Tyypillisen hyökkäyksen eteneminen verkossa

Tässä luvussa katsotaan, miten heikkouksia ja hyökkäyskohteita voidaan etsiä verkosta. Nykyään suuri osa hyökkäysten valmistelusta ja suorituksesta on automatisoitu. Toisaalta viime vuosina harmia aiheuttaneet botnet-verkot leviävät eri tavalla, mutta tässä ei tarkastella niitä.

Verkon rakenteen selvittäminen I

- Eräs hyökkääjän ensimmäisistä tavoitteista on saada tietoa verkon isäntäkoneista. Tämä tieto sisältää mm. ne IP-osoitteet, jotka ovat käytössä, koneiden käyttöjärjestelmät ja tarjotut palvelut.
- *Verkon kartoitus* tarkoittaa toimintaa, jolla saadaan selville verkon isäntäkoneet, kun taas *tunnustelu* (probing) tarkoittaa toimintaa, jolla pyritään saamaan tietoa yksittäisestä koneesta.
- Tyypillinen hyökkäys etenee siten, että ensin kartoitetaan verkkoa aktiivisten koneiden selville saamiseksi. Sen jälkeen koneita tunnustellaan. Tavoitteena on saada selville palvelu, jolla on tunnettuja heikkouksia turvallisuudessa. Lopuksi tehdään varsinainen hyökkäys valittua konetta ja palvelua vastaan.

Verkon kartoitus I

- Yksinkertaisin kartoitusmenetelmä on yleislähettää ping-sanoma ja katsoa, kuka vastaa. Jos kohdeverkko on luokan B verkko *10.10.x.x*, hyökkääjä lähettää esimerkiksi paketin

  ```
  11:42:16.33 attacker.com > 10.10.255.255 ICMP: Echo Reques
  ```

- Useita paketteja lähetetään, jolla varmistetaan, että mahdollisimman moni saa paketin. Lähdeosoitetta ei voi väärentää, sillä vastaussanomien täytyy saapua hyökkääjälle.
- Hyökkääjä voi kuitenkin lähettää samalla paketteja, joissa on väärennetty lähdeosoite. Tämä vaikeuttaa todellisen hyökkääjän selville saamista. Tällainen hyökkäys on helppo havaita ja palomuuri voidaan virittää siten, ettei se päästä läpi yleislähetyksiä.

Verkon kartoitus II

- Toinen tapa hyökätä on lähettää paketti jokaiselle mahdolliselle koneelle verkossa. Hyökkääjä valitsee TCP:tä käyttävän portin, josta pääsee palomuurin läpi kohdekoneille. Sen jälkeen hän alkaa lähettää paketteja jokaiseen mahdolliseen IP-osoitteeseen.
- Oletetaan, että palomuuri sallii telnet-yhteyden (portti 23) jokaisesta ulkopuolisesta IP-osoitteesta. Tällöin ensimmäiset paketit voisivat näyttää seuraavilta:

```
11:47:34.09 attacker.com.2213 > 10.10.1.1.23 S
11:47:34.19 attacker.com.2213 > 10.10.1.2.23 S
11:47:34.27 attacker.com.2213 > 10.10.1.3.23 S
    jne
```

- Tämäkin on helppo havaita, jos hyökkääjä on noin yksinkertainen. Jos kohdekoneet valitaankin satunnaisesti ja paketit lähetetään epäsäännöllisin välein, hyökkäys voi olla vaikea havaita.

Verkon kartoitus III

- Oveluudella on kuitenkin rajansa. Jos verkossa on 65 000 mahdollista konetta ja lähetetään yksi paketti minuutissa, kestää puolitoista kuukautta käydä kaikki läpi.
- Jos hyökkäjä on näin kärsivällinen, menetelmä saattaa toimia. Ei kuitenkaan tarvita kuin yksi valpas ylläpitäjä, joka näkee yhteyspyynnöt ja tulee uteliaaksi. Siten minuutin väli ole aina riittävä.
- On mahdollista hyökätä vielä älykkäämmin. Hyökkääjän kannalta on riittävää löytää pieni määrä koneita, jolloin hidas ja satunnaistettu kartoitus onnistuu paremmin. Etsintä lopetetaan, kun riittävä määrä koneita on löytynyt.
- On muitakin tapoja tehdä kartoitus vaikeasti havaittavaksi. Edellä kuvatuissa lähetyksissä käytettiin SYN-paketteja, joista pidetään kirjaa monissa koneissa. Samoin monet tunkeutumisen estojärjestelmät seuraavat näitä paketteja. Vaihtamalla lippua tehdään havaitseminen vaikeammaksi:

Verkon kartoitus IV

```
10:47:34.33 attacker.com.2213 > 10.10.17.121.23 S ack
11:13:21.24 attacker.com.2213 > 10.10.3.207.23 S ack
12:11:11.53 attacker.com.2213 > 10.10.51.14.23 S ack
```

- Tässä tapauksessa näyttää siltä, että hyökkäjä vain vastaa yhteyspyyntöihin verkon sisältä. Edelleen voidaan lisätä älykkyyttä muokkaamalla paketteja niin, että ne näyttävät tulevan Web-palvelimelta surfailun tuloksena (portti 80):

```
10:47:34.33 attacker.com.80 > 10.10.17.121.23 S ack
11:13:21.24 attacker.com.80 > 10.10.3.207.23 S ack
12:11:11.53 attacker.com.80 > 10.10.51.14.23 S ack
```

- Jotta tällainen hyökkäys havaittaisiin, tunkeutumisen estojärjestelmän tulee olla joko tilat muistava (muistaa, ettei ollut SYN-paketteja, joihin odotetaan vastauksia) tai pitää kirjaa kaikista aktiivisista verkon koneista.

Verkon kartoitus V

- Toinen vaihtoehto on lähettää reset-paketteja:

```
11:47:34.09 attacker.com.2213 > 10.10.17.121.23 R
11:53:43.12 attacker.com.2213 > 10.10.3.207.23 R
12:31:24.01 attacker.com.2213 > 10.10.51.14.23 R
```

- Reset-paketit ilmaisevat, että jotain on mennyt vikaan yhteyden solmimisessa. Nämä paketit läpäisevät useimmat palomuurit, joskin tilat muistava palomuuri pystyy päättelemään, ettei ollut mitään yhteysyrityksiä, joten reset-paketit voidaan hylätä.
- Näitä paketteja ei myöskään kirjata ylös niin usein kuin esim. SYN-paketteja. Ne ovat myös tarpeeksi yleisiä niin, etteivät tunkeutumisen estojärjestelmät kiinnitä niihin huomiota.
- Ellei systeemi ole tilansa muistava, on hyvät mahdollisuudet, että tällainen hyökkäys menee läpi. Reset-hyökkäyksistä on itse asiassa monia versioita. Näitä on käsitelty artikkelissa [**?**].

Sormenjäljet I

- Sormenjäljet tarkoittaa yksityiskohtia, joita käyttöjärjestelmät jättävät paketteihin. Ohjelmien avulla voidaan seurata näitä paketteja ja päätellä käyttöjärjestelmä.
- Sormenjälkien selvittely voidaan tehdä hyvin deterministisesti. Eri käyttöjärjestelmät reagoivat eri tavalla eri ärsykkeisiin, ja tätä voidaan käyttää hyväksi käyttöjärjestelmää ja sen versiota määriteltäessä.
- On kuitenkin tilanteita, joissa myös tilastollisilla menetelmillä on merkitystä. Hyvä yleislähde sormenjälkiin on [**?**].
- Perustekniikkka on lähettää paketti, jossa on outo lippukombinaatio. Eri KJ:t vastaavat outoihin kombinaatioihin eri tavalla.
- Esimerkiksi odottamaton FIN-paketti saa aikaan joissakin systeemeissä vastauksen, vaikkakin oikeaoppinen tapa olisi hylätä paketti. Vanhat Linux-versiot toistivat oudot lippukombinaatiot vastauspaketeissaan.

Sormenjäljet II

- Toinen tapa erotella käyttöjärjestelmiä on tutkia pakettien järjestysnumerointia. Numerointi voi olla satunnaista tai determinististä. Keräämällä tilastoa koneen lähettämistä paketeista voidaan koneita jakaa erilaisiin luokkiin.
- Tämä tapa on passiivista sormenjälkien tunnistamista, koska siinä ei lähetetä omia paketteja. Samoin se vaatii suuren määrän paketteja, ennenkuin johtopäätökisä voidaan tehdä.
- Monet KJ:t asettavat oletusarvoisesti don't fragment -bitin. Toiset taas asettavat ne vain tietyissä tilanteissa. Uudet KJ:t asettavat bitin useammin kuin vanhat.
- Käyttöjärjestelmillä on eri vaihtoehtoja virhetilanteista raportointiin ICMP:n avulla. Jos esimerkiksi kone saa suuren määrän paketteja suljettuun porttiin, sen ei tarvitse generoida sanomaa ICMP destination unreachable jokaiselle. Eri KJ:t tekevät erilaisia valintoja näissä tilanteissa.

Sormenjäljet III

- Valinnainen kenttä (options field) on hyvä lähde sormenjäljille. Koska kenttä on valinnainen, sen toteutus on vapaata. Siten kentän piirteet antavat paljon tietoa KJ:tä määriteltäessä.
- Muita yksityiskohtia, joita voidaan käyttää sormenjälkien tunnistamisessa, ovat TTL-kenttä (time to live), ikkunan koko, palvelun tyyppi ja urgent-osoitin. Nämä sopivat erityisesti passiiviseen lähetymistapaan.

Tunnustelu I

- Useilla palveluilla on tunnetut heikkoutensa, joita hyökkääjä voi käyttää hyväkseen. Eri versioilla on eri heikkoudet, joten ei riitä, että löydetään tietty palvelu tietyssä koneessa, vaan on lisäksi selvitettävä palvelun versio.
- Aluksi hyökkääjä selvittää palvelut porttiskannauksen avulla. Yksinkertaismillaan voidaan tutkia kaikki portit, mutta tämä on helppo havaita. Parempi menetelmä on tutkia vain ne portit, joita voidaan käyttää hyväksi:

```
13:12:22:33 attacker.com.2113 > 10.10.17.121.telnet S
13:12:22:54 attacker.com.2114 > 10.10.17.121.smtp S
13:12:22:73 attacker.com.2115 > 10.10.17.121.finger S
13:12:22:97 attacker.com.2116 > 10.10.17.121.http S
13:12:23:13 attacker.com.2117 > 10.10.17.121.imap S
```

Tunnustelu II

```
13:12:23:27 attacker.com.2118 > 10.10.17.121.rlogin S
13:12:23:41 attacker.com.2119 > 10.10.17.121.printer S
```

- Kun palvelu on löytynyt, helpoin tapa selvittää sen versio on katsoa sen lähettämää vastausta. Jos esimerkiksi kirjoitetaan komento `telnet mycomputer.com 25`, saadaan seuraavan kaltainen vastaus

  ```
  220 mycomputer.com ESMTP Sendmail 8.9.3/8.9.3; Sat, 8 D
  17:31:05-0500
  ```

  Komentamalla `help` tämän viestin kohdalla tuottaa selityksen
  214-This is Sendmail version 8.9.3
  214-Topics:
  214-HELO EHLO MAIL RCPT DATA
  214-RSET NOOP QUIT HELP VRFY
  214-EXPN VERB ETRN DSN
  214-For more info use "HELP <topic>".

Tunnustelu III

214-To report bugs in the implementation send email to
214-sendmail-bugs@sendmail.org.
214-For local information send email to postmaster at
214- your site
214 End of HELP info

Skannerit I

- Skannerilla tarkoitetaan tässä ohjelmaa, joka analysoi verkon yli toisia verkkoja ja niiden palvelimia ja palomuureja haavoittuvuuksien löytämiseksi.
- Skannerit ovat tehokas tapa löytää tietoturva-aukkoja ja niitä käytetäänkin paljon omia verkkoja analysoitaessa. Toisten verkkojen tutkiminen skannereilla on kiellettyä, mutta hyökkäyksissä sitä kuitenkin tehdään.
- Ensimmäinen skanneri oli Chris Klausin ISS (Internet Security Scanner) vuodelta 1992.
- Suurta huomiota herätti SATAN (Security Administrator Tool for Analyzing Networks, Dan Farmer ja Wietse Venema 1995).
- Nykyään skannerit ovat yhä kehittyneempiä. Uusimpia ovat mm. Nessus, SARA, Nikto, eEye ja Microsoft Baseline Security Analyser.

Esimerkkejä erilaisista palvelunestohyökkäyksistä

- Nämä hyökkäykset pyrkivät joko ajamaan alas verkon, tietokoneen tai prosessin tai muuten vaikeuttamaan palvelun tarjontaa.
- Seuraava lista ei ole täydellinen, sillä uusia palvelunestohyökkäyksiä keksitään lähes päivittäin. Tässä listassa on lähinnä sellaisia palvelunestohyökkäyksiä, jotka ovat esiintyneet aikaisemmin, jopa kauan sitten, ovat hyvin tunnettuja ja perustuvat TCP/IP-pinon heikkouksiin.

Maahyökkäys

- Maahyökkäyksessä (Land Attack) konstruoidaan TCP SYN -paketti, jossa kohde- ja lähdeosoite ovat samoja.
- Joissakin vanhoissa järjestelmissä tällainen paketti aiheutti vastaanottajan puolella lukkiuman, jonka johdosta kone täytyi käynnistää uudelleen. Hyökkäykseen tarvitaan siis vain yksi ainoa paketti.
- Hyökkäys ei liene kovin relevantti nykyisissä ympäristöissä, mutta ohjelmistojen uusiokäytön ja mahdollisten koodausvirheiden vuoksi se saattaa taas tulevaisuudessa putkahtaa esiin.
- Menetelmä on tyypillinen palvelunestohyökkäyksissä. Outo tai mahdoton paketti konstruoidaan ja lähetetään. Vastaanottopäässä tällainen paketti aiheuttaa hämmennystä, joka johtuu joko toteutusvirheestä tai protokollan epätäydellisyydestä.

Neptunus

- Neptunus tai SYN-tulva käyttää hyväkseen sitä tietoa, että jokaista puoliksi avoinna olevaa TCP-yhteyttä kohti tcpd (tcp-demoni) luo tietueen, jossa pidetään yhteystietoja.
- Jos yhteyttä ei luoda loppuun asti tietyn ajan kuluessa, yhteys katkaistaan ja tietueen varaama tila vapautetaan.
- Jos kuitenkin riittävä määrä yhteyksiä alustetaan ennen kuin ajastin laukeaa, tietorakenne vuotaa yli. Se aiheuttaa taulukon ylivuodon (segmentation fault) ja tietokoneen lukkiintumisen.
- Tässä hyökkäyksessä konstruoidaan paketti, jonka IP-lähdeosoite on saavuttamaton. Toisin sanoen mikään kone ei vastaa SYN/ACK-sanomaan, jonka kohdekone lähettää paketin saatuaan. Siten yhteys jää auki.

Ping O'Death

- Tämä hyökkäys koostuu ICMP echo-pyynnöstä (ping), jossa on liian pitkä (yli 64 KB) "hyötykuorma". Vanhemmat käyttöjärjestelmät lukkiintuvat tai käynnistyvät uudelleen, kun puskuri, johon paketti varastoidaan, vuotaa yli.
- Ensimmäiset Windows 95 -versiot sisälsivät ping-ohjelman, joka salli käyttäjän määritellä paketin koon, vaikka se olisikin ollut liian pitkä. Tämä teki järjestelmän suosituksi hyökkäyskohteeksi.
- Samoin kuin maahyökkäyksessä tämäkin hyökkäys vaatii vain yhden paketin. Nykyiset käyttöjärjestelmät eivät ole enää haavoittuvia tälle hyökkäykselle.

Prosessitaulu I

- Prosessitauluhyökkäys kehitettiin MIT:n Lincoln DARPA-laboratorioissa testaamaan tunkeutumisen estojärjestelmiä. Tavoitteena oli kehittää uusia hyökkäyksiä, jotta nähtäisiin, voitaisiinko ne paljastaa.
- Hyökkäyksen idea perustuu siihen, että joka kerran kun TCP-yhteyspyyntö saapuu, prosessi haarautuu (fork). Jos pyyntöjä saapuu hyvin paljon, prosessitaulu täyttyy. Kun taulu on täynnä, uusia prosesseja ei voida luoda. Seurauksena on tilanne, jossa tietokone ei voi tehdä mitään.
- Tämä hyökkäys täytyy kohdistaa prosesseille, jotka sallivat monia yhteyksiä yhtäaikaa. Esimerkiksi sendmail ei hyväksi uusia yhteyksiä, jos niitä on jo ennestään paljon.

Prosessitaulu II

- Sen sijaan finger sallii aina uusia yhteyksiä. Jotkut aikaisemmat fingerin versiot eivät käyttäneet ajastinta yhteyksien sulkemiseen, vaan ne jäivät auki niin pitkäksi aikaa, kunnes toinen sulki ne. Jokainen yhteys sai oman tunnuksensa ja jos riittävästi yhteyksiä avattiin, prosessitaulu täyttyi.

Targa3

Targa3-hyökkäyksessä lähetetään laittomia paketteja kohteelle. Nämä virheelliset paketit saavat jotkut systeemit kaatumaan. Vaikka systeemi ei kaatuisikaan, virheelliset paketit kuluttavat tavallista enemmän resursseja. Tyypillisesti paketeissa on vialla jotain seuraavista:

- Virheellinen paloittelu, protokolla, koko tai IP-otsakkeen arvo.
- Virheelliset optiot.
- Virheelliset TCP-segmentit.
- Virheelliset reitityslipput.

Smurf-hyökkäys

- Smurf-hyökkäyksessä on kolme osapuolta: hyökkääjä, kohde ja välittäjä, joka höynäytetään tekemään varsinainen hyökkäys.
- Hyökkääjä konstruoi echo request -paketteja, joissa on lähteenä aiottu kohde ja kohteena välittäjä. Nämä paketit yleislähetetään, jotta maksimoidaan vastausten lukumäärä.
- Kaikki koneet välittäjän verkossa vastaavat kohteelle, joka ei voi käsitellä niin suurta määrää paketteja. Kohde kaatuu tai ainakin hidastuu siinä määrin, ettei kykene toimimaan normaalisti.
- Hyökkääjä ei näy kohteen lokitiedostoissa. Hyökkääjä paljastuu vain välittäjän verkon lokeista.

Syslogd-hyökkäys

- Syslogd-hyökkäys tappaa syslogd-demonin Solariksen palvelimilla. Demonin vanhemmat versiot kaatuvat, jos niille annetaan lähdeosoite, jolla ei ole DNS-tietuetta. Siten hyökkäys koostuu syslog-porttiin lähetetyistä paketeista, joiden IP-osoitetta on väärennetty niin, ettei niillä ole DNS-tietuetta.
- DNS-tietueen puuttumista ei voida varmentaa muuten, kuin tekemällä itse DNS-haku. Tämä on tarpeetonta, sillä vain omien koneiden pitäisi olla yhteydessä syslog-tiedostoomme.
- Näin ollen koneet pitäisi pitää palomuurin sisällä, jolloin hyökkäys epäonnistuu, mikäli palomuuri on vähänkään tehokas. Palomuurin sisällä olevat vanhat, päivittämättömät Solaris-koneet eivät tälöin muodosta oleellista riskiä.

Teardrop-hyökkäys

Teardrop perustuu siihen, että jotkut vanhat TCP/IP-toteutukset eivät käsittele asianmukaisesti tilannetta, jossa paketti on pilkottu, mutta palaset eivät ole erillisiä. Jos koneessa on tämän vian sisältävä TCP/IP-toteutus ja hyökkääjä lähettää paketin, joka näyttää oikealta, mutta jonka palaset eivät ole erillisiä, kone kaatuu.

UDP-tulva

- UDP-tulva saa aikaan, että kaksi konetta hyökkäävät toisiaan vastaan. On tietty määrä portteja, jotka vastaavat paketilla saatuaan paketin. Echo (portti 7) ja chargen (portti 19) ovat tällaisia. Echo kaiuttaa paketin takaisin, kun taas chargen generoi merkkivirran.
- Tarkastellaan UDP-pakettia, jolla on lähdeporttina 7 ja kohdeporttina 19. Paketti generoi joitakin merkkejä kohdekoneesta, jotka merkit lähetetään lähdekoneen echo-porttiin.
- Lähde kaiuttaa nämä paketit takaisin, mikä puolestaan aiheuttaa lisää paketteliikennetta koneiden välille jne. Jossain vaiheessa molemmat koneet kuluttavat kaiken aikansa lähettämällä paketteja edestakaisin kunnes toinen tai molemmat kaatuvat.
- Jos ulkopuolelta tulee paketteja, joiden lähdeosoite on sisäpuolella, palomuurin pitäisi pysäyttää nämä paketit. Tällaiset paketit ovat melkein varmasti väärennettyjä. Jos kysymyksessä ei olisikaan hyökkäys, tällaiset paketit ovat merkki siitä, että jotain on mennyt vikaan.

Apache2-hyökkäys I

- Vanhat Apache Web-palvelimet saatiin kaatumaan, jos lähetettiin suuri määrä HTTP-pyyntöjä, joissa oli paljon otsakkeita.
- Tyypilliset HTTP-pyynöt sisältävät vähemmän kuin 20 otsaketta, kun taas hyökkäyksessä otsakkeita on tuhansia. Tämä aiheuttaa koneen keskikuormituksen dramaattisen nousun samoin kuin muistin käytön kasvun, ja tavallisesti kone kaatuu.
- Tällaisen hyökkäyksen paljastaminen edellyttää, että tutkitaan web-palvelimelle tulevat pyynnöt. Jos pyyntöjen normaali jakautuminen on tiedossa, poikkeavuudet on mahdollista havaita. Hyökkäyksestä kerrotaan enemmän www-sivulla *http://www.geek-girl.com/bugtraq/1998_3/0442.html*.

Back-hyökkäys

- Tämäkin on hyökkäys vanhoja Apache Web-palvelimia vastaan. Hyökkäyksessä lähetetään pyyntöjä, joissa on suuri määrä '/'-merkkejä, kuusi tai seitsemän tuhatta. Tämä aiheuttaa koneen tilapäisen hidastumisen. Kone toipuu, kun hyökkäys lakkaa.
- Hyökkäys on esimerkki menetelmästä, jota voidaan kutsua tyhmän käyttäjän hyökkäykseksi. Jos sovellusta ei ole suunniteltu käsittelemään outoja, mutta laillisia syötteitä, se voi joutua alttiiksi hyökkäyksille, joissa esimerkiksi käyttäjä pitää yhtä näppäintä alhaalla tai koskettaa otsallaan näppäimistöä.
- Käyttöliittymän yhteydessä pitäisi varautua tyhmiin käyttäjiin.

Postipommi

- Postipommi on hyökkäys käyttäjää vastaan, mutta se voi kaataa myös koneen. Hyökkäyksessä lähetetään monia viestejä jollekin käyttäjälle sähköpostiosoitteeseen.
- Jos viestien määrä on satoja, käyttäjä kärsii suuresti. Jos määrä on tuhansia ja viestit ovat pitkiä, postijono täyttyy ja kone voi kaatua. Myös levy voi täyttyä, mikä aiheuttaa myös muiden käyttäjien viestien tuhoutumisen tai ainakin välittämisen hidastamisen.
- Postipommi on helppo toteuttaa ja varsin suosittu. Vaikka postiosoite voidaan väärentää, IP-osoite pysyy aitona. Siten uhrin on mahdollista ottaa yhteys lähettävän koneen ylläpitoon ja pyytää keskeyttämään hyökkäys.

Web-pommi

- Tämä on samanlainen hyökkäys kuin postipommi, mutta lähetetään suuri määrä Web-pyyntöjä yhdelle Web-palvelimelle. Vaikutukset ovat samat kuin postipomissa.
- Tämäkin hyökkäys on helppo havaita, sillä suuri määrä pyyntöjä tulee yhdestä ja samasta paikasta. Koska sekä email että Web käyttävät TCP:tä, IP-lähdeosoitetta ei voi väärentää, sillä kättelyn täytyy onnistua ennen kuin sovellukset alkavat toimia.
- Tämän vuoksi hyökkääjät käyttävät usein välikättä, josta aloittavat posti- tai Web-hyökkäyksensä.
- Uudempi tapa on käyttää botnet-verkkoja hajautettuun hyökkäykseen. Tällaista hyökkäystä vastaan ei ole juurikaan keinoja suojautua.

Resurssien ahnehtiminen I

- Mikä tahansa ohjelma, joka kuluttaa koneen resursseja, voi olla palvelunestohyökkäys. Tällaiset ohjelmat vaativat yleensä sen, että hyökkääjällä on pääsy koneelle. Tarkastellaan esimerkiksi seuraavaa ohjelmaa:

```
#!/bin/csh
cd /tmp
while (true)
mkdir foo
cd foo
cp -r ~/* .
end
```

Resurssien ahnehtiminen II

- Ensimmäinen rivi näyttää, että kysymyksessä on cshell-ohjelma. Tämä ohjelma luo sarjan hakemistoja /tmp-hakemistoon täyttäen jokaisen käyttäjän kotihakemistolla.
- Tämä johtaa nopeasti levyn täyttymiseen ja toiminnan hidastumiseen kopioinnin johdosta. Ilman kopiointiakin ohjelma kaataa useimmat järjestelmät.
- Jos kuitenkin käytetään levykiintiöitä, yksittäinen käyttäjä ei pysty kuluttamaan koko levyymuistia.
- Käänteinen resurssien kulutukselle on kuuluisa komento

  ```
  rm -r *
  ```

  Se tuhoaa kaikki tiedostot hakemistossa ja alihakemistoissa. Todellinen palvelunestohyökkäys.

Palvelunestohyökkäysten estäminen I

- Palvelunestohyökkäyksessä vastapuoli estää laillisia käyttäjiä käyttämästä protokollaa täydellisesti.
- Käytännössä palvelunestohyökkäyksiä tehdään sellaisia palvelimia vastaan, joiden on tarkoitus palvella asiakkaita verkon yli.
- Nämä hyökkäykset voidaan jakaa hyökkäyksiin, jotka pyrkivät kuluttamaan palvelimen laskentaresurssit (resource depletion attacks), ja hyökkäyksiin, joiden tavoitteena on ylittää sallittu yhteyksien määrä (connection depletion attacks).
- Periaatteellisella tasolla näyttää siltä, ettei palvelunestohyökkäyksiä voida täysin estää. Jokainen yhteysyritys aiheuttaa sen, että joko yhteys hyväksytään tai se hylätään tietyn laskentamäärän jälkeen.
- On kuitenkin joitakin menetelmiä, joista on hyötyä pienennettäessä estohyökkäysten vaikutuksia. Toiset protokollat ovat haavoittuvaisempia estohyökkäyksille kuin toiset, joten hyökkäys ja sen torjuntamenetelmiä on aiheellista tuntea.

Auran ja Nikanderin ehdotus I

- Aura ja Nikander ovat ehdottaneet tilattomia yhteyksiä suojaamaan yhteyksien ylittymiseltä ([**?**]).
- Ideana on vaatia asiakasta tallentamaan kaikki tilatieto palvelimen sijasta ja palauttamaan tieto palvelimelle sanomien yhteydessä. Tällä tavoin palvelin säästyy tietojen varastoinnilta.
- On selvää, että palvelimelle palaavan tilatiedon tulee olla tarkistettavissa ja sen täytyy olla todennettavissa.
- Sen tulisi olla myös luottamuksellista. Tämä lisää prosessesointia kummassakin päässä.

Evästeet I

- Käytännöllisempi tapa on käyttää evästeitä. Tätä tapaa ovat ensin ehdottaneet Karn ja Simpson Photuris-protokollassaan (1. versio 1995, viimeisin 1999, RFC 2522).
- Kun asiakas yrittää solmia yhteyttä, palvelin lähettää takaisin evästeen. Menetelmä on samanlainen kuin www-palvelimissa käytetty, mutta nyt evästeillä on erityinen muoto: ne on muodostettu vain palvelimen tuntemasta salaisuudesta ja yhteystiedoista.
- Tässä vaiheessa palvelin ei talleta mitään yhteystietoja itselleen. Asiakkaan täytyy lähettää eväste seuraavassa sanomassa ja palvelin tarkistaa evästeen oikeellisuuden lähetetystä datasta ja evästeen sisältämästä salaisuudesta.
- Menetelmän tavoitteena on varmistaa, että asiakas todella aikoo solmia aidon yhteyden.

Evästeet II

- Menetelmä estää sellaiset palvelunestohyökkäykset, joissa yhteyspyyntöjä lähetetään satunnaisesti suuri määrä. Jos palvelimen salaisuutta vaihdetaan aika ajoin (joka 60. sekunti), ei edes sama asiakas voi tehdä rajoittamatonta määrää yhteyspyyntöjä.

Meadowsin ehdotus I

- Meadows ehdottaa yhteyksiin kohdistuvan hyökkäyksen torjumiseksi, että jokainen sanoma tulisi todentaa ([**?**]).
- Jotta turhaa laskentaa ei tulisi liikaa, todennuksen tulisi olla kevyttä protokollan alussa ja vahvistua myöhempien sanomien myötä.
- Evästeet, jotka palvelin lähettää ja jotka täytyy palauttaa, voivat alussa toimia todennuksena.
- Meadows on kehittänyt jopa formaalin kehyksen, joka perustuu Gongin ja Syversonin fail-stop -protokollille. Tällaiset protokollat lopettavat suorituksensa heti, kun epäaito sanoma on havaittu.

Juelin ja Brainardin ehdotus I

- Juels ja Brainard ehdottavat mekanismia, jota he kutsuvat nimellä client puzzles ([**?**]). Mekanismi muodostaa vahvemman todennuksen kuin evästeet.
- Ideana on, että kun palvelimen kuormitus kasvaa suureksi, kenties estohyökkäyksen johdosta, palvelin lähettää asiakkaille jonkin verran laskentaa vaativan probleeman, joka asiakkaan täytyy ratkaista ennen kuin uusi yhteys tehdään.
- Todellisille asiakkaille tästä aiheutuu vain vähän vaivaa, mutta estohyökkäyksen tekijä joutuu ratkomaan monia probleemoja.

Palomuurit

- Palomuurin perustehtäviin kuuluu tietoliikenteen seuraaminen, tunkeutumisyritysten rekisteröiminen ja hälyttäminen tarvittaessa. Lisäksi palomuuri muuttaa osoitteen yleisestä paikalliseen verkkosoitteeseen ja torjuu viruksia. Se myös tunneloi palomuurin läpi (IPSec, L2TP).
- Palomuuri ei voi suojata liikennettä, joka on luotu ohi palomuurin, esimerkiksi yksittäisillä modeemiyhteyksillä. Se ei myöskään suojaa sisäisiltä uhilta eikä se voi estää sovellusohjelmien virheitä hyväksikäyttäviä hyökkäyksiä.
- Jos palomuuriin tunkeudutaan, koko järjestelmä on vaarassa. Vaikka se osallistuu virusten torjuntaan, se ei poista erillisten virustorjuntaohjelmistojen tarvetta.

Paketteja suodattava palomuuri I

Paketteja suodattava reititin soveltaa sääntöjä jokaiseen tulevaan ja lähtevään IP-pakettiin. Jos paketti noudattaa määriteltyä turvapolitiikkaa, se reititetään eteenpäin, muuten se tuhotaan. Suodatussäännöt käyttävät hyväkseen paketin tietoja:

- kohdeosoite,
- lähdeosoite,
- TCP- tai UDP-porttinumero,
- IP-protokollakenttä, joka määrittelee kuljetusprotokollan,
- jos reitittimellä on kolme tai useampia portteja, niin suodatusperusteena voi olla myös, mihin porttiin paketti tulee tai mistä se lähtee.

Esimerkki. Sääntö

Paketteja suodattava palomuuri II

action	*our host*	*port*	*their host*	*port*
allow	*	*	*	25

sanoo, että mikä tahansa sisäverkon kone voi lähettäää postia ulkopuolelle. TCP-paketti, jonka kohdeportti on 25, ohjataan kohdekoneen SMTP-palvelimelle. Ongelmana tässä säännössä on, että ulkopuolella olevassa koneessa portti 25 voi liittyä myös muuhun sovellukseen. Tähän liittyy seuraava sääntö:

action	*src*	*port*	*dest*	*port*	*flags*
allow	our hosts	*	*	25	
allow	*	25	*	*	ACK

Paketteja suodattava palomuuri III

Kun yhteys on luotu, TCP-segmentin ACK-lippu nostetaan, jotta toiselta koneelta lähetetyt segmentit kuitataan. Siten tämä sääntö sanoo, että sallitaan IP-paketit, joiden lähdeosoite on jokin luetelluista koneista ja kohdeportti 25. Se myös sallii sisään tulevat paketit porttiin 25, joissa on ACK-lippu päällä. □

Pakettisuodatuksen heikkouksia

- Sovellustason haavoittuvuuksia hyväksikäyttäviä hyökkäyksiä ei välttämättä havaita.
- Kirjanpito on rajoittunutta.
- Useimmat tämänkaltaiset palomuurit eivät tue käyttäjän todennusta.
- Ne ovat haavoittuvia hyökkäyksille, joissa käytetään hyväksi TCP/IP-protokollaperheen heikkouksia, esimerkiksi osoiteväärennöksiä.
- On myös helppo tehdä virheitä palomuurin konfiguroinnissa, minkä johdosta epätoivottuja paketteja pääsee läpi.

Seuraavassa on puolestaan joitakin hyökkäysyrityksiä ja niiden torjuntakeinoja:

- Hyökkääjä väärentää IP-osoitteeksi sisäverkon osoitteen. Hyökkäys torjutaan, jos hylätään paketit, jotka tulevat ulkopuolelta ja joissa on sisäverkon osoite.
- Lähettävä kone määrittelee reitin, jota paketti noudattaa kulkiessaan internetin läpi. Tavoitteena on hämätä vastustajan palomuuria. Torjunnassa hylätään kaikki paketit, jotka käyttävät lähdereititystä.
- Hyökkääjä käyttää IP:n paloitteluoptiota ja luo hyvin pieniä paketteja ja pakottaa TCP:n otsaketiedon vähintään kahteen palaan. Tavoitteena on kiertää suodatussääntöjä, jotka käyttävät TCP:n otsaketietoja. Hyökkääjä toivoo, että vain ensimmäinen pala tutkitaan ja muut pääsevät läpi. Torjunnassa vaaditaan, että ensimmäisen palan on sisällettävä tietty minimimäärä otsaketta. Jos ensimmäinen pala hylätään, palomuuri muistaa paketin ja hylkää myös seuraavat palat.

Tilat muistava palomuuri I

- Jotta ymmärrettäisiin pakettisuodatuksen heikkoudet ja tilat muistavan palomuurin tarve, tarkastellaan SMTP:n toimintaa (Simple Mail Transfer Protocol).
- Se perustuu asiakas/palvelin-malliin. Asiakas luo uusia sähköposteja ja palvelin hyväksyy tulevat sähköpostit ja vie ne vastaaviin käyttäjien postilaatikoihin. SMTP perustaa TCP-yhteyden asiakkaan ja palvelimen välille. Palvelimen porttinumero on 25 ja asiakkaan välillä 1024-65535. Asiakkaan numeron luo asiakas itse.
- Tyypillisesti kun TCP:tä käyttävä sovellus luo istunnon kaukaisen koneen kanssa, se perustaa TCP-yhteyden, jossa kaukaisen koneen poprttinumero on pienempi kuin 1024 ja paikallisen asiakassovelluksen porttinumero on välillä 1024-65535. Lukua 1024 pienemmät numerot edustavat hyvin tunnettuja protokollia. Luvut väliltä 1024-65535 luodaan dynaamisesti ja ne ovat voimassa vain TCP-istunnon ajan.

Tilat muistava palomuuri II

- Yksinkertaisen paketteja suodattavan palomuurin täytyy päästää sisään kaikki paketit, joiden porttinumero on välillä 1024-65535. Tätä voivat hyökkääjät käyttää hyväkseen. Tilat muistava palomuuri pitää kirjaa TCP-yhteyksistä. Jokaista luotua yhteyttä kohti on yksi tietue. Palomuuri päästää läpi vain ne paketit, jotka sopivat yhteen tietokannan tietojen kanssa.

Sovellustason yhdyskäytävä

- *Sovellustason yhdyskäytävä* (application-level gateway, proxy) välittää sovellustason liikennettä.
- Käyttäjä ottaa yhteyden yhdyskäytävään TCP/IP-sovelluksen avulla ja yhdyskäytävä kysyy sen kaukaisen koneen nimen, jonka kanssa käyttäjä haluaa kommunikoida. Kun käyttäjä vastaa ja todentaa samalla itsensä, yhdyskäytävä ottaa yhteyden kaukaiseen koneeseen ja ryhtyy välittämään paketteja käyttäjän ja kaukaisen koneen välillä. Jos yhdyskäytävä ei tue jotain palvelua, käyttäjä ei voi sitä käyttää. On myös mahdollista, että palvelusta voidaan käyttää vain tiettyjä osia.
- Sovellustason yhdyskäytävät ovat turvallisempia kuin paketti suodattimet, koska yhdyskäytäville voidaan määritellä vain muutamia sovelluksia, joista ne huolehtivat. On lisäksi helpompaa seurata liikennettä sovellustasolla. Haittana on ylimääräiseen prosessointiin kuluva aika.

Piiritason yhdyskäytävä I

- *Piiritason yhdyskäytävä* (circuit-level gateway) ei salli päästä-päähän TCP-yhteyksiä. Sen sijaan se perustaa kaksi TCP-yhteyttä, yhden itsensä ja paikallisen käyttäjän välille ja toisen itsensä ja kaukaisen koneen välille.
- Kun nämä yhteydet on perustettu, yhdyskäytävä välittää toisen yhteyden paketit suoraan toiselle yhteydelle tutkimatta sisältöä tarkemmin. Turvallisuus syntyy siitä, että yhdyskäytävä päättää, mitä TCP-yhteyksiä sallitaan.
- Piiritason yhdyskäytäviä sovelletaan tyypillisesti tilanteissa, joissa ylläpito luottaa sisäverkon käyttäjiin.

Piiritason yhdyskäytävä II

- Yhdyskäytävä voidaan konfiguroida sovellustason yhdyskäytäväksi sisään tulevan liikenteen suhteen ja piiritason yhdyskäytäväksi ulospäin menevän liikenteen suhteen. Tässä konfiguraatiossa yhdyskäytävä joutuu tutkimaan sisääntulevan liikenteen kiellettyjen toimintojen osalta, mutta sen ei tarvitse tehdä samaa ulosmenevän liikenteen suhteen.

Vallikone I

Vallikone (bastion host) on systeemi, joka suunnitellaan erityisen turvalliseksi. Tyypillisesti se toimii sovellustason tai piiritason yhdyskäytävän alustana. Sen ominaispiirteitä ovat:

- Vallikoneen käyttöjärjestelmä on erityisen turvallinen.
- Vain tarpeelliset palvelut ovat käytössä.
- Vallikone saattaa vaatia ylimääräistä autentikointia ennenkuin käyttäjä pääsee proxy-palveluihin.
- Jokainen proxy on konfiguroitu toteuttamaan vain osaa sovelluksen käskyjoukosta.
- Kukin proxy pitää yksityiskohtaista lokikirjaa liikenteestä, liitynnöistä ja kunkin liitynnän kestosta.
- Jokainen proxy-sovellus on oma pieni, suhteellisen yksinkertainen ja erityisesti verkon turvaamiseen suunniteltu ohjelmisto.

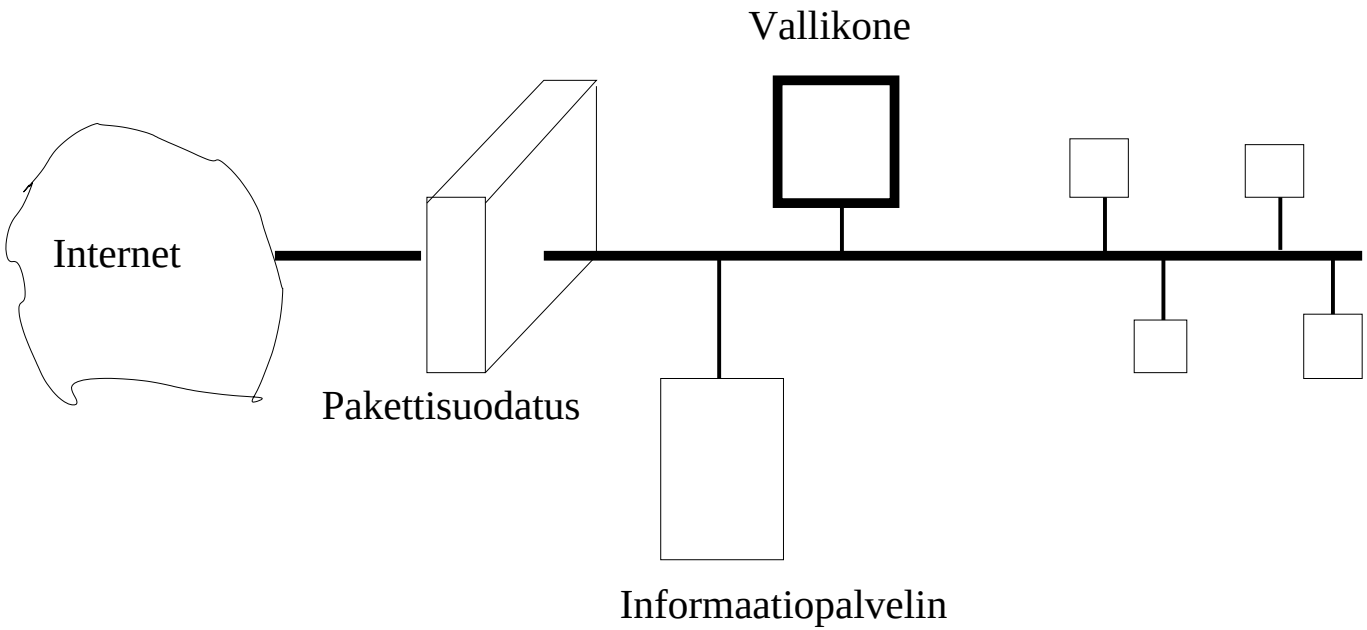
Vallikone II

- Kukin vallikoneella oleva proxy on itsenäinen, toisista proxyistä riippumaton.
- Proxy ei normaalisti tee muita levyoperaatioita kuin lukee oman konfiguraationsa.
- Kukin proxy on oikeudeton käyttäjä vallikoneen yksityisessä ja varmistetussa hakemistossa.

Tyypillisiä palomuuriratkaisuja I

Yksinkertainen palomuuriratkaisu on kuvassa 1. Siinä on pakettisuodatus ja vallikone. Internetistä vain vallikoneelle tuleva liikenne on sallittu. Verkon sisältä vain vallikoneelta ulosmenevä liikenne on sallittu. Vallikoneella on todennus- ja proxy-toiminnot. Järjestelmässä on kaksi turvaporttia, mutta esimerkiksi web-palvelimella on suora yhteys ulkomaailmaan.

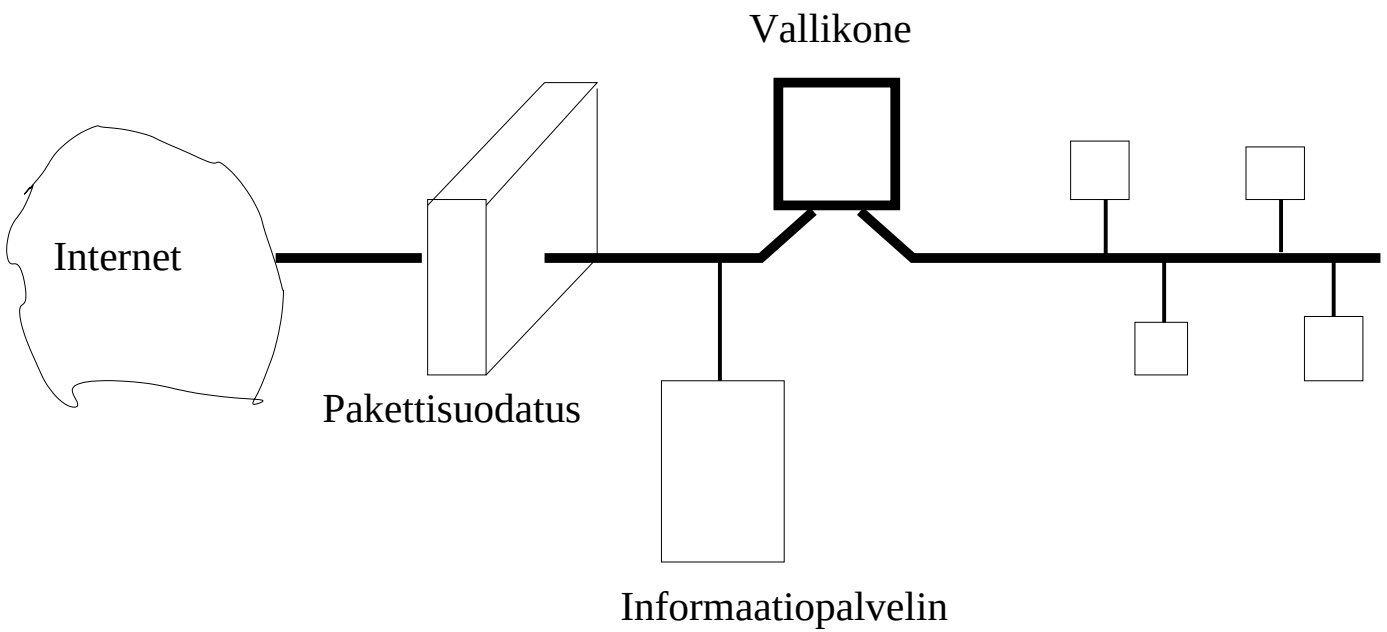
Tyypillisiä palomuuriratkaisuja II



Kuva: Yksinkertainen yhdyskäytävä

Kuvassa 2 on kaksoisyhdyskäytävä. Sillä on samat ominaisuudet kuin yksinkertaisella yhdyskäytävällä, mutta ulkoinen ja sisäinen verkko ovat fyysisesti erossa toisistaan. Ei ole mahdollista päästä sisään poikkeamatta vallikoneessa.

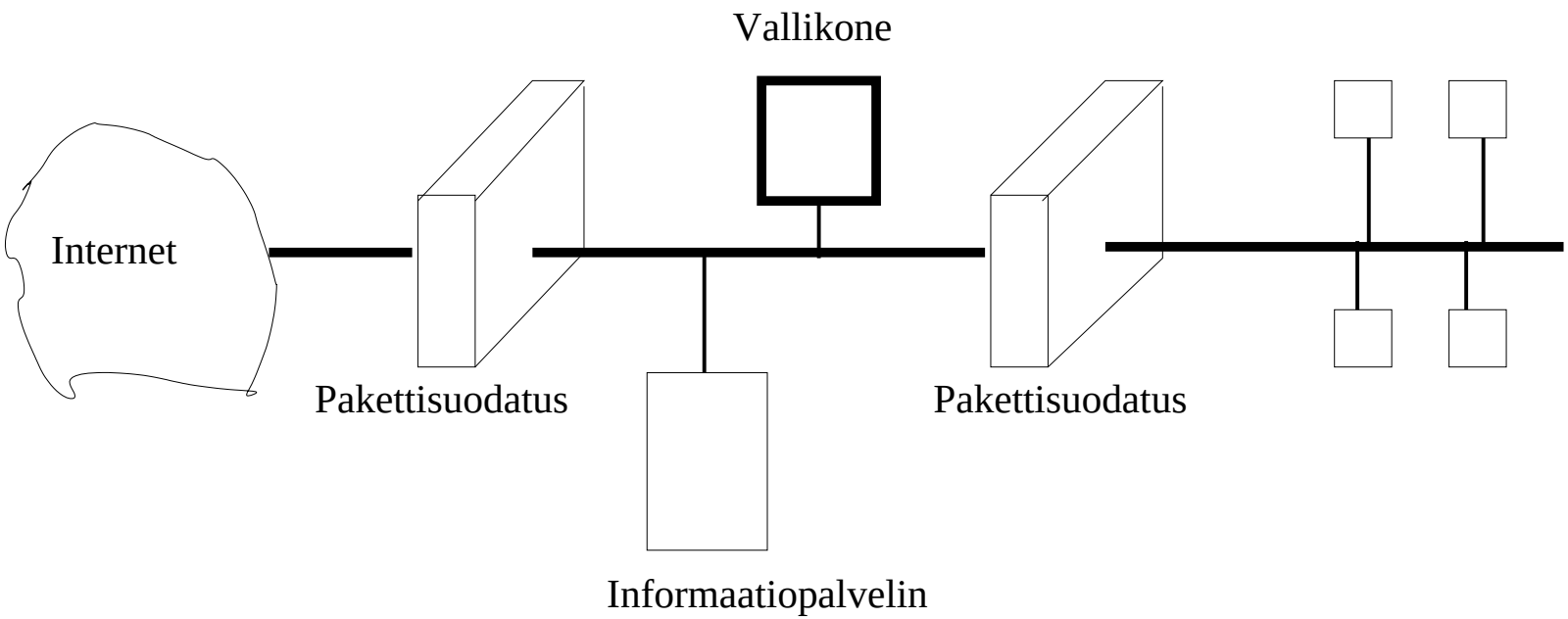
Tyypillisiä palomuuriratkaisuja III



Kuva: Kaksoisyhdyskäytävä

Tyypillisiä palomuuriratkaisuja IV

Kuvassa 3 on rajoitusaliverkkopalomuuri. Se on turvallisin näistä kolmesta. Sen avulla luodaan aliverkko ja liikenne aliverkon läpi ei ole sallittua. Se sisältää kolme suojaustasoa. Sisisäinen verkko on näkymätön ulkopuoliselle maailmalle. Vallikone mainostaa itseään ulkopuolisena verkkona, joten sisäpuolelta ei pääse ulos kuin vallikoneen kautta.



Kuva: Rajoitusaliverkkopalomuuri

Osapuolten todentaminen I

- Osapuolten todentaminen on yksi tärkeimmistä tehtävistä hajautetuissa järjestelmissä. Monessa tilanteessa todentaminen täytyy automatisoida, joten esimerkisi etukäteen jaettujen salasanojen käyttö ei ole mahdollista.
- Monet todentämistekniikat perustuvat digitaaliseen allekirjoitukseen, joka puolestaan perustuu julkisen avaimen salaukseen.
- Lisäksi on olemassa menetelmiä, joilla yhteinen salaisuus voidaan generoida automaattisesti kahden osapuolen välille, joilla ei ole etukäteen mitään yhteistä salaisuutta. Tällaista tarvitaan tilanteissa, joissa täytyy lähettää luottamuksellista tietoa toiselle osapuolelle, vaikka osapuolilla ei ole ennestään mitään yhteistä salaista avainta.

Julkisen avaimen salaus RSA I

- Julkisen avaimen salauksen eli *epäsymmetrisen salauksen* edellytyksenä on, että löytyy salausmenetelmä, jossa on mahdotonta saada selville salaista purkuavainta D_K, vaikka salausavain E_K tunnetaan.
- Julkisen avaimen salauksen etuna on, että kuka tahansa voi lähettää salakirjoitetun viestin vastaanottajalle ilman, että molempien täytyy sopia etukäteen salaisesta avaimesta. Vastaanottaja on kuitenkin ainoa, joka pystyy purkamaan viestin salaisella avaimellaan.
- Julkisen avaimen salauksen idean julkaisivat ensimmäisinä Diffie ja Hellman vuonna 1976. Joissakin lähteissä mainitaan myös Merkle samanaikaisena keksijänä.
- Heidän ehdottamansa menetelmä oli kuitenkin lähinnä teoreettinen ja varsin epäkäytännöllinen.

Julkisen avaimen salaus RSA II

- Ensimmäinen julkisen avaimen julkinen käytännöllinen menetelmä oli RSA, jonka kehittivät Rivest, Shamir ja Adleman 1977. RSA on edelleenkin laajimmin käytössä oleva julkisen avaimen järjestelmä.
- Vuonna 1997 CESG (brittiläinen kryptografian osasto) julkaisi dokumentteja, jotka osoittivat, että James Ellis oli jo vuonna 1970 keksinyt julkisen avaimen salauksen.
- Samoin Clifford Clocks oli 1973 kuvannut RSA:n version, missä salauseksponentti e oli sama kuin modulus n.
- RSAn jälkeen on ehdotettu monia muita, joista tärkeimmät ovat:
  - *Merklen ja Hellmanin selkäreppu*. NP-täydelliseen selkäreppuongelmaan perustuva menetelmä, joka on kuitenkin osoittautunut epävarmaksi. Versioita on ollut useita ja vain Chorin ja Rivestin versio on kestänyt toistaiseksi murtoyritykset.
  - *McEliecen järjestelmä* perustuu algebralliseen koodausteoriaan.

Julkisen avaimen salaus RSA III

- *Elliptisiin käyriin perustuva menetelmä*. Elliptinen käyrä on kompleksitasossa määritelty, tietynlainen toisen asteen käyrä. Kompleksilukujen sijasta voidaan käyttää äärellisiä kuntia, jolloin käyrän pistejoukko on äärellinen. Tätä joukkoa voidaan käyttää hyväksi salauksessa. Etuna RSA:aan verrattuna on lyhyempi avaimen pituus.

- Julkisen avaimen menetelmät eivät voi taata turvallisuutta joka tilanteessa. Jos vastapuolella nimittäin on salateksti, hän voi salata jokaisen mahdollisen selvätekstin ja verrata tulosta salatekstiin. Mikäli tulos on sama, on selväteksti paljastunut. Siten mahdollisia selvätekstejä on oltava suunnaton määrä.

RSA I

RSA:ssa avaimet ovat muodoltaan seuraavia:

- julkinen avain on lukupari (e, n);
- salainen avain on lukupari (d, n);
- selväteksti jaetaan lohkoihin, lohkon koon binäärilukuna tulee olla alle n eli lohko käsittää korkeintaan $\log_2(n)$ bittiä.

Salaus tapahtuu kaavalla

$$C = M^e \mod n.$$

Purkuun käytetään kaavaa

$$M = C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n.$$

Ennenkuin systeemi toimii,

- on löydettävä sopivat luvut e, d ja n,

RSA II

- on laskettava tehokkaasti M^e ja C^d kaikilla $M < n$,
- d ei saa olla helposti laskettavissa e:stä ja n:stä.

Luvut e, d ja n valitaan seuraavasti.

1. Generoidaan kaksi suurta alkulukua p ja q.
2. Lasketaan $n = pq$ ja $\Phi(n) = (p-1)(q-1)$.
3. Valitaan satunnaisluku e siten, että $1 < e < \Phi(n)$ ja $\mathrm{syt}(e, \Phi(n)) = 1$.
4. Lasketaan $d = e^{-1} \mod \Phi(n)$ (ts. $de = 1 \bmod \Phi(n)$).
5. julkaistaan e ja n.

Valinnoista johtuu toivottu tulos

$$(M^e \mod n)^d \mod n = M.$$

Tämän osoittaminen vaatii lukuteorian perustuloksia kuten esimerkiksi *Fermat'n pientä lausetta*. Perustulokset on osoitettu useissa tietoturvan ja kryptografian oppikirjoissa.

Esimerkki

- $p = 101$, $q = 113$, $n = 11413$, $\Phi(n) = 100 \cdot 112 = 11200$.
- Valitaan ensin e. Koska $11200 = 2^6 5^2 7^1$, niin e ei saa olla jaollinen 2:lla, 5:llä tai 7:llä. Olkoon $e = 3533$.
- Silloin $e^{-1} = 6597$ modulo 11200.
- Julkinen avain on siis $(3533, 11413)$.
- Olkoon $M = 9726$. Salateksti saadaan laskemalla $9726^{3533} \mod 11413 = 5761$.
- Purku tuottaa alkuperäisen selvätekstin: $5761^{6597} \mod 11413 = 9726$.

Vaatimukset avaimille I

Tällä hetkellä vaatimukset avaimille ovat:

- n:n on oltava välillä $1024 - 2048$ bittiä,
- p:n ja q:n tulee olla "lähellä"toisiaan, välillä $10^{75} - 100^{100}$,
- $p-1$:n ja $q-1$:n tulee sisältää suuri alkulukutekijä ja
- $\mathrm{syt}(p-1, q-1)$:n tulee olla pieni.

Tarkastellaan vielä avainten pituuksia. Koska laitteistot kehittyvät koko ajan, täytyy avaimen pituuksia kasvattaa aika ajoin. Seuraavassa taulukossa on sekä symnmetrisiltä että epäsymmetrisiltä salausmenetelmiltä vaadittavia avainpituuksia tuleville vuosille. Arviot perustuvat ECRYPT II -suosituksiin vuodelta 2009. Vastaavia suosituksia on NIST:ltä, NSA:lta ja monilta muilta sekä yksittäisiltä tutkijoilta että organisaatioilta. Taulukossa DL tarkoittaa diskreettiin logaritmiiin perustuvia tekniikoita ja EC elliptisiin käyriin perustuvia.

Vaatimukset avaimille II

Taso	Suoja	Symm.	Epäsymm
1	Attacks in "real-time" by individuals	32	
2	Very short-term protection against small organizations	64	816
3	Short-term protection against medium organizations	72	1008
4	Smallest general-purpose level, 2009 to 2012	80	1248
5	Legacy standard level, 2009 to 2020	96	1776
6	Medium-term protection from 2009 to 2030	112	2432
7	Long-term protection from 2009 to 2040	128	3248
8	"Foreseeable future", against quantum computers	256	15424

Digitaalinen allekirjoitus RSA:n avulla I

- Esimerkiksi elektronisessa kaupankäynnissä ja varmenteiden käytössä täytyy kyetä osoittamaan, että tietty taho on viestin lähettäjä. *Digitaalinen allekirjoitus* tähtää siihen, että se osoittaa kiistatta lähettäjän ja lähetyksen ajankohdan sekä todentaa sanoman.
- Lisäksi kolmannen osapuolen tulee kyetä verifioimaan allekirjoitus. Todentaminen tässä yhteydessä tarkoittaa, että allekirjoitettua viestiä ei voi muuttaa vaikuttamatta allekirjoitukseen.
- *Suora digitaalinen allekirjoitus* perustuu julkisen avaimen salaukseen. Edellytyksenä on, että salausmenetelmälle pätee ehto

$$E_{K_p}(D_{K_s}(M)) = M.$$

- Esimerkiksi RSA toteuttaa yllä olevan kaavan. Jos ehto pätee, sanoma allekirjoitetaan "salaamalla"se lähettäjän salaisella avaimella D_{K_s}.

Digitaalinen allekirjoitus RSA:n avulla II

- Vastaanottaja purkaa salauksen lähettäjän julkisella avaimella ja toteaa, että tuloksena on selväkielinen viesti.
- Jos lähettäjä haluaa, ettei viestiä voi lukea muut kuin vastaanottaja, viesti voidaan vielä salata vastaanottajan julkisella avaimella.
- Oletetaan, että vastustaja valitsee ensin luvun y_1 ja muodostaa sitten sanoman $m_1 = y_1^{e_A}$.
- Nyt A ei voi kieltää, etteikö hän allekirjoittanut sanomaa m_1. Toisaalta on hyvin epätodennäköistä, että m_1 on mielekäs sanoma. Se on luultavasti satunnainen bittijono.
- Usein allekirjoituksen yhteydessä käytetään myös tiivistefunktiota. Tiiviste allekirjoitetaan, ei alkuperäistä lähdettä.
- On suuri joukko muita tekniikoita, joita voidaan käyttää allekirjoitukseen. *ElGamalin allekirjoitus* perustuu diskreetin logaritmin ongelmaan. Se ei ole kovin käytännöllinen, sillä allekirjoitus on varsin pitkä.

Digitaalinen allekirjoitus RSA:n avulla III

- *Digitaalisen allekirjoituksen standardi* on muunnos ELGamalista. Siinä allekirjoitus on kohtuullisen kokoinen. Se kehitettiin osaksi sen vuoksi, että RSA:n käyttöä säätelivät patentit.
- Näiden perustekniikoiden lisäksi on lukuisia muita, joita on lueteltu teoksessa Menezes, van Oorschot, Vanstone: *Handbook of Applied Cryptography,* joka on saatavissa ilmaiseksi verkosta.

Julkisen avaimen infrastruktuuri ja X.509 I

- Jotta julkisen avaimen salausta ja digitaalista allekirjoitusta voitaisiin käyttää, osapuolten julkiset avaimet on löydettävä. Lisäksi on oltava varma, että tietty julkinen avain on juuri tietyn osapuolen avain. Jos näin ei olisi, yksi osapuoli voisi lähettää salaista tietoa kolmannelle osapuolelle tietämättään.
- Julkisten avainten löytyminen on asia, jota ei vielä ole ratkaistu täydellisesti. Paremmin, joskaan ei täydellisesti, on sen sijaan ratkaistu, miten varmistaudutaan julkisen avaimen omistajasta. Yleisesti käytetään varmenteita, jotka luotettu kolmas osapuoli on varmentanut. Tavallisin varmennesysteemi on X.509.
- X.509 on yksi osa X.500-suositussarjassa, joka määrittelee hakemistopalvelun. X.509 määrittelee hakemiston todennuspalvelut. X.509 on tärkeä standardi, koska X.509:n julkisiin avaimiin liittyvät varmenteet ja todennusprotokollat ovat käytössä ainakin ohjelmistoissa S/MIME, IP:n turvapalvelu, SSL/TLS ja SET.

Julkisen avaimen infrastruktuuri ja X.509 II

- X.509 julkaistiin 1988. Myöhemmin standardia muokattiin vastaamaan paremmin turvallisuusvaatimuksia. Uudistettu versio julkistettiin 1993 ja kolmas versio 1995.
- X.509 perustuu julkisen avaimen salaukseen ja digitaalisiin allekirjoituksiin. Standardi suosittelee RSA:ta, muttei vaadi sitä. Digitaalinen allekirjoitus perustuu tiivistefunktion käyttöön. Sitäkään ei ole kiinnitetty.
- X.509:n perustana on julkisen avaimen varmenne, joka liittyy jokaiseen käyttäjään. Oletuksena on, että varmenteet on luonut jokin luotettava osapuoli (CA, Certification Authority, *varmenneviranomainen*). Joko CA tai käyttäjä on pannut varmenteet hakemistoon. Hakemistopalvelin itse ei ole vastuussa julkisten avainten luonnista eikä varmenneperiaatteesta.
- Varmenne sisältää seuraavat osat:

Julkisen avaimen infrastruktuuri ja X.509 III

- *Versio*: Oletusarvo on 1. Jos *Initiator Unique Identifier* tai *Subject Unique Identifier* ovat läsnä, version pitää olla 2. Jos yksi tai useampi laajennus on läsnä, versio on 3.
- *Sarjanumero*: Kokonaisluku, yksikäsitteinen CA:n alueella.
- *Allekirjoitusalgoritmin tunnus*: Tämä tieto toistetaan Signature-kentässä varmenteen lopussa, joten sillä ei ole merkitystä tässä kohdassa.
- *Julkaisijan nimi*: Sen CA:n X.500-nimi, joka loi ja allekirjoitti tämän varmenteen.
- *Voimassaoloaika*: Alku- ja loppuajankohta.
- *Subjektin nimi*: Käyttäjän nimi, johon tämä varmenne viittaa.
- *Subjektin julkisen avaimen informaatio*: Subjektin julkinen avain, algoritmin tunnus ja parametrit.
- *Julkaisijan yksikäsitteinen tunnus*: Valinnainen bittijono, joka määrittelee yksikäsitteisesti CA:n siinä tapauksessa, että samaa X.500-nimeä on käytetty eri olioille.
- *Subjektin yksikäsitteinen tunnus*: Kuten edellä.

Julkisen avaimen infrastruktuuri ja X.509 IV

- *Laajennukset*: Yksi tai useampi laajennuskenttä. Laajennukset lisättiin versioon 3. Ne selitetään myöhemmin.
- *Allekirjoitus*: Kattaa varmenteen kaikki muut kentät. Sisältää toisten kenttien tiivistekoodin salattuna CA:n yksityisellä avaimella. Kenttä sisältää allekirjoitusalgoritmin tunnuksen.

- Standardi käyttää seuraavaa merkintää varmenteen määrittämiseksi:

$$CA << A >>= CA\{V, SN, AI, CA, T_A, A, A_P\},$$

missä $Y << X >>$ on käyttäjän X varmenne, jonka on julkaissut varmenneviranomainen Y, ja $Y\{I\}$ on Y:n muodostama I:n allekirjoitus.

- Se koostuu I:stä, johon on lisätty salakirjoitettu tiivistekoodi. CA allekirjoittaa varmenteen salaisella avaimellaan. Jos vastaava julkinen avain on käyttäjälle tuttu, niin käyttäjä voi varmistua, että varmenne on todella CA:n vahvistama.

Julkisen avaimen infrastruktuuri ja X.509 V

- Kuka tahansa käyttäjä, joka voi käyttää CA:n julkista avainta, voi ottaa toisen käyttäjän varmennetun julkisen avaimen itselleen.
- Ainoastaan varmenneviranomainen voi muokata varmennetta ilman, että tämä paljastuisi. Koska varmenteet ovat väärentämättömiä, ne voidaan asettaa hakemistoon ilman sen kummempia suojauksia.
- Jos kaikki käyttäjät käyttävät samaa CA:ta, kaikki varmenteet voidaan asettaa hakemistoon kaikkien käyttäjien saataville. Lisäksi käyttäjä voi lähettää varmenteensa suoraan muille käyttäjille.
- Kummassakin tapauksessa B:n omistaessa A:n varmenteen B voi luottaa siihen, että salaus A:n julkisella avaimella on turvallista ja A:n allekirjoitukset väärentämättömiä.
- Jos käyttäjiä on paljon, yhteen CA:han turvautuminen ei ole käytännöllistä.

Julkisen avaimen infrastruktuuri ja X.509 VI

- Koska CA allekirjoittaa varmenteet, jokaisella käyttäjällä täytyy olla kopio CA:n julkisesta avaimesta. Tämä julkinen avain täytyy toimittaa jokaiselle käyttäjälle absoluuttisen turvallisesti.
- Siten jos käyttäjiä on monia, on edullisempaa, jos varmenneviranomaisia on useampia, joista jokainen toimittaa oman julkisen avaimensa osalle käyttäjiä.
- Oletetaan, että A on saanut varmenteen CA_1:ltä ja B CA_2:lta. Jos A ei varmasti tunne CA_2:n julkista avainta, niin CA_2:n julkaisema B:n varmenne on hyödytön A:lle.
- A voi lukea B:n varmenteen, muttei voi verifioida allekirjoitusta. Jos kuitenkin CA_1 ja CA_2 ovat turvallisesti vaihtaneet omia julkisia avaimia, niin A voi saada B:n julkisen avaimen seuraavasti:
  1. A hankkii hakemistosta CA_1:n allekirjoittaman CA_2:n varmenteen. Koska A turvallisesti tuntee CA_1:n, A saa CA_2:n julkisen avaimen kyseisestä varmenteesta, joka on CA_1:n vahvistama.

Julkisen avaimen infrastruktuuri ja X.509 VII

2. A palaa hakemistoon ja hankkii CA_2:n allekirjoittaman B:n varmenteen. Koska A tuntee nyt varmasti CA_2:n julkisen avaimen, A voi verifioida allekirjoituksen ja turvallisesti hankkia B:n julkisen avaimen.

- Edellä kuvatussa menetelmässä A on käyttänyt varmenteiden ketjua hankkiessaan B:n julkisen avaimen. X.509:n merkinnöin tämä ketju ilmaistaan kaavalla

$$CA_1 << CA_2 >> CA_2 << B >> .$$

Samalla tavalla B voi hankkia A:n julkisen avaimen käänteisellä ketjulla

$$CA_2 << CA_1 >> CA_1 << A >> .$$

Menetelmä ei rajoitu kahteen varmenteeseen. Ketju voi olla mielivaltaisen pitkä

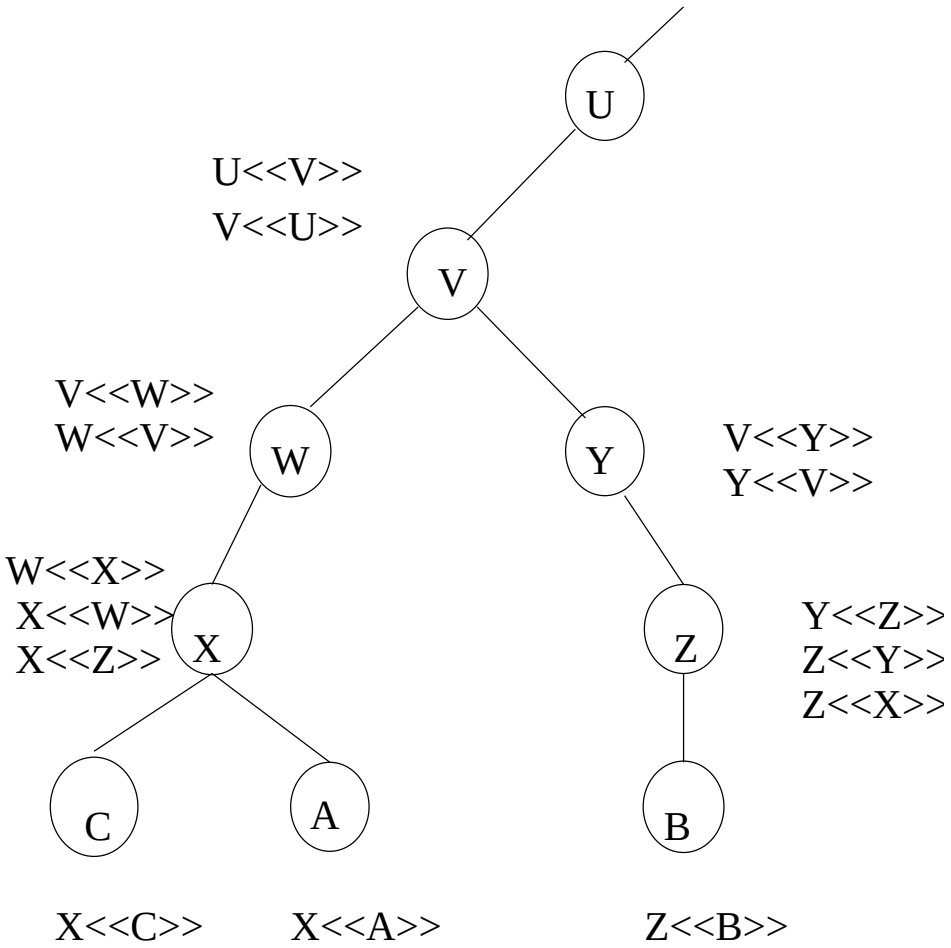
$$CA_1 << CA_2 >> CA_2 << CA_3 >> \cdots CA_n << B >> .$$

Julkisen avaimen infrastruktuuri ja X.509 VIII

Jokaisen parin (CA_i, CA_{i+1}) yllä olevassa ketjussa tulee luoda varmenteet toisilleen.

- Kaikkien näiden varmenteiden tulee olla hakemistossa ja käyttäjän pitää tietää, kuinka ne on linkitetty seuratakseen polkua toisen käyttäjän julkisen avaimen varmenteeseen.
- X.509 ehdottaa, että CA:t järjestetään hierarkkisesti siten, että navigointi on suoraviivaista. Seuraavassa kuvassa on esimerkki hierarkiasta.

Julkisen avaimen infrastruktuuri ja X.509 IX



Kuva: CA-hierarkia

Julkisen avaimen infrastruktuuri ja X.509 X

- Kuvassa U, V, W jne ovat varmenneviranomaisia. Laatikot osoittavat varmenteita, joita kyseinen CA ylläpitää hakemistossaan. CA:n hakemistoalkio sisältää kahden tyyppisiä varmenteita:
  - *Etuvarmenteet* (forward): Muiden varmenneviranomaisten generoimat X:n varmenteet.
  - *Käänteisvarmenteet* (reverse): X:n generoimat varmenteet, jotka ovat muiden varmenneviranomaisten varmenteita.

  Esimerkissä A voi hankkia seuraavat varmenteet hakemistosta saadakseen B:n varmenteen:

  $$X << W >> W << V >> V << Y >> Y << Z >> Z << B >> .$$

  Kun A on saanut nämä varmenteet, se voi purkaa ja todentaa B:n varmenteen.

Julkisen avaimen infrastruktuuri ja X.509 XI

- Varmenteella on päättymisajankohta. Uusi varmenne julkistetaan juuri ennen kuin edellinen vanhenee. Seuraavissa tilanteissa varmenne on aiheellista peruuttaa jo ennen sen vanhenemista.
  1. Käyttäjän salainen avain on luultavasti joutunut väärii käsiin.
  2. CA ei enää varmenna kyseistä käyttäjää.
  3. CA:n varmenne ei ole enää luotettava.
- Jokainen CA ylläpitää listaa kaikista peruutetuista varmenteista, jotka eivät vielä ole vanhentuneita.
- Nämä varmenteet käsittävät sekä CA:n käyttäjille myönnetyt että toisille varmenneviranomaisille vahvistetut varmenteet.

Julkisen avaimen infrastruktuuri ja X.509 XII

- Jokainen peruutettujen varmenteiden lista (CRL, certification revocation list) sijoitetaan hakemistoon. Sen allekirjoittaa julkaisija ja se sisältää julkaisijan nimen, listan luontipäivämäärän, seuraavan CRL:n ilmestymisajankohdan ja alkion jokaista peruutettua varmennetta kohti. Alkio sisältää sarjanumeron ja peruutuspäivämäärän.
- Kun käyttäjä saa varmenteen sanomassa, käyttäjän täytyy ratkaista, onko varmenne peruutettu vai ei. Käyttäjä voi tarkistaa asian hakemistosta joka kerran.
- Viiveitä välttääkseen käyttäjä voi myös ylläpitää paikallista käteismuistia varmenteita, listoja ja peruutettuja varmenteita varten.

Versio 3 I

Versiossa 2 oli puutteita:

1. Subjektikenttä on riittämätön välittämään avaimen omistajan identiteettiä julkisen avaimen käyttäjälle. X.509-nimet ovat lyhyitä, eivätkä anna täyttä tietoa nimen haltijasta.
2. Subjektikenttä on myös riittämätön monille sovelluksille, jotka tunnistavat olion sen sähköpostiosoitteen, URL:n tai muun Internetiin liittyvän tunnuksen avulla.
3. On tarvetta näyttää informaatiota turvapolitiikasta. Tämä mahdollistaisi turvasovelluksen tai -funktion, kuten IPSec, ja X.509-varmenteen yhteensovittamisen.
4. On tarvetta rajoittaa vahinkoja, joita virheellinen tai pahantahtoinen CA voi aiheuttaa. Vahinkoja voidaan vähentää asettamalla rajoituksia tietyn varmenteen käytettävyydelle.

Versio 3 II

5. On tärkeää kyetä tunnistamaan erikseen saman käyttäjän eri avaimet eri aikoina. Tämä piirre tukee avaimen elinkaaren hallintaa.

Yllä esitettyjä vaatimuksia on toteutettu ottamalla käyttöön valinnaisia laajennuksia, joita voidaan lisätä version 2 formaattiin. Jokainen laajennos sisältää *laajennuksen tunnuksen*, *kriittisyysindikaattorin* ja *laajennusarvon*.

Kriittisyysindikaattori osoittaa, voidaanko laajennus jättää turvallisesti huomiotta. Jos indikaattorilla on arvo true ja toteutus ei sisällä laajenusta, varmennetta tulee pitää virheellisenä. Varmenteen laajennukset jakautuvat kolmeen osaan:

- avain- ja käyttötiedot,
- subjektin ja julkaisijan attribuutit,
- varmenteen polkurajoitukset.

Avain- ja käyttötiedot sisältävät seuraavia kohtia:

Versio 3 III

- *Avainauktoriteetin tunnus* identifioi sen julkisen avaimen, jota voidaan käyttää verifioimaan tämän varmenteen tai CRL:n allekirjoitus.
- *Subjektin avaimen tunnus* identifioi varmennetun julkisen avaimen. Hyödyllinen päivitettäessä subjekti-avain -paria. Myös subjektilla voi olla useita avaimia ja vastaavasti erilaisia varmenteita eri tarkoituksiin.
- *Avaimen käyttö* osoittaa rajoitukset, joita avaimella on. Esimerkiksi avainta voidaan käyttää digitaaliseen allekirjoitukseen, kiistämättömyyden osoittamiseen, avaimen salaukseen, datan salaukseen, avaimesta sopimiseen, CA:n allekirjoituksen verifioimiseen varmenteissa tai CRL:ssä.
- *Yksityisen aivaimen käyttöaika.*
- *Varmennepolitiikka*. Varmenteita voidaan käyttää ympäristöissä, joissa voidaan soveltaa monentyyppistä käyttöpolitiikkaa. Tämä laajennus luettelee politiikat, joita varmenne tukee.

Versio 3 IV

- *Käyttötapakuvaukset* ovat ainoastaan CA:n julkistamissa toisen CA:n varmenteissa. Kuvaus osoittaa, että yhtä tai useampia julkaisijan politiikkoja voidaan pitää ekvivalentteina subjektin alueella käytetyn politiikan kanssa.

Attribuutit ovat seuraavia:

- *Subjektin vaihtoehtoiset nimet* ovat tärkeitä tuettaessa joitakin sovelluksia kuten e-mail, EDI, IPSec.
- *Julkaisijan vaihtoehtoiset nimet.*
- *Subjektin hakemistoattribuutit* välittävät minkä tahansa X.500-hakemiston attribuutin arvon.

Rajoitteita voidaan asettaa CA:n julkaisemille toisen CA:n varmenteille:

- *Perusrajoitteet* osoittavat, voiko subjekti toimia CA:na. Jos voi, varmentamisen polun pituus voidaan asettaa.

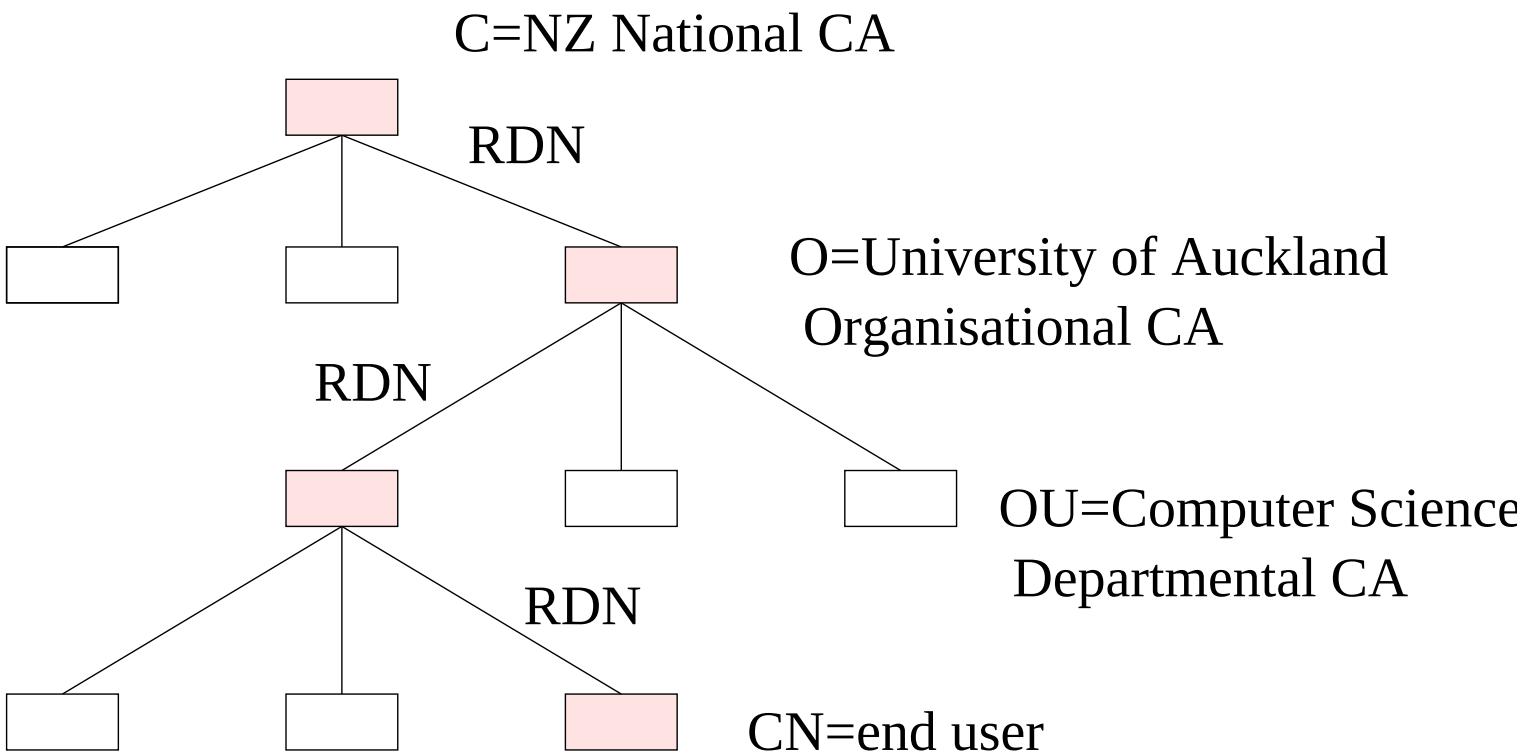
Versio 3 V

- *Nimirajoitteet* määrittelevät nimiavaruuden, josta varmennepolun subjektin tulee olla.
- *Käyttörajoitukset*.

X.509:n ongelmia I

- Seuraavassa käydään läpi julkisen avaimen infrastruktuurin, erityisesti X.509:n, ongelmia. Esitys perustuu lähteeseen [**?**].
- X.509:n alkuperäisenä tavoitteena oli ratkaista X.500-hakemiston pääsynvalvonta. X.500 perustui hierarkkiseen tietokantamalliin, jossa määriteltiin polkuja käyttäen sarjaa suhteellisia RD-nimiä (relative distinguished name, RDN).
- Sarja RD-nimiä muodostaa puolestaan D-nimen (distinguished name, DN). Polun päässä on sitten tietue, jonka attribuutit sisältävät varsinaisen tiedon. Kuviossa 5 on eräs konkreettinen tilanne.

X.509:n ongelmia II

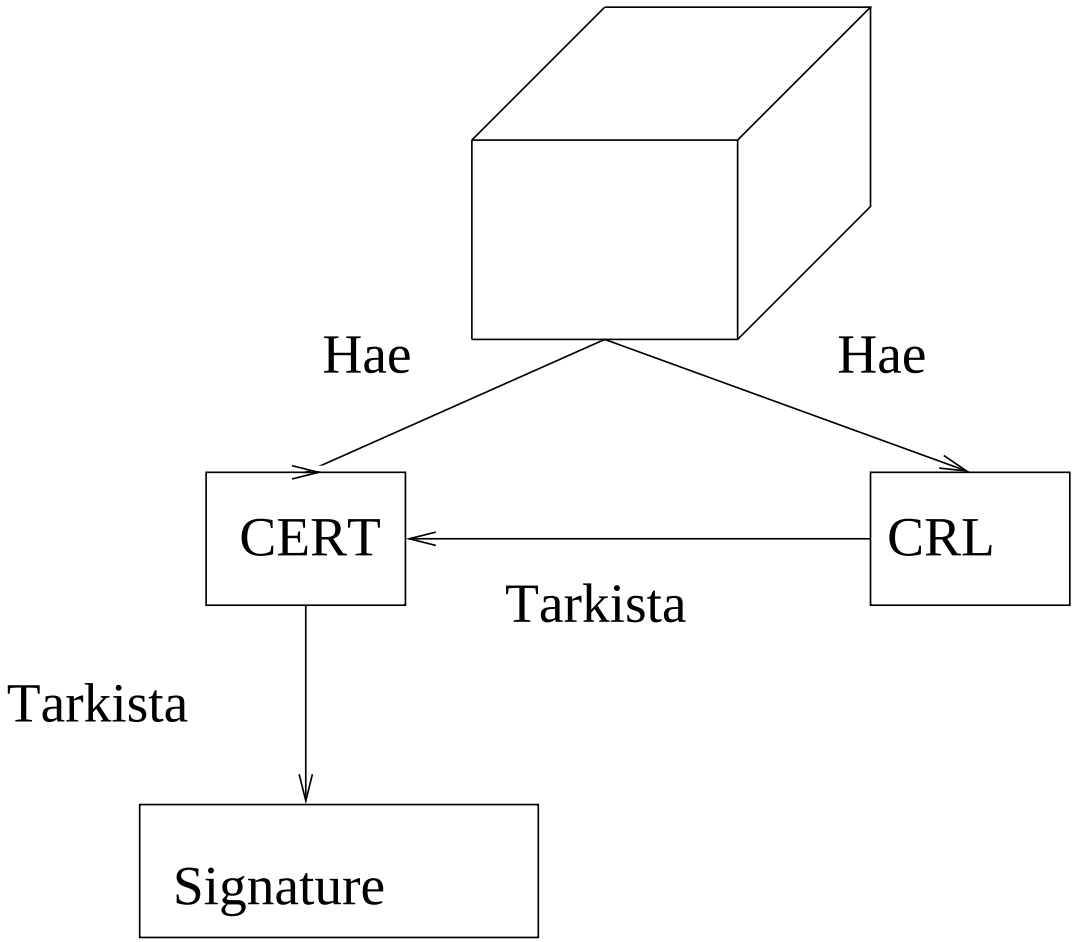


Kuva: X.509:n lähtökohta

X.509:n ongelmia III

- Pääsynvalvonnaksi suunnniteltiin erilaisia mekanismeja, salasanoista digitaalisiin allekirjoituksiin. Allekirjoitusten tapauksessa määriteltiin, että tietokannan eri osilla on omat CA:nsa, jotka luovat varmenteet pääsynvalvontatarpeisiin.
- Alkuperäisen X.509:n version 1 rakenteessa nämä lähtökohdat näkyvät hyvin selvästi. On myöntäjä DN ja subjekti DN, voimassaoloaika ja julkinen avain.
- Mikään ei osoita, kuuluuko varmenne CA:lle vai muulle (koska hakemistossa tämä määräytyy implisiittisesti).
- Ei käy myöskään selville, mihin avainta voidaan käyttää (koska alunperin oli vain yksi käyttötarkoitus, pääsynvalvonta). Ei määritelty edes minkä periaatteen mukaan varmenteet oli myönnetty.
- Nykymuodossa X.509:ää käytetään digitaalisen allekirjoituksen verifiointiin suoran todennuksen asemesta kuvion 6 mukaisesti.

X.509:n ongelmia IV



Kuva: X.509:n käyttö todennuksessa

X.509:n ongelmia V

- Varmennetta käytetään yleensä valtuutukseen hajautetussa ympäristössä: "Saako käyttäjä U suorittaa operaation O resurssiin R?" Tällainen valtuutus vaatisi kolme mekanismia: todennuksen, valtuutuksen ja jälkikäteisen tarkistuksen (audit), eli AAA:n.
- Kerberos suunniteltiin AAA:n pohjalta, mutta lopulta toteutettiin vain todennus ja puolet valtuutuksesta. (Tähän vaikutti julkisen avaimen salauksen tulo markkinoille samaan aikaan.)
- X.509 tuottaa todennuksen, muttei valtuutusta eikä jälkikäteistä tarkistusta.
- Kuvio 6 sisältää monia ongelmia. Suurin ongelma on, ettei ole selvää, mistä varmenne pitäisi hakea. (Globaalia X.500 hakemistoa ei koskaan toteutettu.)
- Ei ole myöskään selvää, mistä haetaan CRL. Käytäntö onkin nykyään, että ohjelmisto sisältää kaikki tarvittavat varmenteet.

X.509:n ongelmia VI

- Uuden varmenteen hankkiminen tapahtuu joko ulkopuolisesti (esim. postitse) tai laiskan evaluoinnin perusteella, jolloin sovellus pitää kirjaa kaikista kohtaamistaan varmenteista myöhempää tarvetta varten.
- Tämä käytäntö ratkaisee varmenteiden jakeluongelman, mutta tekee CRL:n käsittelyn paljon hankalammaksi.
- Vaikka tietäisikin, mistä hakemistosta etsiä varmenteita, ei ole selvää, mitä DN:ää tulisi käyttää tai mikä saman nimisistä on oikea valinta.
- Globaalisti yksikäsitteiset nimet eivät näytä realistisilta. X.509 tähtäsi globaaleihin nimiin, mutta DN-käsitettä ei ymmärretty käytännössä. Tämä johti lokaaleihin nimiin ja DN:t olivat mitä sattuivat. X.509v3 tosin lisäsi vaihtoehtoisen identiteetin varmenteeseen.
- Varmenteiden peruutukseen liittyvät ehkä suurimmat ongelmat:
  - Peruutuslistoja ei päivitetä tarpeeksi tiheästi.
  - Ei tiedetä, mistä CRL haetaan.

X.509:n ongelmia VII

- Haku kestää kauan.
- On tarpeen julkaista uusi CRL usein. Jos se tapahtuu minuutin välein, niin käytännössä se riittää, mutta aiheuttaa paljon rasitetta ja tarjoaa mahdollisuuden palvelunestohyökkäykselle.
- Jos tunnin tai päivän välein, niin ei hyötyä sovelluksille.

- CRL-tarkistukset ovat ilmaisia. Jos käyttäjiä on kymmenisä tuhansia, CRL:t saattavat olla monen megatavun kokoisia. Kansallisen PKI:n CRL:n koko olisi mahtava.
- Ilmaisuus johtaa siihen, ettei CRL:stä välitetä. Esimerkiksi Ruotsin kansalaisvarmenteet ovat ilmaisia, mutta CRL-tarkistus maksoi 25 senttiä 2002.
- Monessa tapauksessa varmenteet vanhenevat samaan aikaan ja kaikki hakevat CRL:n yhtäaikaa. Olisi parempi porrastaa vanhenemisajat. Miten tämä järjestettäisiin ilman, että luotettavuus vähenee, on hankalampi kysymys.

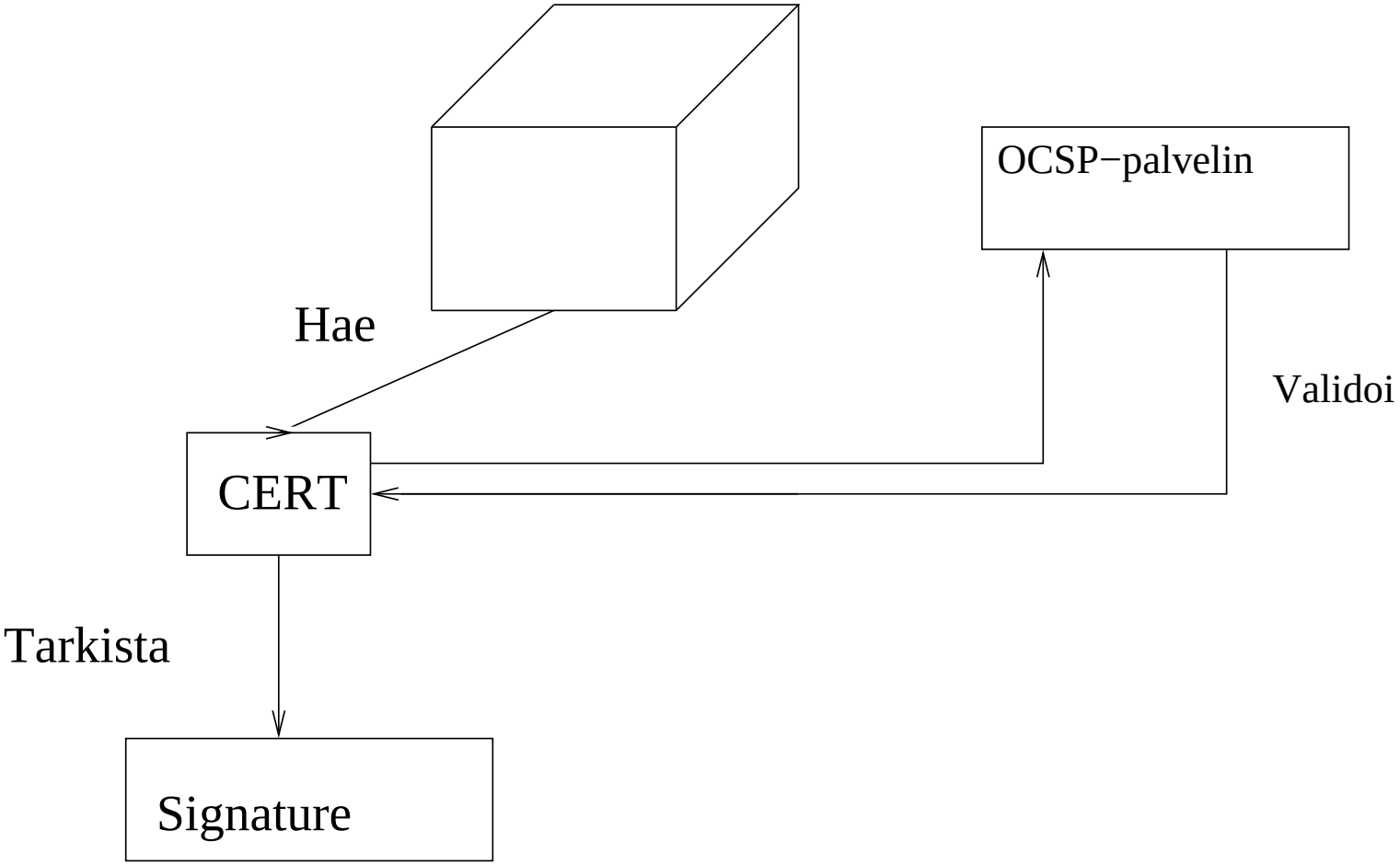
X.509:n ongelmia VIII

- Toinen mahdollisuus olisi käyttää useita päällekkäisiä peruutuslistoja. Jos tarvitsija noutaa CRL:n hetkellä n, sille annetaan hetkellä $n + 5$ vanheneva lista.
- Kolmas mahdollisuus on segmentoida CRL kiireellisyyyden perusteella. Esim. avaimen paljastuminen olisi kiireellisin, työntekijän siirtyminen toiselle osastolle vähemmän kiireellinen. Tämä saattaa aiheuttaa tietoturvaongelman, jos avaimen anastaja vaatii vastaavan varmenteen sijoittamista alempaan kiireellisyysluokkaan.
- Neljäntenä on vielä käyttää yhtä isoa, pitkäaikaista CRL:ää ja samaan aikaan joukkoa pienempiä lyhytkestoisia CRL:iä, jotka modifioivat varsinaista CRL:ää. Tästä ei näytä olevan kovin paljon käytännön kokemuksia.

Varmenteiden hallintaprotokollat I

- Varmenteiden hallintaan on kehitetty joukko protokollia. OCSP:ssä (Online Certificate Status Protocol) on palvelin, joka vastaa peruutuksia koskeviin kyselyihin.
- Palvelin perustaa vastauksensa joko CRL:ään tai johon muuhun tietoon. Kuva 7 näyttää OCSP:n toiminnan.

Varmenteiden hallintaprotokollat II



Kuva: OCSP

Varmenteiden hallintaprotokollat III

- Nyt asiakkaan ei tarvitse hakea koko CRL:ää kerralla, vaan kyselyt kohdistuvat yksittäisiin varmenteihin.
- Toisaalta asiakas ei voi valmistella CRL:n käyttöä eräajona. Jotta toiminta olisi taloudellisesti mahdollista, CA:n tulee laskuttaa OCSP-kyselyistä. Tästä syystä kyselyt tulee allekirjoittaa identifioimista varten.
- OCSP:llä on monia hyviä puolia CRL:ään verrattuna, mutta myös ongelmia. Tärkein ongelma on vastausten laatu, joka voi olla "not-revoked" (= "good" in OCSP), "revoked" ja "unknown".
- "Not-revoked" ei välttämättä merkitse samaa kuin OK. "Unknown" voi merkitä mitä tahansa väliltä "varmennetta ei ole luotu" ja "varmenne on luotu, mutta sille ei löytynyt CLR:ää".

Varmenteiden hallintaprotokollat IV

- Osittain tämä epämääräisyys johtuu alkuperäisestä CRL-mekanismista, sillä CRL voi ainoastaan antaa negatiivisen tiedon. Sekä OCSP että CRL kysyvät väärän kysymyksen "onko tämä varmenne peruutettu", kun niiden pitäisi kysyä "onko tämä varmenne voimassa".
- Tätä on yritetty oikaista toisissa protokollissa kuten SCVP (Simple Certificate Validation Protocol) ja DVCS (Data Validation and Certification Server Protocols). Harjoitustehtävänä on tutustua näihin hieman tarkemmin.
- Lisäksi on paljon muita protokollia, joiden tavoitteena on antaa tietoa varmenteiden statuksesta: ICAP (Integrated CA Services Protocol), RCSP (Real-Time Certificate Status Protocol), WebCAP (Web-based Certificate Access Protocol), PARP (Peter's Active Revocation Protocol), OpenCDP (Open CRL Distribution Process), DCS (Directory Supported Certificate Status Options).

Varmenneketjujen ongelmia I

- Varmenteiden tarkistuksissa täytyy kulkea polku lehdestä hierarkiassa ylemmällä olevaan CA:han. Tämä voi johtaa ongelmiin, kun CA:t varmentavat toisiaan.
- Voi olla esimerkiksi useita polkuja lehdestä CA:han, ja polut voivat sisältää silmukoita. Pahimmassa tapauksessa varmenteen semantiikka voi muuttua silmukan eri kierroksilla.
- *Silta-CA:t* välttävät nämä ongelmat jossain määrin lisäämäällä yhden superjuuren, joka liittää kaksi tai useampia CA:ita yhteen.
- Monissa nykyisissä ohjelmistoissa on se käytäntö, että luotetaan kaikkiin varmenneviranomaisiin. Tämä mahdollistaa väärinkäytökset, sillä on helppoa perustaa uusi CA.

Varmenneketjujen ongelmia II

- Esimerkiksi selaimissa on yli sata varmenneviranomaista, joiden poistaminen vaatisi yli 600 hiiren klikkausta. Monet näistä viranomaisista ovat tuntemattomia. Ne saattavat käyttää lyhyitä julkisia avaimia ja niiden varmenteiden elinaika voi olla 40 vuotta. Saattaa myös olla, että alkuperäinen yritys on myynyt toiminnan kolmannelle osapuolelle.
- Käytössä on myös järjestemiä, joissa varmenteen ja peruutuslistan haku yhdistetään. Voidaan myös joissain tilanteissa luopua jopa varmenteista, jos voidaan hakea julkisen avaimen kopio joltain luotettavalta osapuolelta. Tarkemmat analyysit ja esimerkit sivuutetaan.

IPSec I

- IPSec on IP-verkkoprotokollien laajennus, millä estetään IP-pakettien urkkiminen ja muuntaminen. IPSec on syntynyt uuden IPv6-protokollan yhteydessä ja IPv6 onkin IPSec:in luonteva alusta. IPSec voidaan kuitenkin sovittaa myös IPv4-protokolliin.
- Verkkotason suojaus ei vaikuta sovellusohjelmiin tai sovellusprotokolliin ja IPSec-paketteja voivat käsitellä jo käytössä olevat reitittimet ja reitittävät isäntäkoneet. IPSec:iä käytetään nykyisin erityisesti virtuaalisten yksityisten verkkojen toteutukseen.
- Yritys voi rakentaa turvallisen virtuaalisen yksityisen verkon Internetin tai julkisen WAN-verkon yli. Tämä mahdollistaa sen, että yritykset voivat luottaa Internetiin ja säästää yksityisen verkon perustamis- ja käyttökustannukset.
- Loppukäyttäjä, jolla on IPSec implementoituna, voi ottaa paikallisen yhteyden Internetin palveluntarjoajaan, jota kautta hän voi edelleen saada turvallisen yhteyden yrityksensä suljettuun verkkoon.

IPSec II

- IPSec:iä voidaan käyttää varmistamaan kommunikointi toisten organisaatioiden kanssa niin, että todennus ja luottamuksellisuus taataan.

IPSec-arkkitehtuurin yleiskuva I

- IPSec on varsin monimutkainen ja terminologiakin on erikoista. Protokolla on suunniteltu toteuttamaan luottamuksellisuus (salauksen avulla) ja todennus.
- Kummallekin suojaustavalle on määritelty oma otsikkonsa, koteloitu salattu data ja todennusotsikko. Yksi ja sama IP-paketti voi sisältää yhden tai molemmat otsikot riippuen tarvittavasta turvapalvelusta.
- Todennusotsikko (AH, Authentication Header) sisältää eheyden tarkistustietoa, millä voidaan tarkistaa, onko paketti väärennös tai onko sitä muutettu matkalla epäluotettavan verkon läpi.
- Otsikko sisältää tätä varten tarkistussumman. Tarkistussumma sisältää salaista tietoa, josta syystä ulkopuolinen ei pysty laskemaan toista tarkistussummaa, mikä osoittaisi sisällön aitouden.
- Koteloitu salattu data -otsikkoa (ESP, Encapsulating Security Payload) käyttämällä salataan paketin loppuosan datasisältö.

IPSec-arkkitehtuurin yleiskuva II

- ESP-otsikon muoto vaihtelee sen mukaan, mitä salausalgoritmia käytetään. Kaikissa tapauksissa käytettävä salausavain valitaan parametrin SPI (security parameter index) avulla.
- IPSec-protokolla koostuu siten kahdesta versiosta, joista ensimmäinen kattaa pelkästään todennuksen todennusotsikon avulla. Toinen versio on yhdistetty todennus- ja salausprotokolla, jonka yhteydessä käytetään otsikkoa koteloitu salattu data yksinään tai todennusotsikon kanssa, jos halutaan salauksen lisäksi todennus.
- IPSec tarjoaa kuitenkin enemmän kuin pelkästään yksinkertaisen todennuksen ja salauksen. Seuraavassa on lueteltu IPSec:in tarjoamat palvelut:
  - Pääsynvalvonta.
  - Yhteydetön eheys.
  - Datan alkuperään liittyvä todennus.
  - Toistohyökkäysten torjunta.
  - Luottamuksellisuus.

IPSec-arkkitehtuurin yleiskuva III

- Rajoitettu liikennevirran luottamuksellisuus.

Turvayhteydet I

- Turvayhteydet (security associations) on avainsana toteutettaessa todennusta ja luottamuksellisuutta. Kummankin IPSec-suojaukseen pyrkivän koneen tulee muodostaa aluksi turvayhteys toinen toiseensa.
- Turvayhteys määrittelee, mitä ja miten IPSec-suojausta käytetään, eli mitä turvapalvelua milloinkin käytetään, miten salaus ja/tai todennus suoritetaan ja mitä avaimia pitää käyttää. Eli turvayhteys sisältää kaiken sen informaation, mitä tarvitaan luotettavan yhteyden määrittelemisessä ja toteutuksessa.
- IETF:n dokumentit käsittelevät turvayhteyttä ja sen sälytyspaikkaa, *SAD*:ia (security association database), hypoteettisinä käsitteinä, koska ne ovat osapuolten sisäisiä asioita.

Turvayhteydet II

- Ne sisältävät kommunikoinnin kannalta oleellisia tietoja, mutta itse SA kokonaisuudessaan ei ole osa kommunikointia. Sen tähden dokumentit eivät ota kantaa sen muotoon tai sijaintiin. Käytännössä SAD on taulukko, jota säilytetään suojatussa muistissa, ja SA on tietue taulukossa.
- Jokainen turvayhteys sisältää tietoa, jonka avulla IPSec-prosessi voi päättää, sovelletaanko SA:n määrittelemää suojaa tiettyyn lähtevään tai tulevaan pakettiin. Ratkaisu tehdään SA:n *valitsimien* (selectors) perusteella. Valitsimet sisältävät seuraavaa:
  - Lähde- ja kohdeosoite. Toistaiseksi sallitaan vain yksittäiset osoitteet, ei yleislähetyksiä. Kohdeosoite voi olla joko loppukäyttäjä tai palomuuri tai reititin.
  - Nimi on joko käyttäjätunnus tai systeemin nimi.
    - Käyttäjätunnus rajaa SA:n vain erityisen käyttäjän aloittamaan tai vastaanottamaan kommunikointiin.

Turvayhteydet III

- Jos ainoat valitsimet ovat kommunikoivien osapuolten käyttäjätunnuksia, SA:ta kutsutaan käyttäjäsuuntautuneeksi (user-oriented).
- Jos taas käytetään systeeminimiä, se rajaa liikenteen tiettyjen systeemien välille. Systeemi voi olla isäntäkone, turvayhdyskäytävä tms.

- Kuljetuskerroksen protokolla (TCP tai UDP).
- Lähde- ja kohdeportti. Yleensä käytetään yhtä ainoaa porttinumeroa, jolla rajataan SA:n käyttö tiettyyn sovellukseen (esim. FTP).

- Jokainen SA sisältää myös seuraavia tietoja:
  - *Järjestysnumerolaskuri* on 32 bitin arvo, jota käytetään AH- ja ESP-otsakkeissa järjestysnumeroiden generoimiseen.
  - *Järjestysnumeron ylivuoto* on lippu, joka osoittaa, kirjataanko järjestysnumeron ylivuodosta lokitapahtuma vai ei. Jos kirjataan, niin seuraavien pakettien lähetys tässä turvayhteydessä on estetty.
  - *Uudelleenlähetysikkunaa* (anti-replay window) käytetään ratkaisemaan, onko saapunut AH- tai ESP-paketti uudelleenlähetys vai ei.

Turvayhteydet IV

- *AH-informaatio* sisältää todennusalgoritmin, avaimet, avainten eliajan ja parametrit, joita tarvitaan AH-paketin ja todennuksen yhteydessä.
- *ESP-informaatio* sisältää salaus- ja todennusalgoritmit, avaimet, alustusarvot, avainten elinajat ja muut parametrit, joita tarvitaan ESP:n kanssa.
- *Turvayhteyden elinaika* on aikaväli tai tavumäärä, jonka jälkeen turvayhteys täytyy korvata uudella tai päättää. Elinaikaan liittyy vielä tieto, kumpi noista kahdesta on käytössä.
- *IPSecin protokollamoodi* tarkoittaa *tunneli-, kuljetusmoodia* tai *villiä korttia*, joiden merkitystä selvitetään myöhemmin.
- *Polun MTU* (maximum transmission unit) tarkoittaa maksimaalista pakettikokoa, joka voidaan välittää pilkkomatta. Lisäksi paketteihin liittyvät aikamääreet kuuluvat MTU-parametriin.

SPI I

- On varsin todennäköistä, että kommunikoivat osapuolet sopivat useammasta kuin yhdestä SA:sta. Esimerkiksi sähköposti ja Web-sovellus vaativat vähemmän kuin maksuja siirtävä protokolla.
- Kun suojattua pakettia ollaan lähettämässä, lähettäjän täytyy tiedottaa vastaanottajalle, mitä SA:ta on käytetty paketin kohdalla, jotta vastaanottaja tietäisi valita saman SA:n. Tätä palvelee turvaparametri-indeksi (SPI).
- Koska jokainen SA on *yksisuuntainen*, turvallinen kaksisuuntainen yhteys vaatii kahden SA:n määrittelemistä: sisään tulevan ja ulos menevän.
- SPI yhdessä kohdeosoitteen ja turvaprotokollan (AH, ESP) kanssa on riittävä, jotta sisään tulevan paketin SA osataan hakea SAD:sta. Jotta taataan SPI:n yksikäsitteisyys, kumpikin osapuoli valitsee oman sisääntulevan SPI:n.

Turvapolitiikan tietokanta I

- Kaikki liikenne IPSec-verkoissa jaetaan turvayhteyksiin ja muuhun liikenteeseen. Turvayhteyksiä voidaan yhdistellä monella tavalla halutun tuloksen aikaansaamiseksi. Turvayhteyksiin liittyvää liikennettä säädellään turvapolitiikan tietokannan (SPD) avulla.
- Yksinkertaisimmillaan SPD sisältää tietueita, joista kukin liittyy tiettyyn osaan IP-liikennettä ja tiettyyn turvayhteyteen.
- Monimutkaisemmissa tilanteissa moni tietue voi liittyä samaan turvayhteyteen tai moni turvayhteys voi liittyä yhteen SPD-tietueeseen. Tällä kurssilla ei kaikkia mahdollisuuksia käsitellä yksityiskohtaisesti.
- Jokainen SPD-tietue määritellään IP- ja ylemmän kerroksen kenttäarvojen avulla, joita kutsutaan valitsimiksi (selectors). Näitä valitsimia käytetään suodattamaan ulosmenevä liikenne siten, että se kyetään yhdistämään tiettyyn turvayhteyteen.

Turvapolitiikan tietokanta II

- Ulosmenevän liikenteen käsittely noudattaa seuraavia periaatteita:
  1. Etsi paketin sopivien kenttien perusteella liikennettä vastaava SPD-tietue, joka puolestaan viittaa nollaan tai useampaan turvayhteyteen.
  2. Poimi SPD-tietueen ja paketin SPI:n perusteella pakettiin liittyvä turvayhteys.
  3. Prosessoi paketti turvayhteyden mukaisesti.
- SPD-tietueen määrittelemiseksi käytetään seuraavia valitsimia:
  - *Kohteen IP-osoite* voi olla joko yksittäinen osoite, osoitelista, osoiteväli tai villi kortti -osoite. Jos osoite käsittää useita yksittäisiä osoitteita, niiden haltijat sijaitsevat saman palomuurin takana ja niihin liittyy sama turvayhteys.
  - *Lähteen IP-osoite* voi myös olla yksittäinen, lista, väli tai villi kortti.
  - *Käyttäjätunnus* on käyttöjärjestelmään liittyvä käyttäjätunnus. Tätä ei käytetä IP- tai ylemmissä otsakkeissa, mutta se on saatavilla, jos IPSec toimii saman käyttöjärjestelmän alaisuudessa kuin käyttäjäkin.

Turvapolitiikan tietokanta III

- *Tiedon luottamuksellisuusaste* on esimerkiksi salainen tai luokittelematon.
- *Kuljetuskerroksen protokolla* saadaan IPv4:n tai IPv6:n kentästä Next Header. Se voi olla yksittäisen protokollan numero, lista protokollanumeroita tai protokollanumeroiden väli.
- *Lähde- ja kohdeportit* voivat jälleen olla yksittäisiä tai usean portin joukkoja.

Todennusotsake I

- Todennusotsakkeeseen (AH) perustuva protokolla huolehtii siis tiedon eheydestä ja IP-pakettien todennuksesta. Pakettien todennus varmistaa käyttäjän tai palvelun identiteetin, joiden pohjalta suodatus tapahtuu. AH suojaa myös uudelleenlähetyksiä vastaan.
- Todennus perustuu MAC-koodiin, joka edellyttää samaa salaista avainta lähettäjällä ja vastaanottajalla. Todennusotsake koostuu seuraavista kentistä:
  - Seuraavan paketin otsakkeen tyyppi (8 b).
  - Hyötykuorman pituus (8 b).
  - Varattu osa (16 b).
  - SPI (32 b).
  - Järjestysnumero (32 b).
  - Todennustieto (muuttuva). Tämä kenttä sisältää eheyden tarkistusarvon (ICV, integrity check value) tai MAC-arvon.

Todennusotsake II

- Uudelleenlähetysten torjuntaan käytetään AH:n järjestysnumerokenttää. Kun uutta turvayhteyttä perustetaan, lähettäjä alustaa järjestysnumerolaskurin nollaksi.
- Joka kerran kun paketti lähetetään käyttäen perustettua turvayhteyttä, lähettäjä kasvattaa laskuria ja asettaa sen arvon järjestysnumerokenttään.
- Siten ensimmäinen arvo on 1. Laskurin suurin arvo on $2^{32} - 1$. Laskuria ei saa päästää tämän jälkeen takaisin nollaan, vaan jos lisäpaketteja on tulossa, on perustettava uusi turvayhteys uudella avaimella.
- Koska IP on yhteydetön, epäluotettava palvelu, protokolla ei takaa, että paketit luovutetaan perille järjestyksessä tai että edes kaikki paketit menevät perille. Siksi IPSec vaatii, että vastaanottajan on toteutettava ikkuna, jonka oletusarvoinen koko on $W = 64$. Ikkunan oikea reuna sisältää suurimman tähän asti vastaanotetun järjestysnumeron, N.

Todennusotsake III

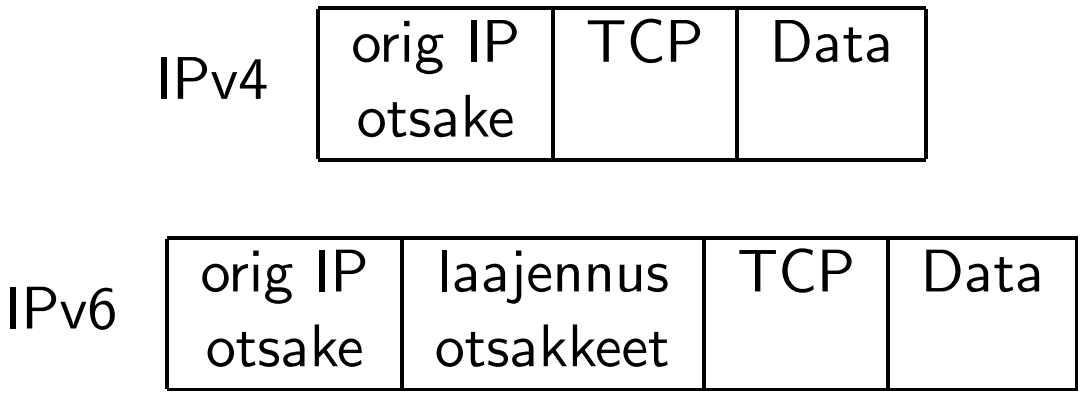
- Jos saapuvan paketin järjestysnumero on välillä $[N - W + 1, N]$, vastaava paikka ikkunassa merkitään. Tarkemmin kuvattuna vastaanottopäässä tehdään seuraavaa:
  1. Jos saapuneen paketin järjestysnumero sisältyy ikkunan lukuihin ja on uusi, MAC tarkistetaan. Jos todennus onnistuu, järjestysnumeroa vastaava paikka ikkunassa merkitään.
  2. Jos saapuneen paketin järjestysnumero menee oikealta ikkunan ulkopuolelle ja on uusi, MAC tarkistetaan. Jos todennus onnistuu, ikkunaa siirretään oikealle niin, että vastaanotetusta järjestysnumerosta tulee ikkunan uusi oikea reuna.
  3. Jos saapuneen paketin järjestysnumero menee vasemmalta ikkunan ulkopuolelle tai jos todennus epäonnistuu, paketti hylätään. Hylkäys kirjataan lokiin.

Todennusotsake IV

- Eheyden tarkistusarvo on tiivistefunktion tai MACin arvo. IPSec:in tulee tarjota ainakin kaksi tiivistefunktiota, HMAC-MD5-95 ja HMAC-SHA-1-96. Molemmat käyttävät HMAC-algoritmia, edellinen MD5-tiivistefunktion, jälkimmäinen SHA-1 -tiivistefunktion kanssa. Kummassakin lasketaan ensin kryptografinen tiivistekoodi, mutta siitä otetaan mukaan vain ensimmäiset 96 bittiä.
- Tiivistearvo lasketaan seuraavista kentistä:
  - IP:n tunnusosan kentät, jotka eivät muutu liikenteessä tai joiden arvo vastaanotettaessa on ennustettavissa. Kentät, jotka muuttuvat matkalla eivätkä ole ennustettavissa, asetetaan nollaksi tiivistettä laskettaessa.
  - AH-otsake paitsi todennustietokenttää, joka asetetaan nollaksi.
  - Kaikki ylemmän tason tieto, joka oletetaan muuttumattomaksi liikenteessä.

AH:n kuljetus- ja tunnelimoodi I

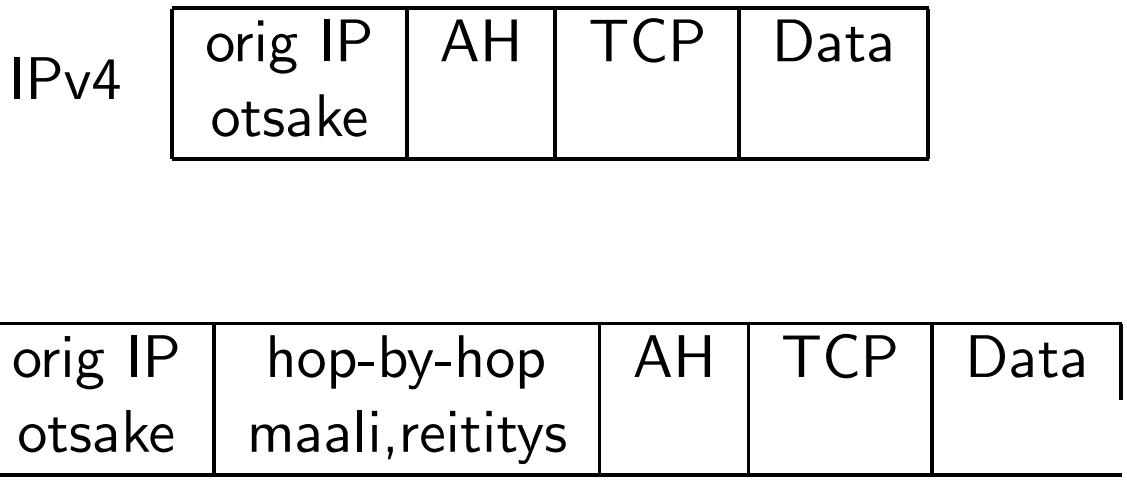
IPSec:in todennuspalvelua voidaan käyttää kahdella tavalla. Näitä tapoja kutsutaan kuljetusmoodiksi ja tunnelimoodiksi. Kuvassa 8 nähdään pakettien tilanne ennen AH:n soveltamista.



Kuva: Ennen AH:n soveltamista

AH:n kuljetus- ja tunnelimoodi II

Kuvassa 9 puolestaan on pakettien tilanne kuljetusmoodissa AH:n soveltamisen jälkeen.

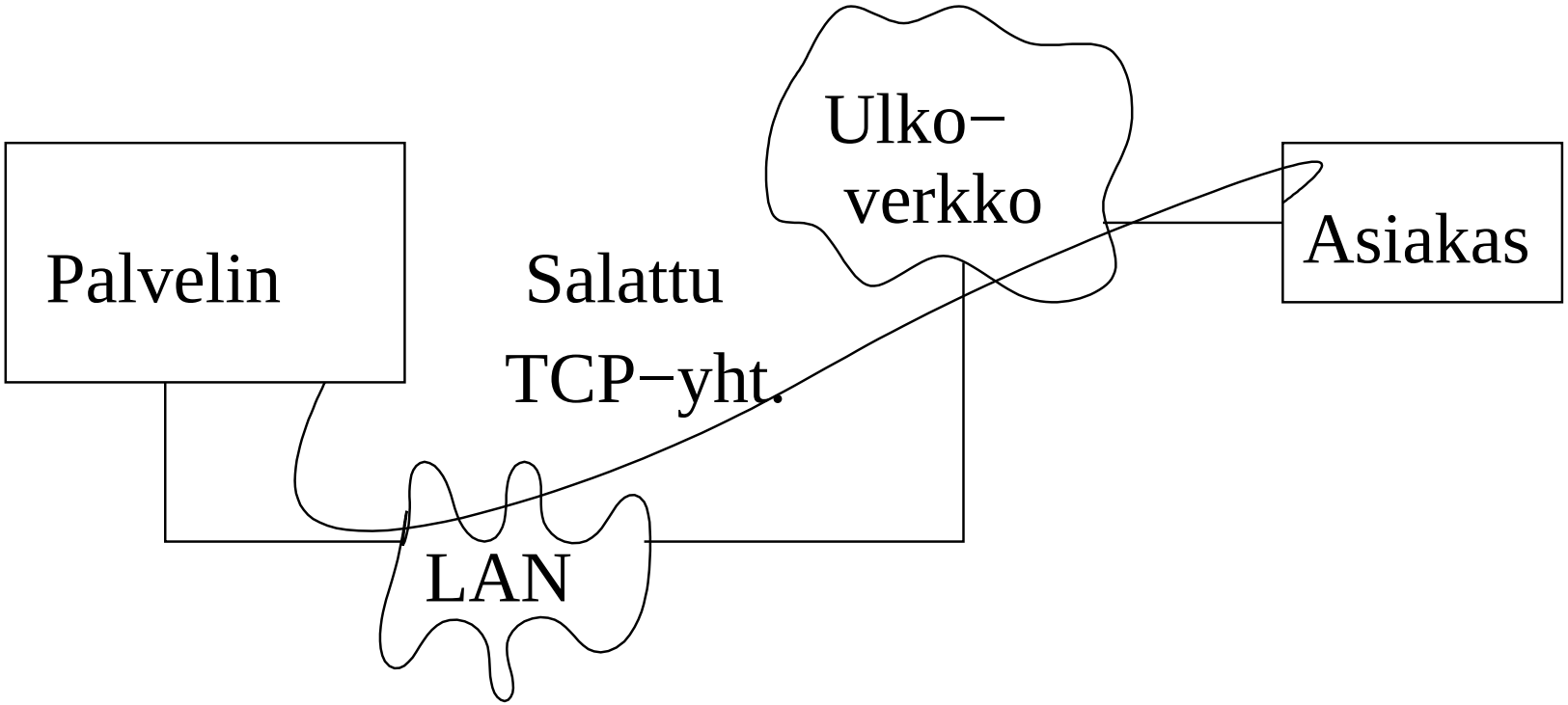


Kuva: AH:n kuljetusmoodi

AH todentaa koko kentän mahdollisia muuttuvia kenttiä lukuunottamatta. Huomattakoon, että tilanne on erilainen IPv4:n ja IPv6:n välillä.

AH:n kuljetus- ja tunnelimoodi III

Kuljetusmoodia käytetään esimerkiksi kuvan 10 tilanteessa, jossa asiakas ja palvelin kommunikoivat suoraan ja niillä on yhteinen salainen avain. Asiakas voi olla joko samassa verkossa palvelimen kanssa tai eri verkossa.



Kuva: AH:n kuljetusmoodin soveltaminen

AH:n kuljetus- ja tunnelimoodi IV

Tunnelimoodissa AH lisätään puolestaan pakettiin kuvan 53 mukaisesti. Edelleeen todennus koskee koko pakettia muuttuvia kenttiä lukuunottamatta.

IPv4

uusi IP otsake	AH	orig IP otsake	TCP	Data

IPv6

uusi IP otsake	laajennus otsakkeet	AH	orig IP otsake	laajennus otsakkeet	TCP	Data

Kuva: AH:n tunnelimoodi

AH:n kuljetus- ja tunnelimoodi V

Siis tunnelimoodissa koko alkuperäinen paketti todennetaan ja AH lisätään alkuperäisen IP-otsakkeen ja uuden, ulomman IP-otsakkeen väliin. Sisempi IP-otsake sisältää varsinaisen lähde- ja kohdeosoitteen, kun taas ulompi IP-otsake voi sisältää muita, esimerkiksi reitittimien, osoitteita.

Tunnelimoodia käytetään tyypillisesti tilanteessa, jossa ulkoinen työasema todentaa itsensä palomuurille päästäkseen sen jälkeen palomuurin suojaamaan verkkoon. Tunnelimoodia käytetään erityisesti rakennettaessa ns. virtuaalisia yksityisiä verkkoja(VPN).

Koteloitu salattu data I

Koteloitu salattu data eli ESP tarjoaa siis salauksen ja haluttaessa myös todennuksen. ESP-otsake koostuu seuraavista kentistä:

- SPI (sama kuin AH:ssa).
- Järjestysnumero (AH:ssa).
- Hyötykuorma on kuljetuskerroksen segmentti (kuljetusmoodi) tai IP-paketti (tunnelimoodi), joka suojataan salauksella.
- Täyte (0-255 B) selitetään myöhemmin.
- Täytteen pituus (8 b) on täytteen pituus tavuissa.
- Seuraava otsake (8 b) määrittelee sen datan tyypin, joka sijaitsee hyötykuormakentässä. Tyyppi määräytyy ensimmäisen tunnusosan mukaan.
- Todennustieto (AH).

ESP-palvelu salaa kentät

Koteloitu salattu data II

- hyötykuorma,
- täyte,
- täytteen pituus ja
- seuraava otsake.

Jos salausalgoritmi vaatii esimerkiksi alustusvektorin, se välitetään yleensä kentän hyötykuorma alussa salaamattomana.

Täyte palvelee montaa tarkoitusta. Jos salausalgoritmi vaatii, että selväteksti on tavujen monikerta, selvätekstiin voidaan lisätä täyte. Täyte voidaan lisätä myös salatekstin ja kenttien täytteen pituus ja seuraava otsake väliin. Täytettä voidaan käyttää myös salaamaan hyötykuormakentän todellinen pituus.

ESP:n kuljetus- ja tunnelimoodi I

Samoin kuin AH:n kohdalla myös ESP-protokollaa voidaan käyttää kuljetus- ja tunnelimoodissa. Kuvassa 12 nähdään, mitkä kentät salataan ja todennetaan ESP-paketeissa kuljetusmoodissa.

IPv4:

orig IP otsake	ESP otsake	TCP	Data	ESP pääte	ESP tod

IPv6:

orig IP otsake	hop-by-hop, kohde reititys, palat	ESP otsake	kohde	TCP	Data

ESP pääte	ESP tod

Kuva: ESP:n kuljetusmoodi

ESP:n kuljetus- ja tunnelimoodi II

Kuljetusmoodin toiminta etenee seuraavasti:

1. Lähettäjän puolella ensin salataan kentät 3-5 (IPv4) tai 4-7 (IPv6). Selväkieliset vastaavat kentät korvataan salatekstillä. Todennus lisätään, jos sitä halutaan. Todennus kattaa kentät 2-5.
2. Paketti reititetään kohteeseen. Jokainen välillä oleva reitittäjä tutkii IP-otsakkeen ja selväkielisen laajennusotsakkeen, mutta ei salattua osaa.
3. Vastaanottaja tutkii selväkieliset kentät. ESP-osan SPI-tietojen perusteella vastaanottaja purkaa salauksen.

ESP:n kuljetus- ja tunnelimoodi III

Tunnelimoodissa koko IP-paketti plus ESP-perä salataan. Reititystä varten alkuperäisestä IP-paketista kerätään tarvittavat tiedot, joita käytetään ulomman IP-paketin tunnusosassa. Kuvassa 13 näkyy salaukseen ja todennukseen käytetyt kentät.

IPv4:

Uusi IP otsake	ESP otsake	orig otsake	TCP	Data	ESP pääte	ESP tod

IPv6:

Uusi IP otsake	laaj. otsakkeet	ESP otsake	orig IP otsake	laaj. otsakkeet	TCP	Data

ESP pääte	ESP tod

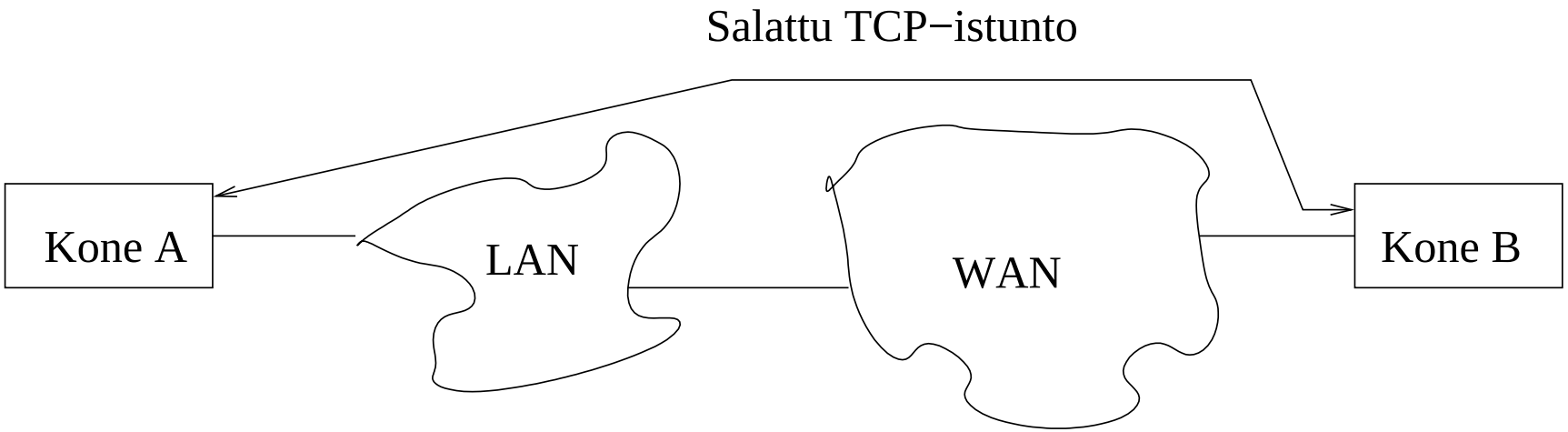
Kuva: ESP:n tunnelimoodi

ESP:n kuljetus- ja tunnelimoodi IV

- Kuljetusmoodi sopii suojaamaan yhteyksiä kahden koneen välillä, joissa kummassakin on ESP.
- Tunnelimoodi on hyödyllinen, kun toisena osapuolena on palomuuri tai muu turvallinen yhdyskäytävä, joka suojaa verkkoa ulkopuolisilta.
- Salaus on käytössä tässä tapauksessa yleensä vain ulkoisen koneen ja yhdyskäytävän välillä. Suojatun verkon sisällä salausta ei tarvita.

AH ja ESP kuvioina I

Seuraavassa esitetään AH:n ja ESP:n toimintaa kuvioiden avulla. Näitä voidaan sitten käyttää hyväksi kuvattaessa havainnollisesti turvayhteyksien yhdistämistä. Kuvassa 14 on tyypillinen tilanne, jossa kahden koneen yhteys on suojattu kuljetusmoodin avulla. Tällä saavutetaan TCP-istunnon salaus.

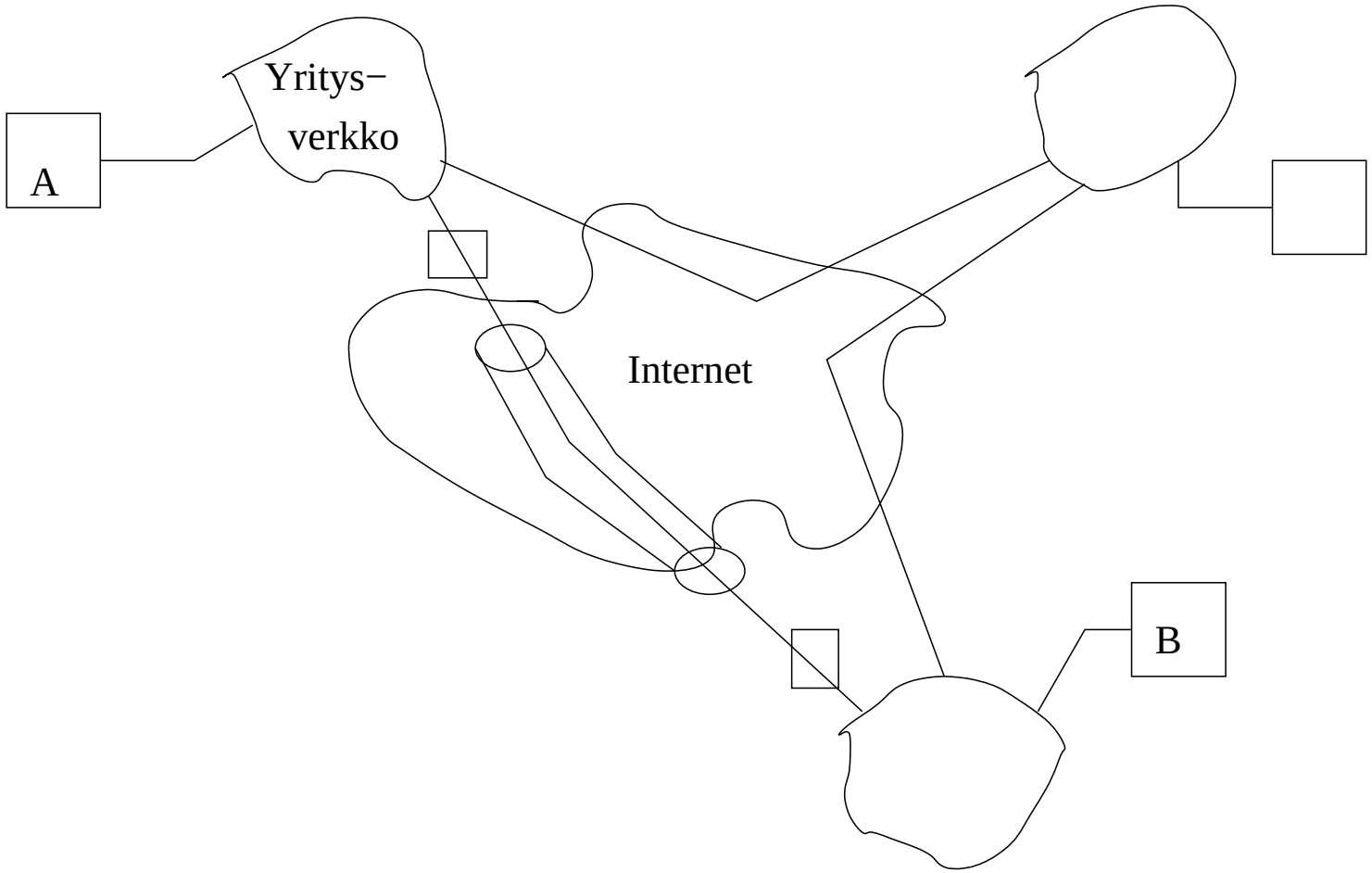


Kuva: Kuljetusmoodi

AH ja ESP kuvioina II

Kuvassa 15 puolestaan on toteutettu virtuaalinen yksityinen verkko IPSecin tunnelimoodin avulla.

AH ja ESP kuvioina III

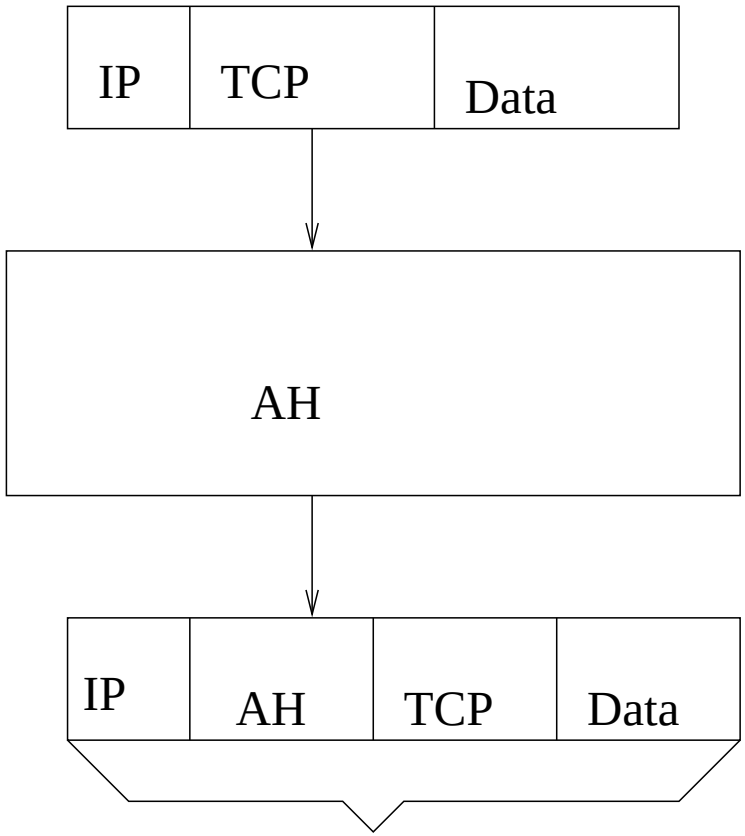


Kuva: Tunnelimoodi

AH ja ESP kuvioina IV

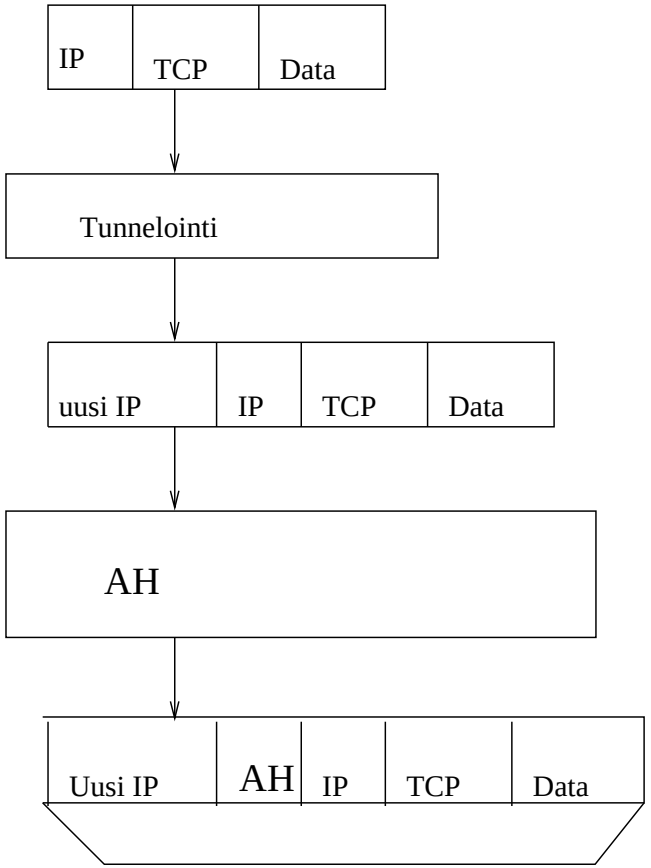
Kuvasarja 16...23 puolestaan esittää pakettien muodostumista eri moodeissa. Aluksi AH ja ESP ovat erillään, mutta viimeisissä kuvissa käsitellään näiden yhdistämistä.

AH ja ESP kuvioina V



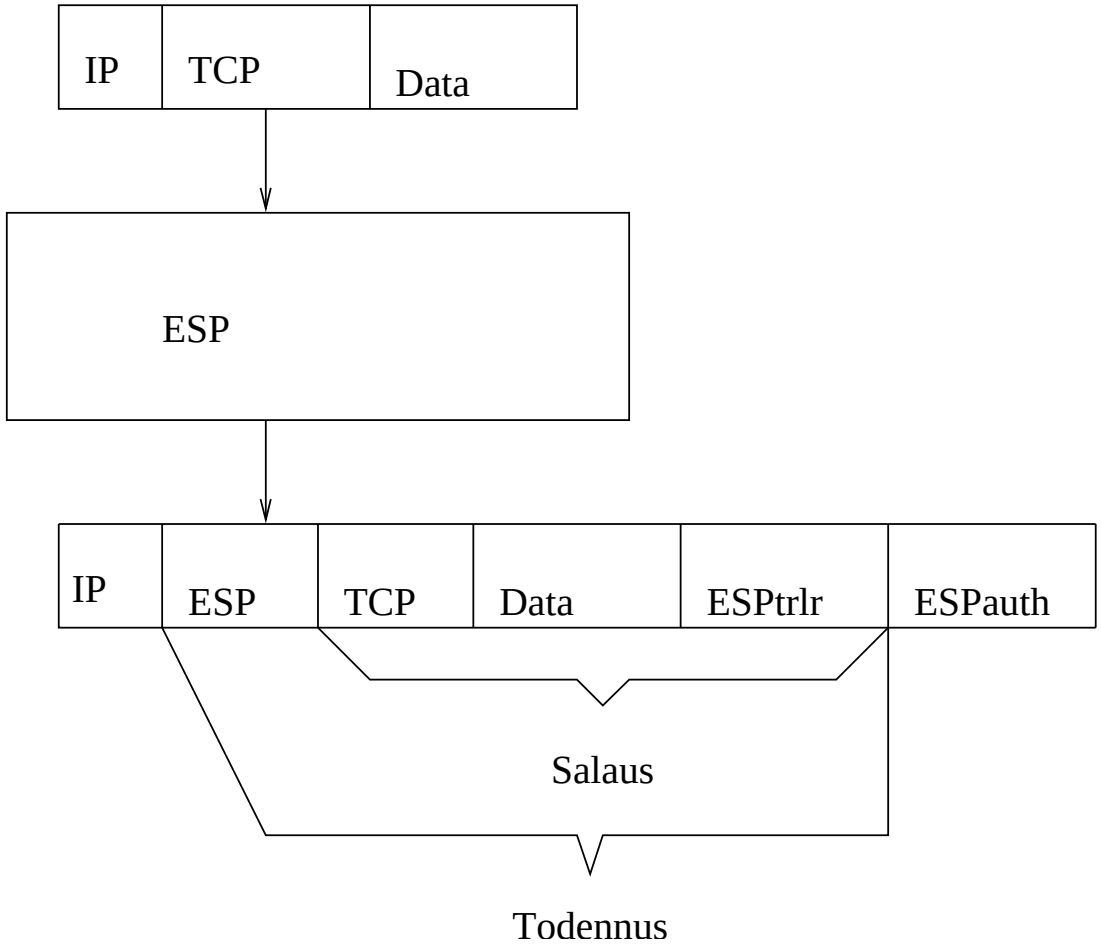
Kuva: AH kuljetusmoodissa

AH ja ESP kuvioina VI



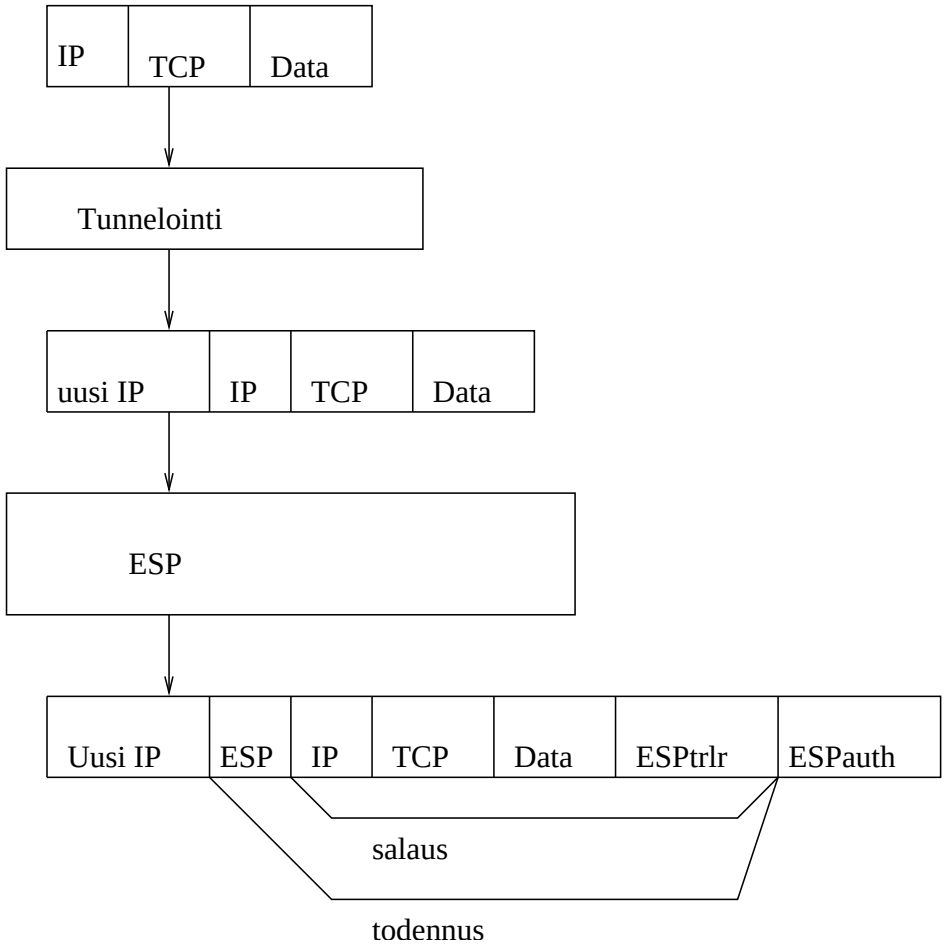
Kuva: AH tunnelimoodissa

AH ja ESP kuvioina VII



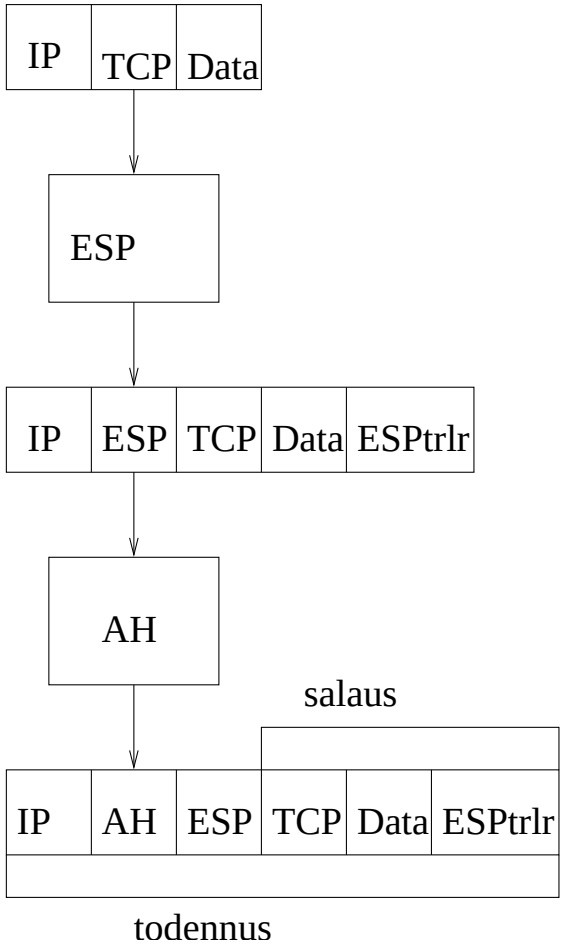
Kuva: ESP kuljetusmoodissa

AH ja ESP kuvioina VIII



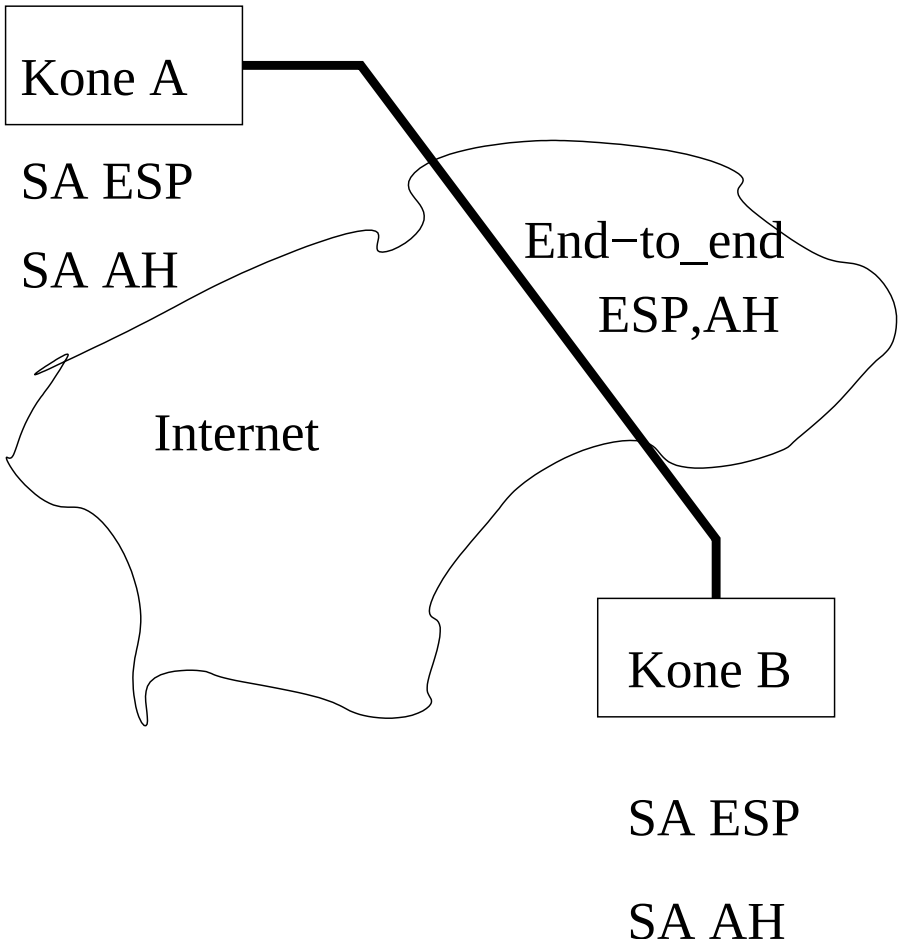
Kuva: ESP tunnelimoodissa

AH ja ESP kuvioina IX



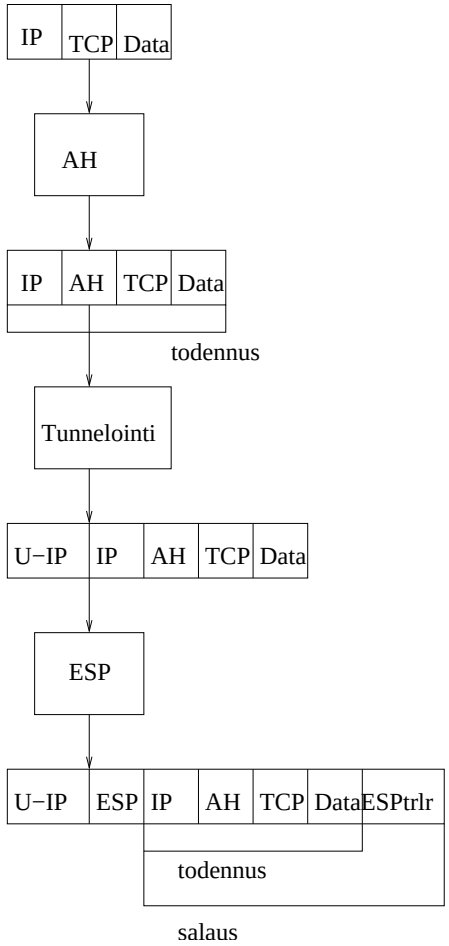
Kuva: ESP ja AH, molemmat kuljetusmoodissa

AH ja ESP kuvioina X



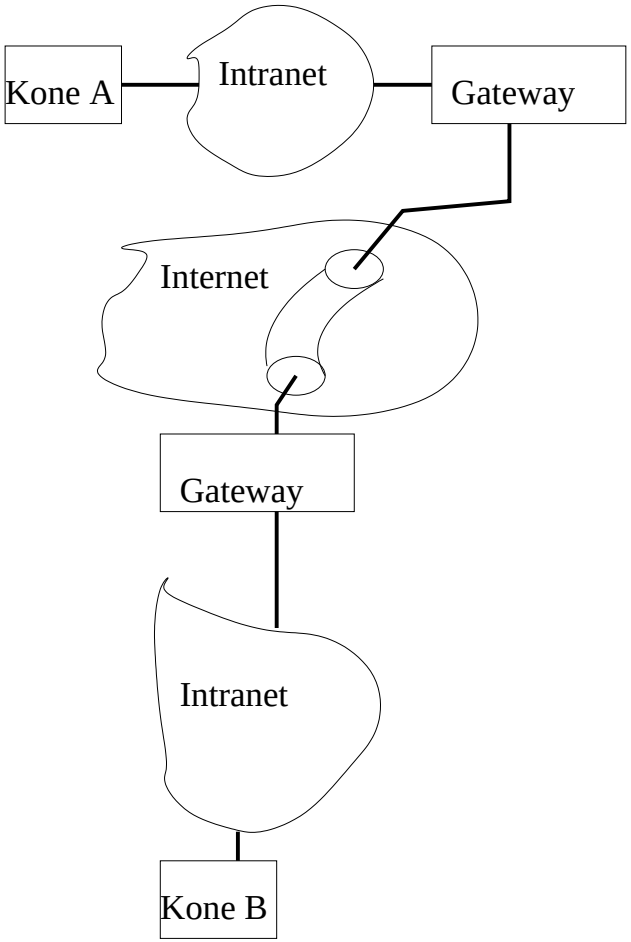
Kuva: ESP ja AH -yhdistelmän sovellustilanne

AH ja ESP kuvioina XI



Kuva: AH kuljetus- ja ESP tunnelimoodissa

AH ja ESP kuvioina XII



Kuva: AH-kuljetus, ESP-tunneli: sovellustilanne

IPSec:in kritiikkiä I

- IPSec on syntynyt komiteatyönä, jossa mukana on ollut useita tahoja. Tämä on johtanut siihen tyypilliseen tilanteeseen, että yhteen ja samaan protokollaan on sovitettu monia piirteitä ja näkökantoja.
- Tällainen suunnittelu on johtanut IPSec:in ja monen muunkin protokollan (erityisesti ISOn OSI-protokollat aikoinaan) yhteydessä laajuuteen ja monimutkaisuuteen, joka ei enää palvele käyttäjiä.
- Näyttääkin siltä, että parempi tulos saavutetaan kilpailujen avulla kuten AES:n yhteydessä. Tällöin yksi ja sama, suppeahko tiimi suunnittelee protokollan, jolloin sen koko pysyy kohtuullisena.

Opetus 1. Kryptografisia protokollia ei pitäisi suunnitella komiteatyönä.

- Seuraavassa käsitellään IPSec:in ongelmia kohta kohdalta Niels Fergusonin ja Bruce Schneierin artikkelin *A Cryptographic Evaluation of IPSec* pohjalta .

IPSec:in kritiikkiä II

- Esityksessä keskitytään vain varsinaiseen toimintaan. IPSec:in avaintenhallinta oli pitkään kehityksen kohteena ja artikkelin kirjoituksen aikana se oli erityisen sekavaa. Siitä syystä emme käsittele artikkelin avaintenhallintaan kohdistuvaa kritiikkiä.
- Artikkelin kaikki suositukset on esitetty myös tässä. Toisaalta ei ole aivan selvää, että kaikki olisivat artikkelin ehdotuksiin tyytyväisiä. Erityisesti käytännön verkkosuunnittelijan näkökulma saattaa olla toinen kuin pelkästään tietoturvaihmisten.

IPSec:in dokumentointi I

- IPSecin dokumentteja on hyvin vaikea ymmärtää. Niissä ei yleensä ole yleiskatsausta tai johdantoa. Siten ei ole uskottavaa, että kukaan oppisi IPSeciä virallisista dokumenteista.
- Tässä yhteydessä erityisesti mainitaan alustava avaintenhallintaprotokolla ISAKMP, jonka dokumentointi sisältää virheitä, josta monet oleelliset selitykset puuttuvat ja joka on sisäisesti ristiriitainen.
- Dokumenteista ei käy selville protokollan tavoite. Tällöin protokollan analysointi on hankalaa. Samoin sunnittelijan, joka yrittää soveltaa IPSec:iä käytäntöön, työ vaikeutuu.
- IPSec tuottaa IP-tason turvallisuutta ja on siten oleellisesti VPN-protokolla. Kuitenkin on ollut tapauksia, jossa protokollaa on käytetty sovellustason turvallisuuden saavuttamiseen kuten esimerkiksi henkilön todentamiseen, kun tämä yrittää lukea sähköpostiaan.

IPSec:in dokumentointi II

- IPSec perustaa pakettien todennuksen siihen, että paketti on lähtenyt joltakulta, joka tuntee salaisen avaimen. Kuitenkin monet näyttävät uskovan, että se todentaa lähettävän IP-osoitteen, jota sitten voidaan käsitellä palomuurissa.
- Dokumentit eivät myöskään sisällä selityksiä tai perusteluja valinnoille, joita on tehty. Vaikka nämä eivät ole niin tärkeitä kuin tavoitteet, ne ovat myös oleellisia.

Opetus 2. Systeemin dokumentin tulisi sisältää johdattelevaa materiaalia, yleiskatsaus niille, jotka tutustuvat asiaan ensimmäistä kertaa, asetetut tavoitteet ja perustelut.

Datan käsittely: Moodit I

Turvaomisuuksien kannalta katsottuna tunnelimoodi sisältää kuljetusmoodin (verkkokerroksesta katsoen tilanne saattaa olla päinvastainen). Kuljetusmoodi kuluttaa tosin vähemmän kaistanleveyttä. Tunnelimoodiakin voitaisiin tehostaa, joten tekijät suosittavat

Suositus 1. Kuljetusmoodi voidaan jättää pois.

Dokumenteissa ei perustella kahden moodin olemassaoloa. Kuljetusmoodin poisjättäminen välttäisi myöskin koneiden jakamisen kahteen luokkaa, isäntäkoneisiin ja turvayhdyskäytäviin. Näiden pääero näyttää olevan, etteivät turvayhdyskäytävät voi käyttää kuljetusmoodia.

Datan käsittely: AH ja ESP I

- Dokumentit eivät selitä, miksi IP-otsakkeet pitää todentaa. Hyötykuorman todennus takaa, että kuorma tulee sellaiselta, joka tuntee salaisen avaimen. IP-otsakkeeet vain auttavat pakettia menemään vastaanottajalle, eikä niiden pitäisi vaikuttaa paketin tulkintaan.
- AH todentaa alempien kerrosten IP-otsakkeita. Tämä selvästi rikkoo protokollapinon modulaarisuutta. Se aiheuttaa monia ongelmia, koska jotkut kentät muuttuvat matkan aikana. Siten AH:n täytyy tuntea kaikki alempien kerrosten dataformaatit, jotta muuttuvat kentät voidaan välttää. Tämä ei tunnu järkevältä.

Suositus 2. Jätetään AH pois.

- ESP sallii hyötykuorman salauksen ilman todennusta. Hyvin harvoin salaus ilman todennusta on hyödyllistä.

Datan käsittely: AH ja ESP II

- IPSecin yhteydessä tällainen tilanne on kuljetusmoodissa, jossa ESPin todennus ei ole kovin kattava ja on käytettävä lisäksi AH:ta. Jos kuljetusmoodi ja AH jätettäisiin pois, voitaisiin suositella:

Suositus 3. Muutetaan ESPiä siten, että se tuottaa aina todennuksen; vain salaus voisi olla valinnainen.

Operaatioiden järjestys I

- Kun käytetään sekä salausta ja todennusta, IPSec salaa ensin ja todentaa sitten. Tämä on Fergusonin ja Schneierin mukaan väärä järjestys. Todentaa pitäisi se, mitä tarkoitetaan, ei sitä, mitä sanotaan.
- IPSecin todennus mahdollistaa myös hyökkäyksen. Oletetaan, että kaksi konetta ovat neuvotelleet AH:ta käyttävän SA:n, jossa avaimet on jaettu manuaalisesti.
- Merkitään tätä SA:ta symbolilla $\mathrm{SA}_{\mathrm{AH}}$. Koska avaimet on sovittu manuaalisesti, AH ei tarjoa suojaa uusintahyökkäyksille.
- Oletetaan nyt, että koneet neuvottelevat kuljetusmoodin ESP:n, jossa käytetään vain salausta. Merkitään vastaavaa turvayhteyttä symbolilla $\mathrm{SA}_{\mathrm{ESP1}}$.
- Tietoa välitettäessä käytetään kimppua, joka koostuu näistä kahdesta turvayhteydestä. Sovellus voi olettaa saavansa tällä tavalla luottamuksellisuuden ja todennuksen, mutta ei suojaa uusinnoilta.

Operaatioiden järjestys II

- Kun kimppuun perustuva yhteys lopetetaan, turvayhteys $\mathrm{SA}_{\mathrm{ESP1}}$ puretaan. Muutamia tunteja myöhemmin samat koneet neuvottelevat taas uuden kuljetumoodin ESP:n, jossa on vain salaus ($\mathrm{SA}_{\mathrm{ESP2}}$) ja vastaanottaja valitsee saman SPI:n arvon kuin edellisen ESP:in yhteydessä.
- Dataa välitetään taas kimpun avulla, joka sisältää sekä $\mathrm{SA}_{\mathrm{ESP2}}$:n että $\mathrm{SA}_{\mathrm{AH}}$:n.
- Hyökkääjä ujuttaa sanomien joukkoon nyt jonkin vanhan paketin edellisestä istunnosta. Tämä paketti oli salattu $\mathrm{SA}_{\mathrm{ESP1}}$:n avulla ja todennettu $\mathrm{SA}_{\mathrm{AH}}$:n avulla. Vastaanottaja toteaa todennuksen päteväksi. (Koska uusintojen suojausta ei käytetä, järjestysnumerokenttää ei käytetä.)
- Vastaanottaja purkaa sitten salauksen käyttäen $\mathrm{SA}_{\mathrm{ESP2}}$:ta, mikä tuottaa eri tuloksen kuin jos olisi käytetty $\mathrm{SA}_{\mathrm{ESP1}}$:tä.

Operaatioiden järjestys III

- Seurauksena on, että vastaanottaja hyväksyy todennetun paketin, purkaa sen väärällä avaimella ja luovuttaa vääristyneen datan sovellukselle. Eli todennus on epäonnistunut.

Opetus 3. Älä todenna pelkästään sanomaa, vaan kaikki se, mitä käytetään sanoman merkityksen määräämiseksi.

- Salatekstin todennus tekee mahdolliseksi hylätä paketteja nopeasti käyttämättä aikaa salauksen purkamiseen.
- Tämä auttaa konetta palvelunestohyökkäyksissä. Mikäli tämä koetaan tärkeäksi, salatekstin todennus voidaan säilyttää, mutta vain jos samalla todennetaan purkuavain.
- Tämä olisi mahdollista, mutta se sotisi pahasti modulaarisuutta vastaan, kun AH joutuisi kaivamaan ESP:n rakenteista salausvaimen.

Suositus 4. Modifioi ESP:tä siten, että todentaa datan lisäksi hyötykuorman salauksen purkuavaimen.

Turvayhteydet I

On aika vähän tilanteita, joissa kone lähettää IP-paketin toiselle, mutta vastausta ei lähetetä eikä oleteta. On myös vähän tilanteita, joissa täytyy turvata liikenne yhteen suuntaan, mutta ei vastakkaiseen suuntaan. Siten lähes kaikissa tilanteissa SA:t esiintyvät pareittain muodostaen symmetrisen kaksisuuntaisen kanavan. Olisi siten selvempää, että SA:t olisivat kaksisuuntaisia.

Turvapolitiikat I

- Turvapolitiikan tietokanta SPD sallii monien valitsimien käytön päätettäessä, mihin pakettiin sovelletaan mitäkin SA:ta. On mahdollista, että yksi SA hoitaa kaiken liikenteen kahden koneen välillä tai että kullakin sovelluksella on oma SA:nsa.
- Mikäli ylläpidon pitää luokitella paketit sen mukaan, mitkä vaativat IPSec-prosessointia ja mitkä eivät, vaaditaan ehkä jo liian paljon.
- Kun lisäksi ylläpidon pitää asettaa lukuisia muita IPSec-asetuksia salaus- ja todennusmenetelmistä alkaen, on todennäköistä että monet konfiguraatiot sisältävät heikkouksia.

SSL I

SSL (Secure Socket Layer) on standardi, joka on suunniteltu web-selainten ja -palvelimien turvamekanismiksi. SSL käyttää termejä istunto ja yhteys. Istunto on perustettu osapuolten (asiakas-palvelin) välille ja yhteys on kokoelma mekanismeja, joille tietoa siirretään istunnossa. Yhdellä istunnolla voi olla monta yhteyttä. Kahdella osapuolella voi olla myös monia istuntoja yhtäaikaa keskenään, mutta se ei ole tavallista. Jokaiseen istuntoon liittyy tietoa, jonka avulla istunto identifioidaan ja hallitaan:

- istunnon tunnus,
- osapuolen X.509-varmenne,
- tiivistysmenetelm,
- salausmenetelmä,
- pääsalaisuus (48 bittiä, symmetrinen).

Myös jokaiseen yhteyteen liittyy tietoa, jonka avulla yhteyttä hallitaan:

SSL II

- satunnaista dataa palvelimelle ja asiakkaalle (tarvitaan esimerkiksi lohkon täytteissä),
- salausavaimet,
- MAC-avain,
- alustusvektorit,
- järjestysnumerot.

Alempi kerros I

Alempi kerros huolehtii yhteydestä. Sovellus on tämän kerroksen päällä. Esimerkiksi SSL:n alempi kerros ottaa vastaan sanomia HTTP:ltä ja käsitelee ne ennen kuin luovuttaa ne kuljetuskerrokselle. Erityisesti kerros huolehtii salauksesta ja eheydestä. Se tekee seuraavat toimenpiteet:

- Jokainen ylemmältä tuleva sanoma pilkotaan lohkoiksi. Sen jälkeen jokaiseen lohkoon sovelletaan seuraavaa.
- Lohko tiivistetään.
- Lohkon MAC lasketaan.
- Tiivistetty lohko MAC-arvon kanssa salataan.
- Lopuksi viimeistellään SSL-tietueen otsake, joka sisältää mm. viestin tyypin, versionumerot ja lohkon pituuden.

Ylempi kerros I

Ylempi kerros toteuttaa SSL:n kättelyprotokollan, jonka avulla istunto perustetaan. Se käsittää 4 kierrosta ja sen jälkeen on todennettu käyttäjät, sovittu avaimista, salausmenetelmästä ja MAC-algoritmista. Kättely muodostuu seuraavista askelista (C client, S server):

1. $C \longrightarrow S$: *version, rand-1, session-id, cipher-list, compression-list*
   Eli asiakas lähettää istuntopyynnön palvelimelle ja ilmoittaa samalla istunnon numeron ja listat niistä salaus- ja tiivistysalgoritmeista, joita asiakas tukee. Lisäksi välittyy versionumero ja satunnaisluku, jota käytetään uusintahyökkäysten (vanha paketti lähetetään uudestaan) havaitsemiseen.
2. $S \longrightarrow C$: *version, ran-2, session-id, cipher, compression*
   Palvelin valitsee salaus- ja tiivistysalgoritmin listoista ja palauttaa tiedon näistä uuden satunnaisluvun kera asiakkaalle.
3. $S \longrightarrow C$: *server-cert*
   Palvelin lähettää myös oman varmenteensa.

Ylempi kerros II

4. $S \longrightarrow C$: $\{mod, exp, hash(rand\text{-}1, rand\text{-}2, mod, exp)\}_{K_S}$
   Palvelin lähettää varmenteeseensa liittyvät tunnusluvut eli modulonarvon ja julkisen avaimen sekä allekirjoittaa nuo tiedot salaisella avaimellaan. Allekirjoitus sisältää myös satunnaisluvut.
5. $S \longrightarrow C$: *cert-type, good-cert-authorities*
   Nyt palvelin pyytää asiakkaan varmennetta. Sen tulisi olla ehdotettua tyyppiä ehdotettujen luotettavien viranomaisten varmentama.
6. $S \longrightarrow C$: *end-round-2*
7. $C \longrightarrow S$: *client-cert*
   Asiakas lähettää pyydetyn varmenteen.
8. $C \longrightarrow S$: *premasterkey*
   Ja yhteisen salaisuuden, josta avaimia voidaan johtaa.

Ylempi kerros III

9. $C \longrightarrow S$: *hash(mastr, opad, hash, hash(messages, mastr, ipad))*
   Molemmat voivat laskea nyt pääavaimen yhteisestä salaisuudesta. Lisäksi lasketaan MAC:ia varten tarvittavat parametrit ipad ja opad. Nämä asiakas välittää myös palvelimelle ja niiden lisäksi myös kaikki aikaisemmat viestit. Tiedot lähetetään kryptografisen tiivistysfunktion kautta.
10. $C \longrightarrow S$: Viimeiset kaksi viestiä ovat eräänlaisia kuittauksia. Ensin asiakas ilmoittaa palvelimelle, että kaikki mikä seuraa seuraa on todennettua ja salattua (jos salauksesta on sovittu). Sen jälkeen lähtee vielä sanoma "finished".
11. $S \longrightarrow C$: Palvelin lähettää samanlaisen viestin asiakkaalle, sen jälkeen myös finished-sanoman ja kaiken tämän jälkeen varsinainen tietoliikenne voi alkaa.

Ylempi kerros IV

Tässä on kuvattu vain tilanne, jossa kumpikin todentaa toisensa. SSL sallii myös muunlaisia tilanteita. Lisäksi SSL:ssä on mahdollisuus uudelleen neuvotella istunto. Tähän on kuitenkin löydetty hyökkäysmahdollisuus, kts. [**?**].