

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



A Postfix-E-Mail-Relay

Eine komplette Beschreibung des Postfix-Mail-Transport-Agenten (MTA) würde den Rahmen dieses Buches sprengen und soll hier auch nicht durchgeführt werden. Alternative Literatur, auch in deutscher Sprache, existiert inzwischen reichlich im Internet und auch in Buchform. Hier möchte ich Ihnen nur kurz die Installation und Konfiguration eines einfachen E-Mail-Relays vorstellen, das Sie für den Austausch der E-Mails in der DMZ nutzen können.

A.1 Postfix als E-Mail-Relay

Der Postfix-MTA besitzt zwei wesentliche Konfigurationsdateien: `/etc/postfix/master.cf` und `/etc/postfix/main.cf`. Die Konfiguration des Postfix als Relay kann damit sehr einfach durchgeführt werden. Sie müssen lediglich in der Datei `/etc/postfix/main.cf` einige Parameter anpassen.

Zunächst sollten Sie dafür sorgen, dass die Parameter `myhostname` und `mydomain` für Ihren Mailserver passen. Tragen Sie hier den Namen Ihres Postfix-Servers und der Domäne ein, für die der Postfix als Relay arbeiten soll. Der Zugriff auf den Postfix-Mailserver in der DMZ soll nicht direkt durch die Clients erfolgen, sondern Sie verwenden intern in Ihrem Netzwerk einen eigenen Mailserver. Dies kann ein beliebiges Produkt (auch ein Microsoft Exchange Server) sein. Erzeugen Sie in der Datei eine eigene Variable, `internal_mail`, und weisen Sie dieser die IP-Adresse Ihres internen Mailservers zu. Die Variable `inet_interfaces` setzen Sie entweder auf `all` oder `$myhostname`, damit Postfix auch tatsächlich über das Netzwerk E-Mails entgegennimmt. Häufig ist dieser Parameter auf Localhost beschränkt!

Damit Postfix nun E-Mails aus dem Internet für Ihre Domäne annimmt, setzen Sie den Parameter `mydestination=$mydomain`. Damit Postfix E-Mails von Ihrem internen Mailserver zur Weiterleitung in das Internet annimmt, tragen Sie Ihren internen Mailserver in der Variable `mynetworks` ein: `mynetworks = $internal_mail, 127.0.0.0/8`. Prüfen Sie, ob die Variable `smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination` gesetzt ist. Das bedeutet, dass Clients in der Gruppe `mynetworks` jeden beliebigen Empfänger in der E-Mail verwenden dürfen. Alle anderen Clients dürfen nur an die Domänen senden, für die Postfix zuständig ist (`mydestination`).

Nun müssen Sie Postfix noch sagen, wie er die E-Mails für Ihre Domäne zustellen soll. Hierzu setzen Sie den Parameter `transport_maps = hash:/etc/postfix/transport`. In

dieser Datei müssen Sie nun alle Domänen eintragen, die Postfix zum internen Mailserver weiterleiten soll. Legen Sie also die Datei mit folgendem Inhalt an:

```
example.com    smtp:[ip_internal_mail]
```

Ersetzen Sie dabei `example.com` durch Ihre Domäne und `ip_internal_mail` durch die IP-Adresse Ihres internen E-Mail-Servers. Achten Sie darauf, dass die eckigen Klammern gesetzt sind! Haben Sie die Datei erzeugt, erzeugen Sie die Hash-Datenbank mit `postmap /etc/postfix/transport`.

Das wars. Herzlichen Glückwunsch. Jetzt müssen Sie nur noch Ihren internen Mailserver so konfigurieren, dass er die E-Mails nicht direkt in das Internet zustellt, sondern an dieses Postfix-E-Mail-Relay. Ist der interne E-Mail-Server ein Sendmail-Server, so müssen Sie den Smart-Host setzen. Ist er ein Postfix-Server, heißt der Parameter `relayhost`.

A.2 Adressverifizierung

Wenn Sie das Relay so betreiben, werden Sie schnell den Spaß verlieren, da das Postfix-Relay zunächst jede E-Mail annimmt und zum internen System zustellt. Das Postfix-Relay führt keine Verifizierung der E-Mail-Adresse durch. Dadurch werden auch E-Mails an nicht existente Benutzer angenommen und nach innen weitergeleitet. Dies führt zu Fehlermeldungen auf dem internen System und einer Bounce-E-Mail. Diese Bounces werden auch als Spam angesehen, und Sie landen mit Ihrem Relay recht schnell auf der einen oder anderen Spam-Block-Liste (häufig auch Blacklist genannt).

Damit das nicht passiert, müssen Sie die Empfänger überprüfen. Postfix bietet hier zwei Möglichkeiten. Entweder stellen Sie die Empfängerliste direkt Postfix zur Verfügung, oder Sie verwenden den Verify-Server.

Um Postfix die Liste der gültigen Empfänger zur Verfügung zu stellen, benutzen Sie den Parameter `local_recipient_maps` und übergeben diesem eine Datei. Sie können auch LDAP oder SQL-Datenbanken hier einbinden. Sie finden weitere Informationen unter http://www.postfix.org/LOCAL_RECIPIENT_README.html.

Angenehmer ist die Verwendung des Verify-Servers. Diese Funktion in Postfix verzögert zunächst jede E-Mail an einen unbekanntem lokalen Empfänger. Gleichzeitig versucht Postfix eine scheinbare Zustellung an den internen Mailserver. Lässt dieser eine Zustellung an den Benutzer zu, merkt sich Postfix diesen Benutzer und nimmt auch für diesen Benutzer E-Mails an. Lässt der interne Mailserver die Zustellung für diesen Benutzer nicht zu, lehnt auch Postfix die E-Mail ab. Das Ergebnis speichert Postfix in der Address Verification Database.

Um diese Funktion zu aktivieren, müssen Sie lediglich den Parameter `smtpd_recipient_restrictions` in der Datei `/etc/postfix/main.cf` modifizieren:

```
smtpd_recipient_restrictions =  
    permit_mynetworks
```

```
reject_unauth_destination
reject_unknown_recipient_domain
reject_unverified_recipient
```

Postfix wird zunächst jeden Benutzer, der von dem internen Mailserver abgelehnt wird, nur temporär mit dem Fehler 450 ablehnen. Das bedeutet, dass die Clients später erneut eine Zustellung versuchen sollen. Wenn Sie die Protokolle eine Weile beobachtet haben und sicher sind, dass Postfix alles richtig macht, können Sie den Fehlercode ändern:

```
unverified_recipient_reject_code = 550
```

A.3 Amavisd-New

Für eine zusätzliche Filterung auf einem Postfix-Mail-Relay bietet sich Amavisd-New (<http://www.ijs.si/software/amavisd/>) an. Viele Distributionen enthalten bereits ein Amavisd-New-Paket. Für alle anderen können Sie von der Homepage ein Paket herunterladen. Installieren Sie das Paket nach der Anleitung. Je nach Ihrer Distribution müssen Sie einige Pakete der Distribution oder aus dem Internet nachinstallieren, die Amavisd-New voraussetzt. Amavisd-New kann einen Virensscan und auch eine Spam-Analyse über SpamAssassin durchführen und benötigt dafür auch die entsprechenden Pakete.

Die Filterung mit Amavisd-New erfolgt als externes Mail-Relay (Abbildung A.1). Postfix nimmt die E-Mail zunächst entgegen und prüft die DNS-Namen und Adressen. Dann übergibt Postfix per SMTP die E-Mail an Amavisd-New, das als lokaler Dienst auf dem Port 10024 läuft. Dieses zerlegt die E-Mail und scannt die E-Mail mit einem Virens Scanner und SpamAssassin. Wird ein Virus gefunden, nimmt Amavisd-New die E-Mail in Quarantäne. Ist das nicht der Fall, so übergibt Amavisd-New die E-Mail wieder per SMTP an eine zweite Postfix-Instanz auf Port 10025. Diese wird so konfiguriert, dass keine erneuten Prüfungen der E-Mail durchgeführt werden und die E-Mail direkt weitergeleitet wird.

Für die Konfiguration müssen Sie zwei Einträge in der Postfix-Konfigurationsdatei `/etc/postfix/master.cf` hinzufügen:

```
127.0.0.1:10025 inet n - y - - smtpd
  -o content_filter=
  -o local_recipient_maps=
  -o smtpd_helo_restrictions=
  -o smtpd_client_restrictions=
  -o smtpd_sender_restrictions=
  -o smtpd_recipient_restrictions=permit_mynetworks, \
reject_unauth_destination
  -o mynetworks=127.0.0.0/8

smtp-amavis unix - - y - 2 smtp
```

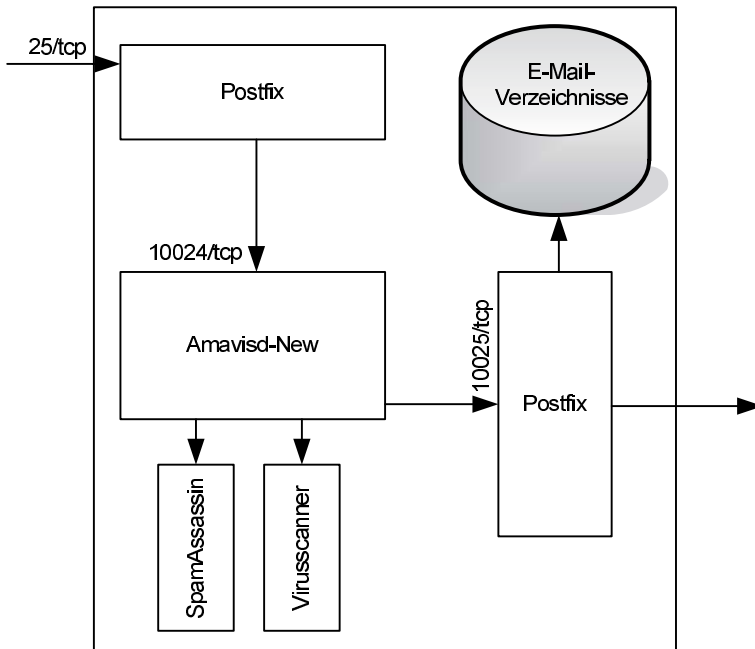


Abbildung A.1: Amavisd wird als Content-Filter über SMTP eingebunden.

```
-o smtp_data_done_timeout=1200
-o disable_dns_lookups=yes
```

Zusätzlich müssen Sie einen Eintrag in der Datei `/etc/postfix/main.cf` hinzufügen:

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Damit weisen Sie Postfix an, als Content-Filter Amavisd-New zu verwenden.

Nun können Sie Amavisd-New in der Konfigurationsdatei `/etc/amavisd.conf` konfigurieren und anpassen. Diese Datei ist sehr gut kommentiert. In den meisten Fällen entspricht die Default-Konfiguration bereits den meisten Ansprüchen.



B Firewall-Skript

In diesem Kapitel finden Sie ein Skript zum Stoppen der Firewall.

B.1 Stopp der Firewall

Dieses Skript stoppt die Firewall. Es nimmt ein Argument entgegen. Dieses Argument kann entweder `stop` oder `panic` sein. Mit `stop` setzt das Skript den Rechner in den Ausgangszustand zurück. Alle Ketten werden geleert, alle benutzerdefinierten Ketten gelöscht, alle Policies auf `ACCEPT` gesetzt und das Routing deaktiviert. Bei dem Argument `panic` handelt es sich um die gleichen Schritte mit dem Unterschied, dass anschließend alle Policies der Ketten auf `DROP` gestellt werden.

Hinweis



Dieses Skript ist auch auf der CD im Verzeichnis »Beispielskripte« vorhanden.

```
#!/bin/sh
# Dieses Skript stoppt die Firewall
#
# (c) 2005 Ralf Spenneberg
# OpenSource Training
# http://www.opensource-training.de
#
#
IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl

flush_und_loeschen() {
    # Flush
    # Check if firewall is configured (has TABLES)
    TABLES=$(cat /proc/net/ip_tables_names 2>/dev/null)
```

```
[ -z "$TABLES" ] && return 1

echo -n "$Lösche Firewall-Regeln und benutzerdefinierte Ketten: "
ret=0
# Für alle Tabellen
for i in $TABLES; do
    # Flush
    $IPTABLES -t $i -F;
    let ret+=${?};

    # Loesche Ketten
    $IPTABLES -t $i -X;
    let ret+=${?};

    # Reset Zaehler
    $IPTABLES -t $i -Z;
    let ret+=${?};
done

[ $ret -eq 0 ] && echo "OK" || echo "Fehler"
echo
return $ret
}

setze_policy() {
    # Setze die Policy auf den Übergabeparameter
    policy=$1

    TABLES=$(cat /proc/net/ip_tables_names 2>/dev/null)
    [ -z "$TABLES" ] && return 1
    echo -n "$Setze die Policy der Ketten auf $policy: "
    ret=0
    for i in $TABLES; do
        echo -n "$i "
        case "$i" in
            filter)
                $IPTABLES -t filter -P INPUT $policy \
                    && $IPTABLES -t filter -P OUTPUT $policy \
                    && $IPTABLES -t filter -P FORWARD $policy \
                    || let ret+=1
                ;;
            nat)
                $IPTABLES -t nat -P PREROUTING $policy \
                    && $IPTABLES -t nat -P POSTROUTING $policy \
                    && $IPTABLES -t nat -P OUTPUT $policy \
```

```
        || let ret+=1
        ;;
    mangle)
        $IPTABLES -t mangle -P PREROUTING $policy \
            && $IPTABLES -t mangle -P POSTROUTING $policy \
            && $IPTABLES -t mangle -P INPUT $policy \
            && $IPTABLES -t mangle -P OUTPUT $policy \
            && $IPTABLES -t mangle -P FORWARD $policy \
            || let ret+=1
        ;;
    *)
        let ret+=1
        ;;
    esac
done

[ $ret -eq 0 ] && echo " OK" || echo " Fehler"
echo
return $ret
}

case "$1" in
    stop)
        flush_und_loeschen
        setze_policy ACCEPT
        RETVAL=?
        ;;
    panic)
        flush_und_loeschen
        setze_policy DROP
        RETVAL=?
        ;;
    *)
        echo $"Usage: $0 {stop|panic}"
        exit 1
        ;;
esac

exit $RETVAL
```




C Netzwerkgrundlagen

Dieses Kapitel soll eine kurze Einführung in die im Internet verwendeten Protokolle geben. Es soll sowohl als Einführung wie auch als Wiederauffrischung und als Nachschlagewerk dienen. Die wesentlichen Punkte, um die Netzwerkprotokolle im Zusammenhang mit der Intrusion Detection zu verstehen, sollen erklärt werden. Ausführliche Darstellungen finden Sie in [29] und [11].

C.1 TCP/IP

Die TCP/IP-Protokollfamilie wird seit 1973 entwickelt. 1978 wurde die Version IPv4 fertig gestellt. Diese Version findet heute die meiste Verwendung. Ab 1982 wurde das damalige ARPANET auf das neue Protokoll IP umgestellt. Heutzutage ist TCP/IP das im Internet verwendete Protokoll. Viele Firmen haben in den vergangenen Jahren ihre Netze ebenfalls auf TCP/IP umgestellt, um so intern Internet-ähnliche Dienste anbieten zu können und eine einfache Kommunikation auch mit dem Internet zu ermöglichen.

Das OSI-Referenzmodell wird verwendet, um die Netzwerkprotokolle in sieben verschiedene Schichten aufzuteilen. Hierbei handelt es sich um die folgenden Schichten: Physical, Data-Link, Network, Transport, Session, Presentation und Application. Das TCP/IP-Protokoll wurde vor dem OSI-Modell entwickelt. Es besitzt daher nicht diese exakte Unterteilung in sieben Schichten, sondern verwendet nur vier Schichten:

1. Netzzugang: Diese Schicht entspricht im OSI-Modell der Schicht 1 und 2.
2. Internet: Diese zweite Schicht des TCP-Modells entspricht im OSI-Modell der Schicht 3. Hier arbeitet das IP-Protokoll. Daher wird häufig umgangssprachlich auch von der Schicht 3 gesprochen, wenn das IP-Protokoll gemeint ist.
3. Transport: Dies entspricht der Schicht 4 des OSI-Modells. Hier arbeiten die Protokolle TCP und UDP.
4. Anwendung: Diese Schicht bildet die Schichten 5-7 des OSI-Modells ab. Hier arbeiten die Applikationsprotokolle wie HTTP, Telnet, FTP etc.

C.2 IP

Das Internet Protocol (IP, RFC 791, STD 5) ist dafür verantwortlich, IP-Datagramme in Paketen von einer Quelle zu einem Zielrechner zu übertragen. Hierbei kümmert sich das IP-Protokoll um die Zustellung des Pakets zu diesem Zielrechner. Die revolutionäre Neuerung bei der Einführung des IP-Protokolls war die Tatsache, dass eine IP-Kommunikation keine dedizierte Verbindung (circuit switched network) mehr benötigte, sondern die Daten in einzelnen Paketen (packet switched network) unabhängig ihr Ziel erreichten (Abbildung C.1).

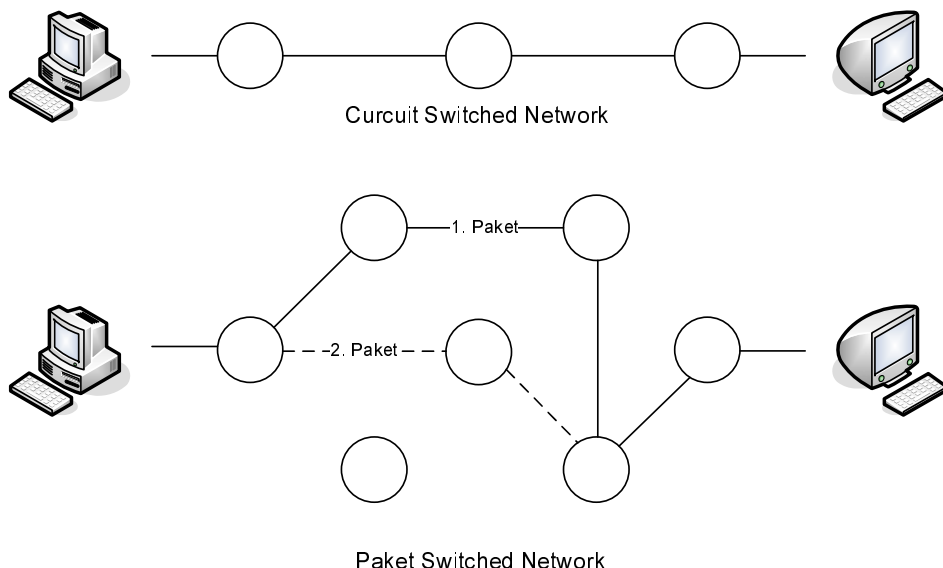


Abbildung C.1: Im Paket-Switched Network können die Pakete über unterschiedliche Wege das Ziel finden.

Dies ermöglicht es, die Kommunikation bei Ausfall redundanter Netzwerkkomponenten durch dynamische Routing-Protokolle aufrechtzuerhalten. Diese sind in der Lage, den Ausfall zu erkennen und die Daten über andere Knoten weiterhin zu übermitteln. Die hierzu benötigten Informationen werden im IP-Header des Pakets abgespeichert (Abbildung C.2). Dennoch ist IP ein Protokoll, das die Zustellung des Pakets nicht garantiert und überprüft. Es wird auch als Best-Effort-Protokoll bezeichnet. Die höheren Protokolle oder die Anwendungen müssen die erfolgreiche Übertragung prüfen, wenn dies erforderlich ist.

Die Felder und ihre Bedeutung sollen nun kurz vorgestellt werden.

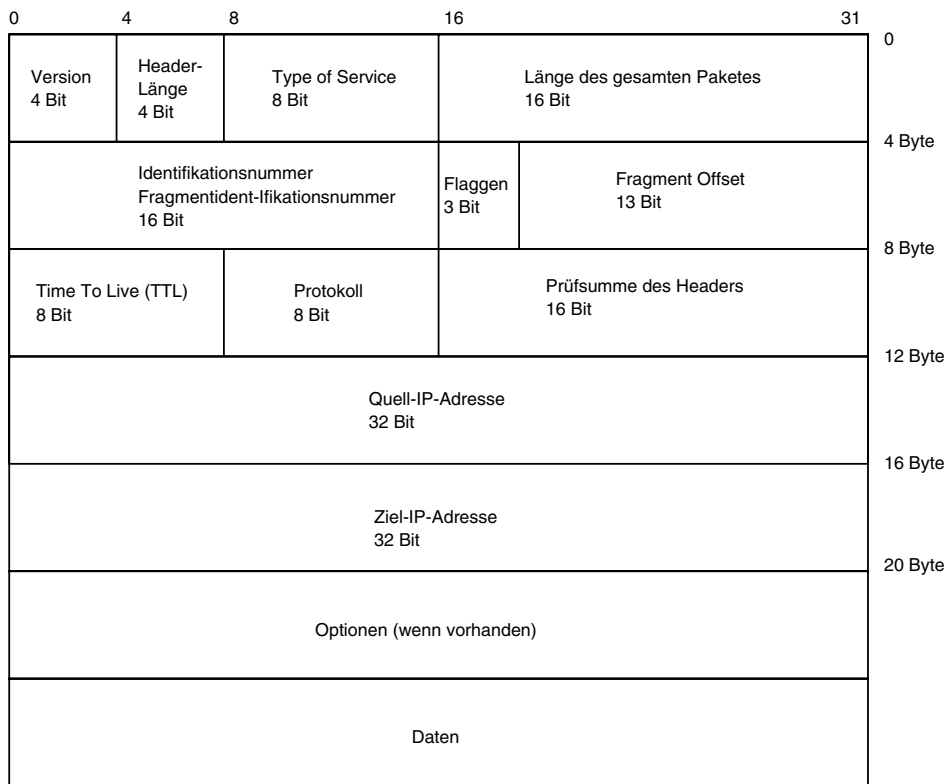


Abbildung C.2: IP-Header Version 4

C.2.1 Version

Dieses Feld ist vier Bits lang. Es enthält die IP-Version. Üblicherweise enthält dieses Feld im Moment die Zahl 4. Jedoch werden in der nahen Zukunft sicherlich vermehrt auch IPv6-Pakete auftreten. Diese enthalten dann hier die Zahl 6.

C.2.2 Header-Länge

Dieses Feld enthält die Länge des Headers. Es ist selbst 4 Bits lang. Jedoch wird die Länge nicht in Bits oder Bytes gemessen, sondern in Doppelworten. Ein Doppelwort entspricht 4 Bytes oder 32 Bits. Ein üblicher IP-Header ohne Optionen ist 20 Bytes lang. Daher befindet sich bei den meisten Paketen hier eine 5. Weist der Header weitere IP-Optionen auf (z.B. Source Routing, s.u.), so befindet sich hier entsprechend eine größere Zahl. Der Header kann maximal eine Länge von 15 (4 Bits) Doppelworten, also 60 Bytes einnehmen.

C.2.3 Type-of-Service

Das Type-of-Service-Feld ist 8 Bits lang. Es steht seit der Entwicklung des IPv4-Protokolls zur Verfügung. Es wurde ursprünglich implementiert, um eine Art Quality-of-Service zu bieten. Es wird heutzutage nur von wenigen Anwendungen gesetzt. Die Anwendungen unter Linux nutzen es jedoch recht häufig. Dennoch unterstützen nur wenige Router im Internet seine Auswertung. Eine Verwendung durch die Anwendungen hat daher nur wenig Auswirkung auf den tatsächlichen Transport des Pakets. Es existieren die folgenden Werte: Minimize-Delay 16 (0x10), Maximize-Throughput 8 (0x08), Maximize-Reliability 4 (0x04), Minimize-Cost 2 (0x02) und Normal-Service 0 (0x00). Sie werden heute abgelöst durch Diffserv und ECN (s.u.).

C.2.4 Gesamtpaketlänge

Dieses Feld definiert die Gesamtpaketlänge in Bytes. Das Feld ist 16 Bit lang, daher kann ein Paket maximal 65.535 Byte lang sein. Üblicherweise sind die übertragenen Pakete wesentlich kleiner, da die Übertragungsmedien nicht in der Lage sind, derartig große Pakete zu transportieren.

C.2.5 Identifikationsnummer

Jedes Paket wird üblicherweise mit einer eindeutigen 16 Bit langen Identifikationsnummer verschickt. Dies erfolgt, damit im Falle einer Fragmentierung der Empfänger die Fragmente eines Pakets zuordnen kann. Daher wird in der Literatur häufig bei einem nicht fragmentierten Paket von der IP-Identifikationsnummer und bei einem fragmentierten Paket von der Fragment-Identifikationsnummer gesprochen.

Der Absender inkrementiert diese Zahl üblicherweise immer um 1. Hiermit sind jedoch gespoofte Portscans möglich (siehe Abschnitt »Gespoofter Portscan« auf Seite 63). Daher existieren einige Betriebssysteme (z.B. OpenBSD), die diese Zahl zufällig vergeben. Der aktuelle Linux-Kernel 2.4 setzt diesen Wert auf null, wenn das Paket gleichzeitig das DF-Flag (s.u.) gesetzt hat. In diesem Fall darf das Paket nicht fragmentiert werden, daher hat diese Zahl auch keinen Sinn.

C.2.6 Flaggen

Der IP-Header enthält drei Bits, die Informationen über die Fragmentierung des Pakets enthalten.

Das erste dieser drei Bits wird momentan nicht verwendet und muss immer null sein.

Das folgende Bit ist das *Don't Fragment*-(DF-)Bit. Ist dieses Bit gesetzt (1), so darf das Paket nicht fragmentiert werden. Ein Router, der dieses Paket aufgrund seiner Größe nicht zustellen kann, muss das Paket verwerfen und eine ICMP-Fehlermeldung an den Absender schicken.

Das dritte und letzte Bit ist das *More Fragments follow*-(MF-)Bit. Dieses Bit zeigt an, dass weitere Fragmente folgen. Beim letzten Fragment und allen nicht fragmentierten Paketen ist dieses Bit gelöscht.

C.2.7 Fragment-Offset

Dieses Feld gibt bei einem Fragment dessen Beginn im Gesamtpaket an. Hierbei erfolgt die Angabe in Vielfachen von 8. Das bedeutet, dass ein Fragment (außer dem letzten) immer eine Länge aufweisen muss, die durch 8 ohne Rest teilbar ist. Das Feld ist 13 Bit lang und kann damit den ganzen Bereich von maximal 65.535 Bytes eines Pakets abdecken.

Der Empfänger verwendet diese Information, um das Paket in der richtigen Reihenfolge zusammenzusetzen.

Der Ping of Death nutzte Fragmente, um den TCP/IP-Stack einiger Betriebssysteme mit einem Bufferoverflow zum Absturz zu bringen. Es ist möglich, Fragmente so zu konstruieren, dass bei der Defragmentierung ein Paket größer 65.535 Bytes entsteht. Dies ist möglich, da das erste Fragment eine Größe von 1500 Bytes aufweist. Jedes weitere Fragment ist 1480 Byte lang. 1500 Bytes plus 43 mal 1480 Bytes ergeben 65.140 Bytes. Nun kann ein weiteres Fragment erzeugt werden, das erneut 1480 Byte lang ist. Erlaubt wären jedoch nur noch 395. Bei der Defragmentierung kommt es daher zum Bufferoverflow. Dieser Angriff erhielt den Namen Ping of Death, da es besonders einfach war, mit dem Kommando `ping` diese Pakete zu erzeugen.

C.2.8 Time To Live (TTL)

Bei dem Feld Time To Live handelt es sich um ein 8 Bit langes Feld. Es kann somit Werte von 0 bis 255 aufnehmen. Dieser Wert wird von jedem Router, der das Paket weiterleitet, gelesen und pro Hop um 1 dekrementiert. Erreicht hierbei das Feld den Wert null, so muss der Router das Paket verwerfen und eine Fehlermeldung an den Absender zurücksenden.

Diese Funktion erlaubt es, mit dem Werkzeug `traceroute` die Route eines Pakets zu ermitteln. Hierzu werden Pakete mit steigenden TTL-Werten an den Zielrechner gesendet. Jeder Router wird entsprechende Pakete verwerfen müssen und eine Fehlermeldung an den Absender schicken. So wird die IP-Adresse eines jeden Routers ermittelt.

Diese Funktion eignet sich jedoch auch zur Verwirrung von IDS-Systemen und zur Ermittlung von Firewall-Regeln. Hierzu werden spezielle Pakete erzeugt. Diese weisen eine TTL auf, so dass die Firewall beziehungsweise das IDS-System die Pakete sieht, sie aber nie den entsprechenden Empfänger erreichen. Bei dem *Firewalking* handelt es sich um eine ähnliche Technik.

Verschiedene Betriebssysteme verwenden üblicherweise unterschiedliche Standard-TTL-Werte. Linux verwendet 64.

C.2.9 Protokoll

IP ist in der Lage, eine große Anzahl von Protokollen zu übertragen. Dies sind zum Beispiel ICMP (1), TCP (6) und UDP (17). Dieses Feld gibt die Nummer des enthaltenen Protokolls an. Unter Linux werden alle Nummern in der Datei `/etc/protocols` aufgeführt.

Werden ungewöhnliche Protokolle im Netzwerk entdeckt, so kann es sich hierbei auch um einen Tunnel handeln. Das Honeynet Project hat ein Werkzeug entdeckt, das zum Beispiel einen NVP-(11-)Tunnel aufbaut.

C.2.10 Prüfsumme

Dies ist die Prüfsumme des IP-Headers des Pakets. Hiermit können Netzwerkgeräte und auch der Empfänger die Validität des Paket-Headers bestimmen. Die Prüfsumme enthält nicht den Datenanteil des Pakets. Da einige Werte des Headers sich während der Übertragung ändern (z.B. TTL), wird diese Prüfsumme von jedem Netzwerkgerät (z.B. Router), das Änderungen durchführt, neu berechnet. Pakete mit fehlerhaften Header-Prüfsummen können daher nur im lokalen Netzwerk erzeugt worden sein. Der Empfänger verwirft Pakete mit fehlerhaften Prüfsummen. Dies kann verwendet werden, um ein IDS-System zu verwirren, wenn dieses keine Prüfsummenermittlung durchführt.

C.2.11 Quell-IP-Adresse

Dieses Feld enthält die IP-Adresse des Absenders. Wenn beim Transport des Pakets ein Source-NAT (Network Address Translation) durchgeführt wurde, befindet sich hier die entsprechende Adresse. Ein Source-NAT ist im Header nicht erkennbar.

C.2.12 Ziel-IP-Adresse

Dieses Feld enthält die IP-Adresse des Empfängers. Wenn beim Transport des Pakets ein Destination-NAT (Network Address Translation) durchgeführt wurde, befindet sich hier die entsprechende Adresse. Ein Destination-NAT ist im Header nicht erkennbar.

C.2.13 IP-Optionen

Sämtliche für den Transport des IP-Pakets erforderlichen Informationen werden in den 20 Bytes des IP-Headers gespeichert. Jedoch besteht gelegentlich die Notwendigkeit, zusätzliche Informationen zu senden, die das Verhalten des IP-Protokolls modifizieren. Diese werden als Option übertragen. Insgesamt können 40 Bytes Optionen übertragen werden. Standardmäßig werden keine Optionen verwendet.

Die Optionen bestehen aus einem Byte, das den Typ definiert, einem Byte für die Länge und entsprechenden Bytes für die Daten. Abbildung C.3 veranschaulicht das.

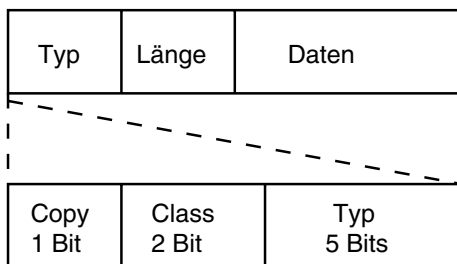


Abbildung C.3: IP-Optionen-Aufbau

End of Optionlist

Dies definiert das Ende der Optionenliste. Es ist eine Option der Klasse 0 und des Typs 0. Diese Option besitzt kein Längenfeld und kein Datenfeld.

No Operation

No Operation ist eine Option der Klasse 0 mit Typ 1. Sie wird verwendet, um die Optionenliste aufzufüllen. Die Länge des IP-Headers kann nur in Doppelworten angegeben werden. Wenn die Länge der Optionen nicht durch 4 teilbar ist, werden sie mit dieser Option aufgefüllt. Diese Option weist weder ein Längenfeld noch ein Datenfeld auf.

Security Options

Diese Option der Klasse 0 mit Typ 2 ist 88 Bits lang und wird für militärische Zwecke genutzt. Üblicherweise wird diese Option nicht im Internet beobachtet. Sie hat keine Verwandtschaft mit IPsec.

Record Route

Diese Option verwendet die Klasse 0 und den Typ 7. Router, die diese Option im Paket erkennen, sollen ihre IP-Adresse im IP-Header aufzeichnen. Da die Größe des IP-Headers beschränkt ist, können hier nur maximal acht Router ihre IP-Adressen eintragen. Daher wird diese Funktion nur selten im Internet genutzt und stattdessen auf `traceroute` zurückgegriffen. Während diese Funktion jedoch für jedes Paket tatsächlich die Route protokolliert, zeigt `traceroute` nur die wahrscheinliche Route an.

Loose Source Routing

Loose Source Routing ist eine Option der Klasse 0 und mit Typ 3. Die Größe dieser Option hängt von der Anzahl der angegebenen Router ab. Loose Source Routing erlaubt es dem Absender, eine Liste von Routern im IP-Header anzugeben, über die das Paket neben anderen transportiert wird. Es können aus Platzgründen maximal

acht Router angegeben werden. Verschiedene Befehle unterstützen diese Funktion: `tracert`, `tracert`, `netcat` etc.

Loose Source Routing erlaubt es, Pakete zu routen, die ansonsten nicht geroutet werden würden. Es erlaubt zum Beispiel, auf den meisten Internet-Routern Pakete an RFC-1918-Netze (z.B. 192.168.0.0/24) zu routen. So kann ein Angreifer von außen ein Paket an diese Adresse senden und eine Loose Source Route angeben, damit die Internet-Router auch wissen, wohin das Paket gesendet werden soll.

Strict Source Routing

Strict Source Routing ist eine Option mit der Klasse 0 und mit Typ 9. Hier definiert die Liste die exakte Abfolge der zu verwendenden Router. Es dürfen keine weiteren Router verwendet werden.

Router Alert

Diese Option (Klasse 0, Typ 20) definiert, dass der Router das Paket modifizieren muss, bevor er es weiterroutet. Es wird für experimentelle Erweiterungen genutzt.

TimeStamp

Diese Option (Klasse 2, Typ 4) verlangt, dass der Router ähnlich der Record-Route-Option seine IP-Adresse im Header ablegt, aber zusätzlich noch die Zeit hinterlegt, zu der das Paket verarbeitet wurde.

Es besteht die Möglichkeit, nur die Timestamps oder Router-IP-Adressen und Timestamps anzufordern.

C.3 UDP

Das User Datagram Protocol (UDP, RFC 768, STD 6) stellt eines der beiden am häufigsten eingesetzten Protokolle auf Basis von IP dar. Die meisten Internetapplikationen verwenden TCP (s.u.). TCP garantiert die vollständige und korrekte Übertragung der Daten. Hierzu generiert es jedoch einen gewaltigen Overhead. Viele Anwendungen benötigen diese Garantie nicht oder können sie gar nicht nutzen, da es sich um Multicast- oder Broadcast-Anwendungen handelt, bei denen ein Paket gleichzeitig an mehrere Empfänger zugestellt wird. In diesen Fällen wird meist das UDP-Protokoll verwendet.

UDP ist ein unzuverlässiges und Datagramm-orientiertes Protokoll. Es garantiert weder die Ablieferung eines Datagramms noch bietet es Vorkehrungen gegen eine Duplizierung oder eine Vertauschung der Reihenfolge der Daten. Aufgrund seiner Unzuverlässigkeit bieten die höheren Protokolle meist eine gewisse Fehlerkontrolle. Dies äußert sich oft in einer gewissen Unempfindlichkeit gegenüber verlorenen Paketen. Wenn zum Beispiel ein Paket bei einer Videostreaming-Anwendung verloren geht, so macht sich das meistens nur durch ein leichtes Zittern der Darstellung be-

merkbar. Eine TCP-ähnliche Fehlerkontrolle mit einer erneuten Sendung des Pakets würde meist zu einem Stottern und Anhalten des Streams führen.

UDP ist nur in der Lage, ein Datagramm gleichzeitig zu verarbeiten. Wenn gewisse Informationen mit UDP versendet werden, so werden diese nicht bereits von UDP sinnvoll auf einzelne Pakete aufgeteilt (TCP teilt die Daten entsprechend der MSS-Maximum Segment Size auf, s.u.), sondern es werden unter Umständen sehr große UDP-Datagramme gebaut, die anschließend von der darunter liegenden IP-Schicht auf IP-Paketfragmente aufgeteilt werden müssen.

Um mehreren Anwendungen die Verwendung des UDP-Protokolls zu ermöglichen, verwendet UDP einen Multiplexer. Client- wie Server-Anwendungen müssen sich vor der Verwendung des UDP-Protokolls registrieren. Während dieser Registrierung weist die UDP-Schicht diesen Anwendungen einen Port zu. Die Verwendung der Ports durch die verschiedenen Dienste ist grundsätzlich willkürlich, jedoch haben sich im Laufe der Jahre bestimmte Ports für bestimmte Dienste etabliert. Die Zuweisung der Ports zu den einzelnen Diensten erfolgt durch die Internet Assigned Number Authority (IANA), die die Liste der *well-known ports* pflegt (<http://www.iana.org/assignments/port-numbers>).

Der UDP-Header (Abbildung C.4) ist acht Byte lang. Er enthält den Quell- und den Zielport, die Datagrammlänge und eine Prüfsumme.

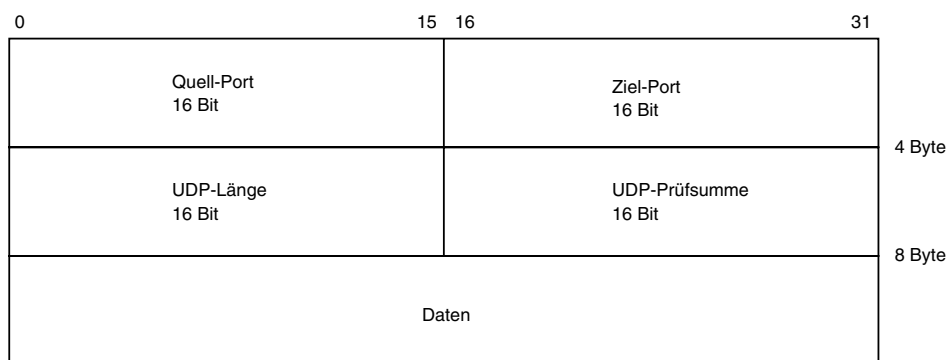


Abbildung C.4: UDP-Header

Der UDP-Header enthält keine IP-Adressen. Diese werden im IP-Header spezifiziert. Der UDP-Header stellt bei einem UDP-IP-Paket die ersten acht Bytes im Datenanteil des IP-Pakets dar.

Der UDP-Quellport (Source Port) ist wie der UDP-Zielport (Destination Port) 16 Bit oder 2 Byte lang. Die UDP-Protokollschicht verwendet diese Information, um die übertragenen Daten einzelnen Anwendungen zuzuordnen.

Die Länge des UDP-Datagramms wird ebenfalls im UDP-Header übertragen. Da ein UDP-Header mindestens acht Byte lang ist, ist die kleinste mögliche UDP-Nachricht acht Byte lang. Ein IP-Paket darf maximal 65.535 Byte lang sein. Abzüglich des IP-

Headers von mindestens 20 Bytes kann eine UDP-Nachricht maximal 65.515 Byte lang werden. Die Größe des UDP-Datagramms wird von der Anwendung bestimmt. Erzeugt diese Anwendung ein UDP-Datagramm, das größer ist als die MTU-Maximum Transmission Unit (Maximum Transmission Unit), so wird das Paket auf IP-Ebene fragmentiert.

Die UDP-Prüfsumme im UDP-Header ist optional. Das bedeutet, dass die Anwendung entscheiden kann, ob diese Prüfsumme berechnet werden soll oder nicht. Viele Anwendungen verzichten auf die Erzeugung einer Prüfsumme zugunsten der Performance. Wenn jedoch eine Prüfsumme erzeugt wurde, ist der Empfänger des Pakets laut RFC 1122 verpflichtet, diese Prüfsumme zu überprüfen und im Zweifelsfall das Paket zu verwerfen. Bei der Berechnung der Prüfsumme wird der Inhalt des UDP-Datagramms zusammen mit einem Pseudo-Header aus Quell- und Zielport, der UDP-Protokollnummer 17 und der Größe als Eingabe verwendet.

Es existieren verschiedene Anwendungen, die das UDP-Protokoll nutzen. UDP wird zum Beispiel verwendet, um Namensauflösungen durchzuführen. Hier sendet der Client ein UDP-Paket an einen DNS-Server mit der Bitte, den enthaltenen Namen aufzulösen. Der DNS-Server sendet seine Antwort in einem UDP-Paket an den Client zurück. Da UDP kein verlässliches Protokoll darstellt, achtet der DNS-Server darauf, maximal 512 Bytes Daten in seinem UDP-Datagramm zurückzusenden. Dies garantiert, dass nicht durch eine mögliche Fragmentierung des Pakets Daten verloren gehen können. Ist die zu sendende Antwort jedoch größer, so sendet der DNS-Server eine trunkierte (truncated) Antwort. Der Client ist nun verpflichtet, den DNS-Server erneut zu kontaktieren und die Anfrage erneut mit dem TCP-Protokoll zu stellen. TCP ist in der Lage, die fehlerfreie und komplette Übertragung größerer Datenmengen zu garantieren.

C.4 TCP

Das Transmission Control Protocol (TCP, RFC 793, STD 7) ist das im Internet hauptsächlich eingesetzte Protokoll. TCP garantiert die korrekte und vollständige Übertragung der Informationen. Hierzu setzt TCP unter anderem eine sehr intelligente Flussüberwachung und -steuerung ein. TCP ist ein verbindungsorientiertes Transportprotokoll für den Einsatz in paketvermittelten Netzen. Der häufigste Einsatz baut auf dem Internet-Protokoll IP auf. Es konzentriert sich auf die Verbindung und ihre Integrität. Hierzu bietet es die folgenden Dienste:

- **Virtuelle Verbindung.** Die beiden TCP-Endpunkte kommunizieren über eine dedizierte virtuelle Verbindung. Diese ist für die Flusskontrolle, die garantierte Übertragung und das I/O-Management verantwortlich.
- **I/O-Management**
 - **für die Anwendung.** TCP bietet der Anwendung einen I/O-Puffer. Die Anwendung kann ihre Informationen als fortlaufende Daten in diesen Puffer schreiben, beziehungsweise aus ihm lesen. TCP wandelt diesen fortlaufenden Strom anschließend in Pakete um. Nicht die Anwendung definiert die Paketgröße (wie bei UDP), sondern das TCP-Protokoll erzeugt die Pakete.

- **auf Netzwerkebene.** TCP wandelt den Datenstrom der Anwendung in Segmente um. Hierbei werden die Segmente so erzeugt, dass sie effizient über das Netzwerk transportiert werden.
- **Flusskontrolle.** TCP bietet eine fortgeschrittene Flusskontrolle, die die unterschiedlichen Send- und Empfangsfähigkeiten der vielfältigen Geräte berücksichtigt. Es ist in der Lage, vollkommen transparent für die Anwendung die Geschwindigkeit der Verbindung optimal anzupassen.
- **Zuverlässigkeit.** Jedes übertragene Byte wird mit einer Sequenznummer versehen. Dies ermöglicht eine Einordnung der übertragenen Daten in der richtigen Reihenfolge. Zusätzlich bestätigt der Empfänger mit dieser Nummer den Empfang der Daten. Stellt TCP fest, dass bestimmte Informationen nicht übertragen wurden, so werden sie transparent für die Anwendung automatisch erneut gesendet.

Zusätzlich bietet TCP wie UDP einen Multiplexer, so dass mehrere Anwendungen auf einem Rechner gleichzeitig das TCP-Protokoll verwenden können. Hier werden ebenfalls Ports eingesetzt. Jede Anwendung muss vor der Verwendung des Protokolls einen derartigen Port reservieren.

C.4.1 Auf- und Abbau einer TCP-Verbindung

Damit TCP die oben erwähnten Funktionen wahrnehmen kann, ist einiger Verwaltungsaufwand erforderlich. TCP muss zunächst eine Verbindung öffnen. Hierbei führen die beiden TCP-Kommunikationspartner eine Synchronisation ihrer Sequenznummern durch. Anschließend können die Daten ausgetauscht werden. Nach der Verbindung sollte diese auch wieder korrekt abgebaut werden, so dass die beiden Kommunikationspartner die Verwaltungsstrukturen wieder freigeben können.

Die Abbildung C.5 zeigt schematisch den Auf- und Abbau einer TCP-Verbindung. Diese Abbildung führt bereits in einige Funktionen von TCP ein.

Der TCP-Handshake beginnt zunächst mit einem so genannten SYN-Paket. TCP besitzt sechs Bits im TCP-Header (s.u.), die die Funktion des TCP-Pakets bestimmen. Es handelt sich hierbei um die Bits SYN, FIN, ACK, RST, URG und PSH. Ein SYN-Paket ist ein Paket, bei dem die eigene Sequenznummer an den Kommunikationspartner übermittelt wird. Dies ist für die Synchronisation der Verbindung erforderlich. Der Kommunikationspartner erhält hiermit die Information, dass die Datenzählung mit dieser Sequenznummer beginnt. Jedes übertragene Byte besitzt eine eigene Sequenznummer. In diesem Beispiel überträgt der Client an den Server die Sequenznummer 1234. Diese initiale Sequenznummer wird für jede Verbindung neu bestimmt.

Der Server beantwortet dieses Paket mit einem eigenen SYN/ACK-Paket. Das ACK- oder Acknowledge-Bit zeigt an, dass dieses Paket den Empfang von Daten bestätigt. Um der Gegenseite mitzuteilen, welche Daten bestätigt werden, übermittelt der Server eine Acknowledgement-Nummer: 1235. Diese Zahl definiert das nächste erwartete Byte von der Gegenstelle. Es entspricht also der Sequenznummer des letz-

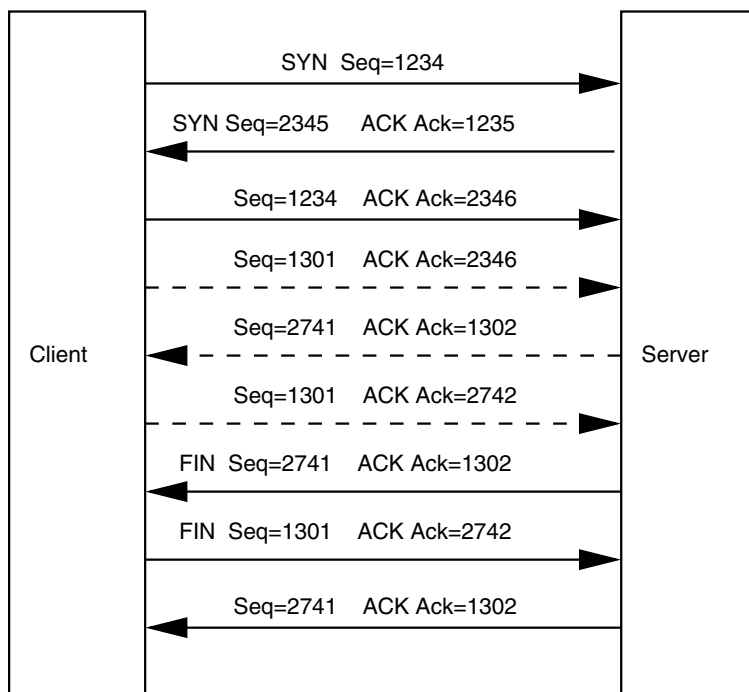


Abbildung C.5: Der TCP-Handshake

ten empfangenen Bytes plus 1. Mit dieser Bestätigung ist die Verbindung zwischen Client und Server in einer Richtung geöffnet worden. Man spricht von einer halb offenen Verbindung. In demselben Paket übermittelt jedoch der Server seinerseits seine Sequenznummer mit dem gesetzten SYN-Bit. Hiermit fordert er den Client auf, seine Seite mit dieser Nummer zu synchronisieren.

Eine Vollduplex-Verbindung, die in beide Richtungen Daten versenden kann, entsteht, wenn der Client diese Synchronisationsanfrage im dritten Paket bestätigt. Da der Client keine Daten übermittelt, erhöht er seine Sequenznummer nicht. Nun können Daten zwischen Client und Server ausgetauscht werden.

Der Client übermittelt nun zunächst 67 Bytes an den Server (4. Paket). Dies kann aus der Differenz der Sequenznummern des dritten und vierten Pakets errechnet werden. Anschließend bestätigt der Server den Empfang dieser Daten und sendet seinerseits 396 Bytes (5. Paket). Der Empfang wird erneut vom Client im 6. Paket bestätigt.

Nun beschließt der Server, dass die Verbindung beendet werden soll. Hierzu bestätigt er den letzten Empfang vom Client und setzt selbst das Bit FIN. Dieses Bit ist die Aufforderung, die Verbindung zu schließen. Hierbei werden keine Daten übermittelt. Daher wird die Sequenznummer nicht erhöht. Der Client bestätigt das FIN.

Hiermit ist die Verbindung halb geschlossen. Um nun seine Richtung zu schließen, sendet der Client in demselben oder einem zusätzlichen Paket ebenfalls ein FIN an den Server. Dieser bestätigt dieses FIN ebenfalls in einem letzten Paket, und die Verbindung ist in beiden Richtungen geschlossen.

C.4.2 TCP-Header

Das TCP-Segment besteht ähnlich wie ein UDP-Datagramm aus einem Header und den Daten. Bevor nun das weitere Verhalten von TCP bezüglich der Flusskontrolle und der Zuverlässigkeit besprochen wird, soll kurz der TCP-Header mit seinen Informationen vorgestellt werden.

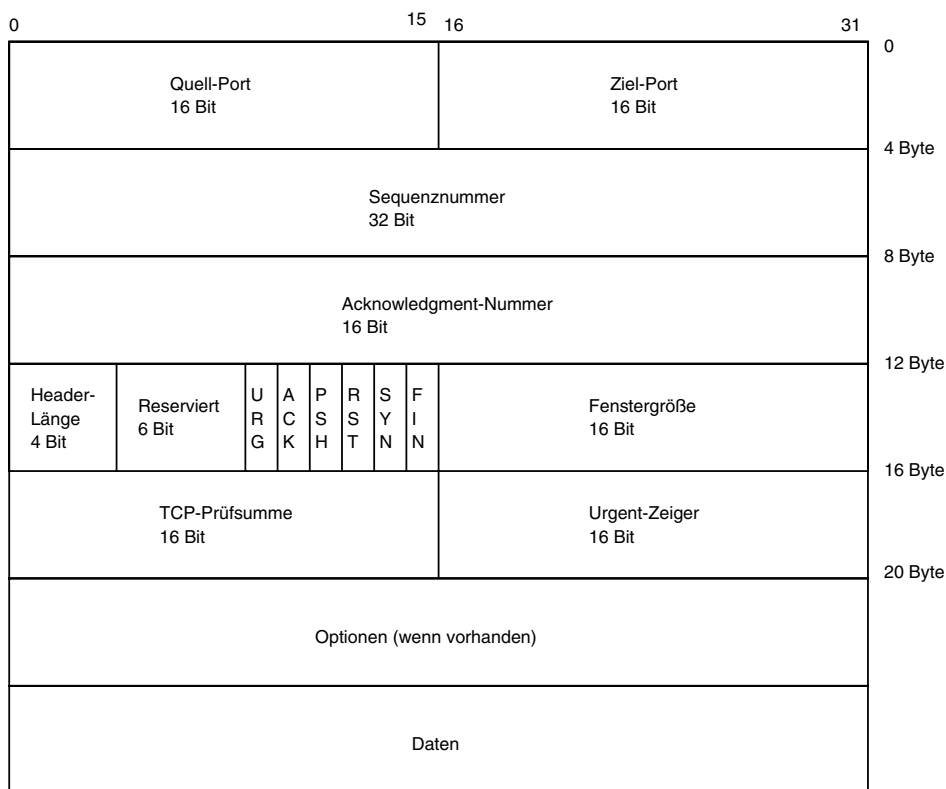


Abbildung C.6: TCP-Header

Quell- und Zielport

Die TCP-Ports werden vom TCP-Protokoll verwendet, um Multiplexer-Funktionalität zur Verfügung zu stellen. Das TCP-Protokoll ist hiermit in der Lage, mehrere Anwendungen gleichzeitig zu unterstützen. Hierzu bindet sich eine Anwendung

zunächst an einen Port. Anschließend kann das TCP-Protokoll dieser Anwendung spezifisch ihre Daten zukommen lassen.

Das TCP-Protokoll unterstützt 65.536 Ports von 0 bis 65.535. Die Zuordnung der Ports zu den Applikationen ist grundsätzlich willkürlich. Jedoch wurden von der IANA einige Ports gewissen Diensten fest zugewiesen (s. UDP).

Sequenznummer

TCP garantiert die Zustellung der zu übertragenden Daten. Damit die Kommunikationspartner diese Übertragung bestätigen und die Daten in der richtigen Reihenfolge zusammensetzen können, wird jedes übertragene Byte mit einer Sequenznummer versehen. Die Zuordnung der Daten und die Entscheidung über die Annahme der Daten erfolgt über diese Sequenznummer (s.u.). Die initiale Sequenznummer sollte für jede Verbindung zufällig ermittelt werden. Besteht die Möglichkeit, die Sequenznummer von einer Verbindung zur nächsten vorherzusehen, so kann dies für gespoofte Angriffe auf das TCP-Protokoll genutzt werden (siehe auch Mitnick-Angriff).

Die Sequenznummer ist eine 32-Bit-Zahl.

Acknowledgement-Nummer

Wie bereits bei der Sequenznummer angesprochen, erhält jedes übertragene Byte eine eindeutige Sequenznummer. In dem Maße, in dem die Daten empfangen werden, übermittelt der Empfänger die Information an den Absender, dass er in der Lage ist, weitere Daten zu empfangen. Die Acknowledgement-Nummer enthält die Information, welches Byte als nächstes erwartet wird. Alle Bytes mit einer kleineren Sequenznummer wurden bereits empfangen.

Sendet der Absender vier Pakete mit den Bytes 1-100, 101-200, 201-300 und 301-400, so antwortet der Empfänger entsprechend mit einer Acknowledgement-Nummer von 101, 201, 301 und 401. Geht das zweite Paket während der Übertragung verloren, so antwortet der Empfänger mit 101, 101, 101 und 101. Dies weist den Absender darauf hin, dass das zweite Paket erneut gesendet werden muss. Sendet der Absender anschließend das Paket 101-200 erneut, so bestätigt der Empfänger dies mit 401, da er die Daten 201-400 ebenfalls bereits erhalten hat.

Um dies noch zusätzlich zu optimieren, besteht die Möglichkeit, verzögerte oder selektive Bestätigungen des Empfangs zu versenden (s.u.).

Header-Länge

Dieses vier Bit lange Feld gibt die Länge des Headers in Doppelworten an. Die Länge in Bytes lässt sich aus dem Wert dieses Feld nach Multiplikation mit 4 ermitteln. Ein TCP-Header ist immer mindestens 20 Byte lang. Dieses Feld trägt also mindestens die Zahl 5. Maximal kann ein TCP-Header 60 Byte lang sein (15 mal 4). TCP definiert also nicht die Gesamtlänge des Pakets wie UDP, sondern nur die

Länge des Headers. Die Gesamtlänge des Pakets lässt sich jedoch aus der Gesamtlänge des IP-Pakets minus der Länge des TCP-Headers ermitteln.

Reservierte Bits

Hierbei handelt es sich um sechs Bits, die bis vor kurzem keine weitere Verwendung hatten. Diese Bits mussten daher bisher gelöscht sein. Viele Firewalls und Intrusion-Detection-Systeme lösen bei einer Verwendung dieser Bits einen Alarm aus. Viele Werkzeuge zur Erkennung von Betriebssystemen nutzen diese Bits, um entfernte Systeme zu untersuchen. Unterschiedliche Betriebssysteme reagieren meist unterschiedlich, wenn diese Bits gesetzt sind.

Seit einigen Jahren existieren jedoch Bemühungen, einige dieser Bits für die Explicit Congestion Notification (ECN) zu verwenden (RFC 3168). Hierbei handelt es sich um einen Mechanismus, bei dem die Kommunikationspartner eine Verstopfung des Internets im Vorfeld erkennen und die Übertragungsrate automatisch anpassen, um eine echte Verstopfung zu vermeiden.

Da ECN sowohl den IP- als auch den TCP-Header betrifft, wird es in einem eigenen Abschnitt behandelt (C.5).

TCP-Flags

Bei den TCP-Flags handelt es sich um sechs Bits. Diese Bits klassifizieren die übertragenen Daten. Es sind nur sehr wenige Kombinationen dieser Bits in einem gültigen Paket erlaubt. Sie haben die folgenden Funktionen:

- **URG.** Das URG-Bit (Urgent) wird verwendet, um der Gegenstelle mitzuteilen, dass das Paket wichtige Daten enthält, die sofort verarbeitet werden müssen. Um den Anteil der wichtigen Daten zu definieren, wird der Urgent-Zeiger verwendet (s.u.). Ist das URG-Bit nicht gesetzt, so ist der Wert des Zeigers zu ignorieren.
- **ACK.** Das ACK-Bit (Acknowledgement) wird verwendet, um den Empfang von Daten zu bestätigen. Jedes TCP-Segment einer Verbindung, außer dem ersten Segment und RST-Segmenten zum Abbruch einer Verbindung, muss dieses Bit gesetzt haben.
- **PSH.** Das PSH-Bit (Push) kennzeichnet Segmente, die von der sendenden Anwendung »gedrückt« werden. Häufig müssen Anwendungen (z.B. telnet) nur sehr wenige Daten versenden. TCP würde zur Optimierung die Daten aber erst versenden, wenn ein komplettes Segment gefüllt wäre. Das PSH-Bit weist das TCP-Protokoll an, die Daten sofort zu versenden, da keine weiteren Daten in diesem Moment folgen.
- **RST.** Das RST-Bit (Reset) wird nur verwendet, wenn ein Fehler in einer Verbindung aufgetreten ist oder wenn der Wunsch eines Verbindungsaufbaus abgelehnt wird.
- **SYN.** Das SYN-Bit (Synchronize) wird verwendet, um die eigene Sequenznummer zur Synchronisation an den Kommunikationspartner zu übermitteln. Dies

erfolgt während des TCP-Handshakes (s.o.). Das SYN-Bit darf nur in diesem Moment gesetzt werden.

- **FIN.** Das FIN-Bit (Finish) wird gesetzt, wenn eine Seite der Kommunikation diese beenden will. Wenn die Gegenseite ebenfalls in der Lage ist, die Verbindung zu beenden, so antwortet sie ebenfalls mit einem FIN-Paket. Ein Paket mit gesetztem FIN-Bit ist nur in einer zuvor aufgebauten Verbindung gültig.

Grundsätzlich müssen alle TCP-Pakete das ACK-Bit gesetzt haben. Lediglich zwei Ausnahmen sind erlaubt. Hierbei handelt es sich um das erste Paket einer TCP-Verbindung, in dem lediglich das SYN-Bit gesetzt ist, und ein RST-Paket, das einen Fehler in einer TCP-Verbindung anzeigt und diese abbricht. Die weiteren Bits können in Kombinationen mit dem ACK-Bit vorkommen.

Kombinationen von SYN/FIN, SYN/RST oder RST/FIN sind jedoch nicht erlaubt. Netzwerk-Intrusion-Detection-Systeme ermitteln üblicherweise sämtliche Pakete mit fehlerhaften TCP-Flag-Kombinationen. Der *Unclean-Match* des Linux-Paketfilters Netfilter ist ebenfalls in der Lage, diese Kombinationen zu erkennen und sogar zu verwerfen (Achtung: Ältere Implementierungen von *Unclean* wiesen hier Fehler auf).

Fenstergröße

Das TCP-Fenster (Receive Window) gibt die Größe des Empfangsspeichers des sendenden Systems an. Diese Angabe wird von der TCP-Flusskontrolle verwendet. Die Größe dieses Feldes ist 16 Bit. Damit kann das Fenster maximal 64 Kbit groß werden. Da dies für einige Netzwerke (z.B. Token Ring) nicht ausreicht, existiert zusätzlich die Möglichkeit, die Window Scale-Option zu verwenden. Hiermit können 30 Bits für die Angabe der Fenstergröße verwendet werden. Dies erlaubt Fenster bis zu 1 Gbyte Größe.

Wurden so viele Daten gesendet, dass dieses Empfangsfenster gefüllt ist, so muss der Sender zunächst auf eine Bestätigung dieser Daten warten, bevor weitere Daten gesendet werden dürfen.

TCP-Prüfsumme

Dieses Feld speichert die TCP-Prüfsumme. Diese Prüfsumme erstreckt sich sowohl auf den Header als auch auf die Daten des TCP-Segments. Zusätzlich wird ein Pseudo-Header mit aufgenommen, der die IP-Adressen und das Protokoll (TCP, 6) enthält.

Die TCP-Prüfsumme ist 16 Bit lang und nicht optional. Der Absender muss die TCP-Prüfsumme berechnen, und der Empfänger muss diese Prüfsumme kontrollieren. Wenn die Prüfsumme nicht gültig ist, wird das Paket verworfen. Der Absender erhält keinerlei Fehlermeldung.

Diese Eigenschaft wird sehr häufig verwendet, um Network-Intrusion-Detection-Systeme oder zustandsorientierte Paketfilter zu verwirren. Wenn diese Systeme ver-

suchen, den Zustand der Verbindung zu überwachen, aber nicht die Prüfsumme testen, so besteht die Möglichkeit, weitere Pakete (z.B. RST-Pakete) einzuschleusen, die vom NIDS oder vom Paketfilter als gültige Pakete akzeptiert werden (und scheinbar die Verbindung abbrechen), aber vom echten Empfänger verworfen werden.

Urgent-Zeiger

TCP ist in der Lage, bestimmte Teile der Nachricht als wichtig zu kennzeichnen. Hierzu wird das URG-Flag in den TCP-Flags gesetzt. Zusätzlich wird der Urgent-Zeiger gesetzt. Der Urgent-Zeiger (oder Pointer) zeigt auf das Ende der wichtigen Informationen im aktuellen Segment.

Pakete, die das URG-Bit gesetzt haben, müssen vom Empfänger sofort bearbeitet werden. Dieses Paket sollte Vorrang vor allen weiteren Paketen in der Empfangswarteschlange haben.

Wenn der Urgent-Zeiger gesetzt, das URG-Bit jedoch gelöscht ist, muss das Paket wie ein normales Paket behandelt werden.

Optionen

Die bisher im TCP-Header spezifizierten Angaben genügen für eine erfolgreiche TCP-Verbindung. Jedoch werden häufig zusätzliche Optionen genutzt, um eine Anpassung der Eigenschaften zu ermöglichen.

TCP unterstützt die folgenden neun Optionen:

- **End of Option List.** Diese Option markiert das Ende der Optionen im TCP-Header. Die Option ist acht Bit lang und hat den Typ 0.
- **No Operation.** Diese Option wird verwendet, um Bereiche zwischen den TCP-Optionen aufzufüllen. Es ist sinnvoll, dass einige Optionen auf einer 32-Bit-Grenze beginnen. Dann wird der Bereich zwischen diesen Optionen mit NOP aufgefüllt. NOP ist acht Bit lang und vom Typ 1.
- **Maximum Segment Size.** Die Maximum Segment Size (MSS) wird von den Endpunkten verwendet, um die Gegenseite über ihre MTU/Maximum Transmission Unit (Maximum Transmission Unit) beziehungsweise MRU (Maximum Receive Unit) zu informieren. Diese Funktion kann auch von Routern gesetzt werden. Netfilter besitzt zum Beispiel eine TCPMSS-Funktion. Dieser Austausch erfolgt lediglich bei der Synchronisation der Verbindung. Die MSS ist üblicherweise gleich der MTU minus 40 Bytes.

Es handelt sich um eine Option von 32 Bit Länge und Typ 2. Wird keine MSS angegeben, so verlangt RFC 1122, dass als MSS 536 (IP-Default-Größe 576 - Header 40) verwendet wird.

- **Window Scale.** Hiermit ist es möglich, größere Empfangsfenster anzugeben als die üblichen 64 Kbyte. Maximal sind Fenster in der Größenordnung von 1 Gbyte möglich. Diese Option von Typ 3 ist drei Byte lang. Das dritte Byte definiert

den Maßstab des im Header angegebenen Empfangsfensters. Die Window Scale-Option darf lediglich in den beiden ersten Paketen ausgetauscht werden. Ansonsten wird sie ignoriert.

- **Selective Acknowledgment Permitted.** Bevor selektive Bestätigungen erlaubt sind, müssen beide Endpunkte sich darauf einigen. Die Option *Selective Acknowledgment Permitted* vom Typ 4 ist 16 Bit lang. Sie muss ebenfalls in den ersten beiden TCP-Segmenten ausgetauscht werden. Später wird sie ignoriert.
- **Selective Acknowledgment Data.** Hiermit besteht für den Empfänger die Möglichkeit, einen unterbrochenen Strom von Daten zu bestätigen. Normalerweise kann der Empfänger nur das letzte Byte eines ununterbrochenen Datenstroms bestätigen. Dass der Empfänger bereits weitere Pakete erhalten hat, kann er dem Absender nicht mitteilen. Mit dieser Option *Selective Acknowledgment Data* vom Typ 5 und einer variablen Länge ist der Empfänger in der Lage, exakt ein fehlendes Segment nachzufordern und die bereits empfangenen diskontinuierlichen Segmente dem Absender mitzuteilen.

Diese Option darf in jedem TCP-Segment verwendet werden.

- **Timestamp.** Die Option *Timestamp* vom Typ 8 erlaubt den beiden Endpunkten, die Latenzzeit kontinuierlich zu messen. Diese Option mit einer Länge von zehn Byte enthält die Zeitstempel beider Endpunkte in jedem Paket.

Diese Option darf in jedem TCP-Segment verwendet werden.

C.4.3 Fortgeschrittene Eigenschaften von TCP

Flussskontrolle

Wenn eine Anwendung unter Verwendung des TCP-Protokolls Daten versendet, so schreibt sie die Daten in einen Sendepuffer. TCP wird in regelmäßigen Abständen die Daten dieses Sendepuffers in TCP-Segmenten versenden. Die Senderate wird hierbei ständig angepasst.

Ursprünglich wurde hierzu die Möglichkeit geschaffen, dass der Empfänger die Geschwindigkeit über sein Empfangsfenster (Receive Window Sizing) anpasst. Dazu übermittelt der Empfänger in jedem Paket die maximale Datenmenge, die er im Moment zu verarbeiten in der Lage ist. Der Sender darf nicht mehr Daten als die vorgeschriebene Menge übertragen. Anschließend muss der Sender zunächst auf eine Bestätigung des Empfangs und der Verarbeitung warten.

Das Empfangsschiebefenster (Sliding Receive Windows) erlaubt es dann dennoch dem Sender, einige weitere Pakete zu versenden, da er eine Bestätigung der bereits gesendeten Pakete in Kürze erwartet. Hierdurch ist ein wesentlich reibungsärmerer Austausch der Daten möglich.

Dies stellt jedoch eine rein vom Empfänger gesteuerte Flusskontrolle dar. Benötigt der Empfänger mehr Zeit zur Verarbeitung der Daten, so kann er sein Empfangsfenster verkleinern. Ist er in der Lage, die Daten schnell zu verarbeiten, so kann er es derart vergrößern, dass die Geschwindigkeit nur noch durch das Netzwerk selbst

gesteuert wird. Dies reicht jedoch nicht aus. Es ist auch eine durch den Sender gesteuerte Flusskontrolle sinnvoll.

Die vom Sender gesteuerte Flusskontrolle verwendet ein Congestion Window (Verstopfungsfenster), den langsamen Start (Slow Start) und die Verstopfungsvermeidung (Congestion Avoidance).

Lediglich der Sender ist in der Lage, Verstopfungen zu erkennen. Hierzu existieren drei Möglichkeiten:

1. Der Sender erhält eine ICMP-Source-Quench-Meldung eines Routers. Dies zeigt an, dass der Router nicht in der Lage ist, die Pakete schnell genug zu verarbeiten.
2. Der Sender erhält mehrfache Acknowledgements mit identischer Acknowledgment-Nummer. Man bezeichnet diese auch als doppelte Acknowledgements. Der Empfänger sendet ein Acknowledgement, wenn er ein weiteres Paket erhält. Wenn ihm jedoch ein Paket fehlt, so weisen alle diese Pakete dieselbe Acknowledgement-Nummer auf. Die Acknowledgement-Nummer zeigt das nächste vom Empfänger erwartete Datenbyte an! Daher sind für den Sender doppelte Acknowledgements ein Hinweis darauf, dass wahrscheinlich ein Paket zwischendurch verloren gegangen ist.
3. Der Sender erhält innerhalb einer bestimmten Zeit kein Acknowledgement (Acknowledgment Timer). Dies weist ebenfalls auf verloren gegangene Pakete hin.

Wenn nun der Sender weiterhin die Pakete mit gleicher Geschwindigkeit sendet, wird die Verstopfung bestehen bleiben und möglicherweise schlimmer werden. Es ist also erforderlich, dass der Sender reagiert und die Rate senkt, um die Verstopfung zu beheben und schließlich den alten Paketdurchsatz wieder zu erreichen.

Hierzu wird ein Congestion Window verwendet. Dieses definiert, wie viele Daten der Sender ohne eine Bestätigung der Gegenseite versenden darf. Zu Beginn weist dieses Fenster die gleiche Größe auf wie das Empfangsfenster (Receive Window). Dieses Fenster wird nun in Abhängigkeit von der Verstopfung reduziert.

- Wurden mehr als drei doppelte Acknowledgements erkannt, so wird das Congestion Window halbiert. Anschließend wird die Congestion Avoidance aktiviert. Diese vergrößert das Fenster wieder in sehr kleinen Schritten.
- Wurde eine Source-Quench-Meldung erhalten oder fehlen Acknowledgements, so wird das Fenster so stark reduziert, dass jeweils nur ein Segment gesendet werden kann. Anschließend wird der Slow Start aktiviert.

Der Slow Start vergrößert das Congestion Window exponentiell. Würde das Congestion Window sofort wieder auf den Ausgangswert reinitialisiert, so würde die Verstopfung sofort wieder auftreten. Beim Slow Start wird das Congestion Window **für jedes** bestätigte Segment um **ein** weiteres Segment vergrößert. Diese Technik wird verwendet, wenn eine neue Verbindung aufgebaut wird und eine Verstop-

fung aufgetreten ist. Im Falle einer Verstopfung wird jedoch beim Erreichen des halb maximalen Congestion Windows auf Congestion Avoidance umgeschaltet.

Congestion Avoidance stellt eine langsamere vorsichtigere Methode zur Vergrößerung des Congestion Windows dar. Wurden **alle** Pakete, die innerhalb eines Congestion Windows versandt wurden, bestätigt, so wird das Congestion Window um **ein** Segment vergrößert.

Zuverlässigkeit

Die Zuverlässigkeit ist eine der wichtigsten Eigenschaften des TCP-Protokolls. Daher soll hier kurz beschrieben werden, welche Mechanismen von TCP verwendet werden, um dies effizient garantieren zu können.

RFC 793 verlangt, dass TCP in der Lage ist, Daten, die beschädigt, verloren, dupliziert oder in falscher Reihenfolge übermittelt wurden, so zu handhaben, dass dies transparent für die Anwendung erfolgt. Um dieses Ziel zu erreichen, verwendet TCP Prüfsummen, Sequenznummern, Acknowledgement-Nummern und Zeitgeber.

Die TCP-Prüfsummen ähneln den UDP-Prüfsummen. Im Gegensatz zu UDP sind sie bei TCP jedoch obligatorisch. Hierbei werden zusätzlich zum TCP-Header und zu Daten die IP-Adressen und das IP-Protokoll zur Ermittlung der Prüfsumme verwendet. Diese Prüfsumme garantiert die fehlerfreie Übertragung der Daten.

Die Sequenznummern ermöglichen es dem Empfänger, die Daten in der richtigen Reihenfolge zu verarbeiten, auch wenn die Daten möglicherweise in einer anderen Reihenfolge erhalten wurden. Jedes übertragene Byte besitzt seine eigene eindeutige Sequenznummer. Diese Sequenznummern erlauben auch die Erkennung duplizierter Informationen, da diese identische Sequenznummern aufweisen. Um eine Störung durch veraltete Pakete beim Empfang zu vermeiden, sollte der Empfänger nur Pakete mit Sequenznummern verarbeiten, die seinem Empfangsfenster (Receive Window) entsprechen.

Die Acknowledgement-Nummern erlauben es dem Absender, den korrekten Empfang der gesendeten Daten zu prüfen. Der Empfänger bestätigt den Empfang der Daten und bestätigt, dass er in der Lage ist, weitere Daten zu verarbeiten. Wurden gesendete Daten nicht mit der entsprechenden Acknowledgement-Nummer bestätigt, so werden diese Daten nach Ablauf des entsprechenden Zeitgebers erneut versendet.

Da häufig nur wenige Pakete verloren gehen, wurde mit dem RFC 1072 die Möglichkeit geschaffen, selektive Bestätigungen (Selective Acknowledgements) zu versenden. Meist hat der Empfänger bereits zehn Segmente erhalten, jedoch fehlt das erste Segment. Ein klassisches Verhalten des Empfängers, bei dem dieser mehrfach nur dieselbe Acknowledgement-Nummer versendet, führt häufig dazu, dass sämtliche Pakete erneut versendet werden. Dies ist jedoch nicht erforderlich und beeinträchtigt die Bandbreite. Die Selective Acknowledgements erlauben es mithilfe der TCP-Option *Selective Acknowledgement*, trotz doppelter Acknowledgement-Nummern weitere Segmente selektiv zu bestätigen.

Wenn der Datendurchsatz sehr hoch ist, ist es sinnvoller, nicht jedes Paket zu bestätigen, sondern die Bestätigung verzögert zu versenden (Delayed Acknowledgements). Hierbei wird nur der Empfang jedes zweiten oder dritten Pakets bestätigt. Dies stellt kein Problem dar, da ein Acknowledgement bedeutet, dass alle Daten bis zu dieser Acknowledgement-Nummer empfangen wurden.

C.5 Explicit Congestion Notification

TCP ist in der Lage, bereits sehr gut mit einer Netzwerkverstopfung umzugehen und trotz Verstopfung die Datenübertragung zuverlässig zu garantieren. Jedoch kann es weiterhin zu einer Verstopfung und damit auch zu einer Verzögerung der Übertragung kommen. Die Explicit Congestion Notification (ECN) versucht nun, dies zu verhindern, indem die Kommunikationspartner bereits vor dem Auftreten der Verstopfung gewarnt werden und dem Entstehen entgegenwirken können.

Die Erweiterung des IP-Protokolls um die Explicit Congestion Notification wird in RFC 3168 beschrieben. Linux ist eines der ersten Betriebssysteme, die dieses RFC umgesetzt haben. Es beschreibt die notwendigen Modifikationen des IP- und des TCP-Protokolls zur Umsetzung von ECN.

Die bisher besprochenen Verfahren des TCP-Protokolls zur Vermeidung einer Netzwerkverstopfung gehen davon aus, dass es sich beim Netzwerk um eine Black Box handelt. Das Netzwerk selbst ist nicht in der Lage, eine Verstopfung anzuzeigen. Die Verstopfung zeichnet sich dadurch aus, dass keine Pakete mehr transportiert werden.

Moderne Netzwerke und ihre Komponenten sind jedoch wesentlich intelligenter und sehr wohl in der Lage, eine Verstopfung bereits in ihrem Entstehen zu erkennen. Ein Router ist bei aktiver Verwaltung (Random Early Detection, RED) seiner Warteschlangen in der Lage, eine Verstopfung zu erkennen, bevor die Warteschlange überläuft und der Router die Pakete verwerfen muss. Er ist in der Lage, diese Informationen an die Endpunkte einer Kommunikation zu übermitteln, wenn die Protokolle dies vorsehen und verstehen. Dies erfolgt durch die Angabe *Congestion Experienced (CE)*.

Damit dies möglich ist, müssen das IP-Protokoll und das Transport-Protokoll (TCP) dies auch unterstützen. Hierzu werden im IP-Header ein ECN-Feld und in dem TCP-Header zwei neue ECN-Flags definiert. Diese Definition erlaubt eine fließende Migration, da diese Felder von Systemen, die nicht ECN-fähig sind, ignoriert werden. Leider werden Pakete, die diese Felder verwenden, von vielen Firewalls und Network-Intrusion-Detection-Systemen noch als gefährlich eingestuft.

Das ECN-Feld im IP-Header ist zwei Bits lang. Ist dieses Feld gelöscht, so ist der Absender des Pakets nicht ECN-fähig. Tragen die beiden Bits den Wert 01 oder 10, so ist der Absender des Pakets ECN-fähig. Trägt dieses Feld den Wert 11, so hat ein Router dies gesetzt, da er eine Congestion bemerkt hat: Congestion Experienced. Das ECN-Feld entspricht den bisher ungenutzten Bits 6 und 7 des Type-Of-Service-

Feldes. Die ehemalige Verwendung des Type-Of-Service-Feldes wird nun durch die Differentiated Services ersetzt. Diese nutzen die Bits 0 bis 5 dieses Feldes.

Erhält ein Endpunkt ein Paket, bei dem *Congestion Experienced (CE)* gesetzt ist, so muss dieser Endpunkt sich so verhalten, als ob das Paket verloren gegangen wäre. Im Falle von TCP muss das TCP-Protokoll das Congestion Window halbieren.

ECN benötigt eine Unterstützung durch das Transport-Protokoll. Dieses muss zunächst die ECN-Fähigkeit der Endpunkte aushandeln. Anschließend sollten die Endpunkte jeweils Informationen über CE-Pakete austauschen.

Insgesamt sind drei Funktionalitäten für ECN in TCP erforderlich. Hierbei handelt es sich um die Aushandlung der ECN-Fähigkeit zwischen den beiden Endpunkten. Zusätzlich ist ein ECN-Echo erforderlich, mit dem der Empfänger eines CE-Pakets dem Absender diese Tatsache mitteilt, und ein Congestion Window Reduced – (CWR-)Flag, mit dem der Absender dem Empfänger mitteilt, dass das Congestion Window reduziert wurde.

Diese Fähigkeiten werden in TCP mit zwei neuen Flags realisiert. Hierbei handelt es sich um zwei bisher reservierte Bits im TCP-Header. Das Bit 9 der Bytes 13 und 14 im TCP-Header ist das ECN-Echo-Bit (ECE). Das CWR-Bit ist das Bit 8 im TCP-Header. Abbildung C.7 zeigt den veränderten Header.

Header-Länge 4 Bit	Reserviert 4 Bit	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N
-----------------------	---------------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------

Abbildung C.7: ECN-Header

Während des Verbindungsaufbaus sendet ein ECN-fähiger Rechner nun ein TCP-SYN-Paket, bei dem das ECE- und das CWR-Bit gesetzt sind. Ein ECN-fähiger Rechner kann dies mit einem TCP-SYN/ACK-Paket beantworten, bei dem das ECE-Bit gesetzt und das CWR-Bit gelöscht ist. Anschließend kann ECN genutzt werden.

Weitere Informationen finden Sie im RFC 3168. Für die Intrusion Detection ist nun wichtig zu erkennen, dass derartige Pakete nicht grundsätzlich als fehlerhaft und gefährlich einzustufen sind. Wenn die gesamte TCP-Kommunikation überwacht wird, lässt sich während des Verbindungsaufbaus der Austausch der ECN-Informationen überwachen.

Der Linux-Kernel bietet die Möglichkeit, die ECN-Funktionalität ein- oder abzuschalten. Um die Funktionalität einzuschalten, genügt:

```
# sysctl -w net.ipv4.tcp_ecn=1
```

Das Abschalten erfolgt über die Zuweisung einer 0.

C.6 ICMP

IP ist ein Protokoll, das die Zustellung des Pakets nicht garantiert. Dies ist die Aufgabe der höheren Protokolle. Diese müssen bei Verlust eines Pakets diesen bemerken und das Paket erneut senden. Jedoch können Situationen auftreten, in denen kein Paket zum Ziel übertragen wird. In diesem Fall sollte der Absender informiert werden, um dauernde Neuübertragungen zu vermeiden oder um die Pakete modifiziert zu versenden. Es ist die Aufgabe des Internet Control Message Protocol (ICMP, RFC 792, STD 5), diese Informationen zu übertragen.

Hinweis



ICMP kann eingesetzt werden, um das Betriebssystem eines Rechners zu bestimmen. Ofir Arkin und Fyodor Yarochkin haben das Werkzeug X entwickelt, das mit einigen wenigen ICMP-Paketen und den Antworten ermitteln kann, mit welchem Betriebssystem es sich gerade unterhält. Das Werkzeug und Whitepapers sind verfügbar unter <http://www.sys-security.com/html/projects/X.html>.

ICMP ist das IP-Protokoll Nummer eins. Es wird in einem IP-Datagramm übertragen. Der ICMP-Header besteht aus dem acht Bit langen ICMP Type-Feld, dem acht Bit langen ICMP Code-Feld und einer 16 Bit langen Prüfsumme. Anschließend können Daten in Abhängigkeit vom Typ und Code angehängt werden.

Die folgenden ICMP-Typen und -Codes sind definiert:

Nachricht	Type	Code
Echo reply	0	0
Destination unreachable	3	
Network unreachable	3	0
Host unreachable	3	1
Protocol unreachable	3	2
Port unreachable	3	3
Fragmentation needed but DF set	3	4
Source route failed	3	5
Destination network unknown	3	6
Destination host unknown	3	7
Source host isolated (obsolete)	3	8
Destination network admin. prohibited	3	9
Destination host admin prohibited	3	10
Network unreachable for TOS	3	11
Host unreachable for TOS	3	12
Communication admin prohibited	3	13
Host precedence violation	3	14
Precedence cutoff in effect	3	15
Source quench	4	0

Redirect	5	
Redirect for network	5	0
Redirect for host	5	1
Redirect for TOS and network	5	2
Redirect for TOS and host	5	3
Echo request	8	0
Router advertisement	9	0
Router solicitation	10	0
Time exceeded	11	
Time exceeded during transit	11	0
Time exceeded during reassembly	11	1
Parameter problem	12	
IP header bad	12	0
Required option missing	12	1
Timestamp request	13	0
Timestamp reply	14	0
Information request (obsolete)	15	0
Information reply (obsolete)	16	0
Address mask request	17	0
Address mask reply	18	0

Die wichtigsten dieser Nachrichten sollen im Weiteren erläutert werden.

C.6.1 Destination Unreachable

Destination Unreachable ist eine der wichtigsten ICMP-Nachrichten. Sie wird verwendet, um dem Absender mitzuteilen, dass der Empfänger nicht erreichbar ist. Diese Nachricht verwendet verschiedene Subtypen, die über den ICMP-Code unterschieden werden.

Die häufigsten Subtypen sind: `Network Unreachable`, `Host Unreachable` und `Port Unreachable`. Die Nachricht `Network Unreachable` wird versendet, wenn ein Router keine Route für das Zielnetzwerk kennt. Die Nachricht `Host Unreachable` wird vom letzten Router versendet, wenn er den Zielrechner nicht erreichen kann (z.B. weil er ausgestellt ist). `Port Unreachable` wird vom Zielrechner verwendet, wenn das Paket versucht, einen nicht existenten UDP-Dienst auf dem Zielrechner anzusprechen. Handelt es sich um einen TCP-Dienst, so versendet der Zielrechner ein TCP Reset-Paket (siehe TCP).

Firewalls in Form eines Paketfilters verwenden ebenfalls häufig diese Meldungen, um einen Zugriff auf bestimmte Rechner und Dienste abzulehnen. Einfache Paketfilter können so auch erkannt werden, da sie häufig TCP-Anfragen mit einem ICMP-`Port Unreachable` beantworten. Das Zielsystem hätte ein TCP-Reset geschickt.

Die Meldung *Fragmentation Needed but DF-Bit set* wird verwendet, wenn ein Router das Paket nicht weiter senden kann, da es für das nächste Netzwerk zu groß ist. Zusätzlich wird dann die MTU des nächsten Netzwerks mit der Fehlermeldung übertragen. Diese Fehlermeldung wird von der Path Maximum Transmission Unit

Discovery (Path MTU Discovery) eingesetzt. Hierbei versucht das Betriebssystem, eine Fragmentierung der Pakete zu vermeiden, indem es zunächst die MTU Maximum Transmission Unit für den gesamten Pfad ermittelt. Anschließende Pakete werden dann mit dieser PMTU versandt.

Damit der Empfänger der Fehlermeldung `Destination Unreachable` erfährt, auf welches Paket sich die Fehlermeldung bezieht, enthält diese den IP-Header des originalen Pakets mit den ersten acht folgenden Bytes des Pakets. Dies erlaubt eine eindeutige Zuordnung des Pakets durch den Empfänger.



Achtung

Viele Paketfilter erlauben die Definition einer Network Address Translation. Das bedeutet, dass die Adressen der Pakete im IP-Header modifiziert werden. Einige Paketfilter kontrollieren jedoch nicht die IP-Adressen, die im IP-Header enthalten sind, der in der ICMP-Meldung eingebettet ist. So können private IP-Adressen nach außen gelangen!

C.6.2 Source Quench

Dies ist eine sehr einfache Fehlermeldung. Mit ihr teilt der Absender mit, dass er die Pakete nicht schnell genug verarbeiten kann und einige Pakete verwerfen muss. Diese Meldung wurde früher auch von Routern versendet. Diese verwenden jedoch heutzutage modernere Quality-of-Service-Funktionen. Source-Quench-Meldungen tauchen daher nur noch recht selten auf.



Achtung

Source-Quench-Meldungen können für einen Denial-of-Service-Angriff genutzt werden, wenn sie geeignet gespoofed werden.

C.6.3 Time Exceeded

Die Time Exceeded-Meldungen werden in erster Linie heute vom Werkzeug `traceroute` verursacht. Dieses Werkzeug versendet Pakete mit steigendem Time To Live (TTL)-Wert an den Empfänger. Die Router, über die die Pakete zum Ziel transportiert werden, werden die entsprechenden Pakete verwerfen und Fehlermeldungen mit ihrer Absender-IP-Adresse zurücksenden. So kann der Weg des Pakets rekonstruiert werden.

Es existieren zwei wesentliche Varianten des Programms Traceroute: `traceroute` (Unix) und `tracert.exe` (Microsoft Windows). Diese Programme versenden un-

terschiedliche Pakete. `tracert.exe` versendet an die Ziel-IP-Adresse jeweils drei Echo-Request-Pakete mit identischer TTL und inkrementiert dann den TTL-Wert. Die Rücklaufzeit der Pakete wird gemessen und angezeigt. Das Unix-Programm `traceroute` versendet UDP-Pakete an die Ports 33434 ff. Hierbei werden ebenfalls immer drei Pakete mit identischer TTL an einen Port gesendet. Jedes Mal, wenn das Programm die TTL inkrementiert, wird auch der Port inkrementiert.

C.6.4 Redirect

Die Redirect-Nachrichten werden von einem Router versendet, der den Absender eines Pakets über einen kürzeren Pfad informieren möchte. Router können ihre Routing-Tabellen dynamisch untereinander mit dem Routing Information Protocol (RIP) austauschen. Hierdurch kennt ein Router alle weiteren Router und sämtliche möglichen Routen. Erhält ein Router nun ein Paket und stellt fest, dass sich in demselben Netzwerk ein weiterer besser geeigneter Router befindet, so übermittelt er diese Information an den Client. Der speichert die Information in seiner Routing-Cache ab und wird das nächste Paket direkt an diesen geeigneteren Router versenden. Der Routing-Cache kann unter Linux mit dem Befehl `route -Cn` betrachtet werden.



Achtung

Redirect-Meldungen können verwendet werden, um Router zu spoofen. Wenn ein Netzwerk keine dynamischen Routing-Protokolle einsetzt, sollten derartige Meldungen starkes Misstrauen hervorrufen. Unter Linux kann die Annahme derartiger Meldungen mit der Kernel-Variablen `/proc/sys/net/ipv4/conf/*/accept_redirects` abgestellt werden.

C.6.5 Parameter Problem

Die Fehlermeldung `Parameter Problem ICMP` wird versendet, wenn das IP-Datagramm selbst Fehler aufweist. Meistens wurde eine IP-Option falsch verwendet. Da die meisten IP-Implementierungen inzwischen jedoch recht ausgereift sind, kommen diese Fehlermeldungen eigentlich nur noch gehäuft vor, wenn Pakete manuell fehlerhaft konstruiert werden.

C.6.6 Echo-Request und Reply

Echo-Request und -Reply sind die ICMP-Nachrichten, die vom Kommando `ping` verwendet werden. Hierbei sendet ein Rechner den ICMP-Echo-Request. Der Empfänger antwortet auf jedes Echo-Request-Paket mit einem Echo-Reply-Paket. Um die Pakete voneinander trennen zu können, enthalten sie eine eindeutige Identifikationsnummer und eine Sequenznummer. Die Identifikationsnummer wird verwendet, um die Pakete mehrerer gleichzeitiger `ping`-Aufrufe voneinander zu trennen. Die Sequenznummer wird für jedes versandte Paket inkrementiert und identifiziert

die einzelnen Pakete. Dies erlaubt die eindeutige Zuordnung eines Reply-Pakets zu dem Request-Paket und die Ermittlung der Übertragungszeit.

Die meisten Implementierungen des `ping`-Kommandos erlauben es, die Anzahl der zu übertragenden Bytes und den Inhalt zu definieren. So kann unter Linux mit der Option `-p` ein Muster (Pattern) definiert und mit der Option `-s` die Größe angegeben werden. Werden diese Informationen nicht modifiziert, so erlauben häufig die Größe und der Inhalt des Pakets einen Rückschluss auf das sendende Betriebssystem.

Eine weitere Option, die vor allem in Unix-Implementierungen des Befehls existiert, ist `-b`. Diese erlaubt ein Broadcast-Ping. Hierbei werden die Echo-Request-Pakete an eine Broadcast-Adresse gesendet. Üblicherweise antworten sämtliche Unix- und Linux-Rechner auf eine derartige Anfrage. In der Vergangenheit konnten hiermit Denial-of-Service-Angriffe erzeugt werden. Der Angreifer spoofte ein Echo-Request-Paket und sendete es an eine Broadcast-Adresse. Sämtliche Rechner antworteten und schickten ihre Antwort an den gespoofen Rechner. Wurden hierzu Netzwerke mit mehreren hundert Rechnern verwendet, konnte der gespoofte Rechner häufig überflutet werden. Heute existieren kaum noch Netzwerke, die diese Pakete, die an die Broadcast-Adresse gerichtet sind, hineinlassen. Dieser Angriff ist unter dem Namen SMURF berühmt geworden. Die Netzwerke bezeichnet man als SMURF-Amplifier-Netzwerk (Verstärker). Informationen hierzu finden Sie zum Beispiel unter <http://www.powertech.no/smurf/>.

Ping-Pakete sind in modernen Netzwerken vollkommen normal. Das Vorkommen lediglich von Echo-Reply-Paketen sollte jedoch Ihre Aufmerksamkeit erregen. Hierbei könnte es sich um einen Tunnel handeln.

C.6.7 Address Mask Request und Reply

Diese beiden Nachrichten können verwendet werden, um die Subnetzmaske eines Rechners zu ermitteln. Diese Nachrichten verwenden ähnlich dem Echo eine Identifikationsnummer und eine Sequenznummer, um die Nachrichten zuzuordnen zu können.

Diese Anfragen werden häufig verwendet, um herauszufinden, ob ein bestimmter Rechner erreichbar ist und welche Adressmaske er verwendet. Leider existiert kein klassisches Kommandozeilenwerkzeug für die Erzeugung der Anfrage. Ein Werkzeug, das jedoch genutzt werden kann, ist `icmpquery`. Dieses Werkzeug ist unter <http://www.angio.net/security/> erhältlich. Hiermit können Rechner erreicht werden, bei denen ein Ping durch eine Firewall blockiert wird. Linux reagiert auf einen Address Mask Request nicht.

C.6.8 Timestamp Request und Reply

Diese Meldungen sind in der Lage, die Latenz des Netzwerks zu messen. Hierzu ist es jedoch erforderlich, dass sowohl der Absender als auch der Empfänger synchrone Uhrzeiten verwenden. Ähnlich den Echo-Meldungen und den Address-

Mask-Meldungen verwenden diese Meldungen auch eine Identifikationsnummer und eine Sequenznummer, um die Pakete zuordnen zu können. Ein Werkzeug, das in der Lage ist, diese Meldungen zu erzeugen, ist `icmpquery`. Es wurde bereits bei den Address-Mask-Meldungen erwähnt. Der Linux-Kernel 2.4 reagiert auf eine derartige Anfrage nicht mehr. Der Linux-Kernel 2.2 beantwortet diese Anfrage.

C.6.9 Router Solicitation und Advertisement

Wenn ein Netzwerkgerät, das das Router-Discovery-Protokoll unterstützt, eingeschaltet wird, sendet es eine Router-Solicitation-Meldung. Alle weiteren Router in demselben Netzwerk antworten mit einer Router-Advertisement-Nachricht. Damit alle Router die Solicitation-Nachricht erhalten, wird diese entweder an die Broadcast-Adresse 255.255.255.255 oder die *All-Routers-Multicast-Adresse* 224.0.0.2 gesendet. Alle Router antworten auf diese Anfragen mit einem Unicast-Paket. Zusätzlich versenden die Router regelmäßig ohne Aufforderung Router-Advertisement-Meldungen an die Adresse 224.0.0.1.

C.7 ARP

Wenn zwei IP-fähige Netzwerkgeräte sich in einem lokalen Netz unterhalten möchten, so müssen sie zunächst ihre Hardware-Adressen austauschen. Die tatsächliche Kommunikation erfolgt nicht auf Basis der IP-Adressen, sondern anhand dieser Hardware-Adressen. Für das Medium Ethernet wurde das Address Resolution Protocol (ARP) entwickelt. Dieses Protokoll wurde inzwischen auf die meisten anderen Netzwerkmedien portiert. Es erlaubt einem Netzwerkgerät, eine Anfrage (ARP Request) zu senden, die von der entsprechenden Gegenstelle mit einer Antwort (ARP Reply) beantwortet wird.

ARP-Pakete werden auf der Data-Link-Schicht versendet. Das ist dieselbe Schicht, die von IP-Paketen genutzt wird. ARP-Pakete sind also unabhängig von IP-Paketen.

ARP-Anfragen werden üblicherweise an alle Rechner eines Netzes versandt. Die Zieladresse des Pakets ist daher `ff:ff:ff:ff:ff:ff`. Dies ist die Ethernet-Broadcast-Adresse. Alle Rechner des Netzes verarbeiten das Paket. Es antwortet aber nur derjenige Rechner, der die richtige IP-Adresse besitzt.

Die Ergebnisse dieser Anfragen werden von den Rechnern in einem ARP-Cache zwischengespeichert. Das Verhalten des ARP-Caches wird unter Linux über das `sysctl`-Interface in `/proc/sys/net/ipv4/neigh/*` gesteuert. Die Manpage des ARP-Kernel-Moduls `arp` (7) gibt nähere Auskunft über die Werte. Der Inhalt des ARP-Caches kann mit dem Befehl `arp` angezeigt werden:

```
# tcpdump -np arp
tcpdump: listening on eth1
15:22:23.374171 0:10:a4:c3:26:cb Broadcast arp 42: arp who-has 192.168.0.101
  tell 192.168.0.202 15:22:23.374625 0:e0:7d:7d:70:69 0:10:a4:c3:26: arp 60:cb
  arp reply 192.168.0.101 is-at 0:e0:7d:7d:70:69
```

```
Ctrl-C
# arp -an
? (192.168.0.1) auf 00:50:BF:11:23:DF [ether] auf eth1
? (192.168.0.101) auf 00:E0:7D:7D:70:69 [ether] auf eth1
```

Der Linux-Kernel kann maximal 1024 Einträge in seinem Cache verwalten.

Hinweis: ARP-Spoofing



ARP führt keine Authentifizierung durch. ARP-Antworten können daher gefälscht werden. Viele Betriebssysteme verarbeiten alle ARP-Antworten, die sie sehen, ohne dass sie jemals eine ARP-Anfrage gesendet hätten. Des Weiteren gibt es die Möglichkeit des so genannten Gratuitous ARP. Hierbei aktualisiert der Empfänger einen bereits vorhandenen Eintrag.



Literaturverzeichnis

- [1]. Anonymous: *Der neue Hacker's Guide*. 2., überarbeitete Aufl. München: Markt+Technik 2001.
- [2]. Anonymous: *Der neue Linux Hacker's Guide*. 1. Aufl. München: Markt+Technik 2001.
- [3]. Bace, Rebecca Gurly: *Intrusion Detection*. 1. Aufl. Indianapolis: Newriders 2000.
- [4]. Barman Scot: *Writing Information Security Policies*. 1. Aufl. Indianapolis: New Riders 2002.
- [5]. Barrett, Daniel J., Richard E. Silverman: *SSH: Secure Shell – Ein umfassendes Handbuch*. 1. Aufl. Köln: O'Reilly 2001.
- [6]. Bellovin, William, Steven Cheswick: *Firewalls und Sicherheit im Internet*. 2., überarbeitete Aufl. Bonn u.a.: Addison-Wesley 1995.
- [7]. Blaze, Matt, Whitfield Diffie, Ronald I. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thomson, Michael Wiener: *Minimal Key Length of Symmetric Ciphers to Proved Adequate Commercial Security*. 1996. <http://www.counterpane.com/keylength.html>
- [8]. Cavallar, Stefania, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, Gérard Guillerm, Paul Leyland, et al.: *Factorisation of a 512-bit RSA modulus*. In: *Theory and Application of Cryptographic Techniques*, <ftp://ftp.gage.polytechnique.fr/pub/publications/jma/rsa-155.ps>.
- [9]. Erickson, Jon: *Hacking: The Art of Exploitation*. San Francisco: No Starch Press 2003.
- [10]. Friedl, Jeffrey E.F.: *Reguläre Ausdrücke*. 1. Aufl. Köln: O'Reilly 1997.
- [11]. Hall, Eric A.: *Internet Core Protocols: The Definitive Guide*. 1. Aufl. Sebastopol u.a.: O'Reilly 2000.
- [12]. Hildebrandt, Ralf, Patrick B. Koetter: *Postfix*. Heidelberg: Dpunkt Verlag 2005.
- [13]. Kahn, David: *The Codebreakers*. 2., überarbeitete Aufl. New York: Simon & Schuster Inc. 1997.

-
- [14]. Klein, Tobias: *Buffer Overflows und Format-String-Schwachstellen*. Heidelberg: Dpunkt Verlag 2003.
- [15]. Kurtz, George, Stuart McClure, Joel Scambray: *Das Anti-Hacker Buch*. 3. Aufl. Bonn: MITP 2001.
- [16]. Lenstra, Arjen K., Eric R. Verheul: *Selecting Cryptographic Key Sizes*. In: Journal of Cryptology. Bd. 14(4) 2001. S. 255-293.
- [17]. MacDonald, Alistair: *SpamAssassin*. München u.a.: Addison-Wesley 2005.
- [18]. Mandia, Kevin, Chris Prosis: *Incident Response: Investigating Computer Crime*. 1. Aufl. New York u.a.: Osborne/McGraw Hill 2001.
- [19]. Mann, Scott, Ellen L. Mitchell: *Linux System Security*. 1. Aufl. Upper Saddle River: Prentice Hall 2002.
- [20]. Northcutt, Stephen, Mark Cooper, Matt Fearnow, Karen Frederick: *Intrusion Signatures and Analysis*. 1. Aufl. Indianapolis: New Riders 2001.
- [21]. Northcutt, Stephen, Judy Novak: *Network Intrusion Detection: An Analyst's Handbook*. 2. Aufl. Indianapolis: New Riders; 2001; ISBN 0-7357-1008-2
- [22]. Proctor, Paul E.: *The Practical Intrusion Detection Handbook*. 1. Aufl. Upper Saddle River: Prentice Hall PTR 2001.
- [23]. The Honeynet Project: *Know Your Enemy*. 1. Aufl. Reading u.a.: Addison Wesley 2002.
- [24]. Toxen, Bob: *Real World Linux Security*. 1. Aufl. Upper Saddle River: Prentice Hall 2001.
- [25]. Schneier, Bruce: *Applied Cryptography*. 2., überarbeitete Aufl. New York u.a.: John Wiley & Sons 1995.
- [26]. Schultz, E. Eugene, Russell Shumway: *Incident Response*. 1. Aufl. Indianapolis: New Riders 2001.
- [27]. Spenneberg, Ralf: *Intrusion Detection und Prevention mit Snort und Co*. Bonn u.a.: Addison-Wesley 2004.
- [28]. Spenneberg, Ralf: *VPN mit Linux*. München u.a.: Addison-Wesley 2003.
- [29]. Stevens, W. Richard: *TCP/IP Illustrated*. Bd. 1., 1. Aufl. Reading u.a.: Addison Wesley 1994.
- [30]. Stoll, Clifford: *Kuckucksei*. 5. Aufl. Frankfurt am Main: Fischer Taschenbuchverlag 1998.
- [31]. Wyk, Kenneth R. van, Richard Forno: *Incident Response*. 1. Aufl. Sebastopol u.a.: O'Reilly 2001.
- [32]. Ziegler, Robert L.: *Linux Firewalls*. 2., überarbeitete Aufl. München: Markt+Technik Verlag 2002.



Stichwortverzeichnis

2.6.14 359, 373, 375, 379, 435, 489, 587
26sec 560, 562

A

-A 84
ACCEPT 87, 199, 375, 516, 517
accept_redirects 446
accept_source_route 135, 446
ACCOUNT 393, 395
account 395
ACK 106, 402, 410
Address Resolution Protocol *siehe* ARP
address-mask-reply 570, 580
address-mask-request 570, 580
addrtype 363
ADSL 380
Advanced Maryland Automatic Network
 Disc Archiver *siehe* Amanda
AH *siehe* Authentication Header
ah 363
Alarmierung, E-Mail 258
AltaVista Firewall 39
Amanda 543
Amavisd-New 593
Angriffe
 gespoofter Portscan 63
 Session Hijacking 64
 SMURF 625
Antirez 339
--append 84
Appletalk 510
Application-Level-Gateway 43, 48
Architektur 45
Arkin, Ofir 621
ARP 498, 506, 626
 Cache 626
 Reply 626
 Request 626
 Spoofing 464
arp_announce 446
arp_filter 446
arp_ignore 446

ARPANET 37, 599
--arp-htype. 515
--arp-ip-dst. 515
--arp-ip-src 515
--arp-mac-dst. 515
--arp-mac-src 515
--arp-opcode. 515
--arp-ptype. 515
arpreply 516
--arpreply-mac 516
--arpreply-target 516
arptables 505
Assigned Number Authority 463, 607
--atomic-commit 514
--atomic-file 514
--atomic-init. 514
--atomic-save. 514
Audit 319
Ausspähen von Daten 56
Authentication Header 559
Authentizität 55
Avolio, Frederick 38

B

Balabit 228
BALANCE 412
Bandbreitenkontrolle 296, 308
Banner-Grabbing 329, 338
Base64 527
Basel-2 56
BASH 97, 321
Bastille-Linux 143, 155
Bellovin, Steven M. 39
benutzerdefinierte Kette 195, 198
Bernstein, Dan 443
Betriebssystem-Bestimmung 329
Binary Increase Congestion 436
BitTorrent 535
bootp_relay 447
Bootvorgang 100
Bot-Netz 58
Boundary Checking 67

Bridge 273, 296, 370, 501, 507, 523
 Broadcast 302, 370, 515, 606
 Ping 431, 581
 Brouter 507, 508
 BROUTING 508, 517
 BSI 205, 352
 BSI OSS Security Suite 352
 Bufferoverflow 57
 Bugtraq 38
 Bundesamt für Sicherheit in der
 Informationstechnik *siehe* BSI

C

Chaos Computer Club 52
 Check Point 39, 42, 575
 Cheswick, William R. 39
 chkconfig 149, 252
 Chroot 155, 217
 CIDR 86
 -Notation 86, 359
 Cisco PIX 273
 CLASSIFY 132, 375, 421
 Classless Internet Domain Routing
siehe CIDR
 CLOSE 410
 CLOSE_WAIT 410
 Cluster 469
 CLUSTERIP 375
 Cohen, Dr. Fred 51
 comment 364
 Computer Emergency Report Team
 Coordination Center 38
 condition 396
 connbytes 364
 Connection Tracking 401
 connlimit 386
 CONNMARK 377, 412, 420–422, 481
 connmark 365
 connrate 396
 Conntrack 111
 conntrack 365, 490
 Content-Filter 594
 CONTINUE 516, 517
 Cookie 528
 Cracker 51
 Crond 237
 Cron-Daemon 148
 ct_sync 425, 477
 Curcuit Relay 42
 -Proxy 39

D

-D 87
 -d 86, 359
 DARPA 37
 Datagram Congestion Control Protocol
siehe dccp
 Datagramm 606
 dccp 365
 DCF-77 242
 Debian 220, 308
 DEC-LAT 510
 Decoy-Scan 337
 DEC-SEAL 39
 Deep-Inspection 42
 Default Policy 83, 324
 Defragmentierung 435
 --delete 87
 Denial-of-Service 56, 57, 389, 435, 623, 625
 Deraison, Renaud 356
 Desktop-Firewall 172
 --destination 86, 359, 515
 Destination-Header 589
 Destination-NAT 44, 122, 412, 415
 --destination-port 89
 --destinationport 361, 362
 destination-unreachable 569–571
 DHCP 303, 315, 368, 521
 Differentiated Services *siehe* ECN
 DiffServ-Code-Point 365
 disable_policy 447
 disable_xfrm 447
 Disaster-Recovery 146
 Discretionary-Access-Control 162
 DMZ 44, 46, 171, 185, 204, 418
 DMZDEV 186
 DNAT 377, 412, 418
 dnat 516
 DNS 99, 103, 176, 523
 Server 187
 Tunnel 524
 DNSSEC 524
 Domain Name System *siehe* DNS
 Doppelwort 601
 --dport 89
 DROP 84, 140, 199, 377, 516
 DSCP 131, 377, 421, 422
 dscp 365
 dst 86, 589
 Dynamic Host Configuration Protocol
siehe DHCP

E

Eastep, Tom 288
 Ebtables 507
 ebtables 506
 Echo-Reply 117
 echo-reply 570, 580
 Echo-Request 117, 431
 echo-request 183, 570, 580
 ECN 132, 292, 366, 377, 421, 422, 437,
 613, 619
 Congestion Experienced 619
 eDonkey 532, 539, 541
 Einbruch 47
 ELSTER 529
 E-Mail 96, 546
 -Relay 591
 -Server 47, 187
 Encapsulated Security Payload 118,
 335, 559
 EpyLog 271
 ESP *siehe* Encapsulated Security Payload
 esp 366
 ESTABLISHED 93, 111, 117, 373, 401, 403,
 404, 410
 /etc/hosts.deny 234
 Ethereal 320, 486, 527
 Ethernet 501
 eui64 589
 ExecShield 69
 eXecute-Disable-Bit 69
 expire 387
 Explicit Congestion Notification *siehe* ECN
 Exploit 52
 EXTDEV 186

F

-f 360
 Fail-Over 471
 Fe3d 349
 Fedora Core 228, 271, 274, 297, 507
 Fehler 319
 Fehlersuche 395
 File Transfer Protocol *siehe* FTP
 Filesharing 534
 Filter-Tabelle 82
 FIN 410
 FIN_WAIT 410
 Firestarter 283, 307
 Firewalking 603
 Firewall Builder 273, 298, 307

Firewall-Markierung 369, 423, 458, 460, 566
 Fli4L 144
 force_igmp_version 447
 Formatstring-Angriff 57, 70, 76
 For-Schleife 182
 FORWARD 82, 191, 203,
 324, 421, 504, 505, 508
 Forward Acknowledgment 437
 Forward RTO Recovery 438
 Forwarding 94, 188, 302, 324
 forwarding 135, 447
 --fragment 360
 fragmentation-needed 569, 571, 572
 Fragmentierung 341, 435, 571, 602
 FreeSwan 560
 FTP 43, 96, 116, 190, 367, 526, 537
 fuzzy 387
 Fwlogwatch 251

G

Gauntlet 39
 geoip 396
 Geschwindigkeit 493
 Gnomemeeting 396, 555
 goto 396
 GRE 118, 398, 545
 grsecurity 74, 161
 Grub 154

H

H.323 396, 555
 h323-contrack-nat 396
 Hacker 51
 Härtung 143
 hashlimit 366
 Hashsize 405, 407
 hbb 589
 helper 367
 HL 588
 hl 589
 Hochverfügbarkeit 375, 469
 HoneyNet Project 604
 Hop-by-Hop-Optionen 589
 Hoplimit 589
 Hot-Standby 469
 hping2 339, 361
 HTTP 43, 121, 176, 526
 HTTPS 526
 hunt 65
 Hydra 351

I

- i 86, 360, 504
- IANA *siehe* Assigned Number Authority
- ICMP 401, 403, 569, 604, 621
 - Address Mask Request 625
 - Code 621
 - Destination Unreachable 622
 - Echo-Reply 624
 - Echo-Request 624
 - Header 621
 - Identifikationsnummer 624
 - Parameter Problem 624
 - Redirect 624
 - Router Solicitation
 - und Advertisement 626
 - Sequenznummer 624
 - Source Quench 617, 623
 - Time Exceeded 623
 - Timestamp Request 625
 - Type 621
- icmp_echo_ignore_all 430
- icmp_echo_ignore_broadcasts 431
- icmp_errors_use_inbound_ifaddr 431
- icmp_ignore_bogus_error_messages 431
- icmp_ratelimit 431
- icmp_ratemask 432
- icmpquery 625
- icmp-type 362
- identd 96, 191, 541
- Idle-Scan 339
- IDS *siehe* Intrusion-Detection-System
- IGMP 118
- igmp_max_memberships 432
- igmp_max_msf 432
- IKE *siehe* Internet Key Exchange
- IMAP 551
- in-interface 86, 360, 504, 515
- init-table. 514
- INPUT 82, 120, 421, 505, 508
- INTDEV 186
- Integrität 55
- Internet Key Exchange 559
- Internet Message Access Protocol
 - siehe* IMAP
- Internet Protocol *siehe* IP
- Internet Relay Chat *siehe* IRC
- Internet Software Consortium 242
- Internet-Group-Management-Protokoll 432
- Intrusion-Detection-System 42, 48, 165, 192, 241
- Intrusion-Prevention-System 165
- INVALID 111, 373, 402, 403
- IP 600, 621
 - Adresse 604
 - DF-Bit 602, 622
 - Fragment-Offset 603
 - Header 600
 - Länge 601
 - Identifikationsnummer 602
 - MF-Bit 603
 - Optionen 604
 - Loose Source Routing 605
 - No Operation 605
 - Record Route 605
 - Router Alert 606
 - Security Options 605
 - Strict Source Routing 606
 - TimeStamp 606
 - Paketlänge 602
 - Prüfsumme 604
 - Time To Live 603, 623
 - Type-of-Service 602
 - Version 601
- ip6tables 587
- ip_autoconfig 433
- ip_contrack 111, 115, 404, 408, 430
- ip_contrack_amanda 543
- ip_contrack_buckets 407
- ip_contrack_count 407
- ip_contrack_expect 430
- ip_contrack_ftp 116, 191, 539
- ip_contrack_generic_timeout 407
- ip_contrack_h323 555
- ip_contrack_icmp_timeout 407
- ip_contrack_irc 549
- ip_contrack_log_invalid 407
- ip_contrack_max 405, 408, 433
- ip_contrack_pptp 545
- ip_contrack_proto_gre 545
- ip_contrack_sip 557
- ip_contrack_tcp_be_liberal 408
- ip_contrack_tcp_loose 408
- ip_contrack_tcp_max_retrans 408
- ip_contrack_tcp_timeout_close 408
- ip_contrack_tcp_timeout_close_wait 408
- ip_contrack_tcp_timeout_established 409
- ip_contrack_tcp_timeout_fin_wait 409
- ip_contrack_tcp_timeout_last_ack 409
- ip_contrack_tcp_timeout_max_retrans 409
- ip_contrack_tcp_timeout_syn_recv 409
- ip_contrack_tcp_timeout_syn_sent 409
- ip_contrack_tcp_timeout_time_wait 409
- ip_contrack_timeout_max_retrans 408

- ip_conntrack_udp_timeout 410
 - ip_conntrack_udp_timeout_stream 410
 - ip_default_ttl 134, 433
 - ip_dynaddr 434
 - ip_echo_ignore_broadcasts 134
 - ip_forward 94, 134, 324, 434
 - ip_local_port_range 434
 - ip_nat_amanda 543
 - ip_nat_ftp 191, 420, 540
 - ip_nat_irc 549
 - ip_nat_pptp 546
 - ip_nat_proto_gre 546
 - ip_nat_sip 557
 - ip_nat_snmp_basic 542
 - ip_nat_tftp 550
 - ip_no_pmtu_disc 435
 - ip_nonlocal_bind 434
 - ip_queue_vwmark 397
 - ip_tables_matches 430
 - ip_tables_names 430
 - ip_tables_targets 430
 - IPchains 103, 105, 110, 115
 - IPCop 144, 307, 318
 - ip-destination 516
 - ip-destination-port 516
 - IP-Filter 42, 273, 406
 - ipfrag_high_thresh 435
 - ipfrag_low_thresh 435
 - ipfrag_max_dist 435
 - ipfrag_secret_interval 435
 - ipfrag_time 435
 - iphash 463, 465, 466
 - ipmap 462, 463
 - IPMARK 393, 421, 422
 - ipp2p 397
 - IP-Personality Patch 331
 - ippool 505
 - ipporthash 463, 466
 - ip-protocol 516
 - iprange 368
 - IPS *siehe* Intrusion Prevention System
 - IPsec 39, 296, 301, 308, 382, 397, 425, 448, 559
 - ipset 390, 461, 505
 - ip-source 516
 - ip-source-port 516
 - IP-Spoofing 431, 448
 - ipt_ULOG 452
 - Iptables 81, 110
 - ip-tos 516
 - iptree 463, 466
 - IPTState 263
 - ipv4options 388
 - IPV4OPTSSTRIP 421, 423
 - IPv6 241, 335, 587
 - Header 589
 - ipv6header 589
 - IRC 419, 541, 548
 - itunnel 581
- J
- j 84
 - Jiffie 431
 - jump 84
- K
- Kadlecsik, Jozsef 406
 - Kaminsky, Dan 525
 - KaZaA 534, 539, 541
 - Keepalive 438
 - KeepAlived 473
 - Kleinrock, Leonard 37
 - KLIPS 560, 561
 - Klogd 215, 218
 - Kollision 502
 - Krauz, Pavel 65
 - Kwiatkowski, Michal 458
- L
- L 86
 - l2drop 481
 - LAST_ACK 410
 - Lastverteilung 375
 - Latenzzeit 616
 - Layer-2-Drop 481, 483
 - Lebensdauer 387
 - length 368
 - Libpcap 457
 - Libsafe 70
 - Licklider, J.C.R. 37
 - LIDS 74, 161
 - Lilo 154
 - limit 366, 368
 - line-numbers 86
 - Linux-Virtual-Server 473
 - list 86
 - Loadbalancer 375
 - LOG 136, 323, 377, 451
 - log 516
 - log_martians 135, 447
 - log-arp 516
 - logger 219

--logical-in 515
 --logical-out 515
 Logical-Volume-Manager 146
 --log-ip 516
 --log-level 516
 --log-prefix 516
 Logwatch 271
 lokkit 297
 low hanging fruits 50

M

mac 369
 MAC-Adresse 61, 501, 517
 macipmap 463, 464
 Mafia 54
 Mail-Routing 523
 Mandatory Access Control 161
 Mangle-Tabelle 377, 393–395, 421, 504
 MARK 132, 378, 421–423, 566
 mark 369, 516
 --mark 516
 --mark-target 517
 MASQUERADE 94, 378, 412, 415
 Maximum Segment Size 361, 373, 380, 436, 575, 607, 615
 Maximum Transmission Unit 380, 572, 608, 615, 623
 mc_forwarding 447
 McHardy, Patrick 566
 MD5 226
 medium_id 447
 Merrill, Thomas 37
 Mitnick, Kevin 57, 612
 Mkdrec 145
 Mogul, Jeffrey C. 41, 45
 Mondo-Rescue 145
 Morris, Robert Tappan 38
 mport 397
 MRTG 269
 MSS *siehe* Maximum Segment Size
 Msyslogd *siehe* Syslogd
 MTU *siehe* Maximum-Transmission-Unit
 Multicast 606
 multiport 370
 MySQL 219, 225
 Erzeugung der Datenbank 225

N

-n 86
 Named-Pipe 235

Nameserver 323, 524
 NAT *siehe* Network Address Translation
 National Security Agency 38
 NCP *siehe* Network Control Protocol
 NDiff 345
 Nessus 319, 350
 NETBEUI 506, 508
 Netfilter 81
 nethash 463, 465
 NETMAP 378, 387, 412, 416
 Netmeeting 396, 555
 Network Address and Port Translation 414
 Network Address Translation 43, 94, 122, 188, 412, 504, 542, 544, 604, 623
 Tabelle 94, 111, 121, 377, 411
 Network Control Protocol 37, 537
 Network News Transfer Protocol
 siehe NNTP
 Network Time Protocol *siehe* NTP
 NEW 93, 111, 373, 401–403
 nf-HiPAC 493
 NFLOG 452
 Nfnetlink 489
 NFQUEUE 379
 Nikto 352
 Nmap 63, 140, 319, 324, 351, 422, 579
 -Audit 343
 -Parser 347
 nmap 105
 NNTP 554
 No-eXecute-Bit 69
 NOP Sled 68
 NOTRACK 132, 379, 425
 NSFNET 37
 nstx 524
 nth 388
 NTP 241, 315, 554
 NuFW 270, 397
 Null-Scan 336
 Nulog 270
 --numeric 86
 NVP 604

O

-o 86, 360, 504
 OpenBSD 242, 388
 Openswan 560
 OpenVPN 296
 OpenWRT 149
 osf 388
 OSI-Modell 599

--out-interface 86, 360, 504, 515
 OUTPUT 82, 95, 111, 120, 121, 132, 411,
 421, 425, 505, 508, 517
 Owner 178
 owner 370
 owner-socketlookup 397

P

-p 84, 359
 Paketfilter 41, 45
 parameter-problem 570, 578
 passives Betriebssystem-Fingerprinting 388
 Patch-O-Matic 359, 374, 383, 392, 416, 419,
 425, 461, 555, 566, 587
 Path Maximum Transmission Unit
 Discovery 435, 572, 623
 Peer-to-Peer 397
 PEO-Hash 219
 Personal-Desktop-Firewall 283
 Phishing 54
 php-syslog-ng 235
 physdev 370
 --physdev-in 504
 --physdev-is-bridged 504
 --physdev-is-in 504
 --physdev-is-out 504
 --physdev-out 504
 Ping 84, 117, 428, 580
 Sweep 333
 ping 569
 Ping of Death 400, 603
 pkttables 489
 pkttype 370
 --pkt-type 515, 516
 Pluggable Authentication Modules 158
 PMTUD *siehe* Path-Maximum-
 Transmission-Unit-Discovery
 Point-to-Point Tunneling Protocol 419
 Point-to-Point-Protokoll 398, 544
 Policy 199
 policy 397, 567
 pool 390
 POP3 552
 Port Scan Attack Detector 160
 Port-Forwarding 126, 415, 418
 portmap 462, 463
 Portscan 172
 Detektor 389
 gespoofed 63
 POSIX-ACLs 161
 Post Office Protocol *siehe* POP3

Post Office Protocol 3 *siehe* POP3
 Postfix 190, 591
 Mail-Relay 190
 PostgreSQL 219
 POSTROUTING 94, 121, 411, 421, 504, 508
 PPPoE 380
 PPP-over-Ethernet *siehe* PPPoE
 PPTP 118, 380, 382
 pptp-contrack-nat 398
 Preboot Execution Environment 549
 Prelude-IDS 70
 PREROUTING 95, 111, 121, 132, 188, 377,
 379, 411, 421, 425, 504, 508, 516, 517
 private IP-Adressen 123
 /proc 94, 427
 /proc/net/ip_contrack 263
 promote_secondaries 448
 ProPolice 69
 --protocol 84, 359
 protocol-unreachable 571
 Protokolle
 Analyse 251
 Verkettung 227
 Protokollierung 215, 235, 451
 MySQL 235, 265, 456
 PostgreSQL 235, 265, 457
 SQLite 458
 Protokoll-Scan 335
 Protokollserver 215
 Proxy 42, 46, 119, 187, 188
 proxy_arp 448
 ProxyARP 294, 447, 497
 psd 389
 ptrace 74
 PXE *siehe* Preboot Execution Environment

Q

quake3-contrack-nat 398
 Quality-of-Service 131, 602
 QUEUE 379, 397
 quota 390

R

Race-Condition 57, 71
 random 390
 Random Early Detection 619
 Ranum, Marcus J. 38, 43
 Raptor Eagle 39
 RAS 498
 Raw-Tabelle 132, 373, 379, 395, 425

- Real Time Transport Protocol *siehe* RTP
 realm 371
 recent 371
 RED *siehe* Random Early Detection
 Red Hat 218, 220, 228, 274, 297, 308, 643
 REDIRECT 379, 412, 418, 517
 Redirect 448
 redirect 516, 569
 --redirect-target 517
 Reed, Darren 42
 regulärer Ausdruck 224
 Reid, Brian 38
 REJECT 85, 140, 199, 379, 504
 RELATED 93, 111, 191, 373, 401, 404, 569
 Remote-Access-Service *siehe* RAS
 RETURN 199, 516
 Reverse-Path-Filter 448
 RFC 1918 123, 413
 RIPEMD160 226
 Rooij, Guido van 406
 ROUTE 132, 394, 421, 423
 router-advertisement 569, 578
 router-solicitation 578
 router-solicitation 570
 Routing Information Protocol 624
 Routing-Header 590
 rp_filter 135, 448
 rpc 398
 RPM 151
 RSBAC 161
 rsh 398
 RST 402, 408
 rt 590
 RTP 556
 Runlevel 100
- S**
- s 86, 359
 SAME 412, 417
 Samhain 166
 Sarbanes-Oxley-Act 56
 Scanulog 267
 Schulung 643
 screend 41, 45
 Screening-Routers 45
 Script-Kiddie 52
 sctp 372
 Secure Shell 63, 88, 530, 531
 Dienst 120
 Secure Socket Layer 526, 528, 551, 554
 secure_redirects 448
 Selective Acknowledgement 442
 SELinux 74, 298
 send_redirects 448
 Sequenznummer 59, 106
 Session Initiation Protocol *siehe* SIP
 set 390
 SetGID 144, 152
 --set-mark 517
 SetUID- 144, 152
 SF Firewall 42
 SHA-1 226
 shared_media 448
 Shorewall 287, 298, 307
 Silent Host 64, 339
 Simple Mail Transport Protocol
 siehe SMTP
 Simple Network Management Protocol
 siehe SNMP
 Simple-Network-Time-Protocol
 siehe SNTP
 sing 579
 Single-Point-of-Defense 77
 Single-Point-of-Failure 45, 48
 Single-Sign-On 537
 SIP 556
 sip 399
 Smoothwall 317
 SMTP 541, 546
 SMURF *siehe* Angriffe
 Smurf 134
 Angriff 431
 SNAT 380, 412, 417
 snat 516
 --snat-target 517
 SNMP 420, 542
 Snort 315, 389
 -Inline 379
 Snort 2 165
 SNTP 241
 Social Engineering 57
 SOCKS 39, 42
 Solar Designer 389
 --source 86, 359, 515
 Source Routing 414
 Loose 605
 Strict 606
 Source-NAT 44, 122, 412, 413
 --sourceport 360, 362
 source-quench 569
 Spam 53, 592
 Spanning-Tree-Protokoll 503, 516
 Specter 458

- Spoofing 59, 218, 341
 - ARP 59, 65, 627
 - DNS 59
 - IP 59
 - sport 89
 - Spyware 172
 - SQL-Injektion 57, 75
 - Squid 119, 419, 526
 - src 86
 - SSH 176, 181
 - SSL *siehe* Secure-Socket-Layer
 - Stackguard 69
 - star 164
 - state 373
 - Stateful Inspection 539, 545, 549, 550
 - Stealth-Scans 336
 - Steuererklärung 529
 - STP *siehe* Spanning-Tree-Protokoll
 - Stratum-1-Server 242
 - Stream Control Transmission Protocol 372
 - string 373
 - strongSwan 560
 - stunnel 223
 - Subversion 384
 - sudo 153
 - SUSE 218, 220, 252, 274, 298
 - SUSEfirewall2 298
 - Sweep-Scan 333
 - SYN 106, 402
 - Cookies 59
 - Flood 58
 - RECV 410
 - SENT 410
 - syn 108, 361, 472
 - SYN_SENT 116
 - SYN/ACK 402
 - syncdev 481
 - SYN-Cookies 442
 - sysctl 94
 - Syslog 137
 - Syslogd 60, 215
 - BSD 219, 220
 - BSD-Syslogd 218
 - Modular Syslog 219
 - Msyslogd
 - PEO 226
 - Secure Syslogd 219
 - Syslog-ng 228
 - Syslog-ng 137
 - system-config-securitylevel 297
 - SysVInit 149
- T
- talk-contrack-nat 399
 - tar 164
 - TARPIT 394
 - tc 421
 - TCP 224, 228, 401, 604, 606, 608
 - Acknowledgment Timer 617
 - Acknowledgement-Nummer 612
 - ACK-Scan 337
 - Congestion Window 617
 - Congestion Avoidance 617, 618
 - Congestion Window 617, 620
 - Congestion Window Reduced 620
 - Connect-Scan 326
 - ECN-Echo 620
 - FIN-Scan 336
 - Flag 613
 - ACK 613, 620
 - FIN 614
 - PSH 613
 - RST 613
 - SYN 64, 613, 620
 - URG 613, 615
 - Flusskontrolle 616
 - Handshake 609
 - Header 611
 - Länge 612
 - MRU 615
 - Optionen 615
 - End of Option List 615
 - MSS 615
 - No Operation 615
 - Selective Acknowledgment
 - Data 616
 - Selective Acknowledgment
 - Permitted 616
 - Timestamp 616
 - Window Scale 614, 616
 - port 611
 - Prüfsumme 614, 618
 - Receive Window 614, 617
 - Reno 444
 - reservierte Bits 613
 - Reset 622
 - Segmentation-Offload 443
 - Selective Acknowledgment 618
 - Sequenznummer 65, 609, 612, 618
 - Slow Start 617
 - SYN-Scan 328
 - Timestamps 331
 - Tuning-Guide 442

- Urgent-Zeiger 615
 - Vegas 444
 - Vollduplex 610
 - Westwood 445
 - Window-Scaling 445
 - Window-Tracking 406
 - Wrappers 158
 - tcp_abort_on_overflow 436
 - tcp_adv_win_scale 436
 - tcp_app_win 436
 - tcp_congestion_control 436
 - tcp_dsack 437
 - tcp_ecn 135, 437
 - tcp_fack 437
 - tcp_fin_timeout 438
 - tcp_frto 438
 - tcp_keepalive_intv 438
 - tcp_low_latency 439
 - tcp_max_orphans 439
 - tcp_max_syn_backlog 439
 - tcp_max_tw_buckets 439
 - tcp_mem 439
 - tcp_moderate_rcvbuf 440
 - tcp_no_metrics_save 440
 - tcp_orphan_retries 440
 - tcp_reordering 441
 - tcp_retrans_collapse 441
 - tcp_retries1 441
 - tcp_retries2 441
 - tcp_rfc1337 441
 - tcp_rmem 436, 441
 - tcp_sack 442
 - tcp_stdurg 442
 - tcp_syn_retries 443
 - tcp_synack_retries 442
 - tcp_syncookies 135, 442
 - tcp_timestamps 332, 443
 - tcp_tso_win_divisor 443
 - tcp_tw_* 444
 - tcp_vegas_* 444
 - tcp_westwood 445
 - tcp_window_scaling 445
 - tcp_wmem 445
 - Tcpdump 319
 - tcp-flags 361
 - TCPMSS 380
 - tcpmss 373
 - Telnet 530
 - TFTP 549
 - time 390
 - TIME_WAIT 116, 264, 409, 410, 441, 444
 - time-exceeded 570, 577
 - Timestamp 334
 - timestamp-reply 570, 579
 - timestamp-request 570, 579
 - Time-to-Live 132, 382, 433, 577
 - TIS Firewall-Toolkit 39
 - TLS *siehe* Transport Layer Security
 - Token-Ring 501, 572
 - TOS 132, 382, 421, 423
 - tos 374
 - to-source 517
 - TPROXY 412, 419
 - tproxy 399
 - TRACE 133, 395, 425
 - Traceroute 582
 - traceroute 424, 569, 603, 623
 - Transmission Control Protocol *siehe* TCP
 - transparenter Proxy 399
 - Transport Layer Security 547, 551
 - Transport-Modus 563
 - Tripwire 166
 - Trivial File Transfer Protocol *siehe* TFTP
 - Trojaner 172
 - TSIG 524
 - TTL 421, 424
 - TTL *siehe* Time-To-Live
 - ttl 374
 - Tunnel-Modus 563
 - Type-of-Service 132, 374, 421
- U
- u32 390
 - UDP 60, 218, 224, 228, 401, 403, 604, 606
 - Destination Port 607
 - Header 607
 - Länge 607
 - Prüfsumme 608
 - Scan 332
 - Source Port 607
 - Uhrzeit 390
 - ULOG 136, 265, 267, 270, 289, 382, 451, 452
 - ulog-acctd 267
 - Ulogd 270, 452
 - unclean 399
 - Unicast 515
 - Unix-Socket 217
 - UNTRACKED 111, 373, 402
 - USENIX 41
 - User Datagram Protocol *siehe* UDP
 - User-Mode-Linux 90, 92, 205

V

- v 86
- Variable 91, 92, 322
- Verbindungstabelle 42
- verbose 86
- Verfügbarkeit 55
- Versionskontrollsystem 274
- Vertraulichkeit 55
- Virenschanner 48
- Virtual Router Redundancy Protocol 473
- Virus 51
 - Scanner 42
- Vixie, Paul 38
- VLAN 516
- VMware 205, 308
- Voice over IP 556
- VoIP *siehe* Voice over IP
- VPN 544

W

- War Dialing 57
- Webfwlog 265
- Webserver 187
- Welte, Harald 267, 452

- Wesslowski, Boris 251
- WLAN 438
- wrapper 229

X

- X.509 526
- Xen 205
- xinetd 148
- Xmas-Scan 336
- XOR 395, 421, 424

Y

- Yarochkin, Fyodor 621
- YaST 299
 - Online-Update 147

Z

- Zeitsynchronisation 241
- Zombie 339
- ZORP 419
- Zorp 228
- Zustandstabelle 402
- Zwangstrennung 415



Über den Autor

Ralf Spenneberg ist seit vielen Jahren als freier Trainer, Berater und Autor im Linux- und Unix-Umfeld tätig. Das vorliegende Buch ist aus verschiedenen Kursen und Workshops entstanden, die er im Laufe der Jahre gehalten hat. Nach dem Studium der Biochemie war er lange Zeit als Systemadministrator tätig. Anschließend begann er bereits 1998, Schulungen für Linux und Unix anzubieten. Seit Januar 2000 ist er zertifiziert als RHCE und RHCX und führt auch für Red Hat in Deutschland und England Schulungen durch. Im Rahmen dieser Zusammenarbeit hat er für Red Hat zwei Kurse zu den Themen Firewalling und VPN entwickelt. Weitere Informationen bietet er auf seiner Homepage <http://www.spenneberg.com> an. Wenn Sie eine Schulung bei ihm besuchen möchten, können Sie sich auf <http://www.opensource-training.de> informieren.