



# 7 Intrusion Detection selfmade

Einfache Intrusion-Detection-Systeme können mit Linux-Hausmitteln selbst gebaut werden. Diese einfachen Lösungen zeigen sehr gut, wie auch professionelle Systeme arbeiten und funktionieren.

## 7.1 Ausnutzen der Protokollfunktionen allgemeiner Dienste

Bevor ein komplettes Intrusion-Detection-System eingesetzt wird, sollte auch die Konfiguration der bereits vorhandenen Netzwerkdienste überprüft werden. In vielen Fällen kann durch eine kluge Anwendung der normalen Protokollfunktionen dieser Dienste bereits eine zielgerichtete und sehr genaue Alarmierung bei einem Angriff erzeugt werden. Viele Dienste ermöglichen es heute, die klassischen Probleme unsicherer Protokolle zu umgehen. Hier haben die Linux-Distributoren in den letzten Jahren auch viel dazugelernt. Viele der Linux-Distributionen sind nun bereits bei einer Standardinstallation relativ sicher. Die installierten Dienste werden nicht, wie noch vor wenigen Jahren, alle automatisch gestartet. Ein Zugriff auf die Dienste und mögliche Sicherheitslücken können meist protokolliert werden. Auch diese Funktion ist bei modernen Distributionen oft aktiviert. Dies kann bereits als eine Form der Intrusion Detection gewertet werden. Je spezieller der Dienst und die angegriffene Sicherheitslücke sind, desto unwahrscheinlicher ist auch ein Fehlalarm. Viele Dienste und Werkzeuge erlauben auch die Protokollierung ungewöhnlichen Verhaltens. So erlauben viele Dienste die Einschränkung ihrer Ressourcen oder den Zugriff auf besondere Funktionen. Ein Erreichen dieser Grenzen oder ein Zugriff auf diese Funktionen kann als ein ungewöhnlicher Zustand ausgelegt werden. Werden diese Ereignisse protokolliert, so kann oft festgestellt werden, ob es sich um eine erlaubte Nutzung dieses Dienstes handelte oder ob gerade ein Denial-of-Service-(DoS)-Angriff stattfand.

Der folgende Abschnitt wird kurz zwei Beispiele vorstellen. Im ersten Beispiel wird gezeigt, wie durch eine sinnvolle Konfiguration des DNS-Nameservers Bind 9 außergewöhnliche Anfragen protokolliert werden können. Das zweite Beispiel zeigt, wie mit dem Linux-Paketfilter Netfilter ein SYN-Flooding erkannt werden kann. Hierbei handelt es sich um einen Denial-of-Service-Angriff, bei dem das Zielsystem mit sehr vielen SYN-Paketen pro Zeiteinheit überschwemmt wird.

### 7.1.1 Konfiguration des DNS-Nameservers Bind 9

Ein DNS-Server ist verantwortlich für die Auflösung von DNS-Namen in IP-Adressen. Jedes Mal, wenn ein Benutzer eine URL in seinem Browser eingibt, wird dieser einen DNS-Server kontaktieren, um die entsprechende IP-Adresse zu ermitteln. Diesen Vorgang bezeichnet man als *Query*. Um die Verwaltung des DNS-Namensraumes delegieren zu können, wird er in einer verteilten Datenbank aus mehreren tausend DNS-Servern verwaltet. Hierbei besteht die Möglichkeit, dass ein DNS-Server, der für den Namensraum \*.de zuständig ist, die einzelnen weiteren Domänen (z.B. \*.deutschland.de) zur Verwaltung an weitere DNS-Server abgibt. Damit dennoch jeder DNS-Server den gesamten Namensraum auflösen kann, sind die DNS-Server in der Lage, rekursiv die IP-Adresse eines DNS-Namens zu ermitteln. Hierbei kontaktiert der Browser des Benutzers den eigenen DNS-Server (oder den des Providers). Dieser prüft zunächst, ob er bereits selbst die IP-Adresse kennt. Ansonsten kontaktiert er rekursiv den DNS-Server, der für diesen DNS-Namen zuständig ist (Reursion). Diese Funktion ist erforderlich, wenn Benutzer den DNS-Server verwenden, um auf das Internet zuzugreifen. Wenn der DNS-Server aber nur verwendet wird, um eine Domäne zu verwalten, benötigt er diese Funktion nicht. Schließlich ist das DNS-System derartig wichtig für die Funktion des Internets, dass es ausfallsicher implementiert wird. Hierzu muss jede Domäne von mindestens zwei DNS-Servern verwaltet werden. Diese Rechner sollten sich in unterschiedlichen Netzen befinden, so dass ein Netzausfall nicht beide DNS-Server betrifft. Einer dieser Rechner wird als primärer DNS-Server (Master) implementiert. Alle weiteren DNS-Server werden als sekundäre DNS-Server (Slave) bezeichnet und kopieren die kompletten Informationen des primären DNS-Servers. Dieser Vorgang wird als Transfer bezeichnet.

Aus Sicherheitsgründen können und sollen diese Funktionen eingeschränkt werden. Hierzu bietet der DNS-Server Bind 9 Funktionen an, die als Access Control Lists bezeichnet werden. Hiermit können die erlaubten Funktionen eingeschränkt werden.

Die folgenden Zeilen definieren zunächst zwei Gruppen:

```
// our-nets sind die eigenen Clients
acl our-nets {192.168.0.0/24;};
// sec-dns sind die beiden sekundären Nameserver
acl sec-dns {192.168.0.101/32; 10.0.5.1/32;}
```

Mithilfe dieser Gruppen können nun die Funktionen eingeschränkt werden.

```
options {
    directory "/var/named";
    allow-recursion { our-nets; };
    allow-query { our-nets; };
    allow-transfer { sec-dns; };
};
```

Nun kann die Protokollierung noch speziell angepasst werden. Hierzu bietet Bind 9 die *logging*-Direktive:

```
logging {
    channel "intrusion" {
        file "intrusion.log" versions 3 size 20m;
        print-time yes;
    };
    category "security" { "intrusion"; };
};
```

Nun wird der DNS-Server unerlaubte Anfragen protokollieren. Diese Protokolle haben dann folgenden Inhalt:

*Listing 7.1 Datei /var/named/intrusion.log*

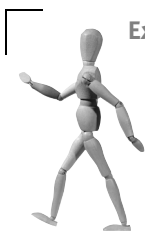
```
Sep 21 12:39:50.286 client 192.168.5.4#32781: query 'nohup.info/IN' denied
Sep 21 12:41:00.222 client 192.168.5.3#33073: zone transfer 'nohup.info/IN'
↳ denied
```

Diese Informationen geben bereits Aufschluss über die verschiedenen Clients, die verbotene Anfragen an den DNS-Server stellen. Die Implementierung der Regeln kann wesentlich feinfühlicher und für jede einzelne Domäne (Zone) einzeln erfolgen.

### 7.1.2 Definition eines dDoS-Schwellwertes mit iptables

Verteilte DoS-Angriffe (distributed Denial of Service, dDoS) zeichnen sich durch eine große Anzahl von Paketen in kurzen Zeitabständen aus. Der Linux-Paketfilter Netfilter bietet die Möglichkeit, derartige Ereignisse zu protokollieren. Hierzu bedient man sich der Erweiterung `--match limit` mit der Option `--limit`. Handelt es sich bei dem überwachenden Rechner um einen Webserver mit geringer Auslastung, so kann wahrscheinlich bei 100 SYN-Paketen pro Sekunde bereits von einem Angriff ausgegangen werden. Dies stellt einen ungewöhnlichen Zustand dar. Eine Protokollierung dieser Tatsache kann mit folgendem Befehl erreicht werden:

```
iptables -A FORWARD --protocol tcp --dport 80 --match limit --limit
100/second \ -j
LOG --log-level 2 --log-prefix "SYN-ATTACK: "
```



#### Exkurs: Netfilter/iptables

Netfilter bietet drei verschiedene Tabellen (*tables*). Diese drei Tabellen beinhalten die unterschiedlichen Funktionen: *filter*, Network Address Translation (*nat*) und *mangle* (Paketmodifikationen). Firewalling-Funktionen werden in der Tabelle *filter* zur Verfügung gestellt. Diese Tabelle bietet drei verschiedene Ketten: INPUT, OUTPUT

und FORWARD. Jede dieser drei Ketten betrachtet eine bestimmte Menge von Paketen:

- INPUT: Pakete, die an den Rechner gerichtet sind
- FORWARD: Pakete, die durch den Rechner weitergeleitet werden (geroutete Pakete)
- OUTPUT: Pakete, die vom Rechner erzeugt werden

Einer klassischen Firewall genügt es daher vollkommen, lediglich die FORWARD-Kette zu betrachten. Sollen zusätzlich die Pakete gefiltert werden, die an die Firewall selbst gerichtet sind oder von dieser erzeugt werden, so müssen auch die Ketten INPUT und OUTPUT betrachtet werden.

Die Regeln der Firewall werden mit dem Kommando `iptables` erzeugt und verwaltet. Dieses Kommando bietet sehr viele Optionen und Argumente. Eine Auswahl wird im Folgenden kurz erklärt:

- `-t table`: Angabe der Tabelle. Default ist `filter`.
- `-A KETTE`: Anhängen einer Regel an die angegebene Kette
- `--protocol protokoll`: Diese Regel betrifft nur Pakete mit dem angegebenen IP-Protokoll.
- `--destination Adresse`: Diese Regel betrifft nur Pakete mit der angegebenen Zieladresse.
- `--dport Port`: Diese Regel betrifft nur Pakete mit dem angegebenen Port (TCP oder UDP). Die Angabe des Protokolls ist obligatorisch.
- `-j Target`: Führe diese Aktion aus, wenn die Regel auf ein Paket zutrifft. Einige Ziele, zum Beispiel `LOG`, bieten hierbei noch die Möglichkeit, die Aktion weiter zu spezifizieren.
- `--match Erweiterung`: Lade die entsprechende Erweiterung.

In einer Regel können nun mehrere Optionen spezifiziert und mehrere Testerweiterungen geladen werden. Nur wenn alle Optionen zutreffen, wird die Aktion der Regel ausgeführt.

Wenn derartige Anfragen sporadisch vorkommen, kann zunächst auch ein weiterer Schwellwert definiert werden, bevor diese Regel zum Tragen kommt.

```
iptables -A FORWARD --protocol tcp --dport 80 --match limit --limit-burst 120 \
limit 100/second -j LOG --log-level 2 --log-prefix "SYN-ATTACK: "
```

Nun müssen zunächst 120 Pakete gezählt werden. Wenn anschließend die Rate gleichmäßig mindestens 100 Pakete pro Sekunde beträgt, wird die Protokollmeldung ausgegeben.

Weitere Informationen zum Kommando `iptables` und zum Paketfilter `Netfilter` können Sie dem Buch `Linux-Firewalls` von Robert Ziegler entnehmen, welches im Markt+Technik-Verlag erschienen ist (siehe Literaturhinweise auf S. 817).

## 7.2 Einfache lokale IDS

Die Aufgabe eines Host-Intrusion-Detection-Systems ist es, Einbrüche zu erkennen und zu melden. Viele Systeme erreichen diese Funktion durch eine Überwachung der Systemdateien. Werden Änderungen an den Systemdateien festgestellt, so schlagen diese Systeme Alarm. `Tripwire` führt zum Beispiel solche Integritätstests durch. Bestehen nicht so hohe Anforderungen an das IDS, so können einfache Systeme auch mit den unter Linux verfügbaren Befehlen `find` und `diff` implementiert werden. Diese Systeme weisen natürlich nicht ähnlich viele Funktionen und Sicherheitseigenschaften auf. Dennoch lässt sich an derartig handgestrickten Lösungen recht gut die Funktionsweise der großen integrierten IDS nachvollziehen.

Die Implementierung eines einfachen rechnergestützten IDS soll hier erläutert werden. Dieses IDS soll die Integrität der Dateien in den Verzeichnissen `/etc`, `sbin` und `bin` überprüfen.

Um zunächst den *Ist*-Zustand des Systems einzufangen, wird mit dem `find`-Befehl eine Baseline angelegt:

```
# find /etc -type f -exec ls -ail {} \; > /root/baseline.2002-07-08
# less /root/baseline.2002-07-08
    ... gekürzt ...
↳ 99141 -rw-r--r-- 1 root root 12 Nov 4 2001
↳ /etc/pam_smb.conf
  99133 -rw-r--r-- 1 root root 7172 Apr 19 05:53 /etc/screenrc
  99135 -r--r----- 1 root root 580 Apr 18 18:35 /etc/sudoers
↳ 99139 -rw-r--r-- 1 root root 5803 Jul 20 2001
↳ /etc/ltrace.conf
↳ 99127 -rw-r--r-- 1 root root 1913 Jun 24 2001
↳ /etc/mtools.conf
    ... gekürzt ...
```

In der Datei `baseline.2002-07-08` ist nun jeder Dateieintrag (`find -type f`) des Verzeichnisses `/etc` mit seiner Inode-Nummer (`-i`), den Rechten, dem Besitzer, der Größe und dem Änderungsdatum (`-l`) abgespeichert. Hierzu ruft der Befehl `find` für jede gefundene Datei den Befehl `ls -ail` auf und übergibt ihm den Namen der gefundenen Datei `{}`. Diese Datei kann nun auf einer Diskette mit Schreibschutz gespeichert werden und anschließend in regelmäßigen Abständen mit dem Systemzustand verglichen werden. Dieser Vergleich erfolgt sinnvollerweise mit dem `diff`-Befehl. Dazu wird die Diskette gemountet und die Datei verglichen.

```
# mount /mnt/floppy
# find /etc -type f -exec ls -ail {} \; | \
> diff /mnt/floppy/baseline.2002-07-08 - > aenderungen.txt
```

Wurde nach Erzeugung der Baseline ein Benutzer angelegt, so enthält die Datei *aenderungen.txt* folgende Einträge:

```
326c326
< 99212 -rw-r--r-- 1 root root 693 Jul 9 13:01 /etc/group
---
> 99211 -rw-r--r-- 1 root root 706 Jul 9 13:02 /etc/group
332c332
< 100538 -rw-r--r-- 1 root root 1766 Jul 9 13:01 /etc/passwd
---
> 100533 -rw-r--r-- 1 root root 1805 Jul 9 13:02 /etc/passwd
574c574
< 96744 -rw----- 1 root root 681 Jun 7 16:15 /etc/group-
---
> 96744 -rw----- 1 root root 693 Jul 9 13:01 /etc/group-
1275,1276c1275,1276
< 100534 -rw----- 1 root root 1729 Jun 7 16:15 /etc/passwd-
< 99150 -rw----- 1 root root 1229 Jun 7 16:15 /etc/shadow-
---
> 100534 -rw----- 1 root root 1766 Jul 9 13:01 /etc/passwd-
> 99150 -rw----- 1 root root 1256 Jul 9 13:01 /etc/shadow-
1278,1279c1278,1279
< 99211 -r----- 1 root root 1256 Jul 9 13:01 /etc/shadow
< 99191 -r----- 1 root root 569 Jul 9 13:01 /etc/gshadow
---
> 100538 -r----- 1 root root 1284 Jul 9 13:02 /etc/shadow
> 99212 -r----- 1 root root 579 Jul 9 13:02 /etc/gshadow-
1283c1283
< 99151 -rw----- 1 root root 560 Jun 7 16:15 /etc/gshadow-
---
< 99151 -rw----- 1 root root 569 Jul 9 13:01 /etc/gshadow-
```

In dieser Datei werden nun alle Dateien aufgeführt, deren Informationen sich geändert haben. Hierbei handelt es sich um die Dateien */etc/group*, */etc/passwd*, */etc/group-*, */etc/passwd-*, */etc/shadow*, */etc/shadow-*, */etc/gshadow* und */etc/gshadow-*.

Soll dieses System nun auf die oben angesprochenen Verzeichnisse ausgedehnt werden, so ist nur der Befehl *find* für die entsprechenden Verzeichnisse aufzurufen, um die Baseline zu erzeugen.

```
# find /etc /sbin /bin -type f -exec ls -ail {} \; \ /root/baseline.2002-07-08
```

Ein Integritätstest erfolgt dann mit:

```
# find /etc /sbin /bin -type f -exec ls -ail {} \; | \
diff /mnt/floppy/baseline.2002-07-08 - > aenderungen.txt
```

Ein Angreifer kann dieses System jedoch noch leicht aushebeln. Hierzu bieten sich ihm zwei Möglichkeiten. Er erzeugt die Datei, in der die Baseline gespeichert wird, neu. Wurde die Datei auf einer Diskette mit Schreibschutz abgespeichert, so kann er die Diskette aushängen und im verbleibenden Verzeichnis die Datei neu erzeugen. Überprüft der Systemadministrator nicht vor jedem Integritätstest den Mount-Zustand der Diskette, so wird der Vergleich mit der modifizierten Datei durchgeführt.

Eine zweite Gefahr stellt die Tatsache dar, dass der Angreifer in der Lage ist, eine Datei so zu verändern, dass der Befehl `ls -ail`, der die Eigenschaften der Datei ausliest, keine Modifikation anzeigt. Hierzu muss die veränderte Datei im gleichen Inode abgespeichert werden, das gleiche Änderungsdatum und die gleiche Größe aufweisen. Der Editor `vi` ist ein Editor, der Änderungen an einer Datei durchführt, ohne ihren Inode zu ändern. Der Befehl `touch` ist in der Lage, einer Datei einen willkürlichen Zeitstempel zuzuweisen. Schließlich können häufig Bereiche (z.B. Kommentare) aus Dateien ohne weitere Probleme gelöscht werden, um so die Originalgröße wiederherzustellen. Um derartige Modifikationen zu erkennen, sollte zusätzlich von jeder Datei eine Prüfsumme erzeugt werden und beim Integritätstest verglichen werden. Das kann sehr einfach mit dem Befehl `md5sum` erfolgen. Dieser Befehl berechnet einen kryptografischen MD5 Hash (siehe Anhang D, »Hash-Algorithmen« auf S. 784).

```
# find /etc /sbin /bin -type f -exec ls -ail {} \; | \
-exec md5sum {} \; > /root/baseline.2002-07-08
```

Nun enthält die Baseline-Datei zusätzlich auch den MD5 Hash für jede Datei. Dieser kann bei einem Integritätstest überprüft werden.

```
# find /etc /sbin /bin -type f -exec ls -ail {} \; -exec md5sum {} \; | \
> diff /mnt/floppy/baseline.2002-07-08 - > aenderungen.txt
```

## 7.3 Einfache netzwerkbasierte IDS

Die Aufgabe eines Network Intrusion Detection Systems (NIDS) ist es, Einbrüche zu erkennen und zu melden. Viele Systeme erreichen diese Funktion durch eine Überwachung der Netzwerkaktivitäten. Werden ungewöhnliche Dienste genutzt oder enthalten die Netzwerkpakete verbotene Informationen, so lösen diese Systeme einen Alarm aus. Snort überprüft so zum Beispiel alle Netzwerkpakete. Bestehen nicht so hohe Anforderungen an das IDS, so können derartige Systeme auch mit den unter Linux verfügbaren Befehlen implementiert werden. Diese Systeme weisen natürlich nicht ähnlich viele Funktionen und Sicherheitseigenschaften auf, dennoch lässt sich an derartig handgestrickten Lösungen recht gut die Funktionsweise der großen integrierten IDS-Lösungen nachvollziehen.

Es werden die Werkzeuge *tcpdump* und *ngrep* vorgestellt.

### 7.3.1 Tcpcdump

*tcpdump* ist inzwischen fester Bestandteil der meisten Distributionen geworden. Es wird unter der GPL-Lizenz vertrieben. Die Entwicklung von *tcpdump* wurde von der Network Research Group (NRG) der Lawrence Berkeley National Laboratory (LBNL) betrieben. Diese hat jedoch die weitere Entwicklung eingestellt. Neuere, erweiterte Versionen können nun auf der inoffiziellen Homepage <http://www.tcpdump.org> erstanden werden.

Im Folgenden soll ein NIDS für einen Webserver entwickelt werden. Hierbei wird davon ausgegangen, dass der Webserver lediglich seine Dienste über den Port http (80) und https (443) zur Verfügung stellt. Die Administration und Pflege erfolgt lokal. Eine Fernadministration ist nicht vorgesehen.

Der Rechner, der für die Überwachung des Webservers eingesetzt wird, muss sich in demselben Netzwerk befinden, in dem auch der Webserver lokalisiert ist. Dieses Netzwerk darf nicht von einem Switch verwaltet werden. Wenn ein Switch eingesetzt wird, so muss der Überwachungsrechner an einen so genannten Monitor- oder Spanning-Port angeschlossen werden. Die Dokumentation des Switches wird weitere Informationen liefern.

Der Befehl *tcpdump* weist nun eine Vielzahl von Optionen auf. Im Folgenden können nicht alle verfügbaren Optionen besprochen werden. Das würde den Umfang dieses Kapitels sprengen. Es soll lediglich an dem angesprochenen Beispiel des Webservers gezeigt werden, wie *tcpdump* eingesetzt werden kann. Einzelne Optionen, insbesondere die Berkeley-Paketfilter-Syntax (BPF-Filter), werden auch von anderen Werkzeugen wie *ngrep* und *snort* unterstützt. Werden weitere Informationen benötigt, sollte zunächst die Manpage konsultiert werden.

Ein einfacher Aufruf von *tcpdump* fordert das Programm auf, mithilfe der *libpcap*-Bibliothek (*pcap = Packet capture*) Netzwerkpakete zu sammeln und ihre Header anzuzeigen. Dabei arbeitet *tcpdump* automatisch im *promiscuous*-Modus. Das bedeutet, dass es sämtliche Pakete im Netzwerk anzeigt, einschließlich der Pakete, die nicht für den Rechner bestimmt sind.

```
# tcpdump
tcpdump: listening on eth1 07:45:58.371618 kermit.32794 > search.ebay.de.http:
↳ S 1900772718:1900772718(0) win 5840 <mss 1460,sack0K,timestamp 467536
↳ 0,nop,wscale 0> (DF)
↳ 07:45:58.564903 search.ebay.de.http > kermit.32794:
↳ S 4032051610:4032051610(0)
↳ ack 1900772719 win 8767 <mss 1460>
07:45:58.564995 kermit.32794 > search.ebay.de.http: . ack 1 win 5840 (DF)
↳ 07:45:58.565219 kermit.32794 > search.ebay.de.http: P 1:1186(1185)
↳ ack 1 win 5840 (DF)
07:45:58.800691 search.ebay.de.http > kermit.32794: . ack 1186 win 23431
↳ 07:45:59.573594 search.ebay.de.http > kermit.32794:
```

```

↳ P 1:119(118) ack 1186 win 24616
07:45:59.573670 kermit.32794 > search.ebay.de.http: . ack 119 win 5840 (DF)

```

Es wird nun kurz das Ausgabeformat von *tcpdump* erläutert. Hierbei beschränken wir uns auf das einfache Ausgabeformat bei IP-Paketen. In einigen Fällen wird auch hier die Ausgabe anders gestaltet sein. *tcpdump* ist in der Lage, DNS- und NFS-Kommunikation zu dekodieren. Wenn sich bei der Besprechung des Ausgabeformates Lücken in Ihrem TCP/IP-Wissen auftun, sollten Sie die entsprechenden Informationen im Anhang oder in den Literaturhinweisen auf Seite 817 und 818 genannten Titeln nachlesen.

Die erste Information, die von *tcpdump* ausgegeben wird, ist der Zeitstempel. Dieser erlaubt es, die Pakete zeitlich zuzuordnen. Dafür ist es jedoch wichtig, dass die Uhrzeiten der Rechnersysteme abgeglichen sind, damit Netzwerkvorkommnisse lokalen Ereignissen zugeordnet werden können (siehe Abschnitt »Zeitsynchronisation« auf S. 572).

#### Listing 7.2 *tcpdump*-Ausgabe: Zeitstempel

```

07:45:58.371618 kermit.32794 > search.ebay.de.http: S 1900772718:1900772718(0)
↳ win 5840 <mss 1460,sackOK,timestamp 467536 0,nop,wscale 0> (DF)

```

Der zweite Abschnitt in der Ausgabe zeigt die beiden Kommunikationspartner einschließlich der verwendeten Ports an. In diesem Fall kommuniziert *kermit* über seinen Port 32794 mit *search.ebay.de* Port http. *tcpdump* versucht die Namen der Kommunikationspartner über einen Reverse-DNS-Lookup aufzulösen. Diese Funktion lässt sich mit `-n` abschalten. Zusätzlich werden auch die Ports mithilfe der Datei `/etc/protocols` durch ihre Namen ersetzt, wenn das möglich ist. Diese Funktion lässt sich mit `-nn` abschalten.

#### Listing 7.3 *tcpdump*-Ausgabe: Kommunikationspartner

```

07:45:58.371618 kermit.32794 > search.ebay.de.http: S 1900772718:1900772718(0)
↳ win 5840 <mss 1460,sackOK,timestamp 467536 0,nop,wscale 0> (DF)

```

Im Weiteren kann in dieser Zeile erkannt werden, dass es sich um eine TCP-Information handelt. Im Falle eines normalen UDP-Paketes würde im Anschluss an die Kommunikationspartner die Information `udp` folgen.



#### Achtung:

*tcpdump* ist in der Lage, viele Protokolle zu interpretieren. DNS ist ein derartiges Protokoll. In dem Fall werden die übertragenen DNS-Informationen angezeigt. Eine Auflösung des Namens *www.pearson.de* ergibt folgende *tcpdump*-Ausgabe:

```

08:19:37.089352 kermit.32770 > DNS.SERVER.DE.domain: 25840+ A?
↳ www.pearson.de. (32) (DF)
↳ 08:19:37.209113 DNS.SERVER.DE.domain > kermit.32770: 25840*
↳ 1/2/2 A 194.163.213.75 (127)

```

Der Rechner *kermit* kontaktiert den DNS-Server und stellt eine Anfrage (A?). Diese Anfrage erhält die Identifikationsnummer 25840, damit der Client später die Antwort zuordnen kann. Möglicherweise stellt er ja mehrere Anfragen, bevor der Server antwortet. Die Anfrage enthält den Namen des zu ermittelnden Rechners *www.pearson.de*.

Wenige Sekundenbruchteile später antwortet der DNS-Server auf die Anfrage 25840 mit einem Answer Record, zwei Nameserver Records und zwei Authority Records (1/2/2). Der erste Record ist ein Address-Record mit dem Wert 194.163.213.75. Die Gesamtlänge der Anfrage war 127 Bytes lang.

*tcpdump* ist bereits in der Lage, außer UDP DNS-Anfragen auch SMB/CIFS- und NFS-Anfragen zu dekodieren und anzuzeigen. □

In unserem Beispiel handelt es sich jedoch nicht um ein UDP-Paket, sondern um ein TCP-SYN-Paket, mit dem eine TCP-Verbindung aufgebaut werden kann. Dies ist das erste Paket, in dem der Client dem Server seine Sequenznummer übermittelt, um seine Seite der Verbindung zu synchronisieren (siehe Abschnitt »Auf- und Abbau einer TCP-Verbindung« auf S. 735).

#### Listing 7.4 *tcpdump*-Ausgabe: SYN-Bit

```
07:45:58.371618 kermit.32794 > search.ebay.de.http: S 1900772718:1900772718(0)
↳ win 5840 <mss 1460,sackOK,timestamp 467536 0,nop,wscale 0> (DF)
```

Die Sequenznummer ist 1900772718. Der Client überträgt in diesem Paket keine Daten: (0)

#### Listing 7.5 *tcpdump*-Ausgabe: Sequenznummern

```
07:45:58.371618 kermit.32794 > search.ebay.de.http: S 1900772718:1900772718(0)
↳ win 5840 <mss 1460,sackOK,timestamp 467536 0,nop,wscale 0> (DF)
```

Damit der Server weiß, wie viele Informationen der Client verarbeiten kann, übermittelt der Client seine TCP Window-Größe. Dieses Fenster definiert die maximale Datenmenge, die der Client in seinem Empfangspuffer speichern kann. Der Server muss nach der angegebenen Menge zunächst auf ein Acknowledgment des Clients warten, bevor weitere Daten gesendet werden.

#### Listing 7.6 *tcpdump*-Ausgabe: TCP-Optionen

```
07:45:58.371618 kermit.32794 > search.ebay.de.http: S 1900772718:1900772718(0)
↳ win 5840 <mss 1460,sackOK,timestamp 467536 0,nop,wscale 0>> (DF)
```

Für den Aufbau der Verbindung übermittelt der Client einige weitere Informationen. Diese werden als TCP-Optionen übermittelt.

*Listing 7.7 tcpdump-Ausgabe: TCP-Optionen*

```
07:45:58.371618 kermit.32794 > search.ebay.de.http: S 1900772718:1900772718(0)
↳ win 5840 <mss 1460,sackOK,timestamp 467536 0,nop,wscale 0> (DF)
```

Zusätzlich verlangt der Client, dass das Paket auf seinem Weg zum Server nicht von Routern fragmentiert werden darf ((DF)). Wenn ein Router das Paket aufgrund einer zu kleinen Maximum Transmission Unit (MTU) nicht weiter transportieren kann, muss er eine Fehlermeldung (*ICMP-Destination unreachable: Fragmentation needed*) zurücksenden.

Das zweite Paket in der angegebenen Paketfolge stellt die Bestätigung dieses ersten SYN dar. Es enthält zusätzlich zu diesen Angaben eine Acknowledgment-Nummer, die den Empfang des ersten Paketes bestätigt. Alle weiteren Pakete enthalten Sequenz- und Acknowledgment-Nummern. Diese werden von *tcpdump* automatisch relativ berechnet, so dass die Umrechnung nicht mehr selbst erfolgen muss. Wird dieses Verhalten nicht gewünscht (oder gar vermutet, dass *tcpdump* hier Fehler macht), kann es abgeschaltet werden (-S).

```
07:45:58.564903 search.ebay.de.http > kermit.32794: S 4032051610:4032051610(0)
↳ ack 1900772719 win 8767 <mss 1460>
```

Das sind bereits alles sehr interessante Informationen, jedoch werden ein Großteil der Pakete vollkommen normal sein und keinerlei Hinweise auf einen Einbruch enthalten. Dennoch existieren viele Organisationen, die den kompletten Verkehr in ihren Netzwerken mit *tcpdump* protokollieren. Im Falle eines Einbruchs kann dann häufig zurückverfolgt werden, wie dieser erfolgte und welche Vorbereitungen der Angreifer traf. Zusätzlich lassen sich aus diesem Wissen später Regeln für IDS-Systeme ableiten, die diesen Angriff vielleicht noch nicht kannten und erkennen konnten. Hierbei sollte jedoch die Privatsphäre der Benutzer beachtet werden. *tcpdump* bietet hierfür die Option *snaplen* (-s). Diese Option gibt die Anzahl der zu protokollierenden Bytes eines Paketes an, sozusagen die Länge. Der Defaultwert beträgt 68. Um die Pakete nun protokollieren zu können, kann *tcpdump* die Pakete in einer Datei (-w file) abspeichern. Diese Datei wird im so genannten *libpcap*-Format erzeugt. So besteht die Möglichkeit, den Inhalt später mit *tcpdump*, *snort* oder auch *ethereal* zu lesen und auf alle Informationen innerhalb der Snaplen zuzugreifen.

```
# tcpdump -s 100 -w tcpdump.log
```

Ein Lesen der Datei mit *tcpdump* erfolgt mit der Option -r file. Wenn hierbei der Inhalt der Pakete angezeigt werden soll, so wie er aufgezeichnet wurde, kann die Option -X angegeben werden. *tcpdump* gibt dann den Inhalt sowohl hexadezimal als auch in ASCII aus. Hierbei wird jedoch nur der Inhalt des IP-Paketes ausgegeben. Wenn auch der Ethernet Header mit den MAC-Adressen angezeigt werden soll, kann dies mit der Option -e erreicht werden.

```
# tcpdump -X -r tcpdump.log
07:48:19.621692 kermit.32803 > listings.ebay.de.http: S
```

```

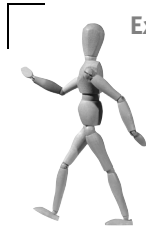
↳ 2047205616:2047205616(0) win 5840 <mss 1460,sackOK,timestamp 481661
↳ 0,nop,wscale 0> (DF)
0x0000 4500 003c b98d 4000 4006 757e c0a8 00ca      E.<..@.u~....
0x0010 d820 721d 8023 0050 7a05 e0f0 0000 0000      ..r..#.Pz.....
0x0020 a002 16d0 f098 0000 0204 05b4 0402 080a      .....
0x0030 0007 597d 0000 0000 0103 0300      ..Y}.....

```

**Achtung:**

Die Option `tcpdump -X` wird nur von der weiterentwickelten Variante unterstützt. Das originale *tcpdump* kennt diese Option nicht. Hier kann nur die Option `tcpdump -x` verwendet werden. Sie erzeugt keine ASCII-Ausgabe!

Wenn diese Vorgehensweise im Falle des Webservers angewendet wird, besteht die Möglichkeit, zu einem späteren Zeitpunkt zu prüfen, ob über Port 80 und 443 hinausgehende Verbindungen existierten. Um jedoch Festplattenplatz zu sparen, ist es unter Umständen sinnvoller, nur derartige Verbindungen zu protokollieren. Hier kommen nun die BPF-Filter zum Einsatz.

**Exkurs: BPF-Filter**

BPF-Filter definieren, welche Pakete von *tcpdump* und auch anderen *libpcap*-basierten Programmen gesammelt werden. Wenn kein Filter angegeben wird, werden alle Pakete gesammelt und von *tcpdump* ausgegeben. Der Filterausdruck kann aus einem oder mehreren Teilen (Primitiven) bestehen. Jeder dieser Teilausdrücke enthält mindestens eine Identifikation (Nummer oder Name), der meist eine der drei folgenden Eigenschaften (Qualifier) vorangestellt wird.

- **Typ.** Hierbei handelt es sich um eine von drei Möglichkeiten: `host`, `net` oder `port`. Der Typ gibt an, auf welche Eigenschaft des Paketes sich eine folgende Zahl bezieht. So trifft zum Beispiel `host 127.0.0.1` auf alle Pakete zu, die als Absender oder Zieladresse 127.0.0.1 tragen.
- **Richtung.** Hierbei kann die Richtung des Paketes bezogen auf den Typ angegeben werden. Mögliche Werte sind `src` (source, Quelle), `dst` (destination, Ziel), `src or dst` und `src and dst`. `dst port 22` trifft auf alle Pakete zu, die an einen SSH-Server gerichtet sind. Wird keine Richtung angegeben, so werden beide Richtungen angenommen (`src or dst`).

- **Protokoll.** Hier kann das Protokoll der zu sammelnden Pakete spezifiziert werden. Erlaubt sind `ether`, `fddi`, `tr`, `ip`, `ip6`, `arp`, `rarp`, `decnet`, `tcp` oder `udp`. Der SSH-Filter des letzten Absatzes lässt sich so weiter einschränken: `tcp dst port 22`

Zusätzlich besteht die Möglichkeit, die Länge des zu sammelnden Paketes einzuschränken. Mit `less <Länge>` werden sämtliche Pakete kleiner oder gleich der angegebenen Länge gesammelt, mit `greater <Länge>` entsprechend die Pakete, die länger oder gleich lang sind.

Häufig ist es einfacher, einen BPF-Filter zu definieren für die Pakete, die ignoriert werden sollen. In diesen Fällen ist eine Negation des Filters erforderlich. Das ist mit der Angabe des `!` oder `not` möglich. Das Ausrufezeichen wird in weiten Bereichen des UNIX-Betriebssystems zur Negation von Angaben benutzt.

Eine Kombination von Teilausdrücken ist mit logischen Operatoren möglich. Eine UND-Verknüpfung erfolgt mit `&&` oder `and`. Eine ODER-Verknüpfung erfolgt mit `||` oder `or`. Klammerausdrücke mit `()` sind möglich. Bei der Verwendung von Sonderzeichen ist jedoch zu beachten, dass die meisten Shells diese auswerten. Daher ist ein Escape der Sonderzeichen unerlässlich, z.B.: `\(` und `\)`.

```
host ftpserver and \( port 21 or port 20 \)
```

Die BPF-Filter erlauben bei fast allen Paketsniffern, die auf der *libpcap*-Bibliothek basieren (*tcpdump*, *ngrep*, *snort*, *ethereal*), die Menge der zu sammelnden Pakete bereits vor der Verarbeitung zu reduzieren und damit die Verarbeitung auch zu beschleunigen!

Um nun lediglich Pakete zu sammeln, die einen unnatürlichen Charakter aufweisen, müssen die normalen Pakete mithilfe eines BPF-Filters ignoriert werden. Als normale Pakete sollen Pakete definiert werden, die in Webzugriffen ausgetauscht werden. Hierbei handelt es sich um Verbindungen auf den Ports `http` (80) und `https` (443).

Im folgenden Beispiel wird davon ausgegangen, dass der Webserver den DNS-Namen *web.beispiel.de* aufweist. Der folgende BPF-Filter trifft auf die erwarteten Pakete zu.

```
host web.beispiel.de
```

Dieser Filter würde aber auch auf Telnet-Verbindungen, bei denen der Webserver beteiligt ist, zutreffen. Eine weitere Einschränkung ist erforderlich.

```
host web.beispiel.de and \( port 80 or port 443 \)
```

Dieser Filter trifft jetzt nur noch auf HTTP- und HTTPS-Verbindungen zu, bei denen der Webserver beteiligt ist. Leider trifft er auch auf Verbindungen zu, bei denen der Webserver selbst die Verbindung als Client aufbaut. Dies ist aber nicht erwünscht

und soll in diesem Beispiel auch nicht erlaubt sein. Daher ist es erforderlich, die Richtung zu berücksichtigen.

```
dst host web.beispiel.de and \( dst port 80 or dst port 443 \)
```

Dieser Filter wird aber lediglich die Pakete identifizieren, die an den Webserver gerichtet sind. Dieser wird aber auch antworten. Diese Pakete gehören auch zum erwarteten Verkehr.

```
\( dst host web.beispiel.de and \( dst port 80 or dst port 443 \) \) or \  
\( src host web.beispiel.de and \( src port 80 or src port 443 \) \)
```

Nun fehlt noch die Angabe des Protokolls: `tcp`. Der Filter ist bereits recht aufwändig und umständlich geworden. Um nun mit `tcpdump` den gesamten ungewöhnlichen Verkehr zu protokollieren, muss `tcpdump` die Negation dieses Filters übergeben werden. Es wird dann nur Pakete sammeln und protokollieren, die dem erwarteten Verkehr nicht entsprechen. Sinnvollerweise werden diese Pakete abgespeichert und regelmäßig untersucht oder auf dem Bildschirm ausgegeben, wenn eine »Echtzeitantwort« gewünscht wird.

#### Listing 7.8 `tcpdump`-Filter: Fertiger Webserver-Filter

```
tcpdump -w webserver.log tcp and not \( \  
\( dst host web.beispiel.de and \( dst port 80 or dst port 443 \) \) or \  
\( src host web.beispiel.de and \( src port 80 or src port 443 \) \) \)
```

Dieser Filter wird alle zutreffenden Pakete im `libpcap`-Format in der Datei `webserver.log` ablegen.

## 7.4 `ngrep`

`ngrep` verwendet wie `tcpdump` und Snort auch die `libpcap`-Bibliothek, um Netzwerkpakete zu sammeln. Hierbei bietet es aber zusätzlich die Möglichkeit, reguläre Ausdrücke ähnlich GNU `grep` zu verarbeiten, die auf den Inhalt des Paketes angewendet werden. So können Pakete mit bestimmtem Inhalt spezifisch gefiltert werden.

`ngrep` wird entwickelt von Jordan Ritter und ist leider noch nicht Bestandteil aller Linux-Distributionen. Es kann von seiner Homepage <http://ngrep.sourceforge.net> bezogen werden. Dort steht es sowohl als Quelltext als auch als RPM-Paket zur Verfügung. Jordan Ritter verwendet eine eigene Lizenz, die jedoch ähnlich der GPL eine Weiterverteilung und Modifikation der Software erlaubt. Jegliche Gewährleistung wird ausgeschlossen.

`ngrep` unterscheidet sich von `tcpdump` in der Tatsache, dass `ngrep` in der Lage ist, reguläre Ausdrücke auf den Inhalt des Paketes anzuwenden. Es bietet viele Funktionen, die auch von `tcpdump` und `grep` unterstützt werden. Die wichtigsten Optionen sollen kurz erklärt werden. Weitere Optionen werden in der Manpage erklärt.

- -i. Ignoriere Groß/Kleinschreibung
- -p. Kein promiscuous-Modus
- -v. Invertiere den regulären Ausdruck
- -s. Aufzuzeichnende Länge des Paketes (Snaps)
- -I *Datei*. Lese die Pakete aus der Datei
- -O *Datei*. Schreibe die Pakete, auf die der reguläre Ausdruck zutrifft, in die Datei.
- -A *Nummer*. Lese weitere <Nummer>-Pakete, wenn der reguläre Ausdruck zutrifft.

Diese Fähigkeiten erlauben den Einsatz von *ngrep* als Intrusion-Detection-System. Eine ganz einfache Variante wird im Folgenden dargestellt.

Firma XYZ besitzt eine große Anzahl von Mitarbeitern. Die Firma befürchtet, dass ein bestimmter Mitarbeiter versuchen wird, auf Daten der Firma zur Weitergabe an die Konkurrenz zuzugreifen. Jede Anmeldung mit seiner Kennung an einem der Systeme soll gemeldet und überwacht werden. Die Firma verwendet FTP-Server zur Speicherung der Daten und POP-Server zur Speicherung der E-Mails. Der Mitarbeiter besitzt die Kennung *jen0432*.

Dieses Problem kann recht einfach mit dem Werkzeug *ngrep* gelöst werden. Hierzu wird *ngrep* angewiesen, die entsprechenden Protokolle bezüglich des Benutzernamens zu untersuchen:

```
ngrep -A 5 'jen0432' tcp port 21 or tcp port 110
```

Dieser *ngrep*-Aufruf wird nun sämtliche Anmeldeversuche des Benutzers erkennen und anschließend weitere fünf Pakete protokollieren.

Leider ist die Anwendung regulärer Ausdrücke recht zeitaufwändig. Daher sollten bei Verwendung von *ngrep* wenn möglich BPF-Filter eingesetzt werden. Wenn mehrere reguläre Ausdrücke gesucht werden sollen, so bietet sich zunächst die Sicherung der Pakete mit *tcpdump* in einer Datei an, die anschließend (auch parallel) in mehreren *ngrep*-Läufen untersucht wird. Sobald jedoch die Anforderungen steigen, sollte der Einsatz eines integrierten NIDS wie *snort* in Erwägung gezogen werden.

Diese Protokollierung kann jedoch nur erfolgreich sein, wenn die Übertragung der Daten in Klartext erfolgt. Eine Verschlüsselung durch die Secure Shell oder mit der Secure Socket Layer (SSL) umgeht die Erkennung und Protokollierung der Daten.

