iptables: lokitus

- Yksi erityinen kohde iptables-säännöille on **LOG**, jolla haluttu (epäilyttävä) paketti saadaan kirjattua lokiin. Lokitusta voidaan tarkemmin säädellä optioilla:

  --log-level *level* # syslog-tasot, debug, notice, info, warn, err jne

  --log-prefix *string* # merkkijono lokiviestien alkuun

  --log-tcp-sequence # kirjataan tcp-pakettien numerot lokiin

  --log-tcp-options # tcp-optiot lokiin

  --log-ip-options # ip-optiot lokiin

- Esim.

  iptables -A ... -j LOG --log-level debug --log-prefix ”nasty packet”

  iptables -A ... -j DROP

- syslog-facility on ”kern”, sitä ei voi vaihtaa. Ts. rsyslog.conf'issa tarvitaan esim.

  kern.debug /var/log/kernel.log

- Käytettävissä on myös ULOG-kohde, joka tarjoaa monipuolisempia lokitusmahdollisuuksia. Sen käyttää omaa protokollaansa ja sen käyttö edellyttää sitä varta vasten kuuntelevaa ohjelmaa (ulogd tms).

iptables: omat ketjut

- Jos samat säännöt toistuvat tai samaa ehtoa käytetään isolle joukolle sääntöjä, ne voi koota omaksi ketjukseen ja kutsua sitä tarvittaessa.
- Esimerkiksi lokitus yhdistettynä hylkäykseen:

  ```
  iptables -N LOGDROP
  iptables -A LOGDROP -j LOG --log-prefix ”bad packet”
  iptables -A LOGDROP -j DROP
  ```

  jota sitten käytetään tähän tapaan:

  ```
  iptables -A INPUT -s 134.170.0.0/16 -j LOGDROP
  ```

- Ellei oma ketju tee jotain joka päättää paketin käsittelyn (kuten DROP tai ACCEPT) sen loputtua käsittely jatkuu siellä mistä sitä kutsuttiin jos käytettiin -j:tä, tai käsittely päättyy kuten kutsuva ketju olisi loppunut jos sen sijaan käytettiin -g:tä. Ylläolevassa esimerkissä näillä ei olisi eroa, koska ehdoton DROP päättää käsittelyn kuitenkin. Ketju voidaan myös keskeyttää eksplisiittisesti kohteella RETURN.

iptables: FORWARD

- Jos palomuurikone jo reitittää liikennettä normaalisti, sen suodattaminen on helppoa, esim.

```
iptables -P FORWARD DROP
iptables -A FORWARD -d 192.168.122.0/24 -s 130.234.0.0/16 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 192.168.122.19 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 192.168.122.19 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
iptables -A FORWARD -s 192.168.122.0/24 -p udp --dport 53 ACCEPT
iptables -A FORWARD -s 192.168.122.0/24 -p tcp --dport 53 ACCEPT
```

- Yllä ESTABLISHED -sääntö toimii tarkoituksella molempiin suuntiin, vaikka ulospäin ei pääsisikään kuin DNS.
- Tässä siis reititys hoidetaan normaaliin tapaan reititystaululla eikä osoitteita eikä portteja muuteta, päätetään vain mitä välitetään eteenpäin ja mitä ei.
- Kaikki säännöt ovat forward-ketjussa ja filter-taulussa.
- Ftp ja muut vastaavat vaatisivat tässäkin omat lisäsäätönsä.

NAT: Network Address Translation

- Muuttaa paketin ip:n ja/tai portin mennen tullen.
- Alunperin viritys IP-osoitteiden säästämiseen, rikkoo joitakin protokollia; käynee harvinaisemmaksi IPv6:n yleistymisen myötä.
- Yleinen virtuaalikoneiden alustakoneen ja sen VM:ien välissä.
- SNAT (Source NAT), masquerade: vaihdetaan lähdeosoite ts. IP, josta paketti on tulossa (source address), yleensä privaatti-IP:stä julkiseksi, mahdollisesti useita samaksi (silloin vaihdetaan myös porttinumero); käytetään palomuurin sisältä ulos lähtevälle liikenteelle.
  - Lähtevälle paketille vaihdetaan IP:ksi palomuurin julkinen IP ja portiksi jokin (satunnainen vapaa) portti ja muistetaan se, paluupaketti ohjataan takaisin tallennetun porttinumero-IP -parin perusteella (connection tracking)
- DNAT (Destination NAT): vaihdetaan paketin kohdeosoite, esim. ohjataan palomuurin ulkopuolelta samaan osoitteeseen tulevia paketteja eri koneisiin palomuurin sisäpuolella (yleensä) porttinumeron perusteella.
- PNAT (Port NAT): yleinen termi edellisille silloin kun muutetaan osoite-portti -yhdistelmää.
- many-to-one NAT (erityisesti PNAT) vs. one-to-one NAT

iptables: SNAT

- SNAT (Source NAT) tarkoittaa lähtevän (välitettävän) paketin lähtöosoitteen (source address) vaihtamista; usein myös portti vaihdetaan. Yleisin käyttö tälle on palomuurin sisältä tulevan liikenteen yksityisten osoitteiden muuttaminen palomuurin ulkopuolen julkiseksi osoitteeksi vaihtaen samalla lähtöporttia vastausten ohjaamiseksi takaisin. Mahdollista on myös vain vaihtaa yksityinen osoite julkiseksi porttia muuttamatta (one-to-one NAT). Jos palomuurilla on useita julkisia osoitteita, niitä voidaan myös käyttää satunnaisesti tms.
- Tyyppitilanne: työasemia palomuurin takana yksityisillä osoitteilla, palomuurilla vain yksi julkinen osoite.
- SNAT tehdään POSTROUTING-ketjussa ja NAT-taulussa. Kohde voi olla SNAT tai MASQUERADE (tai joskus SAME tai NETMAP).

SNAT variantteja

- SNATin kanssa voi käyttää useita kohteita:

  ... -j SNAT --to-source *addr[-addr2][:port1-port2]*

  - *addr* määrää mitä lähtöosoitetta (yleensä jokin palomuurin julkisista osoitteista) käytetään (valitaan satunnaisesti jos useita)

  ... -j MASQUERADE [--to-ports *port1-port2*]

  - lähtöosoite määräytyy automaattisesti, unohtaa vanhat yhteydet verkkoyhteyden katketessa (hyvä jos IP on dynaaminen), hieman tehottomampi kuin SNAT

  ... -j SAME --to *addr1-addr2* [--nodst]

  - kuten SNAT monella osoitteella, mutta muistaa mitä lähtöosoitetta milläkin koneella on käytetty ja yrittää käyttää samaa kun kohdekin on sama (tai aina --nodst -optiolla)

  ... -j NETMAP --to *addr/mask*

  - kohdeosoitteena aina verkkoalue, joka on samankokoinen kuin lähtöalue (-s...), tekee 1-1 NATin pitäen osoitteen lopun samana. Esim.

    ... -s 172.20.209.0/24 … -j NETMAP 130.234.209.0/24

SNAT: esimerkki

- Liikenne lonka7:n sisäverkosta 192.168.122.0/24 olevista koneista (tunnus4 -koneet) ohjautuu ulos seuraavilla iptables-säännöillä (iptables -S -formaatti):

  -A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -p tcp -j MASQUERADE --to-ports 1024-65535

  -A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -p udp -j MASQUERADE --to-ports 1024-65535

  -A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -j MASQUERADE

  -A FORWARD -d 192.168.122.0/24 -o virbr0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

  -A FORWARD -s 192.168.122.0/24 -i virbr0 -j ACCEPT

- Uudelleenohjausta ei tehdä sisäverkon sisällä (! -d ...)
- Ohjaus tehdään erikseen tcp:lle ja udp:lle, joille halutaan määrätä portit (icmp ja muut eivät porttinumeroita käytä, joten niille ei voi käyttää --to-ports -optiota).
- --ctstate on conntrack -modulin versio --state -testistä (tässä toiminta sama, mutta conntrack sisältää paljon enemmän toiminnallisuutta ja on joskus tehokkaampi).
- Tämä on libvirtd:n oletusverkon NAT-säännöstö (siellä on lisäksi joitakin suodatussääntöjä).

iptables: DNAT

- DNAT (Destination NAT) tarkoittaa, että saapuvan (välitettävän) paketin kohdeosoite (destination address) vaihdetaan; myös portti voidaan vaihtaa. Yleisin käyttö tälle on ohjata samaan julkiseen osoitteeseen tulevat portit eri koneisiin palomuurin sisällä. Joskus myös vain vaihdetaan julkinen IP yksityiseksi (one-to-one NAT) porttiin puuttumatta, ja kohdekone palomuurin sisällä voidaan valita muullakin kuin portin perusteella (vaikkapa lähdeosoitteella).
- Yleinen erikoistapaus: virtuaalikoneiden alustakone ohjaa liikennettä valikoiden sisällään eri virtuaalikoneille.
- Voidaan käyttää myös transparent proxy-viritykseen toiseen suuntaan: esim. http- tai smtp-yhteys palomuurin sisältä ulos ohjataankin omaan välityspalvelimeen (tai sensuurin kynsiin...)
- DNAT tehdään PREROUTING-ketjussa ja NAT-taulussa, kohde DNAT (tai NETMAP).
- Jos DNATin tekee koneessa, joka ei muuten toimi reitittimenä ao. koneiden välissä, sen kanssa yleensä tarvitaan myös SNAT paluuliikennettä varten.

DNAT: esimerkki

- Ohjataan tt2-koneen (172.20.209.119) portti 80 koneeseen tt1 (172.20.209.19) palomuurisäännöillä tt2:ssa:

  # ohjataan porttiin 80 saapuva liikenne tt1:een:

  iptables -t nat -A PREROUTING -d 172.20.209.119 -p tcp -m tcp --dport 80 -j DNAT --to-dest 172.20.209.19

  # ohjataan paluuliikenne takaisin SNATilla:

  iptables -t nat -A POSTROUTING -s 172.20.209.19 -p tcp -m tcp --dport 80 -j MASQUERADE

  # sallitaan edelleenohjaus (tarpeen kun forward policy on drop):

  iptables -A FORWARD -d 172.20.209.19 -p tcp -m tcp --dport 80 -j ACCEPT

  # sallitaan paluuliikenteen uudelleenohjaus:

  iptables -A FORWARD -s 172.20.209.19 -p tcp -m state --state ESTABLISHED -j ACCEPT

  # ohjataan ja sallitaan tt2:sta itsestään lähtevä liikenne samalla tavalla:

  iptables -t nat -A OUTPUT -d 172.20.209.119 -p tcp -m tcp --dport 80 -j DNAT --to-dest 172.20.209.19

  iptables -A OUTPUT -s 172.20.209.119 -d 172.20.209.19 -p tcp --dport 80 -j ACCEPT

iptables: sekalaista

- Yleensä halutaan sallia kaikki liikenne localhostiin:

  iptables -A INPUT -i lo -j ACCEPT

  iptables -A OUTPUT -o lo -j ACCEPT
- Kerberos-PAM-autentikointia varten pitää sallia yhteys ulospäin Kerberos-porttiin (88):

  iptables -A OUTPUT -p tcp --dport 88 -j ACCEPT
- LDAP vaatii vastaavasti portin 389 (SSL:n kanssa 636):

  iptables -A OUTPUT -p tcp --dport 389 -j ACCEPT

  iptables -A OUTPUT -p tcp --dport 636 -j ACCEPT
- Yleensä jos jokin palvelu ei toimi, sen tarvitsemia porttinumeroita voi etsiä tiedostosta /etc/services (siellä voi olla lokaaleja lisäyksiä, joista Googlekaan ei tiedä)

iptables: anti-spoofing

- IP-osoitteiden väärentäjien ja verkkosotkujen varalta kannattaa blokata ulkopuolelta tulevat privaatti- yms osoitteet, joita ei tiedetä omiksi tai muuten tarpeellisiksi:

```
iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12  -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16  -j DROP
iptables -A INPUT -i eth0 -s 224.0.0.0/4      -j DROP
iptables -A INPUT -i eth0 -s 240.0.0.0/5      -j DROP
iptables -A INPUT -i eth0 -s 127.0.0.0/8  -j DROP
iptables -A INPUT -i eth0 -s 169.254.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 0.0.0.0/8        -j DROP
iptables -A INPUT -i eth0 -s 239.255.255.0/24 -j DROP
```

iptables: sekalaisia puolustuksia

- Fragmentit ovat nykyisin vain hyökkäyksiä

  iptables -A INPUT -f -j DROP

- NEW-paketit joissa ei ole SYN-bittiä

  iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP

- ”Joulupaketit” (kaikki tcp-liput yhtaikaa päällä)

  iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP

- Rikkinäiset NULL-paketit (ei mitään tcp-lippua päällä)

  iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

iptables: limit

- Testillä --limit voidaan rajoittaa tietynlaisen liikenteen määrää:

  ... -m limit --limit *n/aikayksikkö* [--limit-burst *m*]

  Optio --limit rajoittaa pakettien määrän per aikayksikkö (second, minute, hour, day) ja --limit-burst kuinka monta pakettia saa ensin tulla peräkkäin ennen kuin niitä aletaan rajoittaa.
- Esim. sallitaan vain 5 ssh-yhteyttä peräkkäin ja sitten 3/minuutti:

  ... -p tcp --dport 22 -m limit --limit 3/minute --limit-burst 5 -j ACCEPT
- Yleinen käyttö: estetään lokin täyttyminen, esim.

  -N LognDrop

  -A LognDrop -m limit --limit-burst 5 --limit 2/min -j LOG --log-prefix ”evil packet”

  -A LognDrop -j DROP
- Käytettävissä on myös --hashlimit, jolla voidaan tehdä monimutkaisempia rajoituksia tyyliin ”enintään 100 yhteydenottoa per portti-ja-palvelin” jne.

iptables: connlimit

- connlimit -ehdolla voidaan rajoittaa yhtaikaisten yhteyksien määrää, esim.

  ... -p tcp --syn --dport 80 -m connlimit --connlimit-above 20 --connlimit-mask 32 -j DROP

  rajoittaa porttiin 80 (http) samasta osoitteesta tulevien samanaikaisten yhteyksien määrän 20:een. Jos halutaan tehdä vastaava rajoitus yhteydenottajan aliverkolle, se voidaan tehdä --connlimit-mask -optiolla. Optio --syn rajaa testin uusiin yhteydenottoihin.

iptables: recent

- recent -ehdolla voidaan virittää aikaisemmista tapahtumista riippuvia ehtoja. Esimerkiksi ssh-massahyökkäysten torjunta:

```
# tehdään uusi ketju
iptables -N SshTest
# kutsutaan ketjua aina kun joku kolkuttaa ssh-porttia
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22 -m state --state NEW -j SshTest
# --set tallettaa tapahtuman joukkoon ”Ssh” (mielivaltainen tunniste)
iptables -A SshTest -m recent --set --name Ssh --rsource
# jos yrityksiä tulee enemmän kuin viisi kymmenessä minuutissa, lokitetaan...
iptables -A SshTest -m recent --update --seconds 600 --hitcount 5 --name Ssh --rsource -j LOG
--log-prefix "ssh-attack" --log-level warn
# ja blokataan ne
iptables -A SshTest -m recent --update --seconds 600 --hitcount 5 --name Ssh --rsource -j DROP
# muussa tapauksessa hyväksytään
iptables -A SshTest -j ACCEPT
```

iptables: NFS

- NFS:n toiminta edellyttää isoa joukkoa erillisiä palveluita, jotka tarvitsevat omat porttinsa auki palomuurissa. Asiaa vaikeuttaa se, että osa niistä arpoo portin satunnaisesti, ellei sitä erikseen kiinnitä. Tarvittavat palvelut ovat (suluissa portti jos se on kiinteä): portmapper (111), nfsd (2049), lockd, mountd ja statd, sekä levykiintiöitä käytettäessä rquotad.
- Muuttuvat portit voidaan kiinnittää seuraavasti (porttinumerot voi valita toisinkin):

  tiedostossa /etc/default/nfs-common:

  ```
  STATDOPTS="--port 4000 --outgoing-port 4001"
  ```

  tiedostossa /etc/default/nfs-kernel-server:

  ```
  RPCMOUNTDOPTS="--manage-gids --port 4002"
  ```

  tiedotossa (esim.) /etc/modprobe.d/local.conf:

  ```
  options lockd nlm_udpport=4003 nlm_tcpport=4003
  options nfs callback_tcpport=4004
  ```

  ja tiedostossa /etc/defaults/quota:

  ```
  RPCRQUOTADOPTS="-p 4005"
  ```

- Yllä valitut portit pitää sitten avata palvelimen palomuurissa tyyliin

  ```
  iptables -A INPUT -s  ... -p tcp --dport 111 -j ACCEPT
  iptables -A INPUT -s  ... -p tcp --dport 2049 -j ACCEPT
  iptables -A INPUT -s  ... -p tcp --dport 4000:4005 -j ACCEPT
  ```

  ja vastaavat OUTPUT-säännöt asiakkaisiin ja mahdolliseen välissä olevaan palomuuriin FORWARDit.
- Valitut portit kannattaa lisätä tiedostoon /etc/services.

iptables: debugging

- Palomuurisääntöjen syntaksivirheet saa helpoiten kiinni, kun ne kirjoittaa skriptiin ja ajaa sitä -x ja ehkä myös -e optioilla (alkuun ”#! /bin/bash -ex” tai suoritettaessa bash -ex ...): -x näyttää komennot niitä suoritettaessa ja -e keskeyttää skriptin virheen sattuessa.
- Uudet säännöstöt kannattaa testata ensin iptables-apply -komennolla, jos konetta säätää ssh:n tms verkkoyhteyden yli. (Jos konsoliyhteydenkin saa, se kannattaa silti varmuuden vuoksi avata.)
- Epäilyttävien DROP- ja REJECT-sääntöjen eteen voi laittaa LOG-säännön samalla ehdolla ja katsoa mitä lokissa näkyy. Joskus voi lokittaa myös läpi päästettyjä paketteja.
- Palomuurin toimintaa voi testata mm. komennoilla telnet (tcp-portit), ping (icmp), nmap (kaikki...) ja traceroute (etenkin useamman palomuurin läpi mentäessä).

telnet

- Telnet on alunperin tehty pääteyhteyden muodostamiseksi verkon yli, mutta koska siinä ei ole minkäänlaista salausta, siitä on luovuttu eikä telnet-demonia enää juuri missään käytetä. Telnet-client on kuitenkin yhä hyödyllinen avointen tcp-porttien testaamiseen ja tekstipohjaisten protokollien kokeilemiseen käsin:

  telnet *kone portti*

- Esim. kokeillaan postipalvelimen toimintaa:

  $ telnet smtp.example.org 25

  Trying 10.23.4.25...

  Connected to smtp.example.org.

  Escape character is '^]'.

  220 mail1.example.org ESMTP; Wed, 6 May 2015 07:51:39 +0300

  quit

  221 2.0.0 mail1.example.org closing connection

  Connection closed by foreign host.

traceroute

- Reititys- ja palomuuriongelmien selvittämisessä yksi perustyökalu on traceroute, joka yrittää selvittää mitä kautta paketti kulkee:

  traceroute [options] kone

- Usein mitään optioita ei tarvita, mutta jos jokin palomuuri välissä estää normaalin toiminnan, voi kokeilla vaihtoehtoisia menetelmiä:

  -I käytä ICMP ECHOa (vrt. ping)

  -T käytä TCP SYN-paketteja

- Muitakin optioita erikoistarpeisiin on, mm.

  -n älä hae koneiden nimiä DNS:stä

  -m *max_ttl* aseta yläraja hyppymäärälle (oletus 30)

  -w *waittime* kuinka monta sekuntia vastausta odotetaan (oletus 5)

nmap

- Erityisesti palomuurien toiminnan testaamiseen hyödyllinen työkalu: kokeilee mitkä portit kohdekoneissa ovat auki. Optioita on eri tarkoituksiin mahdottoman paljon, esim.

  -Pn (aikaisemmin -P0): älä pingaa konetta ensin (jos ping blokattu)

  -p22,1024-2047 kokeile portteja 22 ja 1024-2047

  -v verböösimpi tulostus (-vv vielä enemmän)

- Kohteena voi olla kone (nimi tai IP) tai verkkoalue tai useita, esim.

  nmap 172.20.208.16 # etsi avoimet portit lonka6:sta

  nmap --p80,443 172.20.209.0/24 # etsi www-palvelimet verkkoalueelta

- Erilaisia skannaustapoja löytyy man-sivulta lisää, jos kone ei vastaa vaikka pitäisi

identd

- ”identity daemon”: kertoo kuka (käyttäjä) tulee tietystä portista
- Käyttää tcp-porttia 113, avattava palomuurissa, esim.

  iptables -A INPUT -p tcp --dport 113 -j ACCEPT
- Vanha, ei nykyisin kovinkaan hyödyllinen tietoturvamekanismina, käyttäjätietoon ei voi luottaa (saattaa paljastaa osoitekaappauksia joskus, erityisesti jos identd vastaa mutta ei kerro käyttäjää (tcp-wrapperissa UNKNOWN@host) on syytä epäillä IP-spoofausta)
- Nykyisin oikeasti tarpeen lähinnä IRC-käyttäjille, voi olla käyttökelpoinen omassa verkossa tcp wrapperin kanssa muutenkin
- Useita identd-palvelinohjelmia, erityisesti:
  - nullidentd: triviaalitoteutus, ei kerro oikeita käyttäjätunnuksia vaan vakion tai satunnaisen
  - oidentd: yleinen, hyvin ylläpidetty, monipuolinen (käyttäjien voi antaa säätää omaa vastaustaan), konfiguraatiotiedosto /etc/oidentd.conf

tcp wrapper

- TCP Wrapper on yleinen mekanismi (kirjasto), jolla sisääntulevia yhteyksiä voidaan kontrolloida (nimestä huolimatta voi toimia UDP-palveluidenkin kanssa).
- Toiminnallisuus samankaltainen kuin iptables input filter, mutta toimii ylemmällä tasolla (application layer) ja voi suodattaa sovelluskohtaisesti säännöillä, joita pakettifiltteri ei näe (esim. käyttäjäkohtaisesti, myös salattuja yhteyksiä jne).
- Toiminta edellyttää, että ao. sovellus käyttää inetd:tä tai siinä itsessään on libwrap-kirjasto mukana. Yleisten palveluiden kuten ssh ja ftp kanssa se toimii.
- Usein samat (konekohtaiset) rajoitukset tehdään sekä pakettifiltterillä että tcp wrapperilla: jos yhteys ei toimi, tarkista molemmat.
- Etäkäyttäjän tunnistamiseen käytetään identd:tä, joka ei useinkaan ole käytettävissä, ja silloin käyttäjäkohtaiset rajoitukset eivät toimi.
- Konfiguraatiotiedostojen muuttaminen vaikuttaa heti, niitä ei tarvitse erikseen ladata eikä sovelluksia tarvitse käynnistää uudelleen.
- Soveltuu automaattiseen hyökkäysten torjumiseenkin, ks. esim. http://denyhosts.net

tcp wrapper 2

TCP Wrapperin säännöt kirjoitetaan tiedostoihin /etc/hosts.allow ja /etc/hosts.deny. Ensin luetaan hosts.allow, jos sieltä löytyy salliva sääntö yhteys sallitaan saman tien, ellei, luetaan hosts.deny ja jos ei löydy kieltävää sääntöä yhteys taas sallitaan.

- Molempien tiedostojen syntaksi on samanlainen:

  daemon_list : *client_list* [: *shell_command*]

  - *daemon_list* määrää palvelut joita sääntö koskee sitä ohjaavan demoniprosessin nimellä kuten ps sen näyttää, esim. sshd; useita voi luetella, jokereita voi käyttää, ALL tarkoittaa kaikkia, EXCEPT kaikkia paitsi sen jälkeen lueteltuja; jos koneella on useita nimiä niille voi tehdä eri sääntöjä syntaksilla *process@host*
  - *client_list* tarkoittaa koneita tai verkkoalueita, joita sääntö koskee; käyttäjäkohtainen sääntö voidaan tehdä syntaksilla *user@host*; jokereita voi käyttää, samoin ALL ja EXCEPT
  - *shell_command* on komento, joka suoritetaan (ensimmäiselle) täsmäävälle säännölle, esim. hosts.deny -tiedostossa voidaan lähettää varoitus hyökkäyksestä; % on erikoismerkki, jolla voidaan viitata mm. asiakaskoneeseen

tcp wrapper 3

- Demoni- ja client-listoissa toimivat seuraavat erikoismerkit:
  - pisteellä alkava merkkijono vastaa kaikkia siihen loppuvia (esim. ”.jyu.fi”)
  - pisteeseen loppuva osoite vastaa kaikkia sillä alkavia osoitteita (esim. ”130.234.”)
  - @:lla alkavat merkkijonot ovat NIS-ryhmien nimiä
  - verkkoalueen voi esittää muodossa n.n.n.n/m.m.m.m (esim. ”130.234.0.0/255.255.0.0”), IPv6-alueen [n.n.n.n.n.n.n.n]/m
  - kauttaviivalla alkava merkkijono tulkitaan tiedostonimeksi, josta nimiä tai osoitteita katsotaan (sielläkin voi olla erikoismerkkejä)
  - nimissä ja osoitteissa voi käyttää * ja ? -jokerimerkkejä, ei kuitenkaan yhdessä maskinotaation tai em. alku- tai loppupisteen kanssa
  - ALL tarkoittaa kaikkea, LOCAL koneita joiden nimessä ei ole pistettä, UNKNOWN tuntematonta käyttäjää tai konetta, jonka nimi **tai** osoite on tuntematon, KNOWN vastaavasti tunnettua käyttäjää tai konetta, jonka sekä nimi että osoite ovat tunnettuja
  - PARANOID tarkoittaa konetta, jonka nimi ei vastaa sen osoitetta
  - EXCEPT rajaa aikaisempaa ehtoa, voidaan toistaa: ”ALL EXCEPT *d EXCEPT sshd” sallisi kaikki paitsi d-loppuisista vain sshd:n

tcp wrapper 4

- *shell_command* tunnistaa seuraavat %-kirjain-yhdistelmät:
  - %a (%A): asiakaskoneen (palvelimen) osoite
  - %c: asiakastieto, user@kone tai vain kone tai ehkä vain osoite
  - %d: prosessin (demonin) nimi
  - %h (%H): asiakaskoneen (palvelimen) nimi, tai osoite jos nimi ei tiedossa
  - %n (%N): asiakaskoneen (palvelimen) osoite (tai ”unknown” tai ”paranoid”)
  - %p: demoniprosessin PID
  - %s: palvelutieto, daemon@host, daemon@address tai vain daemon
  - %u: asiakkaan käyttäjätunnus, tai ”unknown”
  - %%: prosenttimerkki
- jos komennon päättymistä ei haluta odottaa, sen loppuun voi laittaa &
- shell-komento ei aina ole käytettävissä, riippuu tcp wrapperin (käännösaikaisesta) asetuksesta; vaihtoehtoinen syntaksi ks. man 5 hosts_options

tcp_wrapper 5

- Jos halutaan sallia kaikki mitä ei ole erikseen kielletty (blacklisting): ei lainkaan hosts.allow -tiedostoa ja kiellot hosts.deny -tiedostoon, esim.

  sshd: 134.170. ALL@UNKNOWN

- Jos (suositeltavaa) halutaan kieltää kaikki mitä ei ole erikseen sallittu (whitelisting), tehdään hosts.deny yksinkertaisesti näin:

  ALL: ALL

  ja luetellaan sallitut hosts.allow -tiedostossa:

  ALL: 127.0.0.1

  vsftpd tftpd: ALL EXCEPT 134.170.

  sshd: 130.234. 172.20.

  sshd: .tapanitarvainen.fi

  Saman demonin voi siis antaa monellakin rivillä (usein selkeämpi kuin monen osoitteen luetteleminen tai jokerien käyttö).

- hosts.deny -tiedostossa ei siis voi tehdä poikkeuksia hosts.allow'ssa jo sallittuihin, sitä ei lueta lainkaan, jos hosts.allow'sta löytyi yhteyttä vastaava sääntö.
- Käyttäjärajauksia ei juuri kannata tehdä, käyttäjätunnistus ei ole kovin luotettava. Lähinnä vain ALL, KNOWN ja UNKNOWN ovat hyödyllisiä käyttäjäkentässä.