Janne Kettunen

Palvelunestohyökkäykset sovelluskerroksella

Tieto- ja ohjelmistotekniikan kandidaatintutkielma

30. huhtikuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Janne Kettunen

Yhteystiedot: janne.b.kettunen@student.jyu.fi

Työn nimi: Palvelunestohyökkäykset sovelluskerroksella

Title in English: Application-layer denial of service attacks

Työ: Kandidaatintutkielma

Sivumäärä: 15+0

Tiivistelmä: Tämä kandidaatintutkielma käsittelee palvelunestohyökkäyksiä sovelluskerroksella kirjallisuuskatsauksen muodossa. Palvelunestohyökkäyksiltä puolustautuminen ja niiden tutkiminen on tärkeää, sillä niillä on suuria yhteiskunnallisia merkityksiä. Lisäksi hyökkäykset kehittyvät jatkuvasti, mikä tuo uusia haasteita. Tässä tutkielmassa esitellään kaksi tapaa puolustautua sovelluskerroksen palvelunestohyökkäyksiltä ja riittävät esitiedot niiden ymmärtämiseksi.

Avainsanat: kandidaatintutkielmat, kirjallisuuskartoitus, dos, ddos, sovelluskerros, torjunta

Abstract: This bachelor's thesis is about denial of service attacks at the application layer in the form of a literature review. Defending against denial of service attacks and researching them is important, as they have great social consequences, in addition, the constant development of attacks constantly brings new challenges. This paper presents two ways to defend against application layer denial of service attacks, and sufficient background information to understand them.

Keywords: Bachelor's Theses, literature review, DoS, DDoS, application layer, prevention

Kuviot

Sisällys

1 Johdanto

Palvelunestohyökkäys (engl. *denial-of-service attack, DoS attack*) on yksi vanhimmista tunnetuista hyökkäyksistä verkkosovelluksia ja -palveluita kohtaan. Ensimmäinen havaittu palvelunestohyökkäys tapahtui marraskuussa 1988, ja se esti useiden tutkimuslaitosten järjestelmien normaalin toiminnan (Othman 2000). Palvelunestohyökkäyksillä hyökkääjät ruuhkauttavat kohteena olevan palvelun ja tekevät sen käyttökelvottomaksi sen käyttäjille. Lyhyetkin katkokset palveluissa voivat aiheuttaa merkittävän suuria kustannuksia palvelun omistavalle organisaatiolle ja/tai haittaa palvelun käyttäjille. Lisäksi hyökkäykset voivat vaikuttaa negatiivisesti kohdeyrityksen tai -organisaation imagoon sekä aiheuttaa luottamuksen puutetta asiakkaissa (Praseed ja Thilagam 2018).

Hajautettu palvelunestohyökkäys (engl. *distributed denial-of-service attac, DDoS attack*) ovat yksi suurimmista verkkosovelluksiin kohdistuvista uhkista. DDoS-hyökkäys on tyypillisesti yhdistelmä tai sarja DoS-hyökkäyksiä, yleensä useammasta eri isännästä (engl. *host*) toteutettuja (Othman 2000). Monet suuret organisaatiot ja yritykset sekä tärkeät yhteiskunnalliset palvelut, kuten valtionhallinnon verkkosivustot ja pankit käyttävät erilaisia verkkosovelluksia. Tämän takia kyseisten palveluiden turvaaminen on ensiarvoisen tärkeää. Edellä mainittujen kaltaiset palvelut ovat nykyaikana kriittisessä osassa yritysten ja yhteiskunnan toimintaa, ja niiden merkitys kasvaa edelleen. Palveluiden suuri yhteiskunnallinen merkitys tekee niistä otollisia kohteita hyökkäyksille, joiden motivaationa on usein taloudellinen hyöty. Motivaatio hyökkäysten takana voi olla myös haktivismi tai poliittinen näkemys (Praseed ja Thilagam 2018).

Palvelunestohyökkäykset ovat tyypillisiä verkkokerroksen hyökkäyksiä. Yleisiä verkkokerroksen hyökkäyksiä ovat erilaiset tulvat (engl. *flooding*) kuten ICMP-, SYN- ja UDP-tulvat (Xie ja Yu 2008). Verkkokerroksen hyökkäyksistä ja niiden tunnistamisesta on saatavilla runsaasti tutkimustietoa, mikä on mahdollistanut erilaisten puolustusmekanismien kehittämisen. Verkkokerroksella tapahtuvien palvelunestohyökkäysten puolustuskeinojen kehittyminen on johtanut sovelluskerroksella tapahtuvien edistyneempien palvelunestohyökkäysten kehittymiseen (Dantas, Nigam ja Fonseca 2014; Devi ja Yogesh 2012). Sovelluskerroksen hyökkäykset perustuvat usein esimerkiksi HTTP (engl. *Hypertext Transfer Protocol*) -protokollan

GET -pyynnöillä tapahtuviin tulviin. Tällaisia hyökkäyksiä on hyvin vaikea havaita ajoissa, minkä takia ne ovat merkittävä uhka. Sovelluskerroksella tapahtuva hyökkäys voi tapahtua samanaikaisesti tai näyttää *flash crowd* -ilmiöltä, jossa tavallista suurempi määrä asiakkaita käyttää palvelua samanaikaisesti, vaikka ruuhkautumisen taustalla on vain yksi hyökkääjä (Xie ja Yu 2008).

Tässä kandidaatintutkielmassa perehdytään sovelluskerroksen DoS-/DDoS -hyökkäysten perusteisiin ja eroihin verkkokerroksen vastaaviin hyökkäyksiin. Lisäksi tässä tutkielmassa käydään läpi muutamia tapoja hyökkäyksiltä suojautumiseen. Tutkielma tehdään kirjallisuuskatsauksena aiemman kirjallisuuden pohjalta.

2 Sovelluskerroksen palvelunestohyökkäykset

Palvelunestohyökkäyksissä hyökkäävä taho sulkee todelliset käyttäjät ulos järjestelmästä tai palvelusta. Palvelunestohyökkäykset pyrkivät usein ylikuormittamaan palvelun käytössä olevan kaistaleveyden tai kuluttamaan resursseja. Tämä voidaan tehdä esimerkiksi avaamalla useita yhteyksiä samanaikaisesti (Mantas ym. 2015).

Sovelluskerroksen DDoS-hyökkäykset ovat usein vaikeammin havaittavissa, minkä takia niiden tunnistamiseksi on jouduttu kehittämään uusia menetelmiä sekä jatkojalostamaan jo olemassa olevia. Sovelluskerroksen hyökkäykset voivat perinteisten tulvahyökkäysten lisäksi keskittyä kuluttamaan tiettyä palvelimen resurssia, esim. muistia, kaistanleveyttä, prosessoria tai I/O kaistaa. Kaistanleveyden ruuhkauttamisen lisäksi sovelluskerroksen hyökkäykset voivat olla toistoon perustuvia. Sovelluskerroksen hyökkäykset voivat muistuttaa hetkellistä ruuhkaa, joka tunnetaan *flash crowd* -ilmiönä (Xie ja Yu 2008). Erityisen vaarallisia ja vaikeasti havaittavia sovelluskerroksen hyökkäyksistä tekee niiden samankaltaisuus aitojen käyttäjien kanssa. Sovelluskerroksen palvelunestohyökkäykset eivät tarvitse mitään erityistä heikkoutta, sillä hyökkääjä voi käyttää kaikille käytettävissä olevia ominaisuuksia/resursseja kuten HTTP pyyntöjä/paketteja hyökkäysten toteuttamiseen. Sovelluskerrokselle tyypillisiä ovat niin sanotut low and slow hyökkäykset, kuten slowloris, joka perustuu HTTP GET -pyyntöihin. Tällaisessa hyökkäyksessä hyökkääjä lähettää vajaita HTTP-pyyntöjä, kunnes uusia yhteyksiä ei enää pysty avaamaan (Zargar, Joshi ja Tipper 2013). Koska sovelluskerroksen hyökkäykset on helppo naamioida tavalliseksi verkkoliikenteeksi, niiden havaitseminen on tyypillisesti erittäin vaikeaa. Ainoastaan tarkoitusperä eroaa laillisesta liikenteestä.

3 Hyökkäysten havaitseminen

Sovelluskerroksella tapahtuvien palvelunestohyökkäysten havaitseminen on haastavaa. Monet tunnetut palvelunestohyökkäysten tunnistusmenetelmät tarkkailevat verkkoliikenteen kokonaismäärää, mutta sovelluskerroksella tapahtuvat hyökkäykset eivät välttämättä tarvitse suuria määriä liikennettä saavuttaakseen tavoitteensa. Ne voivat olla tarkemmin suunnattuja kuin verkkokerroksen vastaavat hyökkäykset. Tämä tarkoittaa sitä, että hyökkäys voi kohdistua vain yhteen tiettyyn palvelimen resurssiin, mikä mahdollistaa palvelun kaatamisen jopa pienellä määrällä haitallisia pyyntöjä. Yhden resurssin kaataminen ei välttämättä vaikuta muihin resursseihin, mutta se estää palvelun toiminnan siinä missä koko palvelua vastaan kohdistettu hyökkäys. Tämän vuoksi yhtä resurssia tarkkaileva tunnistusmekanismi ei välttämättä huomaa hyökkäystä jonka kohteena on jokin muu resurssi (Praseed ja Thilagam 2018).

Edellä mainittujen haasteiden vuoksi voidaan todeta, että sovelluskerroksella tapahtuvien hyökkäysten havaitsemiseen ei välttämättä ole yhtä oikeaa tunnistamistapaa. Hyökkäysten luotettavaan havaitsemiseen joudutaan soveltamaan ja yhdistelemään useita eri resursseihin suunnattuja menetelmiä. Tunnettuja havaitsemiskeinoja ovat muun muassa liikenteen, poikkeavuuksien ja käyttäytymisen analysointi (Xie ja Yu 2008). Eräs poikkeavuuksien analysoinnissa käytettävistä työkaluista on pääkomponenttianalyysi (engl. *principal component analysis, PCA*) -aliavaruus, jonka avulla tarkastellaan monimuuttujaista dataa (Najafabadi ym. 2017). Tarkasteltavista muuttujista pyritään määrittelemään pääkomponentit (engl. *principal components*), joiden avulla pystytään saavuttamaan suurin mahdollinen hajonta ja täten selittämään mahdollisimman suuri osuus datasta. Komponentit pyritään muodostamaan siten, että dataa menetetään mahdollisimman vähän (Greenacre ym. 2022). Lakhina, Crovella ja Diot (2004) ovat tutkineet laajalti PCA:n käyttöä poikkeavuuksien tunnistamiseen tietoliikenteessä, ja esittelevät PCA-subspace metodin. Xie ja Yu (2008) käsittelevät PCA:n lisäksi toisenlaista poikkeavuuksien analysointiin käytettävää metodia nimeltä ICA (*Independent Component Analysis*). Siinä missä PCA pyrkii etsimään komponentteja mahdollisimman suurella hajonnalla, ICA pyrkii etsimään komponentit joilla on suurin tilastollinen itsenäisyys (Naik ja Kumar 2011). Xie ja Yu (2008) tekivät kokeita ehdottamallaan puolus-

tusarkkitehtuurilla, joka seuraa verkkoliikennettä käyttäen hyväkseen edellämainittuja PCA- ja ICA-metodeja. Tutkimuksen yhteydessä tehdyt simulaatiot osoittivat että ehdotettu arkkitehtuuri pystyi erottamaan verkkoliikenteessä tapahtuvat hyökkäyksestä johtuvat muutokset myös *flash crowd* ilmiön aikana.

Aidon asiakkaan ja automatisoidun hyökkäyksen erottamiseen on kehitetty myös erilaisia automatisoituja varmistuksia. Esimerkki tällaisesta automatisoidusta varmistuksesta ovat CAPTCHA (engl. *Completely Automated Public Turing test to tell Computers and Humans Apart*) -haasteet (Xie ja Yu 2008). Katabi ja Berger (2005) ehdottavat järjestelmää jossa vain CAPTCHA-haasteen läpäisevät käyttäjät saavat pääsyn palveluun. Samassa tutkimuksessa tiedostetaan myös ettei kyseinen metodi ole täysin ongelmaton, koska CAPTCHA-haasteet saattavat ärsyttää käyttäjiä ja hidastaa palvelun käyttöä.

4 Hyökkäyksiltä suojautuminen

Sovelluskerroksella tapahtuvia hyökkäyksiä vastaan suojautumiseen löytyi monia erilaisia ehdotettuja ja toteutettuja puolustuskeinoja. Monet puolustuskeinot ovat jatkokehitystä verkkokerrokselta tunnetuista puolustuskeinoista. Seuraavaksi käsitellään kaksi tapaa puolustautua sovelluskerroksen palvelunestohyökkäyksiä vastaan. s

4.1 Valikoiva suojaus (SeVen)

Selektiivinen verifikaatio sovelluskerroksella (engl. *Selective Verification in Application Layer, SeVen*) on uudenlainen puolustuskeino, joka perustuu verkkokerroksen hyökkäysten torjumiseen käytettyyn suojaukseen nimeltä ASV (*Adaptive Selective Verification*) (Dantas, Nigam ja Fonseca 2014). Khanna ym. (2008) tekemien simulaatioiden perusteella ASV toimii tehokkaasti verkkokerroksen hyökkäyksiä vastaan. ASV on älykäs verkkoliikenteen suodatin. Se etsii verkkoliikenteestä epäilyttäviä pyyntöjä, ja asettaa tarvittaessa adaptiiviset kaistanleveysmaksut, jolloin asiakas antaa ylimääräistä kaistanleveyttä vastineeksi pääsystä palveluun. Normaalilla asiakkaalla on yleensä saatavilla ylimääräistä kaistanleveyttä, toisin kuin hyökkääjillä joiden voidaan olettaa käyttävän kaiken saatavilla olevan kaistanleveyden itse hyökkäyksen toteuttamiseen (Khanna ym. 2008).

Vaikka ASV on käyttökelpoinen verkkokerroksen hyökkäysten estämisessä, sen kyky tunnistaa haitallista liikennettä ei ole riittävä sovelluskerroksella tapahtuvien hyökkäysten torjumiseen, sillä se olettaa kommunikaation olevan yksinkertaista asiakas-palvelin–vuorovaikutusta (Khanna ym. 2008). Esimerkiksi HTTP POST ja HTTP PRAGMA -hyökkäykset perustuvat HTTP protokollaan, joka on ns. tilariippuvainen (engl. *notion of state*) (Dantas, Nigam ja Fonseca 2014).

ASV:n tavoin SeVen tarkistaa saapuvat pyynnöt valikoivasti, ja pyrkii näin erottamaan lailliset pyynnöt potentiaalisesti haitallisista. Todennuskynnykset ja kriteerit muuttuvat dynaamisesti verkko-olossuhteiden mukaan, jolloin se pystyy tarkemmin erottamaan aidosti haitalliset ja lailliset pyynnöt. SeVen sisältää laajemman ymmärryksen sovelluskerroksen protokollista, kuten HTTP ja niiden tilallisesta luonteesta. SeVen ylläpitää ja hallitsee asiakkaiden ja

palvelimien välisen vuorovaikutuksen tilaa, jolloin voidaan ottaa huomioon suurempia kokonaisuuksia, ja näin ollen ymmärtää myös kontekstin (Dantas, Nigam ja Fonseca 2014).

Saapuvia pyyntöjä ei käsitellä heti, vaan SeVen ylläpitää sisäisiä puskureita, joihin se kerää pyyntöjä kierrokseksi kutsutun ajanjakson ajan. Puskureissa säilytetään käsittelemättömät ja osittain käsitellyt pyynnöt. Kierroksen päätteeksi pyynnöt analysoidaan ja lasketaan todennäköisyysjakauma, joka tallennetaan puskuriin. Todennäköisyysjakauman perusteella voidaan tehdä päätös pyynnön hylkäämiseksi tai pitämiseksi. Tulevia pyyntöjä ei kuitenkaan hylätä ennen kuin puskurissa olevien pyyntöjen määrä ylittää ennalta määritellyn maksimin. Maksimimäärä on sovelluksen määrittämä suurin määrä pyyntöjä jotka on mahdollista käsitellä samanaikaisesti. Mikäli tämä maksimimäärä ylittyy, joudutaan tuleva pyyntö hylkäämään tai jokin puskureissa oleva pyyntö korvataan uudella. Päätös pyyntöjen poistamiseksi tai korvaamiseksi tehdään aina todennäköisyysjakauman perusteella. Kierroksen lopuksi puskuriin jääneet pyynnöt prosessoidaan (Dantas, Nigam ja Fonseca 2014).

Artikkelissaan Dantas, Nigam ja Fonseca (2014) osoittavat simulaatioiden avulla että SeVen pystyy lieventämään sovelluskerroksen tulva- ja slowloris hyökkäyksiä tehokkaasti.

4.2 Hybridipuolustus

Devi ja Yogesh (2012) ehdottavat sovelluskerroksen palvelunestohyökkäysten torjumiseen sopivan järjestelmän, joka perustuu kaksivaiheiseen tarkistukseen. Ehdotetun puolustuksen ensimmäisessä vaiheessa käytetään hyödyksi luottoarvoa, joka on asiakaskohtainen ja annetaan asiakkaan käyttäytymisen perusteella. Tämä luotettavuusarvo annetaan asiakkaalle HTTP-evästeessä. Evästeen avulla asiakas voi tunnistautua ja päästä jonon ohi mikäli luottoarvo on tarpeeksi hyvä. Tämänkaltainen toimintamalli antaa tunnetuille ja luotetuille asiakkaille suuren prioriteetin, kun taas hyökkääjän pääsy palveluun rajoittuu voimakkaasti. Uudelle asiakkaalle annetaan aina huonoin luotettavuusarvo, jota päivitetään tarpeen mukaan.

Alla oleva kaavio kuvaa hybridipuolustuksen toimintaa. Ensimmäisessä vaiheessa asiakkaan luottoarvo tarkistetaan, jos luottoarvo ei ole validi, hylätään pyyntö välittömästi ja yhteys katkaistaan. Muussa tapauksessa siirrytään seuraavaan vaiheeseen, jossa istunnon pyynnöistä lasketaan entropia, joka mittaa pyyntöjen satunnaisuutta. Pyyntöjen entropiaa verrataan

ennalta määriteltyyn satunnaisuuden vertailuarvoon. Mitä suurempi ero vertailuarvoon, sitä todennäköisempää on että kyseessä on hyökkääjä. Jos ero on tarpeeksi suuri annetaan asiakkaalle huonoin mahdollinen luottoarvo ja yhteys katkaistaan. Muussa tapauksessa pyynnön käsittelyä jatketaan ja asiakkaalle annetaan asianmukainen luotettavuusarvo. Pyynnöt käsitellään aina luotettavuusarvon mukaisessa prioriteettijärjestyksessä(Devi ja Yogesh 2012).

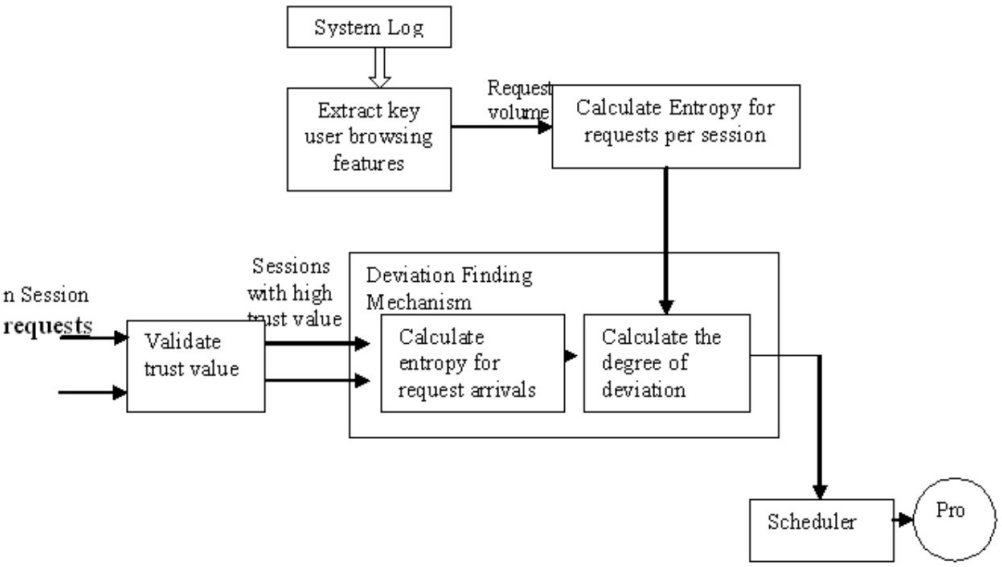

Figure 1. System Architecture

Kuvio 1. Hybridipuolustuksen toiminta vuokaaviossa (Devi ja Yogesh 2012).

5 Yhteenveto

Useissa tutkimuksissa kävi ilmi että verkkokerroksen hyökkäyksiä on tutkittu enemmän, minkä johdosta sovelluskerroksen hyökkäykset ovat nousseet suosioon. Sovelluskerroksen hyökkäyksiä vastaan puolustautumisesta löytyi kiitettävästi materiaalia, vaikka tuoreita artikkeleita oli hieman vaikea löytää. Voidaan todeta että jatkotutkimukselle on aihetta, koska hyökkäykset jatkavat kehittymistä riippumatta siitä kuinka paljon niitä tutkitaan. Erilaisia ehdotettuja ja toteutettuja puolustusmetodeja löytyi myös enemmän kuin tässä tutkielmassa on mielekästä esitellä, joten valitsin tähän kaksi hieman erilaista, mutta on syytä tiedostaa, että muitakin puolustusmenetelmiä on.

Voidaan myös todeta että sovelluskerroksen hyökkäysten havaitseminen on huomattavasti haastavampaa verkkokerroksella tapahtuviin hyökkäyksiin nähden, mikä johtuu erityisesti mahdollisisita hitaista ja kohdennetuista hyökkäyksistä. Kohdennetut ja hitaat hyökkäykset eivät aiheuta suuria määriä liikennettä, jonka seuraamiseen monet tunnetut metodit perustuvat. Monet verkkokerroksella tutkitut ja toimivat metodit eivät suoraan toimi sovelluskerroksella, mutta niitä on mahdollista jatkojalostaa toimimaan myös sovelluskerroksella. Esimerkiksi SeVen on syntynyt, kun ASV:n toimintaa laajennettiin ymmärtämään sovelluskerroksen tilariippuvaisia protokollia.

Lähteet

Dantas, Yuri Gil, Vivek Nigam ja Iguatemi E Fonseca. 2014. “A selective defense for application layer ddos attacks”. Teoksessa *2014 IEEE Joint Intelligence and Security Informatics Conference,* 75–82. IEEE. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1109/JISIC.2014.21.

Devi, S Renuka, ja P Yogesh. 2012. “A hybrid approach to counter application layer DDoS attacks”. *International Journal on Cryptography and Information Security (IJCIS)* 2 (2). Viitattu 30. huhtikuuta 2024. https://doi.org/10.5121/ijcis.2012.2204.

Greenacre, Michael, Patrick JF Groenen, Trevor Hastie, Alfonso Iodice d’Enza, Angelos Markos ja Elena Tuzhilina. 2022. “Principal component analysis”. *Nature Reviews Methods Primers* 2 (1): 100. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1039/C3AY41907J.

Katabi, Srikanth Kandula Dina, ja Matthias Jacob Arthur Berger. 2005. “Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds”. Teoksessa *NSDI’05: 2nd Symposium on Networked Systems Design & Implementation USENIX Association.* Citeseer. Viitattu 30. huhtikuuta 2024. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=278970e70b7c89a4b46fdea0f5dbcd503ab4e82f.

Khanna, Sanjeev, Santosh S Venkatesh, Omid Fatemieh, Fariba Khan ja Carl A Gunter. 2008. “Adaptive selective verification”. Teoksessa *IEEE INFOCOM 2008-The 27th Conference on Computer Communications,* 529–537. IEEE. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1109/INFOCOM.2008.101%22.

Lakhina, Anukool, Mark Crovella ja Christophe Diot. 2004. “Diagnosing network-wide traffic anomalies”. *ACM SIGCOMM computer communication review* 34 (4): 219–230. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1145/1030194.1015492.

Mantas, Georgios, Natalia Stakhanova, Hugo Gonzalez, Hossein Hadian Jazi ja Ali A Ghorbani. 2015. “Application-layer denial of service attacks: taxonomy and survey”. *International Journal of Information and Computer Security* 7, not. 2-4 (marraskuu): 216–239. Viitattu 30. huhtikuuta 2024. https://www.inderscienceonline.com/doi/abs/10.1504/IJICS.2015.073028.

Naik, Ganesh R, ja Dinesh K Kumar. 2011. “An overview of independent component analysis and its applications”. *Informatica* 35 (1). Viitattu 30. huhtikuuta 2024. https://www.informatica.si/index.php/informatica/article/viewFile/334/333.

Najafabadi, Maryam M, Taghi M Khoshgoftaar, Chad Calvert ja Clifford Kemp. 2017. “User behavior anomaly detection for application layer ddos attacks”. Teoksessa *2017 IEEE International Conference on Information Reuse and Integration (IRI),* 154–161. IEEE. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1109/IRI.2017.44.

Othman, Raja Azrina Raja. 2000. “Understanding the various types of denial of service attack”. *Business Week Online,* viitattu 30. huhtikuuta 2024. https://www.cybersecurity.my/data/content_files/13/72.pdf.

Praseed, Amit, ja P Santhi Thilagam. 2018. “DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications”. *IEEE Communications Surveys & Tutorials* 21 (1): 661–685. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1109/COMST.2018.2870658.

Xie, Yi, ja Shun-Zheng Yu. 2008. “Monitoring the application-layer DDoS attacks for popular websites”. *IEEE/ACM Transactions on networking* 17 (1): 15–25. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1109/tnet.2008.925628.

Zargar, Saman Taghavi, James Joshi ja David Tipper. 2013. “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks”. *IEEE communications surveys & tutorials* 15 (4): 2046–2069. Viitattu 30. huhtikuuta 2024. https://doi.org/10.1109/SURV.2013.031413.00127.