Einbrechen in TCP-Verbindungen

Berliner Linux User Group, 16. November 2005

Wilhelm Dolle, Director Information Technology
interActive Systems GmbH

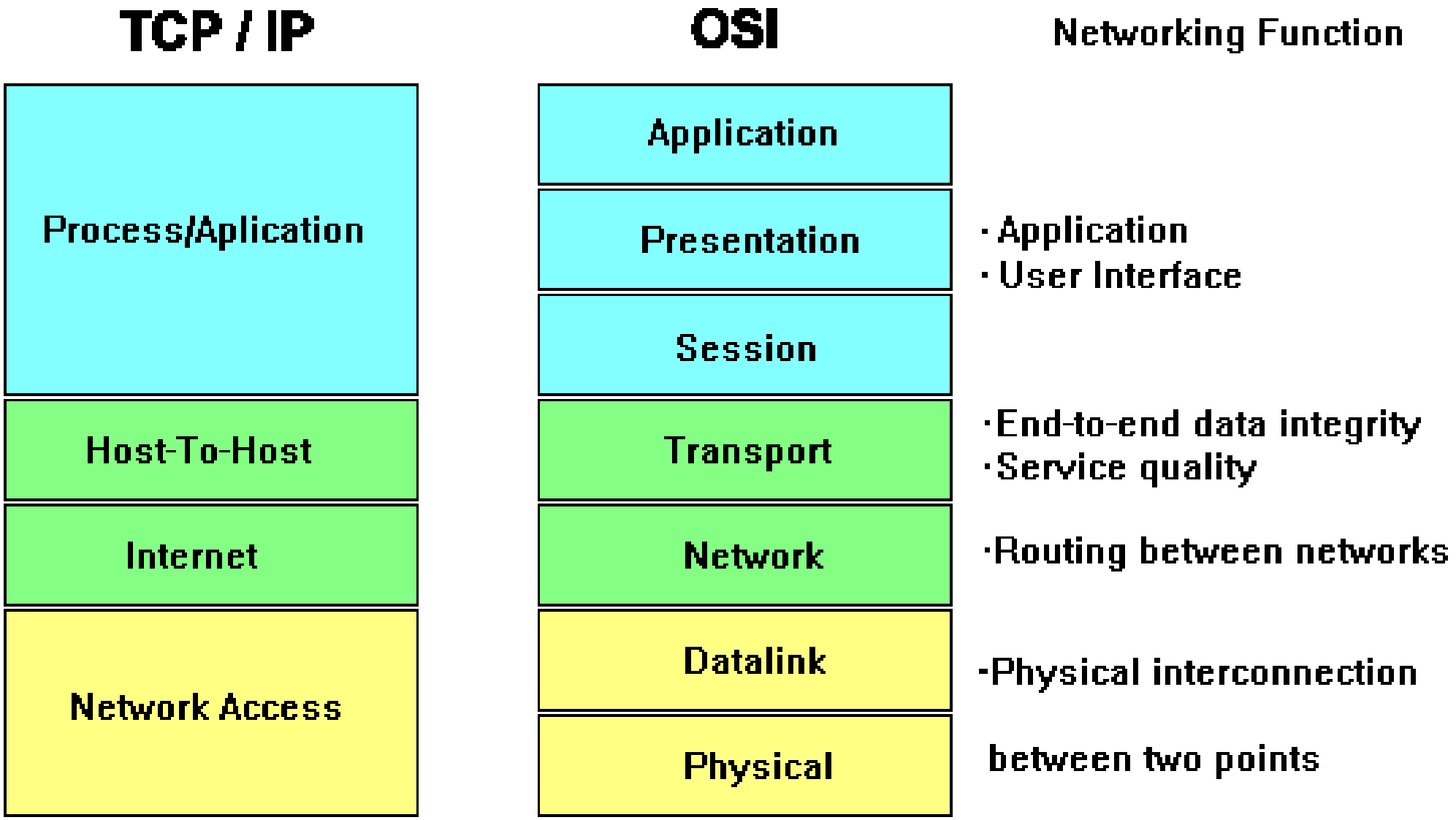
Was werde ich heute nicht erzählen?

- Einbrechen über TCP-Verbindungen in Server / Dienste
- Etwas zu IP, UDP oder ICMP (bzw. nur sehr wenig über diese Themen)

Agenda

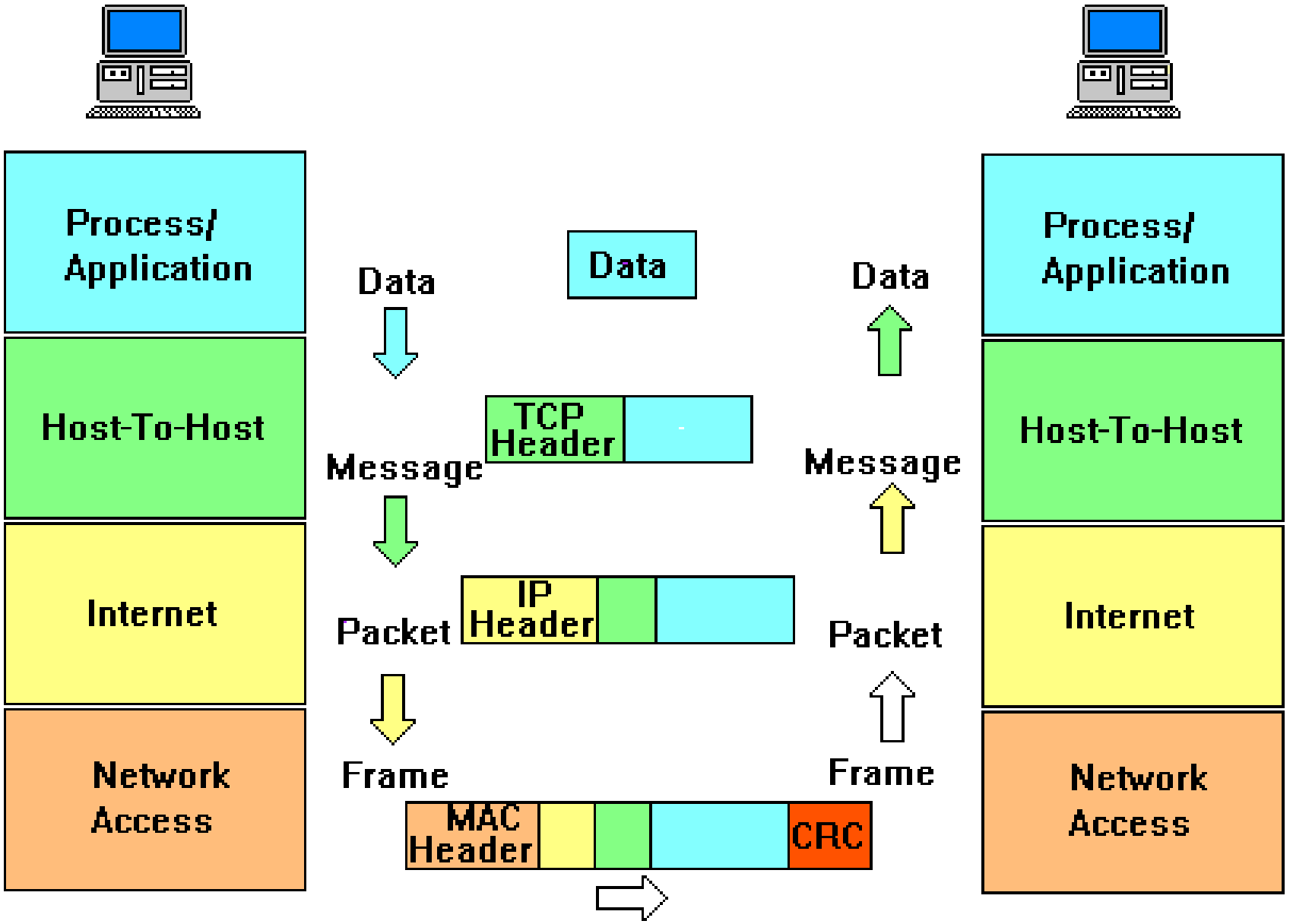
- Kurzer Sprint durch die Grundlagen von TCP
- Bekannte Schwachstellen / ältere Angriffe?
- TCP Reset Angriff (Theorie, Praxis, Schutz)

TCP/IP und ISO OSI Modell



(Quelle: www.netzmafia.de)

TCP/IP



(Quelle: www.netzmafia.de)

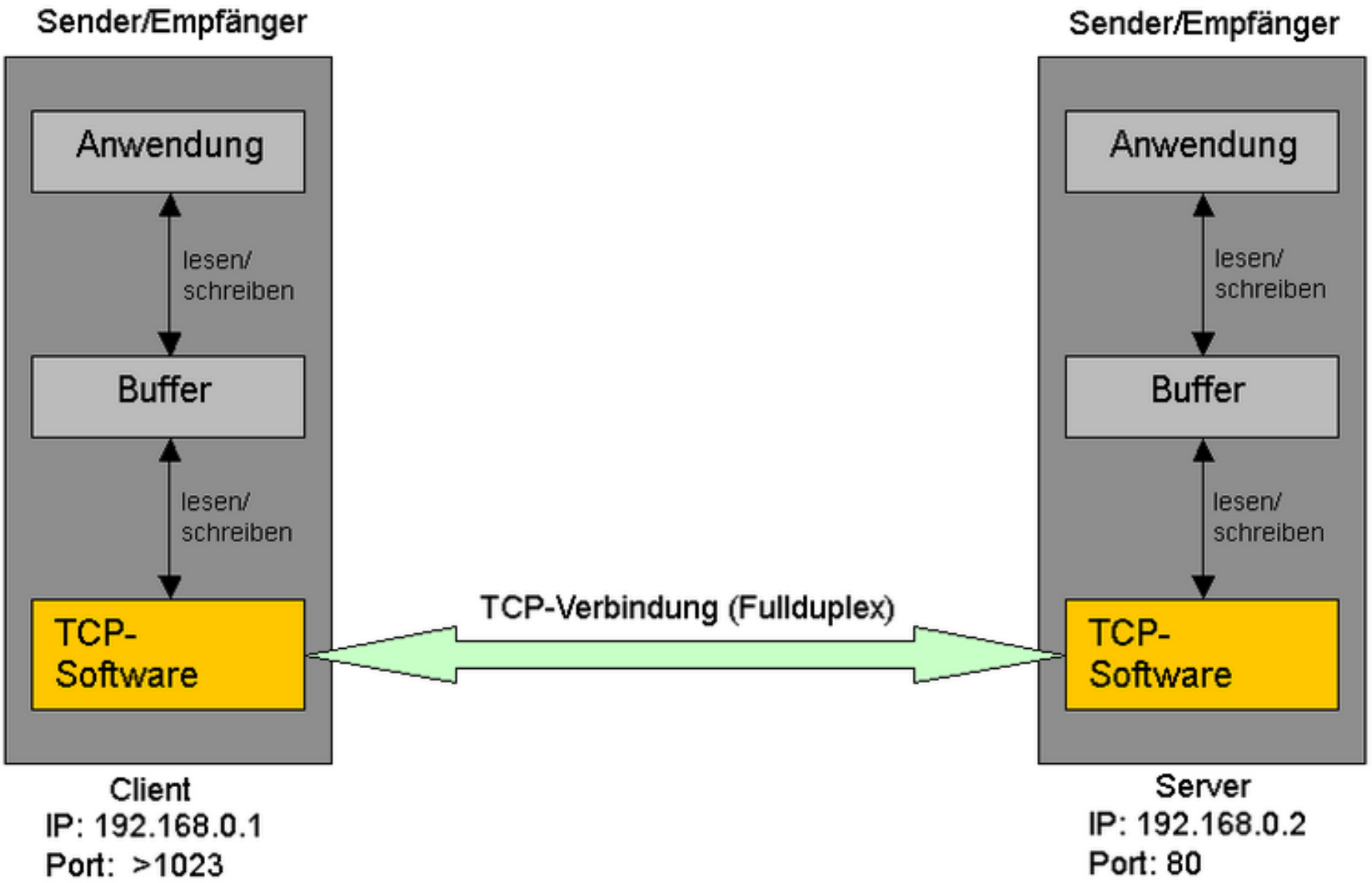
Transmission Control Protocol (TCP)

- Zuverlässiges, verbindungsorientiertes Transportprotokoll
- Entwickelt von Robert E. Kahn und Vinton G. Cerf ab 1973
- Standardisierung 1981 in RFC 793
- Ergänzungen in RFC 1122 und 1213
- Auf Schicht 4 des OSI-Referenzmodell
- Punkt-zu-Punktverbindung in Vollduplex

- Eindeutige Definition durch zwei Endpunkte (Tupel) aus IP-Adresse und Port

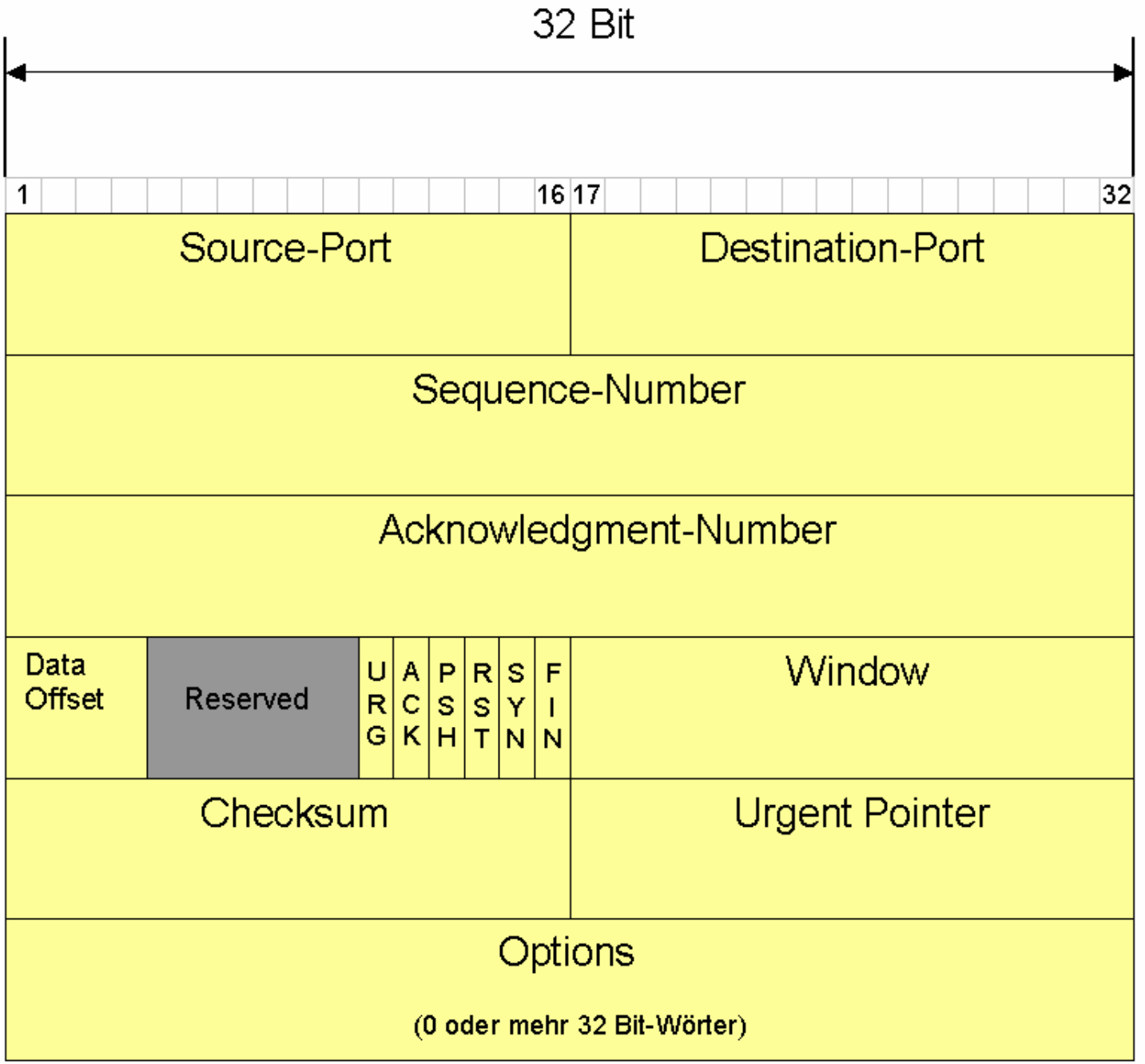
- Sicherung der Datenübertragung durch Prüfsumme und Quittierung mit Zeitüberwachung
- Ein Bytestrom wird in Pakete, ***Segmente*** genannt, aufgeteilt

TCP – Interaktion mit Anwendungen



(Quelle: www.wikipedia.de)

TCP – Header



(Quelle: www.wikipedia.de)

TCP – Header

- **Source Port** - Identifiziert den sendenden Prozess
- **Destination Port** - Identifiziert den Prozess des Zielsystems
- **Sequence Number** - Nummer des ersten Datenbytes (Oktets) im jeweiligen Segment (--> richtige Reihenfolge über verschiedene Verbindungen eintreffender Segmente wieder herstellbar)
- **Acknowledgement Number** - Daten vom Empfänger bestätigt, wobei gleichzeitig Daten in Gegenrichtung gesendet werden. Die Bestätigung wird also den Daten "aufgesattelt" (Piggyback). Die Nummer bezieht sich auf eine Sequence-Nummer der empfangenen Daten; alle Daten bis zu dieser Nummer (ausschließlich) sind damit bestätigt --> Nummer des nächsten erwarteten Bytes.
- **Data Offset** - Da der Segment-Header ähnlich dem IP-Header Optionen enthalten kann, wird hier die Länge des Headers in 32-Bit-Blöcken angegeben.
- **Reserved** - Reserviert für spätere Nutzung

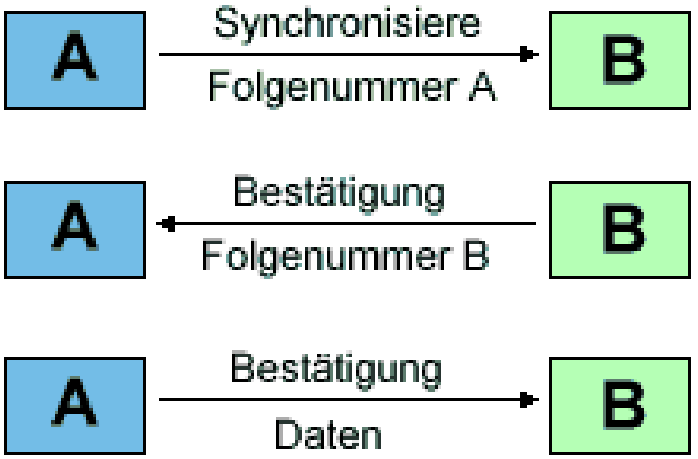
TCP – Header

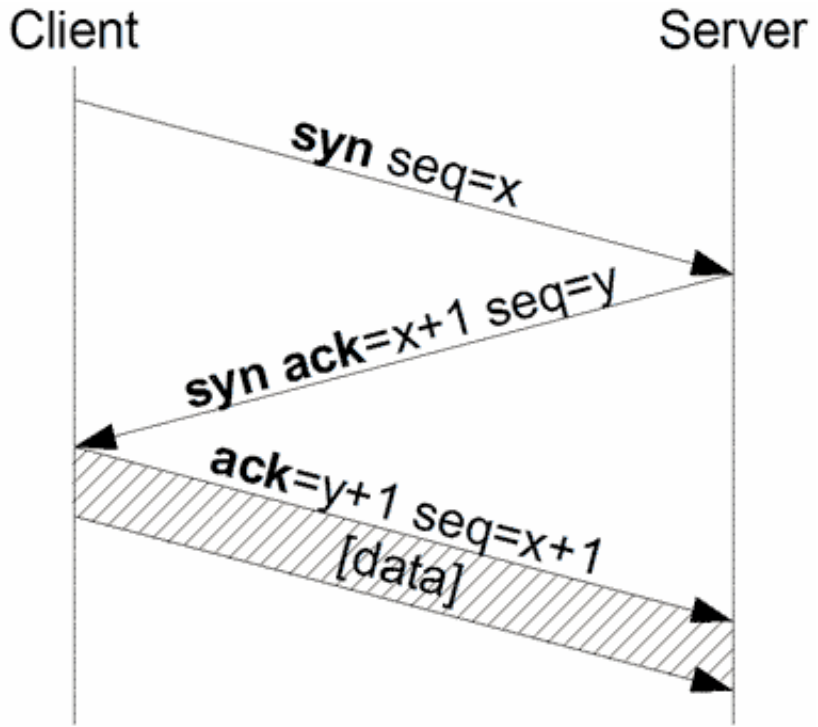
- Code – Angabe der Funktion des Segments:
  - **URG** – Urgent-Pointer (siehe unten); Daten bevorzugt behandeln
  - **ACK** – Quittungs-Segment (Acknowledgement-Nummer gültig)
  - **PSH** – Auf Senderseite sofortiges Senden der Daten aus dem Sendepuffer und auf Empfangsseite sofortige Weitergabe aller Daten aus dem Empfangspuffer an die Applikation z. B. für interaktive Programme
  - **RST** – Reset, Verbindung abbrechen / zurücksetzen
  - **SYN** – Das 'Sequence Number'-Feld enthält die initiale Byte-Nummer (ISN) --> Numerierung beginnt mit ISN + 1. In der Bestätigung übergibt der Empfänger seine ISN (Verbindungsaufbau).
  - **FIN** – Verbindung abbauen (Sender hat alle Daten gesendet), sobald der Empfänger alles korrekt empfangen hat und selbst keine Daten mehr senden möchte.

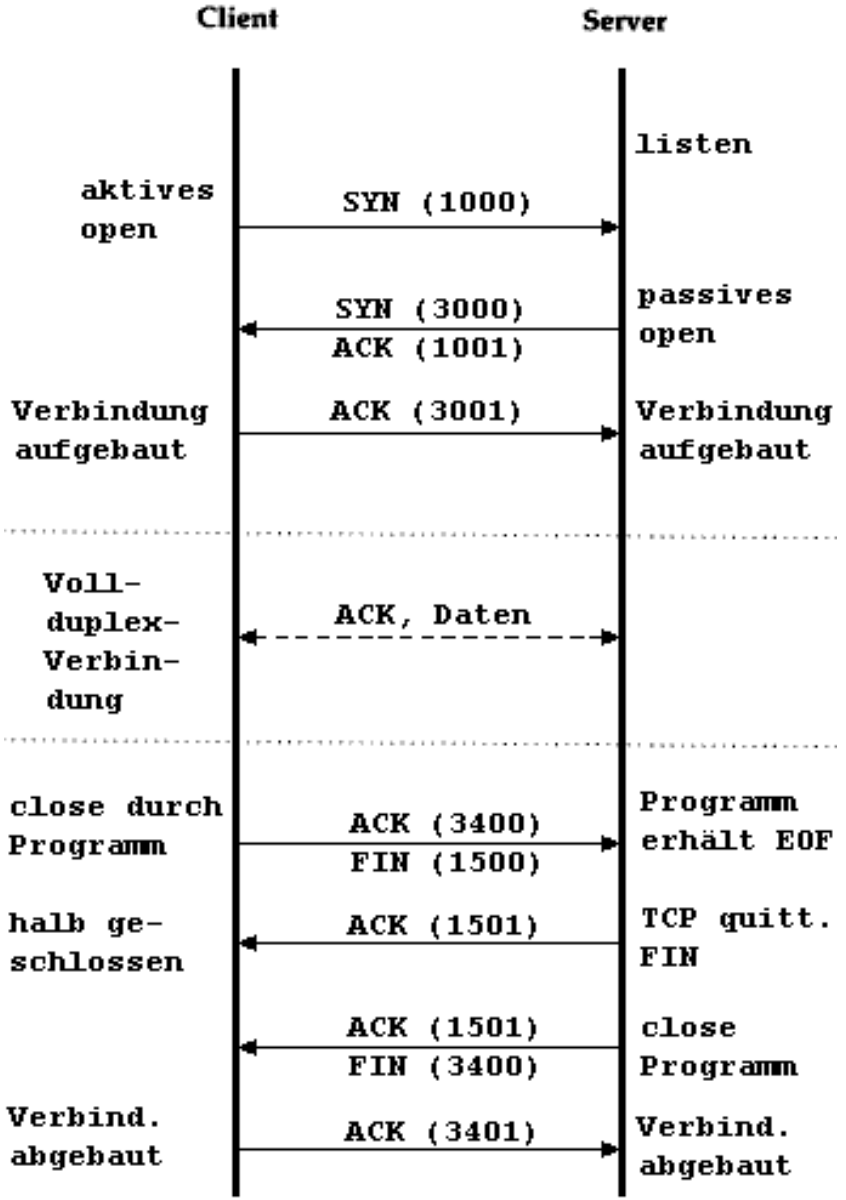
TCP – Header

- **Window** – Spezifiziert die Fenstergröße in Bytes, die der Empfänger bereit ist ohne weitere Bestätigung anzunehmen – kann dynamisch geändert werden.
- **Checksum** – 16-Bit Längsparität über Header und Daten
- **Urgent Pointer** – Markierung eines Teils des Datenteils als dringend. Dieser wird unabhängig von der Reihenfolge im Datenstrom sofort an das Anwenderprogramm weitergegeben (URG-Code muss gesetzt sein). Der Wert des Urgent-Pointers markiert das letzte abzuliefernde Byte; es hat die Nummer <Sequence Number> + <Urgent Pointer>.
- **Options** – Dieses Feld dient dem Informationsaustausch zwischen beiden Stationen auf der TCP-Ebene, z. B. die Segmentgröße (die Ihrerseits von der Größe des IP-Datagramms abhängen sollte, um den Durchsatz im Netz optimal zu gestalten).

Aufbau und Abbau einer TCP Session

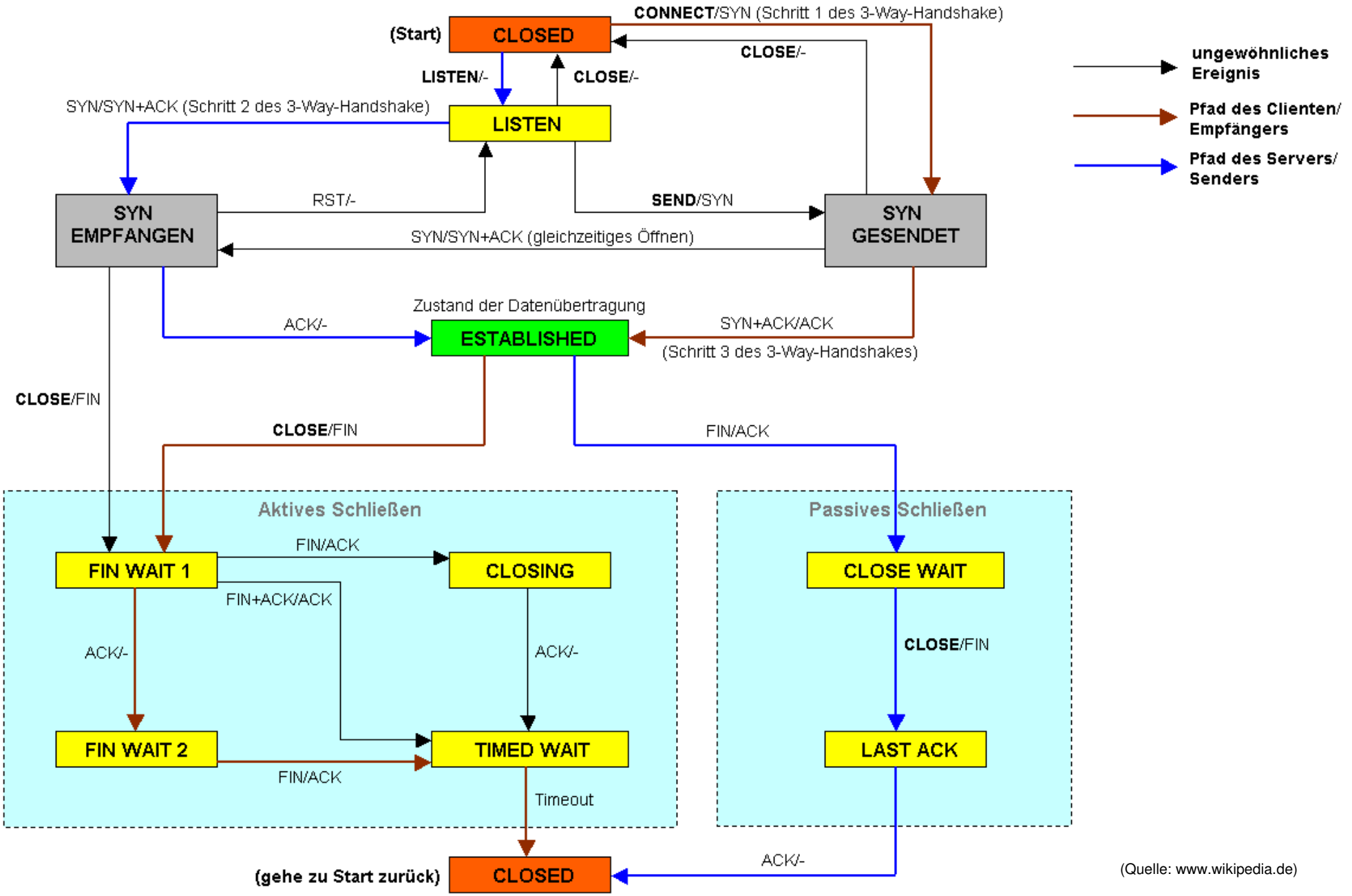






(Quelle: www.wikipedia.de / www.netzmafia.de)

TCP – Zustandsübergangsdiagramm

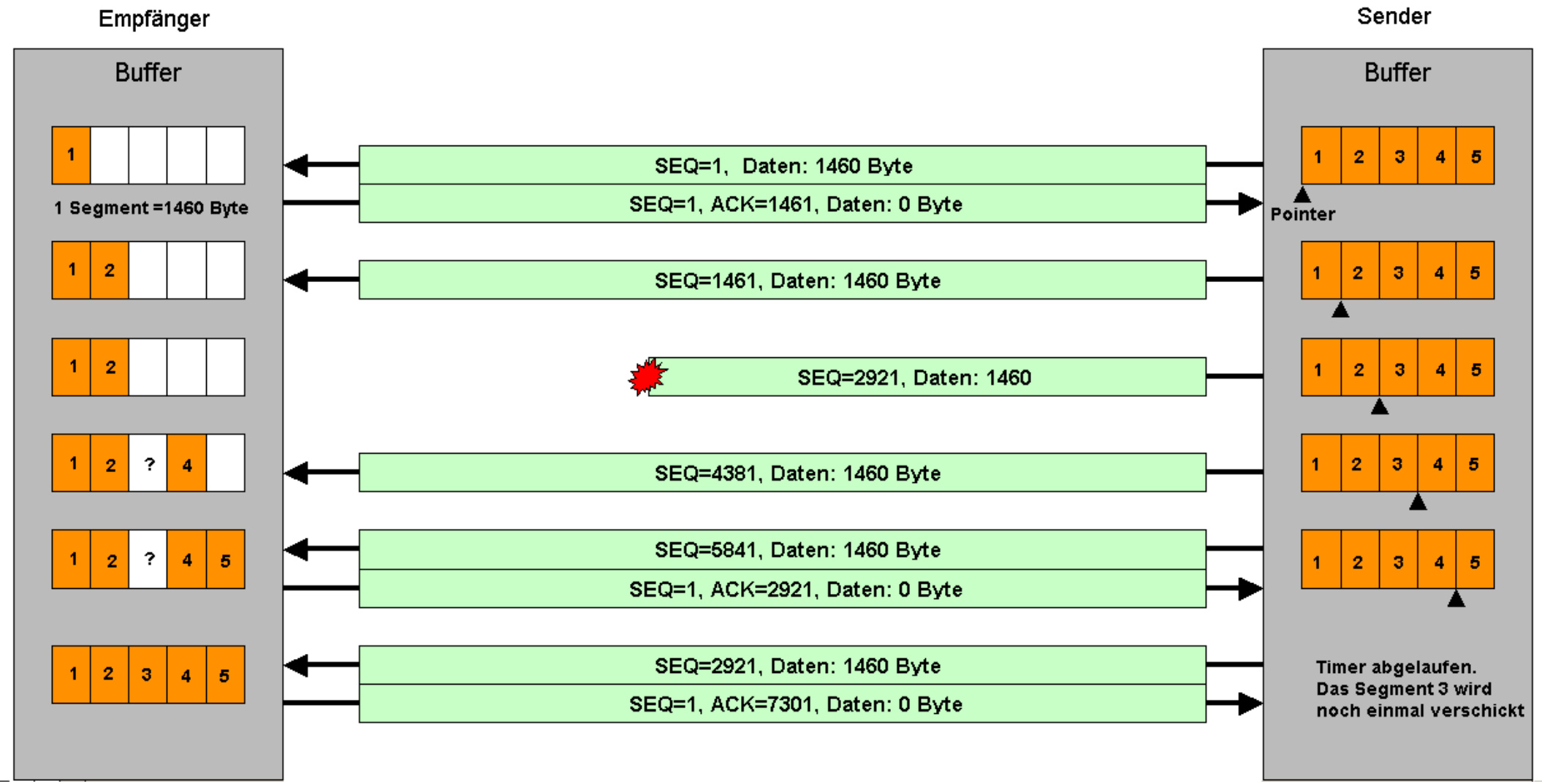


(Quelle: www.wikipedia.de)

TCP – Zustandsübergangsdiagramm

- LISTEN – Warten auf ein Connection Request.
- SYN-SENT – Warten auf passendes Connection Request, nachsenden von SYN
- SYN-RECEIVED – Warten auf Bestätigung des Connection Request Acknowledgement, nachdem beide Teilnehmer ein Connection Request empfangen und gesendet haben.
- ESTABLISHED – Offene Verbindung.
- FIN-WAIT-1 – Warten auf ein Connection Termination Request des Kommunikationspartners oder auf eine Bestätigung des Connection Termination, das vorher gesendet wurde.
- FIN-WAIT-2 – Warten auf ein Connection Termination Request des Kommunikationspartners.
- CLOSE-WAIT – Warten auf ein Connection Termination Request (CLOSE) der darüberliegenden Schicht.
- CLOSING – Warten auf ein Connection Termination Request des Kommunikationspartners. LAST-ACK: Warten auf die Bestätigung des Connection Termination Request, das zuvor an den Kommunikationspartner gesendet wurde.

TCP – Datenübertragung



(Quelle: www.wikipedia.de)

TCP – kleine Sicherheitshistorie

- 1981: Spezifikation von TCP in RFC 793
- 1985: Robert Morris weist auf Schwachstellen in TCP hin
- 1994: Ausnutzen einer TCP-Schwachstelle in Kevin Mitnicks „Christmas Day Attack“ auf das Netz von Tsutomu Shimomura
- 1995: Paul Watson veröffentlicht ein Usenet Posting zu Schwachstellen in TCP
- 1995: Laurent Joncheray präsentiert auf der Usenix-Konferenz sein Paper zu „Simple Active Attacks against TCP“
- 2001: Cert.org weist auf Schwachstellen in mehreren TCP/IP-Sequenznummern-Generatoren und Probleme mit der Window-Size hin
- 2003: Paul Watson zeigt Angriffe auf TCP mit einer einfach DSL-Anbindung
- 2004: Internet Engineering Task Force (IETF) veröffentlicht einen Draft zu „Improving TCP‘s Robustness to Blind In-Window Attacks“

„Einfache“ Angriffe

- Abhören von TCP bzw. darauf basierenden Klartextprotokollen
- Denial of Service durch Reflektion von Paketen (SYN-Floods auf geschlossene Ports) – besonders auf asynchrone Leitungen mit mehr Download- als Upload-Kapazität
- Denial of Service durch SYN-Floods auf offene Ports
- Übernehmen, Resetten, Einschleusen von Paketen im „lokalen“ Netz (Werkzeug „hunt“ von Pavel Krauz)

Abbrechen von TCP-Verbindungen

- Abbrechen durch eingeschleustes RST-Paket - > Verbindungsabbruch (analog auch Einfügen von Daten in TCP-Verbindungen)

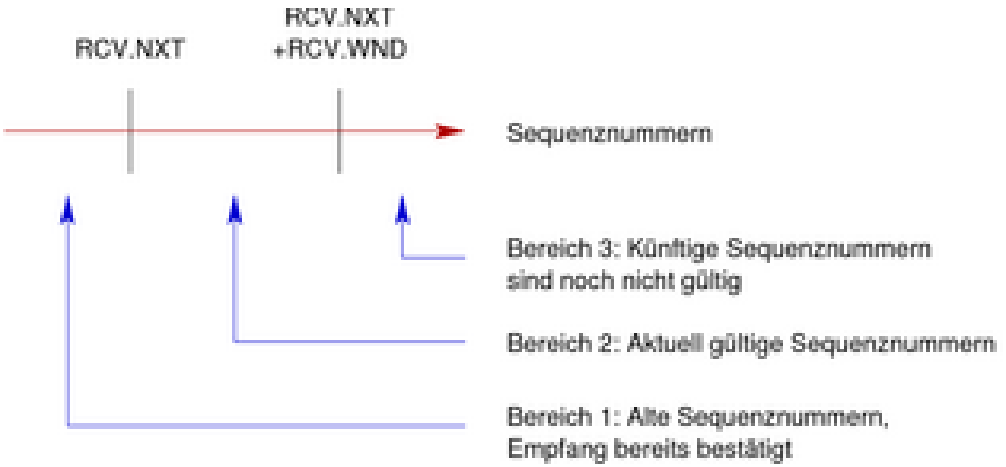
- Ziele (langlebige TCP-Verbindungen)
  - Border Gateway Protocol (BGP)
  - SSL, TLS, SSH, DNS-Zonentransfers, IRC-Server
  - Datenbankverbindungen, Online Spiele, ...

- Aufwand? (IPs unserer Opfer und Zielport sollen bekannt sein)
  - Sequenznummer raten – 2^{32} Möglichkeiten (0 bis 4.294.967.295)
  - Quellport raten – 2^{16} Möglichkeiten (0 bis 65.535)

- Theoretisch $2^{32} * 2^{16} / 2$ Pakete nötig (22.313 Jahre bei 128 KBit/s Verbindung)

Sequenznummern und Window Size

- Für diesen Fall gehen wir von komplett zufälligen initialen Sequenznummern aus (die Realität sieht leider anders aus, u.a. Für Win95, Win98, NT, 2000, MacOS, IRIX, HP/UP, ...)
- Aus RFC 793: Das Receive-Window ist der Bereich an Sequenznummern, die die lokale TCP-Instanz zu empfangen bereit ist. Das lokale TCP akzeptiert jedes Segment, dessen Sequenznummer mindestens der nächsten erwarteten Sequenznummer (RCV.NXT) entspricht, aber kleiner als RCV.NXT plus Fenstergröße (RCV.WND) ist.



(Quelle: Linuxmagazin 08/2005)

Initiale Window Size

- Maximale Größe der Window Size ist 2^{16}
- Initiale Windows Size ist abhängig vom Betriebssystem ...
  - Linux Kernel 2.4 / 2.6 — 5.840 Byte
  - Open BSD 3.6 — 16.382 Byte
  - Windows 2000 SP1, SP2 — 17.520 Byte
  - Windows 2000 SP3, SP4 — 65.535 Byte
  - Windows 2000, ab April 2005 — 17.520 Byte
  - Windows XP Prof., SP2 — 65.535 Byte
- ... und von der Applikation ...
  - Cisco Telnet — 4.192 Byte
  - Cisco BGP — 16.384 Byte

Vergrößern der Window Size

- Die Window Size wird während einer Verbindung oft noch vergrößert falls „größere“ Datenmengen übertragen werden
  - Zwei Linux-Rechner verbunden mit SSH (angezeigt wird die Ausgabe von „top“), initiale Windows Size 5.840, Wert nach einiger Zeit ca. 16.000
  - Linux-Rechner (5.840) per BGP mit Cisco-Router (16.384) nach wenigen Minuten 16.616 auf beiden Seiten

- RFC 1323 „TCP-Extensions for High Performance“ (Leitungen mit hoher Bandbreite und hoher Latenz) erlaubt Window Scaling (bei Windows zum Beispiel um 2^{14}, also Faktor 16.384, also 1 GigaByte Windows!!!)

- Notwendige Anzahl an geratenen Paketen um im richtigen Sequenznummern-bereich zu liegen ist im besten (schlechtesten?) Fall: $2^{32} / 2^{16} / 2^{14} = 2^2 = 4$

Rettet uns der Quellport?

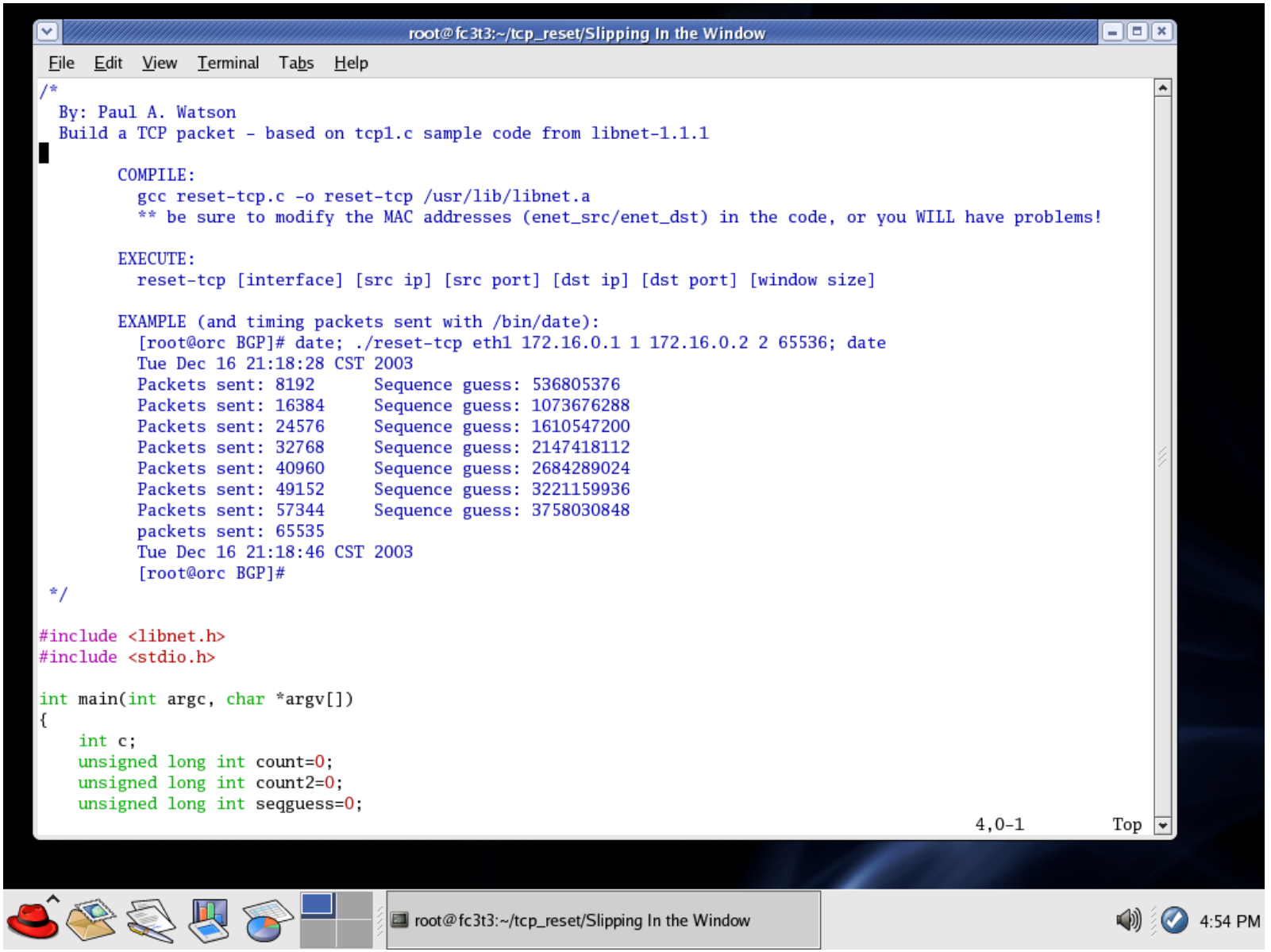
- Theoretisch sind für den Quellport 2^{16} (65.535) Werte denkbar (die 1.024 reservierten Werte ignorieren wir hier mal)
- Betriebssysteme wählen Ports allerdings nicht zufällig und nicht über den kompletten Wertebereich
  - Linux (Kernel 2.4 / 2.6) mit mindestens 128 MB Hauptspeicher (sonst kleinerer Bereich) verwendet Quellports zwischen 32.768 und 61.000 (Ports darüber werden für Masquerading benutzt) siehe „/proc/sys/net/ipv4/ip_local_range“, die übrigen 28.232 Werte werden beginnend bei 32.768 mit jeder Verbindung um 1 erhöht (nett für Angreifer)
  - Windows XP beginnt bei 1050 und erhöht ebenfalls immer um 1
  - Cisco beginnt ebenfalls bei einem festen Wert (abhängig von der Version des OS) und erhöht dann um 1 bzw. Um 512
- Anzahl der zu testenden Ports wird drastisch reduziert

TCP-Reset-Angriff

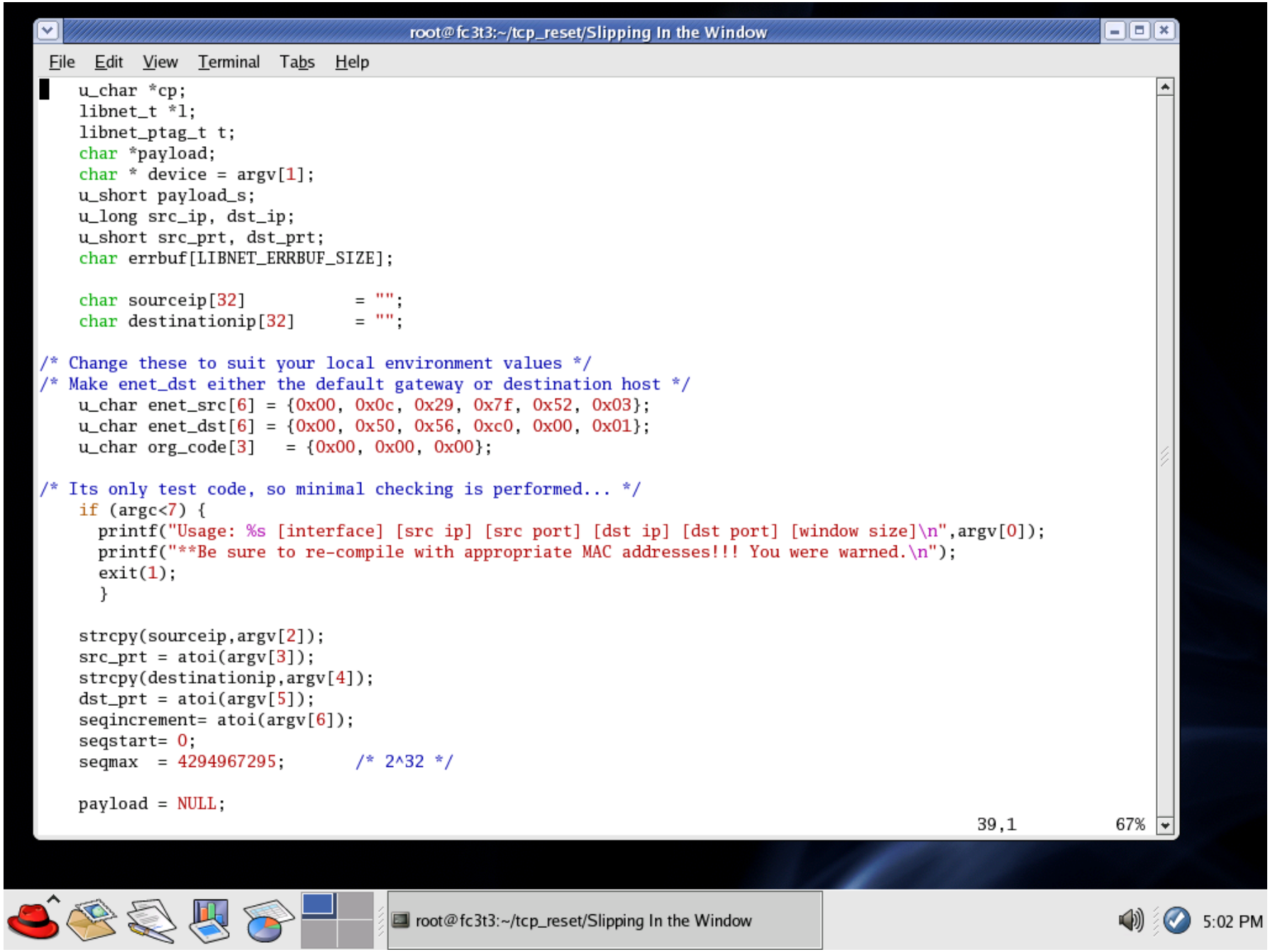
- TCP-Reset-Angriff auf eine TCP-Verbindung über typische T-DSL-Verbindung (128 Kbit/s Upstream)
  - “Erraten” werden muss die Sequenznummer (32 Bit)
  - Größe des Reset-Paketes (IP- und TCP-Header) 40 Byte (320 Bit)
  - Typische Window-Größe bei Linux 2.4 / 2.6 ist 5.840 Bit
  - Quellport bekannt (kann bestimmt werden):
    4.294.967.295/5.840*320/128.000/2 = 15:19 Minuten
  - Quellport unbekannt:
    919s * 65.535 = 697 Tage

TCP-Reset-Angriff in der Praxis

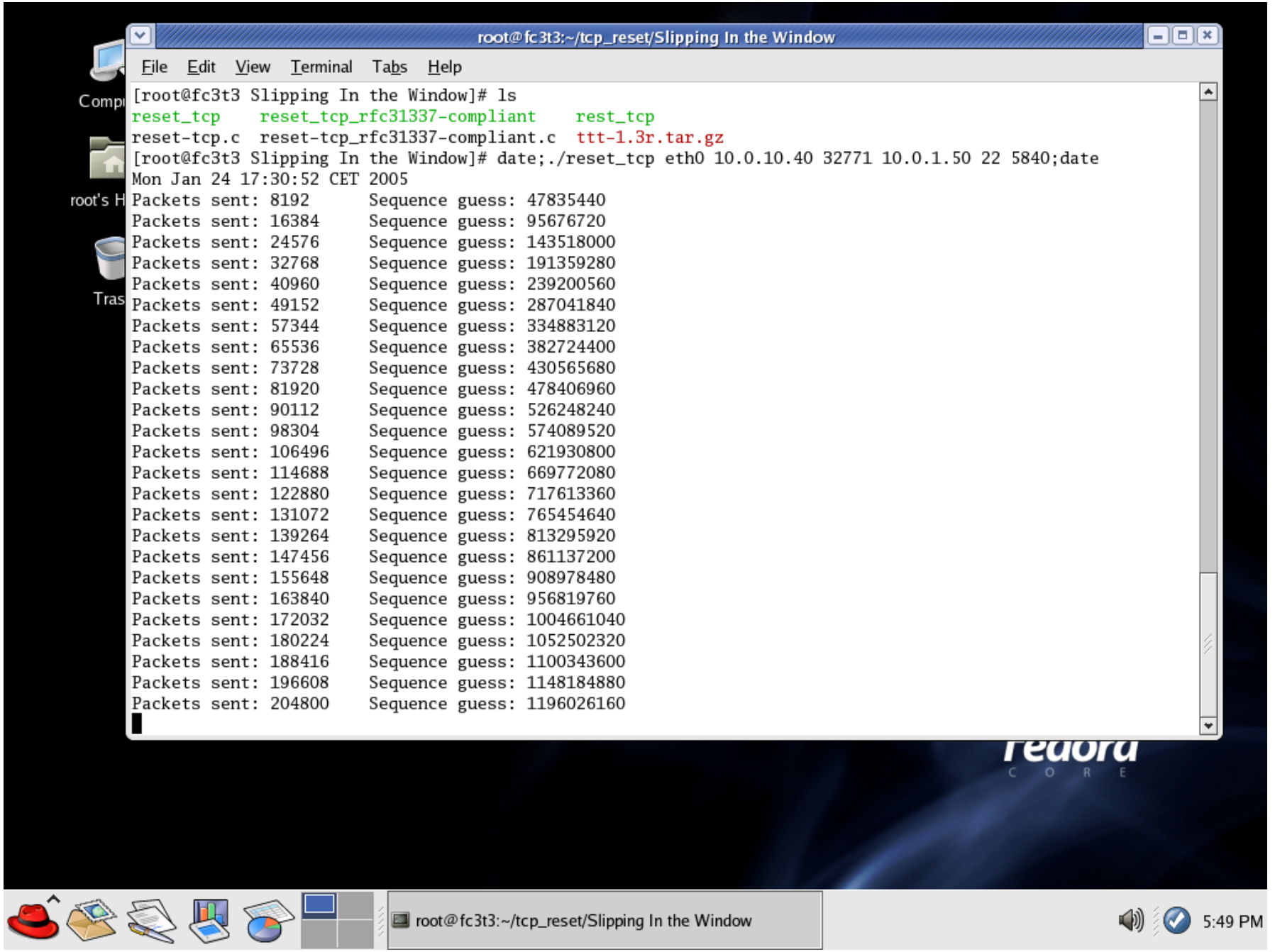
- Werkzeuge: „reset_tcp.c“ von Paul Watson und „Libnet Packet Constrution Library“



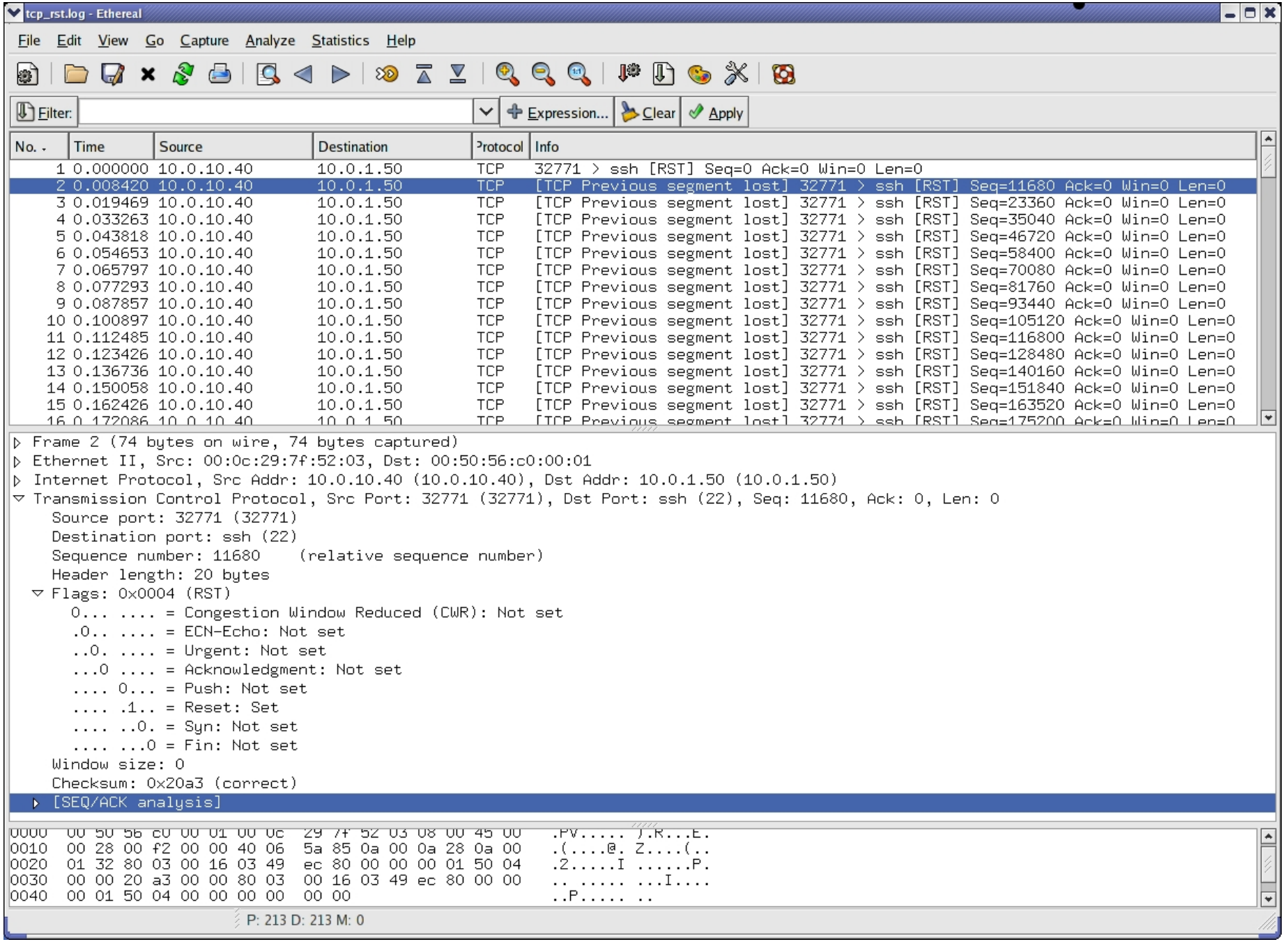
TCP-Reset-Angriff in der Praxis



TCP-Reset-Angriff in der Praxis

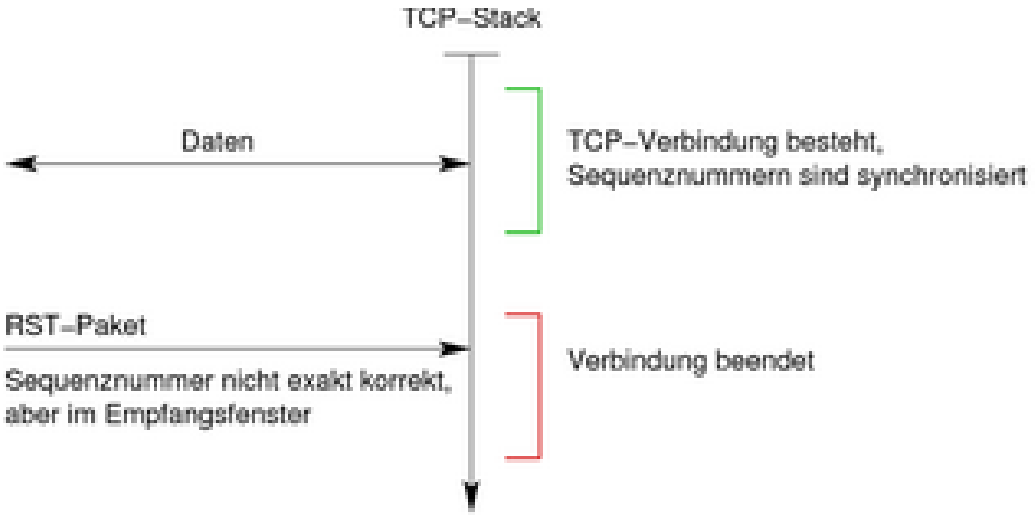


TCP-Reset-Angriff in der Praxis



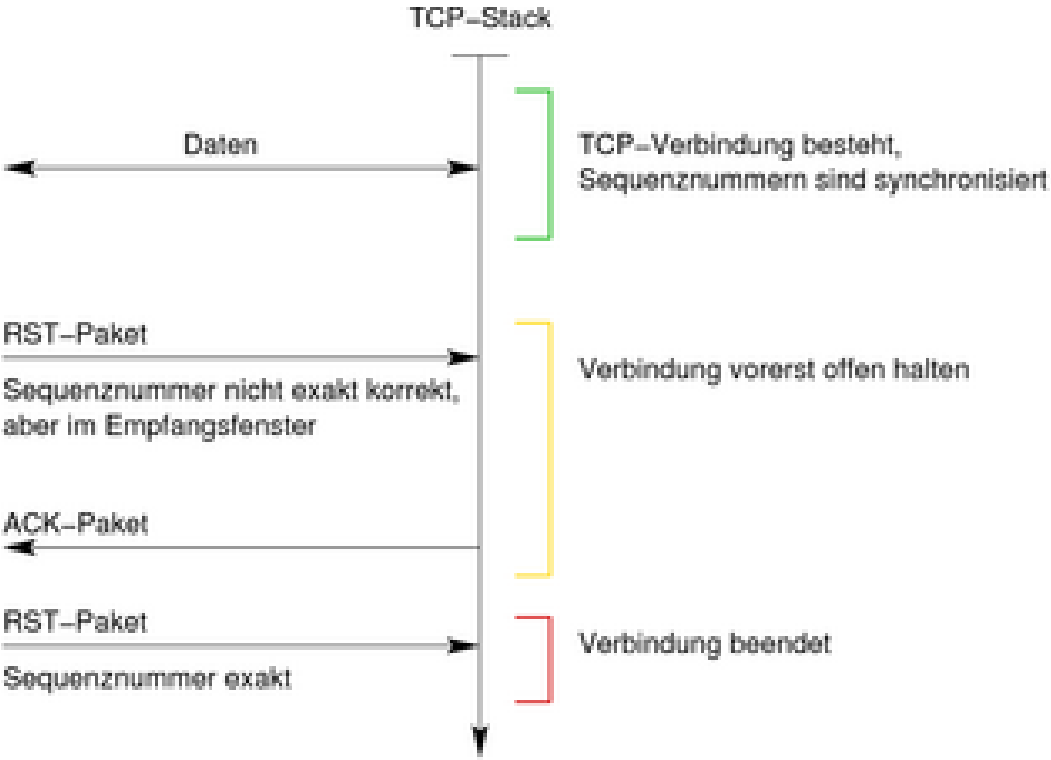
Wie schützen wir uns?

- Modifizieren des TCP-Stack nach IETF Draft (RST-Paket muss genau passen)



Klassischer TCP-Stack nach RFC 793

Modifizierter TCP-Stack nach Vorschlägen der IETF



(Quelle: Linuxmagazin 08/2005)

Randomisierung von TCP-Quell-Ports

wd@T42p:~

```
[wd@T42p ~]$ uname -a
Linux T42p.wdolle.de 2.6.9-1.681_FC3 #1 Thu Nov 18 15:10:10 EST 2004 i686 i686 i386 GNU/Linux
[wd@T42p ~]$ netstat -nt
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 192.168.53.1:32771          192.168.53.150:22           ESTABLISHED
tcp        0      0 192.168.53.1:32773          192.168.53.150:22           ESTABLISHED
tcp        0      0 192.168.53.1:32772          192.168.53.150:22           ESTABLISHED
tcp        0      0 192.168.53.1:32775          192.168.53.150:22           ESTABLISHED
tcp        0      0 192.168.53.1:32774          192.168.53.150:22           ESTABLISHED
tcp        0      0 192.168.53.1:32777          192.168.53.150:22           ESTABLISHED
tcp        0      0 192.168.53.1:32776          192.168.53.150:22           ESTABLISHED
[wd@T42p ~]$
```

Linux:
von 32.768
bis 61.000
(jeweils um
1 erhöhen)

wd@bgpd:~

```
[wd@bgpd wd]$ netstat -atn | grep 127.0.0.1
tcp        0      0 127.0.0.1:6868              0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:6969              0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:22                127.0.0.1:59008             ESTABLISHED
tcp        0      0 127.0.0.1:46953             127.0.0.1:22                TIME_WAIT
tcp        0      0 127.0.0.1:38280             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:43525             ESTABLISHED
tcp        0      0 127.0.0.1:53356             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:45158             ESTABLISHED
tcp        0      0 127.0.0.1:45158             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:45150             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:38280             ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:42504             ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:53356             ESTABLISHED
tcp        0      0 127.0.0.1:59008             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:42504             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:43525             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:51999             127.0.0.1:22                ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:45150             ESTABLISHED
tcp        0      0 127.0.0.1:22                127.0.0.1:51999             ESTABLISHED
[wd@bgpd wd]$
```

Linux mit
grsecurity-
Patch;
OpenBSD hat
dieses Feature
eingebaut

Vielen Dank für die Aufmerksamkeit

Folien zeitnah unter: http://www.dolle.net

Artikel zum Thema: “TCP-Reset – Gefahren im TCP-Protokoll: Angreifer kappen fremde Verbindungen“; Christoph Wegener, Wilhelm Dolle; Linux-Magazin 08/05

Wilhelm Dolle, CISA, CISSP, BSI IT-Grundschutz-Auditor

Director Information Technology

mail **wilhelm.dolle@dolle.net**

web **http://www.dolle.net**