Marko Andersson

IPS-järjestelmien tukeminen Netflown avulla

Tietotekniikan
pro gradu -tutkielma
29. marraskuuta 2006

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Marko Andersson
Yhteystiedot: maenande@cc.jyu.fi
Työn nimi: IPS-järjestelmien tukeminen Netflown avulla
Title in English: Supporting IPS-systems with Netflow
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 140
Tiivistelmä: Verkkoon liitettyjä koneita uhataan jatkuvasti erilaisin keinoin. Verkottuneesta maailmasta löytyy paljon tahoja, jotka pyrkivät murtautumaan verkon muihin laitteisiin tai häiritsemään niiden toimintaa. Tutkielmassa esitetään joitakin verkon yleisimpiä uhkia, sekä niiden havaitsemista ja torjumista IDS-järjestelmien avulla. Tutkielmassa kuvaillaan erilaisia IDS-järjestelmiä ja niiden toimintaa sekä heikkouksia. Uhkia voidaan havaita ja jäljittää myös Netflow-datan avulla, jolloin voidaan tukea IDS-järjestelmien toimintaa. Tutkimusongelmana on lähimmän reitittimen löytäminen, jonka liitynän takana saastunut laite sijaitsee, Netflow-datan avulla. Lopussa määritellään rajapinnat saastuneiden laitteiden liikennöinnin estämiseen Netwrapperin avulla.
English abstract: Devices which are connected to networks are often threatened in many different ways. There are people operating in the wired world, who are trying to break into other devices in the network or interfere to their operation. This thesis introduces some of the most common threats in networks and outlines different manners to detect and prevent them with IDS-systems. This thesis describes different types of IDS-systems and their operations and vulnerabilities. Threats can also be detected and traced back with Netflow-data, and this way support the IDS-systems. The research problem is to find the nearest router, whose interface the compromised device is connected to, with the aid of the Netflow-data. At the end of the thesis, application interfaces are defined into the Netwrapper-product to prevent compromised devices from operating in the network.
Avainsanat: Tietoturva, palomuurit, IDS-järjestelmät, heikkoudet, Netflow, madot, Netwrapper.
Keywords: Data security, firewalls, IDS-systems, vulnerabilities, Netflow, worms, Netwrapper.

Sanasto

ACK *Acknowledgement.* TCP-protokollan lähettämä kuittausviesti, joka esitetään tietyllä bitillä (lippu) TCP-paketissa.

ACL *Access Control List.* (suom. *Pääsylistat*) Verkon laitteisiin määriteltyjä listoja, joiden avulla voidaan kontrolloida verkossa kulkevaa liikennettä.

ASCII *American Standard Code for Information Interchange.* Tietokoneiden merkistö, joka sisältää englannin kielen kirjaimet, numerot, joukon välimerkkejä ja joitakin ohjauskoodeja. Lähes kaikki nykyisin yleisessä käytössä olevat tietokoneiden merkistöt ovat ASCII-yhteensopivia eli niiden 128 ensimmäistä merkkiä ovat samat kuin ASCIIssa.

ARP *Address Resolution Protocol.* Selvittää IP-osoitetta vastaavan MAC-osoitteen.

BIND *Berkeley Internet Name Domain.* Yleisin Internetissä käytetty DNS-palvelinohjelmisto.

CGI *Common Gateway Interface.* Web-ympäristön tekniikka, jonka avulla selain voi välittää dataa palvelimella suoritettavalle ohjelmalle. CGI määrittää standardin tähän datan välitykseen.

Cisco IOS *Internetwork Operating System.* Ciscon käyttämä käyttöjärjestelmä verkon laitteissa. Sisältää reititys- ja kytkentätoimintoja integroituna moniajokäyttöjärjestelmään.

CSMA/CD *Carrier sense multiple access with collision detection.* Tietoliikenteen siirtotien varausmenetelmä, jolla useat lähettävät tietokoneet jakavat samaa siirtotietä. CSMA/CD on perinteinen Ethernet-verkkojen tapa jakaa verkko käyttäjien kesken.

DDoS *Distributed Denial of Service.* Hajautettu palvelunestohyökkäys (DoS).

DIDS *Distributed Intrusion Detection Systems.* Hajautettu tunkeutumisenhavaitsemisjärjestelmä.

DNS *Domain Name System.* (suom. *Nimipalvelujärjestelmä*) Muuttaa Internetin verkkotunnukset IP-osoitteiksi.

DoS *Denial of Service.* (suom. *Palvelunestohyökkäys*) Hyökkääjän toimesta tapahtuva verkon laitteen tai palvelun lamauttaminen niin, ettei laite tai palvelu ole käytettävissä.

FIN *File Transfer Protocol.* TCP-yhteyden lopetusviesti, joka esitetään tietyllä bitillä (lippu) TCP-paketissa.

FTP *File Transfer Protocol.* TCP-protokollaa käyttävä asiakas-palvelin periaatteella toimiva tiedonsiirtoprotokolla.

HIDS *Host-based Intrusion Detection Systems.* Konetason tunkeutumisenhavaitsemisjärjestelmä. Ohjelmisto, joka sijoitetaan yleensä suojattavaan koneeseen.

HTTP *Hypertext Transfer Protocol.* TCP-protokollan päällä toimiva protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.

ICMP *Internet Control Message Protocol.* TCP/IP-pinon kontrolliprotokolla virheviestien lähettämiseen.

IDS *Intrusion Detection Systems.* Tunkeutumisenhavaitsemisjärjestelmä.

IP *Internet Protocol.* Verkkokerroksen protokolla, joka huolehtii IP-pakettien kuljettamisesta kytkentäisissä verkoissa.

IPS *Intrusion Prevention Systems.* Tunkeutumistenestojärjestelmä. Järjestelmä toimii palomuurin tavoin pystyen estämään haitallisen liikenteen pääsyn lävitseen.

IPSec *IP Security Architecture.* Joukko TCP/IP-perheeseen kuuluvia tietoliikenneprotokollia Internet-yhteyksien turvaamiseen. Nämä protokollat tarjoavat salauksen, osapuolten todennuksen ja tiedon eheyden varmistamisen.

IRC *Internet Relay Chat.* Internetissä välitettävä keskusteluprotokolla.

MAC *Media Access Control.* Siirtoyhteyskerroksella käytetty osoitteistus kahden laitteen väliseen liikennöintiin.

MD5 *Message Digest 5.* Tiivistealgoritmi, joka tuottaa vaihtelevanmittaisesta syötteestä 128-bittisen tiivisteen.

MIB *Management Information Base.* Virtuaalinen tietokanta, joka sisältää esimerkiksi verkon laitteen tietoja. Tietoja voidaan lukea SNMP-protokollan avulla.

MITM *"Man in the Middle".* Välimieshyökkäys, jossa hyökkääjä voi lukea, kirjoittaa ja muuttaa kahden laitteen välistä liikennettä.

NetBIOS *Network Basic Input/Output System.* Rajapinta verkon laitteiden keskinäiselle kommunikoinnille lähiverkon yli.

NIDS *Network-based Intrusion Detection Systems.* Verkkotason tunkeutumisenhavaitsemisjärjestelmä. Sijoitetaan yleensä reitittimen yhteyteen tarkkailemaan verkon liikennettä.

Nmap Ohjelma verkon tietoturvan kartoitukseen.

OSPF *Open Shortest Path First.* Suosituin avoimiin standardeihin perustuva organisaation sisäinen TCP/IP-verkkojen reititysprotokolla.

Promiscuous Tila, jossa verkkosovitin lähettää kaikki vastaanottamansa paketit prosessorin käsiteltäväksi, eikä vain niitä, jotka ovat osoitettu ko. laitteelle.

RIP *Routing Information Protocol.* Yleisin Internetissä käytetty reititysprotokolla.

RST *Reset.* Nollata. RST-lippu TCP-protokollassa.

Skripti Pieni tietokoneohjelma. Esimerkiksi WWW-selaimen suorittamia ohjelmanpätkiä.

Snmpwalk Ohjelma SNMP-protokollan avulla tapahtuvaan tietojen lukemiseen verkon laitteista.

SSH *Secure Shell.* TCP-protokollan päällä toimiva protokolla, joka mahdollistaa turvallisen ja salatun yhteyden kahden osapuolen välille.

SYN *Synchronize.* TCP-protokollan käyttämä synkronointiviesti, joka esitetään tietyllä bitillä (lippu) TCP-paketissa. Käytetään TCP-yhteyttä luotaessa vuoronumeroiden synkronointiin.

TCP *Transmission Control Protocol.* Tietoliikenneprotokolla, joka toimii IP-protokollan päällä. Luo luotettavia yhteyksiä verkon yli muihin laitteisiin.

TFTP *Trivial File Transfer Protocol.* Hyvin yksinkertainen tiedostonsiirtoprotokolla. Käytetään vähämuistisissa laitteissa.

TTL *Time to Live.* IP-paketin elinaika verkossa. Jokainen verkon laite joka käsittelee pakettia, vähentää TTL arvoa yhdellä. Kun TTL-arvo saavuttaa arvon nolla, paketti tuhotaan.

Traceroute Työkalu paketin reitin selvittämiseen IP-verkossa. Käyttää hyväkseen paketin TTL-arvoa ja verkon laitteiden lähettämiä virheviestejä.

UDP *User Datagram Protocol.* Tietoliikenneprotokolla, joka toimii IP-protokollan päällä. Käytetään tiedonsiirtoon kahden laitteen välillä. Ei takaa tiedon perillemenoa.

Unicode Merkistöstandardi, joka kattaa suurimman osan maailman kirjoitettujen kielten käyttämistä merkeistä. Unicode määrittelee yksilöivän koodiarvon yli 90 000 erilaiselle kirjoitusmerkille.

UTF-8 Unicoden vaihtelevanpituinen koodaustapa. Sen etuna on osittainen yhteensopivuus vanhempien järjestelmien kanssa, jotka käsittelevät merkkejä kahdeksanbittisinä tavuina. UTF-8 on rakennettu siten, että ASCII-merkistöön kuuluvat merkit säilyvät siinä samoina kuin ASCII:ssa.

WEP *Wired Equivalent Privacy.* Ensimmäinen langattomien verkkojen liikenteen salaukseen käytetty menetelmä.

Sisältö

Sanasto **i**

1 Johdanto **1**
1.1 Tutkimusongelma . . . 2
1.2 Aiemmat tutkimukset . . . 2
1.3 Tutkielman rakenne . . . 2

2 Yleistä verkon turvattomuudesta **4**
2.1 Verkon aktiivilaitteiden turvallisuus . . . 4
2.1.1 Löytäminen ja tunnistaminen . . . 5
2.1.2 Etähallinta . . . 6
2.1.3 Cisco-reitittimien heikot salasanat . . . 7
2.2 Verkkojen salakuuntelu . . . 8
2.2.1 Salasanojen haistelu . . . 8
2.2.2 Man in the middle . . . 9
2.2.3 Istunnon kaappaaminen . . . 11
2.2.4 Langattomat verkot . . . 12
2.3 Palomuurien ohittaminen . . . 14
2.3.1 Palomuurin tunnistaminen . . . 14
2.3.2 Palomuurin läpi tutkiminen . . . 16
2.3.3 Tilattoman palomuurin ohittaminen . . . 17
2.3.4 FTP-huijaus . . . 18
2.3.5 IRC-haavoittuvuus . . . 19
2.3.6 Huonosti konfiguroidut pääsylistat . . . 20
2.4 Palvelunestohyökkäykset . . . 20
2.4.1 Smurf . . . 22
2.4.2 SYN-hukuttaminen . . . 23
2.4.3 DNS-hyökkäykset . . . 24

3 IDS-järjestelmät **25**
3.1 Mikä on IDS? . . . 25
3.2 IDS-järjestelmätyypit . . . 27

3.2.1 Verkkotason havaitsemisjärjestelmä . . . 27
3.2.2 Konetason havaitsemisjärjestelmä . . . 28
3.2.3 Hajautetut havaitsemisjärjestelmät . . . 29
3.2.4 Langattomat IDS-järjestelmät . . . 30
3.3 Analysointimenetelmät . . . 32
3.3.1 Väärinkäytösten tunnistaminen . . . 33
3.3.2 Poikkeuksien havaitseminen . . . 34
3.4 Hälytysten käsitteleminen . . . 35
3.5 Hyökkäyksiin reagointi . . . 36
3.6 IDS-järjestelmän sijoittaminen . . . 37
3.6.1 NIDS-järjestelmän sijoittaminen . . . 38
3.6.2 HIDS-järjestelmän sijoittaminen . . . 39
3.6.3 DIDS-järjestelmän sijoittaminen . . . 40
3.7 IDS-järjestelmien kehittäminen . . . 40
3.7.1 Hi-RDA . . . 40
3.7.2 Tekoäly . . . 41
3.8 Lokien analysointi . . . 43

4 IDS-järjestelmien heikkoudet 46
4.1 Lisäämishyökkäys . . . 47
4.2 Välttelyhyökkäys . . . 48
4.3 Heikkoudet verkkokerroksessa . . . 48
4.3.1 IP-paketin otsikoiden väärentäminen . . . 49
4.3.2 Uudelleenkokoamishyökkäykset . . . 50
4.4 Heikkoudet siirtokerroksessa . . . 53
4.4.1 TCP-otsakkeiden väärentäminen . . . 53
4.4.2 TCP:n tahdistus . . . 54
4.5 Heikkoudet sovelluskerroksessa . . . 57
4.5.1 Merkistökoodaukset . . . 57
4.5.2 Tiedostopolku . . . 58
4.5.3 HTTP-otsikot . . . 59
4.6 Vastatoimenpiteet . . . 59
4.6.1 Normalisoija . . . 60
4.6.2 Hunajapurkit . . . 60

5 IDS-järjestelmiä 62
5.1 Snort . . . 62
5.1.1 Toiminta . . . 63

5.1.2 Säännöt . . . 65
5.2 Cisco IDS 4200 -sarja . . . 67
5.2.1 Cisco HIDS . . . 68
5.2.2 IDS-laitteiden hallinta . . . 69
5.2.3 Säännöt . . . 69
5.3 Bro . . . 70
5.3.1 Toiminta . . . 70
5.3.2 Bro-kieli . . . 72

6 Netflow **74**
6.1 Yleistä . . . 74
6.2 Datan kerääminen . . . 75
6.3 Analysointi . . . 76
6.3.1 Top N ja vertailukohta -analyysi . . . 76
6.3.2 Havaitseminen sääntöjen avulla . . . 77
6.4 Hyökkäyksien havaitseminen Netflown avulla . . . 78
6.4.1 Ping Sweep . . . 79
6.4.2 Porttiskannaukset . . . 79
6.4.3 Palvelunestohyökkäys . . . 79
6.5 Matojen havaitseminen . . . 80
6.6 Netflown käyttö IDS:n tukena . . . 83
6.7 Työkalut . . . 85
6.7.1 Flow-tools . . . 85
6.7.2 Flowd . . . 88

7 Sisäisiin uhkiin reagoiminen **91**
7.1 Cisco ACL . . . 92
7.2 Laila-verkon topologia . . . 94
7.3 Hyökkäyksen lähteen selvittäminen . . . 95
7.3.1 Indeksiin perustuva jäljitys . . . 96
7.3.2 Reitittimien tietoihin perustuva jäljitys . . . 98
7.4 Ongelmia jäljityksessä . . . 102
7.4.1 Reitittimen takana olevat kytkimet ja keskittimet . . . 102
7.4.2 Netflow-datan puuttuminen reitiltä . . . 104
7.4.3 Kaksi samaa IP-osoitetta . . . 105
7.5 Netwrapper . . . 107
7.5.1 Aktiivilaitteen liitynnän sulkeminen . . . 107
7.5.2 ACL-säännön luominen . . . 109

8 Yhteenveto **111**

Lähteet **113**

Liitteet

A Config.py-tiedoston lähdekoodi **119**

B Trace.py-tiedoston lähdekoodi **121**

1 Johdanto

Internet on kasvanut maailmanlaajuiseksi verkoksi, joka yhdistää ihmiset eri puolilta maailmaa. Näihin lukeutuvat myös henkilöt, jotka tietoisesti käyttävät maailmanlaajuista verkkoa vääriin tarkoituksiin ja uhkaavat sen avulla tavallisia ihmisiä, jotka käyttävät verkon palveluja hyväkseen. Tästä syystä Internetiin kytkettyjen koneiden turvallisuus on kokoajan uhattuna tavalla tai toisella. Kuten muissa yhteisöissä, Internetissäkään kaikkia uhkia vastaan ei voida suojautua täydellisesti.

Aiemmin murtautumisia ja väärinkäytöksiä harjoittivat vain pieni osa ihmisistä, sillä se vaati syvällisiä tietoja ja taitoja verkon ja laitteiden toiminnasta, joita vain harvoilla oli. Nykyään hyökkäyksiä voidaan tehdä muiden tekemillä valmiilla työkaluilla, joten tietotaitoa hyökkäyksistä ei enää vaadita ja näin hyökkäyksiä voivat tehdä lähes kaikki halukkaat.

Tänä päivänä yritykset turvautuvat yhä enemmän verkkoon ja sen tuomiin mahdollisuuksiin kuin kertaakaan aiemmin. Kilpailukyvyn parantamiseksi kauppaa pyritään käymään verkossa ja samalla verkostoitumaan yhteistyökumppaneiden kanssa. Myös erilaiset etäyhteydet työntekijöiden joustavaa liikkumista varten ovat arkipäivää. Yhdessä jatkuvan hyökkäysuhan kanssa nämä ovat synnyttäneet suuret markkinat erilaisille tietoturvatuotteille, jotka lupaavat suojata yritysten verkkoja erilaisia uhkia vastaan. Myös kotikäyttäjälle suunnatut tuotteet ovat kehittyneet ja niihin on integroitu ominaisuuksia erilaisista suojausratkaisuista, joita käytetään suurissakin verkoissa.

Palomuurit ovat olleet pitkään välttämättömiä verkkoon kytkettyjen laitteiden suojaamisessa, mutta yhä uudet ja monimutkaisemmat hyökkäystavat ovat synnyttäneet tarpeen järjestelmille, jotka tarkkailevat verkossa liikkuvaa liikennettä, joka on päässyt palomuurin lävitse, ja raportoivat löytämistään uhista ylläpitäjille tai estävät epäilyttävän liikenteen verkossa. Tällaisia ovat IDS-järjestelmät (engl. *Intrusion Detection Systems*), jotka tarkkailevat verkossa liikkuvia paketteja etsien niistä merkkejä hyökkäyksistä ja väärinkäytöksistä. Nämäkään laitteet eivät suojaa verkkoa täydellisesti ja niitä voidaan ohittaa, kuten useimpia kehitettyjä suojausmekanismeja.

1.1 Tutkimusongelma

Verkkojen kapasiteettien kasvaessa IDS-järjestelmät tarvitsevat yhä enemmän resursseja verkon liikenteen tutkimiseen. Ongelmaa voidaan kiertää sijoittamalla verkkoon useita IDS-järjestelmiä rinnakkain, mutta tällöin niiden hallinta ja ylläpito hankaloituu. Eräs tapa on jakaa IDS-järjestelmien taakkaa muiden tekniikoiden avulla. Esimerkiksi verkon aktiivilaitteiden lähettämää Netflow-dataa analysoimalla voidaan siitä etsiä tiettyjä uhkia ja haavoittuvuuksia, joita IDS-järjestelmät etsivät, ja näin helpottaa IDS-järjestelmien kuormitusta.

Pro Gradu -tutkielmassa selvitetään Netflow-datan soveltuvuutta uhkien havaitsemiseen ja IDS-järjestelmien tukemiseen. Netflow-datan avulla voidaan toteuttaa IDS:n toiminnallisuutta kustannustehokkaasti verkoissa, jotka koostuvat Netflow-dataa lähettävistä reitittimistä. Tutkielmassa käsitellään myös saastuneiden laitteiden havaitsemista, niiden sijainnin selvittämistä suurissa verkoissa Netflow-datan avulla sekä niiden liikennöinnin estämistä mahdollisimman aikaisessa vaiheessa, jotta esimerkiksi mato ei pääse saastuttamaan verkon muita laitteita.

1.2 Aiemmat tutkimukset

Useita tutkimuksia ja artikkeleita on kirjoitettu aiemmin Netflow-datan käytöstä IDS-järjestelmien tukemiseen ja uhkien havaitsemiseen, mutta saastuneiden laitteiden etsimistä Netflown avulla käsitteleviä tutkimuksia ei juuri ole. Netflow-datan käytöstä erilaisten hyökkäysten havaitsemiseen kuvataan esimerkiksi Tsang-Long Paon ja Po-Wei Wangin tutkimuksessa ”NetFlow Based Intrusion Detection System” [51] sekä Gong Yimingin artikkeleissa ”Detecting Worms and Abnormal Activities with NetFlow, Part 1 & 2” [49] [50].

Rob Thomas on kirjoittanut Netflown käytöstä väärennetyillä IP-osoitteella liikennöivien laitteiden jäljittämisestä artikkelissaan ”Tracking Spoofed IP Addresses Version 2.0” [1]. Artikkelissa kuvataan manuaalista jäljitystä, jossa seuraavaa laitetta etsitään kirjautumalla verkon reitittimeen ja tutkimalla reitittimen Netflow- sekä reititystietoja. Artikkelissa kuvattu jäljitys vaatii ihmisen toimia, joten sitä ei voida käyttää automaattiseen jäljittämiseen. Periaatteet ovat kummassakin samoja.

1.3 Tutkielman rakenne

Luku 2 kuvaa yleisesti nykyistä verkon turvattomuutta sekä eri verkon aktiivilaitteiden turvallisuutta. Luvussa käsitellään myös verkkojen salakuuntelua, Palomuu-

rien ohittamista sekä palvelunestohyökkäyksiä. Luku 3 kuvaa yleisellä tasolla IDS-järjestelmiä, niissä käytettyjä tekniikoita ja analysointimenetelmiä, IDS-järjestelmien tuottamia hälytyksiä sekä niiden reagoimista hyökkäyksiin. Luvussa käsitellään IDS-järjestelmien sijoittaminen verkkoon, niiden kehittäminen sekä lokien analysointi. IDS-järjestelmien heikkouksia sekä niiden ohittamista kuvataan luvussa 4, kuten erilaiset lisäämis- ja välttelyhyökkäykset sekä heikkoudet eri protokollapinon tasoilla. Luvun lopussa esitetään muutama vastatoimenpide, joilla voidaan ehkäistä IDS-järjestelmien huijaamisyrityksiä. Luvussa 5 on esitelty muutamia markkinoilla olevia kaupallisia ja avoimen lähdekoodin IDS-järjestelmiä, sekä kuvailtu niiden toimintaa. Netflow ja sen toiminta on kuvattu luvussa 6. Luvussa käsitellään Netflown kerääminen, analysointitapoja sekä hyökkäysten ja matojen havaitsemista Netflow-datan avulla. Luvussa käsitellään myös Netflown käyttö IDS-järjestelmien tukena sekä esitellään kaksi Netflow-voiden käsittelyyn tehtyä ohjelmaa. Luku 7 käsittelee Netflown käyttöä uhkien paikallistamiseen ja niiden estämiseen. Luvussa esitellään Pythonilla toteutettu jäljitysohjelma, joka jäljittää verkon laitteita Netflow-voiden avulla. Luvussa käsitellään jäljityksen ongelmia sekä määritellään rajapintakutsut Netwrapper-järjestelmään, joiden avulla voidaan automaattisesti estää saastuneen laitteen liikennöinti verkossa konfiguroimalla reitittimien liityntöjä.

2 Yleistä verkon turvattomuudesta

Internetin räjähdysmäinen kasvu viime vuosituhannen lopulla aiheutti Internetin rakenteessa hyvin voimakkaan muutoksen. Ennen useimpia verkon palvelimia ylläpiti asiantunteva henkilöstö, joilla oli perustiedot tietoturvasta. Hyökkäyksiä tapahtui silloinkin, mutta kohteet olivat harvassa.

Nykyään kuka tahansa voi kytkeä tietokoneensa osaksi Internetiä ja päästä osalliseksi sen tarjoamista palveluista. Nämä ihmiset ovat harvoin perillä tietoturvan tärkeydestä sekä sen puuttumisen aiheuttamista haittavaikutuksista. Tästä syystä potentiaalisia kohteita murtautujille ja ilkivallan tekijöille on huikea määrä. Tuhansien koneiden zombie- ja bottiarmeijat ovat nykypäivää ja niitä käytetään häikäilemättä hyökkääjien tarkoituksiin.

Internetin kasvu on johtanut ympärivuorokautiseen hyökkäysuhkaan yhtiöiden ja muiden tahojen tietojärjestelmiä vastaan. Hyökkääjien motiivit vaihtelevat suuresti näyttämisen halusta ja maineen hakemisesta tihutöiden kautta aina ammattimaiseen vakoiluun ja tiedon varastamiseen. Nykyään huomattavan moni tietojärjestelmä sisältää ja käsittelee arvokkaita tai luottamuksellisia tietoja, kuten luottokorttien numeroita, henkilötietoja ja muuta vastaavaa tietoa. Myös moni järjestelmä on riippuvainen tietotekniikasta, kuten erilaiset kuljetusjärjestelmät, pankkiautomaatit, puhelinverkot, sähköjärjestelmät ja niin edelleen. Näiden järjestelmien vikaantuminen voisi aiheuttaa huomattavia vahinkoja ja taloudellisia menetyksiä tai pahimmassa tapauksessa horjuttaa yhteiskunnan perusteita.

Vaikka verkot on suojattu palomuureilla, kehitetään uusia hyökkäysmuotoja, joilla voidaan ohittaa palomuureja ennen kuin hyökkäyksiin ehditään reagoimaan. Myös sisäverkossa voi tapahtua hyökkäyksiä tai väärinkäytöksiä, joita palomuurit eivät kykene estämään. Varsinkin erilaisten matojen räjähdysmäiset leviämiset ovat suuri uhka verkoille.

2.1 Verkon aktiivilaitteiden turvallisuus

Luvussa käsitellään verkon aktiivilaitteita (kytkin, reititin, jne) ja niiden turvallisuutta. Aktiivilaitteet muodostavat verkkojen selkärangan ja toimivat verkkoliikenteen solmukohtina. Jos hyökkääjä saa haltuunsa aktiivilaitteen, voivat kaikki yrityksen arkaluontoiset tiedot olla hyökkääjän käsissä. Tästä syystä verkon laitteiden

oikeaan konfigurointiin ja suojaamiseen tulisi kiinnittää huomiota.

2.1.1 Löytäminen ja tunnistaminen

Reitittimet ja muut laitteet voidaan havaita käyttämällä *traceroute*-apuohjelmaa, joka listaa kaikki verkon laitteet kahden IP-osoitteen välillä käyttäen hyväksi IP-paketin TTL-arvoa ja kuuntelemalla verkon laitteiden ICMP-vastauksia. [2]

Skannaamalla löydettyjen reitittimien portteja *nmap*-ohjelmalla[1], voidaan joissakin tapauksissa päätellä avoimien porttien perusteella reitittimen merkki ja malli. Esimerkiksi Ciscon reitittimet voidaan tunnistaa avoimista TCP-porteista 2001, 4001, 6001 ja 9001. *Nmap*-ohjelma sisältää ominaisuuden, jonka avulla se yrittää arvata kohdelaitteen käyttöjärjestelmän nimen ja version avoimien porttien perusteella. Komennolla `nmap -O -n [kohdelaitteen_ip]` [3] nmap suorittaa käyttöjärjestelmän arvauksen: [4]

```
MAC Address: 00:05:5D:5F:5D:E7 (D-Link Systems)
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
Too many fingerprints match this host to give specific
OS details
```

Verkon laitteita voidaan myös tunnistaa niiden lähettämien bannerien avulla. Bannerit ovat vakiotyyppisiä vastauksia, joita lähetetään esimerkiksi käyttäjän ottaessa yhteyttä laiteen etähallintaporttiin. Laite voi ilmoittaa bannerissa nimensä ja versionsa tai laite voidaan tunnistaa uniikin bannerin perusteella.

Ciscon reitittimien pääsylistoilla (engl. *Access list, ACL*) voidaan uloin reititin konfiguroida suodattamaan ICMP-viestit, joissa IP-paketin TTL-kenttä saa arvon nolla. Tällöin traceroute-ohjelmalla ei voida havaita sisäverkon reitittimiä, sillä niiden lähettämät ICMP-viestit tuhotaan. Komennolla

```
access-list 101 deny icmp any any 11 0
```
[5]

reititin suodattaa kaikki ICMP-viestit, jotka ovat tyyppiä 11, koodilla nolla (TTL equals 0 during transit). Käyttöjärjestelmän arvaamisen estämiseen ei ole helppoja toimenpiteitä. Arvaaminen perustuu jokaisen käyttöjärjestelmän uniikkiin TCP/IP-pinon toteutukseen ja sen muuttaminen saattaa haitata käyttöjärjestelmän toimintaa. Bannereista tunnistamista voi vähentää estämällä kirjautuminen laitteiden etähallintaportteihin. [4]

[1] http://www.insecure.org/nmap/

2.1.2 Etähallinta

Lähes kaikkia verkon aktiivilaitteita voi etähallita SNMP-protokollan avulla. Kuten aiemmin todettiin, SNMP sisältää yhteisöjä, joiden nimet toimivat myös salasanoina. SNMP:n ensimmäisessä versiossa yhteisön nimi on selkotekstinä SNMP-paketissa ja versiosta kaksi lähtien se on kryptattu md5-sekasummalla. Oletusasetuksilla laitteiden yhteisönimet ovat usein luku-yhteisölle *public* ja luku-kirjoitus-yhteisölle *write* tai *private*. Jos näitä nimiä ei ole muutettu, voi kuka tahansa päästä reitittimiin käsiksi oletusnimien avulla. Selkokielisiä yhteisönimiä voidaan haistella verkkoliikenteestä (luku 2.2), jonka jälkeen tietojen lukeminen on helppoa. Pelkkä tietojen lukeminenkin voi antaa hyödyllistä tietoa, kuten verkkolaitteen valmistajan ja version:

```
lucifer:~> snmpwalk -v 1 192.168.1.1 -c public system
SNMPv2-MIB::sysDescr.0 = STRING: P-335/P-335WT
```

Reitittimissä voi olla takaovia, joiden kautta laitevalmistajat pääsevät reitittimelle, jos järjestelmänvalvoja on lukinnut järjestelmän. Laitteissa on myös valmiita oletustilejä, jotka muodostavat suuren uhan, ellei niiden nimiä tai salasanoja vaihdeta. Jotkin oletustilit voivat tarjota jopa järjestelmänvalvojan oikeudet ilman salasanaa. Usean valmistajan oletustilit ja salasanat ovat julkisesti esillä Internetissä[2], mistä hyökkääjät saavat ne tietoonsa. [4]

Useiden verkon laitteiden konfigurointitiedostoja voidaan varmuuskopioida ja palauttaa TFTP-protokollan (*Trivial File Transfer Protocol*) avulla. TFTP-protokolla toimii UPD-portissa 69 ja sen kautta päästään käsiksi laitteen konfigurointitiedostoon. Jos laitteen konfigurointitiedoston nimi tiedetään, se voidaan ladata laitteelta ja tällöin saadaan selville yhteisönimet sekä pääsyntarkistusluettelot. [4]

Reitittimien ohjelmistoista löydetään jatkuvasti tietoturva-aukkoja, jotka voivat olla hyvinkin vaarallisia. Niiden avulla hyökkääjä voi saada koko reitittimen hallintaansa tai ajaa reitittimessä omia komentoja. Esimerkiksi Cisco IOS versiot 11.1-11.3 sisältävät puskuriylivuodon TFTP-protokollan tiedostonnimissä, jonka avulla reitittimessä voidaan ajaa omaa koodia [6]. Yrittämällä lukea TFTP-protokollan avulla tiedostoa, jonka nimessä on yli 700 merkkiä, saadaan reititin kaatumaan. Antamalla oikein muotoiltua koodia tiedostonnimessä, voidaan reitittimen konfiguraatiotiedosto korvata omalla tiedostolla, jossa esimerkiksi salasanat on poistettu ja näin hyökkääjä pääsee kirjautumaan reitittimeen ylläpitäjänä. [7]

[2]Eräs lista löytyy osoitteesta http://www.phenoelit.de/dpl/dpl.html

SNMP-protokollan avulla tapahtuvat väärinkäytökset voidaan estää poistamalla SNMP-protokolla käytöstä tai estämällä pääsy laiteen etähallintaportteihin epäluotettavista osoitteista. Ciscon pääsylistoilla voidaan estää SNMP-protokollan käyttö komennolla:

```
access-list 101 deny udp any any eq 161 log
```
[5]

Myös käyttämällä SNMP-protokollan uusinta versiota voidaan salasanojen paljastumista ehkäistä tehokkaammin. Oletussalasanojen takia verkon ylläpitäjän onkin ensimmäisenä muutettava nimet vastaamaan tavallisia salasanoja. Oletustilit tulisi poistaa välittömästi laitetta asennettaessa tai vähintään vaihtaa niiden salasana. [4]

Konfigurointitiedostojen lataaminen TFTP-protokollan avulla voidaan estää kieltämällä kokonaan TFTP:n käyttö tai tekemällä reitittimeen suodatinsääntö, joka estää TFTP-yhteydet. Reitittimien ohjelmistot tulisi päivittää tunnettujen tietoturvaaukkojen tukkimiseksi.

2.1.3 Cisco-reitittimien heikot salasanat

Ciscon verkkolaitteiden käyttämä menetelmä salasanojen salaamiseen on hyvin heikko nykyaikaisiin salausmenetelmiin verrattuna. Salasanat tallennetaan laitteen konfiguraatiotiedostoon käyttämällä heikkoa XOR-koodaukseen perustuvaa menetelmää [4]. XOR-menetelmä koodaa salasanan käyttämällä aina samaa merkkijonoa siemenarvona. Tästä syystä salauksen purkaminen on hyvin helppoa ja internetistä löytyy useita ohjelmia, joilla purku onnistuu.

Heikko salaus koskee salasanoja, jotka on luotu `enable password` -komennolla. `Enable secret` -komennolla luotavat salasanat salataan käyttämällä MD5-sekasumma-algoritmia, joka antaa paremman suojan salasanojen murtamista vastaan. Käytetyn salauksen voi tarkistaa konfiguraatiotiedostosta. Jos tiedostossa esiintyy käyttäjätunnuksen kohdalla numero seitsemän on salaukseen käytetty heikkoa algoritmia. Jos numero on viisi, algoritmina on MD5. Esimerkiksi: [8]

```
enable secret 5 $1$iUjJ$cDZ03KKGh7mHfX2RSbDqP.

username jdoe password 7 07362E590E1B1C041B1E124C0A2F2E2068
                         32752E1A01134D
```

Vastatoimenpiteenä heikoille salasanoille on käyttää `enable secret` -komentoa salasanoja luotaessa.

2.2 Verkkojen salakuuntelu

Vuosikymmenten ajan jaettua verkkoa (Ethernet, Token Ring) on käytetty datan siirtämiseen laitteelta toiselle. Jaettu tekniikka on vanhentunutta, eikä sitä käytetä enää uusia verkkoja rakennettaessa. Vielä nykyään osa verkoista on toteutettu tällä vanhalla tekniikalla. Uudet verkkotekniikat käyttävät kytkentäistä verkkoa, joka parantaa suorituskykyä ja lisää turvallisuutta.

Jaetut verkot käyttävät CSMA/CD-tekniikkaa, joka jakaa verkon kapasiteetin kaikkien siihen liitettyjen laitteiden kesken. Jokainen laite näkee kaiken verkossa liikkuvan datan ja poimii vain itselleen osoitetut paketit datavuosta. Tämä mahdollistaa helposti muille tarkoitetun verkkoliikenteen salakuuntelun jonkin verkon laitteen avulla. Kytkentäisissä verkoissa käytetään reititystä, joka ohjaa paketteja laiteosoitteiden perusteella vain vastaanottajalle. Näin paketteja ei pitäisi nähdä kukaan muu. Vaikka kytkentäisissä verkoissa salakuuntelu on vaikeampaa, ei se kuitenkaan ole mahdotonta. [2]

Cisco Catalyst -kytkimissä voidaan valita asetus, joka peilaa kaiken liikenteen tiettyyn kytkimen porttiin ilman, että vastaanottaja sijaitsisi tämän portin takana. Tätä käytetään verkon valvontaan ja IDS-järjestelmien[3] toteuttamiseen. [9]

Verkossa liikkuvia paketteja siepataan ohjelmilla, joita kutsutaan haistelijaksi (engl. *Sniffer*). Niitä käytetään verkkoliikenteen analysointiin ja vikatilanteiden havaitsemiseen. Haistelijat asettavat verkkosovittimen *promiscuous*-tilaan, jolloin verkkosovitin sieppaa kaiken kuulemansa liikenteen. Haistelijat ovat täysin passiivisia, joten ne vain kuuntelevat verkon liikennettä. Tämän takia niitä on hyvin hankala huomata, jos sellainen on asennettu ylläpitäjän tietämättä. [10]

2.2.1 Salasanojen haistelu

Käyttämällä erilaisia haistelija-ohjelmia, voidaan verkkoliikenteestä haistella selkokieliset tai heikosti salatut salasanat. Koska verkossa liikkuu paljon paketteja, pitää haistelijan tallentaa vain mielenkiintoisimmat paketit. Yleensä paketeista tallennetaan vain 200–300 ensimmäistä tavua, sillä käyttäjätunnukset ja salasanat ovat yleensä heti paketin alussa [10]. Paketteja valikoidaan myös porttinumeroiden perusteella. Erilaiset autentikointiportit ovat haistelijoiden suosiossa, kuten portti 21 (FTP), 23 (Telnet), 513 (rlogin) ja niin edelleen.

Useat sovellukset lähettävät salasanat selkokielisinä tai heikosti salattuina, jolloin ne ovat kaikkien luettavissa, jotka saavat verkossa liikkuvan paketin haltuun-

[3]Intrusion detection systems, Tunkeilijan havaitsemisjärjestelmä.

sa. Tunnetuimpia selkokielisiä salasanoja käyttäviä sovelluksia ovat FTP-, Telnet-, POP- ja IMAP-sähköpostiprotokollat, useat pikaviestiprotokollat (ICQ, IRC, AIM), tiedostonjakoprotokollat (NFS, SMB) sekä monet muut sovellukset [4]. Käyttäjällä saattaa olla sama salasana useassa eri järjestelmässä, jolloin salasanan paljastumisesta saattaa aiheutua laajaa vahinkoa.

Internetistä löytyy useita ohjelmia, jotka kaappaavat paketteja ja niissä olevia selkokielisiä tai heikosti salattuja salasanoja ja näyttävät ne käyttäjälle. Eräs tällainen ohjelma on Dug Songin tekemä *dsniff*-ohjelma[4] sekä *mailsnarf*-ohjelma, joka kerää verkossa liikkuvat sähköpostipaketit ja kokoaa niistä alkuperäisen viestin. Näin kuka tahansa pääsee käsiksi salaamattomiin sähköpostiviesteihin.

Yleensä *dsniff*-ohjelman ja muiden haistelijoiden havaitseminen on vaikeaa, muttei mahdotonta. Ohjelman voi löytää tarkastelemalla laitteiden ARP-tauluja ja niiden muutoksia huijausten varalta [11]. Verkosta voidaan myös etsiä *dsniff*-ohjelman aiheuttamaa epänormaalia liikennettä, kuten ICMP-viestejä DNS-palvelimelta tai TCP-RST tai -ACK-viestien tulvaa [11]. Yleisempi tapa estää salakuuntelu on käyttää tiedonsiirron salausta. Kun liikenne tunneloidaan SSH:n (*Secure Shell*) avulla, sitä voidaan kaapata, mutta sen sisällön murtaminen on lähes mahdotonta. Myös erilaisten kertakäyttösalasanojen tai Kerberos-autentikoinnin avulla voidaan suojautua salasanahaisteluja vastaan. Eri järjestelmillä tulee olla omat salasanansa ja käyttäjien ei tule käyttää samaa salasanaa jokaisessa järjestelmässä.

2.2.2 Man in the middle

"Man in the middle"-hyökkäykseksi kutsutaan skenaariota, jossa kahden liikennöivän laitteen välissä on salakuuntelija, jota kummatkaan osapuolet eivät huomaa. "Man in the middle"-hyökkäyksiä voidaan toteuttaa useilla eri tavoilla myös salatuille yhteyksille.

MITM-hyökkäys toteutetaan muuttamalla kohdelaitteen ARP-taulua ohjaamaan oletusyhdyskäytävälle lähetetyt paketit hyökkääjän koneelle. ARP-ohjauksen voi toteuttaa esimerkiksi *arpredirect*-ohjelmalla, joka on osa dsniff-ohjelmapakettia. *Arpredirect*-ohjelma lähettää ARP-vastauksia kohdekoneelle, väittäen olevansa oletusyhdyskäytävä. Kohdekone päivittää oman ARP-taulunsa vastaamaan uutta tilannetta. Tämän jälkeen kaikki liikenne, joka on tarkoitettu oletusyhdyskäytävälle, tulee hyökkääjän koneelle, jolloin hyökkääjä voi kaapata sitä haistelija-ohjelmalla. Jotta hyökkääjän olemassaoloa ei havaita, pitää hyökkääjän ohjata vastaanottamansa liikenne myös oikealle yhdyskäytävälle. Näin mikään osapuoli ei havaitse yhteyden

[4]http://monkey.org/ dugsong/dsniff/

kiertävän kolmannen osapuolen kautta. Kun kohdekone aloittaa uuden yhteyden, hyökkääjä sieppaa TCP:n SYN-paketin, tuhoaa sen ja lähettää uuden paketin alkuperäiselle vastaanottajalle, johon on väärennetty alkuperäisen lähettäjän osoite. Kuvassa 2.1 esitetään "man in the middle"-hyökkäyksen eteneminen. [4] [2]

Kuva 2.1: "Man in the middle"-hyökkäys. [2]

MITM-hyökkäys voidaan toteuttaa myös RIP-huijauksen avulla. RIP (*Routing information protocol*) on UDP-pohjainen reititysprotokolla, jonka versiot 1 ja 2 ovat helposti huijattavissa. Versio 1 ei todenna lähettäjää millään tavalla ja versiossa 2 todennus tapahtuu selkokielisillä salasanoilla, jotka voidaan haistella verkkoliikenteestä. Lähettämällä tekaistu reititystieto RIP-protokollaa tukevalle reitittimille, hyökkääjä saa reitittimen reitittämään liikenteen itselleen. Edelleenlähettämällä liikenne oikealle laitteelle, hyökkääjä saa luotua MITM-tilanteen. [4]

Salauksen käyttäminen ei estä MITM-hyökkäyksiä. Kun osapuolet sopivat istuntoavaimista salauksen toteuttamiseksi, hyökkääjä sopii eri avaimet kummankin osapuolen kanssa. Kun lähettäjä salaa liikenteen käyttäen itsensä ja hyökkääjän kanssa sopimaansa avainta, hyökkääjä purkaa salauksen, sieppaa paketin ja salaa sen uudelleen käyttäen vastaanottajan kanssa sopimaansa salausavainta. [2]

ARP-huijaukset tapahtuvat aliverkon sisällä, jossa laitteet tunnistetaan niiden laiteosoitteiden avulla. Huijauksia voidaan tehdä myös suuremmissa verkoissa, joissa käytetään DNS-nimiä. DNS-nimet on tarkoitettu helpottamaan IP-osoitteiden muistamista. Laitteiden nimi- ja IP-osoiteparit sijaitsevat DNS-palvelimilla, joilta tietoja kysytään tarvittaessa, esimerkiksi otettaessa selaimella yhteyttä `www.jyu.fi`-osoitteeseen, haetaan DNS-palvelimelta osoitetta vastaava IP-osoite. DNS-palvelimelle murtautuminen ja IP-osoitteita vaihtamalla hyökkääjä voi ohjata käyttäjät eri palvelimelle. Tätä keinoa voidaan käyttää huijaamalla asiakkaat väärennetyn verkkopankin sivuille, jonne he antavat verkkopankkitunnuksensa.

Vastatoimenpiteenä ARP-taulujen muokkaamiseen on määritellä pysyvät ARP-tietueet tärkeiden laitteiden välille. Tällöin tietueita ei voi muokata. Toinen vaihtoehto on käyttää erityistä ohjelmaa, joka valvoo ARP-tauluja ja huomaa muutokset. Tällainen ohjelma on esimerkiksi arpwatch. RIP-huijauksilta voi suojautua käyttämällä RIP-protokollan sijaan OSPF-protokollaa (*Open shortest path first*), joka sisäl-

tää huijauksia rajoittavia mekanismeja. Uloimpien reitittimien tulisi myös suodattaa RIP-paketteja, jotta niitä ei pääsisi sisäverkkoon. [4]

2.2.3 Istunnon kaappaaminen

Edellä kuvatut verkkoliikenteen salakuuntelutavat ovat pääsääntöisesti passiivisia, joten hyökkääjän on odotettava kunnes käyttäjä kirjautuu suojaamattomaan palveluun ja paljastaa salasanansa hyökkääjälle. Aktiivisempi hyökkäys on kaapata meneillään oleva TCP-istunto (engl. *TCP hijacking*), jossa käyttäjä on jo kirjautunut etäpalveluun. Hyökkääjä kaappaa yhteyden ja esittää palvelimelle käyttäjää samalla estäen käyttäjää kommunikoimasta palvelimen kanssa. Tällöin palvelin ei välttämättä huomaa käyttäjän vaihtumista.

Istunnon kaappaamisella voidaan murtautua järjestelmiin, joissa on istuntokohtainen salasana, jota ei voida käyttää uudestaan. Tällöin salasanan haistelusta ei ole hyötyä, sillä salasana on jo vanhentunut. [12]

TCP-kaappaus on mahdollista, koska TCP-protokollan suunnittelussa ei otettu huomioon väärennettyjä paketteja, joita voidaan lisätä helposti tietovirtaan [4]. Kaappaus tapahtuu lähettämällä palvelimelle väärennettyjä paketteja, jotka näyttävät tulevan käyttäjältä. Käyttäjälle lähetetään FIN- tai RST-paketti palvelimen nimissä, jolloin käyttäjä luulee palvelimen katkaisseen yhteyden. Hyökkääjä voi joutua käyttämään palvelunestohyökkäystä käyttäjää vastaan, jotta tämä ei pääse vastaamaan palvelimelta tuleviin viesteihin. Kun yhteys on kaapattu, saa hyökkääjä käyttäjän istunnon haltuunsa ja käyttäjän oikeudet palvelimella. Yhteyden kaappaus vaatii, että hyökkääjä pystyy kuuntelemaan yhteyttä ja lisäämään omia paketteja verkkoon. [13]

Hyökkääjän mahdollisuudet ovat paremmat, jos hän pystyy myös poistamaan paketteja datavirrasta pelkän lisäämisen ohella. Jos paketteja ei poisteta, liikkuu verkossa paketteja, joiden tiedot ovat ristiriidassa keskenään. Joissakin paketeissa on oikean vastaanottajan vastaus ja toisissa hyökkääjän lähettämä vastaus. Tällöin kohde- tai IDS-järjestelmä voi huomata ristiriitaisuuden ja päätellä istunnon kaappauksen olevan käynnissä. Joissakin tapauksissa palvelin voi tunnistaa asiakkaan käyttöjärjestelmän (luku 2.1.1) ja näin havaita asiakkaan vaihtuneen. [12]

UDP-yhteyksien kaappaaminen on helpompaa kuin TCP-yhteyksien, sillä UDP-protokollassa ei ole kuittauksia, järjestysnumeroita tai muita ominaisuuksia, jotka varmistavat pakettien perillemenon. TCP:n varmistusmekanismit voidaan kuitenkin implementoida UDP-protokollan päälle, mutta yleensä ne ovat yksinkertaisempia kuin TCP:n vastaavat ominaisuudet. UDP-yhteyksien kaappaaminen on yleen-

sä kilpajuoksua ajan kanssa siitä, kumpi ennättää vastaamaan palvelimelle ensiksi, hyökkääjä vai asiakas. Hyökkääjä tarvitsee vain ohjelman, joka tarkkailee UDP:n kautta tulevia pyyntöjä ja vastaa niihin mahdollisimman nopeasti. [12]

Internetissä on useita valmiita sovelluksia yhteyksien kaappaamiseen. Sovelluksilla on helppo testata onko käytetty yksityinen verkko turvallinen, vai voidaanko siellä olevia yhteyksiä kuunnella ja kaapata. Eräs tällainen ohjelma on *Hunt*[5]. *Hunt* listaa kaikki käynnissä olevat yhteydet ja antaa mahdollisuuden joko salakuunnella viestejä tai kaapata yhteys. *Hunt* pystyy suorittamaan komentoja etäjärjestelmässä niin, että komento ja sen vaste näkyvät vain hyökkääjän järjestelmässä. Näin voidaan esimerkiksi lukea `/etc/password`-tiedoston sisältö käyttäjän tietämättä. [4]

Hunt (versio 1.5) näyttää käynnissä olevan Telnet-yhteyden (portti 23), jota voidaan salakuunnella tai kaapata:

```
    --- Main Menu --- rcvpkt 517, free/alloc 63/64 ------
l/w/r) list/watch/reset connections
u)     host up tests
a)     arp/simple hijack (avoids ack storm if arp used)
s)     simple hijack
d)     daemons rst/arp/sniff/mac
o)     options
x)     exit
-> l
0) 192.168.1.33 [4312]          --> 192.168.1.1 [23]
```

Kaappauksia vastaan voidaan suojautua käyttämällä IPSec- tai SSH-protokollaa, joilla voidaan salata verkkoliikenne [4]. Tällöin hyökkääjä ei pysty lukemaan verkkoliikennettä, eikä kaapata sitä. Myös jokaisen paketin allekirjoittaminen estää hyökkääjää teeskentelemästä yhteyden aloittanutta käyttäjää [13]. Pelkkä yhteyden alussa tapahtuva käyttäjän tunnistus ei riitä, sillä kaappaus tapahtuu tunnistuksen jälkeen.

2.2.4 Langattomat verkot

Langattomien verkkojen lisääntynyt suosio on avannut hyökkääjille uusia reittejä yksityisiin verkkoihin. Enää hyökkääjän ei tarvitse kytkeytyä fyysiseen verkkokaapeliin, vaan verkkoon voi päästä jopa rakennuksen ulkopuolelta. Ensimmäiset versiot langattomista verkoista sisälsivät paljon heikkouksia ja niitä löydettiin myö-

[5]http://www.l0t3k.net/tools/Hijacking/hunt-1.5.tgz.gz

hemmin lisää. Salaamaton tai huonosti salattu verkkoyhteys on lähes kaikkien verkon kantaman sisällä olevien käytettävissä, joka voi olla jopa 100 metriä avoimessa maastossa[14]. Ensimmäisissä langattomissa verkoissa käytettiin WEP-salausta (*Wired equivalent privacy*). Vaikka nykyään on käytettävissä tehokkaampi salauskeino, silti turhan usea langaton verkko on täysin avoin tai suojattu vanhalla WEP-salauksella.

Täysin suojaamattoman verkon avulla voidaan pahimmassa tapauksessa siepata kaikki yrityksen verkossa liikkuva data, varsinkin jos langaton verkko sijaitsee yrityksen palomuurin sisäpuolella. Tällöin hyökkääjä voi helposti ohittaa koko palomuurin. [4]

WEP-salaus sisältää avoimen (engl. *open system*) sekä jaetun avaimen autentikoinnin (engl. *shared-key authentication*). Avoin autentikoituminen ei tarjoa mitään suojaa, siinä verkkoon liittyjä joko hyväksytään tai yhteys hylätään, jos avointa autentikointia ei ole tarjolla. Jaetun avaimen autentikoituminen tapahtuu 4-tie kättelyllä, joka perustuu ennalta määrättyyn salaiseen avaimeen, jonka kummankin osapuolen on tiedettävä. WEP-salaus käyttää RC4-vuosalausta (engl. *RC4 stream cipher*). [14]

WEP-salauksen on todettu sisältävän useita heikkoja kohtia ja toteutuksia, joiden takia WEP-salaus on purettavissa suhteellisen helposti muutamassa tunnissa, riippuen liikenteen määrästä. Internetissä on useita ohjelmia, jotka sieppaavat liikennettä langattomista verkoista ja laskevat käytetyn salausavaimen. Tämän jälkeen langattomaan verkkoon voidaan liittyä käyttämällä tunnettua salausavainta.

Ensinnäkin WEP-salaus ei sisällä kunnollista eheyssuojaa pakettien muokkaukselle. WEP:n käyttämä CRC-32 ei sovellu tiedon eheyden tarkistamiseen ja tästä syystä paketteja voidaan väärentää, jolloin pakettia purettaessa muutosta ei huomata. [14]

Toinen heikkous liittyy RC4-vuosalauksen väärään käyttöön. Koska RC4 on vuosalausalgoritmi, tulisi salausavainta käyttää pelkästään kerran vuokohtaisen salausavaimen laskemiseen. Jotta saavutettaisiin synkronoituva tiedonsiirto (pakettien järjestys säilyttävä, vaikka ne saapuisivat eri aikaan), WEP käyttää pakettikohtaista alustusvektoria (engl. *initialization vector, IV*), jonka pituus on 24-bittiä. Tällöin jokaiselle paketille lasketaan oma vuokohtainen salausavain alkuperäisen salausavaimen avulla. Tämä mahdollistaa alkuperäisen salausavaimen selvittämisen kryptoanalyysien avulla. [14]

Koska 24-bittisyys mahdollistaa vain noin 17 miljoonaa eri vaihtoehtoa on todennäköistä (yli 50% TN), että noin joka 4823. paketti sisältää saman vuokohtaisen

salausavaimen (ns. syntymäpäiväongelma[6]). Kun tunnetaan tarpeeksi samalla salausavaimella salattuja paketteja, voidaan alkuperäinen salausavain laskea ja näin purkaa kaikki tällä avaimella salattu liikenne.

Vastatoimenpiteenä WEP-salauksen purkamiselle on käyttää uudempaa WPA-salausta, sekä SSL- tai SSH-protokollia, VPN-tunnelointia tai IPSec:a [4]. WEP2-salaus käyttää 128-bittiä pitkää IV-vektoria, jolloin saman vuokohtaisen salausavaimen todennäköisyys laskee huomattavasti. Autentikoituminen tulisi tehdä autentikointipalvelinten, kuten Kerberoksen, avulla. SSID-tunnisteen piilottaminen ja MAC-osoitesuodatus auttavat epähuomiossa tai tietämättömyydessä tehtyihin verkon kaappauksiin, mutta hyökkääjää ne eivät pidättele.

2.3 Palomuurien ohittaminen

Useiden vuosien ajan palvelimen liittämistä Internettiin ilman palomuuria on pidetty lähes itsetuhoajatuksena. Palomuureista on tullut yksityisten verkkojen tukipylväitä. Oikein asennettuna, konfiguroituina ja ylläpidettyinä nämä suojamuurit ovat tehokas tapa estää asiattomilta pääsy yksityiseen verkkoon ja ne ovat lähes läpipääsemättömiä. Jos palomuurien ylläpito on jätetty muutaman henkilön sivutoimiseksi tehtäväksi voi hyökkääjä kohdata vanhan ja päivittämättömän palomuurin, joka ei pysty pysäyttämään motivoitunutta hyökkääjää. Väärin konfiguroitu palomuuri voi päästää hyökkääjän liikenteen läpi, vaikka tämä ei ole ollut ylläpitäjän tarkoituksena.

2.3.1 Palomuurin tunnistaminen

Palomuurin löytämiseen tunnistamiseen pätevät pääsääntöisesti samat keinot kuin verkon aktiivilaitteiden tunnistamisessa (luku 2.1.1) eli *traceroute*, porttien tutkinta ja bannereiden sieppaus.

Useimmilla palomuureilla on tiettyjä avoimia oletusportteja[7], joista laitteet voidaan tunnistaa helposti [4]. Käyttämällä *nmap*-ohjelmaa, voidaan etsiä verkosta laitteita, joissa nämä portit ovat auki. Näin voidaan kohtuudella tunnistaa palomuurin tyyppi. *Nmap*-ohjelmalla tutkinta tapahtuu komennolla:

```
nmap -v -n -P0 -p [oletusportit] [ip-osoitteet]
```
[3]

[6]Vanha matemaattinen ongelma: "Kuinka monta henkilöä tarvitaan, jotta olisi todennäköistä, että ainakin kahdella olisi sama syntymäpäivä?" Vastaus: Tuloperiaatteella voidaan laskea, että tarvitaan 23 ihmistä, jotta kahdella olisi yli 50% todennäköisyydellä sama syntymäpäivä.

[7]Esimerkiksi CheckPoint Firewall-1 kuuntelee TCP-portteja 256, 257, 258.

Käyttämällä `P0`-valitsinta *nmap* ei testaa kohdelaitteen tilaa ICMP-pingillä, sillä useat palomuurit jättävät vastaamasta ping-viesteihin.

Palomuurin olemassaolo voidaan myös päätellä tutkimalla jonkin laitteen portteja ja tarkkailemalla vastaanotettuja paketteja, jotka kertovat portin tilasta. Jos ennen kohdetta on palomuuri, se ei päästä porttitutkintapaketteja lävitseen vaan tuhoaa paketit. *Nmap* osaa päätellä onko portti avoin, kiinni tai suodatettu. Avoin portti tarkoittaa porttia, jossa vastaa jokin palvelu. Kiinni oleva portti tarkoittaa porttia, joka on avoin, mutta portissa ei ole kuuntelevaa palvelua. Suodatettu portti tarkoittaa porttia, jonka tilaa ei voitu päätellä, koska palomuuri on estänyt tutkimisen. Suodatettu portti voi tarkoittaa jotakin seuraavista: [4]

- Yhtään SYN/ACK-viestiä ei saatu
- Yhtään RST/ACK-viestiä ei saatu
- Saatiin ICMP-tyyppin 3 -viesti koodilla 13, joka tarkoittaa, että yhteys on estetty hallinnollisin keinoin (engl. *Communication Administratively Prohibited*)

Nmap on löytänyt kolme avointa porttia sekä yhden suljetun. Muut portit ovat suodatettuja, joten voimme päätellä, että ennen kohdetta on palomuuri, joka estää porttitutkimisen:

```
Interesting ports on 192.168.1.34:
(The 1659 ports scanned but not shown below are in state:
filtered)
PORT    STATE  SERVICE
21/tcp  open   ftp
23/tcp  open   telnet
80/tcp  open   http
113/tcp closed auth
```

Nmap voi ilmoittaa portin tilaksi suodattamaton (engl. *unfiltered*). Tällöin *nmap* ei pysty päättelemään pääsikö tutkiminen palomuurista läpi ja portti oli suljettu vai vastasiko palomuuri portin olevan suljettu kohteen nimissä.

```
PORT   STATE      SERVICE
22/tcp UNfiltered ssh
```

Varomaton hyökkääjä voidaan havaita suuresta määrästä porttiskannauksia, jotka tulevat samasta lähteestä. Tällöin on syytä epäillä, että sisäverkon laitteita tutkitaan hyökkäystarkoituksessa. Myös IDS-järjestelmät havaitsevat nämä kovaääniset

tunnustelut. Varovaisempi hyökkääjä hajauttaa porttitutkimuksensa käyttäen useaa eri lähdettä, jotka suorittavat satunnaisia porttiskannauksia. Lopussa hyökkääjä yhdistää nämä tiedot. Tutkimisen hajauttaminen ja satunnaistaminen hämää yleensä perusasetuksilla olevia IDS-järjestelmiä. [4]

Tracerouten ja bannereiden avulla tapahtuvaan tunnistamisen estämiseen soveltuvat samat keinot kuin luvussa 2.1.1. Reitin jäljitykset *tracerouten* avulla voi havaita helposti ICMP- ja UDP-paketeista, joiden TTL-kentän arvo on 1 [4].

Porttien tutkiminen voidaan havaita asentamalla yksityiseen verkkoon IDS-järjestelmä, jonka tehtävänä on havaita verkkoa vastaan tehdyt hyökkäykset ja estää niiden uusiutuminen. Verkon uloimmissa reitittimissä voidaan kieltää yhteydet palomuurien hallintaportteihin, jolloin niiden perusteella ei voida tunnistaa palomuuria. Tällöin kuitenkin palomuurin hallinta Internetin kautta ei ole mahdollista. [4]

2.3.2 Palomuurin läpi tutkiminen

Kun todennäköinen palomuuri on löytynyt voidaan tutkia onko palomuurissa polkuja, joita pitkin päästään palomuurin lävitse. Tähän soveltuu hyvin Salvatore Sanfilippon kirjoittama *hping*-ohjelma[8], joka lähettää TCP-paketteja kohdeporttiin ja raportoi saamansa vastauksen. [4]

Vastausten perusteella voimme päätellä pääsikö paketti palomuurin läpi, oliko kohdeportti auki vai kiinni:

```
lucifer:/ # hping 192.168.1.34 -c 2 -S -p 80 -n
HPING 192.168.1.34 (eth0 192.168.1.34): S set, 40 headers
+ 0 data bytes
len=46 ip=192.168.1.34 ttl=64 DF id=0 sport=80 flags=SA
seq=0 win=5840 rtt=0.8 ms
len=46 ip=192.168.1.34 ttl=64 DF id=0 sport=80 flags=SA
seq=1 win=5840 rtt=0.8 ms
```

Lipuista SA (SYN/ACK) voidaan päätellä, että paketti on päässyt palomuurin läpi ja kohdelaitteen portissa 80 on kuunteleva palvelu. Jos *hping* palauttaa ICMP-tyypin 3 -paketin koodilla 13, voimme päätellä, että palomuuri on estänyt paketin (kts. lista luvussa 2.3.1). Tällöin paluupaketissa on yleensä palomuurin IP-osoite, jolloin saamme selville palomuurin tarkan sijainnin. Jos paluupaketissa on liput RA (RST/ACK), voidaan päätellä joko portin olevan kiinni tai palomuurin suodattaneen paketin. Palomuuri yleensä vastaa kohteen nimissä, jolloin IP-osoite on koh-

[8] http://www.hping.org/

delaitteen osoite. Jos *hping* ei vastaanota paluupakettia on palomuuri suodattanut paketin. [4]

```
len=46 ip=192.168.1.35 ttl=128 id=5085 sport=22 flags=RA
seq=0 win=0 rtt=2.3 ms
```

Palomuurin avoimia portteja skannataan myös *Firewalk*-ohjelmalla[9]. *Firewalk* lähettää paketteja, joiden on tarkoitus olla voimassa (IP-paketin TTL-arvo) yhden hypyn verran palomuurin jälkeen. Jos palomuuri sallii paketin, saadaan vastaukseksi "ICMP TTL expired in transit" -viesti, koska paketti tuhoutuu seuraavassa verkon laitteessa heti palomuurin jälkeen. Jos mitään viestiä ei saada tai saadaan ICMP-tyypin 13 -viesti, voidaan päätellä että palomuuri on suodattanut paketin. *Firewalkin* ongelmana on sen epäluotettavuus. Useat palomuurit havaitsevat, että paketin elinaika loppuu ennen sen pääsyä kohdelaitteelle ja ne lähettävät "ICMP TTL expired in transit" -viestin ja tuhoavat paketin. Tällöin Firewalk näyttää, että kaikki portit ovat avoimia. [4]

Hpingiä on hyvin vaikea estää. Reitittimen pääsylistaan voidaan laittaa esto ICMP-tyypin 13 -viesteille. Tällöin palomuurin tarkkaa sijaintia on vaikeampi päätellä. Firewalk-tunnustelun voi torjua suodattamalla ICMP TTL EXPIRED -paketit uloimmassa reitittimessä, mutta tällöin se voi haitata muuta verkon toimintaa. Toinen tapa on konfiguroida palomuuri suodattamaan ainakin paketit, joiden elinaika loppuu heti palomuurin jälkeen tai hankkimalla palomuuri, joka osaa tarkistaa paketin eliniän riittävän kohteeseen asti.[4]

2.3.3 Tilattoman palomuurin ohittaminen

Jos palomuuri on tilaton, se ei pidä kirjaa sisäverkosta aloitetuista yhteyksistä, voidaan palomuuri ohittaa yksinkertaisesti väärentämällä lähdeportin numero[4]. Esimerkiksi FTP-protokolla käyttää porttia 21 kontrollikanavana. Kun käyttäjä pyytää tiedostonsiirtoa palvelin avaa uuden yhteyden tiedonsiirtoa varten, jossa lähdeportti on palvelimella portti 20 ja kohdeporttina käyttäjän suurinumeroinen portti (yli 1024). Koska palvelin aloittaa tiedonsiirron, tulee palomuurin sallia liikenne sisäänpäin. Tilalliset palomuurit tarkkailevat FTP:n kontrollikanavaa ja havaitsevat tiedonsiirtopyyntöä edeltävän `PORT`-viestin, jossa FTP-asiakas määrittelee vapaan portin, jonne palvelimen tulee ottaa yhteyttä [15]. `PORT`-komento on muotoa

`PORT a,b,c,d,p1,p2<CRLF>` [15]

[9] http://www.packetfactory.net/projects/firewalk/

jossa `a,b,c,d` vastaa asiakkaan IP-osoitetta desimaalein erotettuna. p_1 ja p_2 ilmoittavat halutun portin kaavalla $p_1 * 256 + p_2$. Tilallinen palomuuri osaa täten varautua yhteydenottoon määritettyyn porttiin ja sallii sen. Tilattomalla palomuurilla ei ole aikaisempaa tietoa aloitetusta FTP-yhteydestä, joten se joutuu sallimaan FTP-liikenteen, jonka lähdeportti on 20 ja kohdeporttina jokin suurinumeroinen portti [4].

Nmap-ohjelmalla voidaan testata onko palomuuri tilallinen väärentämällä lähdeportin numero komennolla

```
nmap -sS -P0 -g [portti] -p [kohdeportti] [kohde_ip]
```

[3]

jossa `g`-valitsimella muutetaan paketin lähdeporttia.

Tilattomien palomuurien ohittamiseen voi suojautua vain kieltämällä yhteydet, jotka tarvitsevat useampaa porttia. Kieltäminen hankaloittaa palveluiden käyttöä huomattavasti. Toinen tapa on hankkia tilallinen palomuuri. [4]

2.3.4 FTP-huijaus

Kuten edellä mainittiin (luku 2.3.2), tilalliset palomuurit osaavat tarkkailla FTP-liikennettä ja sallia tilapäisesti yhteydet ennalta neuvoteltuun suurinumeroiseen porttiin. Tätä ominaisuutta voidaan myös käyttää laittomien yhteyksien luomiseksi palomuurin läpi joidenkin palomuurien kohdalla.

Linuxin ytimen versioiden 2.4.x mukana tuleva NetFilter/IPTables-palomuuri ei tarkasta FTP:n `PORT`-komennon porttinumeron oikeellisuutta, vaan lisää annetun lähdeosoitteen ja porttinumeron suoraan reititystauluunsa odottamaan palvelimen yhteydenottoa. Hyökkääjä voi suorittaa FTP-komennon `PORT a,b,c,d,0,22` (jossa `a,b,c,d` on jonkin sisäverkossa olevan palvelimen IP-osoite pilkkuerotetussa muodossa), jolloin palomuuri odottaa yhteyttä sisäverkon palvelimen porttiin 22 (SSH). Vaikka FTP-palvelin ei hyväksy `PORT`-komentoa, palomuuri on jo lisännyt yhteyden reititystauluunsa ja hyökkääjällä on noin 10 sekuntia aikaa ottaa yhteyttä porttiin 22 ennen kuin palomuuri poistaa tilapäisen säännön. Vakavammaksi haavoittuvuuden tekee se, että NetFilter ei tarkasta onko käyttäjä kirjautunut FTP-palveluun sisään. Hyökkääjä voi ottaa yhteyden sisäverkossa toimivaan FTP-palvelimeen ja antaa seuraavat komennot: [16]

```
220 tsunami FTP server ready.
[garbage]
530 Please login with USER and PASS.
```

```
PORT 200,249,193,1,0,22
530 Please login with USER and PASS.
QUIT
221 Goodbye.
```

Tämän jälkeen hyökkääjällä on noin 10 sekuntia aikaa ottaa yhteys IP-osoitteen 200.249.193.1 porttiin 22.

Vastatoimena on tiukentaa palomuurin sääntöjä niin, että palomuuri hyväksyy johonkin toiseen yhteyteen liittyviä ulkopuolisia yhteyksiä vain tietyille sisäverkon koneille, jotka tarvitsevat yhteyksiä. Näin PORT-haavoittuvuuden avulla ei voi ottaa yhteyttä mihin tahansa sisäverkon koneeseen. Toinen tapa on päivittää NetFilter/IPTables uusimpaan versioon tai asentaa korjaustiedosto[10]. Linuxin ytimestä 2.6.11 lähtien FTP-haavoittuvuus on korjattu. [16] [15]

2.3.5 IRC-haavoittuvuus

IRC-protokolla (*Internet Relay Chat*) on Internetissä välitettävä keskusteluprotokolla, jonka avulla useat ihmiset voivat keskustella keskenään. IRC-asiakas ottaa aluksi yhteyden IRC-palvelimeen (yleensä TCP-portti 6667), kun käyttäjä haluaa keskustella toisen käyttäjän kanssa luodaan sitä varten erillinen yhteys suoraan käyttäjien välille. Tämä muistuttaa FTP:n tapaa toimia (luku 2.3.4, jossa on kontrollikanava ja erilliset tiedonsiirtoyhteydet. Palomuurit tarkkailevat yhteyttä IRC-palvelimelle ja etsivät yksityisen keskustelun aloitus -viestejä (DCC CHAT [vrt. PORT-viesti]).

Oletetaan, että palomuuri suojaa yksityistä verkkoa, jonka takana on FTP-palvelin. Palomuuri sallii yhteydet FTP-palvelimelle ulkoverkosta sekä yhteydet IRC-palvelimelle sisäverkosta käsin. Myös edellisiin palveluihin liittyvät tilapäiset yhteydet sallitaan, jos niitä on pyydetty sisäverkosta käsin. Muut yhteydet ovat kielletty. Seuraava haavoittuvuus koskee NetFilter-palomuuria, joka toimitetaan Linuxin ytimen versioiden 2.4 mukana [15].

Tällöin hyökkääjä voi antaa FTP-palvelimelle

`PORT 192,168,100,100,26,11\r\n` [15]

komennon, joka kehottaa palvelinta ottamaan yhteyttä hyökkääjän koneen porttiin 6667. Tämän jälkeen hyökkääjä antaa käskyn

`RETR [tiedostonnimi]` [15]

[10] http://www.netfilter.org/

jolloin palvelin yrittää ottaa yhteyttä hyökkääjän porttiin 6667. Palomuuri hyväksyy tämän yhteyden normaalina FTP:n toimintana. Kun yhteys on luotu, NetFilter erehtyy luulemaan yhteyttä IRC-yhteydeksi porttinumeron perusteella, luullen FTP-palvelinta IRC-asiakkaaksi ja hyökkääjän konetta IRC-palvelimeksi. Se alkaa tarkkailla paketeista DCC CHAT-komentoa, joka ilmaisee uuden keskusteluyhteyden avaamista. Tällöin hyökkääjä voi käyttää aiemmin palvelimelle siirtämäänsä tiedostoa, johon on sisällytetty DCC CHAT -viesti. Tämän jälkeen palomuuri antaa avata yhteyden hyökkääjän koneelta DCC CHAT -komennossa määritettyyn porttiin (esimerkiksi SSH). [15]

Vastatoimenpiteinä ovat samat keinot kuin luvussa 2.3.4. Haavoittuvuus on korjattu NetFilter/IPTables-palomuurin uudemmissa versiossa Linuxin ytimen versiosta 2.6.11 lähtien.

2.3.6 Huonosti konfiguroidut pääsylistat

Palomuurien suojauskyky riippuu pitkälti pääsyntarkistuslistojen (*Access Control List, ACL*) oikeasta konfiguroinnista. Palomuurin ylläpitäjän tulisi aina tarkistaa, etteivät uudet tietueet pääsylistassa ole liian sallivia. Jos halutaan esimerkiksi sallia palveluntarjoajan suorittaa vyöhykesiirtoja, voidaan pääsyntarkistuslistaan konfiguroida sääntö “salli kaikki liikenne TCP-lähdeportista 53” [4]. Tällöin hyökkääjät voivat helposti päästä palomuurin ohi väärentämällä paketin lähdeportin. Parempi sääntö olisi “salli liikenne palveluntarjoajan DNS-palvelimesta (IP-osoite), lähdeportista 53, kohdeporttiin 53” [4], jolloin voidaan rajata pois laittomat lähdeosoitteet sekä laittomat kohdeportit.

Palomuurin ylläpitäjän tulisi aina varmistaa kenellä on oikeus muodostaa yhteys tiettyyn palveluun ja estää muiden pääsy palveluun.

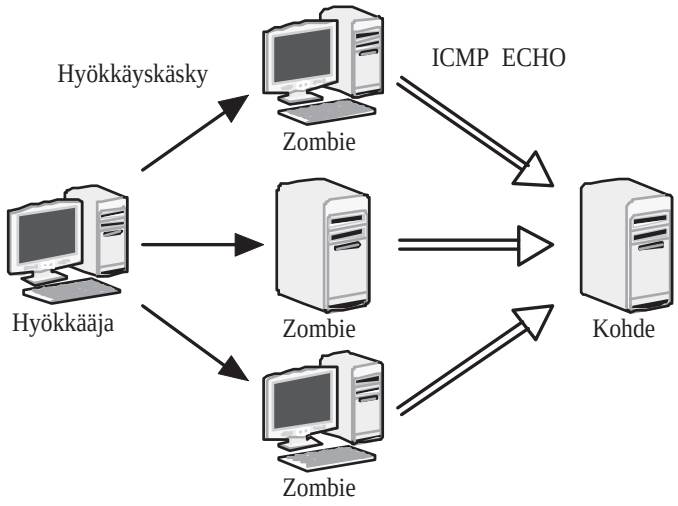
2.4 Palvelunestohyökkäykset

Palvelunestohyökkäysten (engl. *Denial of Service, DoS*) tarkoituksena on lamauttaa kohteena oleva verkon laite niin, ettei se pysty enää käsittelemään uusia yhteyspyyntöjä [4]. Yleensä tämä saavutetaan ruuhkauttamalla esimerkiksi www-palvelin, jolloin se ei enää pysty palvelemaan normaaleja käyttäjiä. Palvelunestohyökkäyksiä voidaan käyttää monista eri syistä. Esimerkiksi murtautumista yrittänyt hyökkääjä voi turhautua puolustuksen toimivuuteen ja viimeisenä tekonaan hän lamauttaa kohteen [4]. Nykyään palvelunestoa käytetään usein kiristyskeinona suuria verkossa toimivia yrityksiä vastaan ja joidenkin yritysten väitetään maksavan “suojelura-

haa", jotta heidän palvelunsa saisivat olla rauhassa.

Hajautettu palvelunestohyökkäys (engl. *Distributed Denial of Service, DDoS*) tarkoittaa hyökkäystä, jossa hyökkääjä on murtautunut useaan verkossa olevaan järjestelmään ja asentanut niihin ohjelman, joka hyökkääjän käskystä aloittaa palvelunestohyökkäyksen määritettyä kohdetta vastaan. Ohjelma on yleensä takaporttiohjelma, joka on asentunut koneeseen madon tai troijalaisen hevosen mukana. Hyökkääjän hallussa olevia koneita sanotaan Zombeiksi, ja niitä voidaan käyttää moneen kertaan eri hyökkäyksissä. Hajautetussa hyökkäyksessä hyökkääjä antaa Zombie-koneille hyökkäyskäskyn ja nämä suorittavat itse hyökkäyksen, joten itse hyökkääjää on hyvin hankala jäljittää. Ensimmäinen julkisesti saatava ollut DDoS-hyökkäystyökalu oli Mixter-nimisen hakkerin kirjoittama TFN (*Tribe Flood Network*), joka sisälsi sekä Zombie-koneeseen asennettavan palvelinohjelmiston että hyökkääjän käyttämän asiakasohjelman. TFN-ohjelma sisältää ICMP, Smurf, UDP ja SYN-hukutushyökkäykset. [13] [4]

Kuvassa 2.2 on esitetty hajautettu palvelunestohyökkäys.



Kuva 2.2: Hajautettu palvelunestohyökkäys. [13]

Nykyään palveluneston aloittaminen on hyvin helppoa valmiiden ohjelmien johdosta, jolloin hyökkäyksen toteuttamiseen ei tarvita erityistä tietämystä tai taitoa. Tästä syystä monien hyökkäysten takana ovat ns. skriptipennut (engl. *script kiddies*), jotka käyttävät valmiita ohjelmia hakkerin maineen saavuttamiseksi.

Palvelunestoja voidaan tehdä myös osana murtautumista, esimerkiksi hyökkäyksen kannalta on tarpeellista saada jokin verkon laite käynnistettyä uudelleen. Tällöin voidaan käynnistää laitetta vastaan palvelunesto, jolloin on mahdollista, että laite kaatuu ja se joudutaan käynnistämään uudelleen. [4]

Palvelunestohyökkäystyyppejä ovat kaistan kuluttaminen, resurssien kuluttaminen, ohjelmointipuutteet sekä reititys- ja DNS-hyökkäykset. [4]

Kaistanleveyden kuluttamisessa pyritään kuluttamaan kaikki kohteen käytettävissä oleva verkkokapasiteetti. Tällöin yleensä suurempi kaistanleveys voittaa. Hyökkääjällä voi esimerkiksi olla 44Mb/s-verkkolinkki (Amerikkalainen T3-linkki) ja kohteella 10Mb/s-verkkolinkki. Tällöin hyökkääjällä on puolellaan kapasiteettiylivoima ja kohteen verkko voidaan tukkia yksinkertaisesti luomalla niin paljon liikennettä kohteen verkkoon, ettei verkon kapasiteetti riitä tiedon siirtämiseen. Hyökkääjät voivat vahvistaa omaa hyökkäystään käyttämällä useita palvelimia, jotka vahvistavat hyökkäystä. Tällöin hyökkääjän verkkokapasiteetilla ei juurikaan ole merkitystä.

Resurssien kuluttamisella tarkoitetaan verkon laitteen resurssien, kuten suorittimen käytön, muistin tai tiedostojärjestelmän, kuluttamista loppuun. Kun resurssit kuluvat loppuun laite yleensä kaatuu, tiedostojärjestelmä täytyy kokonaan tai prosessit hyytyvät. Tällöin laite ei pysty palvelemaan uusia eikä vanhoja yhteyksiä.

Ohjelmointipuutteilla tarkoitetaan sovelluksen tai käyttöjärjestelmän kykenemättömyyttä käsitellä poikkeuksellisia tilanteita kuten väärin muodostettuja paketteja. Puutteet ovat yleensä sovelluksen verkkopinon toteutuksessa, jolloin virhe voi kaataa koko käyttöjärjestelmän tai ohjelman. Puutteet voivat ilmetä esimerkiksi puskureiden ylivuotoina. Ylivuoto tapahtuu kun johonkin puskuriin tallennetaan enemmän dataa kuin on suunniteltu. Jos tallennettavan datan pituutta ei tarkisteta, ylimääräinen data kirjoitetaan muistipaikan perään jonkin toisen ohjelman muistialueelle. Riippuen muistialueesta, päällekirjoitus saattaa kaataa koko sovelluksen. Huomattavasti vaarallisempi tapaus on silloin, kun ylivuodon avulla voidaan kirjoittaa johonkin muistipaikkaan haittakoodia ja saada sovellus suorittamaan sen. Tällöin hyökkääjä voi kaapata jopa koko järjestelmän.

Reititys- ja DNS-hyökkäyksissä hyökkääjä väärentää verkon reititystauluja niin, että verkon laitteet reitittävät kohteeseen pyrkivät paketit muualle tai verkkoon, jota ei ole olemassa.

2.4.1 Smurf

Smurf-hyökkäyksessä hakkeri lähettää johonkin verkkoon broadcast-viestinä ICMP ECHO -viestin, johon on väärennetty lähdeosoitteeksi hyökkäyksen kohteena olevan laitteen IP-osoite. Kaikki verkon järjestelmät vastaavat viestiin ICMP REPLY -viestillä (ellei toisin ole konfiguroitu) ja hyökkäyksen kohde joutuu pakettitulvan kohteeksi, joka voi tukkia sen verkkolinkin kapasiteetin. Smurf-hyökkäys on tyypil-

linen esimerkki vahvistetusta hyökkäyksestä. [4]

Fraggle-hyökkäys on Smurf-hyökkäyksen muunnelma, jossa ICMP ECHO -viestien sijaan lähetetään UDP-paketteja kohdelaitteen porttiin 7 (Echo). Jos laitteessa on käytössä Echo-toiminto, se vastaa viestiin.

Hyökkäykselle ei ole juurikaan vastatoimenpiteitä, mutta verkon laitteet voidaan konfiguroida niin, ettei verkkoa voida käyttää Smurf-hyökkäysten vahvistamiseen. Cisco-reitittimistä voidaan poistaa suunnattu broadcast -toiminto komennolla

`no ip directed-broadcast`. [4]

Lisäksi verkon laitteista tulisi poistaa Echo-toiminto. ICMP-liikenteen rajoittamisesta kannattaa keskustella Internet-palveluntarjoajan kanssa, jolloin hyökkäykset eivät kuluta kaikkea kaistaa. Liikenteen rajoittaminen omissa reitittimissä ei yleensä auta, sillä reitittimelle tuleva yhteys tukkeutuu. Palveluntarjoajaan tulee olla myös yhteydessä mahdollisen hyökkäyksen sattuessa, sillä on mahdollista jäljittää hyökkäyksen lähdettä takaisinpäin, mutta tällöin tarvitaan pääsy reitillä oleville reitittimille. Myös vahvistavan verkon löytäminen ja yhteistoiminta sen ylläpitäjien kanssa on tärkeää hyökkäyksen lopettamiseksi ja hyökkääjän jäljittämiseksi. [4]

2.4.2 SYN-hukuttaminen

SYN-hukuttamisessa (engl. *SYN flood attack*) hyökkääjä lähettää kohteeseen TCP SYN-paketin, jonka lähdeosoite on väärennetty. SYN-paketti tarkoittaa TCP-yhteyden avauspyyntöä, jolloin kohdekone varaa uutta yhteyttä varten resursseja, lähettää SYN/ACK-viestin ja jää odottamaan vastausta. Koska lähdeosoite oli väärennetty, vastausta ei saada koskaan ja varaus lopetetaan tietyn ajan kuluttua (vaihtelee järjestelmästä riippuen 75 sekunnista jopa 23 minuuttiin). SYN-hukuttamisessa hyökkääjä lähettää jatkuvasti SYN-paketteja, jolloin kohdelaitteen resurssit kuluvat loppuun. Vaikka jokin järjestelmä pystyisi käsittelemään satoja yhteyksiä porttia kohti, se saattaa kestää vain kymmenkunta mahdollista yhteyttä porttia kohti, koska kaikki resurssit määritellään yhteyden muodostamiseen. [13] [4]

Netstat-ohjelmalla voidaan havaita helposti SYN-hyökkäys. Jos usea yhteys on `SYN_RECV`-tilassa, se voi merkitä käynnissä olevaa SYN-hyökkäystä.

SYN-hyökkäykseltä voi suojautua kasvattamalla yhteysjonon kokoa ja vähentämällä yhteysajastimen aikaa. Nämä eivät kuitenkaan ole optimaalisia ratkaisuja. Useissa käyttöjärjestelmissä on erilaisia toteutuksia SYN-hyökkäyksen torjumiseksi. Käyttöjärjestelmät pyrkivät hylkäämään SYN-hyökkäykseen kuuluvat yhteydet

sekä estämään yhteysjonon täyttymisen. Myös IDS-järjestelmän asentaminen auttaa havaitsemaan ja torjumaan SYN-hyökkäyksen. IDS-järjestelmät vastaavat aktiivisesti hyökkäykseen esimerkiksi lähettämällä hyökkäyksen alaiselle laitteelle RST-paketteja, jotka sulkevat varattuja yhteyksiä. [4]

2.4.3 DNS-hyökkäykset

DNS-hyökkäyksissä hyökkääjä pyrkii väärentämään nimipalvelimilla olevia domainnimiä vastaavia IP-osoitteita. Jos hyökkääjä väärentää DNS-palvelimella olevan domainnimen (esimerkiksi `jyu.fi`) IP-osoitteen muotoon 0.0.0.10, jota ei ole olemassa, kaikki DNS-palvelinta käyttävät eivät pääse `jyu.fi`-osoitteeseen, elleivät tiedä sen oikeaa IP-osoitetta. DNS-hyökkäyksen avulla hyökkääjä voi seurata tietylle toimialueelle menevää liikennettä tai tehdä "Man in the Middle" hyökkäyksen esimerkiksi pankin verkkopalvelun ja asiakkaan välille. [13]

Yleisin DNS-palvelinohjelmisto on BIND (*Berkeley Internet Name Domain*), jonka useista versioista on havaittu turvallisuusaukkoja, joita hyödyntämällä hyökkääjä on voinut muokata DNS-tauluja. [13]

Jos DNS-palvelin ei tiedä jonkin domainnimen IP-osoitetta, se kysyy sitä muilta verkon DNS-palvelimilta. Jos hyökkääjä havaitsee tällaisen kyselyn, voi hän vastata kyselyyn ja näin DNS-palvelin ja käyttäjä saavat hyökkääjän ilmoittaman IP-osoitteen. [13]

DNS-hyökkäyksen vastatoimenpiteinä ovat BIND-ohjelmiston päivittäminen viimeisimpään versioon, joissa tunnetut tietoturva-aukot on korjattu.

3 IDS-järjestelmät

Nykyään verkon suojaaminen palomuurilla on lähes itsestäänselvyys. Ilman palomuuria verkon laitteita uhkaavat erilaiset haittaohjelmat ja madot, jotka etsivät taukoamatta uusia uhreja verkossa. Palomuurit pystyvät estämään useat tällaiset hyökkäykset, mutta kuten aiemmin kuvattiin, palomuuritkin voidaan ohittaa. Tästä syystä uusi trendi tietoturvan alalla on IDS-järjestelmien (*Intrusion Detection Systems*) asentaminen verkkoon, jolloin mahdolliset palomuurin läpäisyt voidaan vielä havaita. IDS-järjestelmien suosio alkoi tämän vuosituhannen alusta, kun uudenlaiset matoepidemiat koettelivat verkkoja.

IDS-järjestelmän tehtävänä on joko suojata verkkoa edellisessä luvussa esitetyiltä uhkilta tai ainakin niistä ilmoittaminen, jos sellainen havaitaan. Uhkien havaitseminen ja torjunta riippuu IDS-laitteen sijoittamisesta verkkoon ja sen ominaisuuksista.

Tässä luvussa kuvataan erilaisia IDS-järjestelmätyyppejä, niiden ominaisuuksia ja toimintaa sekä yleisempiä IDS-järjestelmien analysointimenetelmiä. Luvussa kerrotaan myös hälytysten käsittelemisestä.

3.1 Mikä on IDS?

IDS-järjestelmien juuret juontavat Yhdysvaltojen puolustusministeriön projekteihin, joita kehitettiin 80- ja 90-lukujen vaihteessa. Projekteissa luotiin malleja tunkeutumisen paljastamiseksi, joita olivat esimerkiksi IDES (*Intrusion Detection Expert System*) ja DIDS (*Distributed Intrusion Detection System*). Nykyaikaiset tunkeutumisenhavaitsemisjärjestelmät suunnitellaan näiden mallien pohjalta. [10]

IDS-järjestelmät kehitettiin helpottamaan ylläpitäjien työtä tunkeutujien ja vahingontekijöiden havaitsemiseksi. IDS-järjestelmät valvovat verkkoa taukoamatta ja raportoivat havaitsemistaan tunkeutumisista tai niiden yrityksistä ylläpitäjille. IDS-järjestelmät voivat myös torjua aktiivisesti tunkeutumisia estämällä liikenteen väärinkäyttäjän IP-osoitteesta valvottuun verkkoon.

Ensimmäiset tunkeutumisen havaitsemiset tapahtuivat järjestelmien ylläpitäjien toimesta, heidän tarkkaillessaan käyttäjien toimia konsoleiden välityksellä. He saattoivat huomata, että lomalla olevan työntekijän tunnuksilla kirjauduttiin järjestelmään. 70- ja 80-lukujen vaihteessa järjestelmien ylläpitäjät käyttivät tarkastuslokeja

etsiessään epätavallisia tapahtumia. Lokit tulostettiin reikäpaperille ja paperipinot saattoivat olla viikon lopulla metrin korkuisia. Lokien tarkastaminen oli hyvin aikaa vievää, joten niihin turvauduttiin vasta sen jälkeen kun hyökkäys oli jo tapahtunut. Reaaliaikaisesta tarkastuksesta voitiin vain haaveilla. [17]

80-luvulla massamuistien hinnan aletessa ja kapasiteetin kasvaessa lokit tallennettiin tietokoneille, jolloin niitä voitiin analysoida koneellisesti. Koneiden laskentatehot olivat kuitenkin vaatimattomia ja siitä syystä analysointiohjelmia ajettiin pääsääntöisesti yöllä, jolloin muuta kuormaa ei ollut. Tästä syystä monet hyökkäykset havaittiin vasta niiden tapahduttua. 90-luvun alussa kehitettiin reaaliaikainen tarkastaja, joka analysoi dataa sitä mukaan kun sitä tuotettiin. Tämä johti hyökkäysten reaaliaikaiseen havaitsemiseen ja jopa torjuntaan. [17]

Nykyään tunkeutumisyritykset ja erilaiset porttien skannaamiset ovat arkipäivää ja IDS-järjestelmien lokit täyttyvät eriasteisista hälytyksistä ja tapahtumista. Tämä on asettanut uuden haasteen tunkeutumisien raportoinnista ylläpitäjille, sillä lokitietojen määrästä johtuen yksittäisiä tunkeutumishuippuja on mahdotonta havaita.

Suojatun verkon skannaaminen ei tarkoita vielä, että hyökkääjät olisivat keksineet keinon tunkeutua suojattuun verkkoon, vaan skannausta tapahtuu jatkuvasti. Yleensä hyökkääjät skannaavat laajoja verkkoalueita automaattisilla skannausohjelmilla, jolloin skannauksia tulee tasaiseen tahtiin [18]. Portteja tutkivat myös erilaiset madot ja virukset, jotka yrittävät löytää tietyn haavoittuvuuden omaavia laitteita saastuttaakseen ne. Porttiskannaukset voidaan siis ajatella internetin normaaliksi taustakohinaksi [18].

Kun hyökkääjät ovat päättäneet tunkeutua suojattuun verkkoon, aloittavat he kohdistetut skannaukset verkon tutkimiseksi ja haavoittuvuuksien löytämiseksi laitteista. Skannaukset tulevat harvoin vain yhdeltä koneelta, sillä hyökkääjät yleensä hajauttavat tutkimista, jottei heidät toimiaan havaittaisi helposti. IDS-järjestelmien tulisi tunnistaa tällainen systemaattinen tutkiminen, sillä se saattaa enteillä tulevaa hyökkäystä suojattua verkkoa vastaan.

IDS-järjestelmät suojaavat verkkoa myös sen sisällä tapahtuvilta luvattomilta toimilta [10]. Eräs yritysten suurimpia tietoturvauhkia ovat yritysten omat työntekijät varsinkin irtisanomis- ja riitatilanteissa. Pääkäyttäjän oikeudet omaavaa käyttäjää on vaikea estää tekemästä tihutöitä yrityksen sisäisessä verkossa. IDS-järjestelmät tarkkailevat myös verkon sisäistä liikennettä ja kykenevät puuttumaan havaitsemiinsa epäkohtiin.

3.2 IDS-järjestelmätyypit

IDS-järjestelmät jaotellaan yleensä niiden toimintaperiaatteiden perusteella kahteen kategoriaan: verkkotason havaitsemisjärjestelmiin ja konetason havaitsemisjärjestelmiin. Luvussa esitetään myös uudempia tyyppejä, kuten hajautetut havaitsemisjärjestelmät ja langattomat IDS-järjestelmät.

3.2.1 Verkkotason havaitsemisjärjestelmä

Verkkotason havaitsemisjärjestelmä (engl. *Network-based Intrusion Detection System, NIDS*) tarkkailee verkon segmenttiä tai aliverkkoa. NIDS on verkossa toimiva haistelija (engl. *Sniffer*), joka tutkii jokaisen sille lähetetyn paketin sisällön. NIDS voidaan asettaa tarkkailemaan liikennettä, joka on osoitettu vain sille (nonpromiscuous) tai se voi tarkkailla kaikkea liikennettä, joka tulee sen verkkoliitäntään (promiscuous). NIDS-laite voidaan myös asettaa täysin huomaamattomaan tilaan, jolloin sen olemassaoloa ei voi mitenkään havaita verkosta käsin. Tämä saadaan aikaan jättämällä NIDS-laitteen verkkokortti ilman IP-osoitetta ja käyttämällä TAP-laitetta, jonka avulla voidaan estää datan lähettäminen takaisin verkkoon. Jos NIDS-laite ei voi lähettää hälytyksiä valvontaan käytetyn verkkoliitännän kautta, tulee siihen asentaa toinen verkkoliitäntä hälytyksiä varten. [18]

NIDS-laitteen etuja on, ettei sillä ole vaikutusta valvottavaan järjestelmään tai verkkoon. Se ei lisää verkon liikennettä tai palvelinten kuormaa. Myös NIDS-laitteen huomaamattomuus passiivisena on sen etuja. Hyökkääjä ei pääse käsiksi havaitsemisjärjestelmään, eikä välttämättä ole tietoinen sen olemassaolosta. [18]

NIDS-järjestelmän heikkouksia on verkkokapasiteetin riittämättömyys nopeissa ja kuormitetuissa verkoissa [18]. Jos kaikki reitittimeen tuleva liikenne monistetaan NIDS-järjestelmälle, voi NIDS-laitteen verkkolinkki tukkeutua ja hidastaa verkon toimintaa. NIDS-laitteita voi olla useita, jolloin niiden kuormitus voidaan jakaa, eikä kaikkea liikennettä tarvitse kierrättää yhden reitittimen kautta. Toinen heikkous perustuu hyökkäyskuvioiden tuntemiseen. Kun uusi hyökkäystapa tai heikkous tulee ilmi, hyökkääjät voivat käyttää sitä hyväkseen niin kauan, kunnes hyökkäyskuvio lisätään NIDS-järjestelmän tuntemiin hyökkäysmalleihin (engl. *Signature, pattern*) [10]. Hyökkäysmallit ovat ennalta määritettyjä sääntöjä, joiden perusteella IDS-järjestelmät päättävät toteuttaako tietty liikenne hyökkäyksen tunnusmerkit. Hyökkäysmallien päivittämisen takia NIDS-järjestelmät ovat aina askeleen perässä viimeisimmistä hyökkäystrendeistä. Hyökkäysmallien päivittämisnopeus onkin yksi kriittisimmistä NIDS-järjestelmien heikkouksista, sillä mitä kauemmin päivittäminen kestää, sitä pidemmän aikaa verkko on alttiina uudelle turva-aukolle. Myös

kytkentäisten verkkojen yleistyminen vaikeuttaa NIDS-järjestelmien toimintaa. Kytkentäisissä verkoissa esimerkiksi kaikki verkon sisäinen liikenne ei välttämättä tule reitittimelle, johon NIDS-laite on kytkettynä [10]. Tällöin tunkeutumisenhavaitsemisjärjestelmä ei pysty analysoimaan kaikkea liikennettä. NIDS-järjestelmä ei kykene myöskään tutkimaan salattuja yhteyksiä, koska sillä ei ole tietoa yhteyden eri osapuolien sopimista salausavaimista [13]. Jos salausavain saataisiin selville verkossa liikkuvista paketeista, voisi kuka tahansa purkaa salauksen.

3.2.2 Konetason havaitsemisjärjestelmä

Konetason havaitsemisjärjestelmä (engl. *Host-based Intrusion Detection System, HIDS*) eroaa verkkotason havaitsemisjärjestelmästä kahdella tavalla. Konetason havaitsemisjärjestelmä suojaa vain järjestelmää, johon se on asennettu, eikä koko verkkoa tai aliverkkoa. Yleensä HIDS-järjestelmä kuuntelee vain laitteelle osoitettua liikennettä ja jättää muun liikenteen huomioimatta.

Konetason havaitsemisjärjestelmät ovat yleensä ohjelmia, jotka asennetaan jokaiseen tarkkailtavaan laitteeseen erikseen. Tästä syystä se vaatii paljon enemmän suunnittelua ja ylläpitoa kuin NIDS-järjestelmä. Koska usean sadan laitteen suojaaminen HIDS-järjestelmällä on epätaloudellista, tyydytään usein vain kriittisten laitteiden suojaamiseen HIDS-järjestelmällä. Muissa laitteissa voidaan käyttää tällöin esimerkiksi NIDS-järjestelmää.

Konetason havaitsemisjärjestelmät voivat erota toisistaan huomattavasti. Yleensä HIDS-järjestelmät tarkkailevat isäntälaitteen lokeja etsien epäonnistuneita kirjautumisyrityksiä, käyttöoikeusrikkomuksia ja muita mahdollisia väärinkäytöksiä. Edistyneemmän HIDS-järjestelmät tarkkailevat järjestelmän prosesseja etsien niiden joukosta haittaohjelmia ja sulkevat löytämänsä haitalliset prosessit. Nämä HIDS-järjestelmät suojaavat laitetta esimerkiksi tunnettuja troijalaisia vastaan. [10]

Lokien tutkiminen voi tapahtua joko ajastetusti tai reaaliajassa. Jos hyökkääjä saa järjestelmän hallintaansa, jossa on HIDS-järjestelmä, voi hyökkääjä yrittää sammuttaa HIDS-järjestelmän. Ajastetulla lokien tutkimisella tämä on ongelma, sillä HIDS ei välttämättä ehdi huomata tunkeutumista. Reaaliajassa tutkiva järjestelmä ehtii todennäköisesti lähettämään hälytyksen ennen kuin hyökkääjä ehtii sulkemaan sitä. Reaaliaikaisen tutkimisen haittapuolena on suurempi järjestelmäresurssien kulutus. [13]

HIDS-järjestelmät voivat myös tarkkailla järjestelmät tiedostoja ja käyttöjärjestelmää ja havaita niissä tapahtuneita muutoksia. Muutoksia voidaan tarkkailla laskemalla tiedostoista MD5-tiivistesummia (*Message Digest 5 Hash*) ja tarkastamal-

la niitä aiemmin laskettuihin tiivistesummiin tai tarkkailemalla tiedostojen kokoja. Näin voidaan huomata järjestelmälle kriittisten tiedostojen pahantahtoinen muuttaminen. HIDS-järjestelmä voi myös tarkkailla järjestelmän sisäisiä kutsuja, joiden avulla voidaan käyttää hyväksi tunnettua tietoturva-aukkoa. HIDS-tarkkailee myös järjestelmän sisäistä liikennettä, joka ei koskaan näkyisi NIDS-järjestelmälle. [18]

Eräs HIDS-järjestelmän etuja on lokitietojen tutkiminen ja keskittäminen [10]. HIDS-järjestelmät voivat keskittää monen laitteen lokitiedot yhteen, helposti tulkittavaan, lokiin. Yleensä verkkojen ylläpitäjät eivät pysty seuraamaan kaikkien verkossa olevien laitteiden lokeja riittävät tarkasti. HIDS-järjestelmien sääntöjä voidaan myös räätälöidä laitekohtaisesti, jolloin HIDS-järjestelmän toimintaa voidaan tehostaa kytkemällä pois tarkastuksia, joita ei tarvita kyseisessä laitteessa [18]. Tämä myös vähentää väärien hälytysten määrää, sillä jos laitteessa ei ole tiettyä palvelua, ei sitä koskevia hälytyksiä tarvitse noteerata.

HIDS-järjestelmillä on samoja haittapuolia kuin NIDS-järjestelmilläkin. Ne suojaavat vain tunnettuilta hyökkäyksiltä ja haittaohjelmilta. Jos järjestelmiä ei ole päivitetty tai niihin ei ole saatavilla uusimpia hyökkäysmalleja nopeasti, eivät ne suojaa järjestelmää halutulla tavalla. HIDS-järjestelmät kuluttavat myös isäntälaitteen resursseja ja tämä saattaa olla joissakin reaaliaikajärjestelmissä kriittinen ominaisuus. [10]

3.2.3 Hajautetut havaitsemisjärjestelmät

Mielenkiinto hajautettuja havaitsemisjärjestelmiä (engl. *Distributed Intrusion Detection Systems, DIDS*) kohtaan on kasvanut viime aikoina, sillä niillä voidaan valvoa suuria hajautettuja tietoverkkoja, joiden valvominen tavallisilla NIDS- tai HIDS-järjestelmillä on hankalaa. DIDS-järjestelmä koostuu yleensä sekä NIDS- että HIDS-sensoreista, jotka ovat hajautettuna eri puolille verkkoa. Sensorit lähettävät hälytykset keskuslaitteelle, joka analysoi ja yhdistää eri sensorien tiedot ja saa näin kokonaiskuvan koko verkon toiminnasta, toisin kuin yksittäinen sensori, joka näkee vain oman aliverkkonsa tapahtumat. Hyökkäysmallit luodaan ja päivitetään keskuslaitteella, josta ne ladataan sensoreille tarvittaessa. Sensoreille voidaan ladata vain sellaisen mallit, joita niiden tulee tarkkailla verkkosegmentissään tai kohdelaitteessaan. Keskuslaitteen ja sensorien välinen kommunikaatio tulisi välittää vain niille tarkoitetussa erillisessä verkossa, jotta hyökkääjät eivät pääsisi käsiksi laitteiden viesteihin. Jos liikenne joudutaan välittämään samassa verkossa, jota DIDS tarkkailee, tulisi liikenne salakirjoittaa, jotta sen oikeellisuus voitaisiin taata. Erillisen verkon käyttö suojaa myös verkon ylikuormittumisella, joka voi tapahtua suojattavassa

verkossa esimerkiksi madon aiheuttaman haitallisen liikenteen johdosta. [18]

Hajautetun järjestelmän ongelmaksi muodostuu yleensä sensoreilta tulevan datan määrä, jonka keskuslaite joutuu käsittelemään. Nykyään tiedon lajittelussa käytetään apuna tiedonlouhintaa (engl. *Data mining*) sekä ryhmittelyä (engl. *Cluster*). Dataa voidaan ryhmitellä sen ominaispiirteiden mukaan erilaisiin ryhmiin, jolloin IDS-laitteiden datankäsittely nopeutuu, kun järjestelmän tarvitsee tutkia vain kyseisen ryhmän sisältämät säännöt. [19]

3.2.4 Langattomat IDS-järjestelmät

IDS-järjestelmät toimivat pääsääntöisesti langallisissa verkoissa, joissa hyökkäykset tapahtuvat yleensä ulkoverkosta käsin tietyn solmukohdan kautta. Langattomien verkkojen suosion kasvu viimeisten vuosien aikana on johtanut IDS-järjestelmien tarpeeseen langattomissa ympäristöissä kuten WLAN- ja mobiiliverkoissa.

Langallisissa verkoissa hyökkääjien on hyvin vaikea päästä kuuntelemaan verkkoa kytkeytymällä siihen fyysisesti, mutta langattomissa verkoissa tämä on helppoa, sillä hyökkääjän tarvitsee olla vain verkon kantaman sisäpuolella päästäkseen käsiksi verkossa liikkuvaan dataan. Langattomat IDS-järjestelmät ovat hyvin samankaltaisia kuin langallisissa ympäristöissä toimivat IDS:t, mutta niiden sijoittelussa on erityisvaatimuksia. Lisäksi ne sisältävät lisäominaisuuksia tunkeutujien havaitsemiseksi WLAN:ssa.

Langattomia IDS-järjestelmiä on kahden tyyppisiä: keskitettyjä ja keskittämättömiä. Keskitetyt järjestelmät sisältävät vaihtelevan määrän sensoreita, jotka keräävät langattomissa verkoissa liikkuvan datan ja lähettävät sen erilliselle analysointilaitteelle, joka analysoi liikenteen etsien siitä viitteitä hyökkäyksistä. Keskittämätön IDS-järjestelmä koostuu muutamasta langattomasta IDS-järjestelmästä, jotka kaappaavat liikenteen ja analysoivat sen itse. Keskittämätön ratkaisu sopii muutaman langattoman tukiaseman (engl. *Access Point, AP*) verkkoihin sen kustannustehokkuuden takia. Suuremmissa verkoissa erillisten sensorien käyttäminen on tehokkaampaa ja IDS-järjestelmän hallinta on helpompaa erillisen analysointilaitteen takia. [20]

WLAN-verkot kattavat yleensä suuren alueen, jolloin käytetään useita tukiasemia tasaisen kuuluvuuden takaamiseksi. Langattoman IDS-järjestelmän sensorit tulisi asettaa jokaisen tukiaseman tuntumaan, jolloin ne havaitsevat suurimman osan verkon väärinkäyttöyrityksistä. Sijoittamalla sensorin tukiasemien läheisyyteen, voidaan hyökkääjä paikallistaa fyysisesti tietyn tukiaseman kuuluvuusalueelle. [20]

Langattomiin verkkoihin, joiden toiminta perustuu tukiasemiin, voidaan asentaa kohtuullisen helposti IDS-järjestelmä, mutta suuremman ongelman muodostavat ad hoc -verkot, joissa erilaiset mobiililaitteet muodostavat verkon solmut ilman tukiasemia. Liikenne ohjataan mobiililaitteelta toiselle kohti määränpäätään. Mobiililaitteiden liikkuminen sulkee pois kiinteästi asennettavien sensorien käytön, sillä verkon koostumus ja sijainti vaihtelevat mobiililaitteiden liikkeen mukaan. [21]

Ad hoc -verkot asettavat useita ongelmia perinteiselle IDS-järjestelmälle. Ensimmäinen ongelma on kiinteän keskipisteen puuttuminen, josta verkossa liikkuvaa dataa voidaan helposti kerätä. Tästä syystä IDS-järjestelmät voivat tarkkailla vain laitteen radiokantaman sisällä liikkuvaa dataa. Toinen ongelma on verkon topologian muuttuminen, joka estää keskitetyn analysointilaitteen käytön, sillä sen sijainnista ja olemassaolosta ei voida olla varmoja. Kolmas ongelma koskee hyökkäysmallien päivittämistä mobiililaitteille, sillä laitteet voivat operoida ilman yhteyttä muihin laitteisiin. Viimeisenä mobiililaitteen fyysinen suojaaminen on hankalaa, jolloin se voidaan kaapata ja hyökkääjä voi käyttää sitä omiin tarkoituksiinsa. [22]

Ad hoc -verkkojen haavoittuvin kohta on reititys, joka voidaan sekoittaa suhteellisen helposti. Langallisissa verkoissa reititys tapahtuu erillisten reitittimien avulla tunnettuja reittejä pitkin, joita hyökkääjät harvoin saavat hallintaansa, mutta langattomissa verkoissa oikeaa reittiä harvoin tiedetään etukäteen. Hyökkääjän tarvitsee vain tuoda oma mobiililaitteensa muiden laitteiden kantoalueelle, jolloin hänen laitteensa osallistuu reititykseen. [21]

Reititystä ja reitinhakua uhkaavat kaapatut laitteet, jotka voivat lähettää väärennettyjä reititystietoja, ilmoittaa olemattomista laitteista sekä tehdä palvelunestohyökkäyksiä hukuttamalla muita laitteita reititysliikenteellä. Kaapatut laitteet voivat hyökätä verkkoa vastaan passiivisesti tai aktiivisesti. Passiivinen hyökkäys sisältää datan kuuntelua, kun taas aktiiviset hyökkäykset pyrkivät muuttamaan tai estämään verkossa liikkuvaa dataa eri tavoin. Autentikoinnilla ja oikeellisuuden tarkistamisella voidaan suojautua verkon ulkopuolisten solmujen tekemiltä aktiivisilta hyökkäyksiltä. Ongelmaksi muodostuu, jos verkossa jo oleva solmu joutuu hyökkääjän valtaan ja alkaa muuttamaan tai estämään liikennettä. [21]

Ad hoc -verkkojen rakenteen takia niissä voidaan käyttää vain HIDS-järjestelmän kaltaisia IDS-järjestelmiä, jotka asennetaan jokaiseen mobiililaitteeseen. Näin verkon solmut tarkkailevat itsenäisesti verkossa liikkuvaa liikennettä. Langallisissa verkoissa käytettävät IDS-järjestelmät eivät sovi suoraan mobiiliverkkoihin, sillä niiden suunnittelussa ei ole otettu huomioon verkon kiinteiden solmujen puuttumista tai virrankulutusta, joka määrää mobiililaitteiden laskentatehoa. Kiinteän solmukohdan puuttuminen haittaa mobiililaitteiden IDS-järjestelmiä, sillä ne voivat

käyttää vain kuuluvuusalueellaan kulkevaa liikennettä tarkastamiseen ja osa hyökkäyksen ilmaisevasta liikenteestä voi kulkea kohteeseen eri reittiä pitkin. Jos IDS-järjestelmät toimivat yhteistyössä muiden mobiilisolmujen kanssa, tulee niiden suhtautua epäluuloisesti naapurisolmuihin, sillä koskaan ei voida olla varmoja, ettei naapurisolmu ole joutunut hyökkääjän hallintaan ja lähetä väärennettyjä tietoja. Ad hoc -verkoissa toimivien IDS-järjestelmien tulee olla yhtä varuillaan verkon sisäpuolelta kuin ulkopuoleltakin tapahtuvista hyökkäyksistä. [21]

Tutkimuksessaan Zhang ja Lee [23] esittivät idean hajautetusta IDS-järjestelmistä, jotka toimivat yhteistyössä keskenään ja jossa naapurisolmut osallistuvat tunkeutumisien havaitsemiseen sekä vastatoimiin. Mallin mukaan jokaisessa mobiililaitteessa on IDS-agentti, joka kerää dataa ja tarkastaa käyttäjien ja järjestelmien toimia laitteessa sekä liikennettä laitteen kuuluvuusalueella. Agentti on jaettu kahteen moduuliin, joista toinen hoitaa lokaalia tarkastusta ja toinen yhteistyötä muiden mobiililaitteiden kanssa. Yhteistyö aloitetaan, jos tarkastetusta datasta löydetään viittaus hyökkäykseen tai analyysin varmistamiseen tarvitaan muiden mobiililaitteiden apua. Myöhemmin Jiang, Gao, Zhang ja He esittivät tutkimuksessaan [24] samanlaisella logiikalla toimivan aluepohjaisen IDS-järjestelmän (engl. *Zone-Based Intrusion Detection System, ZBIDS*) ad-hoc-verkkoihin kuin Zhang ja Lee omassa tutkimuksessaan. ZBIDS-järjestelmä sisältää Markovin ketjuun perustuvan väärinkäytöstenhavaitsemisjärjestelmän. [21]

Lee, Han ja Shin esittivät tutkimuksessaan [25] kahden uuden kontrollointiviestin lisäämistä ad-hoc-verkkojen reititysprotokollaan (DSR-protokolla). Reitin varmistuspyyntöä (engl. *Route Confirmation Request, CREQ*) ja reitin varmistusvastausta (engl. *Route Confirmation Reply, CREP*) käytetään reitin varmistamiseen, jolloin voidaan vähentää väärinkäyttäytyvien solmujen vaikutusta reititykseen. Viestejä käytetään reitinhaun yhteydessä, jolloin seuraavalta reitin varrella olevalta solmulta kysytään ennakkoon varmistus, että myös se tuntee reitin kohdesolmuun.

3.3 Analysointimenetelmät

IDS-järjestelmät käyttävät erilaisia analysointimenetelmiä hyökkäyksien tunnistamiseen. Eri analysointimenetelmien tulisi havaita hyökkäykset mahdollisimman luotettavasti ja niiden ei tulisi tuottaa vääriä hälytyksiä. Väärät hälytykset koostuvat pääosin järjestelmien luvallisesta käytöstä, jota pidetään hyökkäyksenä (engl. *False alarm, false positive*). Vaarallisempi on tapahtuma on hyökkäyksen pitäminen normaalina järjestelmän käyttönä (engl. *False negative*). Tällöin hyökkäyksestä ei tehdä hälytystä. IDS-järjestelmä voi olla liian salliva, jolloin se ei havaitse kaikkia hyök-

käyksiä, tai se voi olla liian tiukka, jolloin väärien hälytysten määrä kasvaa. [13]

IDS-järjestelmien käyttämät analysointimenetelmät voidaan jakaa kahteen ryhmään: väärinkäytösten tunnistamiseen sekä poikkeavuuksien tunnistamiseen. [13]

3.3.1 Väärinkäytösten tunnistaminen

Väärinkäytösten havaitseminen perustuu tunnettujen hyökkäysmallien tarkasteluun. Jos verkon liikenne vastaa aiemmin hyökkäykseksi todettua kuviota, päätellään hyökkäyksen olevan käynnissä. IDS-järjestelmien hyökkäysmallit tulee olla ajan tasalla, sillä päivittämätön IDS-järjestelmä ei pysty tunnistamaan uusimpia hyökkäyksiä. Pienetkin muutokset hyökkäysmalleissa voivat harhauttaa IDS-järjestelmää, joten väärinkäytöksiin perustuvan analyysin kiertäminen on mahdollista. Mallien tulisi olla tarpeeksi yksityiskohtaisia, jotta vältyttäisiin vääriltä hälytyksiltä sekä tarpeeksi laajoja, jotta hyökkäykset tunnistettaisiin, vaikka niissä tapahtuisi pieniä muutoksia. Tämä tekee hyökkäysmallien laatimisesta hyvin haastavaa. [13]

Yksinkertainen esimerkki hyökkäysmallista, jonka tarkoituksena on havaita UDP pommi -hyökkäys (engl. *UDP Bomb attack*):

```
alert udp any 19 <> any 7
```

Hälytys tapahtuu, jos IDS-järjestelmä havaitsee UDP-paketin, joka on lähtöisin jonkin IP-osoitteen portista 19 ja kohteena on jonkin IP-osoitteen portti 7.

Tunnistamisessa tarkkaillaan verkossa liikkuvien pakettien otsikkotietoja. IDS-järjestelmä voi tutkia vain protokollia, jotka se itse tuntee. Väärin muodostetut pakettien otsakkeen johtavat yleensä hälytykseen. [13]

Protokollien analysoinnissa on ongelmallista monimutkaiset protokollat, kuten TCP/IP, jotka sisältävät paljon erilaisia tiloja ja tapahtumia. Näiden protokollien oikea tulkinta vaatii monien aikaisempien pakettien tutkimista, jotta voidaan päätellä, onko protokolla sellaisessa tilassa, että käsiteltävä paketti voidaan sallia. Esimerkiksi TCP-yhteyksissä IDS-järjestelmä joutuu rekonstruoimaan koko yhteyden, jotta se voi tulkita yhteydessä liikkuvaa dataa. [26]

IDS-järjestelmät tarkkailevat myös paketeissa olevaa dataa etsien niistä ennalta määrättyjä bitti- tai merkkijonoja (engl. *packet grepping*). Esimerkkinä merkkijonosta voi olla `/etc/passwd`. `Passwd`-tiedosto sisältää Unix-järjestelmissä käyttäjien salasanat ja näin sen pyytäminen ulkoverkosta käsin voidaan tulkita hyökkäykseksi. Merkkijonojen havaitsemiseen perustuvat analysointimenetelmät voidaan kiertää joissakin tapauksissa. [13]

Eräs väärinkäytösten analysointimenetelmä on protokolla-analyysi (engl. *Protocol analysis*), jossa IDS tutkii paketin sovellustason protokollan sisältöä ja etsii siitä

merkkejä hyökkäyksestä. Esimerkiksi HTTP-protokollassa olevaa satojen merkkien mittaista URL-osoitetta voidaan pitää hyökkäyksenä, jolla yritetään käyttää hyväksi selaimessa olevaa tietoturva-aukkoa. Analyysiä voidaan käyttää vain IDS:n tuntemille protokollille ja IDS:n tulisi käsitellä paketteja samalla tavalla kuin kohdelaitteen. [13]

Tilallinen protokolla-analyysi on neljäs analyysitapa, jolla hyökkäyksiä voidaan tunnistaa. Se toimii samalla tavalla kuin tilallinen suodatus palomuureissa. IDS-järjestelmä muistaa yhteyden tilan ja voi sen perusteella etsiä hyökkäyksiä. Esimerkiksi `passwd`-tiedoston kopiointi ei välttämättä merkitse hyökkäystä, ellei tiedostoa yritetä kopioida `/etc`-hakemistosta. [13]

3.3.2 Poikkeuksien havaitseminen

Poikkeuksien havaitsemisanalyysit perustuvat tietojärjestelmien tarkkailuun ja niiden tavanomaisesta käytöstä poikkeavien tapahtumien etsimiseen. Tavanomaisesta käytöstä poikkeavan tapahtuman sattuessa IDS päättelee hyökkäyksen olevan mahdollisesti käynnissä. [13]

Poikkeavuuksia etsitään tietojärjestelmien resurssien käytöstä kuten verkkoliikenteen määrästä, prosessorin kuormituksesta, kiintolevyjen vapaan tilan määrästä, epäonnistuneiden kirjautumisien määrästä ja niin edelleen. Poikkeavuuksille voidaan asettaa raja-arvoja joiden ylittyessä toiminta ei ole enää normaalia järjestelmän käyttöä. Poikkeavuudet havaitaan yleensä vertaamalla järjestelmän senhetkistä toimintaa aiemmin luotuihin tilastoihin tai jopa tekoälyn avulla. IDS-järjestelmät voivat myös oppia järjestelmän normaalin käytön antamalla niiden seurata järjestelmää tarpeeksi kauan. Tällöin tulee olettaa, ettei järjestelmää vastaan hyökätä opetuksen aikana. IDS-järjestelmät voivat myös muokata aiemmin luotuja malleja oppimisen avulla. Tällöin IDS-järjestelmät voivat tottua tapahtumiin, joita aikaisemmin pidettiin epänormaaleina. [13]

Poikkeuksien havaitsemisjärjestelmien opettaminen on ongelmallista, sillä opetuksen aikana verkon liikenne ei saa sisältää hyökkäyksiä, mikä on hyvin hankala saavuttaa. Data voi sisältää myös uusia hyökkäyksiä, joita ei vielä tunneta. Jos harjoitusdata sisältää hyökkäyksen, voi IDS-laite omaksua sen verkon normaaliksi toiminnaksi, eikä hälytä siitä seuraavalla kerralla. Poikkeuksien havaitsemisjärjestelmiä voidaan opettaa simuloimalla hyökkäyksiä, jolloin järjestelmä osaa paremmin havaita hyökkäykset, mutta simuloinnit voivat tapahtua vain tunnetuilla hyökkäyksillä. [19]

Esimerkkinä poikkeavuuden havaitsemisesta voisi olla käyttäjä, joka kirjautuu

järjestelmään ainoastaan arkipäivinä työaikaan. Kun IDS-järjestelmä havaitsee käyttäjän kirjautuvan järjestelmään viikonloppuyönä, voidaan päätellä, ettei se kuulu käyttäjän normaaliin toimintaan ja IDS antaa siitä hälytyksen.

Poikkeuksien havaitseminen vaatii runsaasti laskentatehoa, joten sen avulla ei voida saavuttaa toistaiseksi reaaliaikaista suojaa kuten väärinkäytösten havaitseminen voi saavuttaa. Poikkeavuuksien tunnistaminen aiheuttaa myös enemmän vääriä hälytyksiä kuin väärinkäytösten havaitseminen. [13]

3.4 Hälytysten käsitteleminen

Tietoturvan kannalta IDS-laite tulisi kytkeä erilliseen verkkoon, jonka kautta hälytykset lähetetään. Jos hälytykset lähetetään samaan verkkoon jossa on hyökkäys käynnissä, voi hyökkääjä havaita hälytysviestit ja onnistua pysäyttämään ne. Rakentamalla erillinen verkko, johon kytketään vain NIDS-laitteet sekä niiden hallintaan käytettävät laitteet, estetään hyökkääjää manipuloimasta viestejä. Konetason havaitsemisjärjestelmiä ei saa liittää tähän verkkoon, sillä silloin on mahdollisuus, että niihin murtaudutaan ja saadaan pääsy suojattuun verkkoon. Jos IDS-laitteen tulisi reagoida hyökkäyksiin, eikä se pysty lähettämään paketteja analysointiportin kautta, tulee laitteeseen kytkeä uusi verkkokortti, joka kytketään reitittimen vapaaseen porttiin. [13]

Hälytysten määrän kasvaessa tulee ongelmaksi IDS-järjestelmän reagointi kuormitustilanteessa (engl. *Flood scenario*). Kuormitustilanne syntyy kun IDS järjestelmää kuormitetaan niin monella hälytyksellä, ettei se pysty käsittelemään niitä kaikkia. Hyökkääjä voi luoda jopa palvelunestotilanteen kuormittamalla IDS-järjestelmää hälytyksillä. [12]

Monet IDS-järjestelmät kärsivät ”ensimmäinen osuma”-paradoksista (engl. *First mach paradox*), jossa järjestelmän tulee päättää hälytyksestä joko ensimmäisestä hyökkäyskuviosta, jonka se löytää signature-tiedoistaan tai etsimällä vastaavuuksia myös muista malleista. Esimerkkinä hyökkääjä voi tunnistaa signaturen, joka voidaan tulkita järjestelmän normaaliksi käytöksi tai muuksi vähäpätöiseksi tapahtumaksi, joka löytyy useista IDS-järjestelmistä. Käyttämällä tätä vähäpätöistä hyökkäyskuviota paljon vaarallisemmassa hyökkäyksessä voidaan hämätä IDS-järjestelmää, joka antaa hälytyksen vain ensimmäisestä löytämästään vastaavuudesta signaturetiedoissaan, ja lokeihin jää merkintä vain vähäpätöisestä tapahtumasta. Toisaalta järjestelmä, joka hälyttää jokaisesta löytämästään hyökkäyskuviosta on altis hälytysten kuormitukselle. Hyökkääjä voi lähettää verkkoon koko IDS:n signaturetiedoston, jonka seurauksena IDS-järjestelmä hukkuu hälytyksiin. Vaikka IDS-järjestelmä

suoriutuisi jopa kymmenien tuhansien hälytysten tulvasta, voidaan hyökkäyksen todellinen kohde hävittää muiden hälytysten sekaan. [12]

3.5 Hyökkäyksiin reagointi

IDS-järjestelmät voivat reagoida hyökkäyksiin eri tavoin. Passiivisessa reagoinnissa hyökkäyksestä ilmoitetaan verkon ylläpitäjälle esimerkiksi sähköpostiviestillä, SNMP-protokollan avulla tai jopa tekstiviestillä. Aktiivinen reagointi ryhtyy hyökkäyksen torjumiseen ilmoittamisen lisäksi.

Aktiivinen torjunta voi olla yksityiskohtaisen lokitietojen keräämistä hyökkäyksestä tai jopa sen torjunta. Yksityiskohtaisiin lokitietoihin voidaan kerätä esimerkiksi kaikki hyökkäyksen kohteena olevalle laitteelle tulevat paketit, jolloin hyökkäyksen myöhempi analysointi on helpompaa. Hyökkäyksen reittiä voidaan yrittää jäljittää traceroutella tai dns-kyselyillä. IDS voi jopa skannata hyökkääjän portteja ja ilmoittaa siinä mahdollisesti olevista tietoturva-aukoista ja hyökkääjän käyttöjärjestelmän. [18]

Hyökkäyksen torjunnassa IDS-järjestelmä voi katkaista hyökkääjän yhteyksiä lähettämällä RST-paketteja joko hyökkääjälle tai kohdelaitteelle tai muuttaa palomuurien sääntöjä, jotta ne estävät hyökkääjältä tuleva liikenne. Yleensä hyökkääjät käyttävät hyökkäykseen tietokoneita, joiden käyttäjät eivät tiedä koneidensa olevan hyökkääjän hallinnassa. Tästä syystä IP-osoitteiden kieltäminen palomuurissa tulisi purkaa hyökkäyksen loputtua, jotta tavallisilta käyttäjiltä ei evätä pääsyä palveluihin. Aktiivisista vastatoimenpiteistä on hyötyä silloin, kun verkon ylläpitäjä ei ole paikalla hyökkäyksen tapahtuessa. [13] [18]

Aktiivisen reagoinnin tarkoituksena on, ettei verkon ylläpitäjän tarvitse tarkkailla verkkoa koko aikaa ja samalla hyökkääjälle annetaan kuva, että hänen toimiaan jäljitetään. Aktiivinen reagointi voi tuoda ongelmia järjestelmän asennusvaiheessa. Jos IDS-järjestelmä on konfiguroitu väärin, voi se käynnistää vastatoimenpiteet esimerkiksi verkon DNS-palvelinta vastaan, jolloin koko verkon toiminta saattaa olla vaarassa. IDS-järjestelmää tulisi testata ilman aktiivista reagointia, kunnes on varmistuttu järjestelmän oikeasta toiminnasta. [18]

IDS-laitetta, joka estää tunkeutumisia, kutsutaan tunkeutumisenestojärjestelmäksi (engl. *Intrusion Prevention System, IPS*). Tunkeutumisenestojärjestelmä on muutoin sama kuin tavallinen IDS, mutta se sijoitetaan yksityisen verkon ja julkisen verkon solmukohdassa olevaan laitteeseen, jolla on hallinta verkossa liikkuvaan dataan (esimerkiksi ethernet silta, reititin tai palomuuri). IPS:n tehtävä on kontrolloida verkkoon pääsevää dataa ja tunkeutumisen havaitessaan se voi aloittaa aktiiviset

vastatoimet estämällä liikennettä pääsemästä verkkoon. Tavallisella IDS-laitteella ei ole kontrollia hyökkääjän ja kohteen väliseen verkkoon. [18]

3.6 IDS-järjestelmän sijoittaminen

Ennen IDS-järjestelmän sijoittamista tulisi verkon ylläpitäjän kartoittaa verkon topologia, jonne IDS-järjestelmä aiotaan sijoittaa. Huomionarvoisia kohtia ovat liityntäkohta internettiin ja extranettiin, etäkäyttöliitynnät sekä intranetin erottaminen.

Verkon internet liityntäkohta on todennäköisin reitti hyökkäyksille, sillä suurin osa hyökkäyksistä tulee verkon ulkopuolelta internetistä. Tästä syystä sen turvaaminen IDS-järjestelmällä on ensisijainen tehtävä. Kartoittamisessa on myös huomioitava muutkin mahdolliset internetliitynnät. Seuraavaksi tulee kartoittaa extranet liitynnät muiden organisaatioiden verkkoihin. Extranetit voivat muodostaa helpon reitin hyökkäyksille muiden organisaatioiden kautta. Toisaalta on huolehdittava siitä, että oman verkon kautta ei voida hyökätä muiden yritysten verkkoihin. Etäkäyttö on yleistynyt ihmisten liikkumisen myötä ja etäkäyttöliityntöjen kartoittaminen ja suojaaminen on hyvin tärkeää. Hyökkääjät etsivät usein suojaamattomia ja unohdettuja etäkäyttöliityntöjä, joiden kautta voidaan parhaimmassa tapauksessa ohittaa verkkoa suojaavat palomuurit ja IDS-järjestelmät. Viimeisenä kartoituksen kohteena on organisaation sisäisten rajojen analysointi. Organisaation sisällä voi olla useita erilaisia osastoja, joilla on yhteisiä palvelimia (esimerkiksi sähköpostipalvelin) sekä osaston sisäisiä palvelimia, joihin muilla osastoilla ei ole oikeutta. Ylläpitäjän tulee olla selvillä osastojen välisistä rajoista sekä liikenteestä, jonka sallitaan ylittää nämä rajat. [27]

Verkon topologian tarkastelun jälkeen ylläpitäjällä tulisi olla selvä käsitys mitä kautta hyökkäykset voivat tapahtua. Tämän jälkeen tulee kartoittaa verkossa olevat kriittiset kohdat ja laitteet. Tällaisia ovat yleensä erilaiset palvelimet (esimerkiksi sähköposti-, DNS-, DHCP ja WWW-palvelimet), verkon laitteet (reitittimet, kytkimet) sekä verkkoa suojaavat laitteet (palomuurit ja IDS-järjestelmät). Koska palomuurit ja IDS-laitteet suojaavat verkkoa, tulee niiden suojaamiseen kiinnittää suurta huomiota. Näiden laitteiden joutuminen hyökkääjän hallintaan voi johtaa koko verkon suojauksen romahdukseen. [27]

IDS-järjestelmä voidaan sijoittaa joko suojattavan verkon sisään tai ulkoisen verkon rajalle. Verkon rajalle sijoitettu IPS-järjestelmä kykenee kontrolloimaan pakettien pääsyä verkkoon, jolloin se voi estää haitallisten liikenteen pääsyä sisäverkkoon palomuurin tavoin. IPS-järjestelmien haittana on normaalin liikenteen estäminen, jos se analysoidaan hyökkäykseksi (engl. *false positive*). Myös IPS-laitteen suoritus-

kyky voi loppua, jolloin siitä muodostuu koko verkon pullonkaula ulkoverkon yhteyksille. Myös IPS-laitteen vikaantuminen jättää verkon suojamattomaksi ja mahdollisesti estää liikenteen sisäverkkoon. [18]

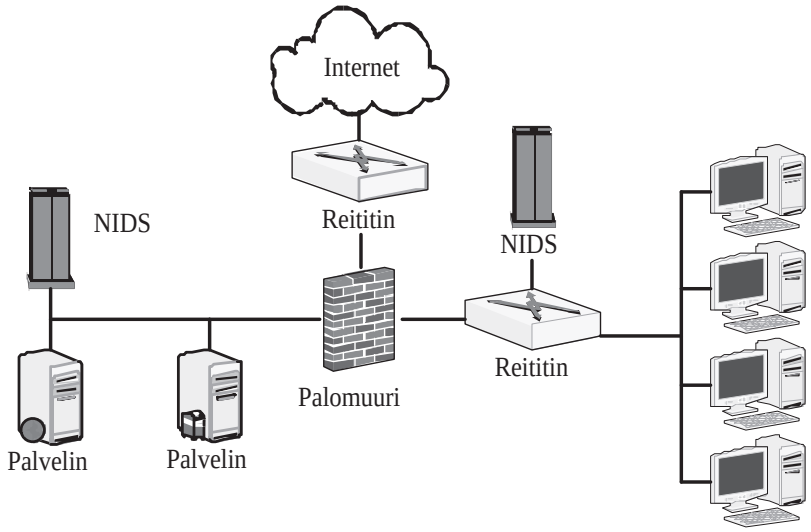
Seuraavissa luvuissa kuvataan eri IDS-järjestelmien sijoittamista verkkoon.

3.6.1 NIDS-järjestelmän sijoittaminen

Normaali NIDS-laitteen sijoittaminen tapahtuu verkon keskeisen keskittimen tai reitittimen yhteyteen, joka sijaitsee palomuurin takana. Jos laite on reititin, tulee kaikki reitittimen liikenne peilata NIDS-laitteelle. Tämä tapahtuu kytkemällä kytkimen jokin portti monitorointitilaan (engl. *Switch Port Analyzer, SPAN*) ja kytkemällä NIDS-laite tähän porttiin. Keskitin toistaa kaiken liikenteen jokaiseen ulostuloporttiin, joten sitä ei tarvitse erikseen konfiguroida. [18]

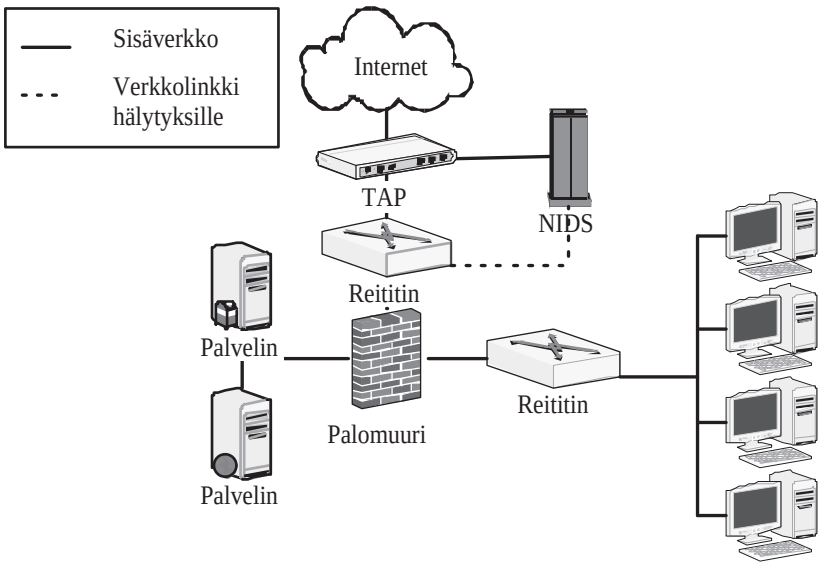
NIDS-järjestelmä on tehokkain verkossa, jossa reitittimiä on vähän, sillä reitittimien määrän kasvaessa liikenne ei välttämättä tule reitittimelle asti, johon NIDS-laite on kytketty. Verkon koon kasvaessa joudutaan verkkoon asentamaan useita NIDS-laitteita, joiden ylläpito saattaa hankaloitua. Varmin tapa on asentaa NIDS jokaiseen aliverkkoon, jolloin kaikki liikenne on varmasti NIDS-järjestelmien havaittavissa. NIDS-laite tulisi sijoittaa niin, että siihen saapuvat paketit kulkevat yhtä monen hypyn kautta IDS-laitteelle kuin ne kulkevat oikealle vastaanottajalle. [18]

Kuvassa 3.1 NIDS-laitteet on asennettu niin, että jokainen aliverkko on suojattu. Jos jokin julkinen palvelin onnistutaan murtamaan, ja sitä käytetään apuna hyökkäykseen, valvoo toinen NIDS-laite yksityisessä verkossa olevia työasemia. [18]



Kuva 3.1: NIDS-järjestelmien sijoittaminen. [18]

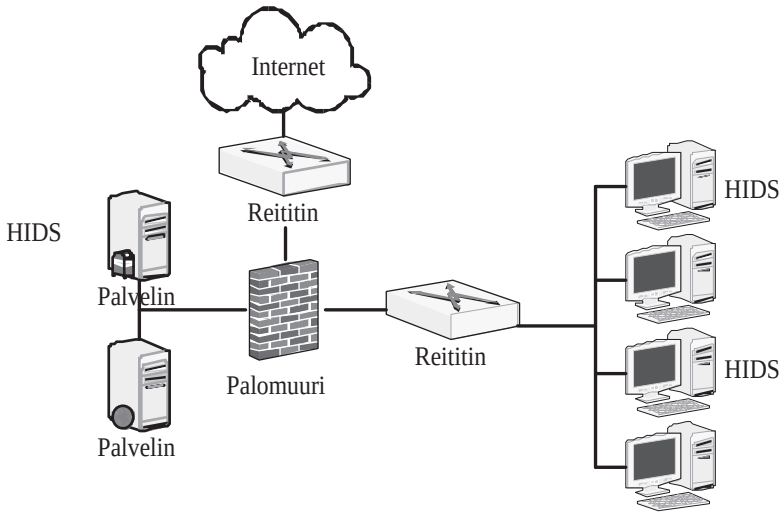
Kuvassa 3.2 on käytetty TAP-laitetta, joka monistaa tulevan liikenteen tarkkailuporttiin. TAP estää tarkkailuportista tulevan liikenteen pääsyn verkkoon, joten NIDS-järjestelmän hälytysviesteille tulee olla toinen verkkoliitäntä. TAP-laitteessa on neljä porttia: A, B, valvonta A (engl. *Monitor*) ja valvonta B. Porttiin A tuleva liikenne lähetetään portteihin B ja valvonta A, samoin porttiin B tuleva liikenne ohjataan porttiin A ja valvonta B. TAP-laitteella voidaan monistaa liikenne IDS-laitteelle ilman, että verkon muu toiminta häiriintyy. [13]



Kuva 3.2: NIDS-järjestelmien sijoittaminen TAP-laitteen avulla.

3.6.2 HIDS-järjestelmän sijoittaminen

HIDS-järjestelmä asennetaan jokaiseen laitteeseen, jota halutaan valvoa, sillä se valvoo vain järjestelmää, johon se on asennettu. Koska useiden kymmenien tai satojen HIDS-järjestelmien asentaminen ja ylläpitäminen on hankalaa, tyydytään HIDS asentamaan usein vain kriittisiin laitteisiin. Kuvassa 3.3 esitetään HIDS-järjestelmien asentaminen kriittiseen palvelimeen sekä muutamaan työasemaan. HIDS-järjestelmät voidaan kustomoida erilaisten tarpeiden mukaan. Esimerkiksi sähköpostipalvelinta suojaava HIDS voidaan konfiguroida tarkkailemaan vain sähköpostipalvelimia hyväksikäyttäviä hyökkäyksiä, kun taas WEB-palvelimen HIDS tarkkailee eri haavoittuvuuksia. [18]



Kuva 3.3: HIDS-järjestelmien sijoittaminen. [18]

3.6.3 DIDS-järjestelmän sijoittaminen

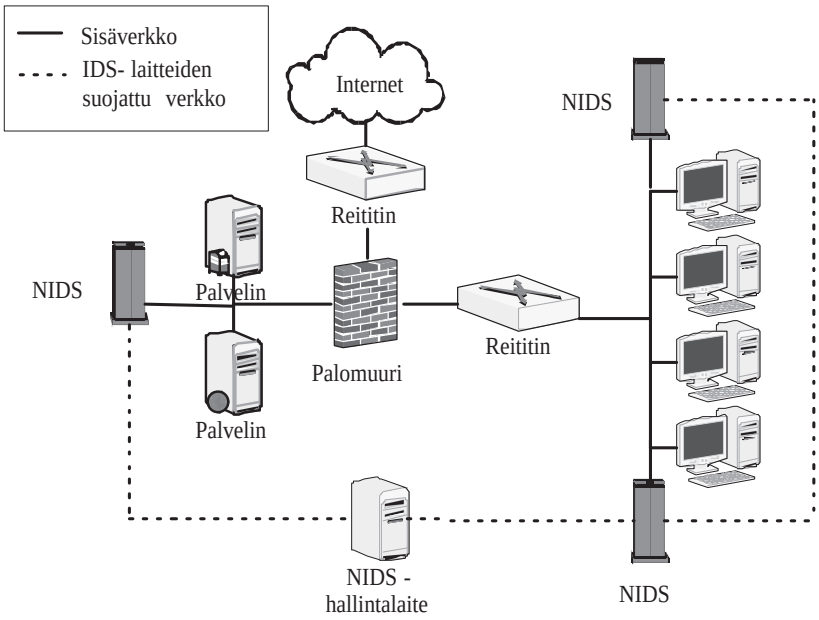
DIDS-järjestelmässä sensorit voidaan asettaa verkon eri kohtiin niin, että kaikki aliverkot ovat valvottuja. DIDS-laitteet keskustelevat keskenään ja hallintalaitteen kanssa erillisen verkon välityksellä, johon ei sallita muuta liikennettä. Kuvassa 3.4 on esitetty sensorien asettaminen verkkoon, yksi sensori valvoo palvelimia ja kaksi työasemia.

3.7 IDS-järjestelmien kehittäminen

Luvussa käsitellään IDS-järjestelmien kehittämistä tulevaisuudessa lisäämällä niihin uusia datankäsittely- ja päättelymenetelmiä järjestelmiin analysointimenetelmien tehostamiseksi.

3.7.1 Hi-RDA

Hi-RDA (*High-speed WAN Detection Response and Analysis*) on Richard Kemmererin ja Giovanni Vignan [28] kehittämä IDS-järjestelmä, joka koostuu verkoissa toimivista NIDS- ja HIDS-järjestelmistä sekä globaalista kontrollointijärjestelmästä. Hi-RDA:n ideana on, että yksityisissä verkoissa olevia IDS-järjestelmiä voidaan ohjata globaalisti, esimerkiksi internetin välityksellä, jolloin IDS-järjestelmät voisivat toimia yhteistyössä muiden verkkojen IDS-järjestelmien kanssa. Verkkojen IDS-laitteet ilmoittavat keskuslaitteelle tietonsa hyökkäyksistä ja keskuslaite voi ohjata muiden



Kuva 3.4: DIDS-järjestelmän sensorien sijoittaminen. [18]

verkkojen IDS-järjestelmiä vastaamaan hyökkäykseen niissä verkoissa, joiden kautta hyökkäys ohjautuu. [28]

Hi-RDA esittää myös tavan tutkia nopeissa verkoissa liikennettä reaaliaikaisesti NIDS-sensorien avulla. Nykyään uusien verkkojen kapasiteetti on gigabitti sekunnissa, joka aiheuttaa suuria ongelmia reaaliaikaiselle tarkastamiselle. Hi-RDA sisältää pilkkojan (engl. *Slicer*), joka pilkkoo datavirtaa pienempiin osiin ja jakaa sen useiden NIDS-sensorien kesken, jotka tarkastavat tietyt hyökkäyskuviot. Jotta tiettyjä hyökkäyskuvioita analysoiva sensori saa kaiken tarvitseman tiedon, pitää pilkkojan lähettää kaikki hyökkäykseen mahdollisesti vaikuttavat paketit kyseiselle sensorille, eikä hyökkäyksiä näin jää havaitsematta, koska tärkeät paketit on jaettu eri sensoreille. Hi-RDA sisältää myös toteutuksen verkon topologian kartoittamiseen, sekä sen käyttöön analysoinnin apuna. [28]

3.7.2 Tekoäly

Nykyiset IDS-järjestelmät perustuvat pääasiassa ennalta määrättyjen sääntöjen tarkastamiseen, jolloin uudet ja muunnellut hyökkäykset voivat päästä IDS-järjestelmän ohi. Säännöt muodostuvat ”jos-niin” pareista, jotka määrittelevät reagointitavat tiettyihin tapahtumiin.

Useita tutkimuksia on julkaistu IDS-järjestelmien kehittämisestä [29] [30] [31]. Tutkimuksissa esitetään tekoälyn, neuroverkkojen, koneoppimisen (engl. *Machine*

learning) ja tietojenlouhinnan (engl. *Datamining*) yhdistämistä IDS-järjestelmiin tarkoituksena luoda IDS-järjestelmä joka pystyy reaaliaikaiseen analysointiin suurellakin datamäärällä ja sopeutumaan uusiin hyökkäyksiin ilman ihmisen apua. Erilaisilla tekoälyratkaisuilla on erilaiset vahvuudet, toiset tekniikat antavat paremman tarkkuuden lyhyilläkin opetusjaksoilla, mutta ne kuluttavat paljon resursseja. Toiset taas kuluttavat vähemmän resursseja, mutta ovat epätarkempia.

Tekoälytekniikat perustuvat usein neuroverkkoihin, ja niiden sovelluksiin. Neuroverkot ovat verkkoja, jotka koostuvat useista solmuista (neuroneista). Ne perustuvat yhdistävään laskentaan, jossa käytetään apuna matematiikan tai laskennan malleja. Neuroverkot pyrkivät jäljittelemään aivojen neuronien toimintaa ja verkottumista. Neuroverkkoja opetetaan esimerkkidatan avulla, jolloin sen tulisi oppia eri muuttujien riippuvuussuhteet toisiinsa nähden. Neuroverkkoihin perustuvien tekoälytekniikoiden vahvuus on niiden oppimiskyky. Päivittämällä verkon rakennetta ja neuronien välisiä linkkejä, järjestelmät voivat oppia uusia laskentamalleja.

Tekoälytekniikoita ovat esimerkiksi tukivektorikone (engl. *Support Vector Machine, SVM*), joka perustuu neuroverkkoihin ja se lähestyy ongelmaa tilastollisen minimoinnin avulla. SVM:n teho perustuu tehokkaaseen luokitteluun, joka takaa hyvän yleiskuvan tilanteesta, ja tämän avulla SVM pystyy paremmin käsittelemään uusia ja muuttuneita hyökkäyksiä. Keinotekoinen neuroverkko (engl. *ANN, Artificial Neural Networks*) on tavanomainen neuroverkko, joka koostuu useista keinotekoisista neuroneista, jotka käyttävät matemaattista tai laskennallista mallia tiedon prosessointiin. [30]

Chavan, Shah, Dave, ym. esittivät keinotekoisten neuroverkkojen yhdistämistä sumeaan logiikkaan [31]. Sumea logiikka laajentaa tavanomaista logiikkaa ilmoittamalla reaalisia totuusarvoja diskreettien totuusarvojen sijaan (tosi/epätosi). Sumea logiikka ilmoittaa kuinka varmasti jokin asia on, tai kuinka paljon se on. Sumean logiikan arvot eivät esitä todennäköisyyttä jollekin tapahtumalle, vaan osallisuutta tiettyyn joukkoon (esimerkiksi 0.8 osallisuus tosi-joukkoon ja 0.2 epätosi-joukkoon).

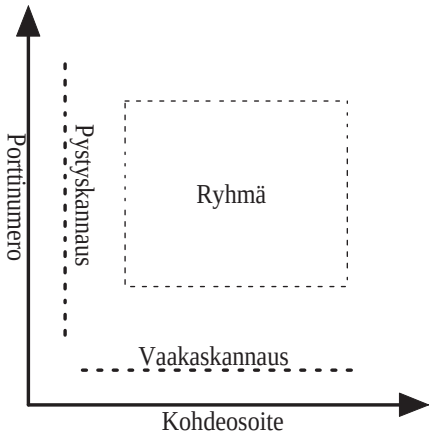
Sumean logiikan etuja on sen kyky jäljitellä ihmisen asiantuntemusta tallentamalla tarvittavat tiedot tietokantaan ja laskemalla tulokset sumealla päättelyllä. Päättelyyn tarvitaan kuitenkin paljon esitietoa järjestelmästä. Yhdistämällä sumea päättely neuroverkkojen oppimiskykyyn, voidaan IDS-järjestelmistä kehittää itsenäisesti toimivia, jolloin ne eivät tarvitse uusien sääntöjen luomiseen ihmisen apua. [31]

3.8 Lokien analysointi

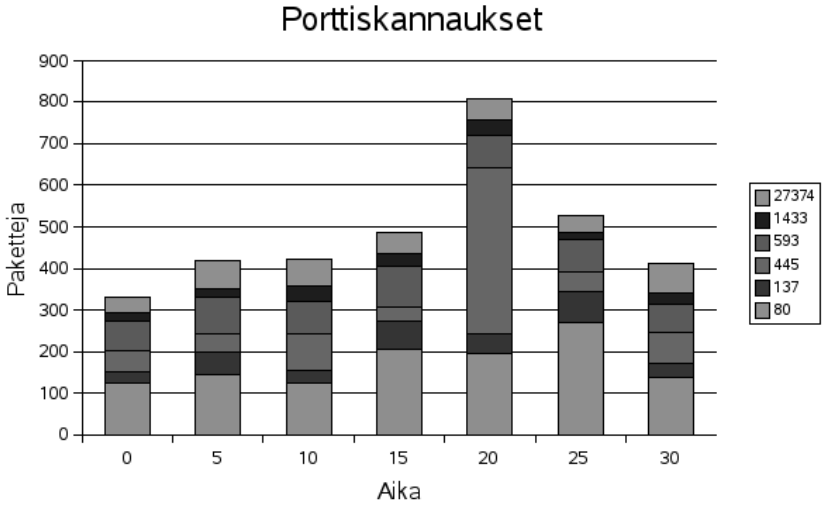
IDS-järjestelmien ongelmana on hälytysten esittäminen helposti tulkittavassa muodossa. Jos hälytyksiä tulee useita tuhansia ja ne esitetään vain listana tapahtumia, on niiden välisten riippuvuuksien tulkinta lähes mahdotonta. Vaikeasti tulkittavat lokit eivät paljasta, onko kyseessä yksittäinen hälytys vai osa suurempaa hyökkäystä. Matojen aiheuttamat suuret porttiskannaushyökkäykset, jotka kohdistuvat useaan laitteeseen samassa verkossa, eivät välttämättä erotu paljon hälytyksiä sisältävästä lokista, sillä laitteita ei skannata järjestyksessä IP-osoitteiden mukaan. Hyvin toteutetun visuaalisen kuvan avulla voidaan esittää enemmän tietoa nopeammin omaksuttavassa muodossa kuin tekstipohjaisessa lokissa. Tästä syystä IDS-järjestelmien käyttöliittymät esittävät hälytykset erilaisina graafeina, joista huomataan helposti hyökkäyspiikit tiettynä aikana. Aikaan ja hyökkäysten määrään perustuvat graafit eivät yleensä kerro hyökkäysten kohteesta tai niiden riippuvuuksista mitään. Tästä syystä hälytysten visualisointien tutkimiseen on panostettu viimeaikoina. Hälytysten graafiseen analysointiin käytetyt ohjelmat eivät välttämättä sisälly IDS-järjestelmän käyttöliittymään vaan ne voivat olla erillisiä ohjelmistoja, jotka generoivat graafeja IDS-järjestelmien tuottamista lokitiedoista.

Skannaukset ovat yleisimmät ja monimuotoisimmat verkossa tapahtuvat hyökkäykset. Ne perustuvat lähde- ja kohdeosoitteisiin sekä skannattavien porttien numeroihin. Yleisimmät skannaustavat ovat pysty- ja vaakaskannaus sekä ryhmäskannaus. Pystyskannaus koostuu saman laitteen eri porttien tutkimisesta. Pystyskannauksessa hyökkääjä on kiinnostunut tietystä verkon laitteesta ja pyrkii löytämään siitä haavoittuvuuksia. Vaakaskannauksessa hyökkääjä käy läpi useita laitteita tutkien niistä tietyn portin. Tietyn portin tutkiminen viittaa tietyn haavoittuvuuden etsimiseen verkon laitteista. Ryhmäskannaus on edellä mainittujen tapojen kombinaatio. Kuvassa 3.5 on esitetty graafisesti eri skannaustavat. IDS-järjestelmien tulisi pystyä visualisoimaan kaikki skannaustavat niin, että niistä voidaan helposti erottaa mikä tapa on kyseessä. [32]

Usein käytettävä visualisointitapa on histogrammipylväs, johon voidaan helposti sisällyttää erilaisia tapahtumia päällekkäin. Pylväiden koosta erottaa hyvin tapahtumien väliset mittasuhteet. Kolmiulotteisten pylväiden käytöllä voidaan esittää ylimääräisen parametrin vaikutusta, mutta kaavion hahmottaminen vaikeutuu pylväiden erilaisten etäisyyksien ja perspektiivien takia. Graafeissa akselien muuttujiksi valitaan yleensä aika ja pakettien tai bittien määrä. Pylväiden histogrammiksi voidaan valita esimerkiksi tietyt portit tai porttialueet. Kuvassa 3.6 on esitetty histogrammipylväsgraafi porttiskannausten määrästä ajan suhteen.



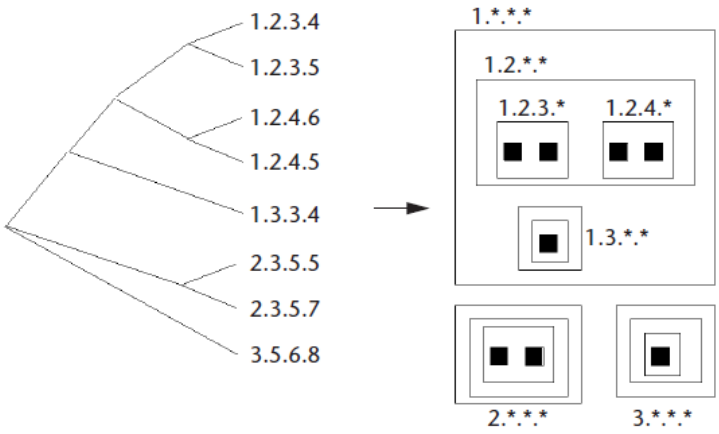
Kuva 3.5: Kolme eri skannaustapaa. [32]



Kuva 3.6: Histogrammi porttiskannauksista jaoteltuna porttien mukaan.

Pylväsgraafin ongelmana on IP-osoitteen puuttuminen. Ajan suhteen luodusta graafista ei voida päätellä kohdistuvatko skannaukset yhteen tai useampaan laitteeseen. Jos taas aika korvataan IP-osoitteella, ei voida nähdä hyökkäysten määrää ajan suhteen, jolloin ei voida päätellä porttiskannausten kiivautta.

Itoh, Takakura, Sawada, ja Koyamada esittivät tutkimuksessaan [33] kolmiulotteisen tavan esittää jopa tuhansiin laitteisiin kohdistuvat hyökkäykset sekä niiden aikaansaamat hyökkäykset. He käyttivät paranneltua suorakulmioiden pakkausalgoritmia (engl. *Rectangle-packing algorithm*), joka piirtää useita suorakulmioita suuremman suorakulmion sisään. Pienimmillä suorakulmioilla esitetään yhtä verkon laitetta ja suuremmilla suorakulmioilla ryhmiä, kuten IP-osoitteiden (aliverkkojen) muodostamia ryhmiä. Suurempi ryhmä sisältää pienempiä ryhmiä, jolloin voidaan esittää koko verkon hierarkia. Kuvassa 3.7 on esitetty aliverkkojen visualisointi suorakulmioiden pakkausalgoritmilla. [33]



Kuva 3.7: Aliverkkojen visualisointi suorakulmioiden pakkausalgoritmilla. [33]

Itohin, Takakuran, Sawadan, ja Koyamadan ehdottama visualisointitekniikka esittää yksittäisen verkon laitteeseen kohdistuvat hyökkäykset ja sen generoimat hyökkäykset eri värillä samassa pylväässä, joka piirretään jokaisen verkon laitteen yhteyteen. Pylvään pituus ilmaisee tapahtumien määrää kyseisessä laitteessa. Pylväiden takia näkymä tulee olla kolmiulotteinen, jotta pylväiden värit ja pituudet on helppo erottaa toisistaan. Pylväiden ja värien avulla voidaan helposti havaita laitteet, joihin kohdistuu paljon hyökkäyksiä. Aliverkkojen visualisoinnin avulla voidaan helposti huomata myös porttiskannaukset, jotka kohdistuvat useaan peräkkäiseen laitteeseen samassa aliverkossa. Hyökkäysten välinen suhde visualisoitiin laitteiden välisten viivojen avulla. Näin voidaan helposti havaita mistä laitteesta hyökkäykset ovat peräisin. Mallin mukaan näkymää voidaan tarkentaa portaattomasti, jolloin päästään näkemään yksityiskohtaisemmin haluttujen laitteiden tapahtumat. [33]

4 IDS-järjestelmien heikkoudet

Kuten aiemmin todettiin IDS-järjestelmät käyttävät erilaisia analysointimenetelmiä hyökkäysten havaitsemiseksi. Nämä menetelmät on yleensä jaettu erillisiin kerroksiin (vrt. protokollapino), joista jokainen tarkastaa tietyt paketin tiedot. Ensimmäiset kerrokset tarkastavat paketin verkko- ja kuljetuskerroksien (engl. *network layer, transport layer*) tiedot, joissa hyökkääjällä on hyvät mahdollisuudet hämätä IDS-järjestelmää. Ylemmät tarkistuskerrokset luottavat alempien kerrosten kykyyn vastaanottaa paketteja samalla tavalla kuin pakettien todellinen kohde. [12]

NIDS-järjestelmien vaikeutena on vastaanottaa paketit juuri samalla tavalla kuin niiden todellinen kohde. Koska TCP- ja IP-protokollat suunniteltiin toimimaan hyvin erilaisissa ympäristöissä dynaamisesti ja niiden määritelmissä on paljon valinnaisuuksia ja vain harvat vaatimukset ovat pakollisia. Tämä määritelmien puute tekee pakettien tarkastamisesta hyvin vaikeaa. Tämä mahdollistaa hyökkääjän hämätä NIDS-järjestelmää generoimalla paketteja, jotka kootaan eri tavalla NIDS-järjestelmässä ja kohdelaitteella. Esimerkiksi, jos NIDS-järjestelmä tarkkailee paketteja etsien niistä tiettyä merkkijonoa, voi hyökkääjä pilkkoa paketit niin, että paketit kasataan eri tavalla NIDS-järjestelmässä kuin kohdelaitteella. Näin NIDS-järjestelmä saa erilaisen viestin kuin kohdekone, eikä reagoi hyökkäykseen. [12]

Esimerkiksi *Nmap*-ohjelmalla voidaan tunnistaa verkon laitteen käyttöjärjestelmä tutkimalla sen TCP/IP-pinon toteutusta. *Nmap* sisältää noin 300 tunnistetta, jotka ilmoittavat eri käyttöjärjestelmien TCP/IP-pinon eroista.

Thomas Ptacek ja Timothy Newsham kirjoittivat vuonna 1998 tutkimuksen NIDS-laitteiden huijaamisesta erilaisten pakettienkäsittelytapojen avulla [26]. Vaikka tutkimuksesta on kulunut kauan, pätevät siinä tehdyt havainnot usein vielä tänäkin päivänä. Ptacekin ja Newshamin mukaan suurin ongelma on verkossa liikkuvan tiedon riittämättömyys tarkkaan paketin analysointiin.

NIDS-laite ja kohdelaite ovat täysin eri järjestelmiä niin toteutustavoiltaan kuin sijainniltaan. Esimerkiksi NIDS-laite, joka sijaitsee verkon solmukohdassa, tutkii paketin, joka on osoitettu kohdelaitteelle, joka sijaitsee usean hypyn päässä solmukohdasta. Kun NIDS-laite on tutkinut paketin, voi alkuperäiselle kohdelaitteelle tapahtua jotakin, joka estää sitä vastaanottamasta pakettia. Tällöin NIDS luulee, että paketti on käsitelty normaalisti kohdelaitteessa ja sen tietämys esimerkiksi TCP-yhteyden tilasta on eri kuin kohdelaitteessa. Vaikka NIDS tietäisikin miten kohde-

laite käsittelee paketin, se ei voi tietää varmasti hyväksyykö kohdelaite pakettia. Kohdelaitteen muisti voi olla täynnä, eikä se hyväksy uusia paketteja. Kohdelaitteen tai verkon resurssit voivat loppua ja pakettia ei käsitellä kohdelaitteessa. [26]

NIDS-järjestelmät tarvitsevat paljon enemmän tietoa kohdelaitteesta ja vallitsevista oloista kuin se pystyy havaitsemaan verkkoliikenteestä ja tutkittavista paketeista.

NIDS-järjestelmät ovat haavoittuvia palvelunestotilanteille, jolloin ne eivät pysty käsittelemään uusia paketteja. Palvelunestohyökkäysten tapahtuessa on tärkeää tietää miten NIDS-järjestelmä toimii jos se lamautetaan hyökkäyksen avulla. Lamaantuessaan tietoturvajärjestelmät voivat joko sallia kaiken liikenteen verkkoon (engl. *Fail-open*) tai kieltää kaiken liikenteen verkkoon (engl. *Fail-closed*). Koska useimmat NIDS-järjestelmät ovat passiivisia, NIDS:n lamaantuminen vastaa fail-open-tilaa, sillä ne eivät vaikuta verkon toimintaan mitenkään. Lamaannuttaessaan NIDS-järjestelmän hyökkääjä voi toimia verkossa rauhassa, kuin IDS-järjestelmää ei olisi. [26]

NIDS-järjestelmät ovat haavoittuvia pääasiassa kahdelle erilaiselle hyökkäykselle: lisäämishyökkäys (engl. *insertion*) ja välttelyhyökkäys (engl. *evasion*), jotka tapahtuvat protokollapinon verkko- tai siirtokerroksessa. HIDS-laitetta voidaan yrittää huijata sovelluskerroksella.

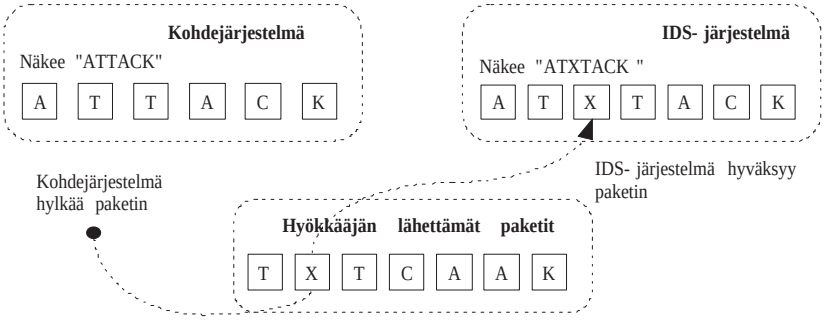
Seuraavissa luvuissa käsitellään näitä hyökkäyksiä, joiden avulla hyökkääjä pystyy huijaamaan NIDS- ja HIDS-järjestelmiä. Hyökkäykset edellyttävät, että hyökkääjä pääsee lisäämään väärennettyjä paketteja datavirtaan. Lisäämistä helpottaa se, ettei hyökkääjän tarvitse kaapata olemassa olevaa yhteyttä vaan oman yhteyden muokkaaminen riittää. [26]

4.1 Lisäämishyökkäys

Lisäämishyökkäyksellä tarkoitetaan tilannetta, jolloin NIDS-järjestelmä hyväksyy paketin olettaen, että kohdejärjestelmä hyväksyy paketin, mutta paketti hylätäänkin kohdejärjestelmässä. Tällöin NIDS:lle on syötetty tietoa, jota kohdejärjestelmä ei havaitse. Tällöin NIDS-järjestelmällä on erilainen käsitys viestin sisällöstä ja se ei huomaa käynnissä olevaa hyökkäystä. Lisäämishyökkäys on tehokkain keino huijata NIDS-järjestelmää. [26] [12]

NIDS-järjestelmien analysointimenetelmät etsivät paketeista tiettyjä merkkijonoja, jotka kuuluvat jonkin tunnetun hyökkäyksen tunnusmerkkeihin. Syöttämällä NIDS-järjestelmälle paketteja, jotka vain se hyväksyy, voidaan muuttaa NIDS:n näkemää viestin sisältöä niin, etteivät analysointimenetelmät havaitse etsittävää merk-

kijonoa. Kuvassa 4.1 esitetään kirjaimen ”X” lisääminen datavirtaan niin, että vain NIDS-järjestelmä näkee sen.



Kuva 4.1: Lisäämishyökkäys kirjaimella ”X”. [26]

Yksittäisten kirjaimien lisääminen datavirtaan ei ole helppoa, mutta heikkouksia voidaan käyttää alemmilla tarkastustasoilla ennen kuin merkkijonoja etsivä analysointimenetelmä saa pakettien sisältämän datan tarkastettavakseen.

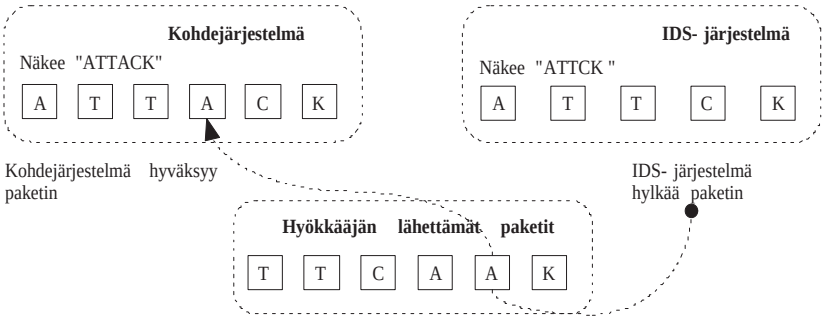
4.2 Välttelyhyökkäys

Välttelyhyökkäyksellä tarkoitetaan hyökkäystä, jossa kohdejärjestelmä hyväksyy paketin, jonka NIDS-järjestelmä hylkää. NIDS menettää koko hylätyn paketin tiedon ja näin sen ohi voidaan livauttaa paketteja, jotka sisältävät tärkeää tietoa hyökkäyksen kannalta. Välttelyhyökkäys on helppo toteuttaa ja sen vaikutus IDS:n tarkkuudelle on hyvin tuhoisa. [26]

Välttelyhyökkäys on vastakohta lisäämishyökkäykselle, mutta sen periaate on sama. Hyökkääjä lisää datavirtaan paketteja, jotka NIDS-järjestelmä hylkää ja kohdejärjestelmä käsittelee. Näin voidaan saada kokonainen TCP-yhteys NIDS:n ohi, ilman että se havaitsee sitä. Kuvassa 4.2 on esitetty välttelyhyökkäys kirjaimella ”X”.

4.3 Heikkoudet verkkokerroksessa

Verkkokerroksen (engl. *Network layer*) analysointimenetelmien heikkoudet voivat heijastua ylempien kerrosten analysointimenetelmiin. Jos verkkokerros hyväksyy hyökkääjän lisäämät väärennetyt IP-paketit, voi hyökkääjä lisätä myös oikein muodostettuja ylemmän protokollapinon paketteja (esimerkiksi UDP tai ICMP). Väärennyskeinoja löytyy useita, mutta yleisimmät tavat ovat väärentää IP-paketin otsikkotietoja tai hyväksikäyttää pilkottujen pakettien uudelleenkasaamisesta syntyviä on-



Kuva 4.2: Välttelyhyökkäys kirjaimella ”X”. [26]

gelmia. [26]

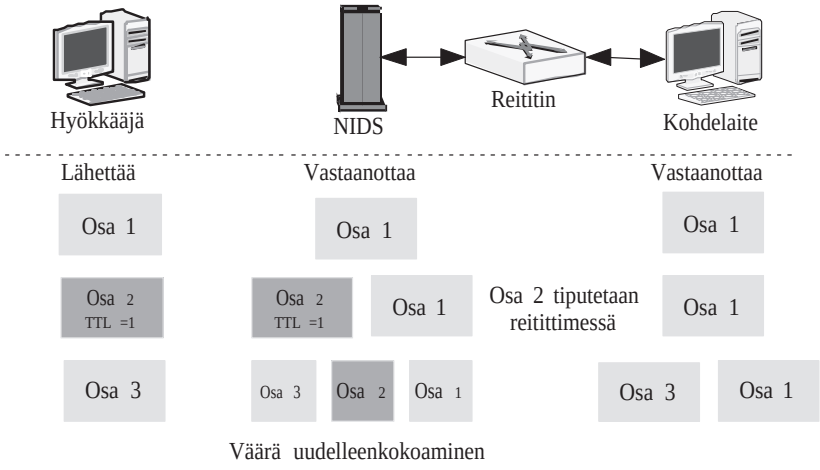
4.3.1 IP-paketin otsikoiden väärentäminen

Verkkokerroksella lisäämis- tai välttelyhyökkäyksen voi toteuttaa väärentämällä IP-paketin otsikkotietoja. Pakettien otsakkeiden väärentäminen on ongelmallista, jos hyökkääjällä ei ole pääsyä samaan verkkoon, jossa IDS-laite on, sillä IP-otsakkeiden väärentäminen voi estää paketin reitityksen verkossa.

IP-paketin versionumeron väärentäminen estää paketin reitityksen verkossa. Joidenkin kenttien arvot tulee olla oikein, jotta paketit voidaan koota hyökkääjän tahtomalla tavalla kohdelaitteissa. Näin esimerkiksi IP-paketin otsakkeen tai koon väärentäminen voi estää ylemmän tason protokollan löytämisen paketista. Eräs helppo väärennettävä on tarkistussumma. Kaikki IDS-järjestelmät eivät välttämättä laske jokaisen paketin tarkistussummaa ja vertaa sitä paketissa olevaan arvoon, mutta lähes kaikki IP-protokollatoteutukset kohdelaitteissa hylkäävät paketin. Näin voidaan saada helposti aikaan lisäyshyökkäys. [26]

Jos NIDS-järjestelmä ei sijaitse kohdelaitteen kanssa samassa verkkosegmentissä, voidaan lähettää paketteja, joiden TTL-arvo asetetaan niin, ettei paketti saavu kohdelaitteelle, mutta NIDS näkee sen. Samantyylinen hyökkäys voidaan toteuttaa IP-paketin *Don't fragment* -lipulla, jos tiedetään, että verkon suurin sallittu pakettikoko on pienempi kohdelaitteen verkossa kuin NIDS:n verkossa. Luomalla paketin, joka on liian iso kohdelaitteen verkkoon, voidaan luoda lisäyshyökkäystilanne. Kuvassa 4.3 on esitetty IP-paketin osien kokoaminen NIDS-laitteessa ja kohdelaitteessa. [26]

Edellä mainitut hyökkäystavat voidaan helposti torjua, jos NIDS-järjestelmällä on käytettävissään tietoa verkon topologiasta. Vaikeammin analysoitavia ovat IP-paketin optiot-kenttä, sillä optioiden käsittelyssä on eroja eri järjestelmien välillä.



Kuva 4.3: Lisäämishyökkäys TTL-arvon avulla. [34]

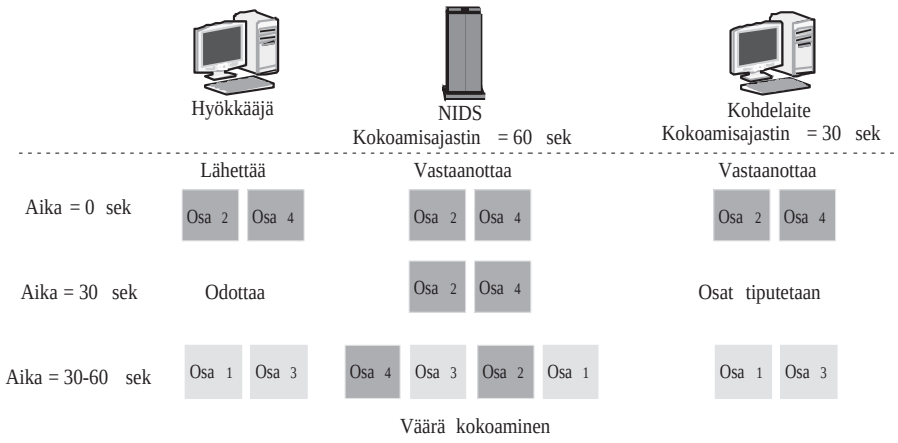
Jos hyökkääjä on samassa verkossa kuin NIDS-laite, voidaan sille lähettää paketteja siirtokerroksella (engl. *link layer*), jotka vain NIDS näkee. Jos NIDS-laitteen MAC-osoite tiedetään, voidaan sitä vastaan tehdä lisäämishyökkäys, sillä muut verkon laitteet eivät käsittele pakettia. Tämä vaatii, ettei IDS-järjestelmä tarkista paketin MAC-osoitteen oikeellisuutta. Vaikka hyökkääjä ei tietäisi NIDS-laitteen MAC-osoitetta, voidaan silloin lähettää paketti, jossa on väärennetty MAC-osoite. Jos NIDS on asetettu *promiscuous*-tilaan, eikä se tarkasta MAC-osoitetta, voidaan sitä vastaan tehdä lisäämishyökkäys. [26]

4.3.2 Uudelleenkokoamishyökkäykset

IP-pakettien pilkkominen ja niiden uudelleenkasaus tulisi tapahtua täysin samalla tavalla sekä NIDS-laitteessa että kohdejärjestelmässä. Yleensä pilkotut IP-paketit saapuvat järjestyksessä, mutta NIDS ei saa automaattisesti olettaa että osat ovat järjestyksessä. Jos NIDS olettaa osien tulevan järjestyksessä, voi hyökkääjä tarkoituksella sekoittaa osien järjestystä ja hämätä NIDS-laitetta. Toinen ongelma on osien tallentaminen siihen asti, kunnes viimeinen osa on saapunut ja paketti voidaan kasata. NIDS-järjestelmän muisti voi kulua loppuun, jos sille lähetetään paketteja osissa, joita ei koskaan lähetetä kokonaan. [26]

Jos NIDS-järjestelmän ja kohdejärjestelmän pakettienkokoamisajastimet (engl. *fragmentation reassembly timer*) eroavat toisistaan, voidaan luoda joko lisäämishyökkäys tai välttelyhyökkäys riippuen kummassa laitteessa ajastin on pienempi. Ajastinta käytetään ilmoittamaan aikaa paketin seuraavan osan odottamiseen. Jos paketin seuraavaa osaa ei ole vastaanotettu ajastimen laskettua nollaan, hylätään vas-

taanotetut pakettien osat. Hyökkääjä voi tarkoituksella viivästyttää paketin osien lähettämistä kunnes toisen laitteen ajastin on umpeutunut. Kuvassa 4.4 on esitetty tilanne, jossa kohdelaitteen ajastin on pienempi kuin NIDS-laitteen ajastin.



Kuva 4.4: Pakettien päällekkäisyyksiä. [34]

Pilkottujen pakettien data voi sisältää päällekkäisyyksiä joko vastaanotetun tai vielä lähettämättä olevan datan kanssa. Eri käyttöjärjestelmät käsittelevät päällekkäisyyksiä eri tavalla. Jos NIDS käsittelee päällekkäistä dataa eri tavalla kuin kohdelaite, voi hyökkääjä käyttää sitä hyväkseen. Vuonna 1997 löydettiin haavoittuvuus monien käyttöjärjestelmien IP-pinon toteutuksesta, jonka takia käyttöjärjestelmät voitiin kaataa lähettämällä niille paketteja, joissa Fragment offset -luvut olivat päällekkäisiä.

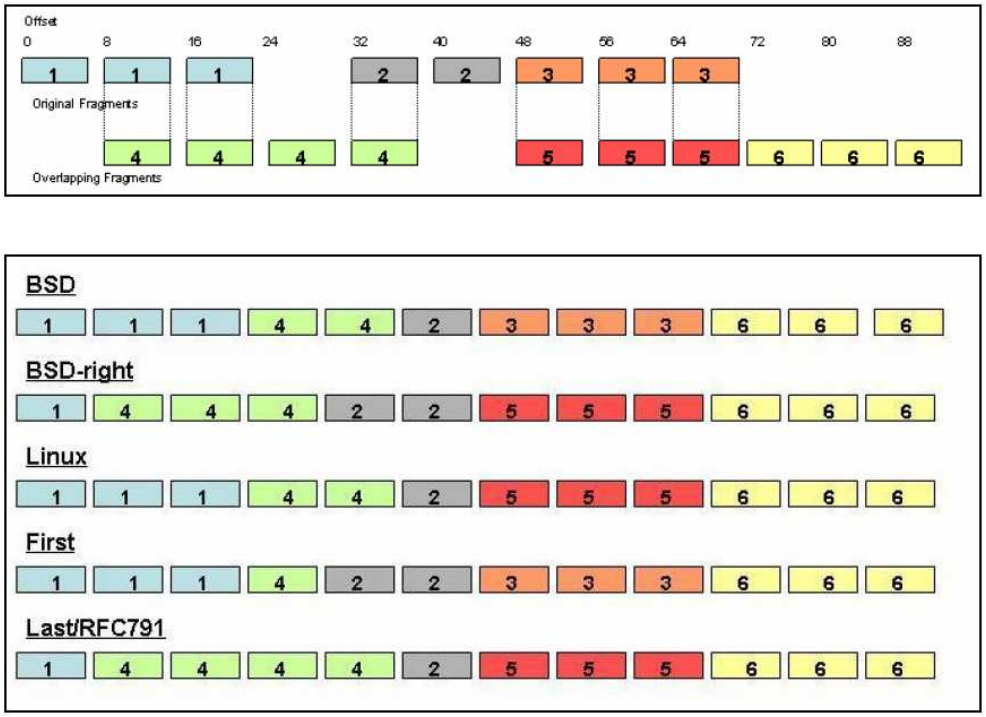
Vern Paxton ja Umesh Shankar tutkivat modernien käyttöjärjestelmien pakettien kasaamistapoja tutkimuksessaan ”Active Mapping: Resisting NIDS Evasion Without Altering Traffic” [35]. He löysivät viisi erilaista tapaa, joilla paketteja kasattiin. Sen pohjalta he kehittivät paradigman, jonka avulla voidaan testata kaikki pakettien osien kokoamistavat kun pakettien data on päällekkäistä. Paradigma sisältää seuraavat fragment-tyypit: [35]

- Ainakin yksi osa (engl. *fragment*) korvataan kokonaan seuraavassa osassa, jolla on sama offset ja pituus.
- Ainakin yksi osa korvataan osittain seuraavassa osassa, jonka offset on suurempi kuin alkuperäisessä.
- Ainakin yksi osa korvataan osittain seuraavassa osassa, jonka offset on pienempi kuin alkuperäinen.

Näiden perusteella he loivat mallin, joka esittää viittä erilaista pakettienkasaamistapaa: [35]

- **BSD** suosii alkuperäistä osaa, jonka offset on pienempi tai yhtä suuri kuin korvaavan osan.
- **BSD-right** (suom. *BSD-oikea*) suosii seuraavaa osaa, kun alkuperäisen osan offset on pienempi tai yhtä suuri kuin korvaavan osan.
- **Linux** suosii alkuperäistä osaa, jonka offset on pienempi kuin korvaavan osan.
- **First** (suom. *Ensimmäinen*) suosii aina alkuperäistä osaa.
- **Last** (suom. *Viimeinen*) suosii aina korvaavaa osaa.

Kuvassa 4.5 on esitetty Paxson/Shankar uudelleenkokoamistavat graafisesti.



Kuva 4.5: Paxson/Shankar uudelleenkokoamistavat. [36]

Taulukossa 4.1 on esitetty Paxsonin ja Shankarin testaamat käyttöjärjestelmät jaoteltuna uudelleenkokoamistavan mukaan.

Pakettien erilaisesta kokoamisesta aiheutuu ongelmia NIDS-järjestelmille. NIDS-järjestelmän tulee tietää millaista kasaustapaa kohdejärjestelmä käyttää tai muuten

hyökkääjät voivat käyttää järjestelmien erilaisuutta hyväkseen toteuttaessaan hyökkäyksiä. Yleensä NIDS-järjestelmille voidaan kertoa kohdejärjestelmän pakettienkasaustavan, jolloin se osaa suojata kohdetta tämäntyyppisiltä hyökkäyksiltä. Esimerkiksi *Snort* sisältää *Frag3*-moduulin, jolle voidaan asettaa jokaisen kohdelaitteen käyttämä kokoamistapa.

Taulukko 4.1: Käyttöjärjestelmät jaoteltuna Paxson/Shankar-mallin mukaan.

Tapa	**Alustat**
BSD-right	HP JetDirect
BSD	AIX 2, 4.3, 8.9.3, FreeBSD, HP-UX B.10.20, IRIX 4.0.5F, 6.2, 6.3, 6.4, NCD Thin Clients, OpenBSD, OpenVMS, OS/2, OSF1, SunOS 4.1.4, Tru64 Unix V5.0A,V5.1, Vax/VMS
Linux	Linux 2.x
First	HP-UX 11.00, MacOS (version unknown), SunOS 5.5.1, 5.6, 5.7, 5.8, Windows (95/98/NT4/ME/W2K/XP/2003)
Last	Cisco IOS

4.4 Heikkoudet siirtokerroksessa

Vaikka NIDS-järjestelmä olisi immuuni IP-protokollan avulla tapahtuville hyökkäyksille, pitää sen kyetä käsittelemään myös TCP-paketit oikein. TCP-paketteja voidaan väärentää samalla tavalla kuin IP-paketteja, eikä NIDS saa olettaa laillisten IP-pakettien sisältävän laillisia TCP-paketteja.

Yleisimpiä TCP-protokollan avulla tapahtuvia hyökkäysyrityksiä NIDS-järjestelmiä vastaan ovat TCP-otsakkeiden väärentäminen ja NIDS:n saaminen epätahdistettuun tilaan, jolloin se ei kykene valvomaan TCP-yhteyttä.

4.4.1 TCP-otsakkeiden väärentäminen

Monet TCP-pinon toteutukset käsittelevät paketin datan ainoastaan, jos paketin ACK-lippu (*Acknowledged*) on asetettu. TCP-protokollan määritelmä kuitenkin määrää, että järjestelmien on käsiteltävä myös SYN-lipulla varustettujen pakettien data. Jos NIDS-laiteen ja kohdelaitteen käsittelytapa eroaa toisistaan, altistuu järjestelmä lisäys- tai välttelyhyökkäykselle. Kuten IP-pakettien kohdalla, jotkin NIDS-järjestelmät jättävät pakettien tarkistussumman laskematta, sama ongelma esiintyy myös TCP-pakettien yhteydessä. [26]

TCP-paketin optioiden oikeellisuuden tarkastaminen on huomattavasti hankalampaa kuin IP-paketin optioiden tarkastaminen, koska TCP asettaa voimassaoloehtoja tietyille optioille eri yhteystiloissa. TCP-paketissa on myös varattuja optioita, joita ei tällä hetkellä käytetä yleisesti. Jos optioita otetaan käyttöön, täytyy NIDS-järjestelmien osata käsitellä niitä.

4.4.2 TCP:n tahdistus

Suurin osa hyökkäyksistä tehdään TCP-protokollan avulla. Tämä vaatii, että NIDS-järjestelmä pystyy rekonstruoimaan TCP-yhteyden, jotta se tietää yhteyden tilan ja siinä sallittavat viestit. Jos NIDS-järjestelmä ei kykene tähän, on se altis hyökkäyksille. Rekonstruointiin käytetään esimerkiksi pakettien vuoronumerointia. Seuraavan validin vuoronumeron päättelyä kutsutaan tahdistukseksi (engl. *synchronization*). Jos NIDS-laite ei pysty päättelemään seuraavaa validia vuoronumeroa, on se epätahdistetussa tilassa (engl. *desynchronization*). Kun NIDS on epätahdistettu yhteydestä, se ei kykene tarkasti rekonstruoimaan yhteyttä. Tällaisissa tapauksissa NIDS ei yleensä havaitse mitään yhteydessä kulkevaa dataa, koska sen käsitys seuraavasta validista vuoronumerosta on väärä. Hyökkääjien suurimpia tavoitteita on saada NIDS-järjestelmä epäsynkronoituun tilaan. [26]

Jokaista luotua TCP-yhteyttä varten NIDS-järjestelmä luo TCB-objektin (engl. *TCP Control Block, TCB [26]*), joka sisältää tiedot TCP-yhteyden tilasta. Ptacek ja Newsham määrittelivät kolme tapahtumaa, joissa yhteyden käsittelyä voidaan häiritä:

- TCB:n luominen. Tilanne, jossa IDS tekee päätöksen uuden TCB-objektin luomisesta havaitulle yhteydelle.
- Datavuon kokoaminen. Prosessi, jota IDS käyttää datavuon rekonstruointiin.
- TCB:n hävittäminen. Tilanne, jossa IDS päättää TCB-objektin tuhoamisesta yhteyden katkettua.

TCB:n luominen tulisi tapahtua aina, kun uusi yhteys muodostuu NIDS:n valvomaan verkkoon sekä sellaisille yhteyksille, jotka olivat jo käynnissä ennen NIDS-järjestelmän käynnistämistä. Jos NIDS-järjestelmä ei kykene tarkastamaan käynnistyksen yhteydessä olemassa olevia yhteyksiä, voidaan yhteys piilottaa NIDS:ltä jos hyökkääjä kykenee uudelleenkäynnistämään NIDS-järjestelmän esimerkiksi palvelunestohyökkäyksellä. TCB:n luomiseen voidaan käyttää erilaisia tapoja, mutta jokaisella tavalla on omat vahvuutensa sekä heikkoutensa.

TCB:n luominen voidaan liittää TCP-protokollan kolmitie-kättelyyn. Kun kättely on suoritettu ja yhteyden osapuolet ovat sopineet yhteisistä parametreista, TCP-yhteys on luotu ja NIDS-järjestelmä voi alkaa seuraamaan yhteyttä. [26]

Kättelyn seuraamiseen liittyy kuitenkin muutamia ongelmia, jotka voivat estää NIDS-järjestelmää havaitsemasta uutta yhteyttä. Jos NIDS luottaa täysin kolmitie-kättelyyn, eikä se havaitse kolmitiekättelyä, jää koko yhteys havaitsematta. Tämä on ongelma varsinkin NIDS:n käynnistyessä, jolloin se ei näe yhteyksiä, jotka on luotu ennen käynnistymistä. Suurempi ongelma muodostuu, kun hyökkääjä kykenee piilottamaan kolmitiekättelyn NIDS-laitteelta, eikä NIDS havaitse muodostettua yhteyttä. Väärentämällä kolmitiekättelyn vuoronumerot, hyökkääjä voi saada NIDS-järjestelmän epäsynkronoitua, jos se käyttää vuoronumeroita seuraavien datapakettien tarkastamiseen. Tämä korostuu optioiden tarkastamisessa, sillä NIDS:n on nähtävä kolmitiekättely, jotta tiettyjä optioita voidaan käsitellä oikein yhteyden aikana. Tällainen on esimerkiksi PAWS-optio (*Protection Against Wrapped Sequence numbers, RFC1323*), joka hylkää vanhoja duplikaattipaketteja, jotka voivat korruptoida olemassa olevan TCP-yhteyden. [26]

NIDS-järjestelmä voi luoda TCB:n havaittuaan vain osan kolmitiekättelystä. TCB voidaan luoda joko asiakkaan SYN-paketista tai palvelimen SYN+ACK-vastauksesta. Jos verkossa on tilallinen palomuuri, joka estää ulkoapäin saapuvien väärennettyjen pakettien saapumisen verkkoon, voi NIDS luottaa SYN+ACK-pakettiin ja päätellä siitä yhteyden olevan muodostumassa. SYN+ACK-paketti sisältää myös oikeat yhteyden vuoronumerot. Vasta kun asiakas vastaa palvelimelle SYN+ACK-viestillä, voi NIDS olla varma, ettei yhteyttä yritetä väärentää. Tätä ennen yhteys ei ole voimassa, joten TCB:tä ei voida luoda. [26]

TCB voidaan luoda myös päättelemällä yhteyden tila datapaketeista, joita lähetetään yhteyden muodostamisen jälkeen. NIDS voi tarkkailla vain verkossa liikkuvia ACK-paketteja, eikä sen tarvitse välittää kättelypaketeista. Menetelmät etuina on, että järjestelmä ei ole riippuvainen kättelyn havaitsemisesta ja se huomaa ennen järjestelmän käynnistystä aloitetut yhteydet. Haittapuolena on mahdollisuus, että NIDS hyväksyy paketin, joka ei kuulu mihinkään voimassaolevaan yhteyteen. [26]

Yleensä TCB:n luomisessa käytetään edellä mainittujen tapojen yhdistelmiä, joilla pyritään havaitsemaan kaikki yhteydet ja samalla minimoimaan eri tapojen heikkoudet. [12]

Datavuon kokoaminen. TCP-yhteyden osapuolet pystyvät pyytämään datan uudelleenlähetystä, jos ne jostain syystä eivät vastaanota toisen osapuolen lähettämää dataa. NIDS-järjestelmät eivät tätä pysty tekemään ja siitä syystä pakettien hukkumiset voivat olla niille kohtalokkaita. Tarpeeksi monen paketin hukkuminen

haittaa yhteyden rekonstruointia ja voi johtaa NIDS-laitteen epäsynkronoituun tilaan, josta sen pitää toipua jollakin tavalla.

TCP-yhteyksissä käytetään ikkunan kokoa ilmoittamaan vastaanottajan kykyä vastaanottaa dataa yhteyden aikana. Ikkunan kokoa kasvatetaan tai pienennetään tarvittaessa yhteyden aikana, riippuen vastaanottajan kyvystä käsitellä ikkunan koon verran dataa kerrallaan. Vastaanottaja hylkää paketit, jotka ylittävät sovitun ikkunan koon. Aika, jossa NIDS-järjestelmä huomaa ikkunan koon muutoksen ja kohdejärjestelmä reagoi siihen, on eripituinen. Tässä ajassa saapuvat paketit voivat saada erilaisen käsittelyn kohdelaitteessa ja NIDS-laitteessa ikkunoiden erilaisten kokojen takia. NIDS-järjestelmän tulee ottaa tämä aika huomioon tai se on alttiina lisäyshyökkäykselle kyseisen ajan. [26]

TCP-toteutukset eroavat toisistaan päällekkäisen datan kokoamisen suhteen samalla tavalla kuin IP-pakettien kanssa. NIDS-järjestelmän tulee tietää kohdelaitteen kokoamistapa, jotta se voi ratkaista päällekkäisten pakettien kohtalon samalla tavalla kuin kohdelaite.

TCB:n hävittäminen. NIDS-järjestelmän tulee määrittää selkeä vaihe, jolloin TCP-yhteys on katkaistu ja TCB-objekti voidaan tuhota. TCB tuhotaan, sillä se kuluttaa järjestelmän resursseja ja turhien TCB-objektien ylläpitäminen ei ole järkevää. NIDS, joka ei tuhoa TCB-objekteja, on altis palvelunestohyökkäykselle, kun yhteyksien määrä ylittää järjestelmän kyvyn käsitellä TCB-objekteja tai niiden tiedoille varattu muistitila loppuu. [26]

TCB:n tuhoamiseen liittyy muutamia ongelmia, jotka NIDS-järjestelmät täytyy ottaa huomioon. TCP-yhteys katkaistaan kun laite vastaanottaa RST/FIN-paketin, mutta yhteys voi katketa milloin tahansa, jopa ilman ennakkovaroitusta. Eräät järjestelmät katkaisevat yhteyden, vaikka RST-paketti ei olisi oikein vuoronumeroitu. Monelle järjestelmälle ICMP-protokollan määränpää saavuttamattomissa -viesti (engl. *Destination unreachable*) on merkki yhteyden lopettamisesta samalla tavalla kuin FIN-paketti. [12]

TCP-yhteyksiin ei välttämättä liity yhteyden katkaisuun tarkoitettua ajastinta, vaan yhteys voi olla auki ikuisesti ilman pakettien vaihtoa osapuolien välillä. TCP tarjoaa tavan varmistaa kummankin osapuolen olevan toiminnassa välittämällä viestejä osapuolten välillä, mutta tämä ei ole yleisesti käytössä. NIDS:n tulee havaita käyttämättömät yhteydet ja katkaista ne tietyn ajan kuluttua, sillä muuten IDS on haavoittuva palvelunestohyökkäykselle, jossa avataan uusia yhteyksiä niin kauan, että IDS ei pysty käsittelemään niitä. Ongelmana on, että NIDS voidaan huijata hävittämään TCB:n, joka kuuluu vielä aktiiviselle yhteydelle ja NIDS menettää tiedon yhteyden tilasta. [26]

NIDS:n tulee tuhota TCB, jos yhteys on todella katkaistu, sillä jos näin ei tapahdu, voi hyökkääjä luoda uuden yhteyden samoilla parametreilla, mutta eri vuoronumeroilla ja tämä voi jäädä NIDS-järjestelmältä huomaamatta ja se joutuu epäsynkronoituun tilaan.

4.5 Heikkoudet sovelluskerroksessa

Kuten verkko- ja siirtokerroksissa, myös sovelluskerroksessa (engl. *application layer*) esiintyy heikkouksia, joiden avulla IDS-järjestelmiä voidaan ohittaa.

4.5.1 Merkistökoodaukset

Eräs ongelma on merkistöjen koodaukset. Tavallinen ASCII-merkistö ei pysty esittämään kuin murto-osan kansainvälisistä merkeistä, joita käytetään eri kielissä. Jotta jokainen merkki voitaisiin esittää ja siirtää oikein, käytetään koodausta muuntamaan erilaiset merkit ASCII-merkeiksi siirtoa varten.

Unicode-merkistö kehitettiin ilmaisemaan tarvittavia merkkejä, joita esiintyy kaikissa maailman kielissä. UTF-8 on Unicode-merkistöjen koodaukseen käytettävä merkistö, jonka avulla tietty Unicode-merkki voidaan ilmoittaa 2-4 tavun pituisella koodilla. UTF-8 suunniteltiin yhteensopivaksi ASCII-koodiston kanssa, joten UTF-8:n ensimmäiset yhdellä tavulla (8 bittiä) esittämät merkit ovat samoja kuin ASCII-merkistössä. UTF-8 voi sisältää useita viitteitä samaan Unicode-merkkiin eri koodien avulla, myös viittaukset toisiin UTF-8-koodeihin on mahdollista. Aina kun UTF-8-merkistöä laajennetaan tavulla, se kartoittaa uudelleen (engl. *Re-mapped*) edelliset viittaukset Unicode-merkistöön, jolloin eri koodeilla viitataan samaan merkkiin. Esimerkiksi kenoviiva-merkki "\" esitetään heksadesimaaleina `5C`, `C19C` ja `E0819C`, riippuen käytettyjen tavujen määrästä. Jokainen heksadesimaalikoodi viittaa samaan Unicode-merkkiin. Nykyään useat viitteet samaan merkkiin ovat kielletty Unicode-standardissa, mutta niitä voidaan vieläkin käsitellä väärin. [37]

Unicode-merkistöä voitiin käyttää esimerkiksi Microsoftin Internet Information Server -palvelimen (*IIS*) huijaamiseen. IIS tarkasti kysytyn tiedoston polun ennen UTF-8-enkoodausta, joten sitä voitiin huijata sijoittamalla tiedoston polkuun UTF-8-koodattuja merkkejä, joita ISS ei osannut tulkita oikein. Esimerkiksi tiedostopolkua `http://victim/../../winnt/system32/cmd.exe` ei saa suorittaa verkosta käsin, mutta kun "`../..`" -osa määriteltiin seuraavasti "`..%C1%9C..`", ei IIS havainnut sitä ja antoi suorittaa komentokehotteen. [37]

Useat IDS-laitteiden valmistajat julkaisivat sääntöjä, joilla pyrittiin havaitsemaan

paketeissa olleita Unicodeväärinkäytöksiä, esimerkiksi tarkkailemalla merkkijonoa ”..%C1%9C..”. Esimerkkinä Snortin sääntö: [37]

```
alert tcp !$HOME_NET any -> $HOME_NET 80 (
  msg: "IDS434 - WEB IIS - Unicode traversal backslash";
  flags: AP; content: "..|25|c1|25|9c"; nocase; )
```

Nämä toimenpiteet eivät kuitenkaan estäneet haavoittuvuuden käyttöä, sillä hyökkääjät käyttivät muita UTF-8-koodeja ja näin IDS-laitteet olivat voimattomia. Ainoastaan NetworkICE-yhtiö oli kehittämässä toimivaa ratkaisua ongelmaan jo ennen kuin se tuli julkisuuteen. He lisäsivät BlackICE-ohjelmaansa toimivan Unicode-parserin, joka muunsi merkit ASCII-merkistöön ja tämän jälkeen voitiin soveltaa normaaleja sääntöjä haavoittuvuuksien etsintään. [37]

Nykyään tämä ei ole ongelma IDS-järjestelmille, sillä useimmat IDS-järjestelmät sisältävät Unicode-parserin, mutta ongelmaksi muodostuu kuitenkin epästandardit ohjelmistot, jotka ovat usein kaupallisia, eikä niiden lähdekoodia ja toteutusta voida tarkastella. Tällöin ei voida olla täysin varmoja miten esimerkiksi merkistöjen koodaus niissä käsitellään. Esimerkkinä on IIS, joka hyväksyy unicode-koodauksen normaalin muodon (`%##`, jossa `##` on heksadesimaalinumero) lisäksi myös epästandardin muodon `%u##`. IDS-järjestelmien tulee osata purkaa myös epästandardin mukaiset koodaukset, jotta niiden analysoijat voivat tutkia niitä. [12]

Esimerkiksi muunnettaessa UTF-8 koodattu merkkijono `%2D` ASCII merkeiksi, saadaan merkki ”-”. Jos IDS etsii HTTP-request-otsakkeesta merkkijonoa:

```
ATTACK=exploit-code
```

Voi hyökkääjä koodata tämän merkkijonon muotoon:

```
ATTACK=exploit%u2Dcode
```

Tällöin olisi mahdollisuus, ettei IDS osaa muuntaa merkkijonoa `%u2D` ”-”-merkiksi ja näin se ei huomaisi hyökkäystä, mutta IIS osaisi muuntaa merkin oikein ja mahdollistaisi väärinkäytön.

4.5.2 Tiedostopolku

Eräs IDS-järjestelmille ongelmia aiheuttava tekijä voi olla tiedostopolun määrittäminen. Hyökkääjä voi tarkoituksella muodostaa hankalasti tulkittavia tiedostopolkuja, joiden tulkitseminen tuottaa hankaluuksia IDS-järjestelmille. Yleensä tiedostojen

poluissa piste tarkoittaa nykyistä hakemistoa ja kaksi pistettä tarkoittaa edellistä hakemistoa. Myös hakemistot erottava "/"-merkki voidaan esittää kahdella kenoviivalla (//). Näin voidaan yrittää sekoittaa IDS-järjestelmän analysaattori, joka yrittää etsiä esimerkiksi HTTP request -viesteistä kiellettyjä merkkijonoja. Kaikki seuraavat HTTP-GET-viestit osoittavat samaan tiedostoon: [12] [38]

```
GET /some/file.cgi HTTP/1.0
GET /.././some////file.cgi HTTP/1.0
GET /./some//..\..///some/./file.cgi HTTP/1.0
```

Jos IDS-järjestelmä ei kykene muuntamaan polkuja yksinkertaisimpaan muotoonsa ennen kuin niitä analysoidaan, voi hyökkääjän viesti päästä IDS:n ohi ilman hälytystä.

4.5.3 HTTP-otsikot

IDS-järjestelmät voivat lopettaa HTTP-GET-viestin tutkimisen, jos ne kohtaavat kysymysmerkin ("?"). Tämä johtuu siitä, että WWW-osoitteisiin liitetään usein sivuston käyttämiä muuttujia, joita IDS-järjestelmän ei tarvitse tarkastaa. Vaarana on, että hyökkääjä voi hämätä merkkijonoja tutkivaa analysoijaa lisäämällä merkkijonon `%3f`, joka on kysymysmerkki unicode koodattuna, HTTP-GET-viestiin. Seuraavat polut osoittavat taas samaan tiedostoon, mutta toinen voi jäädä IDS-järjestelmältä huomaamatta kysymysmerkin takia: [12] [38]

```
GET /real.file HTTP/1.0
GET /%3f/some/../../real.file HTTP/1.0
```

IDS-järjestelmien merkkijonojen etsimiseen perustuvien analysoijien säännöissä ei saa koskaan olla HTTP-viestin metodia (engl. *method*) esimerkiksi GET, HEAD tai POST, sillä joitakin CGI-ohjelmissa olevia aukkoja voidaan käyttää hyväksi muillakin metodeilla kuin HTTP-GET:llä. [38]

4.6 Vastatoimenpiteet

NIDS-järjestelmien verkko- ja siirtokerrosten heikkouksien poistamiseen on esitetty useita erilaisia ratkaisuja. Yksinkertaisimpia tapoja on korvata NIDS-järjestelmät HIDS-järjestelmillä. Niitä ei pysty huijaamaan pakettien kokoamisesta aiheutuvilla eroilla, sillä ne ovat asennettuna kohdelaitteeseen protokollapinon yläpuolelle ja

näin ne näkevät paketit juuri siten kuin kohdelaite on ne koonnut. Suurissa verkoissa HIDS-järjestelmien asentaminen ja ylläpito muodostavat suuren ongelman. Siinä missä muutama NIDS-laite voi valvoa koko verkon, voidaan HIDS-järjestelmiä tarvita useita satoja. Jotta NIDS kokoaisi paketit samalla tavalla kuin kohdelaite, tulisi sen käytössä olla tietokanta kohdelaitteiden protokollapinon toteutuksesta ja verkon topologiasta. NIDS-laite voisi itse tutkia verkkoa ja päätellä sen topologian sekä eri laitteiden käyttöjärjestelmät. Tämän mallin ongelmana on, että NIDS tarvitsisi erillisen kokoamismallin jokaiselle löytämälleen järjestelmälle. Jos NIDS-laitteella on muutama vaihtoehto siitä, miten kohdelaite kokoaa paketit, voi se eriyttää tutkimisen useaan eri osaan, joista jokaisessa paketit kootaan eri tavalla. Tätä kutsutaan haarautuvaksi analyysiksi (engl. *bifurcating analysis*). Jos haaroja tarvitsee vielä haaroittaa lisää, syntyy nopeasti tilanne, jossa haarojen lukumäärä kastaa eksponentiaalisesti ja NIDS-järjestelmän kapasiteetti loppuu. [39]

4.6.1 Normalisoija

Eräänä vastatoimenpiteenä Mark Handley ja Vern Paxson esittävät artikkelissaan [39] erityisen liikenteen normalisoijan (engl. *normalizer*) käyttöä. Normalisoija olisi laite, joka sijoitetaan suojattavan verkon ja yleisen verkon liitoskohtaan ja se ”tasoittaisi” verkkoon saapuvaa liikennettä purkamalla paketit ja luomalla ne uudestaan kohdeverkkoon. Normalisoija poistaisi kaikki epäselvät paketit datavuosta, kuten paketit, jotka sisältävät päällekkäistä dataa. Normalisoija poistaisi monta ongelmaa, jotka johtuvat pakettien erilaisista kokoamistavoista. Koska verkkoon saapuvat paketit eivät voisi sisältää päällekkäistä dataa, näkisivät NIDS ja kohdelaite ne samalla tavalla.

Verkon ylläpitäjät voivat testata IDS-järjestelmäänsä Dug Songin kirjoittamalla *fragrouter*-ohjelmalla[1]. *Fragrouter* generoi erilaisia hyökkäyskuvioita, jotka esimerkiksi sisältävät paketteja, joiden data on osittain päällekkäistä. *Fragrouter* sisältää kaikki IDS-järjestelmien huijaustavat, jotka on esitetty Thomas Ptacekin ja Timothy Newshamin tutkielmassa [26].

4.6.2 Hunajapurkit

Viime aikoina on alettu käyttämään erityisiä ”hunajapurkkeja” (engl. *honeypot*), joiden tehtävänä on kääntää hyökkääjien mielenkiintoa tärkeistä laitteista itseensä ja samalla tallentaa kaiken hyökkäykseen liittyvän datan. Hunajapurkit ovat ohjelmia,

[1] `http://www.monkey.org/~dugsong/fragrouter-1.6.tar.gz`

jotka luovat kokonaisia virtuaalisia aliverkkoja yhdelle ainoalle laitteelle ja analysoivat luomiinsa osoitteisiin tulevaa dataa. Ohjelmat emuloivat erilaisia palveluita luomissaan osoitteissa ja niiden toimintaa voidaan konfiguroida joustavasti. Esimerkiksi *honeyd*[2] ohjelmassa voidaan konfiguroida jokainen IP-osoite erikseen ja niille voidaan määrittää käyttöjärjestelmä, jonka ohjelma valehtelee nmap:lla tutkittaessa. [40]

Ote *honeyd*-ohjelman konfiguraatiotiedostosta, jossa määritellään yhden virtuaalisen laitteen asetukset. [40]

```
create template
set template personality "AIX 4.0 - 4.2"
add template tcp port 80 "sh scripts/web.sh"
add template tcp port 22 "sh scripts/test.sh $ipsrc $dport"
add template tcp port 23 proxy 10.23.1.2:23
set template default tcp action reset
bind 10.21.19.102 template
```

”Hunajapurkit” luovat virtuaalisen verkon käyttämättömään osoiteavaruuteen ja tarkkailevat niille tulevaa liikennettä. Koska virtuaalisia laitteita voi olla jopa 65000, voidaan niiden avulla kerätä informaatiota hyökkäyksestä ja sen mahdollisesta aiheuttajasta. Koska virtuaalilaitteet eivät ole tuotantokäytössä, kaikki niihin tuleva liikenne on periaatteessa peräisin hyökkääjiltä ja haittaohjelmilta. Kerätyistä tiedoista nähdään helposti porttiskannaus- tai matoepidemiat, joita tavallinen NIDS ei välttämättä havaitse. ”Hunajapurkkeja” on esitetty liitettäväksi IDS-järjestelmiin erillisinä moduuleina, jolloin niiden tietoja voitaisiin käyttää IDS-järjestelmissä. [40]

Yleensä ”hunajapurkit” konfiguroidaan niin, että ne houkuttelevat hyökkääjiä puoleensa. Toinen tapa on tehdä virtuaalilaitteet yhtä hankalapääsyisiksi kuin muutkin verkon laitteet ja näin voidaan etsiä verkon tietoturvan heikkouksia. Jos ”hunajapurkit” havaitsevat hyökkäyksen, voidaan niiden lokeja tutkimalla nähdä verkossa olevat heikkoudet. [12]

Vinod Yegneswaran, Paul Barford ja Vern Paxson analysoivat kuuden kuukauden aikana kerättyä dataa ”hunajaverkosta” tutkimuksessaan[41] ja havaitsivat, että erilaiset porttitutkimiset, esimerkiksi väärin konfiguroitu eDonkey, Windows-haavoittuvuuden automaattinen etsintä sekä Code-Red-mato, muodostavat erilaisia kuvaajia lähdeosoitteiden ja ajan suhteen. Näistä tilastoista voidaan päätellä hyökkäyksen aiheuttaja sekä tunnistaa esimerkiksi matoepidemia, ennen kuin NIDS-laitteiden valmistajat kehittävät tunnisteen madolle. [41]

[2] http://www.honeyd.org/

5 IDS-järjestelmiä

Luvussa kuvaillaan yleisimpien IDS-järjestelmien ominaisuuksia ja toimintoja. IDS-järjestelmät jakautuvat kaupallisiin ja vapaisiin järjestelmiin. Kaupalliset järjestelmät ovat maksullisia ja ne sisältävät valmistajan räätälöimät laitteet ja ohjelmistot, jotka on tarkoitettu puhtaasti IDS-järjestelmiin. Kaupallisiin järjestelmiin kuuluu yleensä myös kattavat tukipalvelut ja manuaalit. Vapaat järjestelmät (engl. *Open Source*) sisältävät yleensä vain ohjelmiston, joka asennetaan asiakkaan itse hankkimaan laitteistoon. Laitteistot ovat pääasiassa PC-laitteita, joissa on jokin Unix-pohjainen käyttöjärjestelmä (esimerkiksi Linux ja OpenBSD). Vapaiden ohjelmistojen etuna on kustannustehokkuus, muunneltavuus sekä laaja kehittäjien ja käyttäjien verkosto, jonka avulla uusiin tietoturvariskeihin voidaan reagoida nopeasti.

Suosituimpia IDS-järjestelmiä ovat kaupallinen Ciscon IPS 4200 -sarja, joka sisältää useita eritehoisia sensoreita erinopeuksisia verkkoja varten, sekä vapaan lähdekoodin Snort[1], joka tarjoaa modulaarisuutensa ansiosta suuren muunneltavuuden eri käyttöympäristöjä varten. Snortista on myös olemassa kaupallinen versio, Sourcefire[2], joka tarjoaa Snortin tehokkuuden valmiissa sensorilaitteissa. Toinen tunnettu vapaan lähdekoodin IDS-järjestelmä on Bro[3].

Kaupallisten palomuurien ominaisuudet ovat lähentyneet tunkeutumisenhavaitsemisjärjestelmiä, sillä useimmat palomuurit pystyvät tutkimaan paketteja myös ylemmillä protokollapinon tasoilla ja etsimään niistä merkkejä hyökkäyksistä. Esimerkiksi Checkpoint FireWall-1 sisältää SmartDefence-ominaisuuden sekä muita sovellustason-ominaisuuksia, joiden avulla se pystyy suojaamaan järjestelmiä erilaisia verkko- sekä sovellustason hyökkäyksiä vastaan [42]. SmartDefencen käyttämät säännöt voidaan päivittää verkosta käsin, joten se muistuttaa ominaisuuksiltaan paljon väärinkäytösten analysointiin käytettyä IDS-järjestelmää.

5.1 Snort

Snort on vapaan lähdekoodin IDS-järjestelmä, jonka päätoiminnot ovat pakettien haistelu (engl. *Packet sniffer*), pakettien tietojen tallentaminen lokiin (engl. *Packet log-*

[1] http://www.snort.org
[2] http://www.sourcefire.com
[3] http://bro-ids.org/

ger) sekä verkkopohjainen tunkeutumisenhavaitseminen (NIDS). Snortiin on saatavilla useita lisäosia, jotka laajentavat ohjelman toiminnallisuutta. Lisäosia ovat esimerkiksi erilaiset lokitietojen käsittelijät, Snortin hyökkäysmallien (engl. *Signature, Ruleset*) päivittäjät sekä hälyttäjät, jotka ilmoittavat verkon ylläpitäjälle Snortin havaitsemista hyökkäyksistä. Lisäosien avulla Snort voi ymmärtää muitakin protokollia kuin TCP/IP:tä, kuten esimerkiksi Novellin IPX-protokollaa. Snortin etuja on myös avoin ja laaja käyttäjäyhteisö, joka julkaisee ohjeita, tiedotteita ja uusia hyökkäysmalleja verkossa ositteessa `http://www.snort.org` [18]

Koska Snort käsittää vain tunkeutumisen havaitsemiseen tarvittavan ohjelmiston, tulee verkon ylläpitäjän hankkia sopiva tietokone, johon Snort voidaan asentaa. Laitteiston tehokkuus riippuu pitkälti verkon liikenteen määrästä. Nopeissa verkoissa, joissa liikkuu runsaasti liikennettä, pitää laitteessa olla nopea prosessori ja paljon muistia, jotta Snort kykenee reaaliaikaiseen analysointiin. Laitteen tehokkuus tulisi valita verkon ruuhkahuippujen mukaan, jolloin hetkelliset piikit verkon liikenteessä eivät hidastaisi tarkastusta. Laitteen tallennustilan määrä tulee valita niin, että kaikki halutut hälytykset voidaan tallentaa levylle. Samalla on mietittävä, kuinka pitkän aikaa hälytysten tulee säilyä ja tallennetaanko myös hälytyksiin liittyvät paketit. [18]

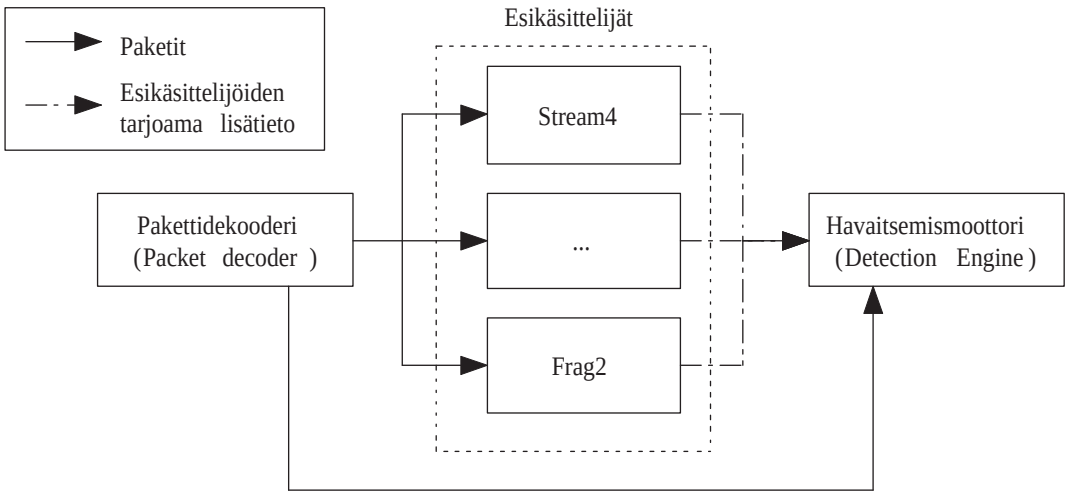
Snortin vahvuuksia on sen siirrettävyys eri järjestelmiin. Snort voidaan asentaa kaikille x86-alustoille kuten Linux, FreeBSD, OpenBSD ja Windows. Myös Sparc-arkkitehtuuri on tuettu, kuten Solaris, MacOS ja MkLinux. Snort on avoimen lähdekoodin projekti, joten sen lähdekoodi on vapaasti saatavilla GPL-lisenssin alaisuudessa ja tästä syystä se voidaan kääntää omatoimisesti myös muille alustoille. [18]

5.1.1 Toiminta

Snortin toiminta perustuu useaan erilaiseen esikäsittelijään (engl. *Preprocessor*) ja havaitsemismoottoriin (engl. *Detection engine*). Esikäsittelijöiden tehtävänä on esimerkiksi poikkeavuuksien havaitseminen ja yhteyden tilan tallentaminen. Esikäsittelijät helpottavat sääntöjen kirjoittamista, vähentävät väärien hälytysten määrää sekä tarjoavat sääntöjen vertailuun perustuvalle IDS-järjestelmälle tavan laajentaa yksinkertaista havaitsemismalliaan ilman tehokkuuden menetystä. [18]

Kun paketti vastaanotetaan, se käsitellään pakettidekooderissa (engl. *Packet decoder*), joka parsii paketin tiedot kentiksi, joita muut Snortin osat voivat käyttää hyväkseen. Dekooderin jälkeen paketti lähetetään esikäsittelijöille, jos sellaisia on asetettu. Esikäsittelijät ovat lisäosia (engl. *Plug-in*), jotka parsivat saamaansa dataa eri tavoin

riippuen, mikä on jatkoa ajatellen hyödyllistä. Ilman esikäsittelijöitä Snort tutkii jokaisen paketin erikseen, riippumatta muista paketeista, ja tästä syystä voivat jotkin hyökkäykset jäädä havaitsematta. Esikäsittelijä kokoaa usean paketin tietoja yhteen ja auttaa havaitsemaan esimerkiksi datan ylikirjoitukset päällekkäin menevissä fragmentaatioissa. Ne tarkkailevat esimerkiksi porttiskannauksia, kokoavat pilkottuja paketteja ja paljon muuta. Snort sisältää kymmeniä esikäsittelijöitä. [18]



Kuva 5.1: Snortin toiminta.

Esimerkkinä esikäsittelijästä on Snortin stream4-esikäsittelijä, joka pitää yllä tietoa yhteyksien tiloista. Tilojen tallennuksen avulla voidaan hälyttää paketeista, joita ei sallita yhteyden silloisessa tilassa. Esimerkiksi yhteyden lopetuksen jälkeen stream4 ei salli paketteja, jotka väittävät kuuluvansa lopetettuun yhteyteen. Esikäsittelijöiden käyttö lisää sääntöjen vertailunopeutta, sillä paketteja ei välttämättä tarvitse vertailla sääntöjen avulla vaan esikäsittelijä hoitaa sen. Esikäsittelijät tuovat lisää säätövaraa sääntöihin, sillä niiden tietoja voidaan käyttää hyväksi sääntöjä kirjoitettaessa. [18]

Seuraavaksi data siirretään esikäsittelijältä havaitsemismoottorille, joka vertaa sitä käyttäjän määrittelemiin sääntöihin ja toimii sääntöjen perusteella. Ennen sääntöjen vertailua Snort tarkastelee mitkä säännöt vastaavat käsiteltävänä olevaa pakettia. Ensimmäisenä tarkastetaan paketin protokolla (TCP, UDP, ICMP tai IP) ja käytetään vain sellaisia sääntöjä, jotka vastaavat paketin protokollaa. Kun relevantit säännöt on saatu selville, Snort käy ne läpi ja vertaa niitä paketin sisältöön. [18]

5.1.2 Säännöt

Ensimmäisissä versioissa Snort käytti ensimmäinen osuma -taktiikkaa (engl. *First match*), jolloin sääntöjen tarkastelu lopetetaan ensimmäisen sopivan säännön löytyessä. Näin Snort generoi vain yhden hälytyksen pakettia kohden, vaikka se olisi vastannut useita eri sääntöjä. Tämä kuitenkin mahdollisti luvussa 3.4 kuvatun väärinkäytöksen, jonka avulla vakavampi hyökkäys voidaan naamioida antamalla Snortin tunnistaa ensin jokin vähäisempi haavoittuvuus. Versiosta 2.1.3 lähtien käyttäjä voi määritellä miten Snort käsittelee useat osumat. [18]

Snortiin voidaan määritellä ohitussääntöjä (engl. *Pass rules*). Joissakin tapauksissa on helpompi määritellä ohitussääntöjä kuin useita erilaisia hälytyssääntöjä. Ohitussääntöjä voidaan käyttää esimerkiksi, jos tietyt verkossa olevat palvelimet aiheuttavat vääriä hälytyksiä, mutta hälytykset tulisi kirjata muilta verkon laitteilta normaalisti. Tällöin palvelimilta tulevat hälytykset voidaan ohittaa. [18]

Versiosta 2.1 lähtien Snortin sääntöihin on voinut määritellä Perlin säännöllisiä lausekkeita (engl. *Regular expression*), joiden avulla voidaan muodostaa mutkikkaita datakuvioiden etsimisen paketista. [18]

Sääntöjen tehtävänä on vastata jotakin säännöllistä kuviota verkkoliikenteessä, jotta tietyt hyökkäyskuviot voidaan tunnistaa. Kun verkkoliikenne vastaa tiettyä sääntöä, toteutetaan säännössä määritelty toimenpide, esimerkiksi tehdään hälytys ylläpitäjälle hälytysmoduulin avulla. Hälytysten lähettämiseen on useita eri tapoja, riippuen valituista hälytys-moduuleista, kuten SMB Pop-up Windows koneelle, lokitiedosto, ilmoittaa sokettien avulla toiselle laitteelle, SNMP-trap ja niin edelleen. Kolmannen osapuolen moduulit jopa siirtävät hälytykset ylläpitäjän hakulaitteeseen tai matkapuhelimeen. [18]

Snortin säännöt koostuvat otsakkeesta (engl. *Header*) ja rungosta (engl. *Body*), joka voi sisältää lisätietoja, kuten viitteitä dokumentteihin. Snort sisältää valmiiksi kattavat säännöt yleisimmistä hyökkäyksistä ja verkon ylläpitäjä voi helposti lisätä omia sääntöjä havaitsemiensa heikkouksien torjumiseen. Näin Snortin käyttäjät eivät ole riippuvaisia ulkopuolisista sääntöjen toimittajista, jotka julkaisevat säännöt uusiin hyökkäyksiin viiveellä. [18]

Säännön otsakkeen syntaksi on seuraava:

```
[toimenpide] [protokolla] [lähde-IP] [lähdeportti] [suunta]
[kohde-IP] [kohdeportti] (runko)
```

jossa `toimenpide` määrää, mitä tehdään, kun sääntö vastaa verkossa liikkuvaa pakettia (esimerkiksi `alert/log/pass`). Toimenpide voi olla `activate`, jol-

loin sääntö aktivoi jonkin dynaamisen säännön. Dynaamisen säännön toimenpiteenä on `dynamic`, jolloin ne ohitetaan, kunnes jokin sääntö aktivoi ne. Tämän jälkeen niitä käsitellään `log`-sääntöinä. Aktivointi tapahtuu määrittelemällä aktivointisääntöön `activates`-attribuutin ja dynaamisiin sääntöihin `activated_by`-attribuutin. Aktivointisääntö aktivoi kaikki samalla numerolla varustetut dynaamiset säännöt. Käyttäjä voi määritellä myös omia toimenpiteitään, jos Snortin omat toimenpiteet eivät riitä. `count`-muuttuja ilmoittaa kuinka monta pakettia Snort käsittelee ennen kuin dynaaminen sääntö kytketään pois päältä. Seuraava sääntö tallentaa lokiin viisi seuraavaa pakettia, jotka tulevat porttiin 143: [18]

```
activate tcp any any -> any 143 (content:``|E8C0FFFFFF|/bin'';
activates: 1;)
dynamic tcp any any -> any 143 (activated_by: 1; count: 5;)
```

Jos Snort toimii inline-tilassa, eli IPS-järjestelmänä, voidaan toimenpiteeksi määritellä myös `drop`, `reject` ja `sdrop`. `Drop` pakottaa IPTablesin pudottamaan paketin ja merkitsemään sen lokiin, `sdrop` on muuten sama, mutta siitä ei jää merkintää lokiin. `Reject` pakottaa IPTablesin pudottamaan paketin, merkitsemään sen lokiin ja lähettämään lähettäjälle joko TCP RST -viestin tai ICMP port unreachable -viestin. `Protokolla` määrittelee paketin protokollan ja `lähde-IP` ja `-portti` määrittelevät paketin lähteen IP-osoitteen ja portin. Vastaavalla tavalla määritellään paketin kohteen IP-osoite ja portti. `Suunta` ilmoittaa paketin kulkusuunnan verkossa, joka voi olla lähteeltä kohteelle (sisäänpäin), kohteelta lähteelle (ulospäin) tai kumpaan suuntaan tahansa (`->`, `<-` tai `<>`).

Säännöillä voidaan tarkkailla esimerkiksi tiettyihin portteihin tulevaa liikennettä tai tietyistä porteista lähtöisin olevaa liikennettä. Esimerkki porttien tarkkailusta: [18]

```
alert udp any 19 <> any 7 (msg: ``DOS UDP echo+chargen bomb'';
reference:cve,CAN-1999-0635; reference:cve,CVE-1999-0103;
classtype:attempted-dos; sid:271; rev: 3;)
```

Sääntö vastaa UDP-liikennettä, joka tulee jonkin IP-osoitteen portista 19, jonkin IP-osoitteen porttiin 7. Säännöissä voidaan käyttää ennalta määritettyjä muuttujia, jolloin sääntöjen muokkaus helpottuu, jos verkon laitteiden IP-osoitteet muuttuvat. [18]

Käyttäjä voi määritellä omia muuttujia lisäämällä Snortin konfiguraatiotiedostoon rivin:

```
var [muuttujan nimi] [arvo]
```

Muuttujaan voidaan määrittää useita arvoja käyttämällä taulukkosyntaksia:

```
var DMZ [10.1.1.1, 10.1.1.2, 10.1.1.3]
```

Säännöillä voidaan tarkkailla myös paketin sisältöä ja verrata sitä merkkijonosääntöihin. Esimerkkinä salasanatiedoston kysely, joka vertaa uri:n sisältöä määriteltyyn merkkijonoon:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg: ``WEB-CGI /www-board/passwd.txt access''; flow:
 to_server, established; uricontent:
 ``/wwwboard/passwd.txt''; nocase; reference:arachnids,463;
 reference:cve,CVE-1999-0953; reference:nessus,10321;
 reference:bugtraq,649; classtype:attempted-recon;
 sid:807 rev:7; )
```

Säännössä käytetään myös hyväksi flow-esikäsittelijää, jota kutsutaan säännön tarkistuksen yhteydessä ja sen palautusarvoa käytetään yhtenä säännön kriteerinä. Flow-esikäsittelijä pitää kirjaa IP-voista ja niiden tiloista. Monet säännöt voivat päteä vain tietyssä tilassa olevaan yhteyteen. IP-vuot voidaan erotella niiden IP-protokollan, lähde- ja kohdeosoitteiden sekä lähde- ja kohdeporttien perusteella. [18]

5.2 Cisco IDS 4200 -sarja

Ciscon IDS-järjestelmät ovat valmiita ratkaisuja, jotka tarjoavat laajat tunkeutumisenhavaitsemiskeinot verkkojen suojaamiseen ja tarkkailuun. Cisco 4200-sarja tarjoaa eritehoisia NIDS-sensoreita 45Mb/s nopeudesta aina gigabitin sekuntinopeuksiin asti. Cisco valmistaa myös konetason havaitsemisjärjestelmiä, joilla voidaan suojata esimerkiksi tärkeät palvelimet. Ciscon tarkoituksena on valmistaa järjestelmiä, joissa on tarkka ja älykäs uhkien tunnistus, helppo hallittavuus sekä joustava asennus eri tilanteisiin. Taulukossa 5.1 on listattu joitakin Ciscon 4200-sarjan verkkotason IDS- ja IPS-laitteita. [43]

Cisco tarjoaa myös IDS-moduulia (engl. *IDS Module, IDSM*) olemassa oleviin Ciscon reitittimiin, jolloin erillisiä NIDS-laitteita ei tarvita. Tämä vähentää kuluja ja helpottaa ylläpitoa. IDS-moduuli sisältää kiintolevyn, jonne tallennetaan lokitiedot sekä 10/100 ethernet portin hallintaliitynnälle. Koska IDS-moduuli saa datan

Taulukko 5.1: Cisco 4200-sarjan laitteita [44]

Sensori	Nopeus	Monitorointiliityntä	Hallintaliityntä
Cisco IDS 4215	65 Mbps	1 10/100BASE-TX	1 10/100BASE-TX
Cisco IDS 4250-SX	350 Mbps	1 10/100/1000BASE-SX	1 10/100/1000BASE-TX
Cisco IDS 4250-XL	800 Mbps	2 1000BASE-SX	1 10/100/1000BASE-TX
Cisco IPS 4240	250 Mbps	4 10/100/1000 BASE-TX	1 10/100BASE-TX
Cisco IPS 4255	500 Mbps	4 10/100/1000 BASE-TX	1 10/100BASE-TX
Cisco IPS 4260	1 Gbps	10/100/1000Base-TX	10/100/1000Base-TX

suoraan reitittimen väylästä (engl. *Bus*), se ei tarvitse erillistä monitorointiporttia. Moduulin toimintanopeus vaihtelee 10 Mb/s:sta 45: Mb/s:iin riippuen reitittimen mallista. IDS-moduuli sisältää saman Cisco IDS 5.0 -ohjelmiston, joka on käytössä 4200-sarjan laitteissa. Reitittimeen asennettaessa lisämoduuli toimii inline-tilassa, jolloin se voi estää kokonaan hyökkäykseen kuuluvat paketit. [43]

Reitittimien lisäksi IDS-moduuli (*IDSM*) on saatavilla myös Ciscon 6500 sarjan kytkimiin. Moduulista on kaksi eri versiota IDSM-1 ja IDSM-2, joista ensimmäistä versiota ei valmisteta enää. IDSM-1 kykenee toimimaan 250Mb/s nopeudella ja IDSM-2 600Mb/s. IDSM-2-moduuli on huomattavasti tehokkaampi kuin IDSM-1, sillä se käyttää samaa ohjelmistoa kuin 4200-sarjan IDS-laitteet. IDSM:n tehokkuus johtuu siitä, että se pystyy näkemään kaiken liikenteen, joka kulkee kytkimessä, sillä moduuli on liitetty suoraan kytkimen sisäiseen väylään (engl. *backplane*). Moduuli takaa myös helpon ja kustannustehokkaan IDS-ratkaisun jo olemassaolevien laitteiden yhteyteen. [43]

5.2.1 Cisco HIDS

Cisco tarjoaa myös konetason IDS-järjestelmiä (*HIDS*), joilla voidaan suojata tärkeitä palvelimia ja laitteita. Cisco HIDS-agentti voi toimia useissa erilaisissa käyttöjärjestelmissä, kuten Windowsissa ja Solariksessa. Agentti integroituu osaksi käyttöjärjestelmää, jolloin se voi suojata isäntä laitetta tehokkaasti. HIDS-sensori tarkkailee laitteelle tulevan liikenteen lisäksi myös käyttöjärjestelmän kutsuja, jolloin se voi havaita tehokkaammin järjestelmässä olevat haittaohjelmat. Sensori sisältää poikkeuksienhavaitsemisrutiineja, joiden avulla sensori voi torjua uusia hyökkäyksiä myös silloin, kun sääntöjä hyökkäyksen estämiseksi ei ole vielä kehitetty. Koska agentti on integroitunut osaksi käyttöjärjestelmää, voi se estää haitallisen koodin suorittamisen, joka käyttää hyväkseen esimerkiksi puskurin ylivuotohyökkäystä (engl. *Buf-*

fer overflow). Tämä ominaisuus on hyvin kriittinen, sillä yli 60% hyökkäyksistä käyttää hyväkseen puskurin ylivuotoja. [43]

HIDS agentteja on kaksi eri mallia, perusagentti (engl. *Standard agent*) sekä Web-agentti (engl. *Web Edition Agent*). Web-agentti sisältää saman toiminnallisuuden kuin perusagentti, mutta siihen on lisätty erityisesti Web-palvelimien suojaamiseen tarkoitettuja lisäominaisuuksia. Agentteja voidaan hallita keskitetyllä ja suojatulla konsolilla (engl. *Secure console*), minkä takia agenttien hallinta on helppoa. Esimerkiksi hallintaan voidaan käyttää Cisco IDS Host Sensor -konsolia, jonka kautta ylläpitäjä voi konfiguroida ja hallita kaikkia sensoreita, jotka sijaitsevat verkossa. Esimerkiksi uusien hyökkäysmallien (engl. *Signature*) päivittäminen agenteille tapahtuu keskitetysti konsolin kautta. [43]

5.2.2 IDS-laitteiden hallinta

Ciscon laitteita voidaan hallita usealla eri tavalla. Yleisimmät ovat komentorivikäyttöliittymä (engl. *Command Line Interface, CLI*), erilaiset graafiset käyttöliittymät (IDS Event Viewer ja IDS Device Manager) sekä Cisco VPN/Security Management Sollution (VMS). Komentorivikäyttöliittymään voidaan ottaa etäyhteys esimerkiksi SSH:n tai telnetin avulla. VMS on tarkoitettu laajojen verkkojen ja niiden tietoturvan hallintaan. VMS:n avulla ylläpitäjä voi hallita myös muita Ciscon laitteita, kuten reitittimiä ja palomuureja. [43]

Cisco on kehittänyt kaksi suojattua protokollaa, jotka on tarkoitettu IDS- ja muiden laitteiden hallintaan, PostOffice-protokolla ja Remote Data Exchange -protokolla (*RDEP*). RDEP tulee syrjäyttämään vanhentuneen PostOffice-protokollan. PostOffice-protokollaa ei tule sekoittaa Post Office -protokollaan (*POP3*), jota käytetään sähköpostien välittämiseen internetissä. PostOffice-protokolla on UDP-pohjainen ja käyttää pääsääntöisesti porttia 45000. Protokollan avulla voidaan siirtää komentoja IDS-laitteen ja hallintakonsolin välillä, virhe- ja hälytysviestejä sekä lokitietoja. Koska protokolla oli IDS-laitteiden pääsääntöinen viestintäprotokolla, kehitettiin siitä hyvin luotettava ja virheistä toipuva. Esimerkiksi hälytysviestit pitää kuitata, jotta IDS-laite tietää niiden menneen perille, muussa tapauksessa hälytysten lähetystä jatketaan.

5.2.3 Säännöt

Cisco jaottelee IDS-järjestelmissä käytettävät säännöt neljään ryhmään: tiedustelu (engl. *Reconnaissance*), informaatio (engl. *Informational*), pääsy (engl. *Access*) ja palvelunesto (engl. *Denial of Service*). Tiedustelu tarkoittaa sääntöjä, jolla pyritään ha-

vaitsemaan hyökkääjän verkon tiedusteluyritykset. Näihin kuuluvat DNS-kyselyt, porttien skannaukset sekä ping-kyselyt. Informaatio-ryhmät säännöt pyrkivät havaitsemaan onnistuneet tiedusteluyritykset. Pääsy-ryhmään kuuluvat säännöt havaitsevat kiellettyihin resursseihin pyrkivät yhteydet. Palvelunesto-säännöt havaitsevat verkon kuormittamisen, jolloin palveluiden käyttö voi estyä. [43]

Sääntöjen rakenne riippuu siitä, kuinka monta pakettia niiden laukaisemiseen tarvitaan. Säännöt voivat olla joko atomisia (engl. *Atomic*) tai yhdistelmäsääntöjä (engl. *Composite*). Atomiset säännöt voidaan havaita vain yhtä pakettia tarkastelemalla, kun taas yhdistelmäsääntöihin tarvitaan useita paketteja. [43]

Cisco jaottelee sääntöjen tarkastuksen erilaisille mikromoottoreille (engl. *Micro engine*), jotka tarkastavat tietyntyyppisiä sääntöjä. Mikromoottoreita ovat: [43]

Atominen Käytetään yksittäisten pakettien tarkastukseen.

Tulva (engl. *Flood*) Käytetään havaitsemaan palvelunestoyrityksiä.

Palvelu (engl. *Service*) Käytetään kun palvelut protokollapinon kerroksilla 5, 6 ja 7 tarvitsevat protokolla-analyysiä.

Tila (engl. *State*) Käytetään kun tarvitaan tilallista analyysiä.

Merkkijono (engl. *String*) Käytetään kun tarkastetaan merkkijonoja paketista.

Pyyhkäisy (engl. *Sweep*) Käytetään tarkastamaan verkkotiedusteluyritykset.

5.3 Bro

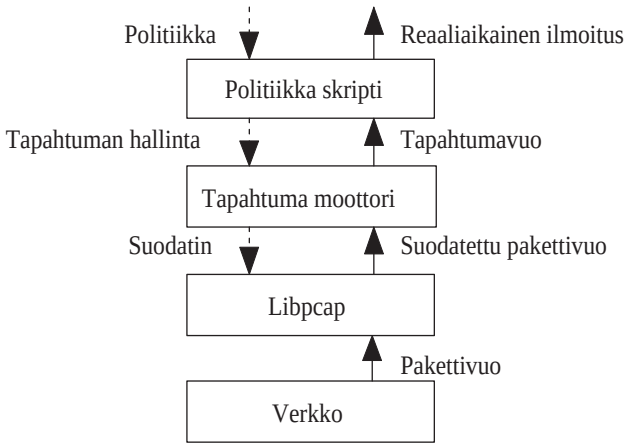
Bro on Vern Paxsonin kehittämä vapaan lähdekoodin NIDS-järjestelmä, joka on suunniteltu toimimaan nopeissa gigabitin verkoissa Unix-alustalla. Bro on tarkoitettu käytettäväksi kohteissa, joissa tarvitaan joustavaa ja erittäin kustomoitavaa järjestelmää. Bro on suunniteltu pääsääntöisesti tutkimuskäyttöön, kuin valmiiksi tietoturvatuotteeksi, jollaisia esimerkiksi Ciscon järjestelmät ovat. Brota voidaan käyttää esimerkiksi kaupallisen järjestelmän rinnalla varmistamassa toisen järjestelmän tulokset.

5.3.1 Toiminta

Bro jakaantuu erilaisiin kerroksiin, jotka toimivat protokollapinon tapaan prosessoiden tietoa ja välittäen käskyjä alemmalle tasolle. Alin kerros tuottaa suurimman

datamäärän, jonka takia sen tekemä työ tulee minimoida suorituskyvyn ylläpitämiseksi. Kuljettaessa tasoja ylöspäin datamäärä pienenee ja tiedon prosessointiin käytettävä aika kasvaa.

Bro koostuu neljästä kerroksesta (Kuva 5.2), jotka ovat verkko, libpcap, tapahtumamoottori (engl. *Event engine*) ja politiikka skriptien käsittelijä (engl. *Policy script intercepter*) .



Kuva 5.2: Bron kerrokset ja niiden viestit. [45]

Libpcap on pakettien kaappaamiseen käytetty kirjasto, jota käytetään yleisesti vapaan lähdekoodin ohjelmissa. Libpcap:n käyttö takaa yhteensopivuuden erilaisten verkkotekniikoiden kanssa, sillä Bro käsittelee libpcap:lta tulevaa dataa, eikä sen näin tarvitse tietää mitään verkon toteutuksesta. Libpcap voi myös suodattaa tiettyjä paketteja pois vuosta, jolloin seuraavien kerrosten työtä voidaan vähentää. [45]

Seuraavaksi libpcap lähettää suodatetun pakettivuon Bro:n tapahtumamoottorille, joka tarkastaa ensimmäisenä, että pakettien otsikkotiedot ovat muodostettu oikein (myös tarkistussumma tarkastetaan). Jos paketin otsikko on muodostettu väärin, Bro generoi tapahtuman, joka kuvastaa ongelmaa, ja hylkää paketin. Tapahtumamoottori huolehtii myös pakettien uudelleenkokoamisesta, jos ne on jaettu aikaisemmin osiin. [45]

Tarkistuksen onnistuttua, tapahtumamoottori etsii yhteyden tilan, joka on tallennettu taulukkoon yhdessä lähde- ja kohdeosoitteiden sekä lähde- ja kohdeporttien kanssa. Jos tietuetta ei löydy, luodaan uusi tietue. Tämän jälkeen paketti siirretään kyseisen yhteyden käsittelijälle, joita ovat esimerkiksi TCP-käsittelijä (engl. *TCP processing*) ja UDP-käsittelijä. [45]

TCP-käsittelijä tarkastaa paketin TCP-otsikoiden oikeellisuuden ja laskee TCP-paketin tarkistussumman. Tämän jälkeen otsikoista tarkastetaan SYN/FIN/RST-

liput ja asetetaan yhteyden tila vastaamaan lippuja. Yhteyden tilan muuttuessa luodaan erilaisia tapahtumia riippuen tilasta. Jos havaitaan ainoastaan SYN-paketti, eikä sen jälkeen yhteydessä kulje muita paketteja, generoidaan *connection_attempt*-tapahtuma. Muita tapahtumia ovat *connection_established*, *connection_rejected* ja *connection_finished*. Lopuksi TCP-käsittelijä siirtää paketin käsittelijälle, joka tarkastaa paketin data-osan, jos sellainen on olemassa. [45]

UDP-käsittelijä vastaa pitkälti TCP-käsittelijää, mutta on yksinkertaisempi, sillä UDP ei sisällä yhteyksien tiloja, paitsi yhdessä tapauksessa. Jos laite A lähettää UDP-paketin laitteelle B, jossa lähdeportti on p_A ja kohdeporttina p_B, Bro tulkitsee sen olevan pyyntö (engl. *Request*) ja luo pseudo-yhteyden laitteiden välille. Samoin, jos laite B lähettää paketin laitteelle A, jossa lähdeporttina on p_B ja kohdeporttina p_A, tällöin paketti tulkitaan vastaukseksi edelliseen pakettiin. Näin UDP-data voidaan erottaa toisista yhteyksistä, jotka noudattavat samaa mallia. Näistä luodaan joko `udp_request` tai `udp_reply`-tapahtumat. [45]

Seuraava taso on politiikkaskriptien käsittelijä, joka ottaa vastaan tapahtumia alemmalta tasolta. Se suorittaa skriptejä, jotka on kirjoitettu Bro-skriptikielellä. Skriptit sisältävät käsittelijät jokaiselle tapahtumalle, jotka tulevat alemmalta tasolta. Skriptit voivat suorittaa Bron omia komentoja, tallentaa tietoja lokiin (syslogin avulla) tai kirjoittaa tietoja kiintolevylle. Uusien ominaisuuksien luominen Brohon on helppoa luomalla vain uusi tapahtumien käsittelijä ja kirjoittamalla sen tuottamien tapahtumien käsittelyyn tarvittavat skriptit. [45]

5.3.2 Bro-kieli

Kuten edellä mainittiin, Bro sisältää erityisen skriptikielin, jonka avulla voidaan kuvata tapahtumien käsittelijöitä.

Bro-kieli käyttää vahvaa tyypitystä ja se sisältää viisi atomista tyyppiä, jotka ovat `boolean`, `integer`, `count` (vain positiiviset luvut), `double` ja `string`. Neljä ensimmäistä tyyppiä ovat aritmeettisia ja niitä voidaan laskea yhteen keskenään. Tarvittaessa erityyppiset muuttujat muunnetaan toiseksi tyypiksi. Aika voidaan ilmoittaa absoluuttisena aikana `time`-tyypillä ja aikaväliä `interval`-tyypillä. `Port`-tyypillä voidaan ilmoittaa joko TCP- tai UDP-porttia, määrittelemällä esimerkiksi `80/tcp`. Tunnetut portit voidaan ilmoittaa myös ennalta määrätyillä nimillä, kuten esimerkiksi `http`. Osoitetta vastaa `addr`-tyyppi, joka ilmoitetaan pistemuodossa. [45]

Bro sisältää myös erilaisia ryhmätyyppejä, kuten `record`, jonka avulla voidaan määritellä kokoelma erilaisia tyyppejä: [45]

```
type conn_id: record {
  orig_h: addr;
  orig_p: port;
  resp_h: addr;
  resp_p: port;
};
```

Tietyn tietueen kenttiin viitataan seuraavasti: `conn_id$orig_p`. Bro-kielessä voidaan käyttää taulukoita suurten tietomäärien tallentamiseen. Muita tyyppejä ovat `file`, jonka avulla voidaan kirjoittaa tiedostoon ja `list`, joka voi sisältää useita samantyyppisiä arvoja. [45]

Bro sisältää samantyyppiset operaatiot kuin C-kieli (`+, -, *, /, %, !, &&, ||, ?`). C-kielestä eroavat operaatiot ovat `in` ja `!in`, joilla voidaan testata jonkin muuttujan osallisuus esimerkiksi tiettyyn taulukkoon: [45]

```
23/tcp in services
```

Muuttujia voidaan määrittää `local`- tai `global`-avainsanalla, joista ensimmäinen luo muuttujan, joka on voimassa vain senhetkisessä funktiossa tai tapahtumankäsittelijässä. Staattisia muuttujia voidaan määritellä `const`-avainsanalla. Esimerkiksi: [45]

```
const allowed_services: set[addr, port] = {
   [ftp.lbl.gov, [ftp, smtp, ident, 20/tcp]],
   [nntp.lbl.gov, nntp]  };
```

Esimerkki skriptistä, joka käsittelee tapahtumaa:

```
event connection_established(c: connection)
{
 local id = c$id;
 local service = id$resp_p;
 local inbound = is_local_addr(id$resp_h);

 if (inbound && [id$resp_h, service] !in allow_my_services)
     NOTICE ([$note=SensitiveConnection, $conn=c,
  $msg=fmt("hot: %s", full_id_string(c)) ]);
  if (inbound && service in terminate_successful_inbound_service)
          terminate_connection(c);
}
```

6 Netflow

Verkon liikenteen kasvaessa ylläpitäjien on entistä hankalampi seurata verkon liikenteen ominaisuuksia tai mistä liikenne on peräisin. Liikenteen määrään ja tyyppiin voivat vaikuttaa erilaiset uudet ohjelmistot, käyttäjät ja aika päivästä. Liikenteen ja sen määrän tunteminen auttaa verkon suunnittelussa, virittämisessä ja laajentamiseen investoitaessa. Netflow sijoittuu SNMP:lla kerättävän datan ja kokonaisten pakettien kaappauksen välille ja se on passiivinen verkon valvontaan käytetty menetelmä. Netflow tarjoaa yksityiskohtaisen datan sijasta yhteenvetoja liikenteestä. [46]

Luvussa kuvataan Netflow-datan keräämistä reitittimiltä, sen analysointia sekä Netflown käyttämistä matojen ja muiden uhkien havaitsemiseen. Netflow-dataa käytetään verkkojen liikenteen mittaukseen ja trendien ennustamiseen. Netflown avulla voidaan havaita erilaisia hyökkäyksiä ja näin vähentää IDS-laitteiden taakkaa, sillä Netflow-datan kerääminen kuluttaa vähemmän resursseja kuin koko paketin kaappaaminen.

6.1 Yleistä

Netflow on Ciscon kehittämä verkon liikenteen keräys- ja mittausteknologia, jonka avulla verkon liikennettä voidaan tarkkailla sen kulkiessa reitittimien tai kytkinten läpi. Netflow on osa Ciscon IOS -ohjelmistoa, joten se on otettavissa helposti käyttöön Ciscon reitittimissä ja kytkimissä. Netflow-datan analysoinnilla saadaan selville ruuhkien syyt ja niiden lähteet. Myös Juniper-reitittimet tuottavat Netflow:n kaltaista dataa, josta saadaan samat tiedot kuin Ciscon laitteista. [47]

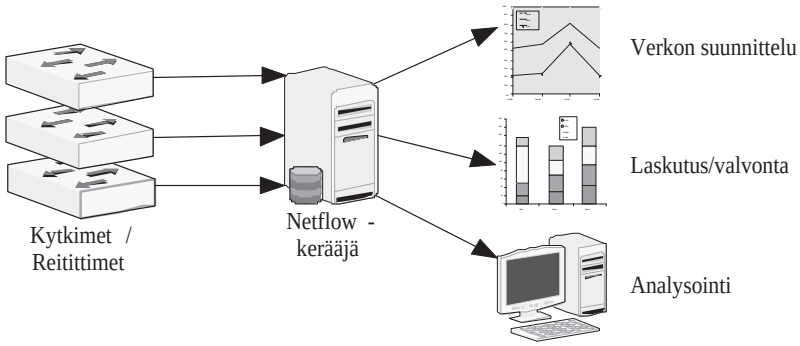
Netflow-dataa voidaan käyttää hyväksi esimerkiksi verkon ohjelmien ja käyttäjien valvontaan, jolloin voidaan saada selville tiettyjen käyttäjien tai ohjelmien kuluttama verkkokapasiteetti. Tämä auttaa myös verkon resurssien suunnittelussa ja jakamisessa. Netflow-data auttaa verkon suunnittelussa ja sen laajentamistarpeiden kartoittamisessa. Pitkältä ajalta kerätty Netflow-data paljastaa verkon käytön trendin ja sitä arvioimalla voidaan ennustaa esimerkiksi uusien palvelinten tarvetta tulevaisuudessa. Kolmas käyttökohde Netflow-datalle on tietoturvauhkien analysointi, esimerkiksi porttiskannausten ja palvelunestohyökkäysten analysointi ja torjuminen. Viimeinen käyttökohde on erilaiset liikenteen määrään perustuvat laskutukset

ja rajoitukset. [47]

6.2 Datan kerääminen

Netflow-arkkitehtuuri sisältää kolme pääkomponenttia: sensorin (engl. *Sensor, Probe*), kerääjän (engl. *Collector*) ja raportoijan (engl. *Reporting system*). Sensori analysoi ja kerää IP-datavuot, jotka kulkevat reitittimen tai kytkimen läpi. Kerääjät vastaanottavat reitittimiltä UDP-paketissa tulevan Netflow-datan ja tallentavat sen käyttäjän haluamalla tavalla. Analysointi- ja raportointityökaluilla voidaan analysoida verkossa kulkevia yhteyksiä ja tuottaa niistä erilaisia raportteja ja graafeja. Kuvassa 6.1 on esitetty Netflow-datan keräys ja käyttö erilaisissa sovelluksissa. [48]

Netflow-dataa voidaan tuottaa myös erillisten sensoriohjelmien (*Sensor, Probe*) avulla, jolloin reitittimien data joudutaan peilaamaan laitteelle, johon on asennettu Netflow-dataa generoiva ohjelma. Tällaisia ohjelmia ovat esimerkiksi *softflowd*[1], *FProbe*[2] ja *nProbe*[3]. Jos verkossa on jo Unix-pohjainen palomuuri, on järkevää sijoittaa sensoriohjelma siihen, sillä silloin kaikki sisään tuleva data kulkee palomuurin läpi. Netflow-datan kerääminen ei tarvitse kohtuuttomia resursseja, vaan 500MHz tietokoneella, jossa on ainakin 256Mb muistia, voidaan kerätä dataa yli 20Mb/s nopeudella toimivasta verkosta. [48]



Kuva 6.1: Netflow-datan keräys ja käyttö. [47]

Netflow-data sisältää tyypillisesti seuraavat kentät: [47]

- Lähde- ja kohde IP-osoitteet
- Lähde- ja kohdeportit

[1]http://www.mindrot.org/projects/softflowd/
[2]http://fprobe.sourceforge.net/
[3]http://www.ntop.org/nProbe.html

- Palvelun laatu (engl. *Type Of Service, ToS*)
- Pakettien ja bittien lukumäärä
- Alku- ja loppuaikaleimat
- Reitittimen IP-osoite, joka käsitteli vuon
- Sisään tulevan ja ulos menevän verkkoliitynnän numerot
- TCP-liput
- Reititystietoja

Robin Sommer ja Anja Feldmann tutkivat [46] Netflow-datan luotettavuutta ja oikeellisuutta erilaisissa tilanteissa verrattuna SNMP:n tai pakettien kaappauksen avulla kerättyyn dataan. Netflown luonteesta johtuen se ei kykene tarjoamaan samaa määrää dataa kuin pakettien kaappauksella saavutetaan. Tutkimus osoitti, että Netflow sisältää joitakin ongelmia, jotka saattavat vääristää tuloksia. Netflow esimerkiksi voi yhdistää paketteja eri TCP-yhteyksistä yhdeksi vuoksi. Tämä voi tapahtua silloin, kun samaa sokettia (engl. *Socket*) käytetään useisiin yhteydenmuodostusyrityksiin, esimerkiksi tiedostonjako-ohjelmissa. Tällöin yhdessä vuossa näkyvät kaikki yhteydenmuodostuksissa liikkuneiden pakettien määrät vaikka ne periaatteessa tulisi erotella omiksi voikseen. Verrattaessa SNMP:n avulla tehtyihin laskelmiin, Netflown avulla saadut tulokset poikkesivat niistä hieman, mutta poikkeamat olivat hyvin pieniä. Heidän mielestään Netflown antamat tiedot olivat tarkimmat käytettäessä suhteellisen pitkiä aikoja tietojen keräämiseen. [46]

6.3 Analysointi

Netflow dataa voidaan analysoida usealla eri tavalla. Analysoinnin avulla voidaan tarkastella verkon liikennemääriä ja etsiä laitteet, jotka muodostavat eniten liikennettä verkkoon. Analysoinnilla voidaan etsiä joko laitepari, joka muodostaa suurimman liikennekuorman tai yksittäinen laite, joka liikennöi usean eri tahon kanssa.

6.3.1 Top N ja vertailukohta -analyysi

Top N -analyysi tarkoittaa *N* ensimmäisen alkion ottamista tietystä joukosta, joka on yleensä järjestetty tietyn parametrin mukaan. Esimerkiksi 10 verkon eniten liikennettä tuottavaa laitetta saadaan poimimalla 10 ensimmäistä laitetta listasta, joka on järjestetty liikenteen määrän mukaan.

Vertailukohta-malli (engl. *Baseline*) on malli verkon normaalista liikenteestä, joka perustuu aikaisemmin havaittuun liikenteeseen. Suuret vaihtelut nykyisessä liikenteessä havaitaan vertaamalla sitä aikaisemmin kerättyyn dataan. Kaikki suuresti poikkeava liikenne merkitään epäilyttäväksi liikenteeksi. [49]

Yleisin tapa analysoida Netflow-dataa trendien selvittämiseksi, on yhdistää Top N ja vertailukohta -analyysit. Tällöin kiinnitetään huomiota tavallisuudesta poikkeaviin liikennemääriin, jolloin voidaan ennustaa verkon liikenteen kasvua. [49]

Top N ja vertailukohta -malli jakautuu kahteen osaan: Top N sessio ja Top N data. Top N sessio tarkoittaa, että yksi laite luo normaalia enemmän yhteyksiä joko yksittäiseen kohdelaitteeseen tai laiteiden ryhmään ja yhteyksien määrä eroaa aiemmin havaituista yhteyksien määristä. Top N ja vertailukohta -analyysia voidaan käyttää myös Top N data analysoinnissa. Top N data tarkoittaa, että yksi laite tuottaa tavallisuudesta poikkeavan määrän verkkodataa joko yksittäiseen kohdelaitteeseen tai niiden ryhmään tietyllä aikavälillä. [49]

6.3.2 Havaitseminen sääntöjen avulla

Netflow-dataa voidaan suodattaa erilaisten sääntöjen avulla, jolloin voidaan luoda hyvinkin monipuolisia raportteja verkon liikenteestä. Sääntöjen avulla voidaan suodattaa sellaiset vuot, jotka sisältävät tietyt portit tai IP-osoitteet, ja tehdä raportteja näiden pohjalta. [49]

IP-osoitteiden tarkastamisella voidaan havaita esimerkiksi vuot, joissa on IP-osoitteeksi väärennetty jokin varattu IP-osoiteavaruus, jota ei pitäisi esiintyä sisäverkkojen ulkopuolella. Esimerkiksi IANA[4] on määritellyt tietyt IP-osoiteavaruudet yksityisten verkkojen käyttöön. Jos sisäverkkoon tulevan paketin lähdeosoite on jokin varattu IP-osoite, voidaan siitä päätellä, että osoite on väärennetty. Sisäverkosta lähtevät paketit sallitaan vain, jos niiden lähdeosoite on jokin käytössä oleva sisäverkon osoite. Samoin sisäverkkoon pyrkivät paketin kohdeosoite tulee löytyä sisäverkosta. [49]

Esimerkkinä IANA:n määrittelemät yksityiset tai varatut IP-osoiteavaruudet:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255
- 169.254.0.0 - 169.254.255.255

[4] http://www.iana.org

- 127.0.0.0 - 127.255.255.255

Palvelimille voidaan antaa ylimääräistä suojaa tarkastelemalla niiden Netflow-yhteyksiä. Yhteydet jonkin palvelimen käyttämättömään porttiin tulisi aiheuttaa hälytys, etenkin jos vuossa on vain ACK-lippu asetettuna (eikä ollenkaan RST/ACK-lippua). Palomuurin tulisi suodattaa yhteydet palvelimien käyttämättömiin portteihin, joten palomuurin konfiguraatiossa voi olla jotain vikaa. Toinen vaihtoehto on, että palvelin on saastunut ja siinä on käynnissä haittaohjelma, johon yritetään ottaa yhteyttä ulkopuolelta. Hälytys tulisi antaa ulospäin kulkevasta vuosta, jonka lähdeosoitteena on jonkin palvelimen käyttämätön portti ja vain ACK-lippu asetettuna. Tällöin kyseisessä palvelimessa voi olla haittaohjelma, joka pyrkii ottamaan yhteyttä internettiin. [50]

6.4 Hyökkäyksien havaitseminen Netflown avulla

IDS-laitteiden heikkoutena on niiden kyky tutkia nopeissa verkoissa liikkuvaa dataa. Tänä päivänä gigabitin sekuntivauhdilla toimivat verkot aiheuttavat ongelmia pakettien kaappaamiseen perustuvalle IDS-järjestelmälle. Tulevaisuudessa verkkojen kapasiteetti kasvaa vielä entisestään pahentaen ongelmaa. IDS-laitteiden apuna voidaan käyttää Netflow-dataa, jonka kerääminen ei kuluta kohtuuttomasti reitittimien ja kytkinten resursseja. Vaikka Netflow-dataa ei voida suoraan käyttää IDS-laitteiden syötedatana, sillä se sisältää aivan liian vähän tietoa, voidaan sitä käyttää erilaisten matojen ja porttiskannausten havaitsemiseen. Matoepidemioiden aikana tiettyihin portteihin tehtyjen skannausten määrä lisääntyy ja se voidaan havaita Netflow-dataa analysoitaessa. Samoin sisäverkossa oleva saastunut kone voidaan havaita sen avaamien uusien yhteyksien kasvuna. Madot pyrkivät tartuttamaan muita koneita ja tästä syystä ne etsivät uusia isäntäkoneita. Tämä etsiminen tuottaa suuren määrän uusia yhteyksiä saastuneesta laitteesta.

Netflow-dataa voidaan kerätä useista verkon reitittimistä samaan aikaan, jolloin Netflown avulla voidaan analysoida liikennettä joissain tapauksissa jopa paremmin kuin IDS-laitteilla, jotka eivät välttämättä näe jokaisen reitittimen läpi kulkevaa dataa. Netflow on myös edullinen ja helposti käyttöönotettava lisätekniikka IDS-laitteiden tueksi, jos verkko on jo rakennettu Ciscon verkkolaitteista.

Tang-Long Pao ja Po-Wei Wang julkaisivat tutkimuksessaan [51] menetelmiä, joiden avulla Netflow-dataa voidaan käyttää erilaisten hyökkäyksien havaitsemiseen, kuten Ping sweep, porttiskannaukset ja palvelunestohyökkäykset.

6.4.1 Ping Sweep

Ping pyyhkäisy (engl. *Ping Sweep*) voidaan tunnistaa samasta lähdeosoitteesta tulevien UDP-pakettien avulla, joiden kohteena on sisäverkon eri IP-osoitteet ja portit. Ping pyyhkäisylle on myös ominaista pakettien pieni koko. Uudet Ping pyyhkäisymenetelmät pyrkivät jakamaan tutkinnan pidemmälle aikavälille, joten hyökkäystä tulee etsiä sekä lyhyessä ajassa kerätystä Netflow-datasta että pitkäaikaisestakin datasta. [51]

Toinen Ping pyyhkäisyn ominaisuus on vain toiseen suuntaan kulkeva liikenne. Koska hyökkääjä ei voi tietää mitkä laitteet tai palvelut ovat käytössä verkossa, tulee myös käyttämättömiin IP-osoitteisiin liikennettä. Lyhyen aikavälin analyysillä tällainen hyökkäys voidaan havaita helposti. [51]

Ping pyyhkäisy voidaan tunnistaa tekemällä sääntö, joka tarkkailee samasta lähdeosoitteesta tulevia paketteja. Hajautettujen skannausten estämiseksi tulisi käyttää useaa eri aikaväliä tulosten keräämiseen, esimerkiksi 10, 60 ja 300 sekuntia. Jokaisella aikavälillä on eri kynnysarvot, joiden ylittyessä päätellään skannauksen olevan käynnissä. Ensimmäisellä kerralla IP-osoite tulisi laittaa varoituslistalle ja jos sama IP-lisätään sinne toisen kerran, voidaan se määritellä hyökkäykseksi. [51]

6.4.2 Porttiskannaukset

Porttiskannaukset voidaan havaita laskemalla jokaiseen sisäverkon IP-osoitteisiin tulevien pakettien määrä, joissa on eri kohdeportti. Pakettien lähteenä on harvoin oikean hyökkääjän osoite, sillä hyökkäykset tehdään murretuilta koneilta ja usein hajautetusti. TCP-porttiskannauksessa käytetään hyväksi joko täysin avoimia yhteyksiä, puoliavoimia yhteyksiä (*Stealth scan*) tai FIN-bitillä varustettuja paketteja. Pitkän aikavälin tutkinnalla havaitaan hajautetut hyökkäykset. [51]

6.4.3 Palvelunestohyökkäys

Palvelunestohyökkäyksen (*DoS, DDoS*) ominaisuutena on suuri määrä yhteydenottopyyntöjä joko yhteen tai useampaan IP-osoitteeseen lyhyessä ajassa. Usein lähdeosoite on väärennetty, joten kohdelaite ei välttämättä pysty vastaamaan oikein hyökkäyksen lähteelle. Ominaispiirteenä hyökkäykselle on kohdelaitteelle tuleva paketti, johon lähetetään vastaus ja tämän jälkeen yhteyttä ei käytetä. Koska hyökkäykset tulevat yleensä kaapatuilta koneilta, on hyökkäyksen estäminen hankalaa. Hyökkäyksen alettua tulisi palomuurin ja verkon reunalla olevien reitittimien pääsylistojen avulla estää hyökkäyksen lähteestä tuleva liikenne. Hyökkäyksen loput-

tua tämä esto pitää purkaa, jotta normaalit käyttäjät voivat käyttää sisäverkon palveluja. [51]

6.5 Matojen havaitseminen

Madon saastuttama verkko voidaan yleensä tunnistaa verkon liikenteen rajusta kasvusta. Liikenteen kasvu aiheutuu madon leviämisyrityksistä, jolloin se skannaa muita verkkoja löytääkseen sopivia uhreja. Liikenteen kasvua aiheuttaa myös reitittimien ja laitteiden lähettämän negatiiviset vastaukset, kun mato pyrkii ottamaan yhteyttä porttiin tai laitteeseen, jota ei ole olemassa. [52]

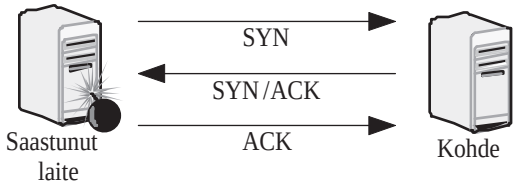
Kiivaasti liikennöidyssä verkossa matojen generoimaa liikennettä saattaa olla hankala havaita verkon muusta liikenteestä. Aiemmin esitetyllä Top N -analyysillä voidaan saada selville verkon suurimmat liikennöijät. Kiivaasti liikennöidyssä verkossa suurimmat liikenteenaiheuttajat ovat yleensä palvelimet. Karsimalla listasta pois palvelimet, nähdään seuraavaksi eniten liikennöivät laitteet, mutta niiden saastumista voi olla hankala todentaa.

Top N -analyysillä ei saada tarpeeksi tarkkoja havaintoja matojen saastuttamista laitteista. Parempia tuloksia saadaan supistamalla tutkittavien Netflow-yhteyksien määrää, esimerkiksi tarkastelemalla TCP-lippuja. Koska madot pyrkivät leviämään etsimällä mahdollisimman monta uhria verkosta, joihin ne yrittävät tarttua. Useimmat madot käyttävät TCP-yhteyksiä etsiessään uusin kohteita, joten verkossa liikkuu paljon TCP SYN -paketteja. [50]

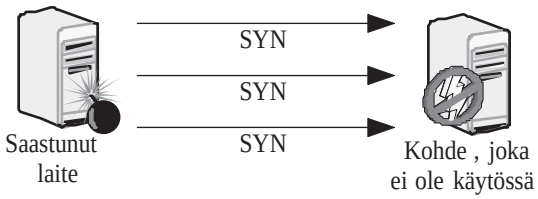
Madon lähettämille SYN-paketeille voi tulla kolme erilaista vastausta, riippuen kohteesta. Jos kohdelaite on toiminnassa ja siinä on käynnissä madon etsimä palvelu, nähdään verkossa normaali kolmitie-kättely, kun mato avaa uuden yhteyden kohteeseen. Mato lähettää SYN-paketin, johon kohdelaite vastaa SYN/ACK-paketilla, lopuksi mato kuittaa yhteyden ACK-paketilla (kuva 6.2). Netflown versio viisi käyttää kumulatiivista TAI-ehtoa (engl. *OR*) TCP-lipuille koko yhteyden ajalta, joten Netflow-tietueesta tulisi löytyä ACK/PUSH/SYN/FIN- tai ACK/SYN/FIN-merkinnät kumpaankin suuntaan kulkevasta liikenteestä. [50]

Jos mato yrittää saastuttaa laitetta, joka ei ole käytössä, nähdään verkossa vain madon lähettämiä SYN-paketteja, joihin ei vastata (kuva 6.3). Netflow-datassa tämä nähdään yhteytenä, jossa on vain SYN-lippu asetettuna. [50]

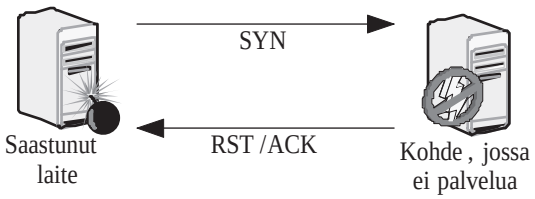
Kolmannessa tapauksessa kohde on käytössä, mutta portti on suljettu. Tällöin kohde lähettää vastaukseksi RST/ACK-viestin (kuva 6.4). TCP:n toteutuksen mukaan laitteen on lopetettava liikennöinti kun se vastaanottaa RST-paketin, tästä syystä Netflow-tietueessa on vain SYN-lippu asetettuna. [50]



Kuva 6.2: Kohde on käytössä ja palvelu vastaa. [50]



Kuva 6.3: Kohde ei ole käytössä. [50]



Kuva 6.4: Kohde käytössä, mutta portti suljettu. [50]

Osa madoista arpoo IP-osoitteet, joihin ne yrittävät levitä. Koska usein generoiduissa IP-osoitteissa ei ole laitetta vastaamassa, generoituu verkkoon paljon SYN-viestejä, jotka tulevat saastuneesta laitteesta. Tällöin saastuneen laitteen havaitseminen helpottuu. Toiset madot käyttävät erilaisia ennalta määritettyjä listoja, joista ne arpovat skannattavat verkot. Listojen avulla matojen leviämisnopeus kasvaa ja olemattomien verkkojen skannaus pienenee. Osa madoista käyttää saarihyppelytekniikkaa (engl. *Island hopping*) leviämiseen. Saarihyppelyssä mato pyrkii kiinnittämään huomionsa pääsääntöisesti nykyiseen verkon osaan, ennen kuin se yrittää hypätä seuraavaan aliverkkoon. Saarihyppely-tekniikkaa käyttävä mato on hyvin tehokas esimerkiksi NAT-verkoissa. Jos mato saastuttaa yhden verkossa olevan kohteen, on hyvin todennäköistä, että seuraavaksi se skannaa samassa verkossa olevat muut laitteet valitsemalla skannattavan aliverkon tietyllä todennäköisyydellä. Yleensä valittavat aliverkot ovat /24, /16 ja /8. [50] [52]

Madon saastuttama laite voidaan havaita Netflow-datasta seuraavasti: Ensiksi tulee hakea kaikki sellaiset tietueet, joissa vain SYN-lippu on asetettuna. Laskemalla yksilöllisten IP-osoitteiden määrän ja järjestämällä ne osumien mukaan suuruusjärjestykseen, saadaan lista potentiaalisista laitteista, jotka voivat olla saastuneita. Ennalta määrätyn osumien määrän mukaan voidaan päätellä, missä laitteissa todennäköisesti on mato. Tämän jälkeen Netflow-data tutkitaan toiseen kertaan hakemalla kaikki vuot, joissa lähdelaitteena on ollut jokin aikaisemmin saadun listan laitteista. Näin saadaan jokaisen epäillyn laitteen kaikki vuot selville. Laskemalla jokainen yksilöllinen kohdeportti edellisen kohdan IP-osoitteille, saadaan lista kaikista porteista, joihin IP-osoitteista on otettu yhteyttä. Järjestämällä portit yhteydenottokertojen mukaiseen järjestykseen saadaan jokaiselle IP-osoitteelle luettelo yleisimmistä porteista, joihin se on ottanut yhteyttä. [50]

Seuraavasta esimerkistä voidaan havaita, että IP-osoitteen `61.236.123.225` laitteessa on W32.gaobot.sa-mato, sillä sen ominaisuuksiin kuuluu ottaa yhteyttä portteihin 3127 (MyDoom backdoor), 2745 (Bagle backdoor) ja 6129 (Dameware):

```
potential host1: 61.236.123.225
------
 84 times probe on dstport 1025
 76 times probe on dstport 80
 72 times probe on dstport 2745
 64 times probe on dstport 3127
 48 times probe on dstport 6129
```

Saastunut laite voidaan havaita, jos ulkoverkosta tulee useita RST/ACK-paketteja

samaan kohdeosoitteeseen. Tällöin kyseessä voi olla madon saastuttama laite, joka skannaa käytössä olevia laitteita, joissa ei ole madon etsimää palvelua käynnissä. [50]

Madot, jotka käyttävät UDP-protokollaa leviämiseen, voidaan havaita ICMP-viestien avulla. ICMP-viestien avulla verkon laitteet ilmoittavat olemattomasta kohdelaitteesta tai portin olevan suljettu. Netflow-tietueessa ei ole erillistä ICMP-type- tai ICMP-code-kenttää, vaan tähän tarkoitukseen käytetään kohdeportti-kenttää. Jotta tyyppi ja koodi saadaan selville, on kohdeportti-kentän sisältö muutettava heksadesimaali muotoon ja näin saadusta luvusta voidaan erottaa tyyppi ja koodi. Esimerkiksi kohdeportti 2048 on heksadesimaalina 800, jossa 8 tarkoittaa tyyppiä ja 00 koodia, eli *ICMP echo request*. Portti 769 on 301 heksadesimaalina, eli *ICMP host unreachable* -viesti. [50]

Tarkkailemalla *ICMP echo request*, *ICMP host unreachable* ja *ICMP port unreachable* -viestien määrää verkossa samalla tavalla kuin vastaavia TCP-paketteja, voidaan havaita matojen saastuttamat laitteet.

6.6 Netflown käyttö IDS:n tukena

Netflow-data on ensisijaisesti tarkoitettu verkon toiminnan tarkkailuun ja sen analysointiin graafien ja raporttien avulla. Netflow-datan avulla ei voida korvata normaalia IDS-laitetta, sillä useimmat IDS-laitteiden havaitsemat hyökkäykset eivät näy Netflow-datassa, sillä se ei sisällä tietoa pakettien datasta. Näin esimerkiksi merkkijonon tunnistukseen perustuvia sääntöjä ei voida toteuttaa Netflown avulla. Liikenteen analysointiin perustuvat haittaohjelmien havaitsemisratkaisut ovat helposti otettavissa käyttöön verkoissa, joissa Netflowta käytetään ennestään erilaisten trendien analysoimiseen. Kustannuksia pienentää myös monen reitittimen yhtäaikainen valvominen yhdellä analysointilaitteella verrattuna jokaisen reitittimen yhteyteen asennettavaan NIDS-laitteeseen.

Netflow-työkaluilla reaaliaikaisen tarkastuksen saavuttaminen on hankalaa, sillä raporttien tarkastelu on tarkoitettu tehtäväksi jälkikäteen. IDS-laitteet pitävät muistissaan paketit, jotka voivat liittyä laajempaan hyökkäykseen ja näin havaitsevat hyökkäyksen heti kun hyökkäyksen varmistava paketti saapuu. Netflow-data taas lähetetään reitittimeltä sopivissa purskeissa ja se voi olla jo vanhentunutta saapuessaan. Lisäksi analysointityökaluja ei ole suunniteltu reaaliaikaiseen tarkastukseen, vaan ne vaativat tietyn määrän kerättyä dataa, jotta tiettyjä hyökkäyksiä voidaan havaita. Jos analysointi tehdään joka kerta kun uusi Netflow-paketti vastaanotetaan reitittimeltä ja samalla analysoidaan siihen asti kertyneet tietueet tietyltä aikavälil-

tä, tulisi tarkastuksesta todella raskas. Joidenkin skannauksien havaitseminen vaatii useiden minuuttien aikana kertyneet tietueet. Esimerkiksi varovasti suoritetun hajautetun porttiskannauksen havaitsemiseen voidaan tarvita jopa 30 minuuttia.

Netflow-dataan perustuvat havaitsemismenetelmät eivät kykene estämään käynnissä olevaa hyökkäystä, koska liikenne on jo ehtinyt mennä reitittimestä ennen kuin analysointiohjelmat saavat tiedon siitä. Analysointi voidaan myös suorittaa viiveellä, jolloin hyökkäys on saattanut olla käynnissä jo kauan. Liikenne voidaan havaitsemisen jälkeen katkaista erilaisilla keinoilla, mutta esimerkiksi nopeasti leviävän madon tapauksessa vahinko on jo tapahtunut.

Netflowta käyttävät havaitsemismenetelmät ovat tehokkaimpia käytettäessä verkon suojaamiseen jo tarttuneiden matojen ja haittaohjelmien leviämiseltä. Netflow-datasta voidaan helposti erottaa laitteet, jotka ovat saaneet tartunnan ja alkavat äkillisesti generoimaan paljon liikennettä verkkoon, joka vastaa jonkin madon leviämiskuviota. IP-osoitteet, joiden epäillään saaneen tartunnan, voidaan tämän jälkeen esimerkiksi estää pääsemästä verkkoon.

Tunnettuja matoja voidaan havaita ennalta määrätyillä säännöillä, mutta uusien matojen havaitsemiseen ei voida käyttää sääntöjä. Tuntemattomia matoja voidaan havaita verkon, tai jonkin laitteen, liikenteen kasvaessa epänormaalin suureksi. Verkon liikennemäärää voidaan verrata aiemmin kerättyihin liikennemääriin ja vertailla niitä keskenään. Vertailu voidaan toteuttaa esimerkiksi Top N -analyysillä, joka tarkkailee joko yhteyksien tai liikenteen määrää (luku 6.3.1). Myös erilaiset graafit paljastavat nopeasti verkossa tapahtuvat liikenteen kasvut. Netflow-dataa voidaan käyttää esimerkiksi uusien matoepidemioiden havaitsemiseen (engl. *Day zero attack*), kun verrataan nykyisiä liikennemääriä aiemmin kerättyihin tilastoihin.

Tämänhetkiset vapaan lähdekoodin työkalut Netflow-datan tarkastamiseen perustuvat suurelta osin raporttien ja graafien tekemiseen. Työkalut tarjoavat tehokkaan tavat suodattaa ja analysoida liikennettä ja tuottaa tuloksista raportteja. Toistaiseksi raporttien käsittelyyn ei ole olemassa kunnollisia välineitä, vaan raporttien pohjalta tehtävät toimenpiteet tulisi suorittaa ihmisen toimesta. Komentoriviltä toimivien raportointityökalujen vastetta voidaan kuitenkin analysoida erilaisilla skripteillä, jotka suorittavat analyysien perusteella toimenpiteitä, kuten konfiguroivat reititintä estämään saastuneen laitteen liikennöinnin verkkoon. Organisaatioiden erilaiset tarpeet ovat osaltaan ruokkineet omien skriptien tekemistä ja toistaiseksi yleistä työkalua raporttien pohjalta tehtäville toimenpiteille ei ole.

Suuntaus Netflow-datan käyttöön IDS-laitteiden tukena on kuitenkin selvä, sillä useat kaupalliset ohjelmat sisältävät jo erilaisia toteutuksia matojen havaitsemiseen ja IDS-järjestelmien tukemiseen Netflow-datan avulla. Myös erilaisia verkon

liikenteen monitorointiin perustuvia kaupallisia poikkeuksienhavaitsemisjärjestelmiä (engl. *Network Behavior Anomaly Detection, NBAD*) on kehitetty useita vuosia, mutta kiinnostus niitä kohtaan on alkanut vasta äskettäin. NBAD-laitteet voivat havaita uusien matojen aiheuttaman epidemian huomattavasti aikaisemmin kuin sääntöihin perustuvat järjestelmät. NBAD-järjestelmät perustuvat verkon liikenteen valvontaan ja sen analysointiin. Aikaisemmin NBAD-järjestelmät perustuivat erillisiin sensoreihin, joita asennettiin verkkoon. Järjestelmän kustannuksista ja asennuksen hankaluudesta johtuen nykyään yhä useampi valmistaja tukee reitittimien Netflow-datan käyttöä järjestelmien syötedatana.

Myös vapaaseen lähdekoodiin perustuvien projektien sähköpostilistoilla käydään jatkuvaa keskustelua Netflow-datan implementoimisesta eri järjestelmiin. Esimerkiksi Bro-sovelluksen kehittäjät ovat ilmoittaneet lisäävänsä tulevaisuudessa tuen Netflow-datalle.

Netflow-dataa voidaan käyttää myös valvomaan palvelimia, kuten luvussa 6.3.2 kuvattiin. Saastunut palvelin huomataan helposti, jos sen käyttämättömissä porteissa liikkuu dataa.

6.7 Työkalut

Netflow-datan käsittelyyn on tehty useita avoimeen lähdekoodiin perustuvia sekä kaupallisia työkaluja. Yleisesti käytetty vapaan lähdekoodin Netflow-työkalu on flow-tools[5], joka sisältää 24 työkalua Netflow-tietojen käsittelyyn.

6.7.1 Flow-tools

Flow-tools on joukko Mark Fullmerin Unixille kirjoittamia apuohjelmia Netflow-datan käsittelyyn. Flow-tools sisältää useita pieniä ohjelmia, jotka esimerkiksi vastaanottavat reitittimiltä tulevaa Netflow-dataa, suodattavat sitä ja tekevät niistä raportteja. Flow-tools työkaluja käytetään komentoriviltä putkittamalla niiden syötteitä toisilleen. Esimerkiksi tiedostoon tallennetun Netflow-data voidaan tulostaa komennolla:

```
$flow-cat [tiedosto] | flow-print
```

Tässä *flow-cat* lukee ensin sille annetun tiedoston, lukee sen ja välittää tiedot *flow-print*-ohjelmalle, joka tulostaa vastaanottamansa Netflow-tietueet.

[5]http://www.splintered.net/sw/flow-tools/

Putkittamalla esimerkiksi erilaisia suodattimia (engl. *Filter*) voidaan luoda monimutkaisia suodatussääntöjä, joiden mukaan liikennettä suodatetaan ja vain halutut vuot saadaan näkyviin. *Flow-stat*-ohjelmalla voidaan luoda erilaisia raportteja esimerkiksi IP-osoiteparien, porttien, pakettien tai bittien mukaan. Esimerkiksi komennolla

```
$flow-cat [tiedosto] | flow-stat -f10 -S3
```

`f10`-tarkentimella tarkoittaa raporttia lähde- ja kohdeosoitepareista, joiden välillä on liikkunut eniten paketteja. Erilaisia raportteja on yli 30 kappaletta, joiden avulla raportit voidaan luoda lähes kaikkien Netflow-vuon kenttien pohjalta. `S3`-tarkentimella lajitellaan tiedot sarakkeen 3 mukaan alenevaan järjestykseen. Yleneävä järjestys saadaan tarkentimella `-s`.

```
#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Disabled
# Sorting:   Descending Field 3
# Name:      Source/Destination IP
#
# Args:      flow-stat -f10 -S3
#
#
# src IPaddr  dst IPaddr     flows  octets    packets
#
192.168.1.36  192.168.1.33   1678   3971302   6751
192.168.1.33  192.168.1.36   1678   808201    6712
192.168.1.1   192.168.1.255  1      288       4
```

Flow-dscan-ohjelma etsii Netflow-datasta porttiskannauksia ja ilmoittaa havaitsemistaan skannauksista. Ohjelmaan voidaan asettaa, kuinka monen portin tutkimista pidetään skannauksena.

```
$flow-cat [tiedosto] |flow-dscan -b -m -W
flow-dscan: port scan: src=192.168.1.33 dst=192.168.1.36
  ts=3554985922 start=0721.11:37:06.920
```

Esimerkissä `b`-tarkennin ajaa ohjelman vain kerran ja estää ohjelman jäämisen muistiin ja suorittamasta reaaliaikaista porttiskannausten etsimistä. Multicast osoitteet suodatetaan `m`-tarkentimella, jolloin niitä ei oteta mukaan analysointiin. Samoin `W`-tarkentimella suodatetaan pois WWW-liikenne. Esimerkissä *flow-dscan*-ohjelma löytää porttiskannauksen, joka on lähtöisin osoitteesta 192.168.1.33 ja kohdistuu osoitteeseen 192.168.1.36.

Tutkimalla edelleen, voidaan laskea kuinka moneen eri porttiin skannauksia on kohdistunut. Ajamalla *flow-stat*, joka raportoi kaikki TCP- tai UDP-kohdeportit (`f 5`-tarkennin), joihin on otettu yhteyttä. Komennolla `wc -l` voidaan laskea raportin rivien määrä ja näin saadaan skannattujen porttien lukumäärä:

```
$flow-cat testflow | flow-filter -S infected_host | \
 flow-stat -f 5 | wc -l
1637
```

Esimerkissä saadaan tulokseksi 1637 eri porttia, joihin skannaus on todennäköisesti kohdistunut. Tämä viittaa selvästi systemaattiseen porttien skannaamiseen. Edellisessä esimerkissä `infected_host`-muuttujalla viitataan seuraavaan riviin `flow.acl`-tiedostossa, jonka avulla *flow-filter* pystyy suodattamaan vain ne tietueet, joissa lähdeosoitteena on ollut IP-osoite `192.168.1.33`:

```
ip access-list standard infected_host permit host 192.168.1.33
```

Flow-nfilter-ohjelmalla voidaan suodattaa Netflow-dataa ennalta kirjoitettujen suodattimien mukaan. Suodattimista voidaan tehdä hyvinkin monimutkaisia ja *flow-nfilter*-ohjelman manuaalisivuilla [53] on esitetty kaikki suodattimissa käytettävät komennot. Seuraavalla suodattimella, joka on määritetty `slammer-filter`-tiedostoon, poimitaan ne tietueet, joissa kohdeporttiin 1434 on lähetetty 404 oktettia dataa. Portti 1434 on Microsoftin SQL-Serverin käyttämä portti, jossa havaittiin vuonna 2003 tietoturva-aukko. SQL-Slammer-mato pyrkii saastuttamaan koneita lähettämällä porttiin haittakoodia.

```
filter-primitive port1434
       type ip-port
       permit 1434
       default deny

filter-primitive ProtoUDP
       type ip-protocol
```

```
        permit udp
        default deny

filter-primitive wormsize
        type counter
        permit eq 404
        default deny

filter-definition SQL-Slammer
        match ip-protocol ProtoUDP
        match ip-destination-port port1434
        match octets wormsize
```

Suodatin koostuu primitiiveistä, joiden avulla voidaan määritellä esimerkiksi IP-osoitteita ja portteja. Suodatin tiedostoon voidaan määritellä useita erilaisia suodattimia, jotka erotellaan uniikeilla nimillä toisistaan. Suodatin ajetaan *flow-nfilter*-ohjelmalla, jolloin se poimii tietueet, jotka toteuttavat suodattimen säännöt. *Flow-nfilter*-ohjelmalle määritetään tiedosto `f`-tarkentimella. Tiedosto voi sisältää useita suodattimia, joten käytettävän suodattimen nimi ilmoitetaan `F`-tarkentimen avulla.

```
$flow-cat [tiedosto] | flow-nfilter -f slammer_filter
 -F SQL-Slammer | flow-print

srcIP        dstIP        prot srcPort dstPort octets packets
192.168.1.33 192.168.1.36 17   1796    1434    404    1
```

Esimerkissä Netflow-data luetaan ensin *flow-cat*-ohjelmalla, jonka jälkeen vuot annetaan *flow-nfilter*-ohjelman suodatettavaksi. Suodatuksen jälkeen jäljelle jääneet vuot tulostetaan *flow-print*-ohjelmalla. Tulosteesta nähdään, että suodatuksen jälkeen on jäänyt yksi vuo, joka täyttää edellä kirjoitetut säännöt ja näin ollen lähdeosoitteesta 192.168.1.33 on lähetetty paketti kohdeosoitteeseen 192.168.1.36, joka vastaa SQL-Slammer-madon toimintaa.

6.7.2 Flowd

Flowd[6] on Damien Millerin kirjoittama ohjelma Netflow-tietojen keräämiseen ja tallentamiseen levylle. Ohjelman tarkoituksena on vain tallentaa reitittimiltä tai muilta

[6] `http://www.mindrot.org/projects/flowd/`

lähteiltä tulevaa dataa, eikä se sisällä minkäänlaisia analysointityökaluja. Tältä osin *Flowd* noudattaa normaalia Unixin ohjelmasuunnittelua, jossa yksi ohjelma tekee yhden asian hyvin, jättäen ylimääräiset ominaisuudet muille ohjelmille. [54]

Flowd tarjoaa C-, Perl- ja Python-ohjelmointikielien avulla käytettävät rajapinnat ohjelman tallentamien tiedostojen käsittelyyn, joten omien analysointiohjelmien kirjoittaminen on helppoa Flowd:n avulla. *Flowd* kykenee tallentamaan Netflown versioita 1, 5, 7 ja 9, mukaan lukien IPv6-vuot. Tästä syystä se on hieman monikäyttöisempi kuin flow-tools-ohjelmisto. Suurin hyöty *Flowd*:n käytöstä on, että se tallentaa enemmän reititystietoja Netflow-datasta kuin flow-tools. Esimerkiksi tietue seuraavasta laitteesta, johon vuo ohjataan, saadaan vain *Flowd*:n avulla. [54]

Flowd:n konfiguroitavuus on hyvä ja konfiguraatiotiedoston avulla voidaan helposti hallita mistä laitteista Netflow-dataa hyväksytään ja minne se tallennetaan. Esimerkiksi seuraavassa konfiguraatiossa on määritetty *Flowd* hyväksymään tietyn IP-osoitteen tietystä portista tuleva Netflow-data sekä kaikista IP-osoitteista, jotka loppuvat numeroon yksi: [54]

```
# Specify what addresses/ports flowd should listen on.
# Multiple addresses may be specified
listen on 127.0.0.1:12345
listen on [::1]:12345
```

Konfiguraatiotiedostossa voidaan myös määritellä mitkä Netflow-datan tietueet tallennetaan, joten ylimääräiset tietueet voidaan karsia pois ja näin voidaan säästää levytilaa. Esimerkiksi seuraavassa konfiguraatiossa tallennetaan vain vuossa havaitut liput, vuon lähettäneen laitteen IP-osoite, sekä vuon lähde- ja kohdeosoitteet: [54]

```
# Specify which flow records are recorded in the log.
# Multiple options may be specified.
store PROTO_FLAGS_TOS
store AGENT_ADDR
store SRC_ADDR
store DST_ADDR
```

Konfiguraatiotiedostossa voidaan myös määritellä erilaisia suodattimia (engl. *Filters*) ja voiden merkitsemistä (engl. *Tagging*). Suodattimien avulla voidaan määritellä mitkä vuot hyväksytään, esimerkiksi tietyltä aikaväliltä. Samalla vuot voidaan merkitä ennalta määritetyllä tavalla, jolloin esimerkiksi tietyltä reitittimeltä

tulevat vuot voidaan merkitä ja niiden löytäminen jatkossa on helpompaa. Esimerkissä on määritelty *Flowd* hylkäämään TCP-vuot tietystä aliverkosta, sekä hyväksymään määritellyistä lähde- ja kohdeosoitteista tuleva liikenne, joka samalla merkitään numerolla yksi: [54]

```
# Example of a filter policy using local variables
discard agent 1.1.1.0/24 proto tcp
accept tag 1 src 2.2.2.0/25 port 666 dst 33.33.0.0/16 port 888
```

7 Sisäisiin uhkiin reagoiminen

Nykyään verkon ylläpitäjien tulee varautua myös verkon sisäisten uhkien varalle. Usein esimerkiksi uudet matoepidemiat pääsevät saastuttamaan verkon laitteita ennen kuin ne voidaan havaita ja estää erilaisilla tietoturvalaitteilla. Madon saastuttaman laitteen liikennöinti verkkoon tulisi estää mahdollisimman nopeasti, jotta leviäminen saataisiin hallintaan ja mato voidaan torjua. Madot ja hyökkäykset tapahtuvat harvoin työaikana, jolloin verkon ylläpitäjä voi nopeasti reagoida uhkiin, ja tällöin automaattinen reagointi on paikallaan.

Verkon reitittimiä voidaan konfiguroida etäyhteyden yli, jolloin laitteiden liikennöinti voidaan estää kohtuullisen nopeasti, mutta liikenteen estäminen tulisi tapahtua automaattisesti, jotta tärkeää aikaa ei kuluisi hukkaan. Eston tekeminen käsin reitittimiin voi kestää jopa useita päiviä, jos toimenpide vaatii useiden eri henkilöiden työpanoksen. Tästä syystä automaattinen estäminen on tehokkain tapa toteuttaa leviämisen estäminen.

Saman yhtiön laitteet pystyvät yleensä kommunikoimaan keskenään ja tekemään muutoksia toistensa konfiguraatioihin automaattisesti. Esimerkiksi Ciscon IDS-laitteet voivat keskustella Cisco PIX-palomuurin kanssa ja näin estää hyökkääjän yhteydet palomuurissa. Myös useimpia Ciscon reitittimiä voidaan konfiguroida muiden laitteiden toimesta. Kommunikointi tapahtuu yleensä Telnetin tai SSH:n välityksellä, jolloin laite kirjautuu normaalisti konfiguroitavan laitteen hallintaliittymään. Tällöin konfigurointiin käytettävän tunnuksen salasana tulee tietää ja tunnuksella pitää olla oikeudet muuttaa konfiguraatiota. [43]

Reitittimien automaattiseen konfigurointiin voidaan käyttää erilaisia skriptejä tai SNMP-protokollaa, jos verkon laitteet tukevat sitä, mutta tällöin ongelmana on usein yhteisen standardin puute, joka estää erilaisten laitteiden konfiguroinnin ja pakottaa tekemään eri skriptit eri laitteille. Usein suuret verkon koostuvat eri valmistajien laitteista, jotka sisältävät erilaiset ohjelmistot ja konfigurointitavat.

Laitteiden liikenteen estämisessä tulee kuitenkin olla tarkkana, jottei jonkin tärkeän palvelimen liikennöintiä estetä vahingossa. Tästä syystä tärkeistä palvelimista tulisi pitää listaa, josta tarkastetaan eston yhteydessä, ettei kohde ole listalla. Laitteen liikennöinnin esto tulee purkaa tietyn ajan kuluttua, jotta reitittimiin ei jää turhia sääntöjä. Estot voivat johtua myös vääristä hälytyksistä (engl. *False positive*), jolloin liikenteen estolla voidaan estää normaalin käyttäjän liikennöinti verkkoon. Jos

estot eivät vanhenisi koskaan, voisi reitittimen ACL-lista kasvaa niin suureksi, että se haittaa reitittimen toimintaa. Normaali rajoitusaika on noin 30 minuuttia. [43]

Luvussa esitellään erilaisia tapoja sisäisten uhkien torjumiseen ja jäljittämiseen. Alussa esitellään Jyväskylän yliopiston tietoliikennelaboratorion Laila-verkon topologia ja tämän jälkeen käydään läpi Ciscon reitittimien ACL-pääsylistat ja niiden konfigurointi. Luvussa käsitellään saastuneen laitteen etsimistä Netflow-tietojen avulla sekä siihen liittyviä ongelmia, kuten laitteet, joilta ei saada Netflow-dataa. Lopuksi esitellään WTS Networksin kehittämä Netwrapper sekä sen käyttö saastuneiden laitteiden liikenteen estämiseen yhdessä Netflown kanssa.

7.1 Cisco ACL

Cisco käyttää laitteissaan erityyppisiä pääsylistoja (engl. *Access Control List, ACL*): vakio IP-pääsylista (engl. *Standard IP Access List*), laajennettu IP-pääsylista (engl. *Extended IP Access List*) sekä laajennettu MAC-pääsylista (engl. *MAC Extended Access List*). Vakiolistaa käytetään kun halutaan estää tietyn IP-osoitteen tai verkon liikennöinti laitteen kautta. Vakiolistan syntaksi on seuraava: [43]

```
access-list access-list-number {deny | permit}
  {source source-wildcard| host source | any}
```

`Access-list-number` on kokonaisluku väliltä 1 – 99 tai 1300 – 1999 ja se määrittää mihin pääsylistaryhmään sääntö kuuluu. `Deny` kieltää ja `permit` sallii sääntöä vastaavan liikenteen käsittelemisen verkon laitteessa. `Source` tarkoittaa lähdeosoitetta, josta paketti on lähetetty ja se ilmoitetaan 32-bittisenä neljän luvun sarjana, jotka erotetaan toisistaan pisteillä. `Source-wildcard` määrää lähdeosoitteen maskin, jolloin sääntö voi vastata useaa IP-osoitetta kerralla. `Host` määrää yksittäisen lähdeosoitteen ilman maskia ja `any` vastaa kaikkia lähdeosoitteita. Esimerkkinä seuraava sääntö, joka sallii liikenteen kaikista 172.16.2-alkuisista osoitteista:

```
access-list 19 permit 172.16.2.0 0.0.0.255
```

Laajennettu pääsylista antaa enemmän vaihtoehtoja, kuten käytetyn protokollan mukaan tehtävä liikenteen kontrollointi. Laajennetun listan syntaksi on:

```
access-list access-list-number {deny | permit} protocol
 {source source-wildcard | host source | any} [operator port]
 {destination destination-wildcard | host destination | any}
 [operator port]
```

`Protocol` määrittelee tarkastettavan IP-protokollan nimen tai numeron. Protokollaksi voidaan määritellä `ip`, jos halutaan säännön vastaavan kaikkia IP-protokollia. `Operator` määrittelee operaattorin, jonka mukaan esimerkiksi porttinumeroita tarkastetaan. Mahdollisia operaattoreita on neljä:

eq yhtä suuri kuin - kun tiedetään tarkalleen mitä porttia tarkastellaan.

gt suurempi kuin - voidaan tarkkailla suurempia portteja kuin annettu.

lt pienempi kuin - voidaan tarkkailla pienempiä portteja kuin annettu.

neq erisuuri kuin - tarkkaillaan kaikkia muita paitsi annettua porttia.

`Port` on porttinumero ja se ilmoitetaan kokonaislukuna väliltä 0 – 65535. Portti voidaan myös ilmoittaa nimeltä. Porttinumeroa verrataan lähde- tai kohdeporttiin riippuen siitä, onko se määritelty säännössä lähde- vai kohdeosoitteen jälkeen. `Destination` määrittelee paketin kohdeosoitteen ja se toimii samalla tavalla kuin lähdeosoite. Esimerkkinä sääntö, joka estää TCP-liikenteen lähdeosoitteista, jotka alkavat 172.16.1, ja jonka kohdeosoite alkaa 172.16.2, kohdeportin ollessa jokin muu kuin 23:

```
access-list 190 deny TCP 172.16.1.0 0.0.0.255
  172.16.2.0 0.0.0.255 neq 23
```

Laajennetun MAC-pääsylistan avulla kontrolloidaan liikennettä lähde- ja kohde-MAC-osoitteiden avulla, sekä valinnaisen protokollan tietojen mukaan. Pääsylistan syntaksi on:

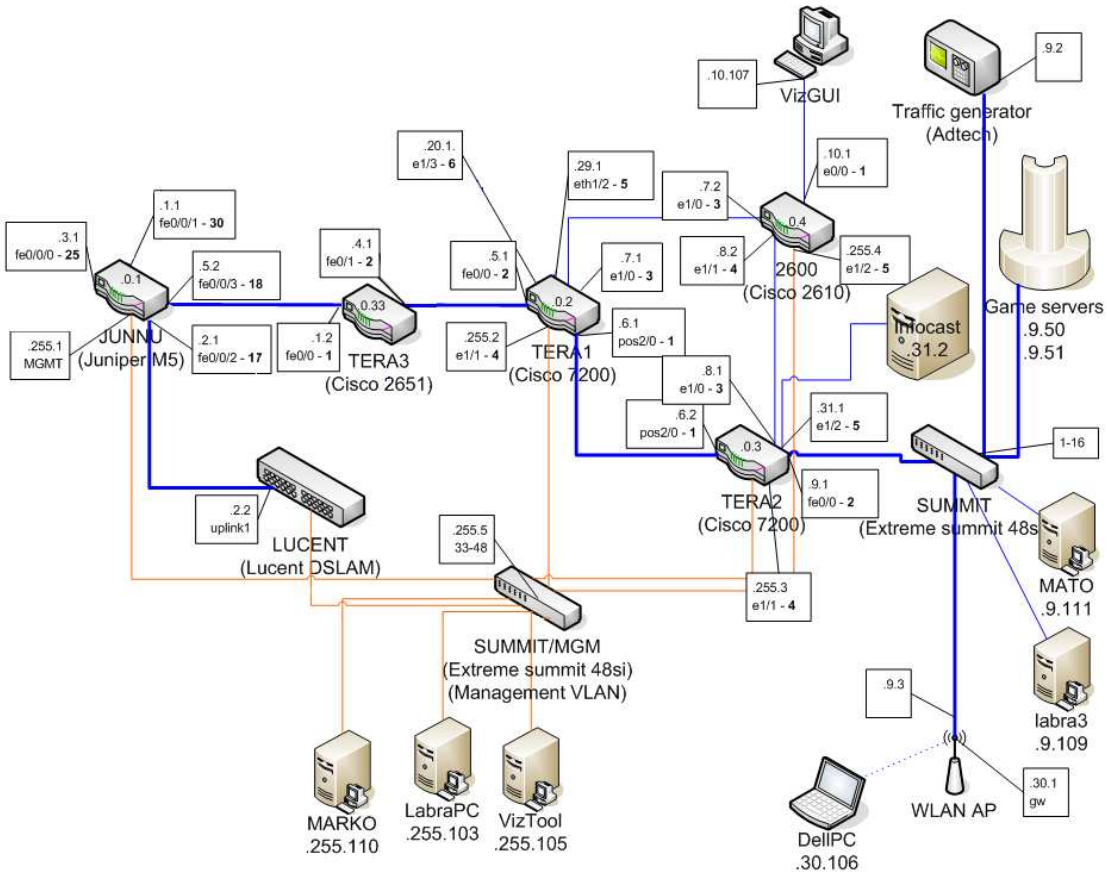
```
{deny | permit} {any | host source MAC address}
  {any | host destination MAC address} [aarp | amber |
  appletalk | dec-spanning | decnet-iv | diagnostic | dsm |
  etype-6000 | etype-8042 | lat | lavc-sca | mop-console |
  mop-dump | msdos | mumps | netbios | vines-echo |vines-ip |
  xns-idp]
```

MAC-pääsylistan avulla voidaan sallia tai estää tietystä MAC-osoitteesta tuleva ja tiettyyn MAC-osoitteeseen kohdistuva liikenne. Loput määreet ovat valinnaisia vaihtoehtoja, joiden tarkoitusta ei käsitellä tässä.

Pääsylistat toimivat vain Ciscon laitteissa, sillä muilla valmistajilla on erilaiset pääsylistasyntaksit. Yleisesti listat toimivat samalla periaatteella kaikkien valmistajien laitteissa.

7.2 Laila-verkon topologia

Käytännönosuuden testeissä on käytetty Jyväskylän yliopiston tietoliikennelaboratorion Laila-testiverkkoa (kuva 7.1), joka koostuu viidestä reitittimestä (Junnu, Tera1-3 ja Cisco 2600) sekä kolmesta kytkimestä, joista yhtä käytetään hallintaverkon toteutukseen (SUMMIT/MGM). Reitittimistä Tera 1-3 sekä Cisco 2600 lähettävät Netflow-dataa, mutta Juniper ei. Testeissä on käytetty kahta laitetta, joista toinen esitti saastunutta laitetta (MATO) ja toinen keräsi Netflow-dataa (MARKO). Kuvaan on merkitty kaikkien reitittimien liitynnät, jotta Netflown avulla tehtävää jäljittämistä voidaan helposti seurata kuvasta.



Kuva 7.1: Laila-verkko, topologia 1.

Verkon topologiaa muutettiin testien aikana niin, että Tera 3 -reititin siirrettiin Juniper-reitittimen taakse, jolloin voitiin testata miten Netflow-jäljitys käyttäytyy kun yksi verkon laitteista ei lähetä Netflow-dataa (Juniper).

Viimeisessä testissä verkkoon luotiin väärennettyä liikennettä jäljityksen testaamiseksi. Tällöin saastunut laite (MATO) siirrettiin Cisco 2600 -reitittimen taakse.

Väärennettyä liikennettä luotiin Adtech traffic generator -laitteella.

7.3 Hyökkäyksen lähteen selvittäminen

Kun mato on saastuttanut verkon laitteen ja alkaa levitä verkossa, tulisi se havaita ja paikallistaa nopeasti. Ongelmaksi voi muodostua isoissa verkoissa, jotka koostuvat useista kymmenistä kytkimistä ja reitittimistä, saastuneen laitteen paikallistaminen, varsinkin jos se käyttää väärennettyä lähdeosoitetta.

Jos mato, joka käyttää väärennettyä lähdeosoitetta, havaitaan esimerkiksi verkon runkoreitittimeen kytketyn IDS-järjestelmän avulla, voi madon todellisen alkuperän selvittäminen olla todella hankalaa. Tässä tapauksessa apuna voidaan käyttää Netflow-dataa, sillä sitä voidaan kerätä useista reitittimistä, ja näin tietty vuo voidaan jäljittää. Väärennetyn lähdeosoitteen tapauksessa jäljitys voidaan saattaa viimeisimpään Netflow-dataa lähettävään verkon laitteeseen ja sen tiettyyn liitäntään.

Netflow-datan avulla voidaan helposti selvittää verkon laitteet, joiden kautta yhteys on kulkenut. Ongelmaksi muodostuu ensimmäisen laitteen löytäminen, joka on välittänyt kyseistä yhteyttä. Koska Netflow-data lähetetään reitittimiltä tietyin väliajoin, kun tarpeeksi monta vuota on saatu kasaan, ei datan saapumisajankohdasta voida päätellä ensimmäistä reititintä. Eri reitittimiltä tuleva Netflow-data kulkee verkossa eripituisia aikoja, joka osaltaan vääristää saapumisajoista päättelyä. Netflown versio yhdeksän sisältää *min-ttl*- ja *max-ttl*-kentät, jotka ilmoittavat pienimmän ja suurimman TTL-arvon, jotka on havaittu yhdessä vuossa. Näiden avulla olisi mahdollista päätellä ensimmäinen laite, jos tietty verkon laite asettaa aina saman TTL-arvon jokaiseen lähettämäänsä pakettiin. Toistaiseksi vain uusimmat Ciscon laitteet tukevat Netflown versiota yhdeksän, joten TTL:n perusteella tehtävä päättely vaatii, että kaikki verkon reitittimet lähettävät version yhdeksän mukaista Netflow-dataa.

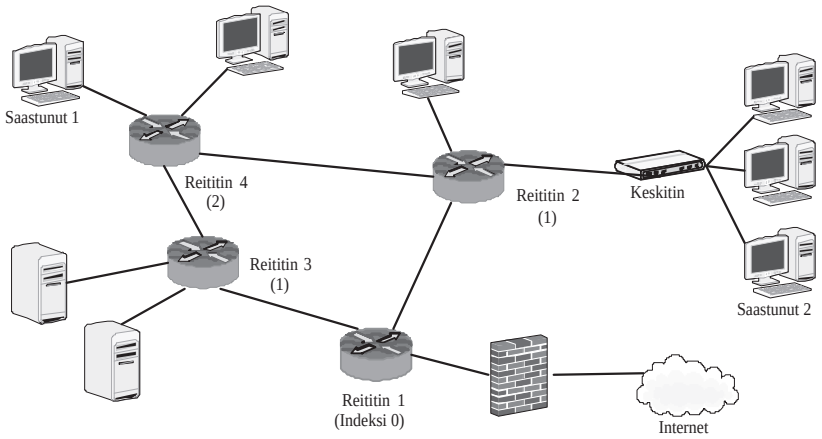
Luvussa 6.2 kuvattiin Netflow-datan sisältö. jäljittämisen kannalta olennaisia kenttiä ovat vuon lähde- ja kohdeosoite sekä portit. Tämän lisäksi Netflow-datasta saadaan sen lähettäneen laitteen (reitittimen) IP-osoite, sekä seuraavan laitteen IP-osoite, johon vuo ohjattiin. Netflow-data koostuu riveistä, joista jokainen kuvaa yhtä vuota. Sama vuo voi esiintyä usealla rivillä riippuen siitä, kuinka monta kertaa reititin on lähettänyt sen tietoja. Vuon lähde- ja kohdeosoitteet ovat aina pakettiin merkityt lähde- ja kohdeosoitteet, joten tietyn yhteyden osoitteet ovat samat, riippumatta sen reitistä verkossa.

Netflow-datan avulla tapahtuva jäljittäminen vaatii verkon topologian tuntemisen, jotta voidaan sanoa, mikä reititin on ensimmäisenä käsitellyt yhteyttä. Tällöin

voidaan toteuttaa erilaisia ratkaisuja ensimmäisen käsittelijän hakemiseen.

7.3.1 Indeksiin perustuva jäljitys

Eräs ratkaisu on indeksoida verkon laitteet niin, että runkoreitittimellä olisi pienin indeksi. Indeksi kasvaisi verkon laitteilla sitä mukaan, kun laitteet sijaitsevat etäämpänä runkoreitittimestä. Tietyn vuon todennäköisimmät lähteet olisivat laitteet, joilla on suurimmat indeksit. Kuvassa 7.2 on esitetty laitteiden indeksointi. Oletetaan, että kuvaan merkityt kaksi laitetta ovat saastuneet ja lähettävät leviämisviestejä runkoreitittimen kautta Internettiin. Jokainen reititin lähettää Netflow-dataa, joka analysoidaan.



Kuva 7.2: Indeksiin perustuva topologia

Saastuneen koneen yksi tapauksessa Netflow-data voi näyttää esimerkiksi seuraavalta (kenttiä karsittu esimerkissä):

Lähde	Kohde	Reititin
192.168.1.110:6456	130.224.134.1:445	Reititin 4
192.168.1.110:6456	130.224.134.1:445	Reititin 1
192.168.1.110:6456	130.224.134.1:445	Reititin 3

Esimerkissä lähdeosoitteesta 192.168.1.110 lähetetään liikennettä Internettiin osoitteeseen 130.244.134.1 ja yhteyden ovat havainneet reitittimet 1, 3 ja 4. Koska reitittimellä 4 on suurin indeksi, voidaan pitää todennäköisenä, että lähde sijaitsee reitittimen 4 takana, vaikka IP-osoite olisi väärennetty. Toisen saastuneen koneen tapauksessa tilannetta hankaloittaa keskitin, jota ei voida konfiguroida. Reititin 2:een voidaan asettaa ACL-sääntö, joka estää liikenteen tietystä IP-osoitteesta, mutta se ei

auta, jos lähdeosoite on väärennetty. Reitittimet voidaan konfiguroida niin, että ne tarkistavat lähdeosoitteen oikeellisuuden, mutta oletamme, ettei näin ole tehty. Toinen vaihtoehto on sulkea koko reitittimen portti, johon keskitin on kytketty. Tällöin kaikilta keskittimeen kytketyiltä laitteilta estetään liikennöinti.

Käyttämällä *Flowd*-ohjelmaa sekä Python-ohjelmointikieltä, voidaan edellä mainittu jäljitysalgoritmi kirjoittaa ohjelmaksi. Ohjelma on kirjoitettu käyttämällä Pythonin versiota 2.4, sekä *Flowd*:n versiota 0.9.

Seuraavassa esimerkissä on Python-ohjelmointikielellä toteutettu funktio, joka käy läpi Netflow-datan tietueet ja ilmoittaa suurimmalla indeksillä löytyneen reitittimen osoitteen. Funktio ottaa parametreiksi *flows*, joka sisältää kaikki Netflow-tietueet sekä *ip*, joka määrittää jäljitettävän IP-osoitteen.

Jäljitysfunktion tulee tuntea reitittimien indeksit, joten konfigurointitiedostossa määritellään verkon reitittimien IP-osoitteet ja niille annetut indeksit. Täydellinen konfiguraatiotiedosto liitteenä (liite A). Testeissä käytetyssä ohjelmassa verkon laitteet oli määritelty pysyvästi osaksi ohjelmaa, normaalisti tiedot verkon laitteista olisivat esimerkiksi tietokannassa, jossa niiden hallinta olisi helpompaa. Tällöin jäljitysohjelma hakisi laitteen tiedot tarpeen mukaan tietokannasta.

```
devices = {
        "172.16.0.2": {
                "name": "Tera-1",
                "index": 0,

                },
        "172.16.0.3": {
                "name": "Tera-2",
                "index": 1,
                },
}
```

Tämän jälkeen voidaan käyttää `traceIpByIndex`-funktiota. Funktiossa esiintyvä `filterFlows`-funktio suodattaa vuot annettujen sääntöjen perusteella. Tässä vuot suodatetaan lähdeosoitteen perusteella, joka on annettu funktiolle parametrina. `GetDeviceIndex`-funktio hakee aiemmin määritellystä `devices`-sanakirjasta (engl. *Dictionary*) laitteen IP-osoitetta vastaavan indeksin. Funktio palauttaa listan reitittimien IP-osoite- ja liityntäpareista, jotka ovat käsitelleen ensimmäisinä voita. Täydellinen lähdekooditiedosto löytyy liitteestä B.

```
def traceIpByIndex(flows, ip):
```

```
"""Traces first router by index"""

possible_sources = []
dFilter = {"src_addr": [ip]}
filtered = filterFlows(flows, dFilter)
hindex = -1

for flow in filtered:
  if getDeviceIndex(flow.agent_addr) > hindex:
    hindex = getDeviceIndex(flow.agent_addr)
    possible_sources.append((hindex, flow.agent_addr,
      flow.if_ndx_in))
    print flow.agent_addr, hindex

res = []

# Filter out routers that have smaller index
for (indx, src, ifndx) in possible_sources:
  if indx == hindex: res.append((src, ifndx))

return res
```

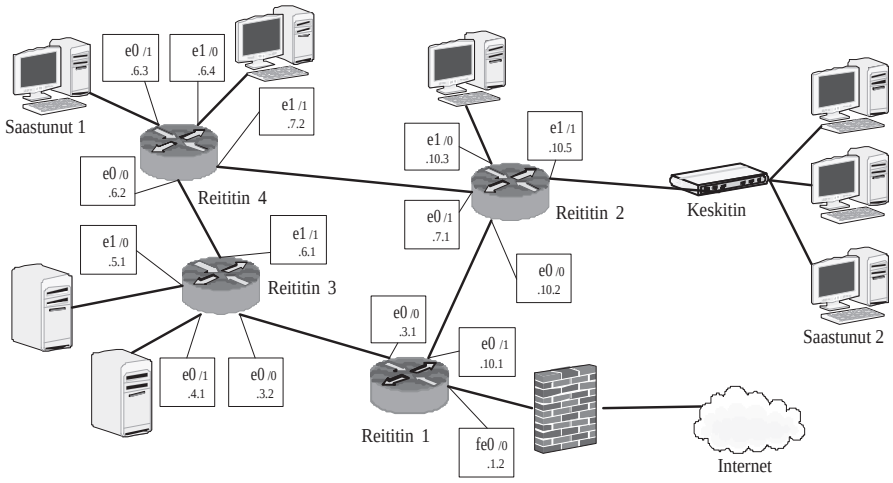
7.3.2 Reitittimien tietoihin perustuva jäljitys

Verkon topologian tietoja laajentamalla verkon reitittimien osoitteilla, sekä niiden liityntöjen osoitteilla, voidaan paremmin jäljittää ensimmäinen reititin, joka on käsitellyt vuota (kuva 7.3). Reitittimen lähettämässä vuon tiedoista ilmenee seuraavan verkkoliitynnän osoite, johon vuo on ohjattu. Reitittimien tapauksessa tämä on reitittimen tietyn liitynnän IP-osoite. Vuon tiedoista ilmenee myös liityntä, josta vuo on tullut reitittimelle sekä liityntä, josta se on lähtenyt reitittimeltä.

Näillä tiedoilla voidaan jäljittää vuon reitti verkossa seuraavan algoritmin mukaan (kuva 7.4):

Algoritmi on toteutettu käytännössä käyttämällä Python ohjelmointikieltä. Funktio tarvitsee tiedot reitittimien liitynnöistä, joten edellisen koodiesimerkin sanakirjaa on laajennettu näiltä osin.

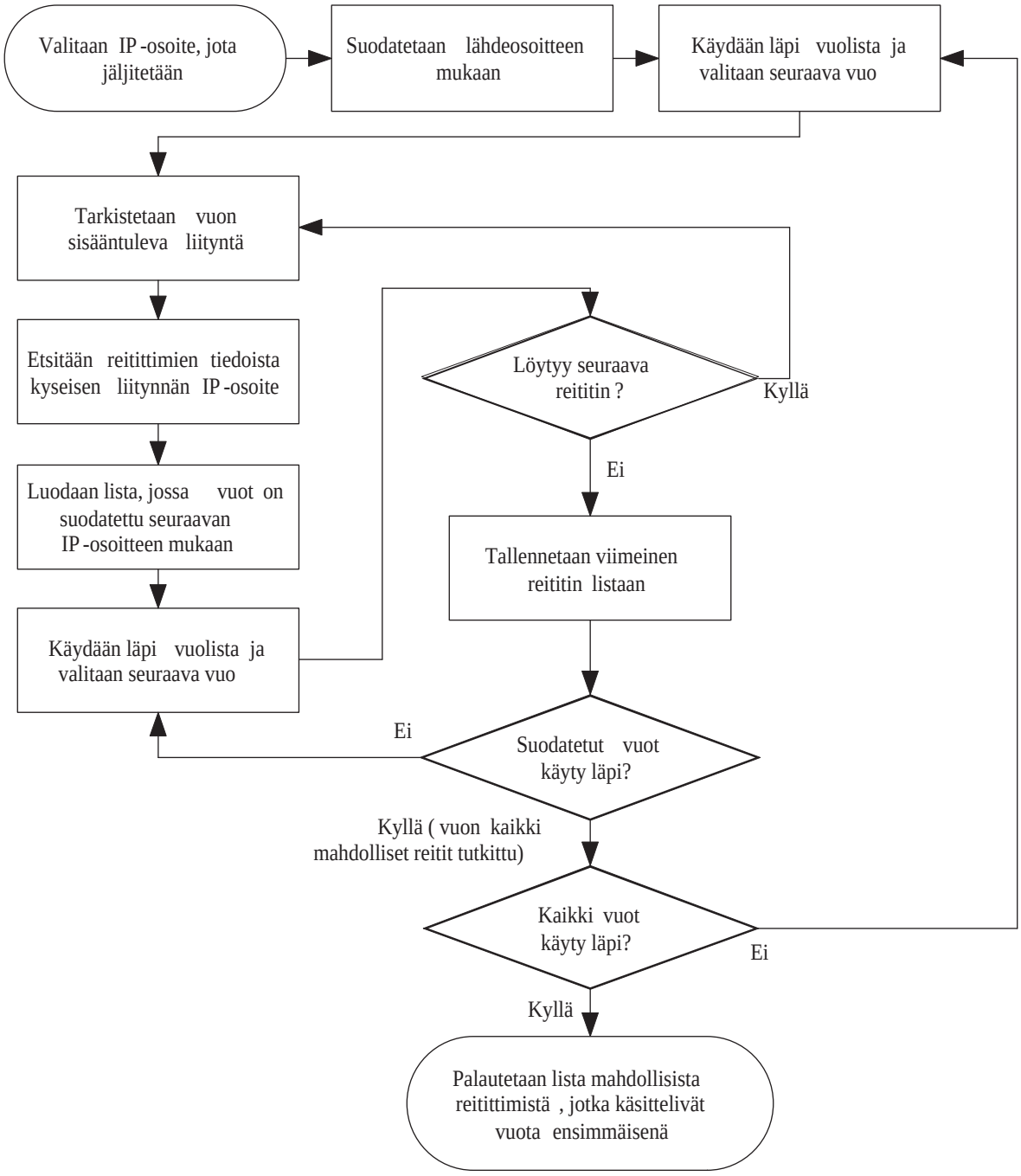
Kuvan 7.5 reitittimen tiedot on esitetty Pythonin sanakirja-tietotyyppinä, josta reitittimen tiedot saadaan nopeasti reitittimen IP-osoitteen perusteella. `Name`-tietueet helpottavat Netflow-tietojen lukua erilaisissa listauksissa.



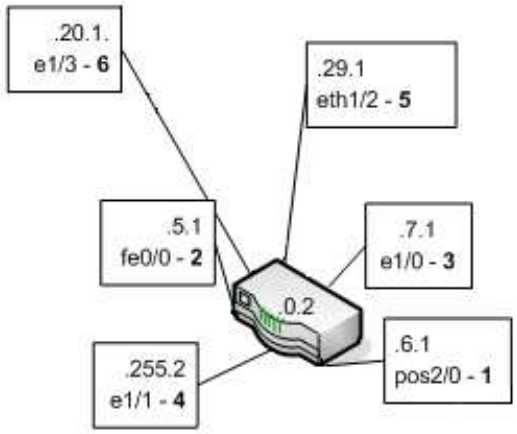
Kuva 7.3: Liityntöjen tietoihin perustuva topologia

```
devices = {
  "172.16.0.2": {
    "name": "Tera-1",
    "interfaces": {
      0: {"name": "-", "ip": None},  # loopbacks
      1: {"name": "pos2/0", "ip": "172.16.6.1"},
      2: {"name": "fe0/0", "ip": "172.16.5.1"},
      3: {"name": "e1/0", "ip": "172.16.7.1"},
      4: {"name": "e1/1", "ip": "172.16.255.2"},
      5: {"name": "e1/2", "ip": "172.16.29.1"},
      6: {"name": "e1/3", "ip": "172.16.20.1"}
     }
   },
}
```

`TraceIpByGateway`-funktio ottaa kaksi parametria, joista ensimmäisessä on Netflow-tietueet, jotka halutaan käydä läpi. Toinen on jäljitettävän laitteen IP-osoite. Funktio suodattaa ensin kaikki tietueet pois, joiden lähdeosoite ei ole haettava IP-osoite. Tämän jälkeen käydään läpi jokainen vuo yksitellen vuolistasta. Ensin etsitään verkon topologiatiedoista vuon käsitelleen reitittimen sisääntulevan liityn-nän IP-osoite `getIfIpByDevice`-funktion avulla. `TraceGatewayIp`-funktio suorittaa vuon polun etsimisen Netflow-tietueiden perusteella ja palauttaa listan IP-osoitteista, jotka ovat käsitelleen viimeisenä vuota. Algoritmia voidaan tehostaa suurilla tietuemäärillä etsimällä tietyn vuon polku vain kerran. Yleensä vuo kulkee samaa polkua verkossa, joten polun etsiminen voidaan tehdä vain kerran. Jos algo-



Kuva 7.4: Jäljitys vuokaaviona.



Kuva 7.5: Tera1-reitittimen liitynnät ja niiden osoitteet

ritmi palauttaa useita eri reitittimiä ja jäljitys on tehty vain kerran, ei voida sanoa minkä reitittimen kautta liikenne on useimmiten kulkenut. Täydellinen lähdekooditiedosto löytyy liitteestä B.

```
def traceIpByGateway(flows, ip):
  """Traces first router by routing information and interface ips"""

  possible_sources = []
  route = []
  processed_gws = []

  dFilter = {"src_addr": [ip] }

  # Filter flows that have given source IP
  filtered = filterFlows(flows, dFilter)

  # Iterate through all flows after filtering
  for flow in filtered:

    if_ip = getIfIpByDevice(flow.agent_addr, flow.if_ndx_in)

    if trace_gws_only_once:
      if not if_ip in processed_gws:
        possible_sources.extend(traceGatewayIp(filtered, flow, if_ip))
        processed_gws.append(if_ip)
    else:
```

```
        possible_sources.extend(traceGatewayIp(filtered, flow, if_ip))

    return possible_sources
```

Lopuksi ohjelma palauttaa listan reitittimistä, jotka ovat käsitelleet ensimmäisenä tietystä lähdeosoitteesta tulevaa vuota. Jos jokainen vuo on jäljitetty loppuun asti, näyttää ohjelma prosentuaalisen osuuden jokaisen reitittimen käsittelemästä vuomäärästä. Tästä voidaan yrittää päätellä ensimmäinen reititin, jos ohjelma palauttaa useita reitittimiä.

Esimerkki jäljityksen tuloksesta:

```
Loading file /var/log/flows/flowd.kopio1...
Tracing ip 172.16.0.33 ...

Possible sources:

Source                    Interface  Count      %  Unique ips
172.16.0.2   (Tera-1)     fe0/0 (2)     15  88.24           4
172.16.0.3   (Tera-2)    pos2/0 (1)      2  11.76           7
Total count: 17

Process took 0.016385 seconds
```

7.4 Ongelmia jäljityksessä

Netflow-tietojen avulla tapahtuva jäljitys ei välttämättä takaa oikeita tuloksia riippuen useasta eri tekijästä. Ongelmia muodostavat esimerkiksi viimeisen reitittimen takana tai vuon reitillä olevat verkon laitteet, jotka eivät kerää Netflow-dataa. Tällaisia voivat olla esimerkiksi kytkimet tai keskittimet. Kolmantena ongelmana on saastunut laite, joka liikennöi väärällä IP-osoitteella, joka on jonkin verkossa olevan laitteen IP-osoite. Tällöin Netflow-datasta havaitaan voita, joiden polut ovat erilaiset, riippuen siitä, missä verkon osassa liikennöivät laitteet sijaitsevat.

7.4.1 Reitittimen takana olevat kytkimet ja keskittimet

Jäljityksen tuloksena saatu reititin, joka on ensimmäisenä käsitellyt vuota, ei välttämättä ole ensimmäinen laite, joka on käsitellyt vuossa liikkuvia paketteja. Jos reitittimen liityntä suljetaan, voidaan samalla estää useiden muiden laitteiden liikennöinti

verkkoon (kuva 7.3). Jäljitysskriptiin on ohjelmoitu funktio, joka laskee tietyn reitittimen liityntään saapuvat vuot uniikeista lähdeosoitteista. Lähdeosoitteisiin perustuva laskenta ei ole varma tapa päätellä, onko liitynnän takana useita laitteita, sillä lähdeosoitteen voi väärentää. Muut liitynnän takana olevat laitteet eivät välttämättä liikennöi juuri sillä hetkellä, kun Netflow-dataa kerätään ja tarkastellaan.

Seuraavassa esimerkissä nähdään, että jäljitys palauttaa Tera-2-reitittimen todennäköisimpänä lähteenä. Uniikkien IP-osoitteiden laskemisen perusteella voidaan olettaa, että reitittimen liitynnän (e1/2) takana on vain yksi verkon laite. Jäljitys palautti myös Tera-1-reitittimen yhtenä mahdollisena lähteenä, mutta se on ollut viimeisenä reitittimenä vain 1.6% tapauksista, joten se voidaan jättää pois laskuista.

```
Tracing ip 172.16.31.2 ...

Possible sources:

Source                   Interface  Count      %  Unique ips
172.16.0.3   (Tera-2)     e1/2 (5)    124  98.41           1
172.16.0.2   (Tera-1)   pos2/0 (1)      2   1.59           6
Total count: 126
```

Seuraavassa esimerkissä lähteeksi on varmistunut Tera-2-reititin, mutta IP-osoitteita laskettaessa huomataan, että reitittimen liitynnän takaa liikennöi ainakin viisi eri IP-osoitetta. Tästä voidaan päätellä, että todennäköisesti liitynnän takana on vielä kytkin.

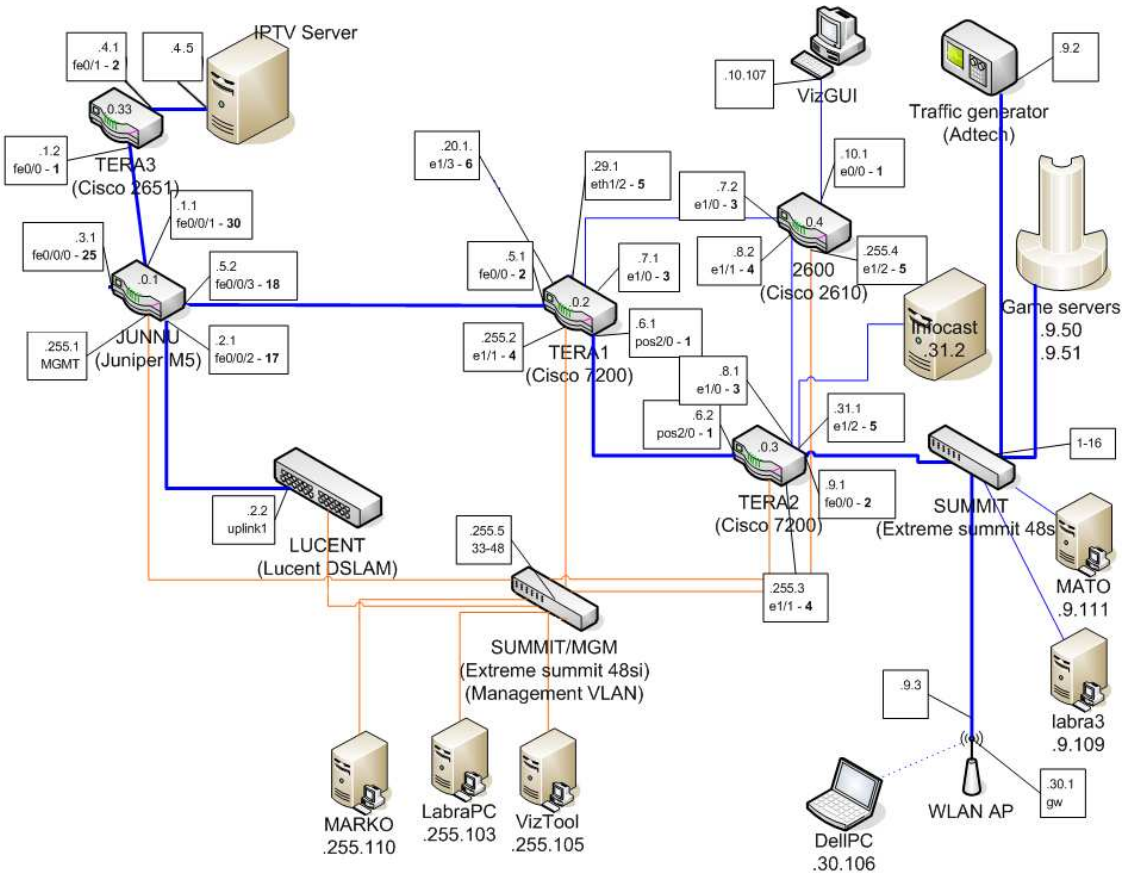
```
Tracing ip 172.16.9.50 ...

Possible sources:

Source                   Interface  Count      %  Unique ips
172.16.0.3   (Tera-2)    fe0/0 (2)      7 100.00           5
Total count: 7
```

Koska Netflow-datan avulla ei voida etsiä kytkimiä tai keskittimiä, jos ne eivät lähetä Netflow-dataa, tulee verkon topologia näiden osalta olla ennalta selvillä, jotta saastuneen laitteen liikenne voidaan estää oikeassa laitteessa, johon saastunut laite on kytketty.

7.4.2 Netflow-datan puuttuminen reitiltä

Jos jokin vuon reitillä oleva verkon laite ei kerää ja lähetä Netflow-dataa, näkyy se tuloksissa useana eri lähteenä. Yleensä puuttuva Netflow-data ei vääristä tuloksia, mutta jos useimmat vuot kulkevat kaikkien tuloksissa näkyvien reitittimien kautta, ei ensimmäistä reititintä voida päätellä.

Seuraavassa esimerkissä Tera-3-reititintä edeltää reititin (Juniper, kuva 7.6), joka ei lähetä Netflow-dataa. Tuloksista voidaan huomata, että Tera-3-reititin on käsitellyt suurimman osan voista. Tämä kuitenkin johtuu siitä, että kaikki jäljitettävästä laitteesta lähtenyt liikenne ei ole kulkenut enää Tera-1- ja Tera-2-reitittimien läpi. Jos kaikki liikenne kulkisi myös muiden reitittimien läpi, olisi ensimmäisen reitittimen päättely hankalampaa ilman tietoa verkon topologiasta.



Kuva 7.6: Laila-verkko, topologia 2.

```
Tracing ip 172.16.4.5 ...

Possible sources:

Source                    Interfac   Count      %  Unique ips
172.16.0.2   (Tera-1)     fe0/0 (2)      2  14.29           4
172.16.0.3   (Tera-2)    pos2/0 (1)      2  14.29           7
172.16.0.33  (Tera-3)     fe0/1 (2)     10  71.43           1
Total count: 14
```

7.4.3 Kaksi samaa IP-osoitetta

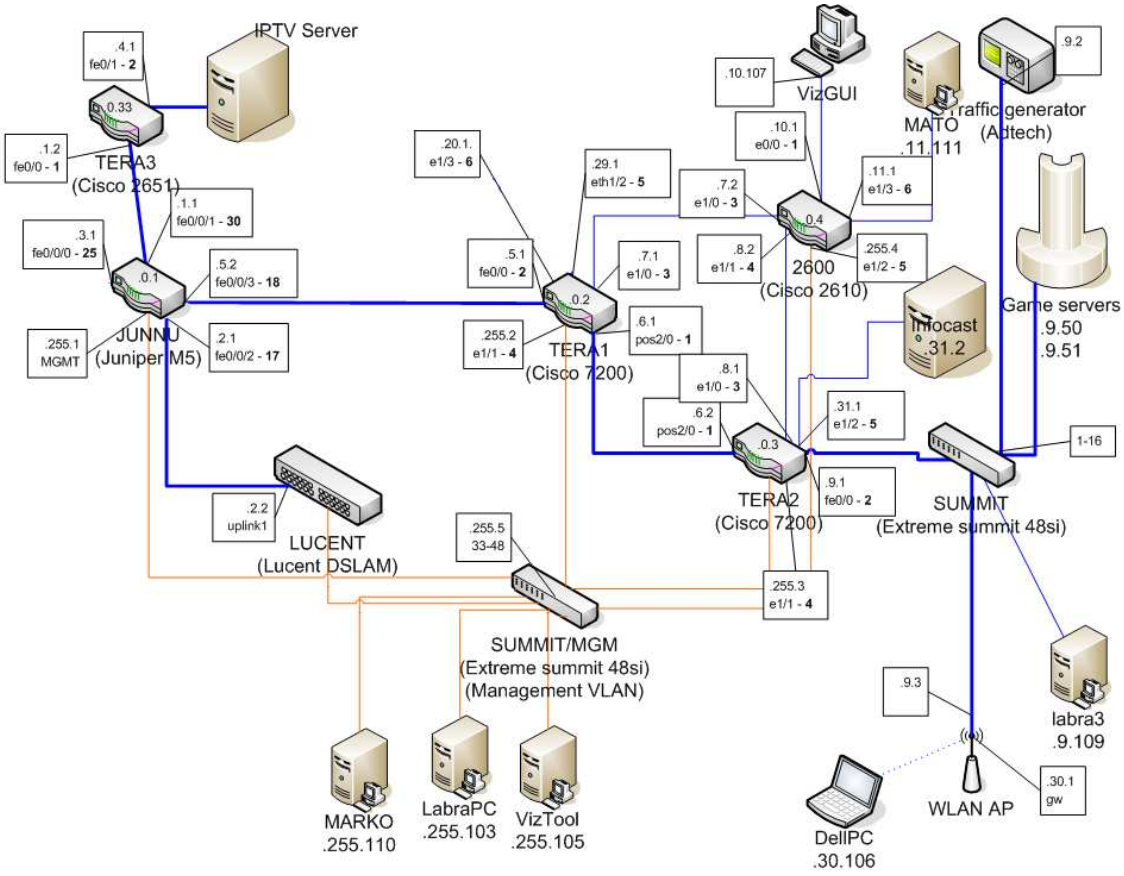
Jos saastunut laite liikennöi väärennetyllä IP-osoitteella, joka kuuluu verkossa toisaalla olevalle lailliselle laitteella, näkyy se jäljityksessä kahtena eri reittinä.

Seuraavassa esimerkissä verkossa liikennöi laillinen laite ja samalla verkkoon on luotu liikennegeneraattorilla (engl. *Traffic generator*) väärennettyä liikennettä samasta lähdeosoitteesta. Sekä laillinen että väärennetty liikenne on osoitettu samalle verkon laitteelle, joten reitin loppu on kummallakin sama.

Tuloksista voidaan nähdä, että sekä reititin Cisco 2600 että Tera-1 (kuva 7.7) ovat käsitelleen lähes saman verran voita jäljitettävästä osoitteesta. Testissä liikennegeneraattori oli Tera-1-reitittimen takana ja laillinen lähde Cisco 2600 -reitittimen takana. Netflow-datan avulla ei voida mitenkään päätellä kumpi lähteistä on laillinen lähde.

```
Tracing ip 172.16.11.111 ...

Possible sources:

Source                    Interface  Count      %  Unique ips
172.16.0.4   (C 2600)      e1/3 (6)     10  41.67           1
172.16.0.3   (Tera-2)      e1/0 (3)      1   4.17           2
172.16.0.2   (Tera-1)    pos2/0 (1)     13  54.17           7
Total count: 24
```

Jos väärennetty IP-osoite sijaitsee saman reitittimen takana kuin laillinen osoite, ei niitä voida erottaa mitenkään toisistaan Netflown avulla, vaan jäljitys antaa tulokseksi yhden reitittimen. Jäljityksen tuloksesta ei voida havaita väärennetyn IP-osoitteen olemassaoloa.



Kuva 7.7: Laila-verkko, topologia 3.

7.5 Netwrapper

Netwrapper on WTS Networks Oy:n kehittämä IP-verkkojen hallintaan käytetty järjestelmä, jonka avulla voidaan hallita eri valmistajien verkkolaitteita yhden järjestelmän avulla. Netwrapper toimii rajapintana verkon laitteiden ja ylläpitäjän välillä, tarjoten helpon ja nopean verkon laitteiden konfiguroinnin. Netwrapperia käyttävät esimerkiksi Internetoperaattorit asiakkaiden tilausten ja muutosten pohjalta tapahtuvaan verkon konfigurointiin (nopeudet, palveluluokat, ym.). [55]

Netwrapperiin kirjoitettujen rajapintojen avulla voidaan konfiguroida automaattisesti haluttuja verkon aktiivilaitteita, esimerkiksi reitittimiä, estämään saastuneiden laitteiden liikennöinti verkkoon. Konfigurointi tapahtuu laitteen merkistä ja mallista riippumatta, sillä Netwrapper pystyy konfiguroimaan useimpia markkinoilla olevia laitteita (Cisco, Nokia, HP, Ericsson, Juniper, Telco Systems, Extreme, jne.). Konfigurointi voidaan tehdä automaattisesti, joten ylläpitäjän toimia ei tarvita. [55]

7.5.1 Aktiivilaitteen liitynnän sulkeminen

Kun löydetään verkon aktiivilaite, jonka liityntään on kytketty saastunut laite, tulee liityntä sulkea mahdollisimman nopeasti, jotta saastunut laite ei pääse aiheuttamaan suurempaa vahinkoa verkossa. Jos edellisissä luvuissa kuvattujen jäljitysmenetelmien avulla saadaan selville verkon aktiivilaite, johon saastunut kone on kytketty, voidaan sen liityntä sulkea automaattisesti Netwrapperin avulla.

Jos jäljityksen avulla ei saada selville todellista verkon laitetta, johon jäljitettävä laite on kytketty, tulee laite saada selville muilla keinoin. Esimerkiksi kytkimet ja keskittimet, jotka eivät lähetä Netflow-dataa, voidaan etsiä erillisestä verkon topologiatietokannasta, jossa on tiedot verkon laitteista. Jäljityksen jälkeen tuloksena saadun reitittimen liitynnän takana mahdollisesti olevat muut laitteet saadaan näin selville ja oikea liityntä voidaan sulkea.

Netwrapperin tulisi sisältää rajapintakutsu, jonka avulla sulkeminen voidaan tehdä. Koska Netwrapperilla on tieto verkon laitteista, voidaan sille antaa tunnisteena verkon aktiivilaitteen IP-osoite, jolloin Netwrapper pystyy identifioimaan konfiguroitavan laitteen. Tämän lisäksi Netwrapperin tulee tietää mikä liityntä suljetaan. Automaattisesti tapahtuva liitynnän sulkeminen pitää purkaa jollakin keinolla, joko ylläpitäjän toimesta tai automaattisesti. Tästä syystä toisen osapuolen tulee voida avata liityntä tietyn ajan kuluttua.

Esimerkkinä Pythonin syntaksilla määritetty metodi, joka ottaa parametreikseen IP-osoitteen ja liitynnän nimen. Liitynnän indeksi voi muuttua verkon laitteen uu-

delleenkäynnistyksen yhteydessä, joten käyttämällä liitynnän nimeä, voidaan varmistua tunnisteen pysyvyydestä. Liitynnän indeksi saadaan helposti selville laitteen tiedoista.

```
shutdownInterface(dev_ip, if_name)
```

Metodin tulisi sammuttaa määritellyn reitittimen tietty liityntä. Seuraavassa esimerkissä toimenpiteet on määritelty Ciscon reitittimien komentoina, mutta muiden valmistajien reitittimissä on vastaavanlaiset komennot. Config-tilassa laitteelle annetaan `interface`-komento ja liitynnän nimi, tämä aktivoi valitun liitynnän konfigurointitilan. Tämän jälkeen annetaan `shutdown`-komento, joka sulkee koko liitynnän. Esimerkiksi: [56]

```
Router(config)# interface ethernet 0
Router(config-if)# shutdown
```

Tämän jälkeen laite ilmoittaa liitynnän onnistuneesta sulkemisesta:

```
08:32:03:%LINK-5-CHANGED:Interface Ethernet 0, changed
  state to administratively down
```

Liitynnän käynnistäminen tapahtuu muuten samalla tavalla, mutta komento on `no shutdown`. Esimerkiksi:

```
Router(config)# interface ethernet 0
Router(config-if)# no shutdown

08:32:16:%LINK-3-UPDOWN:Interface Ethernet 0, changed
 state to up
08:32:17:%LINEPROTO-5-UPDOWN:Line protocol on Interface
 Ethernet 0, changed state to up
```

Tämä voidaan määritellä Netwrapperiin esimerkiksi `startInterface`-metodi:

```
startInterface(dev_ip, if_name)
```

Metodien tulee palauttaa paluuarvo, joka ilmaisee toimenpiteen onnistumisen. Konfiguroinnin onnistuessa metodi palauttaa arvon nolla. Jos toimenpide ei onnistu, voidaan palauttaa virhekoodi, joka on suurempi kuin nolla. Esimerkiksi, jos annetulla IP-osoitteella ei löydy laitetta, palautetaan virhekoodi 1. Jos laitteella ei ole annettua liityntää, palautetaan virhekoodi 2. Paluuarvon avulla Netwrapperia kutsuva ohjelma saa tietoonsa mahdolliset virhetilanteet konfiguroinnissa ja niiden syyt.

7.5.2 ACL-säännön luominen

Jos jäljityksen perusteella havaitaan, että reitittimen portin takana liikennöi useita eri laitteita, mutta porttiin mahdollisesti kytkettyä keskitintä tai kytkintä ei saada selville verkon topologiatiedoista (tai sitä ei voida konfiguroida), voidaan reitittimen pääsylistaan konfiguroida sääntö, joka estää tietystä lähteestä tulevan liikenteen. Reitittimen koko liityntää ei voida sulkea, sillä se estää muiden liitynnän takana olevien laillisten laitteiden liikennöinnin verkkoon.

Väärennettyjen IP-osoitteiden tapauksessa sääntö estää vain senhetkisen väärän IP-osoitteen liikennöinnin verkkoon. Vaihtamalla IP-osoitetta liikennöinti on taas mahdollista. Toisena ongelmana väärennettyjen osoitteiden tapauksessa on, jos saman reitittimen portin takana on laillinen laite, joka käyttää samaa IP-osoitetta. Tällöin estämällä väärennetyn IP-osoitteen liikennöinti, estetään myös laillisen laitteen liikennöinti verkkoon. Yleensä edellinen tapaus voidaan estää verkon laitteessa konfiguroimalla se niin, ettei kahdesta eri portista voi tulla samalla lähdeosoitteella varustettua liikennettä.

Netwrapperin rajapintaan voidaan lisätä metodi, jolla voidaan luoda ja poistaa ACL-listamerkintöjä. Esimerkiksi metodi `addACLRule`, joka ottaa parametreiksi lähdeosoitteen ja kohdeosoitteen.

```
addACLRule(dev_ip, src_ip)
```

Metodin tulisi konfiguroida annettuun verkon aktiivilaitteeseen (`dev_ip`) pääsylistasääntö, joka estää annetusta osoitteesta tulevan liikenteen pääsemästä verkkoon. Samalla voidaan kieltää myös osoitteeseen saapuva liikenne. Esimerkiksi Ciscon ACL-säännöillä metodin suorittamat komennot olisivat:

```
Router(config)# access-list [acl-numero] deny any src_ip any
Router(config)# access-list [acl-numero] deny any any src_ip
```

Ensimmäinen sääntö estää kaiken liikenteen, joka tulee saastuneelta laitteelta ja toinen sääntö estää liikenteen saapumasta saastuneelle laitteelle. Pääsylistan numeroa (`acl-numero`) tulee kasvattaa seuraavia sääntöjä tehtäessä. Säännön lisäämisen yhteydessä tulisi tarkistaa, ettei samalla lähdeosoitteella ole jo sääntöä ennestään.Pääsylistan numeron määritteleminen jätetään Netwrapperin toimeksi, sillä se tietää mitä sääntöjä laitteeseen on jo konfiguroitu ja mikä on seuraava vapaa sääntönumero. Netwrapperiin voidaan myös määritellä tietty numeroavaruus, josta säännön numero valitaa.

Pääsylista voidaan kumota komennolla `no access-list`, esimerkiksi:

```
Router(config)# no access-list [acl-numero]
```

Netwrapperiin voidaan lisätä metodi `removeACLRule`, joka poistaa annetusta laitteesta tietyn ACL-listan:

```
removeACLRule(dev_ip)
```

Netwrapperin tulee käydä laitteessa olevat säännöt läpi ja etsiä sääntöjä, joissa lähde- tai kohdeosoitteena on annettu IP-osoite. Tietyn numeroavaruuden käyttö sääntönumeroa valittaessa estää Netwrapperia poistamasta muita sääntöjä, kuin sen itsensä määrittelemät.

Myös näiden metodien tulisi palauttaa paluuarvo, joka kertoo toimenpiteen onnistumisesta.

8 Yhteenveto

IDS-järjestelmät tulevat yleistymään yritysten ja muiden tahojen verkoissa, jotka vaativat reaaliaikaista uhkien havaitsemista. Tulevaisuudessa IDS-järjestelmiä integroidaan luultavasti myös kuluttajatuotteisiin, kuten palomuureihin ja laajakaistareitittimiin. Oikein sijoitettuna IDS-järjestelmät pystyvät valvomaan tehokkaasti verkon liikennettä ja havaitsemaan käynnissä olevat hyökkäykset. IDS-järjestelmien kehitys pyrkii ratkaisemaan niissä olevat heikkoudet, kuten NIDS-järjestelmien pakettien uudelleenkokoamisohjelmat.

Verkkojen nopeuksien kasvaessa IDS-järjestelmien taakkaa voidaan helpottaa etsimällä hyökkäyksiä muilla keinoilla kuin resursseja kuluttavalla pakettien tutkimisella. Eräs tapa on etsiä hyökkäyksiä Netflow-datan avulla, jota käytetään pääasiassa verkon valvontaan, liikenteen analysointiin ja laskutukseen. Laajentamalla Netflown käyttöä matojen ja erilaisten hyökkäysten havaitsemiseen, voidaan verkkoon toteuttaa pienillä muutoksilla IDS-järjestelmää vastaava järjestelmä, joka pystyy havaitsemaan tunnettuja uhkia sekä nollapäivän hyökkäyksiä. Netflow-data soveltuu hyvin tiettyjen uhkien havaitsemiseen, sillä se sisältää tiedot verkossa tapahtuneista yhteyksistä, niiden osoitteista sekä liikenteen määrästä. Esimerkiksi madon levitessä verkkoon muodostuu paljon yhteyksiä saastuneesta laitteesta verkon muihin laitteisiin, jotka havaitaan analysoimalla tietyltä aikaväliltä kerättyä Netflow-dataa. Toistaiseksi Netflowta tukevat vain Ciscon verkkolaitteet, mutta muilla valmistajilla on vastaavia tekniikoita suunnitteilla tai käytössä.

Netflown avulla tapahtuvaa havaitsemista ei toistaiseksi ole integroitu vapaan lähdekoodin IDS-järjestelmiin, mutta useissa ohjelmistoissa kehitys on meneillään. Kaupallisissa ohjelmistoissa Netflowta käytetään varsinkin Ciscon ohjelmistoissa (esimerkiksi MARS), jotka pystyvät vertaamaan IDS-järjestelmien tietoja Netflow-tietojen kanssa. Netflow-datan analysointiin perustuvia järjestelmiä on kehitetty, mutta ne ovat eri tahojen omaan käyttöön toteuttamia skriptejä, joiden yleinen käyttö on hankalaa.

Netflown avulla voidaan suuressa verkossa jäljittää saastuneita laitteita ja saada selville lähin verkon laite, johon saastunut laite on kytketty. Paras tulos saavutetaan, jos jokaisesta verkon laitteesta saadaan Netflow-dataa, jolloin jäljityksen onnistuminen on todennäköisintä. Ellei kaikista verkon laitteista saada Netflow-dataa, tulee verkon topologiasta ja laitteista olla tietoa, jotta viimeisin laite voidaan tunnistaa.

Tutkielmassa kuvataan testiverkossa suoritettuja jäljityksiä tätä varten kirjoitetulla ohjelmalla. Tuloksista huomataan, että viimeinen verkon laite voidaan jäljittää, jos kaikki verkon laitteet lähettävät Netflow-dataa. Jäljitystä voi haitata monet ongelmat kuten yksittäinen verkon laite, joka ei lähetä Netflow-dataa tai väärennettyä IP-osoitetta vastaavan aidon laitteen liikennöinti verkossa.

Kun viimeisin laite saadaan selville, voidaan sen liityntä sulkea automaattisesti ja näin estää esimerkiksi madon saastuttaman laitteen liikennöinti verkkoon. Jos liitynnän takana on muita laitteita, ei liityntää voida sulkea estämättä muiden laitteiden liikennöinti verkkoon. Tällöin esimerkiksi reitittimeen voidaan konfiguroida tietyn laitteen liikennöinnin estävä pääsylistasääntö. Automaattinen liitynnän sulkeminen tai pääsylistan luominen voidaan tehdä esimerkiksi WTS Networks Oy:n kehittämällä Netwrapper-ohjelmistolla, johon on toteutettu sopiva rajapinta.

Lähteet

[1] Thomas Rob, *Tracking Spoofed IP Addresses Version 2.0*, saatavilla WWW-muodossa <URL:http://www.cymru.com/Documents/tracking-spoofed.html>, 8. helmikuuta 2001.

[2] Kaario Kimmo, *TCP/IP-verkot*, Docendo, Jyväskylä, 2002.

[3] Insecure.org, *Nmap Reference Guide*, saatavilla WWW-muodossa <URL:http://www.insecure.org/nmap/man/>, viitattu 16. maaliskuuta 2006.

[4] McClure Stuart, Scrambray Joel, Kurtz George, *Hakkeroinnin torjunta*, Gummerus, Jyväskylä, 2002.

[5] Cisco Systems Inc, *Cisco IOS Firewall: Configuring IP Access Lists*, saatavilla PDF-muodossa <URL:http://www.cisco.com/warp/public/707/confaccesslists.pdf>, 23. helmikuuta 2006.

[6] Cisco Systems Inc, *Cisco Security Advisory: TFTP Long Filename Vulnerability*, saatavilla PDF-muodossa <URL: http://www.cisco.com/warp/public/707/ios-tftp-long-filename-pub.pdf>, 8. lokakuuta 2004.

[7] FX of Phenoelit, *Burning the bridge: Cisco IOS exploits*, Phrack Magazine, 12, 2002.

[8] Cisco Systems Inc, *Cisco IOS Password Encryption Facts*, saatavilla PDF-muodossa <URL:http://www.cisco.com/warp/public/701/64.pdf>, 16. marraskuuta 2004.

[9] Cisco Systems Inc, *Configuring the Catalyst Switched Port Analyzer (SPAN) Feature*, saatavilla PDF-muodossa <URL:http://www.cisco.com/warp/public/473/41.pdf>, 25. huhtikuuta 2005.

[10] Balding, Shipley, Balder ym., *Hakkerin käsikirja*, Edita, Helsinki, 2002.

[11] Song Dug, *dsniff Frequently Asked Questions*, saatavilla WWW-muodossa <URL:http://www.monkey.org/~dugsong/dsniff/faq.html>, 7. joulukuuta 2001.

[12] Russell Ryan, *Hack Proofing Your Network*, Syngress Publishing, Rockland, MA, 2002.

[13] Ruohonen Mika, *Tietoturva*, WS Bookwell, Porvoo, 2002.

[14] Hardjono Thomas, *Security in wireless LANs and MANs*, Artech House, Norwood, MA, 2005.

[15] Soungjoo Han, *Breaking through a Firewall using a forged FTP command*, Phrack Magazine, 11, 2005.

[16] Lincoln Mattos Cristiano, *Security flaw in Linux 2.4 IPTables using FTP PORT*, saatavilla WWW-muodossa <URL:http://www.netfilter.org/security/2001-04-16-ftp.html>, 16. huhtikuuta 2001.

[17] Kemmerer Richard A., Vigna Giovanni, *Intrusion Detection: A Brief History and Overview*, Computer, Volume 35, Issue 4, huhtikuu 2002, s. 27–30.

[18] Baker Andrew R., Caswell Brian, Poor Mike, *Snort 2.1 Intrusion Detection*, Syngress Publishing, Rockland, MA, 2004.

[19] Zhang Yu-Fang, Xiong Zhong-Yang, Wang Xiu-Qiong, *Distributed intrusion detection based on clustering*, Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, 18. elokuuta 2005, s. 2379–2383.

[20] Yang Hongyu, Xie Lixia, Sun Jizhou, *Intrusion Detection for Wireless Local Area Network*, Electrical and Computer Engineering, 2004. Canadian Conference on, 2-5 elokuuta 2004, s. 1949–1952.

[21] Mishra Amitabh, Nadkarni Ketan, Patcha Animesh, *Intrusion Detection In Wireless Ad Hoc Networks*, Wireless Communications, IEEE, Volume 11, Issue 1, helmikuu 2004, s. 48–60.

[22] Brutch Paul, Ko Calvin, *Challenges in intrusion detection for wireless ad-hoc networks*, Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on, tammikuu 2003, s. 368–373.

[23] Zhang Yongguang, Lee Wenke, *Intrusion Detection in Wireless Ad-Hoc Networks*, saatavilla PDF-muodossa <URL: http://www.cs.huji.ac.il/labs/danss/sensor/adhoc/zhang_2000intrusiondetection.pdf>, 2000.

[24] Jiang Hai, Gao Yun, Zhang Guanyuan, He Yongming, *A Zone-Based Intrusion Detection System for Wireless Ad Hoc Distribution Power Communication Networks*, Transmission and Distribution Conference and Exhibition: Asia and Pacific, elokuu 2005, s. 1–6.

[25] Lee Seungjoon, Han Bohyung, Shin Minho, *Robust Routing in Wireless Ad Hoc Networks*, Parallel Processing Workshops, 2002. Proceedings. International Conference on, elokuu 2002, s. 73–78.

[26] Ptacek Thomas H., Newsham Timothy N., *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*, saatavilla PDF-muodossa <URL:http://www.insecure.org/stf/secnet_ids/secnet_ids.pdf>, tammikuu 1998.

[27] Carter Earl, *Cisco IDS Sensor Deployment Considerations*, saatavilla WWW-muodossa <URL: http://www.ciscopress.com/articles/article.asp?p=25327&rl=1>, 15. helmikuuta 2002.

[28] Kemmerer Richard A., Vigna Giovanni, *Hi-DRA: Intrusion Detection for Internet Security*, Proceedings of the IEEE, Volume 93, Issue 10, lokakuu 2005, s. 1848–1857.

[29] Idris Norbik Bashah, Shanmugam Bharanidlran, *Artificial Intelligence Techniques Applied to Intrusion Detection*, INDICON, 2005 Annual IEEE, 11-13 joulukuuta 2005, s. 52–55.

[30] Mukkamala Srinivas, Sung Andrew H., *A comparative study of techniques for intrusion detection*, Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on, 3-5 marraskuuta 2003, s. 570–577.

[31] Chavan Sampada, Shah Khusbu, Dave Neha, Mukherjee Sanghamitra, Abraham Ajith, Sanyal Sugata, *Adaptive Neuro-Fuzzy Intrusion Detection Systems*, Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on, Volume 1, 2004, s. 70–74.

[32] Ren Pin, Gao Yan, Li Zhichun, Chen Yan, Watson Benjamin, *IDGraphs: Intrusion Detection and Analysis Using Histographs*, CVisualization for Computer Security, 2005. (VizSEC 05). IEEE Workshop on, 26. syyskuuta 2005, s. 39–46.

[33] Itoh Takayuki, Takakura Hiroki, Sawada Atsushi, Koyamada Koji, *Hierarchical Visualization of Network Intrusion Detection Data*, Computer Graphics and Applications, IEEE, maalis-huhtikuu 2006, s. 40–47.

[34] Siddharth Sumit, *Evading NIDS, revisited*, saatavilla WWW-muodossa <URL:http://www.securityfocus.com/infocus/1852>, 6. joulukuuta 2005.

[35] Shankar Umesh, Paxson Vern, *Active Mapping: Resisting NIDS Evasion Without Altering Traffic*, Security and Privacy, 2003. Proceedings. 2003 Symposium on, toukokuu 2003, s. 44–61.

[36] Novak Judy, *Target-Based Fragmentation Reassembly*, saatavilla PDF-muodossa <URL:http://www.snort.org/reg/docs/target_based_frag.pdf>, 21. syyskuuta 2005.

[37] Hacker Eric, *IDS Evasion with Unicode*, saatavilla WWW-muodossa <URL:http://www.securityfocus.com/infocus/1232>, 3. tammikuuta 2001.

[38] Rain Forest Puppy, *A look at whisker's anti-IDS tactics*, saatavilla WWW-muodossa <URL:http://www.wiretrip.net/rfp/txt/whiskerids.html>, 24. joulukuuta 1999.

[39] Handley Mark, Paxson Vern, Kreibich Christian, *Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics*, saatavilla PDF-muodossa <URL:http://www.icir.org/vern/papers/norm-usenix-sec-01.pdf>, toukokuu 2001.

[40] Provos Niels, *Honeyd General Information*, saatavilla WWW-muodossa <URL:http://www.honeyd.org/general.php>, viitattu 12. heinäkuuta 2006.

[41] Yegneswaran Vinod, Barford Paul, Paxson Vern, *Using Honeynets for Internet Situational Awareness*, saatavilla PDF-muodossa <URL:http://www.icir.org/vern/papers/sit-aware-hotnet05.pdf>, 2005.

[42] Check Point Software Tehcnologies Ltd., *FireWall-1 Datasheet*, saatavilla PDF-muodossa <URL:http://www.checkpoint.com/products/downloads/firewall-1_datasheet.pdf>, 12. toukokuuta 2005.

[43] Baumrucker C. Tate, Burton James D., Dentler Scott, Dubrawsky Ido, Osipov Vitaly, Sweeney Michael, *Cisco Security Professional's Guide to Secure Intrusion Detection Systems*, Syngress Publishing, Rockland, MA, 2003.

[44] Cisco Systems Inc, *Cisco Ips 4200 Series Sensors*, saatavilla PDF-muodossa <URL: http://www.cisco.com/application/pdf/en/us/guest/products/ps4077/c1650/ccmigration_09186a008014873c.pdf>, lokakuu 2006.

[45] Paxson Vern, *Bro: A System for Detecting Network Intruders in Real-Time*, saatavilla PS-muodossa <URL:ftp://ftp.ee.lbl.gov/papers/bro-CN99.ps.gz>, 14. joulukuuta 1999.

[46] Sommer Robin, Feldmann Anja, *NetFlow: Information loss or win?*, saatavilla PDF-muodossa <URL:http://www.net.informatik.tu-muenchen.de/~anja/feldmann/papers/netflow.pdf>, 2002.

[47] Cisco Systems Inc, *Cisco IOS Netflow Data Sheet*, saatavilla PDF-muodossa <URL:http://www.cisco.com/application/pdf/en/us/guest/tech/tk812/c1482/cdccont_0900aecd80173f71.pdf>, 2004.

[48] Lucas Michael W., *Monitoring Network Traffic with Netflow*, saatavilla WWW-muodossa <URL:http://www.onlamp.com/pub/a/bsd/2005/08/18/Big_Scary_Daemons.html>, 18. elokuuta 2005.

[49] Gong Yiming, *Detecting Worms and Abnormal Activities with NetFlow, Part 1*, saatavilla WWW-muodossa <URL:http://www.securityfocus.com/infocus/1796>, 16. elokuuta 2004.

[50] Gong Yiming, *Detecting Worms and Abnormal Activities with NetFlow, Part 2*, saatavilla WWW-muodossa <URL:http://www.securityfocus.com/infocus/1802>, 23. syyskuuta 2004.

[51] Pao Tsang-Long, Wang Po-Wei, *NetFlow Based Intrusion Detection System*, Networking, Sensing and Control, 2004 IEEE International Conference on, Volume 2, 2004 s. 731–736.

[52] Nazario Jose, *Defense and Detection Strategies against Internet Worms*, Artech House, Norwood, MA, 2004.

[53] Fullmer Mark, *Man pages for flow-tools 0.67.*, saatavilla WWW-muodossa <URL:http://www.splintered.net/sw/flow-tools/docs/>, viitattu 11. syyskuuta 2006.

[54] Miller Damien, *Flowd*, saatavilla WWW-muodossa <URL:http://www.mindrot.org/projects/flowd/>, viitattu 4. marraskuuta 2006.

[55] WTS-Networks Oy, *Netwrapper*, saatavilla WWW-muodossa <URL:http://www.wtsnetworks.fi>, viitattu 17. lokakuuta 2006.

[56] Cisco Systems Inc, *Interface and Hardware Component Commands*, saatavilla PDF-muodossa <URL:http://www.cisco.com/application/pdf/en/us/guest/products/ps5207/c2001/ccmigration_09186a00801ad971.pdf>, 2005.

A Config.py-tiedoston lähdekoodi

Tämä liite sisältää Pythonilla toteutetun jäljitysohjelman konfigurointitietojen lähdekoodin.

```
# Global definitions

# Network device's information
# This dictionary contains information about networkk's devices, especially their
# interfaces, so that tracing is possible.

# Defined routers: Tera1, Tera2, Tera3, Cisco 2600
# Missing: Juniper

devices = {
        "172.16.0.2": {
                "name": "Tera-1",
                "index": 0,
                "interfaces": {
                        0: {"name": "-", "ip": None},  # interface 0 is device itself
                        1: {"name": "pos2/0", "ip": "172.16.6.1"},
                        2: {"name": "fe0/0", "ip": "172.16.5.1"},
                        3: {"name": "e1/0", "ip": "172.16.7.1"},
                        4: {"name": "e1/1", "ip": "172.16.255.2"},
                        5: {"name": "e1/2", "ip": None},
                        6: {"name": "e1/3", "ip": "172.16.20.1"}

                        }

                },
        "172.16.0.3": {
                "name": "Tera-2",
                "index": 1,
                "interfaces": {
                        0: {"name": "-", "ip": None},
                        1: {"name": "pos2/0", "ip": "172.16.6.2"},
                        2: {"name": "fe0/0", "ip": "172.16.9.1"},
                        3: {"name": "e1/0", "ip": "172.16.8.1"},
                        4: {"name": "e1/1", "ip": "172.16.255.3"},
                        5: {"name": "e1/2", "ip": "172.16.31.1"},
                        6: {"name": "e1/3", "ip": None},

                },
        },
        "172.16.0.33": {
                "name": "Tera-3",
                "index": 1,
                "interfaces": {
                        0: {"name": "-", "ip": None},
                        1: {"name": "fe0/0", "ip": "172.16.1.2"},
```

```
                        2: {"name": "fe0/1", "ip": "172.16.5.2"},
                },
        },
        "172.16.0.4": {
                "name": "C 2600",
                "index": 2,
                "interfaces": {
                        0: {"name": "-", "ip": None},
                        1: {"name": "e0/0", "ip": "172.16.10.1"},
                        2: {"name": "N/A",  "ip": None},
                        3: {"name": "e1/0", "ip": "172.16.7.2"},
                        4: {"name": "e1/1", "ip": "172.16.8.2"},
                        5: {"name": "e1/2", "ip": "172.16.255.4"},
      6: {"name": "e1/3", "ip": "172.16.11.1"},
                },


        },
}

# Defines, if protocol numbers are replaced by their names defined in dictionary

use_protocol_names = True

protocols = {
        1: "icmp",
        6: "tcp",
        17: "udp",
        61: "dummy",
        }

# Defines, if port names are replaced by their names defined in dictionary

use_port_names = True

ports = {
        20: "ftp",
        21: "ftp",
        22: "ssh",
        25: "smtp",

        80: "http",
        }

# Defines, if tcp-flags are converted to letters A(CK), R(ST), ...

convert_flags_to_str = True


# Defines if gateways are traced only once, this speeds up tracing, but
# if multiple devices is found, we cannot calculate amount of flows processed
# by those devices. So we cannot determine which router is likely first router.

trace_gws_only_once = False

# EOF global definitions
```

B Trace.py-tiedoston lähdekoodi

Tämä liite sisältää Pythonilla toteutetun jäljitysohjelman lähdekoodin.

```
#!/usr/bin/python2.3

############################################################
# File: show.py
# Language: Python
# Created: 27.9.2006
# Version: 0.1
# Description: Test script to test filtering and traceback with
#         Netflow-data collected with flowd. Script uses
#         flowd's Python API.
# Author: Marko Andersson (maenande@cc.jyu.fi)
#
############################################################
#
# Changelog:
#
# 27.9.2006
#  * Created
#  + Show-method to print flows
#  + Filter-method to filter flows
#
# 29.9.2006
#  + devices-disctionary that contains all information about devices in network
#  * Improved filter-method with more filtering options
#  + Methods to get devices by Ip or If by device
#
# 3.10.2006
#  + Trace-method
#
# 5.10.2006
#  + Commandline executing
#
# 6.10.2006
#  + Trace by index
#
############################################################


import flowd
import string
import math
import datetime
import time
import timing
import getopt
import sys
```

```
from config import *



def getDeviceByIfIp(ifIp):
  """ Returns device from devices dictionary, which interface ip matches with
      given ip
  """

  for dev, val in devices.iteritems():
    for if_num, if_data in val["interfaces"].iteritems():
      if if_data["ip"] == ifIp:
        return (dev, devices[dev])

  return None

def getIfByDevice(dev, dev_if):
  try:
    return getDevice(dev)["interfaces"][dev_if]
  except KeyError:
    print "ERROR: Missing interface %d information for device %s" % (dev_if, dev)


def getIfIpByDevice(dev, dev_if):

  return getIfByDevice(dev, dev_if)["ip"]

def getIfNameByDevice(dev, dev_if):

  return getIfByDevice(dev, dev_if)["name"]

def getDeviceName(dev):

  if devices.has_key(dev):
    return devices[dev]["name"]
  else:
    return dev

def getDevice(dev):
  try:
    return devices[dev]
  except KeyError:
    print "ERROR: Missing device information for %s" % dev

def getDeviceIndex(dev):

  return getDevice(dev)["index"]

def getDevicesByIndex(index):

  res = []

  for k,v in devices.iteritems():
    if getDeviceIndex(k) == index: res.append(k)

  return res
```

```
def getFlagString(fl):
  """Converts cumulative tcp-flagnumber to string"""

  if not convert_flags_to_str: return fl

  ret = []

  if fl != 0 and fl % 128 == 0:
    ret.append('C')
    fl = fl - 128
  if fl != 0 and fl % 64 == 0:
    ret.append('E')
    fl = fl - 64
  if fl != 0 and fl % 32 == 0:
    ret.append('U')
    fl = fl -32
  if fl != 0 and fl % 16 == 0:
    ret.append('A')
    fl = fl - 16
  if fl != 0 and fl % 8 == 0:
    ret.append('P')
    fl = fl - 8
  if fl != 0 and fl % 4 == 0:
    ret.append('R')
    fl = fl - 4
  if fl != 0 and fl % 2 == 0:
    ret.append('S')
    fl = fl - 2
  if fl != 0 and fl % 1 == 0:
    ret.append('F')

  return ''.join(ret)


def main():
  """ Main method"""

  timing.start()

  #test()

  commandLine()


  timing.finish()

  print "\nProcess took %.6f seconds\n" %  (timing.micro() / float(10**6))

def loadFile(f):
  """Load flowd logfile"""

  print "Loading file %s..." % f
  try:
    flog = flowd.FlowLog(f, 'rb')
```

```
    flows = []
    for flow in flog:
      flows.append(flow)

    return flows
  except:
    print "Load error."
    return []


def commandLine():
  """Handler for commandline execution"""

  args = []
  opts = []

  try:
    opts, args = getopt.getopt(sys.argv[1:], 'htsfc')
  except getopt.GetoptError:
    print >> sys.stderr, "Invalid commandline arguments"



  flows = []


  if args:
    flows = loadFile(args[0])

    flows = mainFilter(flows)

    for o,a in opts:
      if o in ('-h', '--help'):
        usage()

      if o in ('-t', '--trace'):
        if len(args) == 2:
          showSources(flows, traceIp(flows, args[1]))

        else:
          usage()

      if o in ('-c', '--calculate'):
        if len(args) == 1:
          calculateFlows(flows)
        else:
          usage()

      if o in ('-s', '--show'):
        if len(args) in (1,2):
          show(flows)
        else:
          usage()
  else:
    usage()
```

```
def usage():
  """ Pring help for commandline execution"""

  print "Usage:\n"

  print "show.py [options] tracefile [ip]\n"
  print "-h   Help"
  print "-c  Calculate flows"
  print "-s  Show"
  print "-t  Trace Ip"
  print "\nshow.py -t flowd 192.168.1.1"


def test():
  """Test function"""

  flows = loadFile('/var/log/flows/flowd.isooo')

  flowfilter = {
    "src_addr": ["172.16.9.111"],

    }

  #res = filterFlows(flows, flowfilter)

  #show(res)

  #show(flows)

  print getDifferentIpsFromIf(flows, "172.16.0.3", 2)

  #print getDeviceByIfIp("172.16.6.1")

  #print traceIp(flows, "172.16.255.105")
  #showSources(traceIp(flows, "172.16.255.105"))


def mainFilter(flows):
  """ Filter flows when executed from command line"""

  filter= {
    "src_addr": "172.16.31.2",
    }
  #flows = filterFlows(flows, filter)

  return flows



def showSources(flows, sources):
  """Shows sources from traceIp"""

  src = calcSources(sources)
```

```
  print "Possible sources:\n"

  if len(sources) > 0:
    print "%-25s %12s %6s %7s %11s" %
      ("Source", "Interface", "Count", "%", "Unique ips")
    for (s,i),c in src.iteritems():
      uniqueIps = getDifferentIpsFromIf(flows, s, i)

      print "%-16s (%5s)  %7s (%1s) %6d %6.2f %11s" % (s, getDeviceName(s),
      getIfNameByDevice(s,i), i, c, float(c)/len(sources)*100, len(uniqueIps))

    print "Total count: %d" % len(sources)
  else:
    print "No match."


def calcSources(sources):
  """Calculates occurences of same sources"""

  ret = {}
  for source in sources:
    if ret.has_key(source):
      ret[source] += 1
    else:
      ret[source] = 1

  return ret

def calculateFlows(flows):

  print "Total flowrecords: %d" % len(flows)


def traceIp(flows, ip):
  """ Traces router that first received flow"""

  print "Tracing ip", ip, "...\n"

  return traceIpByGateway(flows, ip)


def getDifferentIpsFromIf(flows, dev, if_index):
  """Gets unique ip's behind given interface"""


  filter = {
    "agent_addr": [dev],
    "if_in": [if_index],

  }

  filtered = filterFlows(flows, filter, "and")

  ips = []
  for flow in filtered:
    if ips.count(flow.src_addr) == 0:
```

```
      ips.append(flow.src_addr)

  return ips



def traceIpByIndex(flows, ip):
  """Traces first router by index"""

  possible_sources = []

  dFilter = {"src_addr": [ip]}

  filtered = filterFlows(flows, dFilter)

  hindex = -1

  for flow in filtered:
    if getDeviceIndex(flow.agent_addr) > hindex:
      hindex = getDeviceIndex(flow.agent_addr)
      possible_sources.append((hindex, flow.agent_addr, flow.if_ndx_in))
      print flow.agent_addr, hindex

  res = []

  # Filter out routers that have smaller index that
  for (indx, src, ifndx) in possible_sources:
    if indx == hindex: res.append((src, ifndx))

  return res

def traceIpByGateway(flows, ip):
  """Traces first router by routing information and interface ips"""

  possible_sources = []
  route = []
  processed_gws = []

  dFilter = {"src_addr": [ip] }

  # Filter flows that have given source IP
  filtered = filterFlows(flows, dFilter)


  # Iterate through all flows after filtering
  for flow in filtered:

    if_ip = getIfIpByDevice(flow.agent_addr, flow.if_ndx_in)

    # Checks if user want's to  trace gateways only once to increase performance
    # If statement could be placed before append, but here it's more efficient
    if trace_gws_only_once:
      if not if_ip in processed_gws:
         possible_sources.extend(traceGatewayIp(filtered, flow, if_ip))
        processed_gws.append(if_ip)
    else:
```

```
      possible_sources.extend(traceGatewayIp(filtered, flow, if_ip))

  return possible_sources

def traceGatewayIp(flows, flow, gw_ip):
  """Traces devices by gateway ip"""

  possible_sources = []

  gw_filter = {"gateway": [gw_ip]}
  gw_filtered = filterFlows(flows, gw_filter)

  if len(gw_filtered) == 0:
    possible_sources.append((flow.agent_addr, flow.if_ndx_in))
  else:
    possible_sources.extend(traceIpByGateway(flows, getIfIpByDevice(flow.agent_addr, flow.if_ndx_in)))

  return possible_sources


def filterFlows(flows, flowfilter, oper="or"):
  """
  Filter flows

  if not oper in ("and", "or", None):
    return

  Filter syntax:
    filter = {
      "src_addr": [],
      "src_port": [],
      "dst_addr": [],
      "dst_port": [],
      "proto": [],
      "agent_addr": [],
      "gateway": [],
      "if_in": [],
      "if_out": [],
    }
  Specify only those attributes that you want to filter by.
  """

  res = []
  for flow in flows:
    hits = 0


    if flowfilter.has_key('src_addr') and flow.src_addr in flowfilter['src_addr']: hits = hits + 1
    if flowfilter.has_key('src_port') and flow.src_port in flowfilter['src_port']: hits = hits + 1
    if flowfilter.has_key('dst_addr') and flow.dst_addr in flowfilter['dst_addr']: hits = hits + 1
    if flowfilter.has_key('dst_port') and flow.dst_port in flowfilter['dst_port']: hits = hits + 1
    if flowfilter.has_key('proto') and flow.protocol in flowfilter['proto']: hits = hits + 1
    if flowfilter.has_key('agent_addr') and flow.agent_addr in flowfilter['agent_addr']:
      hits = hits + 1
    if flowfilter.has_key('gateway') and flow.gateway_addr in flowfilter['gateway']: hits = hits + 1
    if flowfilter.has_key('if_in') and flow.if_ndx_in in flowfilter['if_in']: hits = hits + 1
```

```
    if flowfilter.has_key('if_out') and flow.if_ndx_out in flowfilter['if_out']: hits = hits + 1


    if oper == "and":
      if hits == len(flowfilter.keys()):
        res.append(flow)

    elif oper == "or":
      if hits > 0:
        res.append(flow)


  return res


def getProtocolName(prot):
  """Returns protocoll's name if defined"""

  if use_protocol_names and protocols.has_key(prot):
    return protocols[prot]
  else:
    return prot

def getPortName(port):
  """Returns port's name if defined"""

  if use_port_names and ports.has_key(port):
    return ports[port]
  else:
    return port


def printHeader():
  """Prints raport header"""

  print "%20s %5s %6s %7s %7s %7s %18s        %18s      %18s %7s %9s %3s" %
  ("time", "prot", "flags", "agent", "if in", "if out",  "source", "destination",
  "gateway", "packts", "octets", "ver")


def show(flows):
  """Print all flows in collection"""

  printHeader()

  for flow in flows:
        printFlow(flow)

  print "Total flowrecords: %d" % len(flows)

def printFlow(flow):
  """Print single flows  information"""

  agent = getDeviceName(flow.agent_addr)

  prot = getProtocolName(flow.protocol)
```

```
  src_port = getPortName(flow.src_port)

  dst_port = getPortName(flow.dst_port)

  gateway = getDeviceName(flow.gateway_addr)

  if_in = getIfNameByDevice(flow.agent_addr, flow.if_ndx_in)
  if_out = getIfNameByDevice(flow.agent_addr, flow.if_ndx_out)

  agent_time = datetime.datetime.fromtimestamp(flow.recv_sec)

  flags = getFlagString(flow.tcp_flags)

  print "%20s %5s %6s %7s %7s %7s %18s:%-6s %18s:%-6s %16s %7d %9d %3s" %
  (agent_time, prot, flags, agent, if_in, if_out, flow.src_addr, src_port,
  flow.dst_addr, dst_port, gateway, flow.packets, flow.octets, flow.netflow_ver)


#if __name__ == '__main__': main()
main()
```