9. Härten der Netzwerkschnittstelle

- **Routing**
- tcp-Wrapper
    - libwrap
    - tcpd
    - xinetd

9. Härten der Netzwerkschnittstelle
9.1 Routing

Was ist Routing?

Wie sind die Routen gesetzt?

Was passiert wenn Routen zu Hosts nicht gesetzt sind?

Wie werden Routen gesetzt?

Wie werden Routen gelöscht?

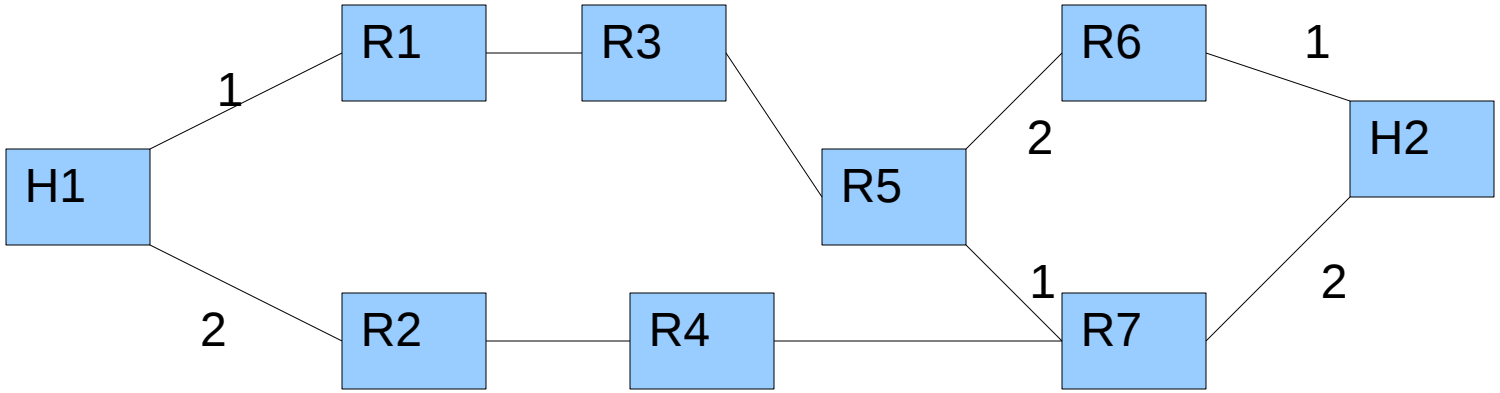
Was können wir mit diesem Wissen anfangen?

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.1 Routing

Was ist Routing?

Herausfinden des Weges eines IP-Paketes von einem Rechner zum anderen.



Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.1 Routing

Default Route:

Route, die benutzt werden soll, wenn für das Ziel keine andere Route bekannt ist. Im Allgemeinen der leistungsfähigste Router in der direkten Umgebung.

Host-Route/Net-Route für spezielle Hosts oder Netzwerke:

Route, die benutzt werden soll, wenn der spezielle Host oder ein Host aus einem speziellen Netzwerk als Ziel angegeben ist.

Was passiert, wenn für einen Ziel keine Route bestimmt werden kann? Wann kann das auftreten?

Was passiert, wenn zwei Routen gesetzt sind?

9. Härten der Netzwerkschnittstelle
9.1 Routing

Wie sind die Routen gesetzt?

Kommando:

netstat -r

Kernel IP Routentabelle

Ziel	Router	Genmask	Flags	MSS	Fenster	irtt	Iface
141.20.20.0	*	255.255.255.0	U	0	0	0	eth0
141.20.192.0	*	255.255.254.0	U	0	0	0	eth1
link-local	*	255.255.0.0	U	0	0	0	eth0
loopback	*	255.0.0.0	U	0	0	0	lo
default	141.20.20.1	0.0.0.0	UG	0	0	0	eth0

9. Härten der Netzwerkschnittstelle
9.1 Routing

Kommando

netstat -nr

Kernel IP Routentabelle

Ziel	Router	Genmask	Flags	MSS	Fenster	irtt	Iface
141.20.20.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
141.20.192.0	0.0.0.0	255.255.254.0	U	0	0	0	eth1
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	141.20.20.1	0.0.0.0	UG	0	0	0	eth0

9. Härten der Netzwerkschnittstelle
9.1 Routing

Was passiert wenn Route zu Zielhost nicht explizit gesetzt ist?

1. Ziel in einem direkt erreichbaren Netz:
   Pakete direkt zustellen

2. Ziel nicht in einem direkt erreichbaren Netz, Default-Route gesetzt:
   Paket an Default-Router zum weiterleiten senden.

3. Ziel nicht in einem direkt erreichbaren Netz, Default-Rout nicht gesetzt:
   ICMP Net Unreachable from gateway knecht2 (141.20.21.11)

9. Härten der Netzwerkschnittstelle
9.1 Routing

Was passiert wenn Route zu Zielhost explizit gesetzt wurde?

Paket werden direkt an den angegeben Router gesendet.
Wenn dieser nicht existiert, erfolgen keine weiteren Aktionen.
Fehlernachrichten werden nicht erzeugt.

mail > **ping 218.64.130.130**
no answer from 218.64.130.130
mail >

9. Härten der Netzwerkschnittstelle
9.1 Routing

Wie werden Routen gesetzt?

Kommando:

Solaris:

route add <destination> [**-netmask** <netmask>]
<gateway> [<metric>]

route add 147.138.20.0 **-netmask** 255.255.255.128 141.20.20.1

Linux:

route add [**-net**|**-host**] <destination> [**netmask** <netmask>
gw <gateway> [**metric** <metric>] [**dev** <device>]

route add -net 147.138.20.0 **netmask** 255.255.255.128 \
gw 141.20.20.1

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.1 Routing

Wie werden Routen gelöscht?

Kommando

Solaris:

route delete [<modifiers>] <destination> <gateway>

route delete 199.141.30.5

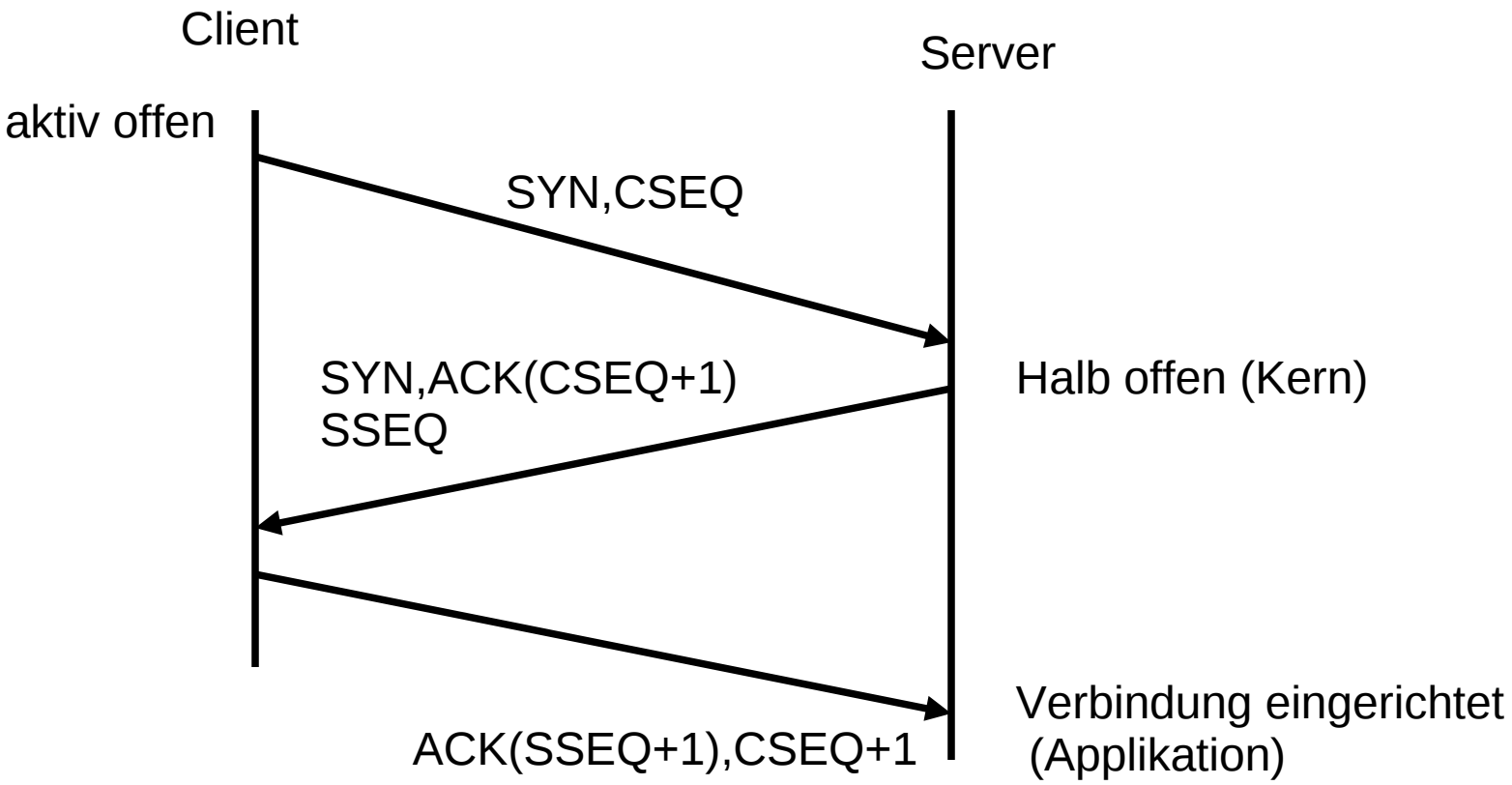
Linux:

route del [**-net**|**-host**] Ziel **gw** <gateway>

route del -host 199.141.30.5 **gw** 141.20.20.1

9. Härten der Netzwerkschnittstelle
9.1 Routing

Wie sieht ein Verbindungsaufbau aus (TCP)?



Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.1 Routing

Was passiert wenn Pakete nicht ankommen?

SYN,CSEQ kommt nicht an:
- Client: wartet, wiederholt, bricht ab
- Server: merkt nichts.

SYN,ACK kommt nicht an:
- Client: wartet, wiederholt, bricht ab
- Server: merkt nichts, halboffene Verbindung wird nach endlicher Zeit gelöscht

ACK kommt nicht an:
- Client: merkt nichts, will kommunizieren
- Server: merkt nichts, halboffene Verbindung wird nach endlicher Zeit gelöscht

9. Härten der Netzwerkschnittstelle
9.1 Routing

Was können wir mit diesem Wissen anfangen?

1. Client erfährt von der Existenz eines Servers erst durch ein SYN,ACK-Paket

2. Der Server reicht eine TCP-Verbindung erst nach einem ACK-Paket an die Applikation durch (accept)

3. Wenn wir das SYN,ACK-Paket an eine falsche oder nicht existente Adresse umleiten, erfährt der Client nichts von dem Server.

9. Härten der Netzwerkschnittstelle
9.1 Routing

Schlussfolgerung:

Wenn wir zielgerichtet Routen zu störenden Hosts wegnehmen, verhindern wir den Verbindungsaufbau für zu schützende Applikationen zu diesen Hosts. Server die nur lokale Dienste verrichten brauchen keine Default-Route. Für Server, die Dienste für die Allgemeinheit verrichten, können wir Routen explizit nicht streichen, da dort eine Default-Route notwendig ist. Also müssen wir eine Route explizit für den störenden Host setzen, die nirgendwohin zeigt.

z.B. **route add -host** 123.246.123.1 141.20.20.254

der Rechner 141.20.20.254 darf nicht existieren!!!

Problem: Wie bestimmen wir störende Hosts mit normalen Mitteln?

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.1 Routing

1. Beispiel:

Schützen eines Hosts, der nur lokale Aufgaben zu erfüllen hat.

z.B. NFS-Server
NIS-Server
DNS-Slave-Server

Wie?

Keine Default-Route setzen

Nur Routen zu den einzelnen Subnetzen setzen

Problem: Dafür gibt es meistens keine Konfigurations-files in den Systemen – also Startup-Scripte ändern.

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.1 Routing

2. Beispiel:

Mailserver gegen Spammer schützen

Merkmale von Spammern:

1. Sie senden Mails von Rechnern, deren IP-Adresse nicht in anständige Namen umgewandelt werden können.
2. Sie bauen mehrere SMTP-Verbindungen zu einem Host parallel auf.
3. Sie bauen mehrere SMTP-Verbindungen zu mehreren Hosts der Zieldomaine parallel auf.

9. Härten der Netzwerkschnittstelle
9.1 Routing

```
#!/bin/sh
# check-smtp
ROUTES=/etc/routes-disable
HOST=`hostname`
FND=`/bin/netstat -a | grep $HOST.smtp \
            | /bin/grep ESTABLISHED | /bin/sort | \
              /usr/local/bin/gawk 'BEGIN{ FOUND = "" ; } \
                                    $2 !~ /[a-zA-Z]/{ split($2,werte,"."); \
                                      SWERT = sprintf("%s",
                                   werte[1]"."werte[2]"."werte[3]"."werte[4]); \
                                   print SWERT; \
                                   if ( FOUND == SWERT ) print SWERT;  \
                                   FOUND = SWERT;    \
}' `
FOUND=""
```

9. Härten der Netzwerkschnittstelle
9.1 Routing

```
FOUND=""
CNT=0
for i in $FND
  do
     if  [ "$i" = "$FOUND" ] ; then
       continue
     fi
     LOCAL=`netstat -na | grep 141.20.20.51.25 | grep $i | grep ESTABLISHED |
                    wc -l`
     FOUND=$i
     FNDMAIL=`/usr/bin/ssh -l root mail "netstat -na | \
          grep 141.20.20.50.25 | grep $i | grep ESTABLISHED | wc -l"`
     if  [ $LOCAL -ge 2 ] ; then
       if [ $FNDMAIL -ge 2 ]  ; then
          HOST=`/usr/sbin/host $i`
          echo "Moeglicher Angriff von:    $i"
          echo "    Verbindungen auf mailslv1: $LOCAL"
          echo "    Verbindungen auf mail: $FNDMAIL"
```

9. Härten der Netzwerkschnittstelle
9.1 Routing

```
        echo "   Verbindungen auf mailslv1: $LOCAL"
        echo "   Verbindungen auf mail: $FNDMAIL"
        if [ "$HOST" -ne "Host not found." ] ; then
            echo "    DNS: $HOST"
        fi
        RTNS=`netstat -nr | grep $i | wc -l`
        if [ $RTNS -eq 0 ] ; then
            echo "   Route fuer Host: $i  sperren"
            /usr/sbin/route add host $i 141.20.20.254
            echo $i >> $ROUTES
            /usr/bin/ssh -l root mail "/usr/sbin/route add host $i 141.20.20.254"
            /usr/bin/ssh -l root mail "/bin/echo $i >> $ROUTES"
        else
            echo "gefundene Routen: $RTNS"
            echo "Route fuer Host: $i  schon gesperrt"
        fi
    fi
  fi
done
```

9. Härten der Netzwerkschnittstelle

- Routing
- **tcp-Wrapper**
  - libwrap
  - tcpd-Applikationen
  - Konfiguration
  - Einbindung in verschiedene Applikationen

9. Härten der Netzwerkschnittstelle
9.2. tcp-wrapper

Leistungen und Funktionsweise

Bestandteile

- Bibliothek: libwrap
- Programme: tcpd, tcpdchk, tcpdmatch, safe_finger, try-from

Konfiguration

- /etc/hosts.allow
- /etc/hosts.deny

Einbindung

- inetd, xinetd, sshd, portmap, rpcbind, sendmail, imap, pop, ldap

Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Der tcp-wrapper bietet folgende Leistungen:

- eingehende Netzwerkverbindungen mittels syslogd zu protokollieren
- eingehende Netzwerkverbindungen blockieren
  - Verbindung verwerfen
  - Verbindung manipulieren
- eingehende Netzwerkverbindungen an den eigentlichen Dienst weiterreichen
- gefälschte Absenderadressen erkennen (Name, IP-Adresse) – paranoide Absender
- IP-Spoofing erkennen
- unterstützt TCP und UDP Protokoll (außer rpc/tcp -Verbindungen und wait-Dienste)

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Einbindung:

- Der tcp-wrapper kann direkt in eine Applikation eingebunden werden (z.B. werden sshd, sendmail, xinetd, imap, pop, ... gegen die Bibliothek libwrap gelinkt)
- Der tcp-wrapper kann als eigenständiges Programm für jede Applikation benutzt werden, die mit dem inetd zusammenarbeiten kann.

Funktionsweise:

- tcp-wrapper bekommt eine eröffnete Verbindung übergeben
- tcp-wrapper legt request-Struktur an (mit Name der Applikation)
- tcp-wrapper bestimmt Absender(Adresse, Port, Nutzer)
- tcp-wrapper überprüft entsprechend den Regeln in den Konfigurationsfiles /etc/hosts.allow und /etc/hosts.deny die Zulässigkeit der Verbindung.

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Bibliothek: libwrap(1)

struct request_info *request_init(request, key, value, ..., 0);
struct reques_info *request;

Initialisieren der Struktur **request.** Diese enthält Informationen über die zu prüfende Verbindung. Die Initialisierungswerte werden mittels Parameterpaaren bestehend aus Schlüssel **key** und Werten **value** übergeben (variable Zahl).

Folgende Schlüssel sind zulässig:

RQ_FILE	- Filedescriptor der Verbindung
RQ_DAEMON	- Name des Server-Daemons

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Bibliothek: libwrap(2)

struct request_info *request_set(request, key, value,...., 0);
struct reques_info *request;

Modifiziert die Struktur **request** mit den Informationen über den Clienten. Die zu modifizierenden Werte werden mittels Parameterpaaren bestehend aus Schlüssel **key** und Werten **value** übergeben (variable Zahl).

Folgende Schlüssel sind zulässig:

RQ_CLIENT_NAME	- Name des Clienten
RQ_CLIENT_ADDR	- IP-Adresse des Clienten
RQ_CLIENT_SIN	- Adresse und Port des Clienten (*sockaddr_in)
RQ_USER	- Nutzername des Prozesses auf dem Clienten

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Bibliothek: libwrap(3)

int hosts_access(request);

struct request_info *request;

hosts_access überprüft an Hand der Konfigurationsfiles /etc/hosts.allow und /etc/hosts.deny die Zulässigkeit des Request. Die in den Konfigurationsfiles angegebenen Aktionen werden ausgeführt. Wenn der Request nicht zulässig ist, ist der Rückkehrwert **0.**

Die Variablen **allow_severity** bzw. **deny_severity** bestimmen das Log-Verhalten für erlaubte bzw. verbotene Verbindungen.

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Bibliothek: libwrap(4)

int hosts_ctl(daemon, client_name, cleint_addr, client_user);
char *daemon;
char *client_name;
char *client_addr;
char *client_user;

hosts_clt ist ein Wrapper für **request_init, request_set** und **hosts_access**, der einen einfacheren Zugang zum tcp-wrapper ermöglichen soll. Nicht belegbare Parameter sind mit **STRING_UNKNOWN** zu belegen.

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Bibliothek: libwrap(5)

Beispiel tcpd

```
#include <sys/types.h>
...
#include „tcpd.h“
int allow_severity = SERVERITY;
int deny_severity = LOG_WARNING;
main(int argc, char **argv);
{
    struct request_info request;
    char path[1024];
```

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

```
umask(DAEMON_UMASK);
if (argv[0][0] == '/') {
    strcpy(path, argv[0]); arv[0] = strrchr(argv[0],'/') + 1;
} else
      sprintf(path,“%s/%s“,REAL_DAEMON_DIR, argv[0]);
(void) openlog(argv[0], LOG_PID);
request_init(&request, RQ_DAEMON,argv[0], RQ_FILE,
                              STDIN_FILENO,0);
fromhost(&request);  /* eintragen des Absenders in request-Struktur */
if (!hosts_access(&request))
     refuse(&request);
syslog(allow_severity, „connect from %s“, eval_client(&request));
closelog();
void) execv(path,argv);
syslog(LOG_ERR, „error: cannot execute %s: %m“, path);
clean_exit(&request);
}
```

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Applikationen(1)

tcpd

tcp-Wrapper. Dieses Programm wird für den Aufruf des eigentlichen Servers benutzt (/.../libexec/tcpd). Die Serverprogramme müssen in einer fest vorgegeben Directory stehen. Der typische Anwendungsfall ist der Aufruf eines Servers mittels inetd. Die entsprechenden Zeilen in der inetd.conf-Datei sehen wie folgt aus:

```
ftp     stream  tcp  nowait  root  /usr/sbin/tcpd   in.ftp
telnet stream  tcp  nowait  root  /usr/sbin/tcpd   in.telnetd
finger stream  tcp  nowait  nobody  /usr/sbin/tcpd  in.fingerd
```

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Applikationen(2)

tcpdchk [-a] [-d] [-i *inet_conf*] [-v]

Überprüfen der tcp-wrapper Konfiguration auf mögliche Fehler
/.../libexec/tcpdchk).

-a Ausgabe von Regeln, die ohne explizitem ALLOW den Zugriff zulassen

-d benutze host.allow und hosts.deny im aktuellen Directory

-i *inet_conf* benutze *inet_conf* anstelle von */etc/inetd.conf*

-v Alle Regeln einzeln ausgeben

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Applikationen(3)

tcpdmatch [-d] [-i *inet_conf***]** *daemon***[***@server***]**
[*user@***]***client*

Überprüfen des tcp-wrappers auf bestimmte Anfragen (/.../libexec/tcpdmatch).

*daemon***[***@server***]** Dienst und Host für den der tcp-Wrapper läuft

[*user@***]***client* Nutzer und Host der des Clienten, der den Dienst benutzen will.

-d benutze host.allow und hosts.deny im aktuellen Directory

-i *inet_conf* benutze *inet_conf* anstelle von */etc/inetd.conf*

Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Applikationen(4)

safe_finger [-hlMmops] [*user*]

safe_finger [-l] [*user@host*]

Sicheres, rekursionsfreies finger-Programm. Dient zum Abfragen des finger-Daemons (/.../libexec/safe_finger).

-l langes Format

-s kurzes Format

-M alle Nutzer anzeigen

-o Office Informationen hinzufügen

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Konfiguration(1)

/etc/hosts.allow – Regeln für zulässige Verbindungen
/etc/hosts.deny – Regeln für unzulässige Verbindungen

einfaches Regelwerk:

<Liste von Diensten> **:** <Liste von Clienten> [**:** <Aktion>]

<Liste von Diensten>::=<Dienstname>**|ALL** { <Dienstname>}
<Liste von Clienten>::=<Client> { <Client> }
<Client>::=<Hostname>|<IP-Adresse>|<Netzwerk>**/**<Netzwerkmaske>|
LOCAL | **UNKNOWN** | **KNOWN** | **PARANOID**
<Aktion>::=**allow** | **deny** | <Shell-Kommando>

Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Konfiguration(2)

Platzhalter in Shell-Kommandos

%A - IP-Adresse des Servers
%a - IP-Adresse des Clienten
%c - Informationen vom Clienten: nutzer@client
%d - Dienstname (z.B. in.ftpd)
%H - Name des Servers, sonst IP-Adresse
%N - Name des Servers, sonst 'unknown'
%h - Name des Clienten, sonst IP-Adresse
%n - Name des Clienten, sonst 'unknown'
%p - Prozessnummer des Daemon

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Konfiguration(3)

Beispiel (einfaches Regelwerk)**:**

/etc/hosts.allow (Regeln für zulässige Verbindungen)

```
rpcbind: 141.20.20.0/255.255.255.255.0 141.20.20.21.0/255.255.255.0
sshd: 141.20.20.0/255.255.255.0 141.20.28.18 141.20.8.135
ALL: bellus3
in.telnetd in.ftpd : 141.20.20.20 141.20.20.22 : ALLOW
```

/etc/hosts.deny (Regeln für unzulässige Verbindungen)

```
rpcbind : ALL : ( /usr/sbin/safe_finger -l @%h | /usr/ucb/mail
        -s "`/bin/hostname`-%d-%h-%a" bell@informatik.hu-berlin.de ) &
in.telnetd : ALL : DENY
ALL: ALL:  (/usr/sbin/safe_finger -l @%h | /usr/ucb/mail
        -s "`/bin/hostname`-%d-%h-%a" bell@informatik.hu-berlin.e) &
```

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Konfiguration(4)

/etc/hosts.allow – Regeln für zulässige Verbindungen
/etc/hosts.deny – Regeln für unzulässige Verbindungen

Erweitertes Regelwerk:
Wie einfaches Regelwerk mit zusätzlichen Möglichkeiten für Aktionen.
Erweitertes Regelwerk kommt ab Solaris 10 zum Einsatz!!!
<Aktion>::= <einfache Aktion> { **:** <einfache Aktion> }
<einfache Aktion>::= **allow** | **deny** | **banners** <Path> | **keepalive** |
linger <sekunden> | **severity** <dienstname>**.info** | **nice** <Wert> |
rfc931 | **setenv** <Name> <Wert> | **spawn** <Shell-Kommando> |
twist <Shell-Kommando> | **umask** <ooo> | **user** <UID> <GID>

Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Konfiguration(5)

banners <Path> - Bannerfile (Path/<Dienstname>) wird an Client übergeben

keepalive - keepalive für Dienst aktivieren

linger <sekunden> - halten der Verbindung nach Beendigung des Dienstes

severity <dienstname>.info - in syslog schreiben

nice <Wert> - nice-Wert für Dienst setzen

rfc931 - rfc931 zur Clientenbestimmung benutzen

setenv <Name> <Wert> - Umgebungsvariable setzen

spawn <Shell-Kommando> - Kommando abarbeiten

twist <Shell-Kommando> - Shell-Kommando abarbeiten(Dienst nicht)

umask <ooo> - Filecration-Maske setzen

user <UID> <GID> - User-ID und Group-ID setzen

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Konfiguration (6)

Beispiele (erweitertes Regelwerk):

/etc/hosts.allow

```
sshd : ALL : ALLOW
in.telnetd : 141.20.21.0/255.255.255.0 : twist /bin/echo abgeschaltet
rpcbind : LOCAL : ALLOW
```

/etc/hosts.deny

```
rpcbind: ALL
in.telnetd: ALL : spawn (/usr/sbin/safe_finger -l @%h |
                         /usr/bin/mailx -s telnet-%d-%h-%a bell) : DENY
```

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Einbindung (1)

Alle Applikationen benutzen /etc/hosts.allow und /etc/hosts.deny !!

1. **inetd** (Solaris <2.10, BSD-Systeme)

   /etc/inetd.conf

   telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd

2. **inetd** (Solaris 2.10)

   inetadm -M tcp_wrappers=TRUE

   svccfg

   select svc:/network/rpc/bind

   setprop config/enable_tcpwrappers=true

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Einbindung (2)

3. **xinetd** (Linux)

/etc/xinet.d/<applikation>

/etc/xinet.d/telnet

```
service telnet
    {   socket_type   = stream
        protocol      = tcp
        wait          = no
        user          = root
        server        = /usr/sbin/in.telnetd
        disable       = no
    }
```

/etc/hosts.allow: in.telnetd : LOCAL : allow

Systems Architecture Group
http://sar.informatik.hu-berlin.de

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Einbindung (3)

4. **portmap** (BSD-Systeme)

portmapper der gegen libwrap gelinkt ist.

/etc/hosts.allow:

portmap : 141.20.20.0/255.255.255.0

/etc/hosts.deny

portmap : ALL : deny

Installieren:

pmap_dump > *table*

kill <old portmapper>

/usr/sbin/portmap # neuen Portmapper starten

pmap_set < *table*

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Einbindung (4)

5. **rpcbind** (System V, Solaris)

Portmapper der gegen libwrap gelinkt ist.

/etc/hosts.allow

rpcbind: 141.20.20.0/255.255.255.0 : ALLOW

/etc/hosts.deny

rpcbind: ALL : spawn (/usr/sbin/safe_finger -l @%h |
/usr/bin/mailx -s %d-%h-%H-%a bell) : DENY

installieren:

kill -TERM <alter rpcbind>

es entsteht /tmp/rpcbind.file und alter rpcbind wird beendet

rpcbind -w # Warmstart des neuen rpcbind und /tmp/rpcbind.file

9. Härten der Netzwerkschnittstelle
9.2 tcp-wrapper

Beispiele