

KAPITEL 3

Zugriffskontrolle im Netz

3.0 Einführung

Eine der wichtigsten Sicherheitsaufgaben besteht darin, die eingehenden Netzwerkverbindungen zu überwachen. Als Systemadministrator können Sie diese Verbindungen auf vielen Ebenen kontrollieren. Auf der untersten Ebene – der Hardware – können Sie die Netzwerkkabel ziehen, aber das ist nur selten notwendig, es sei denn, Ihr Computer wurde allen Hürden zum Trotz geknackt. Praktischer sind da schon die Ihnen zur Verfügung stehenden Möglichkeiten der Software-Kontrolle, die nachfolgend vom Allgemeinen hin zum Dienste-spezifischen sortiert sind:

Netzwerkschnittstelle
: Die Schnittstelle kann komplett herauf- und heruntergefahren werden.

Firewall
: Durch das Festlegen von Firewall-Regeln im Linux-Kernel können Sie die Verarbeitung eingehender (sowie ausgehender und weitergeleiteter) Pakete kontrollieren. Dieses Thema wird in Kapitel 2 behandelt.

Ein Superdaemon oder Internet-Services-Daemon
: Ein Superdaemon kontrolliert den Start (oder Nichtstart) bestimmter Netzwerkdienste anhand verschiedener Kriterien. Nehmen wir an, Ihr System empfängt eine eingehende Anfrage für eine Telnet-Verbindung. Ihr Superdaemon kann diesen auf Grund der Quelladresse, der Tageszeit, der Anzahl gerade aktiver Telnet-Verbindungen etc. akzeptieren oder ablehnen. Oder er könnte einfach den gesamten Telnet-Zugriff verbieten. Superdaemons besitzen üblicherweise eine Reihe von Konfigurationsdateien, damit Sie Ihre vielen Dienste bequem von einem Ort aus kontrollieren können.

Individuelle Netzwerkdienste
: Jeder Netzwerkdienst wie `sshd` oder `ftpd` kann eigene, fest integrierte Einrichtungen zur Zugriffskontrolle besitzen. So besitzt `sshd` beispielsweise das Konfigurationsschlüsselwort `AllowUsers`; `ftpd` verwendet */etc/ftpaccess*, und verschiedene andere Dienste verlangen eine Benutzer-Authentifizierung.

All diese Ebenen spielen eine Rolle, wenn eine Anfrage für einen Netzwerkdienst eintrifft. Nehmen wir an, der entfernte Benutzer joeblow versucht, sich mittels FTP in den Account von smith auf *server.example.com* einzulogen (siehe Abbildung 3-1):

Wenn *server.example.com* physikalisch mit dem Netzwerk verbunden ist...
und seine Netzwerkschnittstelle aktiv ist...
und seine Kernel-Firewall FTP-Pakete von Joes Host erlaubt...
und ein Superdaemon läuft...
und der Superdaemon so konfiguriert ist, dass er `ftpd` aufruft...
und der Superdaemon FTP-Verbindungen von Joes Maschine akzeptiert...
und `ftpd` installiert und ausführbar ist...
und die `ftpd`-Konfiguration in */etc/ftpaccess* die Verbindung akzeptiert...
und joeblow sich als smith authentifiziert...

dann wird die Verbindung hergestellt. (Vorausgesetzt, dass nichts anderes sie verhindert, etwa ein Ausfall des Netzwerks.)

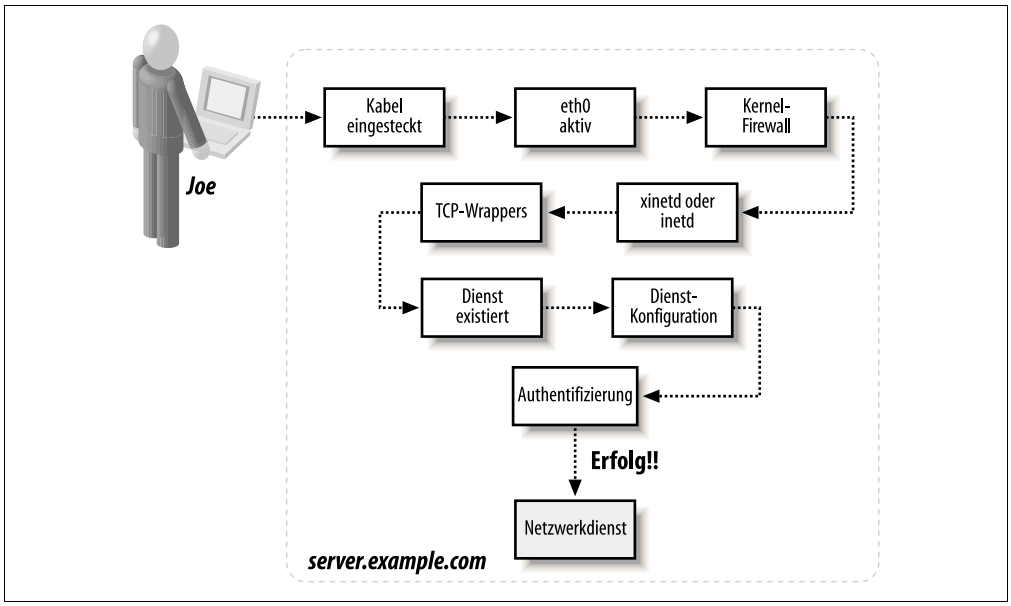


Abbildung 3-1: Sicherheitsebenen eingehender Netzwerkverbindungen

Systemadministratoren müssen an all diese Kontrollebenen denken. In diesem Kapitel erläutern wir:

ifconfig
: Ein auf niedriger Ebene angesiedeltes Programm zur Kontrolle der Netzwerkschnittstellen. Es ermöglicht, sie herauf- und herunterzufahren und Parameter einzustellen.

xinetd

Ein den Aufruf anderer Deamons kontrollierender Superdaemon. Er wird über Konfigurationsdateien gesteuert, die üblicherweise, jeweils eine Datei pro Service, im Verzeichnis */etc/xinetd.d* abgelegt sind. So legt z.B. */etc/xinetd.d/finger* fest, wie der finger-Daemon bei Bedarf aufgerufen werden soll:

/etc/xinetd.d/finger:

```
service finger
{
    server = /usr/sbin/in.fingerd       Pfad auf Executable
    user = nobody                       Als Benutzer "nobody" ausführen
    wait = no                           Multithreading aktiv
    socket_type = stream                Stream-basierter Service
}
```

Red Hat verwendet `xinetd`.

inetd

Ein Superdaemon wie `xinetd`, aber älter. Die Konfiguration erfolgt über */etc/inetd.conf* mit jeweils einem Dienst je Zeile. Ein zu dem vorangegangenen `xinetd`-Beispiel analoger Eintrag sieht wie folgt aus:

/etc/inetd.conf:

```
finger  stream  tcp  nowait  nobody  /usr/sbin/in.fingerd  in.fingerd
```

SuSE verwendet `inetd`.

TCP-Wrapper

Eine Schicht, die eingehende Zugriffe auf bestimmte Hosts oder Domains sowie andere Aspekte kontrolliert. Das Verhalten wird über */etc/hosts.allow* (erlaubte Verbindungen) und */etc/hosts.deny* (nicht-erlaubte Verbindungen) gesteuert. Um beispielsweise alle `finger`-Verbindungen zu unterbinden, geben Sie Folgendes ein:

/etc/hosts.deny:

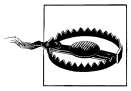
```
finger : ALL : DENY
```

Sollen `finger`-Verbindungen hingegen nur von Hosts der Domain *friendly.org* erlaubt sein, tragen Sie Folgendes ein:

/etc/hosts.allow:

```
finger : *.friendly.org
finger : ALL : DENY
```

Wir geben hier nicht die gesamte von diesen Dateien unterstützte Syntax wieder, weil diese in der Manpage hosts.allow(5) zu finden ist. Denken Sie aber daran, dass TCP-Wrapper auch IDENT-Prüfungen und externe Programme aufrufen sowie andere wichtige Dinge tun können. Sowohl Red Hat als auch SuSE besitzen TCP-Wrapper.



Für alle Rezepte in diesem Kapitel gilt ein großer Vorbehalt: Der Zugriff wird nicht auf einen Host, sondern auf eine IP-Quelladresse beschränkt. So können wir zum Beispiel festlegen, dass nur der Host 121.108.19.42 einen bestimmten Dienst unseres Systems nutzen kann. Die Quelladresse kann

aber ohne große Schwierigkeiten gefälscht werden. Eine Maschine, die vorgibt, der Host 121.108.19.42 zu sein, kann solche Beschränkungen möglicherweise unterlaufen. Wenn Sie den Zugriff wirklich über den Host und nicht bloß über die Quelladresse wünschen, dann sollten Sie eine kryptographische Host-Authentifizierung wie etwa die SSH-Server-Authentifizierung, die hostbasierte Client-Authentifizierung oder IPSec bevorzugen.

3.1 Ausgabe Ihrer Netzwerkschnittstellen

Problem

Sie möchten sich Ihre Netzwerkschnittstellen ausgeben lassen.

Lösung

Um sich alle Schnittstellen, egal ob aktiv oder inaktiv, ausgeben zu lassen, deren Treiber geladen sind, geben Sie Folgendes ein:

```
$ ifconfig -a
```

Um sich alle aktiven Schnittstellen ausgeben zu lassen:

```
$ ifconfig
```

Um sich eine einzelne Schnittstelle anzusehen (üblicherweise *eth0*):

```
$ ifconfig eth0
```

Erläuterung

Wenn Sie nicht als root arbeiten, liegt `ifconfig` möglicherweise nicht in Ihrem Pfad. Versuchen Sie es mit */sbin/ifconfig*.

Erfolgt der Aufruf mit der Option `-a`, gibt `ifconfig` alle Netzwerkschnittstellen (egal ob aktiv oder inaktiv) aus. Physikalisch vorhandene Schnittstellen, deren Treiber nicht geladen sind, werden dabei allerdings nicht berücksichtigt. Nehmen wir zum Beispiel an, Sie arbeiten mit einem Rechner, in dem zwei Ethernet-Karten (*eth0* und *eth1*) verschiedener Hersteller mit unterschiedlichen Treibern eingebaut sind, von denen aber nur die eine (*eth0*) für Linux konfiguriert ist (d.h. es gibt eine entsprechende */etc/sysconfig/network-scripts/ifcfg-**-Datei). Die andere Schnittstelle wird normalerweise nicht verwendet. `ifconfig -a` zeigt die zweite Schnittstelle nicht an, solange Sie den entsprechenden Treiber nicht mit `ifconfig eth1` laden.

Siehe auch

ifconfig(8).

3.2 Die Netzwerkschnittstelle aktivieren und deaktivieren

Problem

Sie möchten alle Netzwerkverbindungen, eingehende und ausgehende, auf Ihren Netzwerkschnittstellen unterbinden.

Lösung

Um eine Netzwerkschnittstelle, z.B. *eth0*, herunterzufahren, geben Sie Folgendes ein:

```
# ifconfig eth0 down
```

Um eine Netzwerkschnittstelle, z.B. *eth0*, wieder zu aktivieren:

```
# ifconfig eth0 up
```

Um alle Netzwerkschnittstellen zu deaktivieren:

```
# /etc/init.d/network stop
```

oder:

```
# service network stop              Red Hat
```

Um das gesamte Netzwerk wieder zu starten:

```
# /etc/init.d/network start
```

oder:

```
# service network start             Red Hat
```

Erläuterung

Linux stellt drei Abstraktionsebenen zu Aktivierung bzw. Deaktivierung Ihrer Netzwerkschnittstellen zur Verfügung:

/sbin/ifconfig
: Das ist die unterste Ebene, auf der eine einzelne Netzwerkschnittstelle aktiviert/deaktiviert werden kann. Darüber hinaus besitzt sie weitere Funktionen zur Konfiguration einer Schnittstelle.

/sbin/ifup, /sbin/ifdown
: Dieses Paar Skripten ist auf mittlerer Ebene angesiedelt und arbeitet mit einer einzelnen Netzwerkschnittstelle. Sie aktivieren bzw. deaktivieren die Schnittstelle über einen Aufruf von `ifconfig` mit den entsprechenden Argumenten. Sie initialisieren auch DHCP und behandeln noch einige andere Details. Diese werden vom Benutzer nur selten direkt aufgerufen.

/etc/init.d/network
: Dies ist ein auf hoher Ebene angesiedeltes Skript, das mit allen Netzwerkschnittstellen arbeitet, nicht nur mit einer. Es ruft `ifup` bzw. `ifdown` für jede Schnittstelle auf und behandelt auch andere Details: Es fügt Routen ein, erzeugt eine Lock-Datei, die anzeigt, dass das Netz aktiv ist, und vieles mehr. Es ändert sogar den Zustand der Loopback-Schnittstelle, was des Guten zu viel sein könnte, wenn Sie nur den ausgehenden Traffic sperren wollten.

Die Skripten `ifup`, `ifdown` und `network` sind recht kurz und durchaus einen genaueren Blick wert.

Siehe auch

ifconfig(8). usernetctl(8) beschreibt, wie Benutzer ohne root-Rechte die Parameter der Netzwerkschnittstellen mittels `ifup` und `ifdown` verändern können, wenn dies vom Systemadministrator erlaubt wurde.

3.3 Aktivieren/Deaktivieren eines Dienstes (xinetd)

Problem

Sie möchten verhindern, dass ein bestimmter TCP-Dienst auf Ihrem System durch `xinetd` aufgerufen wird.

Lösung

Wenn der Name des Dienstes »myservice« lautet, ermitteln Sie die Lage seiner Konfigurationsdatei in */etc/xinetd.d/myservice* oder */etc/xinetd.conf* und fügen

```
disable = yes
```

zu dessen Parametern hinzu. Um zum Beispiel `telnet` zu deaktivieren, editieren Sie */etc/xinetd.d/telnet*:

```
service telnet
{
    ...
    disable = yes
}
```

Dann informieren Sie `xinetd` über die Änderung, indem Sie ein Signal an den Prozess senden:

```
# kill -USR2 `pidof xinetd`
```

Um den Zugriff wieder zu ermöglichen, entfernen Sie die `disable`-Zeile und senden erneut das `SIGUSR2`-Signal.

Erläuterung

Statt einen Dienst zu deaktivieren, könnten Sie auch dessen `xinetd`-Konfigurationsdatei (z.B. */etc/xinetd.d/telnet*) löschen oder sogar das entsprechende Executable entfernen, aber solche Löschungen sind schwerer rückgängig zu machen. (Entfernen Sie aber nicht das Executable, *ohne* den Dienst zu deaktivieren, sonst versucht `xinetd` ihn auszuführen und beschwert sich entsprechend.)

Alternativ können Sie `ipchains` oder `iptables` [2.7] verwenden, um den Dienst verfügbar zu halten, aber nur auf bestimmte Netzwerk-Quelladressen zu beschränken. Darüber hinaus könnten bestimmte Dienste eigene Möglichkeiten zur Verfügung stellen, um den Zugriff auf bestimmte Clients zu beschränken.

Siehe auch

xinetd(8). Die `xinetd`-Homepage lautet *http://www.synack.net/xinetd*.

3.4 Aktivieren/Deaktivieren eines Dienstes (inetd)

Problem

Sie möchten verhindern, dass ein bestimmter TCP-Dienst auf Ihrem System durch `inetd` ausgeführt wird.

Lösung

Zum Deaktivieren kommentieren Sie die Zeile des Dienstes in */etc/inetd.conf* aus, indem Sie ihr ein Doppelkreuz (#) voranstellen. Um also beispielsweise den Telnet-Daemon zu deaktivieren, setzen Sie das Doppelkreuz vor folgende Zeile:

/etc/inetd.conf:
```
# telnet  stream  tcp  nowait  root  /usr/sbin/in.telnetd  in.telnetd
```

Dann informieren Sie `inetd` über die Änderungen, indem Sie ihm ein Signal senden. (Hier ist das Doppelkreuz das Prompt der Root-Shell, und kein Kommentarzeichen.)

```
# kill -HUP `pidof inetd`
```

Um ihn wieder zu aktivieren, entfernen Sie den Kommentar aus der Zeile und senden erneut das `SIGHUP`-Signal.

Erläuterung

Statt einen Dienst zu deaktivieren, können Sie auch die entsprechende Zeile in der `inetd`-Konfigurationsdatei löschen oder sogar das entsprechende Executable entfernen, aber solche Löschungen sind schwerer rückgängig zu machen. (Löschen Sie nicht das Exe-

cutable, *ohne* den Dienst zu deaktivieren, sonst versucht `inetd`, ihn auszuführen und beschwert sich entsprechend.) Alternativ können Sie `ipchains` oder `iptables` [2.6] verwenden, um den Dienst verfügbar zu halten, aber auf bestimmte Adressen zu beschränken.

Siehe auch

inetd(8), inetd.conf(5).

3.5 Neue Dienste einbinden (xinetd)

Problem

Sie möchten einen neuen Netzwerkdienst einbinden, der durch `xinetd` kontrolliert wird.

Lösung

Legen Sie ein neues */etc/xinetd.d*-Verzeichnis an, das zumindest die folgenden Informationen enthält:

```
service SERVICE_NAME                Name aus /etc/services; siehe services(5)
{
     server = /PFAD/AUF/SERVER      Das Executable für den Dienst
     server_args = ARGUMENTE        Alle Argumente oder weglassen
     user = BENUTZER                Dienst wird unter diesem Benutzer ausgeführt
     socket_type = TYP              stream, dgram, raw oder seqpacket
     wait = YES/NO                  yes = einzelner Thread, no = Multithreading
}
```

Nennen Sie die Datei nach dem *SERVICE_NAME*n. Senden Sie danach `xinetd` ein Signal, damit es seine Servicedatei neu einliest [3.3].

Erläuterung

Um eine `xinetd`-Konfigurationsdatei für Ihren Dienst anlegen zu können, müssen Sie natürlich die gewünschten Eigenschaften und das gewünschte Verhalten Ihres Dienstes kennen. Ist er streambasiert? Datagrammbasiert? Läuft ein Thread oder laufen mehrere? Welche Argumente verlangt der Server, wenn überhaupt?

`xinetd`-Konfigurationsdateien verfügen über eine große Zahl zusätzlicher Schlüsselwörter und Werte. Alle Details finden Sie in xinetd.conf(5).

`xinetd` liest die Dateien in */etc/xinetd.d* nur dann ein, wenn */etc/xinetd.conf* es mit der folgenden Zeile auffordert, dies zu tun:

```
includedir /etc/xinetd.d
```

Prüfen Sie in Ihrer */etc/xinetd.conf*-Datei nach, ob der Speicherort für `includedir` auch stimmt.

Siehe auch

xinetd(8), xinetd.conf(5), services(5). Die `xinetd`-Homepage finden Sie unter *http://www.synack.net/xinetd*.

3.6 Neue Dienste einbinden (inetd)

Problem

Sie möchten neue Netzwerkdienste einbinden, die durch `inetd` kontrolliert werden.

Lösung

Nehmen Sie eine Zeile der folgenden Form in */etc/inetd.conf* auf:

```
SERVICE_NAME  SOCKET_TYP  PROTOKOLL  THREADING  BENUTZER  /PFAD/AUF/SERVER  ARGUMENTE
```

Senden Sie `inetd` dann ein Signal, damit die */etc/inetd.conf*-Datei neu eingelesen wird [3.4].

Erläuterung

Die Werte der Zeile haben die folgende Bedeutung:

1. *Service-Name*. Ein in */etc/services* aufgeführter Dienst. Ist er dort nicht aufgeführt, müssen Sie einen neuen Eintrag einfügen, der den Namen des Dienstes, die Portnummer und das Protokoll angibt. Siehe services(5).
2. *Socket-Typ*. `stream`, `dgram`, `raw`, `rdm` oder `seqpacket`.
3. *Protokoll*. Üblicherweise `tcp` oder `udp`.
4. *Threading*. Verwenden Sie `wait` für einen oder `nowait` für mehrere Threads.
5. *Benutzer*. Der Dienst wird unter diesem Benutzer ausgeführt.
6. *Pfad* auf das Server-Executable.
7. *Server-Argumente*, durch Whitespace getrennt. Sie müssen mit dem Argument Nr. 0 beginnen, d.h. mit dem Basisnamen des Servers. Bei */usr/sbin/in.telnetd* wäre dieses Argument beispielsweise *in.telnetd*.

Ein vollständiges Beispiel lautet:

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

Eine *inetd.conf*-Zeile kann auch noch einige weitere Details enthalten, etwa die Puffergrößen, eine lokale Host-Adresse für das Listening und so weiter. Die Details finden Sie in der Manpage.

Siehe auch

inetd(8), inetd.conf(5), services(5).

3.7 Den Zugriff auf bestimmte entfernte Benutzer beschränken

Problem

Sie möchten dafür sorgen, dass nur bestimmte entfernte Benutzer Zugriff auf einen TCP-Dienst haben. Sie können nicht vorhersagen, von welchen Hosts aus diese Benutzer zugreifen.

Lösung

Sperren Sie den eingehenden TCP-Port des Dienstes mit einer Firewall-Regel [2.6]. Führen Sie einen SSH-Server aus, und ermöglichen Sie den Benutzern ein Tunneln über das SSH-Port-Forwarding. Auf diese Weise sorgt die SSH-Authentifizierung für den Zugang (oder eben auch nicht) zu diesem Dienst. Ermöglichen Sie den entfernten Benutzern den SSH-Zugriff über einen öffentlichen Schlüssel.

Um zum Beispiel den News-Server (TCP-Port 119) auf Ihrer Site *server.example.com* zu erreichen, könnte ein entfernter Benutzer auf dem Host *myclient* mittels SSH den folgenden Tunnel vom (willkürlich gewählten) lokalen Port 23456 auf den News-Server aufbauen:

```
myclient$ ssh -f -N -L 23456:server.example.com:119 server.example.com
```

Dann könnte er die Verbindung zu diesem Tunnel über den Newsreader `tin` herstellen:

```
myclient$ export NNTPSERVER=localhost
myclient$ tin -r -p 23456
```

Erläuterung

SSH-Tunneling, oder Port-Forwarding, leitet eine TCP-Verbindung so um, dass der Datenfluss fast transparent über einen SSH-Client und -Server läuft[1] [6.14]. Dieser Tun-

1 Bei Diensten, die bezüglich ihrer Socket-Details etwas empfindlich reagieren (etwa FTP), ist es mit der Transparenz vorbei, aber in den meisten Fällen ist die Kommunikation problemlos möglich.

nel verbindet einen lokalen mit einem entfernten Port, verschlüsselt die auf der einen Seite ausgehenden Daten und entschlüsselt sie wieder, sobald diese auf der Gegenseite eintreffen. Um zum Beispiel NNTP (Usenet-News, Port 119) zu tunneln, kommuniziert der Newsreader mit einem SSH-Client, der die Daten durch den Tunnel an den SSH-Server weiterleitet, der wiederum mit dem NNTP-Server kommuniziert. Abbildung 3-2 macht das deutlich.

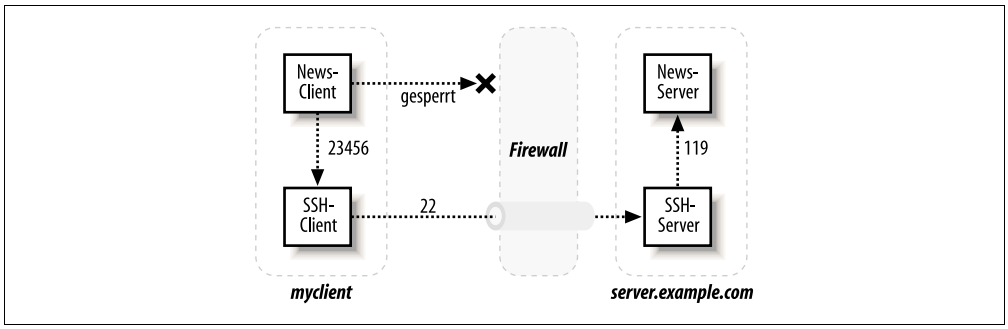


Abbildung 3-2: NNTP über SSH tunneln

Indem Sie den Port (119) des Dienstes für die Außenwelt blockieren, verhindern Sie den ganzen entfernten Zugriff auf diesen Port. Weil SSH aber einen anderen Port verwendet (22), wird es durch die Firewall nicht behindert.

Alternativ können Sie herausfinden, ob der zu verwendende Dienst eine eigene Benutzer-Authentifizierung besitzt. So besitzt beispielsweise `wu-ftpd` die Datei */etc/ftpaccess*, `sshd` kennt das Schlüsselwort `AllowUsers` und so weiter.

Siehe auch

ssh(1), sshd(8), tin(1).

3.8 Den Zugriff auf bestimmte entfernte Hosts beschränken (xinetd)

Problem

Sie möchten mittels `xinetd` den Zugriff auf einen TCP-Dienst nur auf bestimmte entfernte Hosts beschränken.

Lösung

Verwenden Sie die *xinetd.conf*-Schlüsselwörter `only_from` und `no_access`:

```
service ftp
{
    only_from = 192.168.1.107
    ...
}

service smtp
{
    no_access = haxor.evil.org
    ...
}
```

Dann setzen Sie `xinetd` zurück, damit die Änderungen wirksam werden [3.3].

Erläuterung

Dies ist die vielleicht einfachste Möglichkeit, den Zugriff auf einen Dienst zu kontrollieren. Allerdings funktioniert er nur für Dienste, die über `xinetd` gestartet werden.

`only_from` und `no_access` können in einem Service-Eintrag mehrfach vorkommen:

```
{
    no_access = haxor.evil.org  einen bestimmten Host ablehnen
    no_access += 128.220.       alle Hosts eines Netzwerks ablehnen
    ...
}
```

Wird ein die Verbindung herstellender Host sowohl in der `only_from`- als auch in der `no_access`-Liste gefunden, geht `xinetd` wie folgt vor:

- Ist ein Host in beiden Listen enthalten und ist ein Eintrag *spezifischer* als der andere, dann setzt sich der spezifischere Eintrag durch. So ist zum Beispiel 128.220.13.6 spezifischer als 128.220.13.
- Sind die Einträge für einen Host in beiden Listen *gleich spezifisch*, betrachtet `xinetd` dies als Konfigurationsfehler, und der angeforderte Dienst wird nicht gestartet.

In dem Beispiel

```
service whatever
{
    no_access = 128.220.     haxor.evil.org    client.example.com
    only_from = 128.220.10.  .evil.org         client.example.com
}
```

sind Verbindungen von 128.220.10.3 also erlaubt, während diejenigen von 128.220.11.2 unterbunden werden. In der gleichen Weise darf *haxor.evil.org* die Verbindung nicht herstellen, aber alle anderen Hosts aus *evil.org* können das tun. *client.example.com* ist falsch konfiguriert, weshalb seine Anfragen abgelehnt werden. Schließlich wird bei allen Hosts die Verbindung abgelehnt, die keinem dieser Einträge entsprechen.

Siehe auch

xinetd.conf(5).

3.9 Den Zugriff auf bestimmte entfernte Hosts beschränken (xinetd mit libwrap)

Problem

Sie möchten mittels `xinetd` den Zugriff auf einen TCP-Dienst auf bestimmte entfernte Hosts beschränken. `xinetd` wurde mit libwrap-Unterstützung kompiliert.

Lösung

Steuern Sie den Zugriff über */etc/hosts.allow* und */etc/hosts.deny*. Um Telnet-Verbindungen beispielsweise nur von 192.168.1.100 und den Hosts in der Domain *example.com* zu erlauben, nehmen Sie Folgendes in */etc/hosts.allow* auf:

```
in.telnetd : 192.168.1.100
in.telnetd : *.example.com
in.telnetd : ALL : DENY
```

Setzen Sie dann `xinetd` zurück, um die Änderungen zu übernehmen [3.3].

Erläuterung

Wenn Sie Ihre Zugriffskontrollen in */etc/hosts.allow* und */etc/hosts.deny* zusammenfassen wollen, statt die `xinetd`-spezifischen Methoden zu verwenden [3.8], oder wenn Sie die *hosts.allow*-Syntax und -Fähigkeiten bevorzugen, dann ist diese Technik für Sie genau richtig. Diese Dateien unterstützen eine umfangreiche Syntax, mit deren Hilfe Sie die Hosts und Netzwerke festlegen können, die Verbindungen zu Ihrem System über bestimmte Dienste herstellen dürfen oder nicht.

Das funktioniert aber nur, wenn `xinetd` mit libwrap-Unterstützung kompiliert wurde. Um herauszufinden, ob das so ist, sehen Sie sich die folgende Ausgabe an:

```
$ strings /usr/sbin/xinetd | grep libwrap
libwrap refused connection to %s from %s
%s started with libwrap options compiled in.
```

Wenn Sie wie oben Formatstrings im `printf`-Stil sehen, dann unterstützt Ihr `xinetd` libwrap.

Siehe auch

xinetd(8), hosts.allow(5).

3.10 Den Zugriff auf bestimmte entfernte Hosts beschränken (xinetd mit tcpd)

Problem

Sie möchten mittels `xinetd` den Zugriff auf einen TCP-Dienst auf bestimmte entfernte Host beschränken. `xinetd` wurde *nicht* mit libwrap-Unterstützung kompiliert.

Lösung

Tragen Sie die Zugriffsregeln in */etc/hosts.allow* und/oder */etc/hosts.deny* ein. Um zum Beispiel `telnet`-Verbindungen nur von 192.168.1.100 und den Hosts in der Domain *example.com* zu erlauben, erweitern Sie */etc/hosts.allow* um:

```
in.telnetd : 192.168.1.100
in.telnetd : *.example.com
in.telnetd : ALL : DENY
```

Dann ändern Sie */etc/xinetd.conf* oder */etc/xinetd.d/servicename* so ab, dass `tcpd` anstelle des Dienstes ausgeführt wird:

Alte /etc/xinetd.conf oder /etc/xinetd.d/telnet:

```
service telnet
{
    ...
    flags = ...
    server = /usr/sbin/in.telnetd
    ...
}
```

Neue /etc/xinetd.conf oder /etc/xinetd.d/telnet:

```
service telnet
{
    ...
    flags = ... NAMEINARGS
    server = /usr/sbin/tcpd
    server_args = /usr/sbin/in.telnetd
    ...
}
```

Danach starten Sie `xinetd` neu, um die Änderungen zu übernehmen [3.3].

Erläuterung

Diese Technik eignet sich für den seltenen Fall, dass Sie aus irgendeinem Grund die in `xinetd` fest eingebaute Zugriffskontrolle [3.8] nicht nutzen wollen und dass Ihr `xinetd` ohne libwrap-Unterstützung kompiliert wurde. Es bildet die ursprüngliche `inetd`-Methode der Zugriffskontrolle mittels TCP-Wrapper nach [3.11].

Sie müssen das Flag NAMEINARGS aufnehmen, das xinetd anweist, in der server_args-Zeile nach dem Namen für das Service-Executable zu suchen (in diesem Fall */usr/sbin/in.telnetd*).

Siehe auch

xinetd(8), hosts.allow(5), tcpd(8).

3.11 Den Zugriff auf bestimmte entfernte Hosts beschränken (inetd)

Problem

Sie möchten den Zugriff auf einen bestimmten TCP-Dienst über inetd nur auf bestimmte Hosts beschränken.

Lösung

Verwenden Sie tcpd, und legen Sie entsprechende Regeln in */etc/hosts.allow* und/oder */etc/hosts.deny* fest. Hier ein Beispiel für das Wrapping des Telnet-Daemons in.telnetd. Der Zugriff wird auf die IP-Adresse 192.168.1.100 und die Hosts der Domain *example.com* beschränkt. Tragen Sie Folgendes in die Datei */etc/hosts.allow* ein:

```
in.telnetd : 192.168.1.100
in.telnetd : *.example.com
in.telnetd : ALL : DENY
```

Setzen Sie in den entsprechenden Konfigurationsdateien tcpd anstelle Ihres Dienstes ein und starten Sie inetd neu.

Erläuterung

Die Steuerdateien */etc/hosts.allow* und */etc/hosts.deny* definieren Regeln, nach denen entfernte Hosts auf lokale TCP-Dienste zugreifen dürfen (oder nicht zugreifen dürfen). Der Daemon für diese Zugriffskontrolle, tcpd, verarbeitet die Regeln und entscheidet, ob ein Dienst gestartet wird oder nicht.

Zuerst definieren Sie die Zugriffsregeln in */etc/hosts.allow* und/oder */etc/hosts.deny*. Dann passen Sie */etc/inetd.conf* so an, dass es den Dienst über tcpd aufruft:

Alte /etc/inetd.conf:

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

Neue /etc/inetd.conf:

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

Abschließend starten Sie inetd, um die Änderungen zu übernehmen [3.4].

Siehe auch

hosts.allow(5), tcpd(8), inetd.conf(5).

3.12 Tageszeit-bezogene Zugriffsbeschränkungen

Problem

Sie möchten, dass ein Dienst nur zu bestimmten Tageszeiten verfügbar ist.

Lösung

Bei `xinetd` verwenden Sie das `access_times`-Attribut. Um `telnetd` beispielsweise jeden Tag nur von 8 Uhr morgens bis 17 Uhr nachmittags zugänglich zu machen, geben Sie Folgendes ein:

/etc/xinetd.conf oder /etc/xinetd.d/telnet:

```
service telnet
{
    ...
    access_times = 8:00-17:00
}
```

Für `inetd` implementieren wir eine Lösung mit dem `m4`-Makroprozessor und `cron` von Hand. Zuerst führen wir einige Strings ein, die bestimmte Tageszeiten repräsentieren, etwa »working« für 8 Uhr morgens und »playing« für 17 Uhr nachmittags. Dann entwickeln wir ein Skript (z.B. `inetd-services`), das `m4` zur Auswahl von Zeilen in einer Template-Datei verwendet, daraus die `inetd`-Konfigurationsdatei erzeugt und `inetd` dann ein Signal sendet, um die Konfiguration neu einzulesen:

/usr/local/sbin/inetd-services:

```
#!/bin/sh
m4 "$@" /etc/inetd.conf.m4 > /etc/inetd.conf.$$
mv /etc/inetd.conf.$$ /etc/inetd.conf
kill -HUP `pidof inetd`
```

Kopieren Sie die ursprüngliche */etc/inetd.conf* in die Template-Datei */etc/inetd.conf.m4*. Editieren Sie die Template-Datei so, dass Dienste entsprechend dem Wert eines Parameters, z.B. TIMEOFDAY, aktiviert werden. Die Zeile für den Telnet-Dienst, die ursprünglich wie folgt aussah:

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

könnte dann so aussehen:

```
ifelse(TIMEOFDAY,working,telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd)
```

Sie bedeutet: »Wenn `TIMEOFDAY` den Wert `working` enthält, dann nimm die Telnet-Zeile auf, anderenfalls nicht.« Abschließend richten wir *crontab*-Einträge ein, um den Dienst

zu bestimmten Tageszeiten zu aktivieren bzw. zu deaktivieren. Hierzu legen wir den TIMEOFDAY-Parameter entsprechend fest:

```
0  8 * * * /usr/local/sbin/inetd-services -DTIMEOFDAY=working
0 17 * * * /usr/local/sbin/inetd-services -DTIMEOFDAY=playing
```

Erläuterung

Bei xinetd können wir jeden Dienst auf einfache Weise über den access_times-Parameter kontrollieren. Die Zeitangaben erfolgen im 24-Stunden-Format.

Bei inetd müssen wir etwas mehr tun und die Konfigurationsdatei zu verschiedenen Tageszeiten neu generieren, um einen Dienst zu aktivieren bzw. zu deaktivieren. Das Rezept kann natürlich um zusätzliche Parameter und Werte (wie Sie TIMEOFDAY gesehen haben) erweitert werden. Beachten Sie, dass die xinetd-Lösung mit Zeitbereichen arbeitet, während die inetd-Lösung bestimmte Zeitpunkte nutzt (d.h. den Augenblick, in dem cron unser inetd-services startet).

Siehe auch

xinetd.conf(5), inetd.conf(5), m4(1), crontab(5).

3.13 Zugriffe auf einen SSH-Server auf bestimmte Hosts beschränken

Problem

Sie möchten den Zugriff auf sshd auf bestimmte entfernte Hosts beschränken.

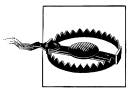
Lösung

Nutzen Sie die in sshd fest eingebaute TCP-Wrapper-Unterstützung. Nehmen Sie einfach entsprechende Regeln in die Dateien */etc/hosts.allow* und */etc/hosts.deny* auf, und legen Sie dabei sshd als Dienst fest. Soll z.B. nur 192.168.0.37 auf Ihren SSH-Server zugreifen können, nehmen Sie die folgende Zeile in */etc/hosts.allow* auf:

```
sshd: 192.168.0.37
sshd: ALL: DENY
```

Erläuterung

Der Aufruf von tcpd oder eines anderen Programms ist nicht notwendig, weil sshd die Regeln direkt verarbeitet.



Die TCP-Wrapper-Unterstützung in `sshd` ist optional und wird bei der Kompilierung festgelegt. Bei Red Hat 8.0 ist sie enthalten, bei SuSE nicht. Wenn Sie nicht sicher sind oder wenn Ihr `sshd` die Einstellungen in */etc/hosts.allow* und */etc/hosts.deny* zu ignorieren scheint, dann prüfen Sie, ob die entsprechende Unterstützung einkompiliert wurde:

```
$ strings /usr/sbin/sshd | egrep 'hosts\.(allow|deny)'
/etc/hosts.allow
/etc/hosts.deny
```

Gibt egrep nichts aus, dann wird der TCP-Wrapper nicht unterstützt. Laden Sie OpenSSH von *http://www.openssh.com* herunter (oder verwenden Sie die Quell-RPM Ihres Anbieters), und kompilieren Sie es neu:

```
$ ./configure --with-libwrap ...weitere Optionen...
$ make
# make install
```

Siehe auch

sshd(8), hosts_access(5).

3.14 Zugriffe auf einen SSH-Server auf bestimmte Accounts beschränken

Problem

Sie möchten, dass nur bestimmte Accounts Ihrer Maschine eingehende SSH-Verbindungen akzeptieren.

Lösung

Verwenden Sie das `sshd`-Schlüsselwort `AllowUsers` in */etc/ssh/sshd_config*. Sollen beispielsweise SSH-Verbindungen auf die Accounts smith und jones von überall erlaubt sein, auf alle anderen Accounts aber nicht, dann geben Sie Folgendes ein:

/etc/ssh/sshd_config:

```
AllowUsers smith jones
```

Damit SSH-Verbindungen von *remote.example.com* auf den Account smith erlaubt sind, alle anderen SSH-Verbindungen aber unterbunden werden, geben Sie ein:

```
AllowUsers smith@remote.example.com
```

Beachten Sie, dass dies *nichts* über den entfernten Benutzer »smith@remote.example.com« aussagt. Es handelt sich um eine Regel über Verbindungen *von* der Site *remote.example.com zum* lokalen Account smith.

Nach der Modifikation von *sshd_config* müssen Sie `sshd` neu starten, um Ihre Änderungen zu übernehmen.

Erläuterung

`AllowUsers` legt eine Liste lokaler Accounts fest, bei denen SSH-Verbindungen erlaubt sind. Die Liste ist definitiv: Jeder nicht aufgeführte Account kann keine SSH-Verbindungen aufbauen.

Die zweite Syntax-Form (`Benutzer@Host`) sieht unglücklicherweise wie eine E-Mail-Adresse aus bzw. wie eine Referenz auf einen entfernten Benutzer, aber das ist nicht der Fall. Die Zeile

```
AllowUsers benutzer@entfernterhost
```

bedeutet: »Erlaube dem entfernten System *entfernterhost* die SSH-Verbindung mit meinem lokalen Account *benutzer*.«

Die Aufnahme in die `AllowUsers`-Zeile garantiert keinen Zugang durch `sshd`: Der entfernte Benutzer muss sich nach wie vor mit normalen Mitteln authentifizieren (Passwort, Public Key etc.), von den anderen Hürden auf dem Weg gar nicht zu reden (Firewall-Regeln etc.).

Siehe auch

sshd_config(5).

3.15 Dienste auf bestimmte Verzeichnisse des Dateisystems beschränken

Problem

Sie möchten einen chroot-»Käfig« (cage) erzeugen, um einen Dienst auf ein bestimmtes Verzeichnis Ihres Dateisystems (und auf dessen Unterverzeichnisse) zu beschränken.

Lösung

Legen Sie einen *chroot-Käfig* an, indem Sie das GNU-Programm `chroot` anstelle des Dienstes ausführen. Übergeben Sie das Executable für den Dienst als Argument. Mit anderen Worten machen Sie aus:

/etc/xinetd.conf oder /etc/xinetd.d/myservice:

```
service myservice
{
    ...
    server      = /usr/sbin/myservice -a -b
    ...
}
```

das:

```
service myservice
{
     ...
     user = root
     server = /usr/sbin/chroot
     server_args = /var/cage /usr/sbin/myservice -a -b
     ...
}
```

Erläuterung

`chroot` verlangt zwei Argumente: ein Verzeichnis und ein Programm. Es zwingt das Programm, sich so zu verhalten, als wäre das angegebene Verzeichnis das Wurzelverzeichnis »/« des Dateisystems. Das verhindert, dass das Programm auf Dateien zugreifen kann, die nicht unterhalb des chroot-Verzeichnisses liegen, weil diese Dateien aus der chroot-Sicht des Dateisystems keine Namen besitzen. Selbst wenn ein Programm mit Root-Rechten ausgeführt wird, kann es diese Einschränkung nicht umgehen. Der von `chroot` verwendete Systemaufruf (der ebenfalls `chroot` heißt), funktioniert nur in eine Richtung: einmal aufgerufen, gibt es keine Möglichkeit, ihn im Kontext des aufrufenden Prozesses, bzw. seiner Child-Prozesse wieder rückgängig zu machen.

Ein chroot-Käfig ist besonders wirksam, wenn das Programm nach seinem Start seine Root-Rechte wieder aufgibt – viele Daemons können so konfiguriert werden, genau das zu tun. Ein in einem chroot-Käfig eingesperrtes Programm mit Root-Rechten kann aber immer noch Schaden anrichten, indem es neue Gerätedateien erzeugt und nutzt, oder Systemaufrufe bösartig nutzt, die nichts mit dem Dateisystem zu tun haben (wie etwa ein Reboot!).

Im normalen Betrieb kann ein Programm auf viele Dateien zugreifen, die nicht direkt etwas mit seiner Aufgabe zu tun haben, was den praktischen Nutzen von `chroot` einschränken kann. Sie könnten einen so großen Teil des Dateisystems innerhalb des Käfigs duplizieren müssen, dass sich der Nutzen des Käfigs negiert – insbesondere wenn es sich um sensitive Dateien (z.B. Ihre Passwortdatei zur Authentifizierung) handelt oder wenn sich die Dateien ändern. Im ersten Fall ist es besser, wenn der Dienst selbst `chroot` unterstützt und die `chroot`-Operation durchführt, nachdem auf alle benötigten allgemeinen Systemressourcen zugegriffen wurde. Im letztgenannten Fall können Sie mit festen Links (hard links) arbeiten, um außerhalb des Käfigs liegende Dateien trotzdem zugänglich zu machen. Das funktioniert allerdings nur mit Dateien, die im gleichen Dateisystem liegen wie der Käfig. Symbolische Links sind unwirksam, weil sie im Kontext des Käfigs verfolgt werden.

Damit `chroot` funktionieren kann, müssen Sie root sein, und das verwendete »Käfig«-Verzeichnis muss eine Linux-Verzeichnisstruktur aufweisen, die ausreicht, um `myservice` auszuführen. Im vorstehenden Beispiel muss */var/cage* die Verzeichnisse */var/cage/usr/sbin/myservice*, */var/cage/lib* (die alle von `myservice` verwendeten Bibliotheken enthalten müssen) und so weiter enthalten. Anderenfalls erhalten Sie Fehler wie:

```
chroot: cannot execute programm_name: No such file or directory
```

Das kann zu einer Art Detektivspiel ausarten. Um zum Beispiel einen so einfachen Befehl wie

```
# chroot /var/cage /usr/bin/who
```

funktionsfähig zu machen, muss das Verzeichnis */var/cage* die folgenden Bibliotheken enthalten:

/usr/bin/who
/lib/ld-linux.so.2
/lib/libc.so.6
/var/log/wtmp
/var/run/utmp

Die Befehle `ldd` und `strings` können Sie dabei unterstützen, die Shared Libraries und Dateien zu ermitteln, die durch den Dienst verwendet werden:

```
$ ldd /usr/sbin/myservice
... Ausgabe...
$ strings /usr/sbin/myservice | grep /
... Ausgabe...
```

Siehe auch

chroot(1), xinetd.conf(5), strings(1), ldd(1). Ist auf Ihrem System keine `ldd`-Manpage installiert, können Sie mit `ldd --help` entsprechende Nutzungshinweise abrufen.

3.16 Denial-of-Service-Angriffe verhindern

Problem

Sie möchten einen Denial-of-Service-Angriff (DOS) gegen einen Netzwerkdienst verhindern.

Lösung

Bei `xinetd` verwenden Sie die Schlüsselwörter `cps`, `instances`, `max_load` und `per_source`.

/etc/xinetd.conf oder /etc/xinetd.d/myservice:

```
service myservice
{
    ...
    cps = 10 30         Auf 10 Verbindungen pro Sekunde beschränken.
                        Wird das Limit überschritten, pausieren wir 30 Sekunden.
    instances = 4       Auf 4 gleichzeitige Instanzen von myservice beschränken.
    per_source = 2      Auf 2 simultane Sessions je Quell-IP-Adresse beschränken.
                        Der Standardwert UNLIMITED steht für kein Limit.
```

```
    max_load = 3.0      Neue Anfragen ablehnen, wenn die durchschnittliche Systemlast pro Minute
                        den Wert 3.0 übersteigt.
}
```

Bei `inetd` können Sie `inetd -R` verwenden, um festzulegen, wie oft ein Dienst pro Minute maximal aufgerufen werden kann. Voreingestellt ist 256.

Erläuterung

Diese Schlüsselwörter können einzeln oder in Kombination verwendet werden. Das Schlüsselwort `cps` schänkt die Zahl der Verbindungen ein, die ein Service pro Sekunde akzeptiert. Wird dieses Limit überschritten, deaktiviert `xinetd` den Dienst kurzfristig. Wie lange der Dienst zu deaktivieren ist, legen Sie mit dem zweiten Argument (in Sekunden) fest.

Das Schlüsselwort `instances` schränkt die Zahl gleichzeitiger Instanzen des gegebenen Dienstes ein. Standardmäßig gibt es kein Limit, auch wenn man das explizit wie folgt ausdrücken kann:

```
instances = UNLIMITED
```

Das Schlüsselwort `per_source` ist ähnlich, beschränkt aber nicht die Server-Instanzen, sondern die Anzahl der Sessions je Quell-IP-Adresse. Um beispielsweise zu verhindern, dass ein entfernter Host mehrere FTP-Verbindungen gleichzeitig herstellen kann, geben Sie Folgendes ein:

/etc/xinetd.conf oder /etc/xinetd.d/ftp:

```
service ftp
{
    ...
    per_source = 1
}
```

Das Schlüsselwort `max_load` deaktiviert schließlich einen Dienst, wenn die lokale System-Durchschnittslast zu hoch wird, um zu verhindern, dass die CPU »abgewürgt« wird.

`inetd` ist weniger flexibel: Es besitzt die Option `-R`, die die Anzahl der Aufrufe eines Dienstes pro Minute beschränkt. Dieses Limit gilt individuell für alle Dienste. Wird diese Grenze überschritten, schreibt inetd eine Meldung wie die folgende in das Log:

```
telnet/tcp server failing (looping), service terminated
```

Tatsächlich wird der Dienst aber nicht beendet, sondern nur für 10 Minuten deaktiviert. Diese Zeitspanne kann nicht angepasst werden.

Einige Firewalls besitzen ähnliche Features: Zum Beispiel kann `iptables` die Gesamtzahl eingehender Verbindungen beschränken. Andererseits unterstützt `iptables` die `per_source`-Funktionalität nicht. Das heißt, es kann die Gesamtzahl je Quelladresse nicht beschränken.

Siehe auch

xinetd.conf(5).

3.17 Umleitung an einen anderen Socket

Problem

Sie möchten eine Verbindung auf einen anderen Host und/oder Port auf der gleichen oder auf einer anderen Maschine umleiten (Redirect).

Lösung

Verwenden Sie das `xinetd`-Schlüsselwort `redirect`:

/etc/xinetd.conf oder /etc/xinetd.d/myservice:

```
service myservice
{
     ...
     server =  Pfad auf Original-Dienst
     redirect = IP_adresse  portnummer
}
```

Das Schlüsselwort `server` ist notwendig, aber sein Wert wird ignoriert. `xinetd` aktiviert einen Dienst nicht, solange es keine `server`-Einstellung besitzt, selbst wenn der Dienst umgeleitet wird.

Erläuterung

Wenn Sie eingehende `finger`-Verbindungen (Port 79) auf die Maschine 192.168.14.21 umleiten wollen, geben Sie Folgendes ein:

/etc/xinetd.conf oder /etc/xinetd.d/finger:

```
service finger
{
     ...
     server = /usr/sbin/in.fingerd
     redirect = 192.168.14.21 79
}
```

Natürlich können Sie Verbindungen auch auf völlig andere Dienste, z.B. `qotd` an Port 17, umleiten:

```
service finger
{
     ...
     server = /usr/sbin/in.fingerd
     redirect = 192.168.14.21 17
}
```

Nun erhalten eingehende `finger`-Anfragen ein amüsantes »Zitat des Tages« (»quote of the day«), wenn der `qotd`-Dienst auf der anderen Maschine aktiviert ist. Sie können Anfragen auch auf einen anderen Port auf der gleichen Maschine umleiten.

Siehe auch

xinetd.conf(5). Eine Einführung finden Sie unter *http://www.macsecurity.org/resources/xinetd/tutorial.shtml*.

3.18 Protokollieren von Zugriffen auf Ihre Dienste

Problem

Sie möchten wissen, auf welche Dienste über `xinetd` zugegriffen wird.

Lösung

Aktivieren Sie das Protokollieren (Logging) in der Konfigurationsdatei des Dienstes:

/etc/xinetd.conf oder /etc/xinetd.d/myservice:

```
service myservice
{
    ...
    log_type = SYSLOG Facility Level
    log_on_success = DURATION EXIT HOST PID USERID
    log_on_failure = ATTEMPT HOST USERID
}
```

`xinetd` schreibt seine Logs standardmäßig an `syslog`. Soll das Logging stattdessen in eine Datei erfolgen, passen Sie die oben stehende `log_type`-Zeile wie folgt an:

```
log_type = FILE dateiname
```

Erläuterung

`xinetd` kann Diagnosemeldungen über `syslog` festhalten oder direkt in eine Datei schreiben. Beim Einsatz von `syslog` wählen Sie eine so genannte Facility (`daemon`, `local0` etc.) und optional einen Log-Level (`crit`, `warning` etc.), wobei `info` voreingestellt ist.

```
log_type = SYSLOG daemon              facility = daemon, level = info
log_type = SYSLOG daemon warning      facility = daemon, level = warning
```

Beim Logging in eine Datei geben Sie einfach einen Dateinamen an:

```
log_type = FILE /var/log/myservice.log
```

Optional können Sie feste und weiche Grenzen (hard and soft limits) für die Größe der Log-Datei festlegen: siehe xinetd.conf(5).

Logmeldungen können erzeugt werden, wenn Dienste erfolgreich gestartet und beendet werden (über `log_on_success`) oder wenn sie fehlschlagen oder Verbindungen ablehnen (über `log_on_failure`).

Wenn das Logging bei Ihnen nicht funktioniert, hat dies sehr wahrscheinlich etwas mit einem fehlerhaften Setup in */etc/syslog.conf* zu tun. Es ist sehr leicht, subtile Konfigurationsfehler zu begehen und Ihre Logmeldungen falsch weiterzuleiten. Führen Sie unser `syslog`-Testskript aus, um zu sehen, wo Ihre Meldungen landen [9.28].

Siehe auch

xinetd.conf(5), syslog.conf(5), inetd.conf(5).

3.19 Root-Logins über Terminals unterbinden

Problem

Sie möchten verhindern, dass sich der Superuser (also root) direkt über ein (Pseudo-)Terminal einloggen kann.

Lösung

Editieren Sie */etc/securetty*. Diese Datei enthält Gerätenamen (einen pro Zeile), die root-Logins erlauben. Stellen Sie sicher, dass keine Pseudo-TTYs (pty) aufgeführt sind, so dass kein Root-Login über das Netzwerk möglich ist, und entfernen Sie alle anderen Geräte, die für Sie von Belang sind. Die Zeilen beinhalten nicht den führenden »/dev/«-Pfad, und mit einem Doppelkreuz (#) beginnende Zeilen sind Kommentare. Ein Beispiel:

/etc/securetty:

```
# Serielle Schnittstellen
tty1
tty2
# devfs-Geräte
vc/1
vc/2
```

Erläuterung

Sie sollten ein direktes root-Login nach Möglichkeit unterbinden. Wenn Sie das nicht machen, eröffnen Sie einen möglichen Weg für einen Einbruch in Ihr System: Ein Außenstehender kann (beispielsweise) einen Dictionary-Angriff gegen das fragliche Terminal führen. Die Benutzer sollen sich vielmehr unter ihrem eigenen Account einloggen und Root-Rechte auf geeignete Art und Weise erlangen (die wir in Kapitel 5 beschreiben).

Siehe auch

securetty(5). Die Dokumentation zu devfs finden Sie auf *http://www.atnf.csiro.au/people/rgooch/linux/docs/devfs.html*.

CH03 Seite 78 Mittwoch, 12. November 2003 1:37 13