HowTo ASIX TCP-Wrappers

Curs 2015 - 2016

xinetd i TCPWrappers

Documentació

Aquest document ha estat elaborant utilitzant com a eina de treball un sistema GNU/Linux Fedora 20.

- Documentació de les pàgines man de les ordres.
- Fedora Documentation, Fedora 19. Security Guide: 3.5 TCP Wrappers and xinetd.

Ordres: /etc/hosts /etc/hosts.allow /etc/hosts.deny man hosts_access(5) man hosts_options(5) tcpd(8)
Ordres: man xinetd (8) man xinetd.conf (5)

xinetd

---

xinetd performs the same function as inetd: it starts programs that provide Internet services. Instead of having such servers started at system initialization time, and be dormant until a connection request arrives, xinetd is the only daemon process started and it listens on all service ports for the services listed in its configuration file. When a request comes in, xinetd starts the appropriate server. Because of the way it operates, xinetd (as well as inetd) is also referred to as a super-server.

The services listed in xinetd's configuration file can be separated into two groups. Services in the first group are called multi-threaded and they require the forking of a new server process for each new connection request. The new server then handles that connection. For such services, xinetd keeps listening for new requests so that it can spawn new servers.

On the other hand, the second group includes services for which the service daemon is responsible for handling all new connection requests. Such services are called single-threaded and xinetd will stop handling new requests for them until the server dies. Services in this group are usually datagram-based.

So far, the only reason for the existence of a super-server was to conserve system resources by avoiding to fork a lot of processes which might be dormant for most of their lifetime. While fulfilling this function, xinetd takes advantage of the idea of a super-server to provide features such as access control and logging. Furthermore, xinetd is not limited to services listed in /etc/services. Therefore, anybody can use xinetd to start special-purpose servers.

```
# yum install tftp tftp-server uw-imap telnet telnet-server

[root@i27 ~]# chkconfig
Nota: Esta salida muestra solo servicios SysV y no incluye servicios nativos de systemd.
Los datos de configuración SysV puede ser sobre escritos por configuración nativa de
systemd.
        Si desea una lista de servicios systemd use  'systemctl list-unit-files'.
        Para ver los servicios activados en un destino  particular use
```

```
        'systemctl list-dependencies [target]'.

netconsole                0:desactivado   1:desactivado   2:desactivado   3:desactivado
4:desactivado   5:desactivado   6:desactivado
network                   0:desactivado   1:desactivado   2:desactivado   3:desactivado
4:desactivado   5:desactivado   6:desactivado

servicios basados en xinetd:
   chargen-dgram:    desactivado
   chargen-stream:   desactivado
   daytime-dgram:    desactivado
   daytime-stream:   desactivado
   discard-dgram:    desactivado
   discard-stream:   desactivado
   echo-dgram:          desactivado
   echo-stream:         desactivado
   imap:                activo
   imaps:               desactivado
   ipop2:               desactivado
   ipop3:               activo
   pop3s:               activo
   tcpmux-server:    desactivado
   tftp:                activo
   time-dgram:          desactivado
   time-stream:         desactivado

[root@i27 ~]# ls -l /etc/xinetd.d/
total 68
-rw------- 1 root root 1157 dic 13  2013 chargen-dgram
-rw------- 1 root root 1159 dic 13  2013 chargen-stream
-rw------- 1 root root 1157 dic 13  2013 daytime-dgram
-rw------- 1 root root 1159 dic 13  2013 daytime-stream
-rw------- 1 root root 1157 dic 13  2013 discard-dgram
-rw------- 1 root root 1159 dic 13  2013 discard-stream
-rw------- 1 root root 1148 dic 13  2013 echo-dgram
-rw------- 1 root root 1150 dic 13  2013 echo-stream
-rw-r--r-- 1 root root  369 feb 17 10:07 imap
-rw-r--r-- 1 root root  365 ago  4  2013 imaps
-rw-r--r-- 1 root root  453 ago  4  2013 ipop2
-rw-r--r-- 1 root root  358 feb 17 10:13 ipop3
-rw-r--r-- 1 root root  334 feb 17 10:07 pop3s
-rw------- 1 root root 1212 dic 13  2013 tcpmux-server
-rw-r--r-- 1 root root  517 feb 13 10:40 tftp
-rw------- 1 root root 1149 dic 13  2013 time-dgram
-rw------- 1 root root 1150 dic 13  2013 time-stream
```

Configuració d'un servei

```
:w# cat /etc/xinetd.d/tftp
# default: off
# description: The tftp server serves files using the trivial file transfer \
#    protocol.  The tftp protocol is often used to boot diskless \
#    workstations, download configuration files to network-aware printers, \
#    and to start the installation process for some operating systems.
service tftp
{
   socket_type           = dgram
   protocol              = udp
   wait                  = yes
   user                  = root
   server                = /usr/sbin/in.tftpd
   server_args           = -s /var/lib/tftpboot
   disable               = no
   per_source            = 11
   cps                   = 100 2
   flags                 = IPv4
}
```

```
# cat ipop3
# default: off
# description: The POP3 service allows remote users to access their mail \
#                 using an POP3 client such as Netscape Communicator, mutt, \
#       or fetchmail.
service pop3
{
   socket_type           = stream
   wait                  = no
   user                  = root
   server                = /usr/sbin/ipop3d
   log_on_success    += HOST DURATION
   log_on_failure    += HOST
   disable               = no
}
```

Opcions de configuració:

disable
This is boolean "yes" or "no". This will result in the service being disabled and not starting. See the DIABLE flag description.

socket_type
Possible values for this attribute include:
stream stream-based service
dgram datagram-based service
raw service that requires direct access to IP
seqpacket service that requires reliable sequential datagram transmission

protocol

determines the protocol that is employed by the service. The protocol must exist in /etc/protocols. If this attribute is not defined, the default protocol employed by the service will be used.

wait
This attribute determines if the service is single-threaded or multi-threaded and whether or not xinetd accepts the connection or the server program accepts the connection. If its value is yes, the service is singlethreaded; this means that xinetd will start the server and then it will stop handling requests for the service until the server dies and that the server software will accept the connection.
If the attribute value is no, the service is *multi-threaded* and xinetd will keep handling new service requests and xinetd will accept the connection. It should be noted that udp/dgram services normally expect the value to be yes since udp is not connection oriented, while tcp/stream servers normally expect the value to be no.

user
determines the uid for the server process. The user attribute can either be numeric or a name. If a name isgiven (recommended), the user name must exist in /etc/passwd. This attribute is ineffective if the effective user ID of xinetd is not super-user.

group
determines the gid for the server process. The group attribute can either be numeric or a name. If a name isgiven (recommended), the group name must exist in /etc/group. If a group is not specified, the group of user will be used (from /etc/passwd). This attribute is ineffective if the effective user ID of xinetd is not superuser and if the groups attribute is not set to 'yes'.

server
determines the program to execute for this service.

server_args
determines the arguments passed to the server. In contrast to inetd, the server name should not be included in server_args.

nice
determines the server priority. Its value is a (possibly negative) number; check nice(3) for more information.

log_type
determines where the service log output is sent. There are two formats:
SYSLOG syslog_facility [syslog_level] The log output is sent to syslog at the specified facility. Possible facility names include: daemon, auth, authpriv, user, mail, lpr, news, uucp, ftp local0-7. Possible level names include: emerg, alert,crit, err, warning, notice, info, debug. If a level is not present, the messages will be recorded at theinfo level.
FILE file [soft_limit [hard_limit]]

log_on_success
determines what information is logged when a server is started and when that server exits

(the service id isalways included in the log entry). Any combination of the following values may be specified:
PID logs the server process id (if the service is implemented by xinetd without forking another process the logged process id will be 0)
HOST logs the remote host address
USERID logs the user id of the remote user using the RFC 1413 identification protocol. This option is available only for multi-threaded stream services.
EXIT logs the fact that a server exited along with the exit status or the termination signal (the process id is also logged if the PID option is used)
DURATION logs the duration of a service session
TRAFFIC logs the total bytes in and out for a redirected service.

log_on_failure
determines what information is logged when a server cannot be started (either because of a lack of resources orbecause of access control restrictions). The service id is always included in the log entry along with the reason for failure. Any combination of the following values may be specified:
HOST logs the remote host address.
USERID logs the user id of the remote user using the RFC 1413 identification protocol. This option is available only for multi-threaded stream services.
ATTEMPT logs the fact that a failed attempt was made (this option is implied by all others).

instances
determines the number of servers that can be simultaneously active for a service (the default is no limit). The value of this attribute can be either a number or UNLIMITED which means that there is no limit.

per_source
Takes an integer or "UNLIMITED" as an argument. This specifies the maximum instances of this service per source IP address. This can also be specified in the defaults section.

cps
Limits the rate of incoming connections. Takes two arguments. The first argument is the number of connections per second to handle. If the rate of incoming connections is higher than this, the service will be temporarily disabled. The second argument is the number of seconds to wait before re-enabling the service after it has been disabled. The default for this setting is 50 incoming connections and the interval is 10 seconds.

max_load
Takes a floating point value as the load at which the service will stop accepting connections. For example: 2 or 2.5. The service will stop accepting connections at this load. This is the one minute load average. This is an OS dependent feature, and currently only Linux, Solaris, and FreeBSD are supported for this. This feature is only avaliable if xinetd was configured with the -with-loadavg option.

only_from

determines the remote hosts to which the particular service is available. Its value is a list of IP addresses which can be specified in any combination of the following ways:

a) a numeric address in the form of %d.%d.%d.%d. If the rightmost components are 0, they are treated as wildcards (for example, 128.138.12.0 matches all hosts on the 128.138.12 subnet). 0.0.0.0 matches all Internet addresses. IPv6 hosts may be specified in the form of abcd:ef01::2345:6789. The rightmost rule for IPv4 addresses does not apply to IPv6 addresses.

b) a factorized address in the form of %d.%d.%d.{%d,%d,...}. There is no need for all 4 components (i.e. %d.%d.{%d,%d,...%d} is also ok). However, the factorized part must be at the end of the address. This form does not work for IPv6 hosts.

c) a network name (from /etc/networks). This form does not work for IPv6 hosts.

d) a host name. When a connection is made to xinetd, a reverse lookup is performed, and the canonical name returned is compared to the specified host name. You may also use domain names in the form of .domain.com. If the reverse lookup of the client's IP is within .domain.com, a match occurs.

e) an ip address/netmask range in the form of 1.2.3.4/32. IPv6 address/netmask ranges in the form of 1234::/46 are also valid.

Specifying this attribute without a value makes the service available to nobody.

no_access
determines the remote hosts to which the particular service is unavailable. Its value can be specified in the same way as the value of the only_from attribute. These two attributes determine the location access control enforced by xinetd. If none of the two is specified for a service, the service is available to anyone. If both are specified for a service, the one that is the better match for the address of the remote host determines if the service is available to that host (for example, if the only_from list contains 128.138.209.0 and the no_access list contains 128.138.209.10 then the host with the address 128.138.209.10 can not access the service).

access_times
determines the time intervals when the service is available. An interval has the form hour:min-hour:min (connections will be accepted at the bounds of an interval). Hours can range from 0 to 23 and minutes from 0 to 59.

redirect
Allows a tcp service to be redirected to another host. When xinetd receives a tcp connection on this port it spawns a process that establishes a connection to the host and port number specified, and forwards all data between the two hosts. This option is useful when your internal machines are not visible to the outside world.
Syntax is: redirect = (ip address) (port). You can also use a hostname instead of the IP address in this field.
The hostname lookup is performed only once, when xinetd is started, and the first IP address returned is the one that is used until xinetd is restarted. The "server" attribute is not required when this option is specified. If the "server" attribute is specified, this

attribute takes priority.

rlimit_as
Sets the Address Space resource limit for the service. One parameter is required, which is either a positive integer representing the number of bytes to set the limit to (K or M may be used to specify kilobytes/megabytes) or "UNLIMITED". Due to the way Linux's libc malloc is implemented, it is more useful to set this limit than rlimit_data, rlimit_rss and rlimit_stack. This resource limit is only implemented on Linux systems.

rlimit_files
Sets the maximum number of open files that the service may use. One parameter is required, which is a positive integer representing the number of open file descriptors. Practical limit of this number is around 1024000.

rlimit_cpu
Sets the maximum number of CPU seconds that the service may use. One parameter is required, which is either a positive integer representing the number of CPU seconds limit to, or "UNLIMITED".

rlimit_data
Sets the maximum data size resource limit for the service. One parameter is required, which is either a positive integer representing the number of bytes or "UNLIMITED".

rlimit_rss
Sets the maximum resident set size limit for the service. Setting this value low will make the process a likely candidate for swapping out to disk when memory is low. One parameter is required, which is either a positive integer representing the number of bytes or "UNLIMITED".

rlimit_stack
Set the maximum stack size limit for the service. One parameter is required, which is either a positive integer representing the number of bytes or "UNLIMITED".

deny_time
Sets the time span that access to all services on all IP addresses are denied to someone that sets off the SENSOR. The unit of time is in minutes. Valid options are: FOREVER, NEVER, and a numeric value. FOREVER causes the IP address not to be purged until xinetd is restarted. NEVER has the effect of just logging the offending IP address. A typical time value would be 60 minutes. This should stop most DOS attacks while allowing IP addresses that come from a pool to be recycled for legitimate purposes. This option must be used in conjunction with the SENSOR flag.

Configuració mínima

You don't need to specify all of the above attributes for each service. The necessary attributes for a service are:

socket_type

```
        user            (non-internal services only)
        server          (non-internal services only)
        wait
        protocol        (RPC and unlisted services only)
        rpc_version     (RPC services only)
        rpc_number      (unlisted RPC services only)
        port            (unlisted non-RPC services only)
```

Exemples

Exemples Generals

```
defaults
        {
                log_type        = FILE /var/log/servicelog
                log_on_success          = PID
                log_on_failure          = HOST
                only_from       = 128.138.193.0 128.138.204.0
                only_from       = 128.138.252.1
                instances       = 10
                disabled        = rstatd
        }
```

```
service login
        {
                socket_type             = stream
                protocol        = tcp
                wait            = no
                user            = root
                server          = /usr/etc/in.rlogind
                instances       = UNLIMITED
        }
```

```
service ftp
        {
                socket_type             = stream
                wait            = no
                nice            = 10
                user            = root
                server          = /usr/etc/in.ftpd
                server_args             = -l
                instances       = 4
                log_on_success          += DURATION HOST USERID
                access_times            = 2:00-9:00 12:00-24:00
        }
```

```
service telnet
        {
                socket_type     = stream
```

```
        wait            = no
        nice            = 10
        user            = root
        server          = /usr/etc/in.telnetd
        rlimit_as       = 8M
        rlimit_cpu      = 20
}
```

```
service unlisted
{
        type            = UNLISTED
        socket_type     = stream
        protocol        = tcp
        wait            = no
        server          = /home/user/some_server
        port            = 20020
}
```

```
# Access restrictions
        only_from       = 192.168.0.31 192.168.2.0
        no_access       = 192.168.2.57
        access_times    = 02:00-13:00


[root@i27 ~]# telnet i27 13
Trying 192.168.2.57...
Connected to i27.
Escape character is '^]'.
Connection closed by foreign host.

[root@i26 ~]# telnet i27 13
Trying 192.168.2.57...
Connected to i27.
Escape character is '^]'.
19 FEB 2015 12:00:36 CET
Connection closed by foreign host.
```

Exemples Redirect

Redirigeix el tràfic dle host 192.168.1.34 port 81 (la ip publica del propi host) al port 80 del host 192.168.1.41. És a dir, accedint al port 81 local en realitat s'accedeix al 80 del remot.

```
service http-switch
{
  disable       = no
  type          = UNLISTED
  socket_type   = stream
```

```
  protocol        = tcp
  wait            = no
  redirect        = 192.168.1.41 80
  bind            = 192.168.1.34
  port            = 81
  user            = nobody
}
```

```
# telnet 192.168.1.34 81
Trying 192.168.1.34...
Connected to 192.168.1.34.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 19 Feb 2015 21:32:01 GMT
Server: Apache/2.2.23 (Fedora)
Last-Modified: Sat, 21 Dec 2013 17:38:43 GMT
ETag: "a3e8b-1c-4ee0eddaefebd"
Accept-Ranges: bytes
Content-Length: 28
Connection: close
Content-Type: text/html; charset=UTF-8

web PRINCIPAL o main server
Connection closed by foreign host.
```

Redirigeix les peticions al port 7 de la ip publica del host local (server-echo) al servidor daytime-stream (port 13) del host remot.

```
service echo
{
# This is for quick on or off of the service
    disable          = no
    type             = UNLISTED
    protocol         = tcp
    wait             = no
    socket_type      = stream
    port             = 7
    redirect         = 192.168.1.41 13
    bind             = 192.168.1.34
    user             = nobody
}
```

```
# telnet 192.168.1.34 7
Trying 192.168.1.34...
Connected to 192.168.1.34.
… mostra la data el servidor daytime del host .41...
```

Exercicis proposats

Practicar de xinetd:

- Activar serveis echo-stream (7) i daytime-stream(13)
- Crear redirecció daytime-bis (14)
  - redirect = 127.0.0.1 13
  - bind = 192.168.1.37
  - port = 14
  - user = nobody
- Aplicar-li restriccions de:
  - Un host si, un host no.
  - Una xarxa si pero un host de la xarxa no.
  - Una xarxa no però un host si.
  - Restriccions per hora. Primer tot si. Ara aquesta hora no.
  - Definir diferents formats de log. Examinar amb journalctl -b -u xinetd
  - #only_from = 192.168.0.0/0.0.255.255
  - only_from = 192.168.1.37
  - access_times = 2:00-23:00
  - log_on_success += PID HOST USERID DURATION EXIT TRAFFIC
  - log_on_failure += HOST USERID ATTEMPT
- Crear redirecció echo-bis (8)
  - redirect = 127.0.0.1 7
  - bind = 192.168.1.37
  - port = 8
  - user = nobody
- Ampliar el servei establint límits de connexions:
  - Establir un màxim de dues instàncies. Provar tres connexions (última no permesa).
  - Establir els cps per exemple 2 connexions màxim per segon i esperar 20 segons si es supera. provar-ho fent: $ telnet 192.168.1.37 8 & telnet 192.168.1.37 8 & telnet 192.168.1.37 8 &
  - Observar que els dos primers s'executen, al tercer li cal esperar 20 segons.
  - instances = 2
  - cps = 1 20

TCP Wrappers

Fedora 19 Security Guide:

> Controlling access to network services is one of the most important security tasks facing a server administrator. Fedora provides several tools for this purpose. For example, an iptables-based firewall filters out unwelcome network packets within the kernel's network stack. For network services that utilize it, TCP Wrappers add an additional layer of protection by defining which hosts are or are not allowed to connect to "***wrapped***" network services. One such wrapped network service is the ***xinetd*** super server. This service is called a super server because it controls connections to a subset of network services and further refines access control.

> The TCP Wrappers package (tcp_wrappers) is installed by default and provides host-based access control to network services. The most important component within the package is the ***/usr/lib/libwrap.a*** library. In general terms, a TCP-wrapped service is one that has been compiled against the ***libwrap.a*** library.
>
> When a connection attempt is made to a TCP-wrapped service, the service first references the host's access files (***/etc/hosts.allow*** and ***/etc/hosts.deny***) to determine whether or not the client is allowed to connect. In most cases, it then uses the syslog daemon (syslogd) to write the name of the requesting client and the requested service to /var/log/secure or /var/log/messages.
>
> If a client is allowed to connect, TCP Wrappers release control of the connection to the requested service and take no further part in the communication between the client and the server.
>
> In addition to access control and logging, TCP Wrappers can execute commands to interact with the client before denying or releasing control of the connection to the requested network service.

> TCP Wrappers provide the following advantages over other network service control techniques:
>
> • Transparency to both the client and the wrapped network service — Both the connecting client and the wrapped network service are unaware that TCP Wrappers are in use. Legitimate users are logged and connected to the requested service while connections from banned clients fail.
>
> • Centralized management of multiple protocols — TCP Wrappers operate separately from the network services they protect, allowing many server applications to share a common set of access control configuration files, making for simpler management.

```
$ locate libwrap
/usr/lib/libwrap.so.0
/usr/lib/libwrap.so.0.7.6
/usr/lib/samba/libwrap_xattr.so

$ rpm -qf /usr/lib/libwrap.so.0
tcp_wrappers-libs-7.6-69.fc17.i686

$ rpm -ql tcp_wrappers
/usr/sbin/safe_finger
/usr/sbin/tcpd
/usr/sbin/tcpdmatch
/usr/sbin/try-from
/usr/share/doc/tcp_wrappers-7.6
/usr/share/doc/tcp_wrappers-7.6/BLURB
/usr/share/doc/tcp_wrappers-7.6/Banners.Makefile
/usr/share/doc/tcp_wrappers-7.6/CHANGES
/usr/share/doc/tcp_wrappers-7.6/DISCLAIMER
/usr/share/doc/tcp_wrappers-7.6/README
/usr/share/doc/tcp_wrappers-7.6/README.IRIX
/usr/share/doc/tcp_wrappers-7.6/README.NIS
/usr/share/man/man8/safe_finger.8.gz
/usr/share/man/man8/tcpd.8.gz
/usr/share/man/man8/tcpdmatch.8.gz
/usr/share/man/man8/try-from.8.gz
```

```
$ ldd $(which sshd) | grep libwrap
   libwrap.so.0 => /lib/libwrap.so.0 (0xb76ba000)

$ ldd $(which xinetd) | grep libwrap
   libwrap.so.0 => /lib/libwrap.so.0 (0xb7698000)

[root@i27 ~]# ldd /sbin/vsftpd | grep libwrap
   libwrap.so.0 => /lib64/libwrap.so.0 (0x00007f36d942c000)
[root@i27 ~]# ldd /sbin/in.tftpd | grep libwrap
   libwrap.so.0 => /lib64/libwrap.so.0 (0x00007ff85bc7c000)

[root@i27 ~]# ldd /sbin/httpd | grep libwrap
[root@i27 ~]# ldd /sbin/in.telnetd | grep libwrap
```

Resolució

Passos que es segueixen per a la resolució d'una connexió client usant TCP-Wrappers.

To determine if a client is allowed to connect to a service, TCP Wrappers reference the following two files, which are commonly referred to as hosts access files:

• /etc/hosts.allow
• /etc/hosts.deny

When a TCP-wrapped service receives a client request, it performs the following steps:

1. It references /etc/hosts.allow. — The TCP-wrapped service sequentially parses the /etc/hosts.allow file and applies the first rule specified for that service. If it finds a matching rule, it allows the connection. If not, it moves on to the next step.

2. It references /etc/hosts.deny. — The TCP-wrapped service sequentially parses the /etc/hosts.deny file. If it finds a matching rule, it denies the connection. If not, it grants access to the service.

The following are important points to consider when using TCP Wrappers to protect network services:

• Because access rules in hosts.allow are applied first, they take precedence over rules specified in hosts.deny. Therefore, if access to a service is allowed in hosts.allow, a rule denying access to that same service in hosts.deny is ignored.

• The rules in each file are read from the top down and the first matching rule for a given service is the only one applied. The order of the rules is extremely important.

• If no rules for the service are found in either file, or if neither file exists, access to the service is granted.

• TCP-wrapped services do not cache the rules from the hosts access files, so any changes to hosts.allow or hosts.deny take effect immediately, without restarting network services.

Format de les regles bàsic:

daemon_list : client_list [: shell_command]

daemon_list is a list of one or more daemon process names (argv[0] values) or wildcards (see below).

client_list is a list of one or more host names, host addresses, patterns or wildcards (see below) that will be matched against the client host name or address.

Format de les regles estès:

The extensible language uses the following format:

daemon_list : client_list : option : option ...

The first two fields are described in the hosts_access(5) manual page. The remainder of the rules is a list of zero or more options. Any ":" characters within options should be protected with a backslash.

An option is of the form "keyword" or "keyword value". Options are processed in the specified order. Some options are subjected to %<letter> substitutions. For the sake of backwards compatibility with earlier versions, an "=" is permitted between keyword and value.

Exemples de ACLs

```
vsftpd : .example.com

sshd : .example.com  \ : spawn /bin/echo `/bin/date` access denied>>/var/log/sshd.log \ :
deny

ALL : .example.com
ALL : 192.168.
ALL : 192.168.0.0/255.255.254.0
ALL : [3ffe:505:2:1::]/64
ALL : *.example.com
vsftpd : /etc/ftp.hosts

ALL: .example.com EXCEPT cracker.example.com
ALL EXCEPT vsftpd: 192.168.0.

sshd : .example.com : severity emerg
sshd : .example.com : severity local0.alert

sshd : client-1.example.com : allow
sshd : client-2.example.com : deny

vsftpd : .example.com \
: spawn /bin/echo `/bin/date` from %h>>/var/log/telnet.log \
: allow

vsftpd : .example.com \
: twist /bin/echo "421 This domain has been black-listed. Access denied!"

sshd : .example.com \
: spawn /bin/echo `/bin/date` access denied to %h>>/var/log/sshd.log \
: deny

vsftpd : .example.com \
: twist /bin/echo "421 %h has been banned from this server!"
```

wildcards

The following wildcards are available:
• ALL — Matches everything. It can be used for both the daemon list and the client list.
• LOCAL — Matches any host that does not contain a period (.), such as localhost.
• KNOWN — Matches any host where the hostname and host address are known or where the user is known.
• UNKNOWN — Matches any host where the hostname or host address are unknown or where the user is unknown.
• PARANOID — Matches any host where the hostname does not match the host address.

• Hostname beginning with a period (.) — Placing a period at the beginning of a hostname matches all hosts sharing the listed components of the name.

• IP address ending with a period (.) — Placing a period at the end of an IP address matches all hosts sharing the initial numeric groups of an IP address.

• IP address/netmask pair — Netmask expressions can also be used as a pattern to control access to a particular group of IP addresses.

• [IPv6 address]/prefixlen pair — [net]/prefixlen pairs can also be used as a pattern to control access to a particular group of IPv6 addresses.

• The asterisk (*) — Asterisks can be used to match entire groups of hostnames or IP addresses, as long as they are not mixed in a client list containing other types of patterns.

• The slash (/) — If a client list begins with a slash, it is treated as a file name. This is useful if rules specifying large numbers of hosts are necessary.

Operators

Except The EXCEPT operator allows specific exceptions to broader matches within the same rule.

severity Option fields let administrators easily change the log facility and priority level for a rule by using the
severity directive.

Access Control Option fields also allow administrators to explicitly **allow** or **deny** hosts in a single rule by adding the allow or deny directive as the final option.

Shell Commands

• **spawn** — Launches a shell command as a child process. This directive can perform tasks like using /usr/sbin/safe_finger to get more information about the requesting client or create special log files using the echo command.

• **twist** — Replaces the requested service with the specified command. This directive is often used to set up traps for intruders (also called "honey pots"). It can also be used to send messages to connecting clients. The twist directive must occur at the end of the rule line.

Expansions

The following is a list of supported expansions:

• %a — Returns the client's IP address.
• %A — Returns the server's IP address.
• %c — Returns a variety of client information, such as the username and hostname, or the username and IP address.
• %d — Returns the daemon process name.
• %h — Returns the client's hostname (or IP address, if the hostname is unavailable).
• %H — Returns the server's hostname (or IP address, if the hostname is unavailable).
• %n — Returns the client's hostname. If unavailable, unknown is printed. If the client's hostname and host address do not match, paranoid is printed.
• %N — Returns the server's hostname. If unavailable, unknown is printed. If the server's hostname and host address do not match, paranoid is printed.
• %p — Returns the daemon's process ID.
• %s —Returns various types of server information, such as the daemon process and the host or IP address of the server.
• %u — Returns the client's username. If unavailable, unknown is printed.

Exemples de filtrat TCPWrappers

Exemple 1 Regles Acls

```
# hostname
i27.informatica.escoladeltreball.org
# yum install uw-imap

# chkconfig
servicios basados en xinetd:
    chargen-dgram:     desactivado
    chargen-stream:    desactivado
    daytime-dgram:     desactivado
    daytime-stream:    desactivado
    discard-dgram:     desactivado
    discard-stream:    desactivado
    echo-dgram:          desactivado
    echo-stream:         desactivado
```

```
   imap:            activo
   imaps:           desactivado
   ipop2:           desactivado
   ipop3:           activo
   pop3s:           activo
   tcpmux-server:   desactivado
   tftp:            activo
   time-dgram:      desactivado
   time-stream:     desactivado

# systemctl start httpd
# systemctl start vsftpd
# systemctl start xinetd
```

```
[root@i27 ~]# ls -l /etc/hosts.{allow,deny}
-rw-r--r--  1 root root 705 feb 13 10:39 /etc/hosts.allow
-rw-r--r--. 1 root root 460 jun  7  2013 /etc/hosts.deny

# cat /etc/hosts.allow
#
# hosts.allow    This file contains access rules which are used to
#          allow or deny connections to network services that
#          either use the tcp_wrappers library or that have been
#          started through a tcp_wrappers-enabled xinetd.
#
#          See 'man 5 hosts_options' and 'man 5 hosts_access'
#          for information on rule syntax.
#          See 'man tcpd' for information on tcp_wrappers
#
vsftpd : 192.168.  \
         : spawn /bin/echo -e "$(/bin/date) from %h to %H, \n\t\tclient %c - %u, daemon %d
- %p, server %s ==> vsftpd"  >> /var/log/proves.log

in.tftpd: 192.168. \
         : spawn /bin/echo "$(/bin/date) from %h (%d, %p, tftpd)" >> /var/log/proves.log

sshd: i27.* \
         : spawn /bin/echo "$(/bin/date) from (%h, %c) edt (%d, %p, sshd)" >>
/var/log/proves.log
```

```
[root@i26 ~]# ftp i27
Connected to i27 (192.168.2.57).
220 (vsFTPd 3.0.2)
Name (i27:root): anonymous
```

```
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.

# telnet i27 80
Trying 192.168.2.57...
Connected to i27.
Escape character is '^]'.
GET / HTTP/1.0

[root@i26 ~]# tftp i27
tftp> get hola.txt
```

```
[root@i27 ~]# cat /var/log/proves.log
Tue Feb 17 08:38:52 CET 2015 from 192.168.2.56 edt vsftpd
Tue Feb 17 09:39:31 CET 2015 from 192.168.2.57 to 192.168.2.57,
        client 192.168.2.57 - unknown, daemon vsftpd - 1411, server
vsftpd@192.168.2.57 ==> vsftpd
Tue Feb 17 09:40:17 CET 2015 from 192.168.2.57 i26 tftpd
```

Observeu el fitxer de log proves.log. Recordatori de la regla generada i llistat dels logs del vsftpd, tftpd, telnet i ssh.

```
vsftpd : 192.168.  \
        : spawn /bin/echo -e "$(/bin/date) from %h to %H, \n\t\tclient %c - %u, daemon %d
- %p, server %s ==> vsftpd"  >> /var/log/proves.log
```

```
Tue Feb 17 09:03:33 CET 2015 from 192.168.2.56 to 192.168.2.57,
        client 192.168.2.56 - unknown, daemon vsftpd - 1476, server
vsftpd@192.168.2.57 ==> vsftpd

Tue Feb 17 09:49:22 UTC 2015 from 192.168.2.56 (in.tftpd, 1806, tftpd)

Tue Feb 17 09:56:50 CET 2015 from i27.informatica.escoladeltreball.org edt telnet.socket

Tue Feb 17 09:58:21 CET 2015 from (i27.informatica.escoladeltreball.org,
i27.informatica.escoladeltreball.org) edt (sshd, 2092, sshd)
```

Exemple ssh i fitxer de logs:

```
sshd: i27.* \
          : spawn /bin/echo "$(/bin/date) from (%h, %c) edt (%d, %p, sshd)"  \
            >> /var/log/proves.log : deny

sshd: *.informatica.escoladeltreball.org \
          : spawn /bin/echo "$(/bin/date) from (%h, %c) edt (%d, %p, sshd)"  \
           >> /var/log/proves.log
```

```
[root@i27 ~]# ssh i27
ssh_exchange_identification: read: Connection reset by peer

Tue Feb 17 10:00:59 CET 2015 from (i26.informatica.escoladeltreball.org,
i26.informatica.escoladeltreball.org) edt (sshd, 2174, sshd)
```

```
[root@i27 ~]# nmap i27
Starting Nmap 6.45 ( http://nmap.org ) at 2015-02-17 10:16 CET
Nmap scan report for i27 (192.168.2.57)
Host is up (0.0000060s latency).
Not shown: 992 closed ports
PORT  STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
110/tcp open  pop3
111/tcp open  rpcbind
143/tcp open  imap
995/tcp open  pop3s

#  cat /etc/hosts.allow
imapd: ALL \
   : spawn /bin/echo "$(/bin/date) from %h (%d, %p, imapd)" >> /var/log/proves.log : allow

ipop3d: .informatica.escoladeltreball.org \
   : spawn /bin/echo "$(/bin/date) from %h (%d, %p, imapd)" >> /var/log/proves.log : allow

# cat /var/log/proves.log
Tue Feb 17 10:18:31 CET 2015 from 192.168.2.57 (imapd, 2527, imapd)
Tue Feb 17 10:22:13 CET 2015 from i27.informatica.escoladeltreball.org (ipop3d, 2540,
ipop3d)
```

Exemple 2: múltiples ordres

```
ipop3d: .informatica.escoladeltreball.org \
          : spawn /bin/echo "$(/bin/date) from %h (%d, %p, ipop3d)" >> /var/log/proves.log \
```

```
        : spawn /usr/bin/nmap %H | mail root -s "ipop3s" \
        : allow

Tue Feb 17 10:31:29 CET 2015 from 192.168.2.57 (imapd, 2720, imapd)

[root@i27 ~]# mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/root": 1 message
>   1 root                  Tue Feb 17 10:31 133/6732
& 1
Message  1:
From root@i27.informatica.escoladeltreball.org  Tue Feb 17 10:35:19 2015
Return-Path: <root@i27.informatica.escoladeltreball.org>
From: root <root@i27.informatica.escoladeltreball.org>
Date: Tue, 17 Feb 2015 10:35:18 +0100
To: ipop3s@i27.informatica.escoladeltreball.org,
        -s@i27.informatica.escoladeltreball.org,
        root@i27.informatica.escoladeltreball.org
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
Status: R

Starting Nmap 6.45 ( http://nmap.org ) at 2015-02-17 10:35 CET
Nmap scan report for 192.168.2.57
Host is up (0.0000060s latency).
Not shown: 992 closed ports
PORT  STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
110/tcp open  pop3
111/tcp open  rpcbind
143/tcp open  imap
995/tcp open  pop3s

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
```

Exemple 3: exemple de classe: hosts.deny

```
sshd: 192.168.1.37 :  \
        spawn /bin/echo -e "$(/bin/date) from %h to %H, \n\t\tclient %c - %u, daemon %d - %p, server %s ==> sshd" >> /var/log/sshd-local.log \
        : spawn /bin/echo "$(pstree -sa 2081) " >> /var/log/sshd-local.log \
        : spawn /bin/echo "%h %H %c %u %d %p %s " >> /var/log/sshd-local.log \
        : spawn /bin/touch /tmp/tag-$(date| tr ' ' '_') \
        : allow
```

```
vsftpd: 192.168.1.37 :  \
        spawn /bin/echo -e "$(/bin/date) from %h to %H, \n\t\tclient %c - %u, daemon %d - %p, server %s ==> vsftpd" >> /var/log/sshd-local.log \
        : spawn /bin/echo "$(pstree -sa 2081) " >> /var/log/sshd-local.log \
        : spawn /bin/echo "%h %H %c %u %d %p %s " >> /var/log/sshd-local.log \
        : spawn /bin/touch /tmp/tag-$(date| tr ' ' '_') \
        : twist /bin/echo -e "tarari que te vi \n\n\n" | /bin/pstree -sa 2081 | who | /bin/cal
```

Exercicis proposats

Practicar TCPWrappers

- ❏ Establir regles ACL de filtrar de serveis:
  - ❏ Un host no, un host si.
  - ❏ Una xarxa no pero un host si.
  - ❏ Una xarxa si però un host no.
  - ❏ Política permisiva per defecte.
  - ❏ Politica restrictiva per defecte.
  - ❏ Exemplles de usar fitxers:
    - ❏ file allow + file deny
    - ❏ només file allow
    - ❏ només file deny
    - ❏ usar except.
- ❏ Exemple mostrant com spawn genera subprocessos fills. Generar echos que mostrin l'arbre de processos, amb pstreee del pid del servei:

  ```
  sshd: 192.168.1.37 :  \
  spawn /bin/echo -e "$(/bin/date) from %h to %H, \n\t\tclient %c - %u, daemon %d - %p, server %s ==> sshd" >> /var/log/sshd-local.log \
  : spawn /bin/echo "$(pstree -sa 2081) " >> /var/log/sshd-local.log \
  : spawn /bin/echo "%h %H %c %u %d %p %s " >> /var/log/sshd-local.log \
  : spawn /bin/touch /tmp/tag-$(date| tr ' ' '_') \
  : allow
  ```

- ❏ Exemple mostrar com twist genera un nou procés que substitueix el del daemon del servei. Per exemple fa un cal.

  ```
  vsftpd: 192.168.1.37 :  \
  spawn /bin/echo -e "$(/bin/date) from %h to %H, \n\t\tclient %c - %u, daemon %d - %p, server %s ==> vsftpd" >> /var/log/sshd-local.log \
  : spawn /bin/echo "$(pstree -sa 2081) " >> /var/log/sshd-local.log \
  : spawn /bin/echo "%h %H %c %u %d %p %s " >> /var/log/sshd-local.log \
  : spawn /bin/touch /tmp/tag-$(date| tr ' ' '_') \
  : twist /bin/echo -e "tarari que te vi \n\n\n" | /bin/cal
  ```

- ❏ El diàleg client servidor és segons el protocol usat, per tant el que genera twist no necessàriament és interpretat pel client, que espera del servidor la seva part de dialeg. Observar la resposta de twist amb:
  - ❏ Un client ftp localhost. Veurem el "tararí que te ví"
  - ❏ Un client telnet localhost 21. Veurem el cal.
  - ❏ Un client ncat localhost 21. veurem tot el cal.

Pendent TCP Wrappers / xinetd:

- Integrar un servei sense libwrap usant tcpd com a embolcall.
- Monitoritzar el tràfic de redireccio xinetd amb wireshark