Spass im Netzwerk mit tcpdump & Co.

Stefan Schumacher
stefan@net-tex.de, PGP:0xb3fbae33

13. Juli 2005

http://www.net-tex.de

Zusammenfassung

Dieser Artikel soll als tcpdump Anleitung bzw. Einführung dienen. tcpdump ist ein effizientes Werkzeug um Netzwerkverkehr mitzuschneiden und zu analysieren. Weiterhin werden Werkzeuge vorgestellt und analysiert, die als Ergänzung zu tcpdump entwickelt wurden.

$Id: tcpdump.tex,v 1.7 2005/07/13 22:26:30 stefan Exp $

Inhaltsverzeichnis

1. Was ist tcpdump und was kann es für mich tun?

tcpdump ist ein Programm um den gesamten Netzwerkverkehr eines Interfaces mitzuschneiden und zu analysieren. Dies kann in vielen Fällen sehr nüzlich sein, z. B. um Fehler in Netzverbindungen zu finden, TCP/IP besser zu verstehen oder um die Netzperformance zu benchmarken.
tcpdump versetzt dazu die entsprechende Netzwerkschnittstelle in den sog. Promiscous Mode, d. h. die Netzwerkkarte liest nun nicht mehr nur für sie bestimmte Pakete, sondern schneidet alle (gewünschten) mit und gibt sie auf dem Bildschirm aus.
Um auf die Netzwerkschnittstelle zuzugreifen benötigt man root-Rechte, da so alle übertragenen Daten (auch Passwörter im Klartext!) abgefangen und mißbraucht werden können. tcpdump hat desweiteren noch ein paar Freunde, die sehr nützlich sind um erzeugte Mitschnitte tiefer zu analysieren oder zu plotten.

2. tcpdump

2.1. Ausgabe

tcpdump ermöglicht es den kompletten Netzwerkverkehr, der an einem Interface vorbeifließt mitzuschneiden. Der Mitschnitt lässt sich nach verschiedenen Gesichtspunkten beeinflussen und analysieren. Startet man tcpdump ohne Argumente bindet es sich an das erste Interface in der Systemliste das up ist. Ein gewünschtes Interface lässt sich mittels -i spezifizieren. Startet man nun tcpdump -i tlp1 so wird der gesamte Verkehr über tlp1 mitgeschnitten und am Bildschirm ausgegeben:

```
tcpdump: listening on tlp0
22:29:37.647072 192.168.2.2.64856 > frnk.dnscache.mediaways.net.domain:
    12045+ A? ftp.netbsd.org. (32)
22:29:37.856241 frnk.dnscache.mediaways.net.domain > 192.168.2.2.64856:
    12045 1/0/0 A ftp.netbsd.org (48)
22:29:37.857544 192.168.2.2.63115 > ftp.netbsd.org.ftp: S 1798388065:1798388065(0)
    win 65535 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF)
22:29:37.857575 192.168.2.2.63123 > dhcp-153-96-93-2.ifak.fhg.de.www: F 1510028614:
    1510028614(0) ack 3324910551 win 65535 <nop,nop,timestamp 2055 1126038302> (DF)
22:29:37.993659 dhcp-153-96-93-2.ifak.fhg.de.www > 192.168.2.2.63123:
R 1:1(0) ack 1 win 65535 <nop,nop,timestamp 2055 1126038302> (DF)
^C
31 packets received by filter
```

tcpdump zeigt hierbei die timestamp, also den Zeiteintrag des Rechners,

danach die IP-Adressen bzw. die FQDN [1] inklusive Portnummer an.

Feld	**Datum**
Zeitstempel	22:29:37.647072
Quelle > Ziel	192.168.2.2.64856 > frnk.dnscache.mediaways.net.domain
Adresseintrag	A? (qtype=A)
nachgefragte Adresse	ftp.netbsd.org

`tcpdump` bietet noch weitere Optionen um die Ausgabe zu beeinflussen, z. B. `-t` oder `-v` um die timestamp bzw. die verbosity zu beeinflussen.
Als weitere Option beeinflußt `-e` die Darstellung des Link Level Headers. In Ethernet Netzen wird die Zieladresse, das Protokoll und die Paketlänge ausgegeben, in FDDI wird zusätzlich das `frame control`-Feld, Token Ring erweitert die Ausgabe noch um das `access control`-Feld. Für SLIP Verbindungen wird die Verbindungsrichtung angegeben (I für einkommend, O für ausgehend), der Pakettyp (`ip, utcp, ctcp`) angezeigt und Kompressionsinformationen ausgegeben.

2.2. Datagrammtypen

2.2.1. ARP/RARP

Angezeigt werden hierbei der Requesttyp und die Argumente, die Ausgabe ist sehr leicht verständlich:

```
arp who-has Boromir tell Arwen
arp reply Arwen is-at 0:3:d3:1a:db:21
```

2.2.2. TCP

Das allgemeine Format für ein TCP Datum ist:
`Quelle > Ziel: Flags Sequenznr. ACK Window Urgent Optionen`
`Flags` sind Kombinationen aus S (Syn), F (Fin), P (Push), R (Reset) oder . für keine Flags.
Quelle, Ziel und Flags werden immer angezeigt, die folgenden Optionen sind vom Paket abhängig.
Die Sequenznr. ist die Anzahl der Sequenzen die von den Daten im Paket benutzt werden. ACK ist die erwartete folgende Sequenznummer in Gegenrichtung, Window zeigt die Anzahl der Bytes an, die im Pufferspeicher in Gegenrichtung verfügbar sind. Urgent zeigt an das ´dringende´ Daten im Paket enthalten sind. Optionen sind enthaltene TCP-Optionen.

2.2.3. AFS

Die Pakete des Andrew File Systems werden folgendermassen ausgegeben:

[1] Full Qualified Domain Name

```
src.sport > dst.dport: rx packet-type
src.sport > dst.dport: rx packet-type service call call-name args
src.sport > dst.dport: rx packet-type service reply call-name args
```

Die Ausgabe kann erweitert werden, `-v` gibt zusätzlich ACK-Pakete und Headerinformationen wie RX ID, Rufnummer, Sequenznummer, Seriennummer und RC Paketflags aus.
`-vv` erweitert noch zusätzlich die Ausgabe um die MTU Informationen, ein weiteres `v` gibt noch die Service ID und Security Index aus.
Da AFS-Calls sehr groß sind, muss die snaplen mit `-s 256` herauf gesetzt werden.

2.2.4. ICMP

Nach ICMP Paketen filtern:

```
root@AragornOfArathorn {7} tcpdump -i ex0 'icmp'
tcpdump: listening on ex0
00:01:57.047524 Aragorn > 192.168.2.1: icmp: echo request seq 0
00:01:57.047654 192.168.2.1 > Aragorn: icmp: echo reply seq 0
```

es wird wie üblich Timestamp und Quelle/Ziel angezeigt, dazu das IP Protokoll (icmp) und der ICMP Typ (8/0). Mehr zu ICMP Pakettypen in [8]

2.3. Filter

Neben den vielen Möglichkeiten die Ausgabe zu beeinflussen hat `tcpdump` die weitaus interessantere Fähigkeit verschiedene Pakete zu filtern. So ist es möglich mit der Option `-T` einen Pakettyp (bspw. rtp, rpc oder snmp) zu spezifizieren.
Zusätzlich dazu kann mittels `-F` eine Datei angegeben werden, aus der `tcpdump` Filterregeln ausliest, diese Regeln können auch einfach mit den entsprechenden Schlüsselwörtern als Kommandoparameter übergeben werden. Das $-Zeichen soll hier als Platzhalter für Netz- / IP-Adressen oder Hostnamen dienen, letztere müssen allerdings von der betreffenden Maschine aufgelöst werden können.

Schlüsselwort	Funktion
`type`	mit `host`, `net` und `port` können die entsprechenden Parameter angegeben werden
`dir`	gibt die Übertragungsrichtung an
`proto`	übergibt das Protokoll: `ether, fddi, tr, ip, ip6, arp, rarp, decnet, tcp, udp`
`and`, `or`, `not`	erlaubt es Filterregeln zu verknüpfen
`dst host $`	filtert das `destination field` der Pakete

`src host $`	filtert das `source field` der Pakete
`host $`	filtert das `source` und `destination field` der Pakete
`ether dst $`	filtert die Ethernet destination address
`ether src $`	filtert die Ethernet source address
`ether host $`	filtert die Ethernet source oder destination address
`gateway $`	prüft ob das Paket das spezifizierte Gateway passiert hat, d. h. die Adresse darf weder Source noch Destination sein
`dst net $`	filtert nach dem Zielnetz (siehe networks(4))
`src net &`	filtert nach dem Quellnetz (siehe networks(4))
`net &`	filtert nach Quell- oder Zielnetz
`net $ mask $`	filtert nach Netz und Netzmaske (Nur in IPv4)
`net $/$`	filtert nach Netz und passender Subnetz Bitlänge in CIDR Notation
dst port $	filtert in den Protokollen ip/tcp, ip/udp, ip6/tcp oder ip6/udp nach dem Zielport
src port $	filtert nach dem Quellport
port $	filtert nach Quell- oder Zielport
`less $`	filtert Pakete die kleiner als $ sind
`greater $`	filtert Pakete die größer als $ sind
`ip proto $`	filtert nach IP-Protokollen: `icmp, icmp6, igmp, igrp, pim, ah, esp, vrrp, udp, tcp`, da `icmp, udp, tcp` Schlüsselwörter sind, müssen sie mit \ escaped werden
`ip6 proto $`	filtert nach dem entsprechenden IPv6 Protokoll
`ip6 protochain $`	filtert Pakete die IPv6 sind und einen passenden `protocol` Header haben
`ip protochain &`	filtert Pakete die IPv4 sind und einen passenden `protocol` Header haben
`ether broadcast`	filtert Ethernet Boradcast Pakete
`ip broadcast`	filtert IP Broadcast Pakete mit Nullen oder Einsen Header und prüft die lokale Subnetzadresse
`ether multicast`	filtert Ethernet Multicast Pakete
`ip multicast`	filtert IPv4 Multicast Pakete
`ip6 multicast`	filtert IPv6 Multicast Pakete
`ether proto $`	filtert Ethernet Pakete in folgenden Protokollen: `ip, ip6, arp, rarp, atalk, aarp, dec- net, sca, lat, mopdl, moprc, iso, stp, ipx, netbeui`, diese müssen wieder per \ escaped werden
`decnet src $`	filtert in DECNET nach dem angegebenen Quellhost
`decnet dst $`	filtert in DECNET nach dem angegebenen Zielhost
`decnet host $`	filtert in DECNET nach dem angegebenen Host

`ip, ip6, arp, rarp, atalk, aarp, decnet, iso, stp, ipx, netbeui`	sind Kurzformen für `ether proto $`
`vlan`	filtert IEEE 802.1Q VLAN Pakete
`vlan $`	filtert VLAN Pakete mit entsprechender ID
`tcp, udp, icmp`	Kurzform von `ip proto $ or ip6 proto $`
`iso proto $`	filter OSI Pakete des Typs `clnp, esis, isis`
`clnp, esis, isis`	Kurzform für `iso proto $`

`tcpdump` ist weiterhin noch in der Lage den Inhalt der Pakete mittels logischer Ausdrücke zu filtern, dies geschieht mit einer einfachen Anweisung der Form:
`Ausdruck Relation Ausdruck` mit `Relation` aus {>, <, >=, <=, =, !=} und `Ausdruck` ist ein arithmetischer Ausdruck bestehend aus Integerzahlen, Binäroperatoren aus {+, −, ∗, /, &, |} und spezielle Ausdrücke um Paketdaten zu filtern.
Daten innerhalb der Pakete werden mit `$Protokoll [ $Ausdruck : $Grösse ]` gefiltert, dabei ist $Protokoll aus {ether, fddi, tr, ppp, slip, link, ip, arp, rarp, tcp, udp, icmp, ip6,} und spezifiziert das gewünschte Protokoll. $Ausdruck übergibt das Byte-Offset relativ zur Protokollschicht. $Grösse übergibt optional die Anzahl (1,2 oder 4) der Bytes ab dem übergebenen Byte-Offset, zusätzlich kann noch mit `len $` die Paketlänge angegeben werden. `tcpdump` ist auch in der Lage einige Feldnamen bzw. Offsets, TCP Flags und ICMP Typen über Namen anzusprechen:

`Feldnamen`

- `icmptype`
- `icmpcode`
- `tcpflags`

`TCP Flags`

- `tcp-fin`
- `tcp-syn`
- `tcp-rst`
- `tcp-push`
- `tcp-push`
- `tcp-ack`
- `tcp-urg`

`ICMP Typen`

- `icmp-echoreply`
- `icmp-unreach`
- `icmp-sourcequench`
- `icmp-redirect`
- `icmp-echo`
- `icmp-routeradvert`
- `icmp-routersolicit`
- `icmp-timxceed`
- `icmp-paramprob`
- `icmp-tstamp`
- `icmp-tstampreply`
- `icmp-ireq,`
- `icmp-ireqreply`
- `icmp-maskreq`
- `icmp-maskreply`

Geklammerte Ausdrücke können wieder mit logischen Operatoren (`!= =`

NOT, && = AND und || = OR) verknüpft werden.

Beispiele:
`tcpdump ip and not net localnet`
zeigt allen Traffic auf einem Gateway an der nicht in das eigene Subnetz geht
`tcpdump 'gateway Arwen and (port ftp or ftp-data)'`
zeigt FTP Traffic der über 'Arwen' gekommen ist
`tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'`
zeigt ICMP Pakete an die nicht Ping und Pong sind
`tcpdump ip host Boromir and not Feanor`
zeigt allen Traffic der 'Boromir' aber nicht 'Feanor' betrifft

3. tcptrace

`tcptrace` analysiert `tcpdump` Dateien und erzeugt dafür Zusammenfassungen bzw. xplot Graphen.

3.1. Ausgabe

`-b` bzw. `-l`	kurze bzw. lange Zusammenfassung
`-r`	Round Trip Time (rtt) Statistik
`-D`	dezimale Ausgabe
`-X`	hexadezimale Ausgabe
`-n`	Namen nicht auflösen
`-s`	Hostnamen statt FQDN
`-G`	erzeugt alle Graphen
`-T`	Durchsatzdiagramme
`-R`	Round Trip Time (RTT) Graphen
`-S`	Zeitsequenzgraphen
`-N`	Congestion Window Graphen
`-F`	Segmentgröße
`-C` bzw. `-M`	Farbe / Monochromgraphik

3.2. Filter

Für gewöhnlich snifft `tcpdump` mehrere Verbindungen mit, die natürlich auch von `tcptrace` erkannt werden. Diese Verbindungen werden meist mit a2b_... bzw. b2a_... usw. bezeichnet. `tcptrace` ist selbstverständlich auch wieder in der Lage dies Verbindungen zu filtern. Mittels `-i $` wird die Verbindung $ ausgeblendet, mit `-o $` wird nur die Verbindung $ betrachtet, `-c` blendet Verbindungen aus die nicht abgeschlossen wurden (keine SYN und FIN

Pakete vorhanden). Es lassen sich auch Segmentnummern übergeben die jeweils obere (`-E $`) bzw. untere (`-B $`)Grenze darstellen. `-u` gibt zusätzlich noch minimale UDP Informationen aus, `-e` extrahiert den Inhalt jedes TCP Streams in eine Datei, `-p` schreibt den Inhalt aller Pakete in eine Datei.

3.3. Plotter und andere Freunde

Geradezu unverzichtbar im Umgang mit `tcpdump` und `tcptrace` ist `xplot` [1] das die erzeugten Diagramme unter X11 graphisch darstellt. Desweiteren gibt es eine Reimplementierung von `xplot` für Java namens `jplot` [2].`tracelook` erzeugt xgraph Plots.
Da das xplot-Format etwas schwierig zu handhaben ist gibt es von der Uni Ohio ein AWK Skript [3] das xplot nach gnuplot konvertiert, und so z.B. für LaTeX aufbereitbar macht.

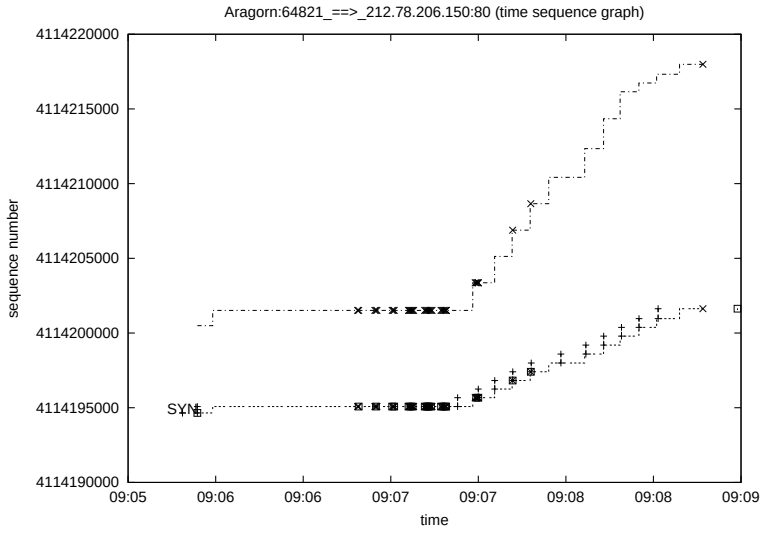

Abbildung 1: mit `gnuplot` geplotteter time-sequence Graph

Neben `tcptrace` gibt es von Sony Japan noch ein Tool namens tcpillust [4], das aus `tcptrace` Dateien Graphiken erzeugt, die denen aus W. R. Stevens 'TCP/IP Illustrated' Reihe gleichen.
`tcpdstat` erzeugt ebenfalls Statistiken aus `tcptrace` Dateien, unter anderem mit Zeit, Größe und Paketen je Protokoll. `flstats` erzeugt ebenfalls Flussstatistiken.

Da `tcptrace` in der Regel große Dateien erzeugt können diese mit `tcpslice` [5] in kleinere zerlegt werden. `tcpdmerge` arbeitet genau entgegengesetzt, es führt 2 dumps wieder zusammen.
`tcpslice` extrahiert Teile eines Mitschnitts,
Äußerst bedenklich ist allerdings die extrem hohe Verletzung der Privatsphäre durch mitgesniffte Daten, daher hat Ethan Blanton mit `tcpurify` [6] einen Sniffer entwickelt der auch Wert auf die Privatsphäre legt. Ebenso gibt es noch `tcpdpriv` [7], das dazu dient `tcptrace` Dateien zu anonymisieren.

Die meisten dieser Programme stehen unter `/usr/pkgsrc/net/` in der Package Collection zur Installation bereit.

4. Literatur

[1] xplot
`/usr/pkgsrc/graphics/xplot`
http://www.xplot.org/

[2] jPlot
nicht in pkgsrc
http://masaka.cs.ohiou.edu/software/tcptrace/jPlot/

[3] xpl2gpl
nicht in pkgsrc
http://tcptrace.org/xpl2gpl/

[4] tcpillust
`/usr/pkgsrc/net/tcpillust`
http://www.csl.sony.co.jp/person/nishida/tcpillust.html

[5] tcpslice `/usr/pkgsrc/net/tcpslice`
ftp://ftp.ee.lbl.gov/tcpslice.tar.Z

[6] tcpurify
nicht in pkgsrc
http://masaka.cs.ohiou.edu/ eblanton/tcpurify/

[7] tcpdpriv
`/usr/pkgsrc/net/tcpdpriv`

[8] Stefan Schumacher: “ICMP Packettypes and Codes”
http://www.net-tex.de/net/icmptypen.pdf

A. tcpdump-Befehlsreferenz

```
tcpdump [ -AdDeflLnNOpqRStuUvxX ] [ -c count ]
 [ -C file_size ] [ -F file ] [ -i interface ]
 [ -m module ] [ -r file ] [ -s snaplen ]
 [ -T type ] [ -w file ] [ -E spi@ipaddr algo:secret,...  ]
 [ -y datalinktype ] [ expression ]
```

Schalter	Bedeutung
-A	jedes Paket ohne Link Level Header (MAC) als ASCII ausgeben
-a	nslookup IP-Adressen
-c *n*	nach *n* Paketen abbrechen
-C *n*	schreibe in eine neue Datei wenn die Alte *n*-Mio Bytes übersteigt
-d	gibt gefilterte Pakete in lesbarer Form aus
-dd	gibt gefilterte Pakete in C aus
-ddd	gibt gefilterte Pakete als Dezimalzahl aus
-D	listet vorhandene und von tcpdump nutzbare NICs auf
-e	gibt Link Level Header aus
-E	ermöglicht es IPsec ESP-Pakete zu dechiffrieren (*Sicherheitsgefahr!*)
-f	IPs als Zahl ausgebe statt sie aufzulösen
-F	lies Filterausdrücke aus angegebener Datei
-i	lausche auf NIC
-l	stdout puffern (z.B. für tee)
-L	listet Verbindungstypen für die NIC
-n	Adressen nicht nach Namen konvertieren
-N	Domain im Hostnamen weglassen
-O	packet-matching code optimizer deaktivieren, nur für debugging
-p	NIC nicht in den Promiscous Mode versetzen
-R	definiere ESP/AH-Pakete als RFC182[5-9]-konform
-r	lies aus Datei (siehe -w)
-S	absolute TCP/IP Sequenznummern statt relative
-s	*n*-Bytes je Paket mitschneiden
-T	gefilterte Pakete als *Type* behandeln
-t	kein Zeitstempel auf jeder Zeile
-tt	unformatierter Zeitstempel auf jeder Zeile
-ttt	Zeitabstand zwischen jeder Zeile
-tttt	Zeitstempel + Datum auf jeder Zeile
-u	NFS-Handles nicht dekodieren
-U	schreibt jedes Paket sofort in die Datei anstatt erst den Puffer zu füllen
-v, -vv, -vvv	mehr Informationen ausgeben
-w	Pakete in Datei schreiben
-x	jedes Paket ohne LLH in hex ausgeben
-X	jedes Paket ohne LLH in hex und ASCII ausgeben
-y	setzt den Datalinktype

Schlüsselwörter:

IP-Protokolle

- icmp
- icmp6
- igmp
- igrp
- pim
- ah
- esp
- vrrp
- udp
- tcp

Protokolle

- ether
- fddi
- tr
- ip
- ip6
- arp
- rarp
- decnet
- tcp
- udp

Feldnamen

- icmptype
- icmpcode
- tcpflags

TCP-Flags

- tcp-fin
- tcp-syn
- tcp-rst
- tcp-push
- tcp-push
- tcp-ack
- tcp-urg

Ethernet-Protokolle

- ip
- ip6
- arp
- rarp
- atalk
- aarp
- dec-net
- sca
- lat
- mopdl
- moprc
- iso
- stp
- ipx
- netbeui

ICMP-Typen

- icmp-echoreply
- icmp-unreach
- icmp-sourcequench
- icmp-redirect
- icmp-echo
- icmp-routeradvert
- icmp-routersolicit
- icmp-timxceed
- icmp-paramprob
- icmp-tstamp
- icmp-tstampreply
- icmp-ireq,
- icmp-ireqreply
- icmp-maskreq
- icmp-maskreply

RFC Database: http://www.rfc-editor.org/
ICMP Pakettypen (PDF): http://www.net-tex.de/net/icmptypen.pdf