NAME

serv.access − Internet service access list

SYNOPSIS

/etc/serv.access

DESCRIPTION

The **serv.access** file contains a list of rules that guide the access checks made by the **servxcheck**(3) function. The file is a text file containing entries that look as follows:

> *service1 service2 ...***:** *check1 check2 ...***;**

Each of the service names is a service name from the **/etc/services** file. The same names are used in the **/etc/inetd.conf** configuration file that guides **inetd**(8).

The checks may look as follows:

+
-

> Allow all, or allow none. Used to explicitly set the initial state.

+*name*

> Grant access to one of the services if the host name of the remote system matches **name**.

−*name*

> Deny access to one of the services if the host name of the remote system matches **name**.

+*ipaddr*
−*ipaddr*
+*netaddr***/***len*
−*netaddr***/***len*

> Grants or denies access to a remote host with IP address *ipaddr*, or the remote host whose IP address is within the network *netaddr*. *Len* tells the number of bits used for the network address, i.e. the top *len* bits of the network address must equal the host address.

log

> This is not a check, but a flag that instruct **servxcheck()** to log the result of the access check whether it succeeds or not to **/usr/adm/log**. By default only failure is logged.

The first "+" or "−" access check sets the tone. Read it as "access denied unless +...", or "access granted unless −...". An access check will therefore almost always start with a "+" check. To make the initial state clear you can start with a lone "+" or "−". Checks are done from left to right. A check that doesn't match does not change the outcome. A check that can't change the outcome is skipped.

Both the service and the host names may contain the ∗ wildcard that matches any number of characters including none. Letters are compared ignoring case. A service name may appear in more than one rule, but a service mentioned explicitly is not matched by wildcard patterns in later rules.

A check for a hostname causes **servxcheck()** to do a reverse lookup on the IP address of the remote host to find its name. This name is then looked up to find the host's IP address(es). If those lookups fail then all **−***name* checks cause access to be denied, and no **+***name* check grants access. The DNS lookup failures may be a misconfiguration, but could indicate a break-in attempt from a badly maintained host. You can use a simple "+*" in an otherwise empty list to just deny misconfigured hosts.

An IP or network address check is simply done on the remote hosts IP address. Such a check has no overhead, but a **log** flag will cause a reverse lookup anyway.

Comments start with "#" and continue until end of line.

EXAMPLES

Example access file on a machine that offers most services only to hosts within the cs.vu.nl domain, and news (nntp) only to two machines and a specific network.

```
# Service                      # Access list
login shell:                   +*.cs.vu.nl log;
telnet pop smtp finger:        + log;
```

```
nntp:                +flotsam.cs.vu.nl +jetsam.cs.vu.nl
                         +172.16.102.0/24 log;
*:                   +*.cs.vu.nl;
```

More paranoid example that limits all services by default, but allows ftp and http to the world:

```
# Service            # Access list
ftp http:            +;
smtp finger:         + log;
nntp:                +flotsam.cs.vu.nl +jetsam.cs.vu.nl
                         +172.16.102.0/24 log;
*:                   +*.cs.vu.nl log;
```

(Note that the last rule doesn't match any of the services mentioned explicitly earlier.)

FILES

/etc/serv.access The service access check file.

SEE ALSO

servxcheck(3), **services**(5), **inetd.conf**(5).

NOTES

It may be wise not to put checks on telnet. It is reasonably secure, since it always requires a password, and your only way in if things are seriously hosed.

BUGS

IP and DNS based access checks will stop most crackers, but not the really determined ones. Luckily Minix is sufficiently strange to thwart the well known cracking schemes. But don't ever allow yourself to feel secure.

AUTHOR

Kees J. Bot <kjb@cs.vu.nl>