

Hacking Workshop – Mathecamp 2016 in Windischleuba

Sven Prüfer

August 13, 2016

1 Hinweise

2 Linux

- System

- Kommandozeile

3 Grundlagen Netzwurkkommunikation

- IPs und DNS

- Internetprotokolle

4 Wichtigste Systeme im Internet

5 Wichtige Kommandozeilenwerkzeuge

- Kommunikation über Netzwerke

- Reconnaissance

6 Verschiedene Attacken

- Allgemeines Vorgehen

- ARP Spoofing

- WEP Verschlüsselung

- Websites

- DNS Tunneling

Hinweise

Macht niemals irgendsoetwas auf Rechnern, auf denen ihr das nicht dürft oder von deren Betreibern ihr kein Einverständnis habt.

Macht niemals irgendsoetwas auf Rechnern, auf denen ihr das nicht dürft oder von deren Betreibern ihr kein Einverständnis habt.

Und auf gar keinen Fall in der Schule!

Viele Menschen wollen euch Böses!

Viele Menschen wollen euch Böses!

Traut keinen zwielichten Websites, installiert niemals (besonders unter Windows) merkwürdige Programme!

Viele Menschen wollen euch Böses!

Traut keinen zwielichten Websites, installiert niemals (besonders unter Windows) merkwürdige Programme!

Informiert euch unbedingt über Skripte und Programme, bevor ihr sie ausführt!

Viele Menschen wollen euch Böses!

Traut keinen zweifelhaften Websites, installiert niemals (besonders unter Windows) merkwürdige Programme!

Informiert euch unbedingt über Skripte und Programme, bevor ihr sie ausführt!

Vertrauenswürdige Websites sind insbesondere
STACKOVERFLOW.COM, SUPERUSER.COM oder
NEWS.YCOMBINATOR.COM.

Linux – System

“Alles ist eine Datei” – Grundprinzip von Unix

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Wichtige Verzeichnisse sind insbesondere:

/dev

Geräte

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Wichtige Verzeichnisse sind insbesondere:

/dev

Geräte

/media

Medien

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Wichtige Verzeichnisse sind insbesondere:

/dev	Geräte
------	--------

/media	Medien
--------	--------

/home	Private Dateien der Nutzer
-------	----------------------------

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Wichtige Verzeichnisse sind insbesondere:

/dev Geräte

/media Medien

/home Private Dateien der Nutzer

/etc Konfigurationsdateien, insb. /etc/ssl

/var Variable Dateien, insb. /var/www

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Wichtige Verzeichnisse sind insbesondere:

/dev	Geräte
/media	Medien
/home	Private Dateien der Nutzer
/etc	Konfigurationsdateien, insb. /etc/ssl
/var	Variable Dateien, insb. /var/www
/bin	Binäre Dateien

Dateistruktur

“Alles ist eine Datei” – Grundprinzip von Unix

Das Wurzelverzeichnis ist “/” anstelle einer Partition (“C” unter Windows).

Wichtige Verzeichnisse sind insbesondere:

/dev	Geräte
/media	Medien
/home	Private Dateien der Nutzer
/etc	Konfigurationsdateien, insb. /etc/ssl
/var	Variable Dateien, insb. /var/www
/bin	Binäre Dateien
/tmp	Temporäre Dateien

Benutzerrechte

Dateisystem speichert Lese-/Schreib-/Nutzungsrechte für jede einzelne Datei und jeden Ordner

Benutzerrechte

Dateisystem speichert Lese-/Schreib-/Nutzungsrechte für jede einzelne Datei und jeden Ordner

Bedeutung von Rechten bei Verzeichnissen anders.

Benutzerrechte

Dateisystem speichert Lese-/Schreib-/Nutzungsrechte für jede einzelne Datei und jeden Ordner

Bedeutung von Rechten bei Verzeichnissen anders.

Bei guter Nutzung von Rechten kann Eindringling im besten Fall nichts machen.

Benutzerrechte

Dateisystem speichert Lese-/Schreib-/Nutzungsrechte für jede einzelne Datei und jeden Ordner

Bedeutung von Rechten bei Verzeichnissen anders.

Bei guter Nutzung von Rechten kann Eindringling im besten Fall nichts machen.

Wichtigster Nutzer: *root*

Benutzerrechte

Dateisystem speichert Lese-/Schreib-/Nutzungsrechte für jede einzelne Datei und jeden Ordner

Bedeutung von Rechten bei Verzeichnissen anders.

Bei guter Nutzung von Rechten kann Eindringling im besten Fall nichts machen.

Wichtigster Nutzer: *root*

Beispiel in Konsole.

Die Kommandozeile

Terminal, Bash und Shell

Eine *Shell* verarbeitet Kommandozeilenbefehle und gibt eine Antwort.

Terminal, Bash und Shell

Eine *Shell* verarbeitet Kommandozeilenbefehle und gibt eine Antwort.

Die *Bash* ist die bekannteste Shell. Es gibt noch viele andere.

Terminal, Bash und Shell

Eine *Shell* verarbeitet Kommandozeilenbefehle und gibt eine Antwort.

Die *Bash* ist die bekannteste Shell. Es gibt noch viele andere.

Ein *Terminal* ist eine Art Verpackung für eine Shell, also z.B. das Fenster in dem die Shell läuft.

Wichtigste Befehle

cd	Wechsle Verzeichnis
ls	Zeige Verzeichnisinhalt
cat	Zeige/Gib wieder Inhalt von Textdateien an
man	Zeige Hilfe zu Befehl an
python/perl/gcc	Kompiliere mit entsprechender Sprache
sh	Führe Shellskript aus
DATEI	Führe binäre DATEI aus
make	Führe make Skript aus
bc -l	Taschenrechner :-)

Pipes

Befehle in der Bash können hintereinander ausgeführt werden mittels einer Pipe "|". Diese gibt die Ausgabe als Eingabe an den nächsten Befehl weiter.

Pipes

Befehle in der Bash können hintereinander ausgeführt werden mittels einer Pipe "|". Diese gibt die Ausgabe als Eingabe an den nächsten Befehl weiter.

```
cat testdatei | uniq -u | sort
```

Pipes

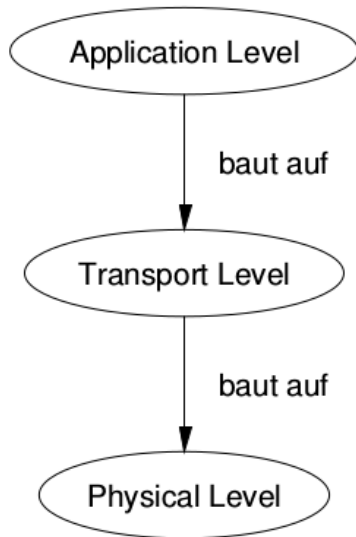
Befehle in der Bash können hintereinander ausgeführt werden mittels einer Pipe "|". Diese gibt die Ausgabe als Eingabe an den nächsten Befehl weiter.

```
cat testdatei | uniq -u | sort
```

Gibt den Inhalt der Datei "testdatei" weiter an "uniq" mit Option "-u", doppelte Zeilen werden weggeschmissen und danach sortiert.

Grundlagen Netzwerkkommunikation

Schichtenmodell



RFC

RFC = Request for Comments

RFC

RFC = Request for Comments

Internetstandards werden damit (in einfacher Textdatei) vorgeschlagen und zur Diskussion gestellt.

RFC

RFC = Request for Comments

Internetstandards werden damit (in einfacher Textdatei) vorgeschlagen und zur Diskussion gestellt.

De facto werden Internetstandards damit definiert.

RFC = Request for Comments

Internetstandards werden damit (in einfacher Textdatei) vorgeschlagen und zur Diskussion gestellt.

De facto werden Internetstandards damit definiert.

	INTERNET STANDARD
	Errata Exist
Network Working Group	Vint Cerf
Request for Comments: 20	UCLA
	October 16, 1969

ASCII format for Network Interchange

For concreteness, we suggest the use of standard 7-bit ASCII embedded in an 8 bit byte whose high order bit is always 0. This leads to the standard code given on the attached page, copies from USAS X3, 4-1968. This code will be used over HOST-HOST primary connections. Break characters will be defined by the receiving remote host, e.g. SRI uses "." (ASCII X'2E' or 2/14) as the end-of-line character, where as UCLA uses X'0D' or 0/13 (carriage return).

USA Standard Code for Information Interchange

1. Scope

This coded character set is to be used for the general interchange of information among information processing systems, communication systems, and associated equipment.

IPs und DNS

IP-Protokoll

Grundlegendes Protokoll um Pakete vom Quell-Host zum Ziel-Host zu senden.

IP-Protokoll

Grundlegendes Protokoll um Pakete vom Quell-Host zum Ziel-Host zu senden.

Pakete bestehen aus "Header" und "Payload".

IP-Protokoll

Grundlegendes Protokoll um Pakete vom Quell-Host zum Ziel-Host zu senden.

Pakete bestehen aus "Header" und "Payload".

Es existieren zwei wichtige Versionen: IPv4 und IPv6.

IP-Protokoll

Grundlegendes Protokoll um Pakete vom Quell-Host zum Ziel-Host zu senden.

Pakete bestehen aus "Header" und "Payload".

Es existieren zwei wichtige Versionen: IPv4 und IPv6.

Beispiel:

```
45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10  
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00  
F9 46 00 00 02 04 05 B4
```

Analyse etwas später!

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

10 – Type of Service (?)

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

10 – Type of Service (?)

00 2C – Länge des Pakets \Rightarrow 44 Bytes

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

10 – Type of Service (?)

00 2C – Länge des Pakets \Rightarrow 44 Bytes

24 B2 – Durchnummerierung der Pakete

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

10 – Type of Service (?)

00 2C – Länge des Pakets \Rightarrow 44 Bytes

24 B2 – Durchnummerierung der Pakete

00 00 (Verschiedene IP Flags)

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

10 – Type of Service (?)

00 2C – Länge des Pakets \Rightarrow 44 Bytes

24 B2 – Durchnummerierung der Pakete

00 00 (Verschiedene IP Flags)

40 – "Time to live"

Beispiel IPv4 Paket 1

Beispiel (jedes "Paar" sind zwei Hexadezimalzahlen, also jeweils vier Bit, also insgesamt 1 Byte) von Michael Egan:

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

4 – IPv4

5 – Länge des IP Headers in 32-Bit-Wörtern \Rightarrow 20 Bytes

10 – Type of Service (?)

00 2C – Länge des Pakets \Rightarrow 44 Bytes

24 B2 – Durchnummerierung der Pakete

00 00 (Verschiedene IP Flags)

40 – "Time to live"

06 – Protokoll \Rightarrow TCP

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

AC 10 00 01 – Ziel-IP-Adresse \Rightarrow 172.16.0.1

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

AC 10 00 01 – Ziel-IP-Adresse \Rightarrow 172.16.0.1

Beginn TCP-Header: 04 47 – Quellport \Rightarrow 1095

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

AC 10 00 01 – Ziel-IP-Adresse \Rightarrow 172.16.0.1

Beginn TCP-Header: 04 47 – Quellport \Rightarrow 1095

00 17 – Zielport \Rightarrow 23 (Telnet)

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

AC 10 00 01 – Ziel-IP-Adresse \Rightarrow 172.16.0.1

Beginn TCP-Header: 04 47 – Quellport \Rightarrow 1095

00 17 – Zielport \Rightarrow 23 (Telnet)

60 C6 DF 90 – Sequenznummer SEQ#

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

AC 10 00 01 – Ziel-IP-Adresse \Rightarrow 172.16.0.1

Beginn TCP-Header: 04 47 – Quellport \Rightarrow 1095

00 17 – Zielport \Rightarrow 23 (Telnet)

60 C6 DF 90 – Sequenznummer SEQ#

00 00 00 00 – Bestätigungsnummer ACK# (normalerweise
SEQ# des vorherigen Pakets, hier aber erstes Paket)

Beispiel IPv4 Paket 2

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

FD DF – Checksum des IP-Headers

AC 10 00 09 – Quell-IP-Adresse \Rightarrow 172.16.0.9

AC 10 00 01 – Ziel-IP-Adresse \Rightarrow 172.16.0.1

Beginn TCP-Header: 04 47 – Quellport \Rightarrow 1095

00 17 – Zielport \Rightarrow 23 (Telnet)

60 C6 DF 90 – Sequenznummer SEQ#

00 00 00 00 – Bestätigungsnummer ACK# (normalerweise
SEQ# des vorherigen Pakets, hier aber erstes Paket)

6 – Länge des TCP Headers in 32-Bit-Wörtern \Rightarrow 24 Bytes

Beispiel IPv4 Paket 3

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

0 02 – TCP Flags (?)

Beispiel IPv4 Paket 3

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

0 02 – TCP Flags (?)

02 00 – Fenstergröße für sogenanntes “Sliding Window
Protocoll” zum Verhindern vom Senden zuvieler Pakete

Beispiel IPv4 Paket 3

45 10 00 2C 24 B2 00 00 40 06 FD DF AC 10 00 09 AC 10
00 01 04 47 00 17 60 C6 DF 90 00 00 00 00 60 02 02 00
F9 46 00 00 02 04 05 B4

0 02 – TCP Flags (?)

02 00 – Fenstergröße für sogenanntes “Sliding Window
Protocoll” zum Verhindern vom Senden zuvieler Pakete

F9 46 – Checksum TCP-Header

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen:

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen: 10.0.0.0/8,

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen: 10.0.0.0/8, 172.16.0/12,

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen: 10.0.0.0/8, 172.16.0/12, 192.168.0/16

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen: 10.0.0.0/8, 172.16.0/12, 192.168.0/16

IPv4-Adressen sind alle vergeben :-(

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen: 10.0.0.0/8, 172.16.0/12, 192.168.0/16

IPv4-Adressen sind alle vergeben :-(

Aber es gibt IPv6! :-)

IP-Adressen

IPv4 nutzt 4 Bytes um Rechner zu adressieren, z.B.
5.189.172.46.

Notation: 192.0.2.0/24 bezeichnet alle Adressen 192.0.2.0
bis 192.0.2.255

Spezielle Adressen: 10.0.0.0/8, 172.16.0/12, 192.168.0/16

IPv4-Adressen sind alle vergeben :-(

Aber es gibt IPv6! :-)

IPv6 nutzt 16 Bytes, typischerweise in Hexadezimal und
ohne Nullen, z.B. 2001:0DB8:AC10:FE01:::

Routing

Rechner am Internet schicken nicht jedes Paket direkt an ihr Ziel

Routing

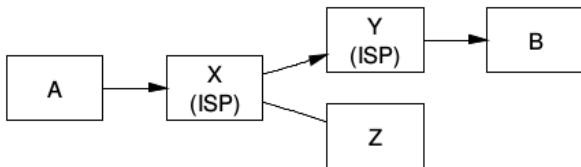
Rechner am Internet schicken nicht jedes Paket direkt an ihr Ziel

Stattdessen hat jeder Rechner eine *Routing-Tabelle* und schickt Pakete nach einem Routingprotokoll.

Routing

Rechner am Internet schicken nicht jedes Paket direkt an ihr Ziel

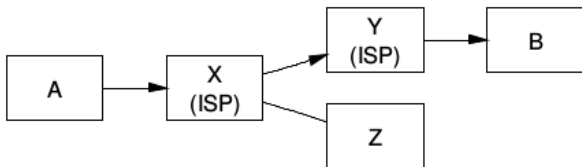
Stattdessen hat jeder Rechner eine *Routing-Tabelle* und schickt Pakete nach einem Routingprotokoll.



Routing

Rechner am Internet schicken nicht jedes Paket direkt an ihr Ziel

Stattdessen hat jeder Rechner eine *Routing-Tabelle* und schickt Pakete nach einem Routingprotokoll.

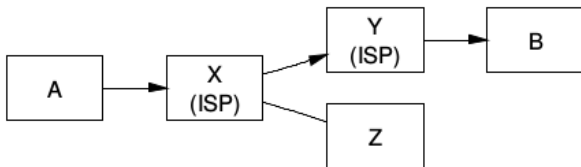


Vor 1993: IPs hierarchisch nach Größe der Netzwerke aufgeteilt ("classfull") ⇒ Riesige Routingtables und nicht genügend IPs für LANs

Routing

Rechner am Internet schicken nicht jedes Paket direkt an ihr Ziel

Stattdessen hat jeder Rechner eine *Routing-Tabelle* und schickt Pakete nach einem Routingprotokoll.



Vor 1993: IPs hierarchisch nach Größe der Netzwerke aufgeteilt ("classfull") \Rightarrow Riesige Routingtables und nicht genügend IPs für LANs

Nach 1993: CIDR \Rightarrow Netzwerk- und Hostanteil der IP.

TCP 1

TCP = Transmission Control Protocol

TCP 1

TCP = Transmission Control Protocol

Erweiterung von IP um

TCP = Transmission Control Protocol

Erweiterung von IP um

- Fehlerkorrektur (Reihenfolge und Wiederholung) durch Nummerierung der Pakete

TCP = Transmission Control Protocol

Erweiterung von IP um

- Fehlerkorrektur (Reihenfolge und Wiederholung) durch Nummerierung der Pakete
- Ports

TCP = Transmission Control Protocol

Erweiterung von IP um

- Fehlerkorrektur (Reihenfolge und Wiederholung) durch Nummerierung der Pakete
- Ports
- "Verbindungsauf- und abbau"

TCP = Transmission Control Protocol

Erweiterung von IP um

- Fehlerkorrektur (Reihenfolge und Wiederholung) durch Nummerierung der Pakete
- Ports
- "Verbindungsauf- und abbau"

TCP hat eigenen Header, der innerhalb der IP-Payload liegt.

TCP = Transmission Control Protocol

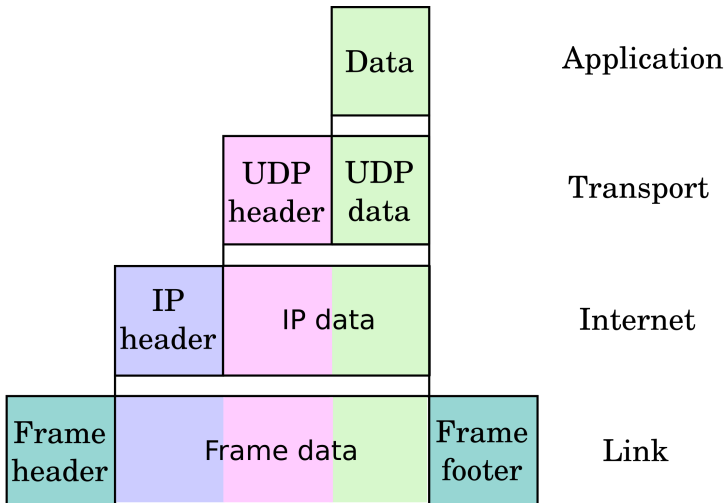
Erweiterung von IP um

- Fehlerkorrektur (Reihenfolge und Wiederholung) durch Nummerierung der Pakete
- Ports
- "Verbindungsauf- und abbau"

TCP hat eigenen Header, der innerhalb der IP-Payload liegt.

TCP regelt "alles" für die Anwendungen: Mittels TCP/IP werden Pakete verschickt bis alles vollständig und korrekt ist, erst dann erhält die Anwendung die Daten.

TCP 2



Zur Fehlerkorrektur werden Pakete durchnummeriert.

Zur Fehlerkorrektur werden Pakete durchnummeriert.

Dazu gibt es SYN und ACK Einträge im TCP-Header:

Zur Fehlerkorrektur werden Pakete durchnummeriert.

Dazu gibt es SYN und ACK Einträge im TCP-Header: Das sind jeweils Zahlen von XXX bis XXX.

Zur Fehlerkorrektur werden Pakete durchnummeriert.

Dazu gibt es SYN und ACK Einträge im TCP-Header: Das sind jeweils Zahlen von XXX bis XXX.

Außerdem gibt es SYN- und ACK-Pakete:

Diese haben keine Payload, aber übermitteln im Header die SYN# und ACK#.

Zur Fehlerkorrektur werden Pakete durchnummeriert.

Dazu gibt es SYN und ACK Einträge im TCP-Header: Das sind jeweils Zahlen von XXX bis XXX.

Außerdem gibt es SYN- und ACK-Pakete:

Diese haben keine Payload, aber übermitteln im Header die SYN# und ACK#.

Antwortpaket hat als ACK# die SYN# des vorigen Anfragepakets \Rightarrow Reihenfolge der Pakete rekapitulierbar.

Zur Fehlerkorrektur werden Pakete durchnummeriert.

Dazu gibt es SYN und ACK Einträge im TCP-Header: Das sind jeweils Zahlen von XXX bis XXX.

Außerdem gibt es SYN- und ACK-Pakete:

Diese haben keine Payload, aber übermitteln im Header die SYN# und ACK#.

Antwortpaket hat als ACK# die SYN# des vorigen Anfragepakets \Rightarrow Reihenfolge der Pakete rekapitulierbar.

Beim Verbindungsaufbau wird mittels ACK- und SYN-Paketen "synchronisiert" und der Aufbau bestätigt.

Nummerierung der Pakete erfolgt fortlaufend.

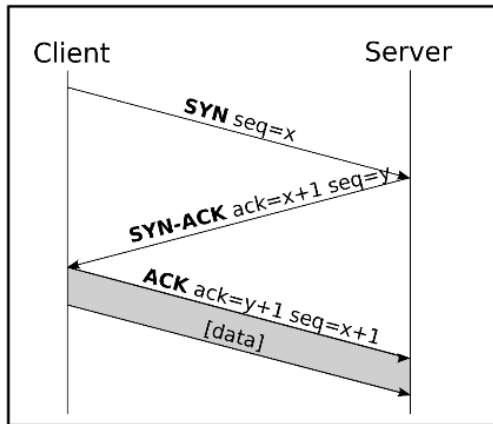
TCP 4

Nummerierung der Pakete erfolgt fortlaufend. Es werden SYN# und ACK# als Nummerierung und Bestätigung/Vorgänger geschickt.

TCP 4

Nummerierung der Pakete erfolgt fortlaufend. Es werden SYN# und ACK# als Nummerierung und Bestätigung/Vorgänger geschickt.

“Handshake”:



UDP

UDP = “User Datagram Protocoll”

UDP

UDP = "User Datagram Protocoll"

Benutzt auch Ports aber hat keine Fehlerkorrektur.

UDP

UDP = "User Datagram Protocoll"

Benutzt auch Ports aber hat keine Fehlerkorrektur.

UDP hat keine Verbingung im engeren Sinne.

UDP

UDP = "User Datagram Protocoll"

Benutzt auch Ports aber hat keine Fehlerkorrektur.

UDP hat keine Verbingung im engeren Sinne.

UDP ist dadurch viel schneller.

UDP = "User Datagram Protocoll"

Benutzt auch Ports aber hat keine Fehlerkorrektur.

UDP hat keine Verbingung im engeren Sinne.

UDP ist dadurch viel schneller.

Wird dort benutzt, wo der Verlust von einzelnen Paketen kein Problem ist, aber Geschwindigkeit zählt.

UDP

UDP = "User Datagram Protocoll"

Benutzt auch Ports aber hat keine Fehlerkorrektur.

UDP hat keine Verbingung im engeren Sinne.

UDP ist dadurch viel schneller.

Wird dort benutzt, wo der Verlust von einzelnen Paketen kein Problem ist, aber Geschwindigkeit zählt.

Spiele, Streaming, etc.

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Es gibt 65532 Ports, typischerweise mit 192.168.1.1:**80** bezeichnet.

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Es gibt 65532 Ports, typischerweise mit 192.168.1.1:**80** bezeichnet.

Typische Ports: 80 – http,

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Es gibt 65532 Ports, typischerweise mit 192.168.1.1:**80** bezeichnet.

Typische Ports: 80 – http, 22 – ssh,

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Es gibt 65532 Ports, typischerweise mit 192.168.1.1:**80** bezeichnet.

Typische Ports: 80 – http, 22 – ssh, 443 – https,

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Es gibt 65532 Ports, typischerweise mit 192.168.1.1:**80** bezeichnet.

Typische Ports: 80 – http, 22 – ssh, 443 – https, 23 – telnet,

Ports

Jeder Rechner (adressiert durch seine IP) hat sogenannte Ports, die von Anwendungen reserviert werden.

Beide Hosts benutzen ggf. verschiedene Ports.

Dadurch kann ein Rechner gleichzeitig mit mehreren Anwendungen/Servern kommunizieren.

Es gibt 65532 Ports, typischerweise mit 192.168.1.1:**80** bezeichnet.

Typische Ports: 80 – http, 22 – ssh, 443 – https, 23 – telnet, 993 – smtp, etc.

ICMP

ICMP = "Internet Control Message Protocol"

ICMP = "Internet Control Message Protocol"

Metaprotokoll, um Informationen über Server und Verbindungen auszutauschen, insbesondere IP, TCP und UDP.

ICMP = "Internet Control Message Protocol"

Metaprotokoll, um Informationen über Server und Verbindungen auszutauschen, insbesondere IP, TCP und UDP.

Beispiele für Pakete:

- Echo Request/Reply

ICMP = "Internet Control Message Protocol"

Metaprotokoll, um Informationen über Server und Verbindungen auszutauschen, insbesondere IP, TCP und UDP.

Beispiele für Pakete:

- Echo Request/Reply
- Destination Network/Host/Port Unreachable

ICMP = "Internet Control Message Protocol"

Metaprotokoll, um Informationen über Server und Verbindungen auszutauschen, insbesondere IP, TCP und UDP.

Beispiele für Pakete:

- Echo Request/Reply
- Destination Network/Host/Port Unreachable
- Time Exceeded

ICMP = "Internet Control Message Protocol"

Metaprotokoll, um Informationen über Server und Verbindungen auszutauschen, insbesondere IP, TCP und UDP.

Beispiele für Pakete:

- Echo Request/Reply
- Destination Network/Host/Port Unreachable
- Time Exceeded

Fehler bei ICMP Paketen werden nicht nochmal gemeldet.
:-)

Pings

Ping = "ICMP-Echo-Request/Reply".

Pings

Ping = "ICMP-Echo-Request/Reply".

Anfrage an Server, ob dieser online ist.

Pings

Ping = "ICMP-Echo-Request/Reply".

Anfrage an Server, ob dieser online ist. Falls ja, sendet dieser eine Antwort.

Pings

Ping = "ICMP-Echo-Request/Reply".

Anfrage an Server, ob dieser online ist. Falls ja, sendet dieser eine Antwort.

Blockieren von pings ist kein Sicherheitsgewinn: Man kann auch anders herausfinden, ob ein Server online ist.

Pings

Ping = "ICMP-Echo-Request/Reply".

Anfrage an Server, ob dieser online ist. Falls ja, sendet dieser eine Antwort.

Blockieren von pings ist kein Sicherheitsgewinn: Man kann auch anders herausfinden, ob ein Server online ist.

Typisches Beispiel:

"ping 8.8.8.8" um zu schauen, ob Google erreichbar ist und damit das Internet funktioniert :-)

Time to live

Jedes IP-Paket hat eine "Time-to-live", eine Zahl von 0 bis 255.

Time to live

Jedes IP-Paket hat eine "Time-to-live", eine Zahl von 0 bis 255.

Bei jeder Weiterleitung beim Routing ("Hop") wird die Zahl um eins vermindert.

Time to live

Jedes IP-Paket hat eine "Time-to-live", eine Zahl von 0 bis 255.

Bei jeder Weiterleitung beim Routing ("Hop") wird die Zahl um eins vermindert.

Sobald $TTL = 0$ ist, wird Paket "dropped", also gelöscht.

Time to live

Jedes IP-Paket hat eine "Time-to-live", eine Zahl von 0 bis 255.

Bei jeder Weiterleitung beim Routing ("Hop") wird die Zahl um eins vermindert.

Sobald $TTL = 0$ ist, wird Paket "dropped", also gelöscht.

Dadurch kann ein Paket nicht in einer Endlosschleife im Routing hin- und hergeschickt werden.

Time to live

Jedes IP-Paket hat eine "Time-to-live", eine Zahl von 0 bis 255.

Bei jeder Weiterleitung beim Routing ("Hop") wird die Zahl um eins vermindert.

Sobald $TTL = 0$ ist, wird Paket "dropped", also gelöscht.

Dadurch kann ein Paket nicht in einer Endlosschleife im Routing hin- und hergeschickt werden.

Anwendungen: Finden von Servern des Ziels mittels *traceroute*, OS-Erkennung anhand von typischen Standardwerten für TTL.

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel:

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: <http://sven.musmehl.de/hacking/index.html>

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: <http://sven.musmehl.de/hacking/index.html>

http – Protokoll

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: <http://sven.musmehl.de/hacking/index.html>

http – Protokoll

de – Topleveldomain

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: <http://sven.musmehl.de/hacking/index.html>

http – Protokoll

de – Toplevel domain

sven.musmehl.de – Hostname

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: <http://sven.musmehl.de/hacking/index.html>

http – Protokoll

de – Topleveldomain

sven.musmehl.de – Hostname

/hacking/ – Unterverzeichnis/Pfad

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: <http://sven.musmehl.de/hacking/index.html>

http – Protokoll

de – Topleveldomain

sven.musmehl.de – Hostname

/hacking/ – Unterverzeichnis/Pfad

index.html – Zieldatei

URLs

IPs kompliziert zu merken \Rightarrow Nutze URLs als Adressen für Menschen.

Beispiel: `http://sven.musmehl.de/hacking/index.html`

`http` – Protokoll

`de` – Toplevel domain

`sven.musmehl.de` – Hostname

`/hacking/` – Unterverzeichnis/Pfad

`index.html` – Zieldatei

Dahinter können noch Parameter mittels URL-Encoding übergeben werden \Rightarrow Nachdem wir das `http` Protokoll verstanden haben.

Ganz allgemeine “Uniform Resource Identifiers”:

Ganz allgemeine “Uniform Resource Identifiers”:

scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

Beispiel: linide.sf.net. (Punkt am Ende!)

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

Beispiel: linide.sf.net. (Punkt am Ende!)

Anfrage an . : Was ist IP von net.?

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

Beispiel: linide.sf.net. (Punkt am Ende!)

Anfrage an . : Was ist IP von net.?

Anfrage an Nameserver von net. : Was ist IP vom Nameserver von sf.net.?

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

Beispiel: linide.sf.net. (Punkt am Ende!)

Anfrage an . : Was ist IP von net.?

Anfrage an Nameserver von net. : Was ist IP vom Nameserver von sf.net.?

Anfrage an Nameserver von sf.net. : Was ist IP von linide.sf.net.?

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

Beispiel: linide.sf.net. (Punkt am Ende!)

Anfrage an . : Was ist IP von net.?

Anfrage an Nameserver von net. : Was ist IP vom Nameserver von sf.net.?

Anfrage an Nameserver von sf.net. : Was ist IP von linide.sf.net.?

Danach werden "normale" http Anfragen und ähnliches an IP von linide.sf.net gesendet.

DNS 1

Protokoll um Hostnamen und URLs in IPs aufzulösen.

Anstatt eine Datei ("hostfile") mit allen IPs an alle Rechner zu schicken, werden diese Auflösungen hierarchisch verwaltet ("DNS-Einträge") und abgefragt.

Beispiel: linide.sf.net. (Punkt am Ende!)

Anfrage an . : Was ist IP von net.?

Anfrage an Nameserver von net. : Was ist IP vom Nameserver von sf.net.?

Anfrage an Nameserver von sf.net. : Was ist IP von linide.sf.net.?

Danach werden "normale" http Anfragen und ähnliches an IP von linide.sf.net gesendet.

Caching-Nameserver speichern Anfragen zwischen um diese schneller zu beantworten.

Beispiele für DNS-Einträge:

DNS 2

Beispiele für DNS-Einträge:

A – Symbolischer Name → IP

Beispiele für DNS-Einträge:

A – Symbolischer Name \rightarrow IP

PTR – IP \rightarrow symbolischer Name (z.B. zur Verifikation mittels "Reverse-Lookup")

Beispiele für DNS-Einträge:

A – Symbolischer Name → IP

PTR – IP → symbolischer Name (z.B. zur Verifikation mittels "Reverse-Lookup")

NS – IP des Nameservers für das Subnetz der Domain

Beispiele für DNS-Einträge:

A – Symbolischer Name → IP

PTR – IP → symbolischer Name (z.B. zur Verifikation mittels "Reverse-Lookup")

NS – IP des Nameservers für das Subnetz der Domain

CNAME – Symbolischer Name → symbolischer Name (z.B. für Domains bei Hostinganbietern, die keinen eigenen Server haben)

Beispiele für DNS-Einträge:

A – Symbolischer Name → IP

PTR – IP → symbolischer Name (z.B. zur Verifikation mittels "Reverse-Lookup")

NS – IP des Nameservers für das Subnetz der Domain

CNAME – Symbolischer Name → symbolischer Name (z.B. für Domains bei Hostinganbietern, die keinen eigenen Server haben)

MX – Symbolischer Name des Mailservers für die Domain

HTTP 1

http = "Hypertext Transfer Protocol"

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

http ist ein Anfrage–Antwort-Protokoll für den Datenaustausch zwischen Server und Client.

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

http ist ein Anfrage–Antwort-Protokoll für den Datenaustausch zwischen Server und Client.

Wichtigste http Methoden:

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

http ist ein Anfrage–Antwort-Protokoll für den Datenaustausch zwischen Server und Client.

Wichtigste http Methoden:

- GET – Anfrage zum Download von Daten

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

http ist ein Anfrage–Antwort-Protokoll für den Datenaustausch zwischen Server und Client.

Wichtigste http Methoden:

- GET – Anfrage zum Download von Daten
- POST – Anfrage zum Upload von Daten (z.B. für Skripte)

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

http ist ein Anfrage–Antwort-Protokoll für den Datenaustausch zwischen Server und Client.

Wichtigste http Methoden:

- GET – Anfrage zum Download von Daten
- POST – Anfrage zum Upload von Daten (z.B. für Skripte)
- PUT – Speichere Daten unter der Adresse

HTTP 1

http = "Hypertext Transfer Protocol"

Es gibt zwei Versionen: HTTP/1.1 und HTTP/2.

http ist auf Anwendungsebene und nutzt TCP auf Transportebene.

http ist ein Anfrage–Antwort-Protokoll für den Datenaustausch zwischen Server und Client.

Wichtigste http Methoden:

- GET – Anfrage zum Download von Daten
- POST – Anfrage zum Upload von Daten (z.B. für Skripte)
- PUT – Speichere Daten unter der Adresse
- TRACE – Gib Befehl wieder zurück (um zu vergleichen)

HTTP 2

Beispiel:

“GET /index.html HTTP/1.1” –

Beispiel:

“GET /index.html HTTP/1.1” – Schicke mir die Datei
“index.html” mittels HTTP/1.1 Standard

HTTP 2

Beispiel:

“GET /index.html HTTP/1.1” – Schicke mir die Datei
“index.html” mittels HTTP/1.1 Standard

Serverantwort:

HTTP 2

Beispiel:

“GET /index.html HTTP/1.1” – Schicke mir die Datei
“index.html” mittels HTTP/1.1 Standard

Serverantwort:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close
```

```
<html>
<head>
<title>An Example Page</title>
</head>
<body>
Hello World, this is a very simple HTML document.
</body>
</html>
```

Statuscodes (dreistellige Zahlen):

- 200 – “OK”

Statuscodes (dreistellige Zahlen):

- 200 – “OK”
- 404 – “Not Found”

Statuscodes (dreistellige Zahlen):

- 200 – “OK”
- 404 – “Not Found”
- 400 – “Bad Request”

Statuscodes (dreistellige Zahlen):

- 200 – “OK”
- 404 – “Not Found”
- 400 – “Bad Request”
- 418 – “I’m a teapot”

Statuscodes (dreistellige Zahlen):

- 200 – “OK”
- 404 – “Not Found”
- 400 – “Bad Request”
- 418 – “I’m a teapot”
- 500 – “Internal Server Error”

Statuscodes (dreistellige Zahlen):

- 200 – “OK”
- 404 – “Not Found”
- 400 – “Bad Request”
- 418 – “I’m a teapot”
- 500 – “Internal Server Error”
- 502 – “Bad Gateway” bei Proxys u.ä.

Statuscodes (dreistellige Zahlen):

- 200 – “OK”
- 404 – “Not Found”
- 400 – “Bad Request”
- 418 – “I’m a teapot”
- 500 – “Internal Server Error”
- 502 – “Bad Gateway” bei Proxys u.ä.

Referer:

XXXX

HTTP 4

User-Agent:

XXXX

HTTP 4

User-Agent:

XXXX

Keep-Alive:

XXXX

HTTP 4

User-Agent:

XXXX

Keep-Alive:

XXXX

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

`"http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg"`

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

"http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg

Reservierte Zeichen haben Bedeutung für URI, z.B. /, !, +

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

"http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg

Reservierte Zeichen haben Bedeutung für URI, z.B. /, !, +

Unreservierte Zeichen können beliebig verwendet werden.

⇒ Zusammen mit dem % Zeichen werden "alle" Zeichen kodiert.

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

“http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg”

Reservierte Zeichen haben Bedeutung für URI, z.B. /, !, +

Unreservierte Zeichen können beliebig verwendet werden.

⇒ Zusammen mit dem % Zeichen werden “alle” Zeichen kodiert.

Beispiele:

- / – %2F

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

"http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg

Reservierte Zeichen haben Bedeutung für URI, z.B. /, !, +

Unreservierte Zeichen können beliebig verwendet werden.

⇒ Zusammen mit dem % Zeichen werden "alle" Zeichen kodiert.

Beispiele:

- / – %2F

- + – %2A

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

"http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg

Reservierte Zeichen haben Bedeutung für URI, z.B. /, !, +

Unreservierte Zeichen können beliebig verwendet werden.

⇒ Zusammen mit dem % Zeichen werden "alle" Zeichen kodiert.

Beispiele:

- / – %2F

- + – %2A

- ü – %C3%BC

URL Encoding

Für viele Webanwendungen müssen Informationen in URI eingebaut werden, z.B. wie in

"http://www.w3.com/form_submit.asp?text=Das+ist+so+%2Fmeg

Reservierte Zeichen haben Bedeutung für URI, z.B. /, !, +

Unreservierte Zeichen können beliebig verwendet werden.

⇒ Zusammen mit dem % Zeichen werden "alle" Zeichen kodiert.

Beispiele:

- / – %2F

- + – %2A

- ü – %C3%BC

⇒ Man kann oft den Input für Webseiten manuell in der URL manipulieren!

HTTPS 1

Erweiterung von http um eine Verschlüsselungsebene mittels TLS (SSL ist unsicher!).

HTTPS 1

Erweiterung von http um eine Verschlüsselungsebene mittels TLS (SSL ist unsicher!).

Zwei Elemente: Authentifizierung der Website
(Verhinderung von Man-in-the-Middle Attacken)

HTTPS 1

Erweiterung von http um eine Verschlüsselungsebene mittels TLS (SSL ist unsicher!).

Zwei Elemente: Authentifizierung der Website (Verhinderung von Man-in-the-Middle Attacks) und Verschlüsselung der Kommunikation.

HTTPS 1

Erweiterung von http um eine Verschlüsselungsebene mittels TLS (SSL ist unsicher!).

Zwei Elemente: Authentifizierung der Website (Verhinderung von Man-in-the-Middle Attacks) und Verschlüsselung der Kommunikation.

URI benutzt anderes Schema: `https://hostname`

HTTPS 1

Erweiterung von http um eine Verschlüsselungsebene mittels TLS (SSL ist unsicher!).

Zwei Elemente: Authentifizierung der Website (Verhinderung von Man-in-the-Middle Attacken) und Verschlüsselung der Kommunikation.

URI benutzt anderes Schema: https://hostname und Port 443.

HTTPS 1

Erweiterung von http um eine Verschlüsselungsebene mittels TLS (SSL ist unsicher!).

Zwei Elemente: Authentifizierung der Website (Verhinderung von Man-in-the-Middle Attacks) und Verschlüsselung der Kommunikation.

URI benutzt anderes Schema: https://hostname und Port 443.

Beim Handshake wird Verschlüsselungsstandard ausgehandelt (\Rightarrow manipulierbar?!).

Typischerweise asymmetrisches Verfahren für Authentifizierung (z.B. RSA) und symmetrisches Verfahren für Verschlüsselung (z.B. DH oder ECDH).

Typischerweise asymmetrisches Verfahren für Authentifizierung (z.B. RSA) und symmetrisches Verfahren für Verschlüsselung (z.B. DH oder ECDH).

Authentifizierung läuft über *Zertifikate*, bei der eine *Certificate Authority* kryptographisch unterschreibt, dass der Besitzer des Zertifikats Eigentümer des Hosts ist.

Typischerweise asymmetrisches Verfahren für Authentifizierung (z.B. RSA) und symmetrisches Verfahren für Verschlüsselung (z.B. DH oder ECDH).

Authentifizierung läuft über *Zertifikate*, bei der eine *Certificate Authority* kryptographisch unterschreibt, dass der Besitzer des Zertifikats Eigentümer des Hosts ist.

⇒ Riesiges Problem bzw. sehr schwierig, sowohl theoretisch als auch praktisch.

Typischerweise asymmetrisches Verfahren für Authentifizierung (z.B. RSA) und symmetrisches Verfahren für Verschlüsselung (z.B. DH oder ECDH).

Authentifizierung läuft über *Zertifikate*, bei der eine *Certificate Authority* kryptographisch unterschreibt, dass der Besitzer des Zertifikats Eigentümer des Hosts ist.

⇒ Riesiges Problem bzw. sehr schwierig, sowohl theoretisch als auch praktisch.

Firmen und einzelne Staaten zwingen Anwender zur Nutzung eines "root Zertifikats", das ihnen ermöglicht MitM-Attacken auf https-Verbindungen durchzuführen und Datenverkehr zu überwachen.

SMTP, POP3, IMAP4

Emailprotokolle zum

SMTP, POP3, IMAP4

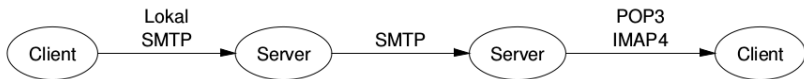
Emailprotokolle zum Abrufen von Mails, wie POP3 und IMAP4

SMTP, POP3, IMAP4

Emailprotokolle zum Abrufen von Mails, wie POP3 und IMAP4 und zum Verschicken von Mails zwischen Mailservern (SMTP)

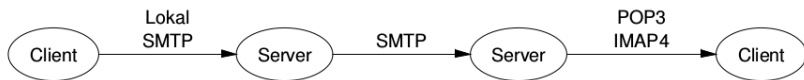
SMTP, POP3, IMAP4

Emailprotokolle zum Abrufen von Mails, wie POP3 und IMAP4 und zum Verschicken von Mails zwischen Mailservern (SMTP)



SMTP, POP3, IMAP4

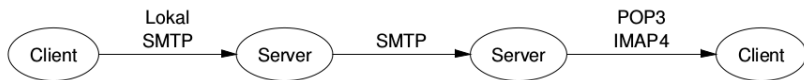
Emailprotokolle zum Abrufen von Mails, wie POP3 und IMAP4 und zum Verschicken von Mails zwischen Mailservern (SMTP)



Ports: SMTP – 25, 587; POP3 – 110, 995; IMAP – 143, 993

SMTP, POP3, IMAP4

Emailprotokolle zum Abrufen von Mails, wie POP3 und IMAP4 und zum Verschicken von Mails zwischen Mailservern (SMTP)



Ports: SMTP – 25, 587; POP3 – 110, 995; IMAP – 143, 993

Protokolle bauen Verbindungen auf, in denen Client und Server Anfragen und Antworten in Plaintext austauschen, ggf. verschlüsselt mittels TLS \Rightarrow Kommunikation kann mittels Telnet manuell übernommen werden.

Envelope-Header (FROM, RCPT \Rightarrow Das sind die Daten, die der Server beachtet. Insbesondere sind sie unabhängig von den Daten im Mail Header)

Aufbau Email

Envelope-Header (FROM, RCPT \Rightarrow Das sind die Daten, die der Server beachtet. Insbesondere sind sie unabhängig von den Daten im Mail Header)

Mail Header (inkl. From, To, Subject, Content-Type, User-Agent wie in Mail angegeben. Für eigentliche Übertragung irrelevant. Jede SMTP Weiterleitung fügt Received-Header oben an!)

Aufbau Email

Envelope-Header (FROM, RCPT \Rightarrow Das sind die Daten, die der Server beachtet. Insbesondere sind sie unabhängig von den Daten im Mail Header)

Mail Header (inkl. From, To, Subject, Content-Type, User-Agent wie in Mail angegeben. Für eigentliche Übertragung irrelevant. Jede SMTP Weiterleitung fügt Received-Header oben an!)

Data (nach Leerzeichen, beendet durch alleinigen "." auf Zeile)

Beispiel Email

Received: from imap.web.de [217.72.192.135]
by localhost with IMAP (fetchmail-6.2.5)
for iblech@localhost (single-drop);
Thu, 05 Aug 2004 18:55:15 +0200 (CEST)
Received: (nullmailer pid 9702 invoked by uid 1000);
Thu, 05 Aug 2004 16:54:02 -0000
Date: Thu, 5 Aug 2004 18:54:02 +0200
From: Ingo Blechschmidt <iblech@web.de>
To: Ingo Blechschmidt <iblech@thestars.gnus>
Subject: Test
Message-ID: <20040805165402.GA9665@thestars.gnus>
Reply-To: iblech@web.de
Content-Type: text/plain; charset=us-ascii
User-Agent: Mutt/1.5.6i
Content-Length: 190
Lines: 7

Hier da

-- (Bindestrich Bindestrich Leerzeichen)

Linux, the choice of a GNU	Mathematicians practice
generation on a dual AMD-	absolute freedom.
Athlon!	-- Henry Adams
Encrypted mails preferred.	

Beispiel SMTP Protokoll

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```


Whois

Apache und Nginx

Curl und wget

Telnet

Wireshark

Whois

Ping

Nmap

Nmap Resultate

Nmap OS Erkennung

Traceroute

Vorgehen

Wireshark mit ARP Spoofing

Aircrack

Session IDs

Cookies

Beispiel: PhPBB

SQL Injection

Iodine

DNS Tunneling