänä on yleisesti ottaen kommunikaation muodostaminen ja päättäminen kahden laitteen välillä. Kuljetuskerroksen palvelut voivat myös protokollasta riippuen huolehtia muun muassa siitä, että paketit vastaanotetaan ja lähetetään oikeassa järjestyksessä, sekä sopivalla nopeudella. Kuljetuskerroksen eri protokollat sisältävät myös muita virheen havaitsemiseen ja korjaamiseen liittyviä mekanismeja ja voivat pyytää esimerkiksi paketin uudelleen lähettämistä virheen tapahtuessa. (Iren, Amer & Conrad, 1999). Kuljetuskerros vastaanottaa paketin verkkokerrokselta, se poistaa paketista kuljetuskerroksen otsakkeet ja muodostaa niistä uusia paketteja, joita kutsutaan myös viesteiksi. Viestit lähetetään ylemmälle kerrokselle. (Noergaard, 2005, s.61).

Viides kerros on nimeltään istuntokerros. Istunnolla tarkoitetaan yhteyttä kahden, eri laitteilla sijaitsevien, verkko-ohjelmistojen välillä. Siinä missä kuljetuskerros huolehtii kommunikaation muodostamisesta yleisesti eri laitteilla sijaitsevien ohjelmistojen välillä, istuntokerros huolehtii juuri istunnon

hallinnoinnista. Istuntokerroksen tehtäviin kuuluu muun muassa istuntojen datan hallinta, datavirran sääteleminen ja istunnoissa tapahtuvien virheiden käsitteleminen. Istuntokerros poistaa kuljetuskerrokselta saaduista viesteistä istuntokerroksen otsakkeet ja siirtää datan ylemmälle kerrokselle. (Noergaard, 2005, s.62).

Kuudes kerros on nimeltään esitystapakerros. Esitystapakerroksen tehtävänä on tarjota ylemmän kerroksen ohjelmistoille esitystapa, jota ne voivat käyttää keskinäisessä kommunikoinnissaan. (ISO/IEC 7498-1, 1994). Käytännössä tämä tarkoittaa datan muokkaamista sellaiseen muotoon, että ylemmän kerroksen protokollat voivat sitä prosessoida. Istuntokerrokselta saaduista viesteistä poistetaan esitystapakerroksen otsakkeet ja viesti välitetään seuraavalla ja ylimmälle kerrokselle, eli sovelluskerrokselle.

Seitsemäs ja ylin kerros OSI-mallissa on sovelluskerros. Sovelluskerros sisältää kaikki toiminnot, joita alemmilla kerroksilla ei ole suoritettu. Nämä sisältävät niin ohjelmistojen kuin myös ihmisten toiminnan. Sovelluskerroksen toimintoihin voi kuulua muun muassa kommunikoinnin osapuolien tunnistaminen esimerkiksi nimen tai osoitteen perusteella, turvallisuudesta, kuten käyttäjien tunnistamisesta huolehtiminen, sekä synkronisointi sovellusten välillä. (ISO/IEC 7498-1, 1994). Sovelluskerros on se kerros, jonka kanssa käyttäjä on yleensä vuorovaikutuksessa. Sovelluskerroksen protokollat mahdollistavat käyttäjän, oli se sitten ihminen tai ohjelmisto, pääsyn tietoverkkoon. Se mahdollistaa käyttöliittymän ja monet ihmisten käyttämät palvelut, kuten tiedostojen jakamisen ja sähköpostin.

2.2 TCP/IP-malli

TCP/IP-malli juontaa juurensa Yhdysvaltaiseen pakettikytkentäiseen tietoverkkoon, joka tunnetaan nimellä Advanced Research Projects Agency Network (jatkossa ARPANET). Vaikka TCP/IP-mallin ensimmäinen versio esiteltiin jo vuonna 1974 (Cerf & Kahn, 1974), otettiin se ARPANET:ssä käyttöön vasta vuonna 1983. Samana vuonna APRANET:ä alettiin yleisesti kutsua internetiksi. TCP/IP-malli sai nimensä kahdesta protokollasta, joihin mallin voidaan sanoa perustuvan. Nämä protokollat ovat TCP-protokolla ja IP-protokolla. Siinä missä OSI-malli kehitettiin viitekehykseksi tukemaan protokollien ja tietoverkkojen kehitystä, TCP/IP-malli kehitettiin jo olemassa olevien protokollien varaan. Tämän lisäksi TCP/IP-mallin laitteisto- ja ohjelmistoriippumattomuus, standardien ilmainen saatavuus ja laaja dokumentaatio olivat syitä sille, että TCP/IP-malli on OSI-malliin verrattuna levinnyt hyvin laajasti käytännön toteutuksiin. (Russel, 2013). OSI-, ja TCP/IP-malli on esitetty kerroksittain alla olevassa kuviossa (kuvio 1).

KUVIO 1 OSI-, ja TCP/IP-mallit kerroksittain

OSI-mallin tapaan myös TCP/IP-malli on kerroksellinen arkkitehtuuri. Kerroksia on kuitenkin seitsemän sijasta vain neljä. Kerrokset ovat alhaalta ylös 1. peruskerros, 2. verkkokerros, 3. kuljetuskerros ja 4. sovelluskerros. OSI-malliin verraten OSI-mallin siirtoyhteyskerros, sekä fyysinen kerros ovat sisällytetty TCP/IP-mallin peruskerrokseen ja sovelluskerros, esitystapakerros, sekä istuntokerros on yhdistetty vain sovelluskerrokseksi. (Alani, 2014).

TCP/IP-mallin ensimmäinen kerros on nimeltään peruskerros. TCP/IP-mallissa peruskerrosta ei määritellä mitenkään tarkasti, eikä sille ole määritelty esimerkiksi mitään tiettyjä protokollia. Ei ole väliä minkälaisia teknologioita tai protokollia käytetään, kunhan tieto siirtyy verkkokerrokselle ja verkkokerrokselta peruskerrokselle. Myös IP-osoitteiden linkittäminen niitä vastaaviin fyysisiin osoitteisiin kuuluu tämän kerroksen tehtäviin. (Alani, 2014).

Toinen kerros on nimeltään verkkokerros ja sen tärkein protokolla on IP-protokolla. IP-protokollan tärkeimpiä tehtäviä on pakettien reitittäminen käyttäen IP-osoitteita, datan siirtäminen peruskerroksen ja kuljetuskerroksen välillä, sekä tarvittaessa pakettien jakaminen pienempiin palasiin ja niiden uudelleen kokoaminen paketin määränpäässä. IP-protokolla ei sisällä mekanismeja virheiden hallinnalle tai esimerkiksi vuonvalvonnalle, vaan nämä on jätetty muiden protokollien tehtäväksi. Verkkokerroksella muita IP-protokollaa tukevia protokollia ovat muun muassa ICMP-protokolla ja ARP-protokolla. (Postel, 1981, a).

TCP/IP-mallin kolmas kerros on kuljetuskerros. Kuljetuskerroksen tehtävänä on mahdollistaa kommunikaatio päätelaitteiden välillä. Kuljetuskerroksen pääprotokollat ovat TCP-protokolla ja User Datagram Protocol -protokolla (jatkossa UDP-protokolla). TCP-protokolla ja UDP-protokolla eroavat toisistaan varsin merkittävästi. TCP-protokolla on yhteydellinen protokolla siinä missä UDP-protokolla on yhteydetön protokolla. Yhteydellisyys tarkoittaa sitä, että laitteiden tulee muodostaa keskenään yhteys ennen tietojen vaihtoa. TCP-protokolla on niin sanotusti luotettava protokolla ja se sisältää monia mekanismeja erilaisten virheiden hallintaan. TCP-protokolla muun muassa varmistaa, että paketit

saapuvat vastaanottajalle oikeassa järjestyksessä, hävinneet paketit lähetetään uudelleen, vastaanottajalle ei lähetetä enempää dataa kuin se voi vastaanottaa ja virheelliset paketit hylätään. (Postel, 1981, b).

UDP on hyvin yksinkertainen ja niin sanotusti epäluotettava protokolla. Toisin kun TCP-protokollassa, UDP:ta käytettäessä ei ole mitään varmuutta siitä, että lähetetty data päätyy koskaan määränpäähänsä. (Postel, 1980). Toisaalta UDP-protokollaa voidaan pitää luotettavuuden varmistavien mekanismien puuttumisen vuoksi TCP-protokollaa tehokkaampana. UDP-protokolla voikin olla parempi protokolla tilanteisiin, jossa esimerkiksi pakettien mahdollinen häviäminen ei ole niin merkityksellistä. Esimerkki tällaisesta tilanteesta voisi olla IP-puhe, jossa ääntä siirretään reaaliaikaisesti IP-verkon välityksellä. (Alani, 2014).

TCP/IP-mallin neljäs ja ylin kerros on nimeltään sovelluskerros. TCP/IP-mallissa sovelluskerrokseen on sisällytetty kolme OSI-mallissa olevaa kerrosta, jotka ovat istuntokerros, esitystapakerros ja sovelluskerros. Sovelluskerroksen tärkein tehtävä on vastaanottaa data sovelluksilta ja siirtää se kuljetuskerrokselle ja toisinpäin. Sovelluskerros sisältää monia erilaisia protokollia, kuten HTTP-protokollan, DHCP-protokollan, DNS-protokollan ja NTP-protokollan. (Alani, 2014).

3 PALVELUNESTOHYÖKKÄYKSET

Tässä luvussa kerrotaan mitä palvelunestohyökkäykset ovat, miten niitä voidaan luokitella, sekä minkälaisia motiiveja hyökkääjillä saattaa olla hyökkäysten suorittamiseen. Lisäksi luvussa kerrotaan miten sovelluskerroksen palvelunestohyökkäykset eroavat muista palvelunestohyökkäyksistä.

3.1 Palvelunestohyökkäyksistä yleisesti

Palvelunestohyökkäys on hyvin laaja-alainen termi, jolla voidaan kuvata kaikenlaisia hyökkäyksiä, joiden tarkoituksena on estää järjestelmien oikeutettuja käyttäjiä käyttämästä palveluja. Carlin ym. (2006) mukaan palvelunestohyökkäyksenä voitaisiin pitää esimerkiksi sitä, että hyökkääjä irrottaa virtakaapelin kohteensa tietokoneesta. (Carl ym., 2006).

Palvelunestohyökkäykset jaetaan yleisesti palvelunestohyökkäyksiin, sekä hajautettuihin palvelunestohyökkäyksiin. Perimmäinen ero näiden välillä on se, että hajautetussa palvelunestohyökkäyksessä hyökkäyksen toteuttamiseen käytetään useampaa laitetta, kun taas ”perinteisessä” palvelunestohyökkäyksessä vain yhtä. Hajautetussa palvelunestohyökkäyksessä käytetyt laitteet ovat usein kaapattuja. Hyökkääjä voi esimerkiksi etsiä, joko manuaalisesti tai automatisoidusti, haavoittuvuuksia käyttäjien tietokoneista ja syöttää niihin tarvittavan hyökkäyskoodin. Hyökkääjä voi tämän jälkeen käyttää laitteita hajautettujen palvelunestohyökkäysten suorittamiseen, eikä kaapattujen laitteiden omistajat ole välttämättä lainkaan tietoisia siitä, että heidän laitteensa ovat mukana hyökkäyksissä. (Mirkovic & Reiher, 2004). Kaapattujen laitteiden ryhmää kutsutaan usein myös bottiverkoksi ja yksittäisiä kaapattuja laitteita zombeiksi. Yhteen bottiverkkoon voi kuulua tuhansia zombeja. Hajautetut palvelunestohyökkäykset ovat hyvin vaarallisia muun muassa siitä syystä, että hyökkääjä saa käyttöönsä usean laitteen resurssit hyökkäyksen toteuttamiseen. Toisaalta useiden laitteiden

käyttö hankaloittaa myös varsinaisen hyökkääjän tunnistamista. (Zargar, Joshi & Tipper, 2013).

Motiivit palvelunestohyökkäysten tekemiseen vaihtelevat, mutta Zargarin, Joshin ja Tipperin (2013) mukaan ne voidaan jakaa viiteen kategoriaan: taloudellinen hyöty, kosto, ideologia, haaste hyökkääjälle, sekä kybersodankäynti. Hyökkäykset, joita motivoi taloudellinen hyöty, ovat yleensä suurin ongelma yrityksille. Tähän kategoriaan kuuluvat hyökkäykset ovat muihin verrattuna usein vaarallisempia ja vaikeimpia pysäyttää. Mikäli motiivina on kosto, on kyseessä usein turhautunut yksilö, joka haluaa kostaa kokemansa epäoikeudenmukaisuuden. Motiivi voi olla myös ideologinen. (Zargar, Joshi & Tipper, 2013). Esimerkiksi Viro siirsi vuonna 2007 Tallinnassa sijaitsevan Neuvostoliiton vapauttamista natseista kuvaavan muistomerkin vähemmän näkyvälle paikalle. Tämä johti erilaisiin hyökkäyksiin Viron kriittistä infrastruktuuria kohtaan. (Herzog, 2011). Joskus hyökkäyksen motiivina voi olla vain se, että hyökkääjä haluaa haastaa itseään ja näyttää, että hän pystyy suorittamaan palvelunestohyökkäyksen. Hyökkäyksen motiivina saattaa olla myös kybersodankäynti. Tällaiset hyökkääjät kuuluvat usein jonkin valtion organisaatioon, ja heillä on usein paljon taitoa, sekä paljon resursseja hyökkäysten suorittamiseen. Kohteena voi olla esimerkiksi toisen valtion kriittinen infrastruktuuri ja ne saattavat vaikuttaa hyvin merkittävästi valtion toimintakykyyn ja aiheuttaa suuria taloudellisia menetyksiä. (Zargar, Joshi & Tipper, 2013).

3.2 Palvelunestohyökkäykset sovelluskerroksella

Palvelunestohyökkäyksiä voidaan toteuttaa kaikilla OSI-, ja TCP/IP-mallin kerroksilla. Sovelluskerroksen palvelunestohyökkäyksissä on kuitenkin muutamia tekijöitä, jotka erottavat ne muista palvelunestohyökkäyksistä. Aikaisemmin palvelunestohyökkäykset toteutettiin lähinnä verkko- ja kuljetuskerroksella käyttäen suuria hyökkäysvolyymeja. Tämä johtui muun muassa siitä, että se oli tehokasta ja suhteellisen helppoa toteuttaa. Suojautumiskeinojen kehittyessä tällaisten hyökkäysten havaitseminen tuli kuitenkin helpommaksi, joka johti sovelluskerroksen palvelunestohyökkäysten yleistymiseen. (Gonzalez ym., 2014).

Sovelluskerroksen palvelunestohyökkäykset ovat usein vaikeampia havaita ja vaativat usein hyökkääjältä paljon pienempiä resursseja verrattuna esimerkiksi alempien kerroksien tulvahyökkäyksiin. Kun kohteena on esimerkiksi vain tietty sovelluskerroksen protokolla, ei hyökkääjällä ole tarvetta pyrkiä ylittämään kohteen koko tietoverkon kapasiteettia. Sovelluskerroksen palvelunestohyökkäykset pyrkivät usein myös matkimaan normaalia verkkoliikennettä ja hyödyntämään sovelluskerroksen protokollien legitiimejä ominaisuuksia. Nämä tekijät vaikeuttavat hyökkäyksen havaitsemista. (Mantas ym., 2015). Toisaalta esimerkiksi Gonzalez ym. (2014) ovat myös epäilleet tiettyjen sovelluskerroksen palvelunestohyökkäysten tehokkuutta käytännössä. Usein sovelluskerroksen

palvelunestohyökkäyksen toteuttaminen vaatii hyökkääjältä vain yhden laitteen hyökkäyksen toteuttamiseen (Tripathi & Hubballi, 2021). Tästä syystä tässäkin tutkielmassa hyökkäyksiin viitataan palvelunestohyökkäyksinä, eikä hajautettuina palvelunestohyökkäyksinä.

4 PROTOTOKOLLAT SOVELLUSKERROKSELLA

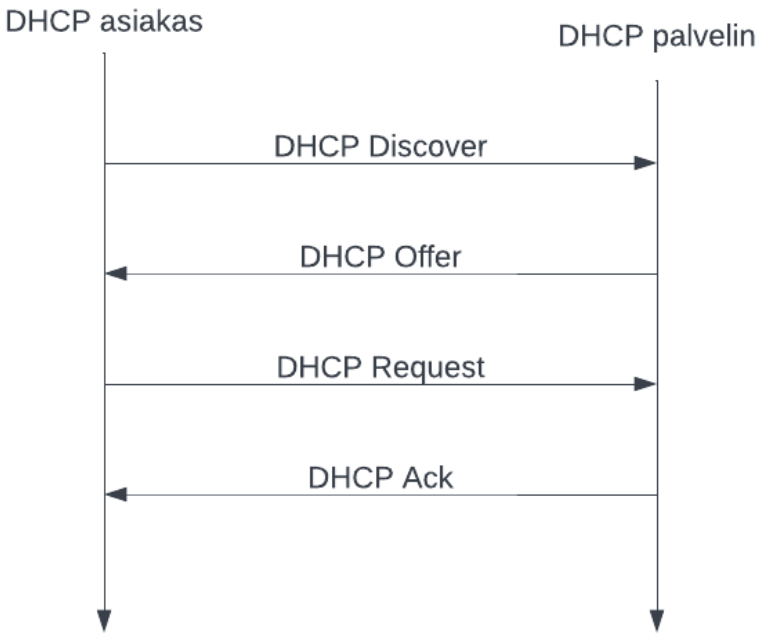
Sovelluskerroksella voi toimia erittäin monia erilaisia protokollia. Protokollien perustoiminnan ymmärtäminen on välttämätöntä, mikäli halutaan ymmärtää niitä hyödyntäviä hyökkäysmenetelmiä ja niiden torjuntakeinoja. Monista sovelluskerroksen protokollista tässä luvussa esitellään DHCP-protokolla, HTTP-protokolla, DNS-protokolla ja NTP-protokolla.

4.1 DHCP

DHCP on protokolla, joka mahdollistaa erilaisten verkkoparametrien automaattisen jakamisen laitteille IP-verkossa (Droms, 1997). Tällaisia asetuksia ovat muun muassa laitteen IP-osoite, yhdyskäytäväpalvelimen IP-osoite, aliverkon peite, sekä nimipalvelimen IP-osoite. (Yaibuates & Chaisricharoen, 2018). DHCP perustuu vanhempaan Bootstrap-protokollaan (jatkossa BOOTP). DHCP eroaa BOOTP-protokollasta muun muassa siinä, että se kykenee tarjoamaan laitteille verkko-osoitteita vain rajalliseksi ajaksi ja jakamaan niitä sitten uusille laitteille. Toinen merkittävä ero on DHCP:n kyky tarjota laitteille kaikki parametrit ja asetukset, jotka se tarvitsee operoidakseen IP-verkossa. (Droms, 1997). Mikäli DHCP ei ole käytössä, verkon ylläpitäjä joutuu konfiguroimaan verkossa sijaitsevat laitteet käsin. Tämä on aikaa vievää ja virheet saattavat johtaa ongelmiin verkon toiminnassa. (Yaibuates & Chaisricharoen, 2018).

Kun laite yhdistyy verkkoon, jossa DHCP on käytössä, DHCP-palvelin jakaa laitteelle tarvittavat verkkoparametrit. Viestit, jotka IP-osoitteen ja muiden verkkoparametrien allokointiprosessissa vaihdetaan ovat DHCP**D**ISCOVER, DHCP**O**FFER, DHCP**R**EQUEST ja DHCP**A**CK. Tätä viestien vaihtoa kutsutaan myös DORA-prosessiksi. (Tripathi & Hubballi, 2018, a).

1) Ensimmäisessä vaiheessa asiakas lähettää yleislähetyksenä DHCPDISCOVER viestin verkkoon tarkoituksenaan löytää siellä operoiva DHCP-palvelin (Droms, 1997).
2) Toisessa vaiheessa viestin vastaanottanut DHCP-palvelin tarkastaa sen jaettavissa olevat IP-osoitteet ja tarjoaa vapaata IP-osoitetta ja muita verkkoparametrejä, kuten käytettävissä olevaa DNS-palvelinta, sitä pyytävälle asiakkaalle. Ennen DHCPOFFER -viestin lähettämistä, DHCP-palvelin tarkastaa Internet Control message Protocol -protokollan (jatkossa ICMP) echo -viestin avulla, ettei tarjottu IP-osoite ole jo käytössä jollain muulla laitteella. (Droms, 1997).
3) Mikäli verkossa on useita DHCP-palvelimia, asiakas saattaa saada monta DHCPOFFER -viestiä. Asiakkaan tulee valita näistä palvelimista yksi ja lähettää yleislähetyksenä DHCPREQUEST viesti, jossa se yksilöi valitsemansa palvelimen ja tarjouksen verkkoparametreistä. (Droms, 1997)
4) Valittu DHCP-palvelin, joka vastaanottaa DHCPREQUEST -viestin, lähettää DHCPACK -viestin), jonka tulisi sisältää samat tiedot kuin aikaisemmassa DHCPOFFER -viestissä. Asiakkaan tulisi tässä vaiheessa vielä varmistaa esimerkiksi Address Resolution Protocol -protokollaa (jatkossa ARP) käyttämällä, että IP-osoite ei ole jo käytössä jollain muulla laitteella. Tämän jälkeen asiakkaan konfigurointi on valmis. (Droms, 1997). DORA-prosessi on esitetty alla olevassa kuviossa (kuvio 2).



KUVIO 2 DORA-prosessi

Ylempi viestienvaihto kuvaa DHCP:n toimintaa tilanteessa, jossa laite liittyy verkkoon ensimmäistä kertaa, eikä mitään ongelmia ilmene. Mikäli DHCP Request -viestin jälkeen DHCP-palvelin havaitsee, ettei se pystykään tarjoamaan haluttua IP-osoitetta, esimerkiksi jos IP-osoite on jo allokoitu jollekin muulle, tulisi sen vastata viestiin DHCPNAK -viestillä. Tämä aloittaa konfigurointi prosessin alusta. Toisaalta jos asiakas huomaa, että esimerkiksi IP-osoite on jo käytössä jollain muulla laitteella, tulisi sen vastata palvelimelle DHCPDECLINE -viestillä, joka myös aloittaa prosessin alusta. Muita mahdollisia viestejä ovat muun muassa DHCPRELEASE, jolla asiakas voi vapauttaa käytössään olevan IP-osoitteen, sekä DHCPINFROM, jolla asiakas voi pyytää muita verkkoparametrejä, mikäli sillä on jo käytössään IP-osoite. (Droms, 1997).

4.2 HTTP

HTTP on sovelluskerroksen protokolla, jota ohjelmistot käyttävät viestiäkseen internetissä. HTTP:llä on monia käyttökohteita, mutta sen tunnetuin käyttötarkoitus on kaksisuuntainen kommunikaatio palvelimien ja verkkoselainten välillä. (Gourley, ym. 2002).

Ensimmäinen HTTP:n versio, HTTP/0.9, julkaistiin jo vuonna 1991. HTTP/0.9 oli erittäin yksinkertainen protokolla, joka tuki ainoastaan Hypertext Markup Language -tiedostojen siirtämistä (jatkossa HTML) ja GET-metodia. (Berners-Lee, 1991).

HTTP:n seuraava versio, HTTP/1.0, määriteltiin RFC1945-dokumentissa vuonna 1996. HTTP/1.0 oli ensimmäinen HTTP:n versio, joka tuli laajasti käyttöön. Vanhempaan versioon verrattuna HTTP sai useita parannuksia kuten esimerkiksi HTTP-otsakkeet, tilakoodit ja uusia metodeja. Lisäksi HTTP/1.0 mahdollisti myös muun kuin HTML-sisällön siirtämisen. (Berners-Lee, Fielding & Frystyk, 1996).

HTTP/1.1 määriteltiin, vuonna 1997 ja muutamaa vuotta myöhemmin se sai vielä parannuksia samalla versionumerolla. Uudempi versio HTTP/1.1-protokollasta julkaistiin vuonna 1999 RFC2616-dokumentissa. (Fielding ym., 1999). HTTP/1.1 keskittyi korjaamaan virheitä HTTP:n arkkitehtuurissa ja tarjosi merkittäviä parannuksia HTTP:n suorituskykyyn. Se tarjosi myös tuen kehittyneemmille verkkosovelluksille. (Gourley ym., 2002, s.20).

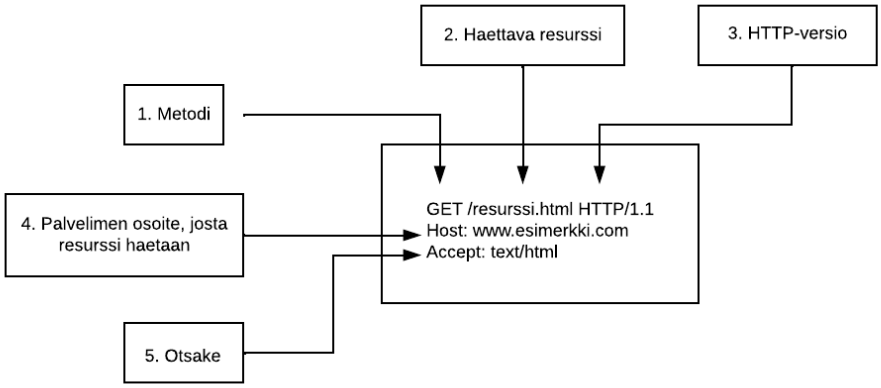
HTTP/2 määriteltiin RFC7540-dokumentissa vuonna 2015. HTTP/2 eroaa aikaisemmista versioista monilla tavoin, mutta merkittävin ero on erilaiset parannukset protokollan suorituskykyyn. HTTP/2 viestit jaetaan tyypeittäin itsenäisiin binäärikehyksiin, kuten otsake- ja datakehyksiin. Protokolla mahdollistaa näiden kehysten limittämisen ja priorisoinnin yhden yhteyden sisällä ja nopeuttaa näin viestien vaihtoa. (Belshe, Peon & Thomson, 2015). HTTP/3 on tällä hetkellä uusin esitelty versio HTTP-protokollasta. Se esiteltiin vuonna 2021 ja on

tällä hetkellä vielä kehityksessä. HTTP/3:n luvataan parantavan HTTP:n suorituskykyä ja turvallisuutta. TCP:n sijaan, se käyttää UDP-pohjaista QUIC-protokollaa tiedonsiirtoprotokollanaan. (Bishop, 2021)

Tällä hetkellä käytetyimmät HTTP protokollan versiot ovat HTTP/1.1 ja HTTP/2 (httparchive, 2021). Huolimatta siitä, että HTTP on saanut vuosien varrella erilaisia päivityksiä, ovat palvelunestohyökkäykset HTTP-protokollaa vastaan edelleen ajankohtainen ongelma. Itse asiassa Tripathin ja Hubballin (2018, b) mukaan HTTP/2 on jopa edeltäjäänsä haavoittuvampi ainakin hitaita hyökkäyksiä vastaan.

HTTP on pyyntö/vastaus -protokolla. HTTP:n perustoimintaa voidaankin kuvailla seuraavasti: asiakas lähettää palvelimelle pyynnön, joka sisältää käytettävän metodin, haettavan resurssin, käytettävän HTTP-versionumeron, isännän osoitteen (eng. host) josta resurssi haetaan, sekä Host -otsakkeen lisäksi mahdollisesti muita otsakkeita, jotka sisältävät tietoa pyynnöstä ja asiakkaasta itsestään. Lisäksi pyynnössä saattaa, metodista riippuen, olla myös sisältöä, joka halutaan palvelimelle lähettää. Palvelin vastaa pyyntöön tilakoodilla, joka voi esimerkiksi ilmoittaa, että pyyntö on onnistunut. Tämän lisäksi vastauksessa on myös HTTP-versionumero, sekä mahdollisesti otsakkeita ja sisältöä, jota asiakkaalle halutaan lähettää. (Fielding ym., 1999).

Alla olevassa kuviossa on esitettynä hyvin yksinkertainen esimerkki GET-metodia käyttävästä HTTP/1.1-pyynnöstä (kuvio 3). Kuvion elementtien selityksen löytyvät kuvion alapuolelta numerojärjestyksessä.



KUVIO 3 yksinkertainen GET-pyyntö

1. **Metodi** kertoo palvelimelle minkä toimenpiteen käyttäjä aikoo suorittaa. Kuviossa 3 käytetään GET-metodia, jonka avulla palvelimelta voidaan hakea erilaisia resursseja. Esimerkissä palvelimelta haetaan HTML-tiedosto nimeltään resurssi.html. Muita metodeja ovat muun muassa POST, PUT, DELETE ja HEAD. POST-metodilla palvelimelle voidaan lähettää sisältöä, kuten esimerkiksi viestejä jonkin verkkosivun kommenttipalstalle. PUT-metodilla palvelimelle voidaan luoda uutta sisältöä haluttuun osoitteeseen tai korvata osoitteessa jo sijaitseva sisältö. DELETE-metodi nimensä mukaisesti poistaa resurssin halutusta

osoitteesta. HEAD-metodi toimii GET-metodin tavoin, mutta palvelin palauttaa ainoastaan otsaketiedot, eikä varsinaista sisältöä. Tämä voi olla tarpeellista esimerkiksi siinä tapauksessa, kun resurssin tietoja halutaan tutkia ennen varsinaisen resurssin vastaanottamista. (Gourley ym., 2002, s. 53-58).

2. **Resurssilla** viitataan kaikenlaiseen palvelimen ylläpitämään verkkosisältöön. Resurssit voivat olla esimerkiksi yksittäisiä tiedostoja eri muodoissa, kuten kuvia, videoita tai HTML-dokumentteja. Kuviossa 3 palvelimelta haettava resurssi on resurssi -niminen HTML-tiedosto. (Gourley ym., 2002, s. 8).

3. **HTTP:n versionumerolla** asiakas kertoo palvelimelle mikä on korkein sen tukema HTTP-versio. Versionumero on ollut pakollinen osa pyyntöä HTTP/0.9 jälkeen. Jos asiakas tukee esimerkiksi korkeintaan HTTP/1.0 protokollaa, palvelin tietää olla käyttämättä ominaisuuksia, jotka on otettu käyttöön esimerkiksi vasta versiossa HTTP/1.1. (Gourley ym., 2002, s. 50).

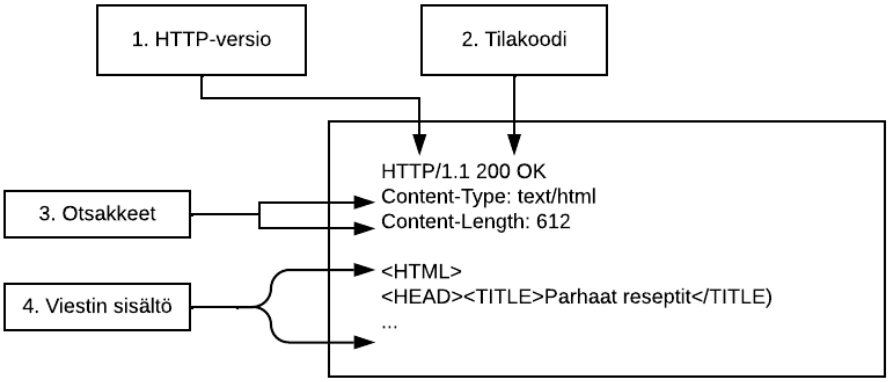
4. **Palvelimen osoite** määritetään Host-otsakkeen avulla. Host-otsake on virallisesti vaadittu HTTP/1.1 versiosta alkaen. Palvelimen osoite kertoo sen osoitteen, jossa pyynnön kohteena oleva resurssi sijaitsee. Palvelimen osoite voidaan antaa myös IP-osoitteena. Myös portti voidaan määrittää erikseen. Jos näin ei tehdä, käytetään vakioporttia eli HTTP:n tapauksessa porttia 80. Kuviossa 3 resurssi haetaan osoitteesta www.esimerkki.com. (Gourley ym., 2002, s.28-29).

5. **Otsakkeet** antavat pyynnöistä ja vastauksista lisäinformaatiota. Otsakkeita on monia erilaisia. Jotkut otsakkeet antavat tietoa pyynnöstä, jotkut vastauksesta ja jotkut voivat sijaita molemmissa. Kuviossa 2 käytetään otsakkeita Host ja Accept. Accept-otsake kertoo palvelimelle millaista sisältöä se ymmärtää. Kuviossa 3 Accept-otsakkeella on arvo text/html. Tämä tarkoittaa sitä, että asiakas hyväksyy vain HTML-sisältöä. Jos arvo olisi esimerkiksi text/html, image/jpeg, hyväksyisi asiakas HTML:n lisäksi myös JPEG-muodossa olevat kuvat. Muita yleisiä pyynnöissä käytettäviä otsakkeita ovat muun muassa Accept-Language, Accept-Encoding ja If-Modified-Since. Accept-Language -otsake kertoo, mitä kieltä asiakas haluaa vastaanottaa. Esimerkiksi Accept-Language: fi kertoo, että asiakas haluaa vastaanottaa suomenkielistä sisältöä. Accept-Encoding -otsake kertoo minkälaista sisällön enkoodausta asiakas ymmärtää. (Gourley ym., 2002, s.69). Esimerkiksi Accept-Encoding: deflate tarkoittaa, että asiakas ymmärtää sisältöä, joka on kompressoitu deflate -menetelmää käyttämällä (Deutsch, 1996). If-Modified-Since-otsakkeen avulla asiakas voi tarkastaa, onko haettavaan resurssiin tehty muutoksia otsakkeessa annetun päivämäärän jälkeen. Resurssi haetaan vain

siinä tapauksessa, että siihen on tehty päivämäärän jälkeen muutoksia. Esimerkiksi If-Modified-Since: Fri, 20, oct 2020 12:00:00 tarkoittaa, että resurssi halutaan vain siinä tapauksessa, että muokkaus on tapahtunut perjantain, 20.10.2020 klo 12.00 jälkeen. (Gourley ym., 2002, s. 70).

Pyynnöissä voi, käytetystä metodista riippuen, olla myös sisältöä, kuten tekstiä, kuvia ja videoita. Pyynnöissä, joissa käytetään esimerkiksi GET-, ja HEAD-metodeja, varsinaista sisältöä ei lähetetä. Toisaalta esimerkiksi POST-metodia käytettäessä jotain sisältöä yleensä on, kuten käyttäjän viesti, jonka hän lähettää esimerkiksi kommenttipalstalle. (Gourley ym., 2002, s. 48-49).

Alla olevassa kuviossa on esitetty vastaus GET-pyyntöön (kuvio 4). Vastauksen elementtien selitykset löytyvät kuvion alta numerojärjestyksessä.



KUVIO 4 Vastaus yksinkertaiseen GET-pyyntöön

1. **HTTP-versio** kertoo mitä HTTP:n versiota palvelin käyttää vastauksessaan.

2. **Tilakoodit** esiteltiin HTTP:n versiossa HTTP/1.0. Tilakoodit ovat kolmen kokonaisluvun yhdistelmiä, jotka kertovat asiakkaalle mitä pyynnölle tapahtui. Numerolla 1 alkavat tilakoodit esiteltiin vasta HTTP:n versiossa HTTP/1.1 ja ne antavat väliaikatietoa. Esimerkiksi tilakoodi 100 Continue kertoo, että palvelin on vastaanottanut osan pyynnöstä ja asiakkaan tulisi jatkaa pyynnön lähettämistä. Numerolla 2 alkavat tilakoodit kertovat pyynnön onnistumisesta. Esimerkiksi kuviossa 4 palvelin vastaa tilakoodilla 200 OK. Tämä tarkoittaa sitä, että asiakkaan pyytämä resurssi on lähetetty asiakkaalle onnistuneesti. Numerolla 3 alkavat tilakoodit kertovat asiakkaalle, että pyynnön toteuttaminen vaatii asiakkaalta lisätoimia. Esimerkiksi tilakoodi 301 Moved Permanently kertoo, että asiakkaan pyytämän resurssin osoite on muuttunut pysyvästi. Palvelin kertoo vastauksessaan myös resurssin uuden osoitteen. Numerolla 4 alkavat tilakoodit kertovat asiakkaan virheestä.

Esimerkiksi tilakoodi 400 Bad Request kertoo, että palvelin ei ymmärrä asiakkaan pyyntöä johtuen esimerkiksi virheellisestä syntaksista. Numerolla 5 alkavat tilakoodit kertovat palvelimen virheestä. Esimerkiksi tilakoodi 501 Not Implemented kertoo, että palvelin ei tue toimintoja, jotka vaaditaan pyynnön toteuttamiseen. Tämä voi tulle kyseeseen esimerkiksi siinä tapauksessa, että palvelin ei tue asiakkaan käyttämää metodia. (Fielding ym., 1999).

3. Kuviossa 4 käytetään **otsakkeita** Content-Type ja Content-Length. Content-Type -otsake kertoo minkä tyyppistä sisältöä vastaus sisältää. Kuviossa 4 otsakkeen arvo on text/html, eli vastaus sisältää HTML-sisältöä. Content-Lenght kertoo lähetettävän sisällön pituuden tavuina. Kuviossa 4 vastauksen kokonaispituus on 612 tavua. Kuviossa vastauksesta on näkyvillä vain pieni osa. Muita vastauksissa käytettäviä otsakkeita ovat muun muassa Server ja Retry-After. Server -otsake kertoo lisätietoja vastauksen lähettäneen palvelimen palvelinohjelmistosta. Retry-After -otsake kertoo kuinka kauan käyttäjän tulisi odottaa ennen uutta pyyntöä. Mikäli palvelin vastaan pyyntöön esimerkiksi tilakoodilla 503 Service Unavailable, Retry-After -otsake voi kertoa milloin palvelu on taas käytettävissä. (Fielding ym., 1999).

4. **Viestin sisältö** voi olla käytännössä mitä tahansa dataa, jota halutaan toiselle osapuolelle lähettää, kuten kuvia, videoita ja tekstiä. Kuviossa 4 vastaus sisältää HTML-sisältöä. Koko sisältö ei ole kuviossa näkyvissä, mutta siitä voimme päätellä, että asiakkaalle lähetetään HTML-tiedosto, jonka otsikkona on ”Parhaat reseptit”. (Gourley ym., 2002, s. 14).

4.3 DNS

DNS on nimipalvelujärjestelmä, jonka ensisijainen tarkoitus on yhdistää verkkotunnukset ja niitä vastaavat IP-osoitteet toisiinsa. Kabelovan ja Dostalekin (2006) mukaan DNS:sää voitaisiin kutsua maailmanlaajuiseksi hajautetuksi tietokannaksi. DNS-protokollan pääasiallinen tehtävä on hakea DNS-tietokannasta haluttua tietoa, kuten tietoa verkkotunnuksista ja niihin liitetyistä IP-osoitteista. Tietoverkoissa sijaitsevat laitteet keskustelevat toistensa kanssa hyödyntäen IP-osoitteita. IP-osoitteet ovat kuitenkin ihmisille varsin epäkäytännöllisiä, koska ne ovat usein hankalia muistaa. Mikäli käyttäjä haluaa käyttää esimerkiksi Googlen hakukonetta, voidaan web-selaimen osoitekenttään kirjoittaa www.google.com. Mikäli DNS-palvelua ei ole käytössä, osoitteeksi tulisi kirjoittaa hankalammin muistettava IP-osoite, eli tässä tapauksessa 142.250.74.78. (Kabelova & Dostalek, 2006, s. 20).

Verkkotunnukset ovat merkkijonoja, jotka on eroteltu tosistaan pisteillä. DNS on hiearkinen tietokanta ja pisteet verkkotunnuksessa erottavat hierarkian eri tasot toisistaan. Verkkotunnus voisi olla esimerkiksi bob.esimerkki.fi.. Verkkotunnuksen oikeassa reunassa oleva piste voidaan useimmiten jättää pois, mutta se tarkoittaa juuriverkkotunnusta. Juuriverkkotunnus viittaa ylätason verkkotunnukseen. Esimerkissä se on fi., joka on Suomen maatunnus. Muita ylätason verkkotunnuksia ovat muun muassa com., net., org. ja us.. Ylätason verkkotunnukset viittaavat alemman tason verkkotunnuksiin. Esimerkissä toisen asteen verkkotunnus on esimerkki.fi. ja kolmannes asteen verkkotunnus bob.esimerkki.fi.. Eniten vasemmalla oleva merkkijono, eli esimerkissä sana bob, kuvaa isännän nimeä. (Rooney, 2011).

Nimenmuunnos rekursiivista nimipalvelinta käyttämällä tapahtuu yksinkertaistetusti seuraavalla tavalla:

1. Resolveri, eli verkkotunnusten muuttamisesta IP-osoitteiksi vastaava ohjelmakoodi, tekee kyselyn nimipalvelimelle. Resolveri tietää nimipalvelimen IP-osoitteen joko manuaalisen konfiguroinnin tai DHCP:n käytön kautta. Tässä esimerkissä verkkotunnus on bob.esimerkki.fi.. Nimipalvelin tarkastaa löytyykö tarvittava tieto sen välimuistista tai tiedostoista. Mikäli ei löydy, nimipalvelin ottaa yhteyttä muihin nimipalvelimiin.

2. Nimipalvelin ottaa ensimmäisenä yhteyttä juurinimipalvelimeen, joka vastaan niiden nimipalvelimien IP-osoitteilla, jotka hallinnoivat haetun verkkotunnuksen ylätason verkkotunnuksia. Esimerkin tapauksessa palvelin vastaa .fi.-maatunnuksesta vastaavien palvelimien IP-osoitteilla.

3. Nimipalvelin ottaa yhteyttä ylätason verkkotunnuksista vastaaviin autoritäärisiin nimipalvelimiin. Esimerkissä nimipalvelin, ottaa yhteyttä .fi.-maatunnuksista vastaaviin nimipalvelimiin, jotka vastaavat niiden nimipalvelimien IP-osoitteilla, jotka hallinnoivat verkkotunnusta esimerkki.fi..

4. Nimipalvelin ottaa yhteyttä autoritääriseen nimipalvelimeen, joka hallinnoi esimerkki.fi. verkkotunnusta. Mikäli osoite on olemassa, tämä palvelin vastaa lopulta alkuperäiseen kysymykseen, eli mikä on verkkotunnuksen bob.esimerkki.fi., IP-osoite. Yleensä asiakkaan nimipalvelin myös tallentaa kyselyn tuloksen välimuistiinsa. (Kabelova & Dostalek, 2006, s. 41-42).

DNS:n sisältö kulkee tietueiden välityksellä. Tietue sisältää ainakin seuraavat neljä kenttää: 1. Type, 2. Name, 3. Class, ja 4. TTL eli Time-To-Live.

Type-elementti on 16 bittinen arvo, joka viittaa tietueen tyyppiin. Tyyppejä on käytössä useita kymmeniä. Tyypit merkitään yleensä kirjaimina ja yleisiä tyyppejä ovat muun muassa A, CNAME, MX ja NS. A-tyyppisiä tietueita käytetään isännän, kuten esimerkiksi web-palvelimen, IP-osoitteen tunnistamisessa. CNAME-tyyppinen tietue mahdollistaa sen, että voidaan käyttää verkko-osoitteita, joilla ei välttämättä ole suoranaisesti omaa IP-osoitetta. Toisin sanoen eri osoitteet voidaan saada osoittamaan johonkin toiseen verkko-osoitteeseen. Esimerkkinä tästä voidaan pitää tilannetta, jossa käytössä on verkko-osoite esimerkki.fi, joka käyttää palvelimen IP-osoitetta 172.16.44.1. Jollekin toiselle verkko-osoitteelle voidaan muodostaa CNAME tietue, joka yhdistetään esimerkki.fi verkko-osoitteeseen. Mikäli tietue luodaan esimerkiksi www.esimerkki.fi osoitteelle, osaa DNS-palvelu yhdistää käyttäjän esimerkki.fi osoitteen kautta samaan 172.16.44.1 IP-osoitteeseen.

MX-tyyppisiä tietueita käytetään sähköpostipalvelimen määrittämiseen, jota käytetään verkkotunnukselle tarkoitettujen sähköpostien vastaanottamiseen. NS-tyyppisiä tietueita käytetään identifioimaan verkkotunnuksen autoritäärinen nimipalvelin. (Mockapetris, 1987).

Name-elementti kertoo kokonaisen verkkotunnuksen, josta tietue löytyy. Class-elementti kertoo mihin protokollaperheeseen tietue kuuluu. Kun kyseessä on tietue, joka sisältää tavallisia tietoja internetin palvelimista, IP-osoitteista ja verkkotunnuksista, luokka on IN, joka on lyhenne sanasta Internet.

TTL-elementti on 32 bittinen kokonaisluku, joka kertoo sekunneissa, kuinka pitkään tietuetta voidaan säilyttää välimuistissa, ennen kuin se tulee hylätä. (Mockapetris, 1987).

4.4 NTP

NTP on sovelluskerroksen protokolla, jota käytetään tietokoneiden kellojen synkronointiin internetissä (Mills ym., 2010). Protokolla on jo vuosikymmeniä vanha ja siitä on julkaistu vuosien varrella monia uusia versioita. Uusin versio on versio 4 (NTPv4), joka esiteltiin dokumentissa RFC 5905 vuonna 2010.

NTP-protokolla voi toimia pääasiallisesti kolmella eri tavalla ja ajan synkronointiin voidaan käyttää myös näiden eri tapojen yhdistelmiä. NTP voi toimia asiakas/palvelin -periaatteella, symmetrisesti, tai käyttäen yleislähetystä. Asiakas/palvelin -variaatiossa asiakas tekee pyynnön palvelimelle, joka lähettää aikatietoja asiakkaalle. Asiakas/palvelin -tilassa asiakkaat eivät jaa aikatietoja muille laitteille. Mikäli NTP toimii symmetrisesti, NTP-verkossa toimivat laitteet voivat toimia niin asiakkaana, kuin myös palvelimena. Symmetrisessä toimintatavassa NTP-verkossa toimivat laitteet toimivat vertaisinaan ja voivat sekä lähettää, että vastaanottaa aikatietoja muiden vertaistensa kanssa. Tämä voi olla käytännöllistä esimerkiksi siinä tapauksessa, että verkossa toimiva laite menettää aikatietoja. Tällöin vertaiset voivat lähettää laitteelle kadonneet tiedot. (Mills, 2011, s.52). Yleislähetystä käyttävä tapa on tarkoitettu tilanteisiin, jossa palvelimella on

paljon asiakkaita. Yleislähetyksessä palvelin lähettää verkkoon viestejä esimerkiksi minuutin välein ja useat asiakkaat voivat vastaanottaa näitä viestejä.

NTP on hierarkkinen järjestelmä, jossa verkossa toimivat laitteet jaetaan kerroksiin. Kerrosta kutsutaan usein termillä ”stratum”. Aika-arvot siirtyvät ensimmäiseltä kerrokselta toisen kerroksen asiakkaille, jotka voivat myös toimia palvelimina seuraavan kerroksen asiakkaille. Ensimmäisen kerroksen palvelimet saavat aika-arvonsa suoraan jostain ulkopuolisesta lähteestä, kuten satelliittipalvelusta. Ensimmäisen kerroksen palvelimet voivat olla myös yhteydessä toisiinsa. Toisen kerroksen palvelimet saavat aika-arvonsa ensimmäisen kerroksen palvelimilta ja mahdollisesti myös muilta saman kerroksen palvelimilta. Kolmannen kerroksen palvelimet saavat aikatietoja toisen kerroksen palvelimilta, sekä mahdollisesti myös toisiltaan. Kolmannen kerroksen palvelimet voivat toimia palvelimina alemman kerroksen asiakkaille ja niin edelleen. (Mills, 2011, s.33).

5 PROTOKOLLAKOHTAISET HYÖKKÄYKSET

Tässä kappaleessa esitellään erilaisia sovelluskerroksen protokollia vastaan kehitettyjä hyökkäyksiä. Eri protokollia toteutetaan käytännön toteutuksissa eri tavoin, eivätkä kaikki ohjelmistot ole yhtä alttiita kaikille hyökkäyksille. Kappale on jaettu alalukuihin eri protokollien mukaan ja ne on vielä jaettu alalukuihin joissa esitellään erilaisia hyökkäyksiä ja niihin ehdotettuja puolustautumiskeinoja. Selkeyden vuoksi hyökkäyksien niminä käytetään niiden englanninkielisiä termejä.

5.1 DHCP-hyökkäykset

5.1.1 DCHP Starvation

DHCP Starvation on palvelunestohyökkäys, jonka kohteena on DHCP-palvelin. Hyökkäyksestä on kehitetty erilaisia versioita, mutta tavallisesti hyökkääjä lähettää palvelimelle useita DCHPDISCOVER-pyyntöjä käyttäen väärennettyjä Meidum Access Control -osoitteita (jatkossa MAC) tavoitteenaan varata kaikki palvelimen tarjottavissa olevat IP-osoitteet, ja estää näin oikeutettuja käyttäjiä vastaanottamasta IP-osoitetta. MAC-osoite tarkoittaa verkkokortin fyysistä osoitetta, jota ei laitetasolla voida muuttaa. MAC-osoite on kuitenkin mahdollista väärentää ohjelmallisesti. (Mukhtar, Salah & Iraqi, 2012). Tavanomaiset DCHP starvation -hyökkäykset voivat olla vaarallisia langallisissa verkoissa, mutta langattomissa verkoissa niiden toteuttaminen voi Tripathin ja Hubballin (2017) mukaan olla hyvin haastavaa.

5.1.2 Induced DHCP Starvation

Induced DHCP Starvation on palvelunestohyökkäys, joka on melko samanlainen perinteisen DHCP starvation -hyökkäyksen kanssa. Myös sen tarkoituksena on estää käyttäjiä saamasta IP-osoitetta DHCP-palvelimelta. Induced DHCP

Starvation hyväksikäyttää kuitenkin IP-osoitteen allokointiprosessin eri vaihetta. Tavanomaisissa DHCP Starvation -hyökkäyksissä hyökkääjä lähettää palvelimelle useita DHCPDISCOVER-viestejä käyttäen väärennettyjä MAC-osoitteita. Tässä hyökkäyksessä sen sijaan hyödynnetään sitä, että asiakkaan tulee DHCPACK-viestin vastaanottamisen jälkeen varmistaa, ettei tarjottu IP-osoite ole käytössä kellään muulla. Tähän käytetään yleensä ARP-protokollaa. Koska aikaisemman vaiheen DHCPREQUEST-viesti on lähetetty yleislähetyksenä, hyökkääjällä on tässä vaiheessa jo tieto käyttäjän MAC-osoitteesta. Käyttäjän IP-osoite on 0.0.0.0, koska IP-osoitteen allokointi ei ole vielä valmis. Kun käyttäjä lähettää ARP-pyynnön tiedustellakseen, onko IP-osoite kellään käytössä, hyökkääjä vastaa viestiin ARP-vastauksella, joka väittää, että IP-osoite on jo käytössä. Kun käyttäjä vastaanottaa ARP-vastauksen, se lähettää DHCP-palvelimelle DHCPDECLINE-viestin. DHCP-palvelin merkitsee IP-osoitteen pois käytöstä määritellyksi ajaksi. Hyökkäystä jatketaan, kunnes DHCP-palvelin ei voi enää tarjota uusia IP-osoitteita. (Hubballi & Tripathi, 2017).

Induced DHCP Starvation -hyökkäystä voidaan pitää perinteisiä DHCP Starvation -hyökkäyksiä vaarallisempana, koska se on mahdollista tehdä helposti myös langattomissa verkoissa. Se vaatii myös vähemmän lähetettäviä viestejä. Induced DHCP Starvation on myös verrattain uusi hyökkäys. joten sen torjumiseksi ei olla ehdotettu niin paljon puolustuskeinoja perinteisiin DHCP Starvation -hyökkäyksiin verrattuna. (Hubballi & Tripathi, 2017).

5.1.3 Suojautuminen DCHP-hyökkäyksiltä

Perinteinen DHCP Starvation -hyökkäys on ollut tiedossa jo melko pitkään, ja sen torjumiseksi on ehdotettu useita keinoja. Yksi mahdollinen suojautumiskeino on käyttää erilaisia autentikointimenetelmiä. Esimerkiksi Demerjian ja Serhrouchni (2004) ovat ehdottaneet digitaalisten varmenteiden käyttöä DHCP-prosessin suojaamiseksi. Muita autentikointiin perustuvia keinoja ovat muun muassa käyttäjien tunnistaminen hyödyntämällä kerberospalvelinta, tai käyttämällä jotain tunnistetta, kuten salasanaa. (Demerijan & Serhrouchni, 2004). Tripathin ja Hubballin (2021) mukaan tällaiset suojautumiskeinot voivat olla tehokkaita, mutta muun muassa niiden monimutkaisuuden vuoksi niitä harvoin nähdään käytännön toteutuksissa.

DHCP Starvation -hyökkäyksiä vastaan voidaan suojautua myös erilaisilla kytkimien suojausominaisuuksilla. Tällaisia ominaisuuksia ovat esimerkiksi porttien suojaus. Porttien suojaamisessa tarkoituksena on rajoittaa MAC-osoitteita porttia kohden ja estää näin MAC-osoitteiden väärentäminen ja sitä kautta DHCP starvation -hyökkäys. Hyökkääjä voi kuitenkin DHCPDISCOVER viesteissään käyttää Ethernet -kehyksen otsakkeissa oikeaa MAC-osoitettaan ja DHCPDISCOVER-viestin CHADDR-otsakkeessa väärennettyä MAC-osoitetta. Koska porttien suojausominaisuudet eivät pysty lukemaan otsaketietoja sovelluskerroksella, ei kyseinen ominaisuus anna riittävää suojaa hyökkäystä vastaan. (Hubballi & Tripathi, 2017).

Toinen kytkimissä usein käytetty suojausominaisuus on englanninkieliseltä nimeltään DHCP Snooping. Tämä ominaisuus kykenee havaitsemaan, mikäli Ethernet -kehyksen otsakkeissa ja DHCPDISCOVER-viestin CHADDR-otsakkeessa käytetään eriäviä MAC-osoitteita. Tämä ominaisuus ei kuitenkaan itsessään rajoita käytettävien MAC-osoitteiden määrää, eli hyökkäys voidaan edelleen toteuttaa käyttämällä molemmissa otsakkeissa samaa MAC-osoitetta. (Hubballi & Tripathi, 2017). Tätä ominaisuutta voidaan kuitenkin käyttää yhdessä porttien suojauksen kanssa, jolloin ne sopivat yhdessä varsin hyvin perusmuotoisten DHCP starvation -hyökkäysten hallintaan.

DHCP starvation -hyökkäyksiltä puolustautumiseen voidaan käyttää myös DHCP-välitysagenttia), joka välittää DCHP-viestejä asiakkaan ja palvelimen välillä. Välitysagentti ei välitä DHCP-viestejä sellaisenaan, vaan voi lisätä viestiin lisäinformaatiota, jota kutsutaan termeillä ”DHCP relay agent information option” tai ”DHCP option 82”. (Patrick, 2001). Välitysagentti lisää viestiin porttinumeron ja välitysagentin, kuten kytkimen, tunnisteen, ennen kuin se lähettää viestin DHCP-palvelimelle. Tätä informaatiota käyttämällä palvelin voi varmistaa, ettei se tarjoa asiakkaalle sallittua määrää enempää IP-osoitteita. Tämä onkin varsin tehokas tapa puolustautua perusmuotoisia DHCP starvation -hyökkäyksiä vastaan. (Mukhtar, salah & Iraqi, 2012).

DHCP Starvation -hyökkäyksiä vastaan on ehdotettu myös erilaisia DCHP-liikenteen seurantaan liittyviä puolustuskeinoja. Esimerkiksi Oconnorin (2010) mukaan voitaisiin seurata, kuinka paljon DHCP-viestejä lähetetään tiettynä aikana. Mikäli viestien määrä ylittää aikaisemmin määritellyn rajan, ilmoitetaan siitä esimerkiksi verkon valvojalle. Tavanomaisissa DHCP Starvation -hyökkäyksissä viestejä lähetetään usein suuri määrä, joten tämä keino voi olla toimiva ainakin niiden havaitsemiseksi. (OConnor, 2010).

Koska Induced DHCP Starvation toimii hieman eri tavalla verrattuna tavanomaisiin DHCP Starvation -hyökkäyksiin, ei sitä vastaan voida käyttää täysin samoja puolustuskeinoja. Erilaiset autentikointiin perustuvat menetelmät voisivat olla kuitenkin toimivia myös tätä hyökkäystä vastaan. (Tripathi & Hubballi, 2021).

Aiemmin mainitut kytkimissä käytetyt suojausominaisuudet eli porttien suojaus ja DHCP Snooping eivät kuitenkaan ole tehokkaita Induced DHCP Starvation -hyökkäystä vastaan, koska tässä hyökkäyksessä ARP-viestejä ei välttämättä lähetetä kytkimen kautta. Se ei kuitenkaan ole edes merkityksellistä, koska hyökkääjä joutuu joka tapauksessa käyttämään vain yhtä MAC-osoitetta, eikä esimerkiksi otsaketietojen muuttamiselle ole tarvetta.

Myöskään DHCP-välitysagentin käyttö ei estä tämän hyökkäyksen suorittamista, koska hyökkäys ei vaadi yhdenkään DHCP-viestin lähettämistä. Tripathin ja Hubballin (2017) mukaan on mahdollista, että välitysagentin käytön seurauksena DHCP-palvelin jopa estää oikeutettuja käyttäjiä saamasta IP-osoitetta, koska he joutuvat hyökkäyksen vuoksi pyytämään uutta IP-osoitetta toistuvasti. (Tripathi & Hubballin, 2017).

DHCP-liikenteen seuranta saattaa toimia Induced DHCP Starvation -hyökkäyksen havaitsemiseen, mutta perinteisiin DHCP Starvation -hyökkäyksiin

verrattuna viestejä lähetetään paljon vähemmän. Toisaalta viestejä lähettävät juuri hyökkäyksen uhrit, eikä hyökkääjä lähetä hyökkäyksen aikana välttämättä yhtäkään DHCP-viestiä. Tästä syystä tämä keino ei varsinaisesti estä hyökkäyksen toteuttamista, vaikka se voikin kertoa, että hyökkäys on käynnissä. (Tripathi & Hubballi, 2017). Tripathi ja Hubballi (2018, a) ovat ehdottaneet myös koneoppimiseen perustuvaa torjuntakeinoa tätä hyökkäystä vastaan ja saivat tutkimuksessaan lupaavia tuloksia (Tripathi & Hubballi, 2018, a). Koneoppimiseen perustuvat toteutukset eivät kuitenkaan ainakaan vielä ole yleisessä käytössä.

5.2 HTTP-hyökkäykset

HTTP:tä vastaan on esitetty lukuisia erilaisia hyökkäyksiä. Tässä luvussa esitellään muutamia HTTP:tä vastaan kehitettyjä hyökkäyksiä, joita kutsutaan hitaiksi hyökkäyksiksi. Monissa perinteisissä hyökkäyksissä hyökkääjä pyrkii vaikuttamaan kohteensa toimintaan lähettämällä palvelimelle esimerkiksi suuren määrän HTTP-pyyntöjä. Hitaat hyökkäykset sen sijaan pyrkivät hyväksikäyttämään HTTP-protokollan ominaisuuksia ja ne ovat usein mahdollista suorittaa hyvin pienillä resursseilla. Tästä syystä ne voivat olla tehokkaita suoritettuna vain yhdellä laitteella, eikä hyökkäystä tarvitse välttämättä toteuttaa hajautettuna palvelunestohyökkäyksenä.

Hitaiden HTTP-hyökkäysten tehokkuus riippuu paljon myös kohteena olevan palvelimen palvelinohjelmistosta. Arkkitehtuuristen erojensa takia esimerkiksi Apache on usein haavoittuvampi hitaille HTTP-hyökkäyksille verrattuna esimerkiksi Nginx-ohjelmistoon. (Tripathi, Hubballi & Singh, 2016). Tässä työssä ei kuitenkaan enempää vertailla hyökkäysten vaikutusta erilaisten palvelinohjelmistojen välillä.

5.2.1 Slow Header -hyökkäys

HTTP-protokolla vaatii, että GET-otsakkeiden tulee olla kokonaisuudessaan vastaanotettu, ennen kuin ne voidaan prosessoida. HTTP on suunniteltu käytettäväksi myös hitaammilla internet-yhteyksillä. Mikäli otsakkeista vastaanotetaan vain osa, protokolla pitää yhteyden auki tietyn ajan odottaen loppujen otsaketietojen vastaanottamista. RFC 7230:n mukaan tyhjä rivi tulee lähettää merkkinä siitä, että otsaketiedot loppuvat. (Fielding & Reschke, 2014).

Slow Header -hyökkäys hyödyntää yllä olevaa, protokollan suunnitellussa ilmenevää, haavoittuvuutta. Hyökkääjä ei lähetä otsaketietoja kokonaan tai hyökkääjä ei lähetä tyhjää riviä merkiksi otsakkeiden loppumisesta. Sen sijaan

hyökkääjä lähettää väärennettyjä otsaketietoja sellaisella nopeudella, että palvelin pitää yhteyden auki uusien otsaketietojen vastaanottamiseksi, mutta ei kuitenkaan ehdi sulkea yhteyttä. Mikäli hyökkääjä onnistuu luomaan tarpeeksi monta yhteyttä palvelimen kanssa, ei palvelin enää lopulta pysty palvelemaan sen oikeutettuja asiakkaita. (Tripathi, Hubballi & Singh, 2016).

5.2.2 Slow Message Body -hyökkäys

Mikäli HTTP-pyynnössä on sisältöä, tulisi pyynnössä muutamaa poikkeusta lukuun ottamatta käyttää Content-Length -otsaketta. Content-Length -otsake kertoo palvelimelle, kuinka pitkä viestin sisältö on tavuina. Mikäli palvelin ei ole vastaanottanut otsaketiedon mukaista sisältöä, se pitää yhteyden auki odottaen koko sisällön vastaanottamista. (Fielding & Reschke, 2014).

Slow Message Body -hyökkäyksessä hyökkääjä ilmoittaa Content-Lenght-otsakekentässä viestin sisällön pituudeksi suuremman arvon, kun se on todellisuudessa lähettämässä. Hyökkääjä voi esimerkiksi lähettää palvelimelle POST-pyynnön ja antaa Content-Lenght-kentän arvoksi 400. Hyökkääjä voi kuitenkin tämän jälkeen lähettää viestin, jonka sisältö on todellisuudessa vain 200 tavua pitkä. Palvelin olettaa, että 200 tavua on vielä vastaanottamatta ja pitää yhteyden auki. Mikäli hyökkääjä avaa tarpeeksi monta yhteyttä palvelimen kanssa ja lähettää vastaavanlaisia viestejä, palvelin ei välttämättä kykene enää palvelemaan sen oikeutettuja asiakkaita. (Tripathi, Hubballi & Singh, 2016).

5.2.3 Slow Read -Hyökkäys

Aikaisemmin esiteltyjen hyökkäysten tapaan myös Slow Read -hyökkäyksen tarkoituksena on pitää palvelimen kanssa yhteyksiä auki ja vaikuttaa näin sen toimintaan. Aikaisemmista hyökkäyksistä poiketen, Slow Read -hyökkäyksessä palvelimelle lähetetään kunnollinen HTTP-pyyntö, mutta palvelimen vastaus luetaan hitaasti. (Shekyan, 2012).

TCP-protokollassa vastaanottajan tulisi päivittää lähettäjälle vastaanottoikkunaa. Tämä on TCP:n vuonvalvonnan ominaisuus ja sen tarkoitus on estää lähettäjää lähettämästä dataa nopeammin, kuin vastaanottaja pystyy vastaanottamaan. Lähettäjä pystyy lähettämään dataa vain vastaanottoikkunan koon verran ja tämän jälkeen sen tulee odottaa vastaanottajalta acknowledgment-kuittausta ja vastaanottoikkunan päivittämistä. (Postel, 1981).

Slow Read -hyökkäyksessä hyökkääjä tekee palvelimelle HTTP-pyynnön, jossa se pyytää resurssia, joka on suurempi kuin lähettäjän lähetyspuskurin koko. Hyökkääjä lähettää lisäksi TCP-paketin, joka mainostaa lähettäjän lähetyspuskuria pienempää vastaanottoikkunaa. Tämän seurauksena hyökkääjä vastaanottaa dataa erittäin hitaasti (tai ei lainkaan, jos vastaanottoikkunan kooksi mainostetaan 0). Mikäli hyökkääjä avaa tarpeeksi monta yhteyttä palvelimen kanssa, ei se välttämättä pysty enää vastaamaan sen oikeutetuille asiakkaille. (Shekyan, 2012).

5.2.4 HTTP PRAGMA -hyökkäys

PRAGMA -otsakekentän avulla palvelimelta voidaan pyytää uutta versiota jo aikaisemmin pyydetystä resurssista. Käytännössä hyökkäys voidaan toteuttaa siten, että hyökkääjä lähettää palvelimelle viestin, odottaa kunnes palvelin on katkaisemassa yhteyden ja lähettää sitten PRAGMA-viestin. Tämän seurauksena palvelin ei katkaisekaan yhtyettä, joten hyökkääjä saa näin kulutettua kauemmin palvelimen resursseja. Mikäli hyökkääjä avaa monia yhteyksiä palvelimen kanssa ja saa pidettyä niitä auki, ei palvelin enää välttämättä pysty palvelemaan sen oikeutettuja asiakkaita. (Dantas, Nigam & Fonseca, 2014).

5.2.5 Suojautuminen hitailta HTTP-hyökkäyksiltä

Hitaita HTTP-hyökkäyksiä vastaan on esitetty useita erilaisia puolustautumiskeinoja. Tripathi, Hubballi ja Singh (2016) ovat esitelleet Hellingerin etäisyyttä hyödyntävän anomalian tunnistusmenetelmän, jossa he vertasivat normaalia HTTP-liikennettä hyökkäyksen aikana esiintyvään liikenteeseen. Tutkimuksessa esitelty menetelmä pystyi havaitsemaan Slow Header -hyökkäyksen ja Slow Message Body -hyökkäyksen tehokkaasti. Ehdotettu menetelmä kuitenkin toimii vain hyökkäysten havaitsemisessa, eikä sinänsä torju hyökkäystä. (Tripathi, Hubballi & Singh, 2016). Hyökkäyksen havaitsemiseen on esitetty myös muita normaalin HTTP-liikenteen ja hyökkäyksen aikana syntyvän HTTP-liikenteen vertailuun perustuvia menetelmiä. Esimerkiksi Aiello ym. (2013) esittivät mallin, jossa he vertasivat normaalia HTTP-liikennettä vieraaseen, mahdollisesti anomalioita sisältävään, HTTP-liikenteeseen, keskittyen tiettyyn HTTP-liikenteessä olevaan informaatioon. Tällaista informaatiota oli esimerkiksi aika, joka kului pyynnön ja vastauksen välillä. Myös tämä malli kykeni tarkasti havaitsemaan ainakin Slow Header -hyökkäyksen. (Aiello ym., 2013). Cambiason ym. (2020) mukaan kyseinen havaitsemismenetelmä ei kuitenkaan sovi hyökkäyksen havaitsemiseen reaaliajassa.

Hitaita HTTP-hyökkäyksiä vastaan voidaan puolustautua myös tekemällä erilaisia muokkauksia palvelinohjelmiston toimintaan. Koska hitaille HTTP-hyökkäyksille on ominaista pyrkiä avaamaan mahdollisimman monta yhteyttä palvelimen kanssa, voidaan samanaikaisten yhteyksien sallittua enimmäismäärää myös lisätä. Tätä ei kuitenkaan voida tehdä loputtomasti ja se kuluttaa runsaasti palvelimen resursseja. Toisaalta hyökkääjälle yhteyksien luominen ja ylläpitäminen on helppoa, joten tämä keino tuskin yksinään riittää estämään hyökkäyksiä. (Hirakawa ym., 2016).

Hyökkäyksiä vastaan voidaan käyttää myös erilaisia palvelinohjelmistoihin asennettavia moduuleja. Apachelle kehitettyjä hitaiden HTTP-hyökkäysten torjuntaan käytettyjä moduuleja ovat muun muassa Antiloris ja mod_reqtimeout. Antiloris -moduuli rajoittaa yhteyksien määrää yhtä IP-osoitetta kohti. Tämä on tehokas keino kaikkien yllä mainittujen palvelunestohyökkäysten torjumiseen, mikäli hyökkäys suoritetaan tavallisena palveluestohyökkäyksenä käyttäen yhtä

IP-osoitetta. Antiloris ei ole tehokas puolustuskeino hajautettuja palvelunesto-hyökkäyksiä vastaan. ”Mod_reqtimeout” -moduulin avulla pystytään säätämään aikaa, joka lähettäjältä saa enintään kulua valmiin pyynnön lähettämiseen palvelimelle, sekä vähimmäisnopeutta, jolla dataa tulee palvelimelle lähettää. Tämä on tehokas puolustuskeino muita, paitsi HTTP PRAGMA-hyökkäystä vastaan. Toisaalta tämä saattaa johtaa siihen, että palvelin katkaisee yhteyden tavallisilta käyttäjiltä, joiden yhteysnopeudet ovat hitaita. (Tripathi & Hubballi, 2021).

5.3 NTP-hyökkäykset

5.3.1 Kiss-O′ Death -hyökkäys

Kiss-O′ -Death -pakettien tarkoitus NTP-protokollassa on vähentää NTP-palvelinten kuormaa. Mikäli NTP-asiakas lähettää NTP-palvelimelle liikaa pyyntöjä, voi NTP-palvelin Kiss-O′ -Death pakettien avulla pyytää vähentämään pyyntöjen määrää tai esimerkiksi lopettamaan pyyntöjen lähettämisen kokonaan (Mills ym., 2010).

RFC5905 (Mills ym., 2010) ei kuitenkaan suoranaisesti vaadi, että Kiss-O′ -Death pakettien alkuperää varmistettaisiin esimerkiksi aikaleimojen avulla. Malhotran ym. (2016) mukaan hyökkääjä pystyy vain NTP-palvelimen ja NTP-asiakkaan IP-osoitteet tietämällä väärentämään Kiss-O′ -Death -paketin, ja tällä tavalla estämään asiakasta saamaan aikatietoja NTP-palvelimelta. Tämän tyylinen Kiss-O′ Death pakettien väärinkäyttö on kuitenkin onnistuttu estämään ohjelmistopäivityksellä ja se on korjattu ainakin ntpd-ohjelmiston versiossa 4.2.8p4. (Malhotra ym., 2016).

5.3.2 NTP Broadcast -hyökkäys

NTP Broadcast -hyökkäyksen avulla NTP-asiakkaita voidaan estää saamasta aikatietoja yleislähetystilaa käyttävältä NTP-palvelimelta. Hyökkäys voidaan suorittaa huolimatta siitä, käytetäänkö yleislähetystilassa autentikointia vai ei. Autentikoimattomassa ympäristössä hyökkäys voisi toimia esimerkiksi seuraavalla tavalla:

1. Aluksi hyökkääjä antaa NTP-asiakkaan synkronoida kellonsa NTP-palvelimen kanssa normaalisti. Kun NTP-asiakas vastaanottaa ensimmäisen yleislähetyspaketin NTP-palvelimelta, vaihtavat palvelin ja asiakas keskenään myös tavallisia asiakas/palvelin -paketteja. Näitä paketteja NTP-asiakas käyttää laskeakseen mahdollisen palvelinten välisen viiveen, joka voi vaikuttaa kellojen synkronoinnin tarkkuuteen. Viive ei saa olla liian suuri, tai asiakas pyytää palvelinta lähettämään aikatiedot uudestaan. (Mills ym., 2010).

2. Hyökkääjä alkaa lähettää tasaisin väliajoin NTP-asiakkaalle yleislähetyspaketteja väärennetystä IP-osoitteesta esittäen legitiimiä NTP-palvelinta. Hyökkääjä asettaa paketteihin väärät aikaleimat, jolloin NTP-asiakas laskee aikapoikkeaman asiakkaan kellon ja palvelimen ilmoittaman kellonajan välillä liian suureksi. Hyökkääjä alkaa tämän jälkeen lähettämänään myös NTP-palvelimelle pyyntöjä väärennetystä IP-osoitteesta matkien NTP-asiakasta.

3. Vääriä aikaleimoja sisältävien pakettien vuoksi, NTP-asiakas alkaa lähettämään uudestaan pyyntöjä NTP-palvelimelle, jotta se voisi yrittää laskea viiveen uudestaan.

4. Koska NTP-palvelin vastaanottaa jatkuvasti hyökkääjältä väärennettyjä pyyntöjä, se vastaa Kiss-O′ -Death -paketeilla NTP-asiakkaan lähettämiin paketteihin. NTP-asiakas ei pysty määrittelemään viivettä, eikä täten myöskään synkronoimaan kelloaan. (Tripathi & Hubballi, 2020).

Autentikointia käyttävässä ympäristössä hyökkäys tapahtuu samalla tavalla, mikäli hyökkääjällä on pääsy samaan verkkoon hyökättävien laitteiden kanssa. Autentikointi NTP:n yleislähetysmallissa perustuu MD5 tiivisteisiin, jotka lasketaan symmetrisen salausavaimen avulla. Koska kaikkien verkossa olevien yleislähetystä käyttävien asiakkaiden tulee pystyä autentikoimaan NTP-palvelimen yleislähetyspaketit, saa samassa verkossa oleva hyökkääjäkin avaimen tietoonsa. Kyseinen tunnistusmenetelmä tekee hyökkäyksen kuitenkin paljon vaikeammaksi verkon ulkopuolelta. (Tripathi & Hubballi, 2020).

5.3.3 Suojatuminen NTP-hyökkäyksiltä

Malhotra ym. (2016) suosittelevat Kiss-O¨-Death-paketteja hyödyntävien hyökkäysten ehkäisemiseksi kyseisen toiminnallisuuden kytkemistä kokonaan pois päältä. Tämä on mahdollista, mutta se estää kyseisten pakettien käyttämistä niiden tosiasialliseen tarkoitukseen, eli NTP-palvelinten kuorman hallitsemiseen. Toisena keinona Malhotra ym. (2016) esittävät, että NTP-asiakkaiden tulisi kryptografisesti todentaa kyselynsä NTP-palvelimelle.

Tripathi ja Hubballin (2020) mukaan NTP:n yleislähetystilaa vastaan kohdistuvilta hyökkäyksiltä voi suojautua muun muassa niin, että NTP:tä käytettäessä ei luoteta puhtaasti vain tähän toimintamalliin. Varalla voidaan käyttää NTP-palvelimia, jotka käyttävät erilaista toimintatapaa, ja joilta aikatiedot voidaan tarvittaessa hakea. Lisäksi Tripathi ja Hubballi (2020) suosittelevat, että NTP-asiakkaat eivät liittäisi kaikkiin NTP-pakettiensa otsakkeisiin tietoa siitä,

miltä palvelimelta he synkronoivat aikatietonsa. Otsaketietojen liittäminen tekee hyökkääjälle hyvin helpoksi selvittää matkittavan NTP-palvelimen IP-osoite. Yleislähetystä käyttävät NTP-palvelimet ja NTP-asiakkaat tulisi sijoittaa myös jonkin osoitteenmuunnos-laitteen (eng. Network Address Translation) taakse, jolloin laitteille ei pystytä lähettämään paketteja verkon ulkopuolelta. Riskinä säilyy kuitenkin verkon sisäpuolelta tulevat hyökkäykset. (Tripathi & Hubballi, 2020).

5.4 DNS-hyökkäykset

DNS-protokolla on yksi internetin peruspalveluista ja hyökkäykset sitä kohtaan voivat vaarantaa todella monet käyttäjät ja verkossa toimivat palvelut. DNS-palvelun häiritsemiseen ja estämiseen on ehdotettu useampia erilaisia keinoja, joista esimerkkeinä ovat muun muassa DNS Water Torture -hyökkäys, sekä erilaiset tulvahyökkäykset. DNS-palvelua vastaan tehdyt hyökkäykset luokitellaan kuitenkin usein hajautettujen palvelunestohyökkäysten luokkaan. (Fang, ym. 2021). Esimerkiksi tulvahyökkäyksen suorittaminen yhdeltä koneelta on tietenkin mahdollista, mutta harvoin realistisesti mitenkään tehokasta. Alla esitelty DNS Chaining -hyökkäys on todennäköisesti myös paljon tehokkaampi suoritettuna hajautettuna palvelunestohyökkäyksenä, mutta sillä voidaan aiheuttaa silti realistista haittaa myös tavallisena palvelunestohyökkäyksenä.

5.4.1 DNS Chaining -hyökkäys

DNS-protokolla mahdollistaa CNAME-tyyppisten tietueiden käyttämisen CNAME-tietueiden avulla verkko-osoite voidaan saada osoittamaan johonkin toiseen verkko-osoitteeseen. DNS mahdollistaa myös CNAME-tietueiden ketjuttamisen, eli verkko-osoite voi osoittaa toiseen verkko-osoitteeseen, joka taas osoittaa seuraavaan verkko-osoitteeseen. (Mockapetris, 1987). Periaatteessa ketjun pituudelle ei ole asetettu tarkkaa ylärajaa, mutta resolverit usein rajoittavat ketjujen pituuksia. Bushartin ja Rossowin (2018) mukaan enemmistö resolvereista hyväksyy kuitenkin keskimäärin noin 9–27 osoitetta sisältävät ketjut.

DNS-palvelussa käytetään DNS-vyöhykkeitä. DNS-vyöhykkeellä tarkoitetaan jonkun organisaation tai järjestelmänhallitsijan osuutta DNS-nimiavaruudesta. Mikäli CNAME-ketjussa käytetyt tietueet sijaitsevat samalla DNS-vyöhykkeellä, osaa autoritäärinen nimipalvelin vastata yhdellä vastauksella koko ketjun tiedot. Mikäli CNAME-ketjun osoitteet sijaitsevatkin esimerkiksi vuorotellen eri DNS-vyöhykkeillä, ei autoritäärinen nimipalvelin osaa vastata kuin ketjussa seuraavana olevan osoitteen.

Tämä mahdollistaa DNS Chaining -hyökkäyksen. Hyökkääjä voi hallinnoida kahta osoitetta, jotka sijaitsevat kahdella eri vyöhykkeellä ja joihin liittyviin kyselyihin vastaa kaksi erillistä autoritääristä nimipalvelinta. Toinen

autoritäärinen nimipalvelin voi olla esimerkiksi hyökkääjän itse ylläpitämä ja niitä voi toki olla useampiakin. Toinen autoritäärinen nimipalvelin on hyökkäyksen kohde. CNAME-tietueet rakennetaan niin, että joka toinen ketjussa oleva osoite on kohteena olevan nimipalvelimen hallinnassa. Ketju voi muotoutua esimerkiksi seuraavalla tavalla: a.kohde.com → b.palvelin2.fi, b.palvelin2.fi → c.kohde.com ja niin edelleen. Koska CNAME-ketjun pituudeksi voidaan hyväksyä jopa 30 (tai loputtomasti) osoitetta, saadaan yhdellä viestillä aiheutettua moninkertainen määrä vastauksia. (Bushart ja Rossow, 2018).

Yhdellä resolverilla suoritettuna hyökkäys ei ole järin tehokas, mutta hyökkääjä voi käyttää hyväkseen esimerkiksi tuhansia avoimia DNS resolvereita. Bushartin ja Rossowin (2018) mukaan hyökkäys voidaan suorittaa vain yhdestä lähteestä tavallisena palvelunestohyökkäyksenä, mutta hyökkäys olisi joka tapauksessa tehokkaampi hajautettuna palvelunestohyökkäyksenä.

5.4.2 DNS-hyökkäyksiltä puolustautuminen

DNS chaining -hyökkäykseltä puolustautumiseen Bushart ja Rossow (2018) suosittelevat muutamia erilaisia keinoja. Yksi keinoista on vaatia, että pyynnöissä käytetään isompaa TTL-aikaa kuin 0. TTL-ajan asettaminen nollaksi on vaatimus siitä, etteivät resolverit talleta pyyntöjä välimuistiinsa. Bushartin ja Rossowin (2018) mukaan valtaosa avoimista resolvereista myös noudattaa tätä vaatimusta. Se tallettaako resolveri tietoja välimuistiinsa vaikuttaa suoraan siihen, kuinka usein se joutuu välittämään pyyntöjä kohteena olevalle nimipalvelimelle. Mikäli pyyntöjä ei välitetä tarpeeksi, ei hyökkäys voi onnistua.

Toisekseen resolverit voisivat sallia vain lyhyet CNAME-ketjut. Ketjut eivät kuitenkaan voi olla liian lyhyitä, jotta niitä voidaan käyttää niiden tosiasialliseen tarkoitukseen. Resolvereissa tulisi kiinnittää huomiota myös siihen, etteivät ne lähetä pyyntöjä liian tiheästi uudestaan ylikuormittuneelle autoritääriselle nimipalvelimelle. (Bushart & Rossow, 2018).

6 HITAIDEN HTTP-HYÖKKÄYSTEN TESTAUS APACHE-PALVELINOHJELMISTOA VASTAAN

Tässä tutkielman tutkimusosiossa testataan miten hitaat HTTP-hyökkäykset vaikuttavat Apache HTTP Server -ohjelmistoon, eli voidaanko kyseisillä hyökkäyksillä vielä nykyäänkin aiheuttaa palvelinestohyökkäys sivustolla, joka käyttää palvelinohjelmistonaan uusinta Apache HTTP Server -ohjelmistoa.

6.1 Testausympäristö

Testaus suoritetaan käyttämällä VirtualBox-ohjelmistoa, jonka versionumero on 7.0.6. VirtualBoxiin on asennettu kaksi virtuaalikonetta. Virtuaalikoneet ovat yhteydessä toisiinsa käyttäen VirtualNoxin ”Host-only Adapter” -verkkotilaa. Hyökkäyksen kohteena olevassa koneessa on asennettuna Ubuntu 20.04.6 ja se käyttää palvelinohjelmistonaan kirjoitushetkellä uusinta Apache HTTP Server -versiota, jonka versionumero on 2.4.56.

Hyökkäävänä koneena on Kali linux, jonka versio on 2023.1. Kaliin on asennettu SlowHTTPTest -ohjelmisto, jonka versionumero on 1.8.2.

Testausympäristössä ei ole käytössä palomuureja tai muitakaan erikseen asennettuja suojausominaisuuksia. Koska testissä käytetään VirtualBoxin ”Host-only Adapter” -tilaa, ovat koneet myös samassa verkossa. Seuraavissa luvuissa kerrotaan hieman tarkemmin testissä käytetyistä ohjelmistoista.

6.1.1 Apache HTTP Server

Apache HTTP Server on avoimeen lähdekoodiin perustuva, yksi maailman suosituimmista, HTTP-palvelinohjelmistoista. Sen on kehittänyt Apache Software Foundation. Apache HTTP Serveriä on kehitetty pitkään ja Apache 1.0 julkaistiin jo vuonna 1995. Apache on yrityksen itsensä mukaan suunniteltu luotettavaksi

ja turvalliskesi, kaupallisen tason ratkaisuksi, joka pystyy tarjoamaan HTTP-palveluita nykystandardien mukaisesti. (Apache , ei pvm.).

6.1.2 SlowHTTPTest

SlowHTTPTest on Sergey Shekyanin kehittämä sovellus, joka on tarkoitettu web-palvelimien testaamiseen. SlowHTTPTest -sovelluksella voidaan suorittaa erilaisia, lähinnä HTTP-protokollaa hyödyntäviä, palvelunestohyökkäyksiä. Sovellusta pystytään konfiguroimaan monipuolisesti eri tarkoituksiin ja se tarjoaa testeistä myös monipuolista statistiikkaa, jota voidaan hyödyntää testien vaikutusten analysointiin. Palvelun saatavuuden SlowHTTPTest selvittää lähettämällä kohteena olevalle palvelimelle tasaisin väliajoin kunnollisen GET-pyynnön. Mikäli palvelin ei kykene vastaamaan määritellyn ajan puitteissa, katsotaan palvelunestohyökkäyksen onnistuneen vaikuttamaan palvelun saatavuuteen. (Sergey Shekyan, 2020).

6.1.3 VirtualBox

VirtualBox on Oraclen kehittämä ilmainen, avoimeen lähdekoodiin perustuva virtuaalikonemonitori. VirtualBox mahdollistaa virtuaalikoneiden luonnin, jolloin niihin voidaan asentaa eri käyttöjärjestelmiä, sekä useiden eri virtuaalikoneiden keskitetyn hallinnoinnin. Isäntäkoneen resursseja jakamalla VirtualBoxin avulla pystytään siis ikään kuin ajamaan useita koneita yhden koneen sisällä. (VirtualBox, ei pvm.)

6.2 Testin kulku

Testissä suoritettiin kolme erilaista, jo aikaisemmin esiteltyä, palvelunestohyökkäystä. Nämä ovat Slow Header- hyökkäys, Slow Read -hyökkäys, sekä Slow Message Body -hyökkäys. SlowHTTPTest-ohjelmiston tarjoaman statistiikan avulla voitiin analysoida hyökkäysten onnistumista ja arvioida, onko Apache HTTP Server -ohjelmisto edelleen altis hitaille palvelunestohyökkäyksille, mikäli käytössä ei ole ylimääräisiä puolustusmekanismeja. Koska SlowHTTPTest näyttää tiedot sekunnin tarkkuudella, eivät tulokset ole aivan tarkkoja. On esimerkiksi vaikea sanoa tarkasti, missä vaiheessa palvelin tarkalleen ottaen lakkasi vastaamasta pyyntöihin.

SlowHTTPtest mahdollistaa hyökkäyksen parametrien muokkaamiseen halutulla tavalla. Testissä käytettyjä parametrejä olivat muun muassa: yhteyksien määrä, jonka slowHTTPtest pyrkii kokonaisuudessaan muodostamaan, aikaväli sekunteina kuinka usein dataa lähetetään, testin tavoitekesto sekunteina, sekä yhteyksien määrä sekunnissa. Testi ei todellisuudessa välttämättä kestä tavoitekestossa määriteltyä sekuntimäärää, jos kohteena oleva palvelinohjelmisto

esimerkiksi sulkee kaikki yhteydet aikaisemmin. Lisäksi tietyt hyökkäykset vaativat kyseiselle hyökkäykselle erityisesti suunniteltuja parametrejä, johtuen hyökkäysten erilaisista toimintatavoista. Nämä parametrit esitellään jokaisen hyökkäyksen kohdalla erikseen.

Testi suoritettiin siten, että ensimmäinen testi tehtiin käyttämällä 500 yhteyttä. Tämän jälkeen testi tehtiin 1000 yhteydellä. Mikäli tämä ei riittänyt palvelunestohyökkäyksen aiheuttamiseksi, nostettiin yhteyksien määrää 1000 yhteyttä kerrallaan. Mikäli hyökkäys ei onnistunut lainkaan, katsottiin palvelinohjelmiston kykenevän suojautumaan kyseiseltä hyökkäykseltä. Testeissä käytetyt kaikille testeille yhteiset parametrit löytyvät alla olevasta taulukosta (taulukko 1).

TAULUKKO 1 Testien yhteiset parametrit

Parametri	Parametrin selitys	Arvo
-c	Yhteyksien määrä	500, 1000
-r	Yhteyksien määrä sekunnissa	50 s
-i	Intervalli datan lähettämisen välillä sekunteina	10 s
-l	Testin kokonaiskesto, mikäli yhteys ei katkea muista syistä sekunteina	240 s
-p	Aika sekunteina, jossa palvelimen tulee kyetä vastaamaan ennen kuin katsotaan, ettei palvelu enää ole saavutettavissa	5 s
-g	Parametri mahdollistaa statistiikan keräämisen hyökkäyksestä	-

6.3 Slow Header -hyökkäyksen simulointi ja tulokset

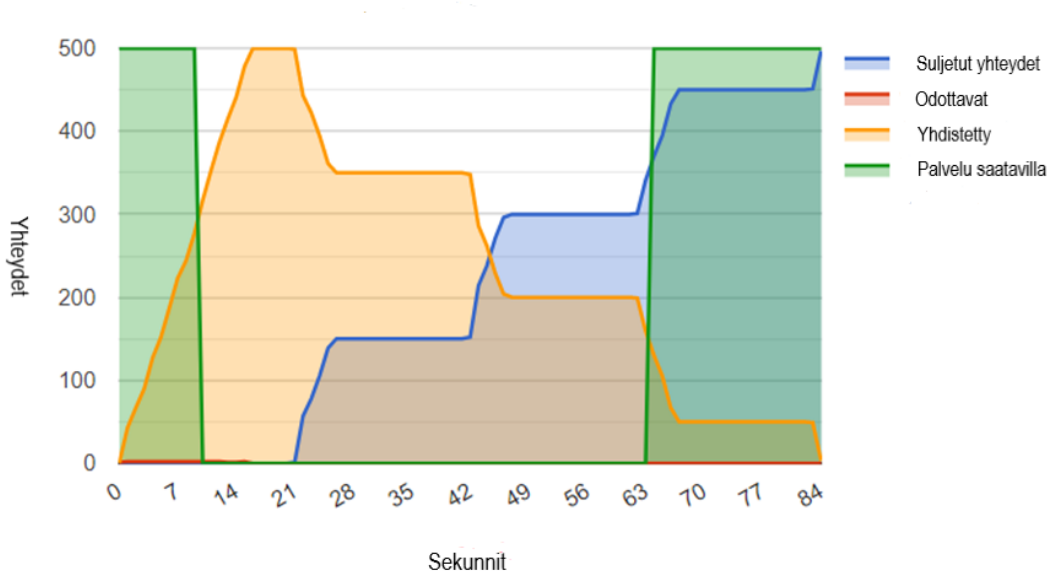
Slow Header -hyökkäys suoritettiin aluksi käyttämällä SlowHTTPTest -ohjelmiston vakioasetuksia. Tämän lisäksi hyökkäyksessä käytettiin yhteisten parametrien lisäksi alla olevassa taulukossa olevia parametreja (taulukko 2).

TAULUKKO 2 Slow Header -hyökkäyksen erityisparametrit

Parametri	Parametrin selitys	arvo
-H	Hyökkäyksen tyyppi, joka on tässä tapauksessa Slow Header.	-
-o	Tiedoston nimi johon tilastot tallennetaan	slowheader
-u	Kohteen osoite ja resurssi.	http://192.168.4.5:80

Ensimmäisenä Slow Header -hyökkäystä testattiin käyttämällä 500 yhteyttä. SlowHTTPTestin tuottama graafi on esitettynä seuraavana olevassa kuviossa (kuvio 5).

$ slowhttptest -H -g -o slowheader -i 10 -p 5 -c 500 -l 240 -r 50 -u http://192.168.4.5:80



KUVIO 5 Slow Header -hyökkäys 500 yhteydellä

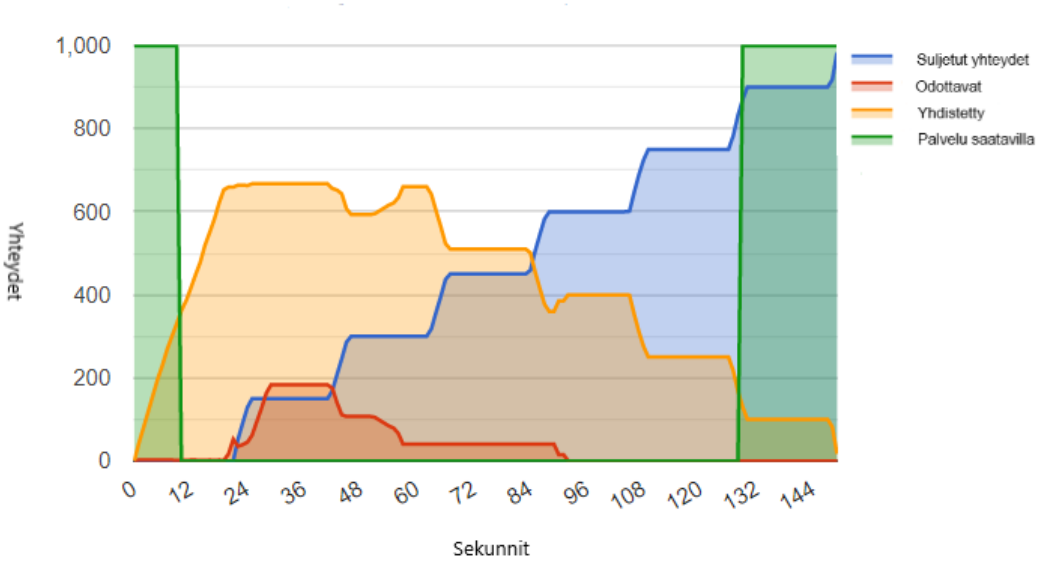
Kuviossa vasemmalla näkyvät yhteyksien kokonaismäärä ja alhaalla on sekunneittain hyökkäyksen kulku. Sininen viiva kuvaa suljettujen yhteyksien määrää ja punainen viiva kuvaa yhteyksiä, jotka vielä odottavat yhteyden muodostamista. Oranssi viiva kuvaa yhteyksiä, jotka SlowHTTPTest on muodostanut palvelimeen kunakin hetkenä. Vihreä väri taas kuvaa sitä, onko palvelu ollut kyseisellä hetkellä saatavilla.

Kuviosta voidaan nähdä, että jo 500 yhteyttä riitti estämään palvelua vastaamasta pyyntöihin. Palvelu alkoi olla saavuttamaton noin 10 sekunnin kohdalla, kun yhteyksiä oli muodostettu noin 317 kappaletta ja rupesi taas vastaamaan pyyntöihin noin 64 sekunnin kohdalla. Yhteensä testi kesti 86 sekuntia,

joista palvelu ei vastannut noin 55 sekuntiin. Palvelinohjelmisto alkoi katkaista yhteyksiä noin 23 sekunnin kohdalla.

Seuraavaksi testi suoritettiin käyttämällä 1000 yhteyttä ja testiohjelmiston tuottama graafi näkyy seuraavassa kuviossa (kuvio 6).

$ slowhttptest -H -g -o slowheader -i 10 -p 5 -c 1000 -l 240 -r 50 -u http://192.168.4.5:80



KUVIO 6 Slow Header -hyökkäys 1000 yhteydellä

Kuviosta voidaan nähdä, että noin 12 sekunnin jälkeen, palvelin ei enää kyennyt vastaamaan SlowHTTPTest -ohjelmiston GET-pyyntöihin. Hyökkäys kesti kokonaisuudessaan 144 sekuntia, joista palvelinohjelmisto ei kyennyt vastaamaan noin 118 sekuntiin. Palvelu ei kyennyt enää vastaamaan pyyntöihin, kun yhteyksiä oli muodostettu noin 362 kappaletta. Apachen palvelinohjelmisto alkoi sulkemaan yhteyksiä, kun aikaa hyökkäyksen alusta oli kulunut noin 22 sekuntia. Missään vaiheessa SlowHTTPTest ei kyennyt muodostamaan kaikkia tuhatta yhteyttä palvelimen kanssa.

6.4 Slow Message Body -hyökkäyksen simulointi ja tulokset

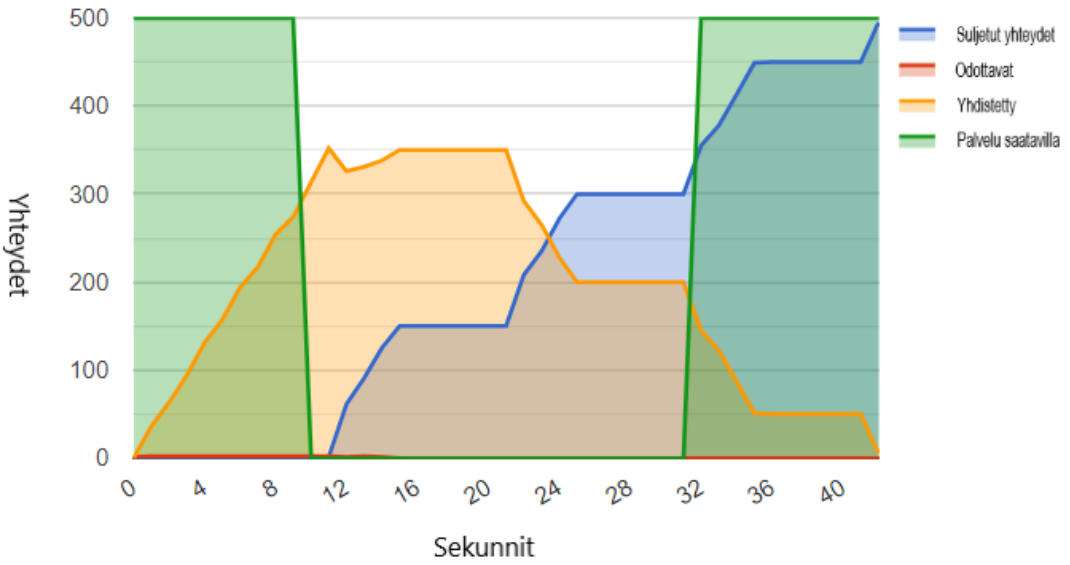
Slow Message Body -hyökkäyksessä käytettiin yhteisten parametrien lisäksi parametrejä, jotka on esitelty alla olevassa taulukossa (taulukko 3).

TAULUKKO 3 Slow Message Body -hyökkäyksen erityisparametrit

Parametri	Parametrin selitys	Arvo
-B	Hyökkäyksen tyyppi, joka on tässä tapauksessa slow body	-
-o	tiedoston nimi, johon tilastot tallennetaan.	slowbody
-s	Content-Lenght -otsakkeen arvo.	12288
-u	Kohteen osoite ja resurssi	http://192.168.4.5:80

Ensimmäisenä Slow Message Body -hyökkäystä testattiin käyttämällä 500 yhteyttä. SlowHTTPTestin tuottama graafi on esitettynä seuraavana olevassa kuviossa (kuvio 7).

$ slowhttptest -B -g -o slowbody -i 10 -p 5 -c 500 -s 12288 -l 240 -r 50 -u http://192.168.4.5:80

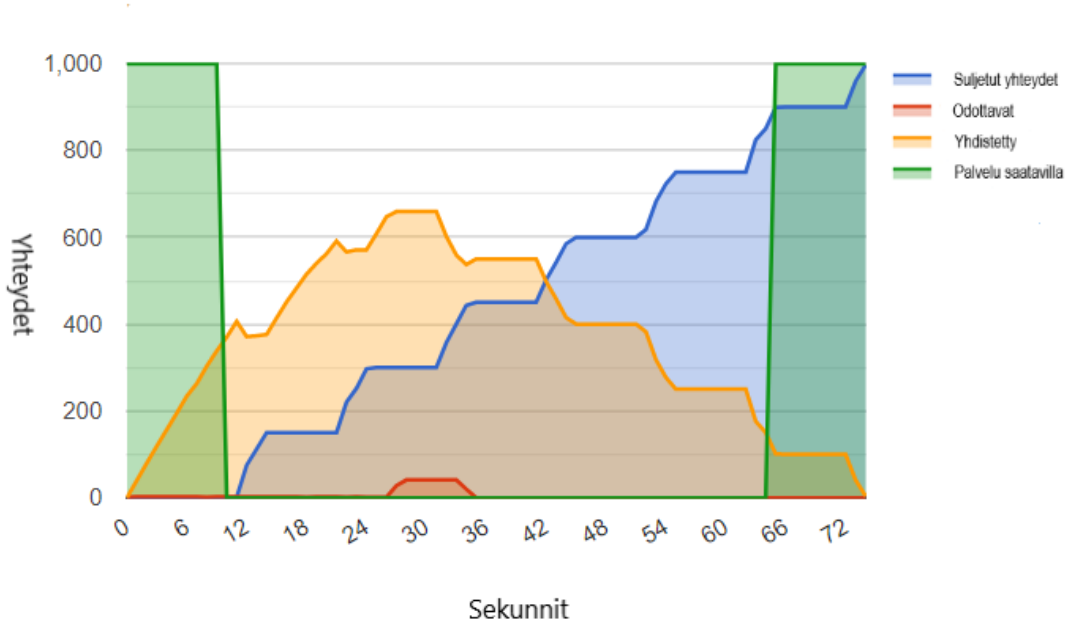


KUVIO 7 Slow Message Body -hyökkäys 500 yhteydellä

Kun käytössä oli yhteensä 500 yhteyttä, palvelu alkoi olla saavuttamaton noin 10 sekunnin kohdalla, kun yhteyksiä oli muodostettu noin 313 kappaletta. Yhteensä testi kesti 42 sekuntia, joista palvelu oli saavuttamaton noin 22 sekuntia. Palvelin rupesi sulkemaan yhteyksiä noin 11 sekunnin kohdalla.

Seuraavaksi suoritetaan hyökkäys käyttämällä 1000 yhteyttä ja tulokset ovat nähtävissä seuraavaksi esitetyssä kuviossa (kuvio 8).

$ slowhttptest -B -g -o slowbody -i 10 -p 5 -c 1000 -s 12288 -l 240 -r 50 -u http://192.168.4.5:80



KUVIO 8 Slow Message Body -hyökkäys 1000 yhteydellä

Palvelin lakkasi vastaamasta testiohjelmiston pyyntöihin noin 12 sekunnin jälkeen, kun yhteyksiä oli muodostettu noin 370 kappaletta. Noin 11 sekunnin jälkeen palvelinohjelmisto alkoi sulkea yhteyksiä ja kokonaisuudessaan hyökkäys kesti 72 sekuntia. Tästä ajasta palvelu oli pois käytöstä noin 56 sekuntia.

6.5 Slow Read -hyökkäyksen simulointi ja tulokset

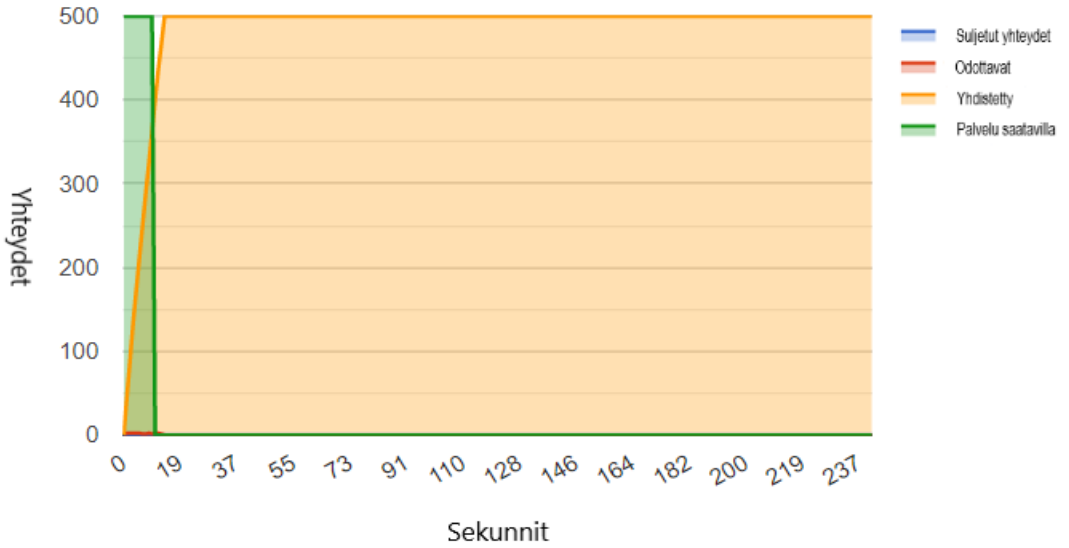
Slow Read -hyökkäys eroaa toimintatavoiltaan melko paljon aikaisemmista hyökkäyksistä ja vaatii ylimääräisiä parametreja. Parametrit on esitelty alla olevassa taulukossa (taulukko 4).

TAULUKKO 4 Slow Read -hyökkäyksen erityisparametrit

Parametri	Parametrin selitys	Arvo
-w	Arvo tavuina, josta alkaen lähetysikkunan koko valitaan	10
-y	Maksimiarvo tavuina, josta lähetysikkunan koko valitaan	20
-z	Kuinka monta tavua luetaan kerrallaan per yhteys	16
-n	Kuinka monen sekunnin välein -z -parametrin määrittelemä tavumäärä luetaan	5
-X	Hyökkäyksen tyyppi, joka tässä tapauksessa on slow read	-
-o	Tiedoston nimi, johon tilastot tallennetaan	slowread
-u	Kohteen osoite ja resurssi	http://192.168.4.5/file.txt

Koska Slow Read -hyökkäyksessä vastaus pyritään lukemaan hitaasti, ladattiin kohteen palvelimelle file.txt -tiedosto, jonka koko on 134 megatavua. Ensin hyökkäystä testattiin 500 yhteydellä. Testin tulokset näkyvät seuraavassa kuviossa (kuvio 9).

$ slowhttptest -X -g -o slowread -c 500 -l 240 -r 50 -w 10 -y 20 -n 5 -z 16 -u http://192.168.4.5/file.txt

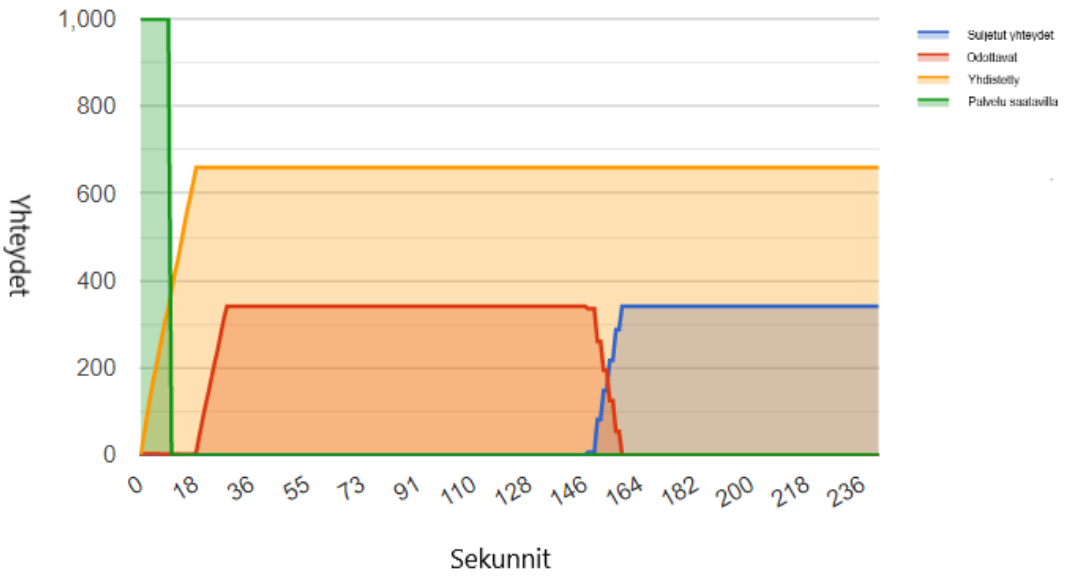


KUVIO 9 Slow Read -hyökkäys 500 yhteydellä

Kun yhteyksiä oli käytössä 500, lakkasi palvelu vastaamasta noin 10 sekunnin kohdalla. Palvelu lakkasi vastaamasta, kun yhteyksiä oli muodostettu noin 400 kappaletta. Palvelu pysyy alhaalla tavoitetestiajan loppuun asti, eli yhteensä 230 sekuntia. Palvelinohjelmisto ei myöskään alkanut katkaista yhteyksiä missään vaiheessa testiä.

Seuraavaksi hyökkäys suoritettiin käyttämällä 1000 yhteyttä. Tulokset näkyvät seuraavassa kuviossa (kuvio 10).

$ slowhttptest -X -g -o slowread_1000 -c 1000 -l 240 -r 50 -w 10 -y 20 -n 5 -z 16 -u http://192.168.4.5/file.txt



KUVIO 10 Slow Read -hyökkäys 1000 yhteydellä

Kun yhteyksien kokonaismäärä oli 1000, palvelu lakkasi vastaamasta noin 10 sekunnin kohdalla. Palvelun estymisen alkamisajankohtana yhteyksiä oli muodostettu palvelimeen noin 375 kappaletta. Palvelu pysyi palvelun estymisen jälkeen alhaalla testiajan loppuun saakka, eli yhteensä 230 sekuntia. Palvelin alkoi katkaista yhteyksiä noin 147 sekunnin kohdalla.

6.6 Tulosten analysointi

Testauksen perusteella voidaan sanoa, että myös Apachen uusin palvelinohjelmisto on altis yleisille hitaille palvelunestohyökkäyksille. Tämä toisaalta korostaa hitaiden palvelunestohyökkäysten luonnetta, eli kykyä hyödyntää protokollien toimintaperiaatteissa olevia puutteita. Hyödynnettävät puutteet

palvelinohjelmistossa eivät siis niinkään ole haavoittuvuuksia, jotka voitaisiin vain korjata uusilla sovellusversioilla. Käytännössä Apachen palvelinohjelmisto ei vakioasetuksillaan ja ilman ylimääräisiä suojausominaisuuksia ole testien perusteella kykenevä puolustautumaan hitailta HTTP-protokollaan hyödyntäviltä palvelunestohyökkäyksiltä.

Kaikissa hyökkäystyypeissä palvelu lakkasi vastaamasta SlowHTTPTest -ohjelmistolle, kun yhteyksiä oli muodostettu palvelimeen 300–400 kappaletta. Testissä käytetyillä parametreillä Slow Headers -, ja Slow Body -hyökkäys vaikuttivat testipalvelimeen suunnilleen samalla tavalla. Slow Headers -hyökkäys kuitenkin onnistui pitämään palvelimen saavuttamattomissa 33 sekuntia pidempään, kun käytössä oli kokonaisuudessaan 500 yhteyttä. Erot korostuivat, kun yhteyksien kokonaismäärä nostettiin 1000 yhteyteen. Slow Headers -hyökkäys onnistui pitämään palvelun alhaalla 62 sekuntia pidempään kuin Slow Body -hyökkäys. Slow Headers -hyökkäyksessä palvelin alkoi katkaista yhteyksiä vasta noin 20 sekunnin jälkeen. Slow Body -hyökkäyksessä yhteyksien katkaisu alkoi jo noin 11 sekunnin kohdalla.

Slow Read-hyökkäys erosi vaikutuksiltaan melko merkittävästi aiemmista hyökkäyksistä. Jo 500 yhteydellä palvelu alkoi olla saavuttamaton 10 sekunnin kohdalla ja pysyi alhaalla testiajan loppuun saakka. Koska palvelinohjelmisto ei myöskään missään vaiheessa alkanut katkaista yhteyksiä, on syytä olettaa, että hyökkäys olisi voinut jatkua merkittävästikin kauemmin. Palvelu pysyi alhaalla 10 sekunnin jälkeen testin loppuun myös silloin kun käytössä oli 1000 yhteyttä. Palvelin kuitenkin alkoi katkaista yhteyksiä n. 147 sekunnin kohdalla. Tässäkin tapauksessa on kuitenkin syytä olettaa, että hyökkäys olisi voinut jatkua kauemminkin. Testien tulokset löytyvät alla olevasta taulukosta. Arvot ovat suuntaa antavia, koska SlowHTTPTest -ohjelmisto tarjoaa informaatiota vain sekunnin välein. Ensimmäinen taulukko näyttää tulokset, kun käytössä oli 500 yhteyttä (taulukko 5) ja toinen taulukko 1000 yhteydellä saavutetut tulokset (taulukko 6).

TAULUKKO 5 Testien tulokset 500 yhteydellä

	Slow Headers	Slow Message Body	Slow Read
Testin kesto (sekunteina)	86 s	42 s	240 s
Palvelu alhaalla (sekunteina)	55 s	22 s	230 s
Yhteyksien määrä, kun palvelu lakkasi vastaamasta	317	313	400
Aika testin alusta kun palvelin alkoi katkaista yhteyksiä (sekunteina)	23 s	11 s	Palvelin ei katkaissut yhteyksiä.

TAULUKKO 6 Testien tulokset 1000 yhteydellä

	Slow Headers	Slow Message Body	Slow Read
Testin kesto (sekunteina)	144 s	72 s	240 s
Palvelu alhaalla (sekunteina)	118 s	56 s	230 s
Yhteyksien määrä, kun palvelu lakkasi vastaamasta	362	370	375
Aika testin alusta alkaen, kun palvelin alkoi katkaista yhteyksiä (sekunteina)	22 s	11 s	147 s

6.7 Testauksen rajoitteet

Testin tarkoitus oli testata, onko Apachen palvelinohjelmisto edelleen haavoittuvainen yleisille hitaille HTTP-protokollaa hyödyntäville palvelunestohyökkäyksille. Testiasetelmaa ei voida pitää erityisen realistisena, koska käytössä ei ole minkäänlaisia puolustusmekanismeja. Tämä oli kuitenkin välttämätöntä, jotta Apachesta ja sen vakioasetuksista riippumattomat asiat eivät vaikuttaisi tuloksiin. Testi suoritettiin kotiverkossa kannettavalla tietokoneella. On syytä olettaa, että tulokset olisivat erilaisia erilaisessa ympäristössä. Vaikutus olisi luultavasti kuitenkin lähinnä siihen, kuinka pienillä resursseilla palvelin voidaan saada saavuttamattomaksi. Ei voida siis sanoa, että palvelinohjelmisto olisi suojattuna hitailta palvelinestohyökkäyksiltä, vaikka sillä olisikin käytössä hieman enemmän resursseja.

Testissä käytettiin osin yhteisiä parametreja erilaisille hyökkäyksille, eikä laajaa testausta tehty siitä, minkälaisilla asetuksilla hyökkäys olisi kullakin hyökkäystyypillä tehokkain. Toisin sanoen, parametrien tarkemmalla muokkaamisella vastaamaan juuri kyseisen hyökkäyksen vaatimuksia ja toimintatapoja, tuloksia luultavasti saataisiin erilaiksi ja hyökkäyksiä todennäköisesti tehokkaammiksi. Testeissä ei myöskään havainnoitu tarkemmin sitä, minkälaisia vaikutuksia hyökkäyksillä oli kohdekoneeseen. Ainoa käytetty mittari oli se, kykenikö palvelinohjelmisto vastaamaan SlowHTTPTest -ohjelmiston GET-pyyntöihin määritellyssä ajassa. Järjestelmän statistiikkaa tutkimalla hyökkäyksen aikana, hyökkäysten tehosta voitaisiin saada parempi kokonaiskäsitys.

Testin numeraaliset tulokset ovat myös lähinnä suuntaa antavia. Tarkasti voidaan esimerkiksi sanoa, montako yhteyttä kullakin sekuntimäärällä yhteyksiä

on muodostunut. Toisaalta tulosten perusteella ei voida sanoa tarkasti, että juuri tietty yhteysmäärä riitti aiheuttamaan palvelun estymisen kohdepalvelimella, koska yhteyksiä ehtii muodostua myös yhden sekunnin aikana. Luultavaa on, että kaikissa hyökkäystyypeissä palvelu lakkasi vastaamasta suunnilleen samalla palvelimeen muodostetulla yhteismäärällä.

Testin tuloksia ei siis tule käyttää muuhun, kun sen toteamiseen, että Apachen palvelinohjelmisto ei uusimmallakaan versiolla ole vakioasetuksillaan suojattuna testissä simuloiduilta palvelunestohyökkäyksiltä.

7 YHTEENVETO

Tässä tutkielmassa esiteltiin erilaisia sovelluskerroksen palvelunestohyökkäyksiä sekä mahdollisia torjuntakeinoja näiden hyökkäysten estämiseksi. Lisäksi tutkielmassa esiteltiin sovelluskerroksen protokollia, joita hyökkääjät voivat hyödyntää hyökkäysten tekemiseen. Tutkielmassa kuvattiin myös kaksi eri mallia, joiden avulla järjestelmiä voidaan kuvata kerroksittain. Nämä mallit ovat OSI-malli ja TCP/IP-malli. Tutkielmassa annettiin myös yleistietoa palvelunestohyökkäyksistä ja niiden taustalla vaikuttavista motiiveista. Tutkielman tutkimusosiossa erilaisia HTTP-protokollaa hyödyntäviä palvelunestohyökkäyksiä testattiin Apachen uusinta palvelinohjelmistoa vastaan, sekä analysoitiin hyökkäysten vaikutuksia. Tutkielma pyrki vastaamaan seuraaviin tutkimuskysymyksiin:

- Minkälaisia protokollia käytetään sovelluskerroksella ja minkälainen on niiden perustoiminta?
- Minkälaisia sovelluskerroksen palvelunestohyökkäyksiä on esitelty aikaisemmassa tutkimuskirjallisuudessa ja miten niiltä voidaan puolustautua?
- Onko Apachen uusin palvelinohjelmisto altis sovelluskerroksen palvelunestohyökkäyksille, jos käytössä ei ole muita suojausmekanismeja?

Tutkielmassa tehtyjen testien perusteella voidaan sanoa, että Apachen uusin palvelinohjelmisto on sellaisenaan edelleen haavoittuvainen hitaita HTTP-protokollaa hyödyntäviä palvelunestohyökkäyksiä vastaan. Ei voida siis olettaa, että esimerkiksi pelkästään sovellusversiota päivittämällä ongelmasta voitaisiin päästä eroon. Palvelinohjelmisto vaatiikin ympärilleen muita puolustusmekanismeja hyökkäysten torjumiseksi.

Testien perusteella Slow Read -hyökkäys toimi tehokkaimmin Apachen palvelinohjelmistoa vastaan. Palvelu ei vastannut ensimmäisen 10 sekunnin

jälkeen enää missään vaiheessa testiohjelmiston lähettämiin pyyntöihin, kun testin kokonaiskesto oli 240 sekuntia.

Testiasetelmaan liittyi kuitenkin huomattavia heikkouksia ja käytännössä tuloksien tulkinnassa tulee noudattaa suurta kriittisyyttä. Testaus testiin pienillä resursseilla ympäristössä, jossa käytössä ei ollut minkäänlaisia suojausominaisuuksia. Myöskään hyökkäämiseen käytetyn työkalun tuottama data ei ole täysin luotettavaa johtuen siitä, että tietoja saatiin vain 1 sekunnin välein. Tutkimus kykeni kuitenkin vastaamaan asetettuun tutkimuskysymykseen. Apachen palvelinohjelmisto ei ole suojattuna testissä simuloiduilta palvelunestohyökkäyksiltä ilman muita toimenpiteitä.

Tulevaisuudessa aiheen tutkimista voitaisiin jatkaa siten, että palvelinohjelmiston asetuksiin tehtäisiin pieniä muutoksia, ja analysoitaisiin niiden vaikutusta palvelinohjelmiston kykyyn sietää sovelluskerroksen palvelunestohyökkäyksiä. Samaan aikaan tutkimuksessa tulisi huomioida palvelinohjelmiston kyky palvella sen oikeutettuja asiakkaita. Tämän tarkoituksena olisi selvittää, voisiko Apachen palvelinohjelmiston vakioasetuksia muuttaa siten, että palvelin olisi paremmassa suojassa ilman merkittävää vaikutusta sen kykyyn palvella sen oikeutettuja asiakkaita. Yleisesti sovelluskerroksen palvelunestohyökkäykset ja niiltä suojautuminen vaativat vielä lisää tutkimusta.

LÄHTEET

Aiello M., Cambiaso, E., Scaglione, S. & Papaleo, G. (2013). A similarity based approach for application DoS attacks detection.Teoksessa: 2013 IEEE Symposium on Computers and Communications (ISCC), (430-435). Split, Croatia, July 7-10, 2013.

Alani, M. (2014). Guide to OSI and TCP/IP Models. Springer.

Apache. (ei pvm.). https://httpd.apache.org/ABOUT_APACHE.html

Belshe, M., Peon, R. & Thomson, M. (2015). RFC: 7540. Hypertext Transfer Protocol Version 2 (HTTP/2).

Berners-Lee, T. The original HTTP as defined in 1991. Haettu osoitteesta: https://www.w3.org/Protocols/HTTP/AsImplemented.html

Berners-Lee, T., Fielding, R. & Frystyk, H. (1996). RFC: 1945. Hypertext Transfer Protocol - HTTP/1.0.

Bishop, M. (2021). Internet-Draft. Hypertext Transfer Protocol Version 3 (HTTP/3).

Bushart, J. & Rossow, C. (2018). DNS Unchained: Amplified Application-Layer DoS Attacks Against DNS Authoratives. Teoksessa: International Symposium on Research in Attacks, Intrusion and Defenses (RAID). (139-160). Heraklon, Crete, Greece, September 10-12, 2018.

Cambiaso, E., Aiello, M., Mongelli, M. & Vaccari, I. (2020). Detection and classification of slow DoS attacks targeting network servers. Teoksessa: Proceedings of the 15th Conference on Availability, Reliability and Security (ARES), (1-7). Virtual Event, Ireland, August 25-28, 2020.

Carl, G., Kesidis, G., Brooks, R. & Rai, s. Denial-of-service attack-detection techiques. IEEE Internet Computing, 10(1).

Catillo, M., Pechhia, A. & Villano, U. (2022). No more DoS? An empirical study on defense techniques for web server Denial of Service mitigation. Journal of Network and Computer Applications, 202.

Cerf, V. & Kahn, R. (1974). A Protocol for Packet Network Intercommunication. IEEE Transactions on Communications, 22(5).

Conard, J. (1983). Services and protocols of the data link layer. Proceedings of the IEEE, 71(12).

Dantas, Y., Nigam, V. & Fonseca, 1. (2014). A Selective Defense for Application Layer DDoS Attacks.Teoksessa 2014 IEEE Joint Intelligence and Security

Informatics Conference. (75-82). The Hague, Netherlands, September 24-26, 2014.

Day, J. & Zimmermann, H. (1983). The OSI Reference Model. Proceedings of the IEEE, 72(12).

Demerjian, J. & Serhrouchni, A. (2004). DHCP AUTHENTICATION USING CERTIFICATES. Deswarte, Y., CuppensSushil, Jajodia, S. & Wang, L. (Toim.) Teoksessa Security and Protection in Information Processing Systems. (457-472), Toulouse, France, August 22-27, 2004.

Deutsch, P. (1996). RFC: 1951. DEFLATE Compressed Data Format Spesification version 1.3.

Droms, R. (1997). RFC: 2131. Dynamic Host Configuration Protocol.

Fang, L., Wu., H., Qian, K., Wang, W. & Han, L. (2021). A Comprehensive Analysis of DDoS attacks based on DNS. Journal of Physics: Conference Series 2024(1).

Fielding, R. & Reschke, J. (2014). RFC: 7230. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L. Leach, P. & Berners-Lee, T. (1999). RFC: 2616. Hypertext Transfer Protocol - HTTP/1.1.

Gonzales, H., Gosselin-Lavigne, M., Stakhanova, N. & Ghorbani, A. (2014). Kirjassa: Case Studies in Secure Computing. Taylor & Francis.

Gourley, D., Totty, B., Sayer, M., Reddy, S. & Aggarwal, A. (2002). HTTP:the definitive Guide. O'Reilly Media.

Herzog, S. (2011). Revisiting the Estonian Cyber Attacks: Digital Threats and Multinational Responses. Journal of Strategic Security, 4(2).'

Hirakawa, T., Ogura, K., Bista, B. & Takata, T. (2016). A Defense Method against Distributed Slow http DoS Attack. Teoksessa: 2016 19th International Conference on Network-Based Information Sytems (NBiS), (152-158). Ostrava, Czech Republic, September 7-9, 2016.

Hubballi, N. & Tripathi, N. (2017). A closer look into DHCP starvation attack in wireless networks. Computers & Security, 65.

Iren, S., Amer, P. & Conrad, P. (1999). The Transport Layer: Tutorial and Survey. ACM Computing Surveys, 31(4).

ISO/IEC 7498-1. (1994). Intermation technology - Open Systems Interconnection - Basic Reference Model: The Basic Model.

Kabelova, A. & Dostalek, L. (2006). DNS in Action: A Detailed and Practical Guide to DNS Implemenatition, Configuration, and Administration. Packt Publishing.

Lovell, D. (2021). Web almanac: Part IV Chapter 24 HTTP. Haettu osoitteesta https://almanac.httparchive.org/en/2021/http#introduction.

Malhotra, A., Cohen, I., Brakke, E. & Goldberg, S. (2016). Attacking the Network Time Protocol. Teoksessa Network and Distributed System Security Symposium (NDSS). San Diego, CA, USA, February 21-24, 2016.

Mantas, G., Stakhanova, N., Gonzales, H., Jazi, H. & Ghorbani, A. (2015). Application-layer denial of service attacks: taxonomy and survey. International Journal of Information and Computer Security, 7(2/3/4).

Mills, D. (2011). Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space. CRC Press.

Mills, D., Martin J., Burbank, J. & Kasch, W. (2010). RFC: 5905. Network Time Protocol Version 4: Protocol and Algorothms Specification.

Mirkovic, J. & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS Defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2).

Mockapetris, P. (1987). RFC: 1034. Domain Names - Concepts and Facilitites.

Mukhtar, H., Salah, K. & Iraqi, Y. (2012). Mitigation of DHCP starvation attack. Computers & Electrical Engineering, 38(5).

Noergaard, T. (2005). Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers. Elsevier.

OConnor, T. (2010). Detecting and Responding to Data Link Layer Attacks. SANS Institute. GIAC Gold Certification.

Oracle VirtualBox. (ei pvm.). User Manual https://www.virtualbox.org/manual/UserManual.html

Patrick, M. (2001). RFC: 3046. DHCP Relay Agent Information Option.

Postel, J. (1980). RFC: 768. User Datagram Protocol.

Postel, J. (1981, a). RFC: 791. Internet Protocol

Postel, J. (1981, b). RFC: 793. Transmission Control Protocol.

Rooney, T. (2011). IP ADDRESS MANAGEMENT: Principles and Practice. IEEE Press.

Russell, A. (2013). The internet that wasn't. IEEE Spectrum, 50(8).

Saxena, P. (2014). OSI Reference Model - A Seven Layered Architecture of OSI Model. International Journal of Research. 1(10).

Shekyan, S. (1.5.2012). Are you ready for slow reading? https://blog.qualys.com/vulnerabilities-threat-research/2012/01/05/slow-read

Shekyan, S. (24.08.2020). InstallationAndUsage. https://github.com/shekyan/slowhttptest/wiki

Suroto, S. (2017). A Review of Defense Against Slow http Attack. International Journal on Informatics Vizualization, 1(4).

Tripathi, H. & Hubballi, N. (2021). Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey. ACM Computing Surveys, 54(4).

Tripathi, N. & Hubballi, N. (2018, a). Detecting Stealth DHCP Starvation Attack using Machine Learning Approach. Journal of Computer Virology and Hacking Techniques, 14(2).

Tripathi, N. & Hubballi, N. (2018, b). Slow Rate Denial of Service Attacks Against HTTP/2 and Detection. Computers & Security, 72.

Tripathi, N. & Hubballi, N. (2020). Preventing Time Synchronization in NTP Mode. Computers & Security, 102(2).

Tripathi, N., Hubballi, N. & Singh, Y. (2016). How Secure are Web Servers? An Empirical Study of Slow http DoS Attacks and Detection. Teoksessa: 11th International Conference on Availability, Reliability and Security (ARES). (454-463), Salzburg, Austria, August 31. – September 2., 2016.

Yaibuates, M. & Chaisricharoen, R. (2018). Implementing of IP address Recovery for DHCP Service. International Journal of Applied Engineering Research, 13(5).

Zargar, S., Joshi, J. & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. IEEE Communications Surveys & Tutorials, 15(4).