

Ralf Spenneberg

# Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6  
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

---

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England  
Don Mills, Ontario • Sydney • Mexico City  
Madrid • Amsterdam



# 23 Das /proc-Dateisystem

Linux besitzt einen sehr mächtigen TCP/IP-Stack und Paketfilter. Das Verhalten dieses Stacks und des Paketfilters ist sehr flexibel. Wenn Sie die Flexibilität kennen lernen möchten und den Stack oder den Paketfilter anpassen und modifizieren möchten, müssen Sie sich mit dem /proc-Dateisystem beschäftigen. Dieses virtuelle Dateisystem gibt Ihnen lesenden und schreibenden Zugriff auf wesentliche Variablen des Betriebssystems. Hierzu gehören auch der Netzwerkstapel und der Paketfilter.

## 23.1 Einführung in /proc

Das /proc-Dateisystem ist ein virtuelles Dateisystem, in dem der Kernel viele Variablen des Betriebssystems für den lesenden und häufig auch schreibenden Zugriff anbietet. Wesentliche Informationen des Betriebssystems können (nur) über diese Struktur ausgelesen werden.

Wenn Sie sich den Inhalt des /proc-Verzeichnisses ansehen, finden Sie sowohl Verzeichnisse als auch Dateien vor (Abbildung 23.1).

```

spenneb@bibo:~$ ls /proc
1          2186  2586  3195  3593  3757  3907  5589  driver      misc
117        2209  2773  3207  3566  3759  3909  5373  execdomains modules
120        2227  2802  3216  3574  3761  3971  5374  fb          mounts
1488       2241  2837  3221  3632  3763  3976  5394  filesystems mtrr
1496       2245  2921  3243  3666  3768  3978  5396  fs         net
1514       2271  2930  3244  3683  3770  3980  6018  ide        partitions
1526       2282  2945  3245  3692  3771  4      7      interrupts pci
165        2283  3      3246  3693  3774  4010  332    iomem     self
167        2291  3000  3247  3696  3778  4015  acpi    ioports   slabinfo
168        2293  3008  3248  3699  3800  4049  asound  irq       stat
1681       2308  3017  3249  3701  3802  4057  bluetooth kallsyms swaps
1682       2345  3036  3339  3705  3803  4065  buddyinfo kcore     sgs
1683       2357  3042  3363  3708  3804  415   bus     keys      sysrq-trigger
1684       2482  3065  3426  3718  3829  4187  cmdline key-users sysvipc
169       2523  3082  3506  3728  3840  4267  cpuinfo kmsg     tty
1781       2537  3104  3628  3745  3841  431   crypto  loadavg  uptime
1870       256  3128  3656  3747  3856  4473  devices locks     version
2         2568  3157  3657  3748  3893  4731  diskstats mdstat   vmstat
2152      2578  3182  3658  3751  3894  5      dma     meminfo  vmstat
[spenneb@bibo ~]$
  
```

Abbildung 23.1: Das /proc-Dateisystem enthält sowohl Informationen über Prozesse als auch das laufende Betriebssystem.

Ein großer Teil der Verzeichnisse stellt Informationen über die laufenden Prozesse zur Verfügung. Jedes Verzeichnis mit einer Zahl als Verzeichnisnamen enthält Informationen über den Prozess mit der entsprechenden Prozess-ID. Die weiteren Einträge und Verzeichnisse enthalten Informationen über das laufende Betriebssystem und erlauben teilweise auch die Konfiguration dieser Parameter. So enthält zum Beispiel der Eintrag `/proc/cpuinfo` Informationen über den von dem Betriebssystem gefundenen Prozessor. In der Datei `/proc/partitions` finden Sie Informationen über die von dem Betriebssystem erkannten Partitionen:

```
$ cat /proc/partitions
major minor #blocks name

 3      0 78150744 hda
 3      1  104391 hda1
 3      2 78043770 hda2
22      0 78150744 hdc
22      1 78150712 hdc1
253     0 75956224 dm-0
253     1  2031616 dm-1
253     2 20971520 dm-2
253     3 20971520 dm-3
253     4 20971520 dm-4

$ cat /proc/cmdline
ro root=/dev/Vo1Group00/LogVo100
```

Die Datei `/proc/cmdline` enthält Informationen über die Kommandozeile, mit der der laufende Kernel aufgerufen wurde.

Teilweise können Sie die Werte in den (virtuellen) Dateien auch verändern. So definiert die Datei `/proc/sys/net/ipv4/icmp_echo_ignore_all`, ob der Kernel auf ICMP-Echo-Pakete reagiert oder diese ignoriert. Diese Datei kann den Wert 0 oder 1 enthalten. Ein Wert von 0 bedeutet, dass diese Funktion nicht aktiv ist. Ein Wert von 1 bedeutet in diesem Fall, dass der Kernel diese Pakete ignoriert.

Sie können das selbst sehr einfach prüfen und nachvollziehen. Zeigen Sie zunächst den Wert der Datei an:

```
$ cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
```

Starten Sie nun in einem eigenen Fenster den Ping-Befehl (`ping 127.0.0.1`). Sie sollten erfolgreich Ihren eigenen Rechner pingen können. Falls es nicht funktioniert, prüfen Sie bitte, ob eine Firewall aktiv ist und deaktivieren Sie diese im Zweifelsfall.

Während der Ping läuft, ändern Sie nun den Inhalt der Datei:

```
$ echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

Nun sollte der Ping sofort aufhören. Sobald Sie die 0 wieder in die Datei schreiben, läuft der Ping weiter. Anstelle des `echo`-Kommandos können Sie auch den Befehl `sysctl` verwenden. Dieser Befehl erwartet den Namen der Variablen etwas anders:

```
$ /sbin/sysctl net.ipv4.icmp_echo_ignore_all
net.ipv4.icmp_echo_ignore_all = 1
```

Entfernen Sie `/proc/sys/` von dem Pfad der Datei, und ersetzen Sie alle weiteren `/` (Schrägstrich) durch einen Punkt. Allerdings kann der Befehl auch die Schrägstriche statt Punkte verwenden:

```
$ /sbin/sysctl net/ipv4/icmp_echo_ignore_all
net.ipv4.icmp_echo_ignore_all = 0
```

Wenn Sie den Befehl nur mit dem Namen der Variablen verwenden, zeigt der Befehl nur den aktuellen Wert an. Wenn Sie den Wert ändern möchten, verwenden Sie zusätzlich die Option `-w` und geben den neuen Wert an:

```
$ /sbin/sysctl -w net/ipv4/icmp_echo_ignore_all=1
net.ipv4.icmp_echo_ignore_all = 1
```

Alle so durchgeführten Änderungen sind nach einem Neustart verloren und müssen neu gesetzt werden. Viele Distributionen verwenden hierzu inzwischen die Datei `/etc/sysctl.conf`. Sie können die Änderungen, die von der Distribution automatisch beim Systemstart durchgeführt werden sollen, in dieser Datei angeben:

```
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0
```

Im Folgenden werde ich die einzelnen Variablen vorstellen, die für die Konfiguration des Paketfilters und TCP/IP-Stacks interessant sind.

## 23.2 /proc/net/

Alle Einträge in diesem Verzeichnis sind lediglich lesbar. Ein Schreibzugriff ist nicht vorgesehen. Die Dateien dienen lediglich informativen Zwecken.

### 23.2.1 ip\_contrack

Dies ist die Connection Tracking-Tabelle. Alle bekannten Verbindungen werden in dieser Tabelle aufgenommen, sobald das `ip_contrack`-Modul geladen wurde.

```
tcp      6 431842 ESTABLISHED src=192.168.0.108 dst=217.160.128.61 sport=56468 dport=993
         packets=79 bytes=5788 src=217.160.128.61 dst=192.168.0.108 sport=993 dport=56468
         packets=51 bytes=5864 [ASSURED] mark=0 use=1
```

Ab Kernelversion 2.6.10 und entsprechender Konfiguration (`CONFIG_IP_NF_CT_ACCT=y`) werden auch die Paket- und Byte-Zähler für die Verbindung angezeigt.

### 23.2.2 ip\_contrack\_expect

Ab Kernel 2.6.9 werden die Expectations in dieser Datei angezeigt. Expectations sind Verbindungen, die von einem Conntrack-Helper-Modul erkannt werden und als `RELATED`-Verbindungen erlaubt werden.

### 23.2.3 ip\_tables\_\*

Diese drei Dateien (`ip_tables_matches`, `ip_tables_targets` und `ip_tables_names`) enthalten die Namen der aktuell geladenen Matches (Tests), Targets (Ziele) und Tables (Tabellen). Sobald eine Regel einen neuen Match oder ein neues Target benötigt, das durch ein Modul zur Verfügung gestellt wird, wird das Modul geladen, und dessen Name taucht in diesen Dateien auf.

## 23.3 /proc/sys/net/ipv4

Dieses Verzeichnis enthält Variablen, auf die Sie sowohl lesend als auch schreibend zugreifen können. In vielen Fällen sollten Sie jedoch sich sehr genau überlegen, ob es sinnvoll ist, die entsprechenden Parameter zu verändern, da dies durchaus Auswirkungen (positive wie negative) auf die Leistungsfähigkeit des Systems haben kann.

### 23.3.1 icmp\_\*

Diese Variablen betreffen das ICMP-Protokoll. Falls Sie mehr oder weniger Optionen besitzen, so hängt dies von der verwendeten Kernelversion ab.

#### `icmp_echo_ignore_all`

Diese Variable entscheidet, ob der Kernel alle ICMP-Echo-Pakete (Ping) ignorieren soll. Schreiben Sie in diese Datei eine 1, und Ihr System wird auf ein Ping grundsätzlich nicht mehr reagieren.

### icmp\_echo\_ignore\_broadcasts

Wenn Sie diese Variable auf den Wert 1 setzen, ignoriert der Kernel alle Echo-Request-Pakete (Ping) an die Broadcast-Adresse. Üblicherweise beantworten alle Linux-/Unix-Betriebssysteme einen Echo-Request an ihre Broadcast-Adresse. So können Sie sehr leicht in einem Netzwerk alle verfügbaren Unix-Systeme ermitteln. Da aber viele Router in der Vergangenheit (und auch heute noch) derartige Broadcast-Ping-Anfragen nicht verworfen haben, konnte damit eine Verstärkung des Verkehrs erzeugt werden. Sie senden ein Ping an die Broadcast-Adresse eines Netzwerks mit 10 Unix-Rechnern und erhalten 10 Antworten zurück. Wenn Sie die Absenderadresse fälschen (IP-Spoofing), können Sie so einen anderen Rechner mit Ping-Antworten überschwemmen. Dieser Angriff trägt den Namen Smurf-Angriff. Die Netzwerke werden als Smurf Amplifier Networks bezeichnet. Auch heute gibt es noch derartige Netzwerke, deren Router die Broadcast-Pakete nicht verwerfen (<http://www.powertech.no/smurf/>). Um zu verhindern, dass Ihr System für diesen Angriff genutzt wird, sollten Sie die Beantwortung von Broadcast-Ping-Anfragen abschalten:

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

### icmp\_errors\_use\_inbound\_ifaddr

Normalerweise versendet ein Linux-System eine Fehlermeldung mit der Absenderadresse der Netzwerkkarte, über die die Fehlermeldung das System verlässt. Dies ist nicht zwingend die Netzwerkkarte, auf der die Fehlermeldung aufgetreten ist. Wird diese Option (ab 2.6.12) gesetzt, so wird die Fehlermeldung mit der Absenderadresse der Netzwerkkarte gesendet, auf der der Fehler auftrat. Da dies die Fehlersuche stark vereinfacht, sollte diese Variable aktiviert werden.

### icmp\_ignore\_bogus\_error\_messages

Einige Router verletzen den RFC 1122 (Requirements for Internet Hosts – Communication Layers) und versenden Fehlermeldungen als Antwort auf Broadcast-Mitteilungen. Normalerweise protokolliert der Kernel diese Tatsache. Wenn Sie diese Variable aktivieren, protokolliert der Kernel dies nicht mehr und kann so Ihre Protokolldateien nicht mit unnötigen Meldungen füllen. Ich empfehle die Aktivierung dieser Option.

### icmp\_ratelimit

Um Denial-of-Service-Angriffe mit ICMP-Nachrichten zu verhindern und allgemein den Netzwerkverkehr zu reduzieren, bieten diese und die nächste Variable Ihnen die Möglichkeit, die Versendung von ICMP-Meldungen zu beschränken. In dieser Variable definieren Sie, wie viel Jiffies der Kernel zwischen zwei bestimmten ICMP-Meldungen an einen bestimmten Rechner warten soll. Ein Jiffie entspricht auf der i386-Plattform 1/100 Sekunde. Der Default-Wert dieser Variable ist 100. Das bedeutet, dass maximal einmal pro Sekunde eine bestimmte ICMP-Nachricht an denselben Rechner versendet wird. Ein Wert von 0 schaltet dieses Verhalten ab.

### icmp\_ratemask

Diese Variable definiert, welche ICMP-Nachrichten das Rate-Limiting betrifft. Hierzu bedient sich diese Variable einer Bitmaske:

```
Bits:   IHGFEDCBA9876543210
Default: 0000001100000011000 (6168)
```

```
0 Echo Reply
1 Nicht definiert
2 Nicht definiert
3 Destination Unreachable *
4 Source Quench *
5 Redirect
8 Echo Request
B Time Exceeded *
C Parameter Problem *
D Timestamp Request
E Timestamp Reply
F Info Request
G Info Reply
H Address Mask Request
I Address Mask Reply
```

### 23.3.2 igmp\_\*

Das Internet-Group-Management-Protokoll (IGMP) wird für die Verwaltung von Multicast-Gruppen verwendet. Der Kernel verfügt über zwei Variablen, die das Verhalten dieses Protokolls beeinflussen.

#### igmp\_max\_memberships

Diese Variable definiert die maximale Anzahl an Multicast-Gruppen, bei denen das Linux-System Mitglied werden kann. Üblicherweise müssen Sie diesen Wert nicht modifizieren.

#### igmp\_max\_msf

Diese Datei definiert die maximale Anzahl der Multicast-Source-Filter (MSF). Das IGMPv3-Protokoll erlaubt es einer Applikation, Filter auf Multicast-Gruppen zu definieren, mit denen die Absender der Multicast-Nachrichten eingeschränkt werden können. Dann akzeptiert der Linux-Kernel diese Nachrichten nur noch von wenigen Systemen. Diese Variable definiert, wie viele Multicast-Source-Filter gesetzt werden dürfen.

### 23.3.3 inet\_peer\_\*

Ein Inet-Peer ist ein anderer Rechner im Internet, mit dem unser System im Moment kommuniziert. Für jeden derartigen Peer speichert unser System langlebige Informationen in einer eigenen Struktur. Aktuell handelt es sich hierbei nur um die IP-Identifikationsnummer (ID) des nächsten Pakets. Diese Nummer wird benötigt, um bei der Defragmentierung der Pakete die richtigen Fragmente eines Pakets zu erkennen. Diese weisen alle dieselbe IP-ID auf. Diese Variablen definieren, wie diese Strukturen verwaltet werden.



#### Achtung

Da ein Linux-System per Default Path-MTU-Discovery verwendet und so eine Fragmentierung des Pakets ausschließen kann, benötigen die versandten Pakete keine IP-ID. Daher werden in diesem Fall der entsprechende Code und diese Variablen nicht verwendet.

Da diese Variablen in den meisten Umgebungen nicht genutzt werden, werden sie hier nicht weiter erläutert. Sie finden eine aktuelle Dokumentation in dem Quellcode Ihres Kernels unter `Documentation/networking/ip-sysctl.txt`.

### 23.3.4 ip\_\*

Diese Variablen beeinflussen das IP-Protokoll.

#### ip\_autoconfig

Der Kernel kann selbst die Protokolle DHCP oder BOOTP verwenden, um eine Autokonfiguration der Netzwerkkarten vorzunehmen. Dazu muss der Kernel entsprechend konfiguriert sein (`IP_PNP=y`).

#### ip\_conntrack\_max

Diese Variable definiert die maximal unterstützten Verbindungen in der Connection Tracking-Tabelle. Der Default-Wert hängt von dem zur Verfügung stehenden Arbeitsspeicher ab. Auf einer Firewall ist es sinnvoll, diesen Wert zu erhöhen. Dann sollten Sie aber auch die Hashsize erhöhen (siehe Abschnitt 19.2).

#### ip\_default\_ttl

Hiermit setzen Sie den Time-to-Live-Wert (TTL) der von dem Linux-System versandten Pakete. Alle normalen Pakete erhalten diesen TTL-Wert. Nach der entsprechenden Anzahl von Hops wird das Paket dann verworfen. Jeder Router, der von einem Paket auf dem Weg zum Ziel passiert wird, ist ein Hop. Hiermit wird verhindert, dass eine Routerfehlfunktion zu Paketen führt, die ewig transportiert

werden. ICMP-Fehlermeldungen werden von Linux unabhängig von diesem Wert immer mit einem TTL von 255 versendet.

### **ip\_dynaddr**

Wenn Sie in Ihrer Firewall dynamische IP-Adressen auf der externen Netzwerkkarte bei gleichzeitigem Masquerading verwenden, sollten Sie diese Variable aktivieren, um fehlerfreie Verbindungen zu ermöglichen. Ansonsten kann es zu Problemen bei der ersten Verbindung kommen, die die Einwahl in das Internet antriggert. Um dies zu verstehen, stellen Sie sich vor, Ihr Firewall-System sei im Moment nicht mit dem Internet verbunden. Ein Client hinter der Firewall möchte auf das Internet zugreifen. Er versendet sein erstes Paket, um die Verbindung aufzubauen. Die Firewall filtert das Paket, maskiert es mit der aktuellen IP-Adresse der externen Netzwerkkarte und übergibt das Paket an die Netzwerkkarte. Dies triggert die Einwahl, und das System baut die Internetverbindung auf. Dabei erhält das System eine neue IP-Adresse auf der externen Netzwerkkarte. Wenn nun das erste Paket unverändert versendet werden würde, hätte es die falsche Absender-IP-Adresse. Diese Variable erzwingt eine erneute Maskierung dieses ersten Pakets.

Wenn Sie eine 1 in die Variable schreiben, aktivieren Sie das Verhalten. Ein Wert größer 1 führt zusätzlich jedes Mal zur Protokollierung.

### **ip\_forward**

Diese Variable schaltet für sämtliche Netzwerkkarten des Linux-Systems die Weiterleitung (Forwarding) ein. Wenn diese Variable den Wert 0 enthält, leitet der Linux-Kernel keine Pakete zwischen zwei Netzwerkkarten weiter.

### **ip\_local\_port\_range**

Diese Datei enthält zwei Zahlen, die den Bereich der lokal erlaubten Client-Ports definieren. Wenn Sie diese Variable verändern möchten, müssen Sie auch gleichzeitig beide Zahlen in diese Variable schreiben:

```
echo "4096 7000" > ip_local_port_range
```

Die erste angegebene Nummer ist der kleinste Port, der verwendet werden kann, während die zweite Nummer der höchste Port ist. Wählen Sie hier keine Ports kleiner 1024, um Problemen vorzubeugen. Üblich sind hier 32768 und 61000.

### **ip\_nonlocal\_bind**

Üblicherweise kann eine Applikation sich nur auf lokal definierte IP-Adressen binden. Wenn diese Variable gesetzt ist, kann eine Applikation auch IP-Adressen verwenden, die auf dem System nicht definiert sind, und mit dieser Adresse als Absenderadresse Pakete versenden und Pakete entgegennehmen. Diese Funktion wird für einen echten transparenten Proxy benötigt, der die Verbindung zum Server mit der IP-Adresse des echten Clients aufbaut.

## **ip\_no\_pmtu\_disc**

Der Linux-Kernel verwendet per Default die Path Maximum Transmission Unit Discovery (PMTU-Discovery). Damit verhindert der Linux-Kernel, dass eine Fragmentierung der Pakete unterwegs auftritt und bei gleichzeitigem Fragmentverlust unnötig viele Pakete wiederholt gesendet werden müssen. In einigen Umgebungen führt dieses Verhalten jedoch zu Problemen. Dann können Sie diese Funktion mit dieser Variablen abschalten (1).

## **23.3.5 ipfrag\_\***

Diese Parameter definieren, wie das Linux-System eine Defragmentierung durchführen soll.

### **ipfrag\_\*\_thresh**

Diese beiden Variablen definieren den Speicherbereich, der für die Speicherung der Fragmente bei der Defragmentierung genutzt wird. Der genutzte Speicher schwankt zwischen dem `ipfrag_low_thresh`- und `ipfrag_high_thresh`-Wert.

### **ipfrag\_time**

Dieser Wert definiert, wie lange Fragmente für eine Defragmentierung aufbewahrt werden. Ist innerhalb dieser Zeit keine Defragmentierung möglich, sendet der Rechner eine ICMP-Time-Exceeded-Fehlermeldung und verwirft das unvollständige Paket.

### **ipfrag\_secret\_interval**

Ältere Kernel besitzen eine Sicherheitslücke in der Fragmentierung, die einen Denial-of-Service der Defragmentierung ermöglichen (CAN-2003-0364). Um dieses zu verhindern, wurde ein Secret eingeführt, so dass der Angreifer die benötigten Pakete nicht mehr vorhersagen kann. Dieses muss zur Sicherheit in regelmäßigen Abständen neu berechnet werden. Das Intervall in Sekunden wird in dieser Variable definiert.

### **ipfrag\_max\_dist**

Diese neue Variable des Linux-Kernels 2.6.14 definiert die maximale Unordnung, mit der Fragmente den Kernel erreichen dürfen.

## **23.3.6 tcp\_\***

Hier sind sehr viele Variablen vorhanden, die das Verhalten des TCP-Protokolls beeinflussen.

### **tcp\_abort\_on\_overflow**

Diese Variable erzwingt einen Reset neuer Verbindungen, wenn der dazugehörige Dienst die neuen Verbindungen nicht schnell genug entgegennehmen kann. Bevor Sie diese Variable setzen, sollten Sie immer versuchen, den Dienst zu optimieren. Wenn es sich nicht um einen Dauerzustand handelt, sondern nur punktuell Probleme auftreten, kann diese Funktion hilfreich sein.

### **tcp\_adv\_win\_scale**

Die Variable `tcp_rmem` definiert den für den Empfang von Paketen reservierten Speicher eines Sockets. Dieser Speicher wird für das TCP-Window und den Applikationspuffer aufgeteilt. Diese Variable definiert dieses Verhältnis. Ist der Wert der Variable positiv, wird für das TCP-Window  $1/(2^{\text{tcp\_adv\_win\_scale}})$  genutzt. Bei dem Default-Wert von 2 wird also ein Viertel des Speichers für das TCP-Window reserviert. Bei einem negativen Wert wird  $1-1/(2^{-\text{tcp\_adv\_win\_scale}})$  verwendet – bei einem Wert von -2 also drei Viertel des Speichers.

### **tcp\_app\_win**

Auch diese Variable definiert, wie viel Speicher für den Applikationspuffer eines Sockets maximal reserviert wird. Der Kernel nutzt folgende Formel:  $\text{window} / (2^{\text{tcp\_app\_win}})$ . Der Wert muss immer mindestens der Maximum Segment Size (MSS) entsprechen.

### **tcp\_bic\***

Die Binary Increase Congestion-(BIC-)Control versucht die Probleme des TCP-Protokolls bei der Übertragung großer Datenmengen über große Entfernungen zu beheben. TCP verschenkt bei großen Entfernungen einen Großteil der Bandbreite. BIC ist eine reine Modifikation des Absenders. Der Empfänger muss nicht ebenfalls das BIC-Protokoll beherrschen.

Das BIC-Protokoll ist noch sehr jung (2004) und stellt hier einige Variablen zur Verfügung, mit denen Sie sein Verhalten einstellen können. Die Erläuterung der Optionen benötigt aber recht viel Wissen über das Protokoll selbst. Sie finden Hintergrundinformationen unter <http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/>.

### **tcp\_congestion\_control**

Der Linux-Kernel 2.6.13 und neuer kann verschiedene Methoden verwenden, um Verstopfungen der Netzwerkverbindungen zu erkennen und zu vermeiden. Aktuell stehen die folgenden zur Verfügung:

- reno (klassisches TCP)
- bic
- highspeed (Sally Floyd)

- htcp (Hamilton TCP)
- hybla (Speziell für Satellitenverbindungen)
- scalable
- vegas
- westwood (speziell für verlustreiche Netzwerke)

Sie können ganz einfach die Methode wählen:

```
sysctl -w net.ipv4.tcp_congestion_control=htcp
```

Sie finden weitere Informationen über die Methoden auf <http://www-iepm.slac.stanford.edu/bw/tcp-eval/>.

### tcp\_dsack

Diese Variable bestimmt, ob doppelte selektive Acknowledgments (sack) versendet werden sollen. Damit kann einem Absender mitgeteilt werden, dass ein Paket doppelt empfangen worden ist. D-SACK ist eine Erweiterung des SACK-Standards und wird in dem RFC 2883 genauer beschrieben. SACK und D-SACK beschleunigt die Übertragung größerer Datenmengen enorm, da ein Empfänger genau die empfangenen Pakete bestätigen kann und nicht immer nur das Paket mit der kleinsten Sequenznummer (siehe auch `tcp_sack`).

### tcp\_ecn

Die Explicit Congestion Notification (ECN) ist eine neue Methode, bei der zwei Kommunikationspartner Informationen über Netzwerk-Verstopfungen austauschen. Beide Kommunikationspartner müssen ECN unterstützen. Leider verwerfen immer noch viele Firewalls Pakete mit ECN-Informationen als bösartige Pakete, da sie oder ihre Administratoren diesen neuen Standard noch nicht kennen. Dann können Sie mit dieser Variable ECN abschalten. Ansonsten können Sie für bestimmte IP-Adressen auch in der Mangle-Tabelle das ECN-Target verwenden (siehe Abschnitt 21.2.4).

ECN wird in den beiden RFCs *RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP* und *RFC 2884 – Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks* besprochen.

### tcp\_fack

Forward Acknowledgment ist eine spezielle Methode, die bei selektiven Acknowledgments wesentlich radikaler nichtbestätigte Pakete zur Wiederversendung auswählt. Dabei geht diese Methode davon aus, dass, sobald ein selektives Acknowledgment empfangen wurde, alle Pakete mit kleineren Sequenznummern verloren gegangen sind und neu gesendet werden müssen. Da dies nur dann korrekt ist, wenn eine Umsortierung der Paketreihenfolge nicht erfolgt ist, wird für Verbindungen, bei denen eine Umsortierung erkannt wurde, das Verfahren abgeschaltet. Forward Acknowledgments sind nur aktiv, wenn auch selektive Acknowledgments

unterstützt werden. Da dies die Performance erhöht, sollte die Option eingeschaltet sein. Weitere Informationen finden Sie auf der Homepage des Entwicklers Matthew Mathis: <http://www.psc.edu/~mathis/>.

### **tcp\_fin\_timeout**

Wenn das lokale System eine Verbindung mit einem FIN-Paket beendet, muss auch der Kommunikationspartner die Verbindung mit einem FIN-Paket beenden. Diese Variable definiert, wie lange Ihr System auf ein Antwort-FIN-Paket des Kommunikationspartners warten muss. Einige Systeme reagieren hier nicht entsprechend den TCP-Standards und beenden ihre Seite der Verbindung nicht korrekt.

Dies kann zu Speicherproblemen auf Webservern führen, wenn viele defekte Clients die Verbindungen aufbauen, da für jeden offenen Socket 1,5 kByte Speicher reserviert werden. Der Default-Wert von 60 Sekunden kann auf diesen Systemen auf 30 Sekunden reduziert werden.

### **tcp\_frto**

Plötzliche Timeouts bei der wiederholten Sendung von verlorenen TCP-Paketen (spurious retransmission timeouts, RTO) können zu langsamen Übertragungsraten führen. Diese Timeouts treten vor allem bei unzuverlässigen Medien, zum Beispiel WLAN auf. Das RFC 4138 beschreibt die Forward RTO Recovery, die nur von dem Absender implementiert wird. Diese Option schaltet diese Methode im Kernel an.

### **tcp\_keepalive\_\***

Eine Applikation kann einen Socket mit der Option `SO_KEEPALIVE` öffnen. Der Kernel sendet dann, selbst wenn der Socket nicht verwendet wird, in regelmäßigen Abständen TCP-Keepalive-Pakete, um die Verbindung geöffnet zu halten und um einen Verbindungsabbruch zu erkennen.

Diese Variablen definieren, wie diese Keepalive-Pakete versendet werden.

#### *tcp\_keepalive\_time*

Diese Variable definiert, in welchen Abständen der Linux-Kernel ein Paket für eine ungenutzte Verbindung aussendet. Der Default-Wert beträgt 7200 Sekunden oder 2 Stunden.

#### *tcp\_keepalive\_probes*

Diese Variable definiert, wie viele unbeantwortete Keepalive-Pakete ausgesendet werden, bevor der Kernel die Verbindung als unterbrochen ansieht (Default: 9).

#### *tcp\_keepalive\_intv*

Diese Variable definiert, in welchem Interval der Kernel ein weiteres Keepalive-Paket aussendet, wenn das letzte Paket nicht beantwortet wurde (Default: 75 Sekunden).

Der Kernel sendet also alle 2 Stunden ein Paket. Wird dieses nicht beantwortet, sendet er anschließend alle 75 Sekunden 9-mal ein weiteres Paket. Werden auch diese nicht beantwortet, ist die Verbindung tot.

### **tcp\_low\_latency**

Diese Option definiert, welches Ziel der Kernel bei den TCP-Verbindungen verfolgen soll:

- Hoher Durchsatz (High Throughput, Default) oder
- Niedrige Verzögerung (Low Latency).

Ein High-Performance-Rechencluster hätte diese Variable typischerweise gesetzt.

### **tcp\_max\_orphans**

Diese Variable definiert, wie viele verwaiste (orphan) TCP-Sockets der Kernel toleriert, die nicht mehr mit einem User-File-Handle verbunden sind. Sobald die Anzahl überschritten wird, setzt der Kernel alle (!) verwaisten Sockets zurück. Damit sollen einfache Denial-of-Service-Situationen behandelt werden, denn jeder verwaiste Socket benötigt 64 kByte Arbeitsspeicher, der nicht in den Swap ausgelagert werden kann.

Sobald diese Situation eintritt, protokolliert der Kernel Folgendes: TCP: too many of orphaned sockets. Dann sollten Sie die Variablen `tcp_fin_timeout` und `tcp_orphans_retries` modifizieren oder diesen Wert hochsetzen.

### **tcp\_max\_syn\_backlog**

Der Kernel hält alle halb offenen Verbindungen in einer Tabelle (SYN-Backlog) fest. Dies sind Verbindungen, bei denen der Client nach seinem SYN noch nicht das ACK-Paket gesendet hat, um die Verbindung zu bestätigen. Diese Tabelle (ist per Default 128 Einträge für Systeme mit weniger als 128 MByte und 1024 Einträge für alle anderen Systeme groß. Auf sehr beschäftigten Systemen (z.B. Webservern) sollten Sie diesen Wert erhöhen.

Siehe auch: `tcp_syncookies`

### **tcp\_max\_tw\_buckets**

Nachdem eine Applikation eine Verbindung beendet hat, wird der Socket in den Zustand Time-Wait versetzt. Diese Variable definiert, wie viele Time-Wait-Sockets insgesamt von dem Kernel unterstützt werden. Der Default-Wert von 180000 sollte nie reduziert, sondern im Fehlerfall nur erhöht werden.

### **tcp\_mem**

Diese Variable enthält drei Werte und definiert, wie der TCP-Stack seinen Speicher verwaltet. Die drei Werte sind:

- `low`: Unterer Schwellenwert.
- `pressure`: Bei Überschreiten dieses Wertes versucht TCP, Speicher zu sparen und Informationen im Speicher zu komprimieren. Das erfolgt so lange, bis der Speicherverbrauch wieder unter `low` sinkt.
- `high`: Mehr Speicher darf TCP nie verbrauchen.

Der Speicher wird nicht in Bytes, sondern in Speicherseiten gemessen. Leider ist die Größe der Speicherseite nicht ganz einheitlich bei den verschiedenen Linux-Plattformen. Die i386-Plattform verwendet meist eine 4 kByte große Speicherseite.

Modifikationen dieser Variable können große, auch positive Auswirkungen auf die Performance haben.

### **tcp\_moderate\_rcvbuf**

Diese Option wurde im Kernel 2.6.7 eingeführt. Diese Funktion wird auch als Dynamic Right Sizing (DRS) bezeichnet. Sie passt die Größe des TCP-Windows automatisch an die Laufzeit (Round-Trip-Time, RTT) der Pakete an. Sie verursacht durch sehr große TCP-Windows mit einigen Routern und Firewalls Probleme. Falls es beim Zugriff auf einige Server zum scheinbaren Totalausfall von TCP kommt, können Sie versuchen, diese Variable auf 0 zu setzen. Der Fehler liegt nicht in den Servern selbst, sondern in den Firewalls und Routern.

Damit diese Funktion aktiv ist, müssen auch `tcp_adv_win_scal` und `tcp_window_scaling` aktiv sein.

### **tcp\_no\_metrics\_save**

Der Linux-Kernel 2.4 und 2.6 hat ein senderseitiges Autotuning. Dazu speichert er den TCP-Slow-Start-Threshold-(`ssthresh`-)Wert in dem Routing-Cache für einen bestimmten Host. Dieses Verhalten wird mit dieser Variable für den Kernel 2.6 abgeschaltet. Das ist für Hochgeschwindigkeitsnetze sinnvoll, da das Autotuning hier häufig nicht korrekt funktioniert.

### **tcp\_orphan\_retries**

Diese Variable definiert, wie häufig das lokale System versucht, eine Verbindung beim Kommunikationspartner zu beenden, bevor die Verbindung lokal gelöscht wird. Diese Variable hat den Defaultwert 7, das entspricht bis zu 16 Minuten. Auf einem sehr beschäftigten Server (z.B. Webserver) ist es sinnvoll, diese Variable zu reduzieren. Die tatsächliche Zeitspanne hängt von dem Retransmission-Timeout (RTO) ab. Dieses Timeout wird dynamisch für die Verbindung bestimmt (RFC 793). Im Wesentlichen wird der Timeout von der beobachteten Laufzeit (Round Trip Time, RTT) abgeleitet.

### tcp\_reordering

Diese Variable definiert, wann ein Paket als verloren gilt. Der Default-Wert von 3 sollte nicht verkleinert werden. Siehe auch die RTO-Erklärung bei `tcp_orphan_retries`.

### tcp\_retrans\_collapse

Einige Systeme, speziell Drucker, besitzen einen Fehler in ihrem TCP/IP-Stack, der eine Kommunikation unmöglich macht. Diese Variable schaltet einen Workaround ein, so dass ein Linux-System dennoch mit diesen Geräten sprechen kann.

Es sind keine negativen Nebenwirkungen bekannt.

### tcp\_retries1

Diese Variable definiert, wie häufig der TCP-Stack ein Paket erneut senden soll, bevor er einen Fehler melden soll (Default: 3). Siehe auch die RTO-Erklärung bei `tcp_orphan_retries`.

### tcp\_retries2

Diese Variable definiert, wie häufig der TCP-Stack ein Paket versenden soll, bevor die laufende Verbindung beendet wird (Default: 15). Siehe auch die RTO-Erklärung bei `tcp_orphan_retries`.

### tcp\_rfc1337

Das RFC 1337, *TIME-WAIT Assassination Hazards in TCP*, beschreibt Probleme, die auftreten, wenn alte duplizierte Pakete neue Verbindungen beeinflussen. Dies kann geschehen, da `TIME_WAIT`-Sockets für neue Verbindungen wiederverwendet werden können. Das RFC beschreibt drei verschiedene Lösungen zu diesem Problem. Der Kernel ignoriert alle RST-Pakete an Sockets im `TIME_WAIT`-Zustand, wenn diese Variable gesetzt ist.

### tcp\_rmem

Diese Variable bestimmt die Größe des TCP-Receive-Buffers. Die Variable enthält drei verschiedene Werte:

- `min`: Garantierte Mindestgröße des Receive-Buffers für jeden Socket (Default: 4 oder 8 kByte).
- `default`: Default-Größe des Receive-Buffers (Default: 87380). Dieser Wert überschreibt den Wert `/proc/sys/net/core/rmem_default` für das TCP-Protokoll.
- `max`: Dies ist die maximale Größe des Receive-Buffers für TCP. Dieser Wert wird von `/proc/sys/net/core/rmem_max` überschrieben, falls der IPv4-Wert größer ist.

### Tipp



Wenn Sie diese Werte ändern möchten, sollten Sie sich den TCP-Tuning-Guide für Linux auf <http://www.didc.lbl.gov/TCP-tuning/linux.html> ansehen. Dort wird vorgeschlagen, die Maximalwerte zu vergrößern:

```
# increase TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

### tcp\_sack

Diese Variable aktiviert selektive Acknowledgments entsprechend dem RFC 2018, *TCP Selective Acknowledgement Options*, und dem RFC 2883, *An Extension to Selective Acknowledgement (SACK) Option for TCP*. Selektive Acknowledgments erhöhen insbesondere bei Netzwerkmedien mit Paketverlusten den Durchsatz. Da S-ACKs keine negativen Auswirkungen haben, sollte diese Option immer eingeschaltet werden.

### tcp\_stdurg

Es gibt zwei verschiedene Varianten, wie ein TCP/IP-Stack mit dem Urgent-Pointer und dem URG-TCP-Flag umgehen kann. RFC793 beschreibt das Verhalten, das auch von dem Betriebssystem BSD 4.2 verwendet wird. Dies ist das Standardverhalten des Linux-Kernels. Wenn Sie diese Variable aktivieren, verhält sich jedoch der Linux-Kernel wie in RFC 1122, *Requirements for Internet Hosts – Communication Layers*, beschrieben. Dieses Verhalten ist nicht kompatibel und kann zu Funktionsstörungen führen, daher ist die Funktion per Default nicht aktiv.

### tcp\_synack\_retries

Diese Variable definiert, wie häufig ein Kernel versuchen soll, SYN/ACK-Pakete als Antwort auf einen Verbindungsaufbau zu senden. Der Default-Wert ist 5. Da ein Versuch etwa 30-40 Sekunden dauert, beträgt der gesamte Timeout 180 Sekunden.

### tcp\_syncookies

Hiermit aktivieren Sie SYN-Cookies im Linux-Kernel. Der Linux-Kernel wird anfangen, SYN-Cookies zu versenden, sobald der SYN-Backlog überläuft. Damit können Sie sich gegen einen SYN-Flood wehren.



### Achtung

Diese Funktion ist lediglich ein Fallback für den Fall eines Angriffs. Sie kann zu großen Problemen führen. Achten Sie darauf, dass Ihr Linux-System die SYN-Cookies nicht im Normalbetrieb versendet! Wenn Ihr Server so beschäftigt ist, dass der SYN-Backlog im normalen Betrieb überläuft, sollten Sie den SYN-Backlog vergrößern (siehe `tcp_max_syn_backlog`).

Die TCP-SYN-Cookies ermöglichen es dem Linux-System, bei einem SYN-Flood ein späteres ACK-Paket eines echten Clients zu erkennen und trotzdem die Verbindung aufzubauen. Die SYN-Cookies wurden u.a. von DanBernstein entwickelt und hier beschrieben: <http://cr.yp.to/syncookies.html>.

### tcp\_syn\_retries

Diese Variable definiert, wie häufig der Linux-Kernel versuchen soll, selbst eine Verbindung aufzubauen. Der Default-Wert beträgt 5. Da ein Versuch etwa 30-40 Sekunden dauert, beträgt der gesamte Timeout 180 Sekunden.

### tcp\_timestamps

RFC 1323, *TCP Extension for High Performance*, beschreibt die Verwendung von Timestamps, um die Round Trip Time (RTT) zu ermitteln. Grundsätzlich verbessert diese Option die Funktion des TCP-Protokolls.



### Hinweis

Nmap verwendet die TCP-Timestamps, um die Uptime eines Systems zu ermitteln. Wenn Sie das verhindern möchten, müssen Sie diese Option abschalten.

### tcp\_tso\_win\_divisor

Die TCP-Segmentation-Offload-(TSO-)Funktion erlaubt es, die Aufteilung großer Pakete in kleine Pakete an die Netzwerkkarte abzugeben. Der `e1000`-Treiber unterstützt dies seit dem Kernel 2.5.33. Der TCP-Stack übergibt 64 kByte große Pakete an die Karte, die sie in 1500 Byte große Pakete entsprechend der MTU aufteilt. Dies reduziert die Prozessorlast enorm.



### Achtung

Die Linux-Kernel 2.6.11 und älter haben einen Fehler in der Implementierung. Hier sollten Sie die TSO-Funktion für die Netzwerkkarte abschalten:

```
ethtool -K eth0 tso off
```

Diese Variable definiert, welchen Anteil (Prozent) des Netzwerkkartenpuffers ein TSO-Paket einnehmen darf.

### tcp\_tw\_\*

Diese Parameter beeinflussen die Behandlung der TIME\_WAIT-Sockets. Die Variable `tcp_tw_reuse` bestimmt, ob eine Applikation einen TIME\_WAIT-Socket wiederverwenden darf. Die Variable `tcp_tw_recycle` definiert, ob eine andere Applikation einen TIME\_WAIT-Socket nutzen darf.



### Achtung

Die Verwendung dieser Optionen kann zu großen Problemen führen. Stellen Sie auf jeden Fall sicher, dass gleichzeitig auch die Variable `tcp_rfc1337` aktiviert wurde.

Sie sollten diese Optionen nur nutzen, wenn Sie nicht über genug Sockets für Ihre Applikation verfügen.

### tcp\_vegas\_\*

TCP-Vegas ist eine senderseitige Veränderung des TCP-Protokolls, die versucht, Verstopfungen durch Schätzung der Bandbreite zu ermitteln. Hierzu ermittelt es die Verzögerungen der Pakete, anstatt den Paketverlust auszuwerten (TCP-Reno). TCP-Vegas modifiziert dann die Sendegeschwindigkeit durch Modifikationen des Congestion-Fensters. TCP-Vegas erzeugt damit einen geringeren Paketverlust als TCP-Reno (Default-TCP).

Sie finden weitere Informationen auf der Vegas-Homepage (<http://www.cs.arizona.edu/protocols/>).

Anstelle von TCP-Vegas verwenden moderne Linux-Kernel jedoch bereits TCP-BIC.

### tcp\_westwood

TCP-Westwood ist eine weitere Methode, mit der Verstopfungen der Netzwerkverbindung vorgebeugt werden kann. Auch hier handelt es sich um eine nur senderseitige Modifikation des Reno-TCP-Stacks. Auch diese Methode schätzt die Bandbreite der Verbindung, um das Congestion Window und die Slow Start Threshold (ss-thresh) zu setzen.

Weitere Informationen über diese Methode finden Sie auf <http://193.204.59.123/c3lab/westwood.php> und <http://www.cs.ucla.edu/NRL/hpi/tcpw/>.

### tcp\_window\_scaling

Diese Variable schaltet das TCP-Window-Scaling an. Damit können Absender und Empfänger größere TCP-Windows als 64 kByte verwenden. Dies verbessert die Performance bei Netzwerkmedien mit hoher Bandbreite und hoher Latenz und reduziert die Verluste durch nicht ausreichende Nutzung der Bandbreite.

### tcp\_wmem

Diese Variable verwaltet ähnlich `tcp_rmem` den Sende-Puffer des TCP-Stacks. Es handelt sich genauso um drei Werte: `min`, `default` und `max`. Sie finden bei `tcp_rmem` eine Beschreibung der Werte. Eine Anpassung des `max`-Wertes nach oben kann erstaunliche Leistungssteigerungen bringen.

## 23.4 /proc/sys/net/ipv4/conf

Dieses Verzeichnis enthält für jede verfügbare Netzwerkkarte ein Unterverzeichnis. Zusätzlich existieren die Verzeichnisse `all/` und `default/`. Änderungen der Variablen in dem Verzeichnis `all/` haben Auswirkungen auf alle Netzwerkkarten. Das Verzeichnis `default/` definiert, welche Werte den Variablen neuer Netzwerkkarten zugewiesen werden. Die Werte bereits aktivierter Netzwerkkarten werden nicht verändert.

```
$ ls -l /proc/sys/net/ipv4/conf
insgesamt 0
dr-xr-xr-x 2 root root 0 2. Nov 13:36 all
dr-xr-xr-x 2 root root 0 2. Nov 13:36 default
dr-xr-xr-x 2 root root 0 2. Nov 13:36 eth0
dr-xr-xr-x 2 root root 0 2. Nov 13:36 lo
dr-xr-xr-x 2 root root 0 2. Nov 13:36 vlnet1
dr-xr-xr-x 2 root root 0 2. Nov 13:36 vlnet2
dr-xr-xr-x 2 root root 0 2. Nov 13:36 vlnet3
dr-xr-xr-x 2 root root 0 2. Nov 13:36 vlnet8
```

### 23.4.1 accept\_redirects

Diese Variable definiert, ob das System ICMP-Redirect-Nachrichten auswertet. Diese Nachrichten werden von Routern versendet, um andere Router und Rechner auf eine bessere Route hinzuweisen. Linux akzeptiert per Default ICMP-Redirects (1). Dies erlaubt aber auch ein Router-Spoofing, daher sollte es mindestens auf der Firewall ausgeschaltet werden.

#### Tipp



Wenn Sie gleichzeitig `secure_redirects` anschalten, ist die Gefahr des Router-Spoofings geringer.

### 23.4.2 accept\_source\_route

Dieser Parameter definiert, ob das System IP-Pakete mit der IP-Option Source-Routing akzeptiert. Da damit ein Angreifer Pakete über selbst gewählte Routen versenden kann, sollte diese Option abgeschaltet sein (0). Dies ist auch der Default.

### 23.4.3 arp\_announce

Diese Variable definiert, welche Source-IP-Adresse in ARP-Requests verwendet wird.

- 0: Verwendet eine beliebige IP-Adresse (Default).
- 1: Vermeidet IP-Adressen, die sich nicht in demselben Subnetz wie das Ziel der ARP-Anfrage befinden.
- 2: Verwendet die IP-Adresse, die auch für die Kommunikation mit dem Zielsystem verwendet werden würde.

### 23.4.4 arp\_filter

Diese Variable definiert, wie der Kernel auf ARP-Anfragen antwortet. Wenn Sie mehrere Netzwerkkarten für das Load-Balancing in demselben Subnetz betreiben, sollten Sie diesen Parameter auf 1 setzen.

### 23.4.5 arp\_ignore

Diese Variable definiert, wie das Linux-System auf ARP-Anfragen reagiert.

- 0: Beantworte jede ARP-Anfrage für jede lokale IP-Adresse, selbst wenn diese nicht auf der betroffenen Netzwerkkarte definiert ist (Default).

- 1: Beantworte nur ARP-Anfragen für IP-Adressen, die auf der ankommenden Netzwerkkarte definiert sind.
- 2: Beantworte nur ARP-Anfragen für IP-Adressen, die auf der ankommenden Netzwerkkarte definiert sind und die sich im selben Subnetz mit der Source-Adresse der ARP-Anfrage befinden.
- 8: Beantworte keine ARP-Anfragen.

### 23.4.6 bootp\_relay

Diese Option muss gesetzt werden, wenn Sie auf dem Linux-System einen Bootp-Relay-Daemon betreiben möchten. Die Netzwerkkarte akzeptiert dann auch Pakete mit der Source-Adresse 0.b.c.d.

### 23.4.7 disable\_policy

Dies deaktiviert IPsec-Policies für diese Netzwerkkarte.

### 23.4.8 disable\_xfrm

Dies deaktiviert IPsec-Verschlüsselung für diese Netzwerkkarte.

### 23.4.9 force\_igmp\_version

Hiermit können Sie die IGMP-Version für diese Netzwerkkarte erzwingen:

```
/sbin/sysctl -w net.ipv4.conf.eth0.force_igmp_version=3
```

### 23.4.10 forwarding

Hiermit aktivieren Sie das Forwarding nur für diese Netzwerkkarte. Wenn Sie diese Variable nutzen möchten, sollten Sie nicht die Variable `ip_forward` verwenden.

### 23.4.11 log\_martians

Diese Variable protokolliert Pakete mit unmöglichen IP-Adressen. Dies sind Pakete, die von `rp_filter` erkannt wurden oder Source-Routing-IP-Optionen verwenden.

### 23.4.12 mc\_forwarding

Wenn Sie einen Multicast-Routing-Daemon betreiben möchten, müssen Sie diese Variable einschalten.

### 23.4.13 medium\_id

Diese Variable unterscheidet Geräte anhand des Netzwerkmediums, mit dem sie verbunden sind. Diese Variable wird für ProxyARP verwendet. ProxyARP wird

von dem Kernel nur für Pakete aktiviert, die zwischen zwei Geräten weitergeleitet werden, die nicht mit demselben Medium verbunden sind. Ein Wert von -1 bedeutet, dass das Medium unbekannt ist. Der Wert 0 bedeutet, dass dieses Gerät das einzige ist, das mit dem Medium verbunden ist.

#### 23.4.14 promote\_secondaries

Wenn Sie einen Alias auf einer Netzwerkkarte definieren und anschließend die primäre IP-Adresse löschen, wird automatisch auch der Alias gelöscht. Die Option `promote_secondaries` sorgt dafür, dass der Alias automatisch zur primären IP-Adresse wird.

#### 23.4.15 proxy\_arp

Diese Variable aktiviert ProxyARP für diese Netzwerkkarte.

#### 23.4.16 rp\_filter

Diese Variable schaltet einen Reverse-Path-Filter für diese Netzwerkkarte an. Dieser Filter prüft, ob eine Antwort an den Absender des Pakets auch über diese Netzwerkkarte versendet werden würde. Diese Variable bietet so einen gewissen Schutz vor IP-Spoofing-Angriffen. Jedoch verursacht diese Funktion auch häufig Probleme, wenn Sie Policy-Routing, Advanced-Routing oder IPsec einsetzen. Dann ist es sinnvoll, diese Funktion abzuschalten. Der Schutz, den diese Funktion bietet, kann auch leicht mit Firewall-Regeln erreicht werden.

#### 23.4.17 secure\_redirects

Normalerweise akzeptiert ein Linux-System sämtliche ICMP-Redirects. Wenn Sie diese Variable aktivieren, werden nur Redirects von bekannten Default-Gateways akzeptiert. Diese Funktion erfordert zusätzlich, dass die Variable `shared_media` gesetzt ist.

#### 23.4.18 send\_redirects

Diese Variable weist das Linux-System an, ICMP-Redirects auszusenden, falls das System ein Router ist und eine bessere Route kennt.

#### 23.4.19 shared\_media

Diese Variable definiert, ob das angeschlossene Netzwerkmedium von mehreren Netzen gleichzeitig genutzt wird. Nur dann versendet der Kernel ICMP-Redirect-Nachrichten als Router und akzeptiert sie als Host.

### 23.4.20 tag

Hier können Sie eine beliebige Nummer speichern. Diese Nummer wird nur von Ihnen verwendet. Der Kernel ignoriert diese Variable.

## 23.5 /proc/sys/net/ipv4/neigh

Diese Parameter betreffen das ARP-Protokoll. Jeden Parameter hier aufzuführen, würde das Format sprengen, daher möchte ich Sie auf die `arp(7)`-Manpage verweisen, in der die Parameter erläutert werden.

## 23.6 /proc/sys/net/ipv4/netfilter

Ab der Version 2.6.9 verfügt der Linux-Kernel über das Verzeichnis, in dem sich verschiedene Variablen befinden, die das Connection Tracking betreffen. Diese Variablen werden in dem entsprechenden Kapitel besprochen (siehe Kapitel 19).

Zusätzlich kann sich in diesem Verzeichnis auch ein weiteres Unterverzeichnis `ct_sync` befinden. Die Variablen in diesem Verzeichnis werden ebenfalls in dem Kapitel zu `ct_sync` besprochen (siehe Abschnitt 26.6).

## 23.7 /proc/sys/net/ipv4/route

Diese Variablen verwalten den Routing-Cache und die Versendung von Fehler-Nachrichten.

## 23.8 /proc/sys/net/ipv6

Dieses Verzeichnis enthält die Variablen für das IPv6-Protokoll. Da diese Variablen zum größten Teil identisch mit den IPv4-Variablen sind, sollen sie hier nicht erneut aufgeführt werden.

