

# Eine erste Einführung in die Sicherheit von (verteilten) Systemen<sup>[1]</sup>

Dozent: Prof. Dr. Michael Eichberg  
Kontakt: michael.eichberg@dhbw.de  
Version: 1.3.1

---

Folien: [HTML] <https://delors.github.io/sec-tcp-ssh-firewalls/folien.de.rst.html>  
[PDF] <https://delors.github.io/sec-tcp-ssh-firewalls/folien.de.rst.html.pdf>  
Fehler melden: <https://github.com/Delors/delors.github.io/issues>

---

[1] Die Folien basieren unter anderem auf einem Foliensatz von Prof. Dr. Henning Pagnia.  
Alle Fehler sind meine eigenen.

# Themen

- Transmission Control Protocol (TCP)
- Einmal-Passwörter
- Secure Shell (SSH)
- Firewalls

# 1. Transmission Control Protocol (TCP)

## TCP Grundlagen




- Protokoll der Schicht 4 (Transport Layer) basiert auf IP
- verbindungsorientierte Kommunikation zweier Rechner im Internet zuverlässig und geordnet:
  - Verwerfen von Duplikaten und fehlerhaft übertragener Pakete
  - automatisches Wiederversenden fehlender Pakete
  - Nachrichtenpuffer: Daten werden in korrekter Reihenfolge an Applikation zugestellt
- Verbindungsaufbau immer zwischen zwei Sockets (Socket-Adresse: IP Adresse und 16 Bit-Port-Nummer)

# Aufbau einer TCP Verbindung

Dreifacher Handshake:

---

## Terminologie:

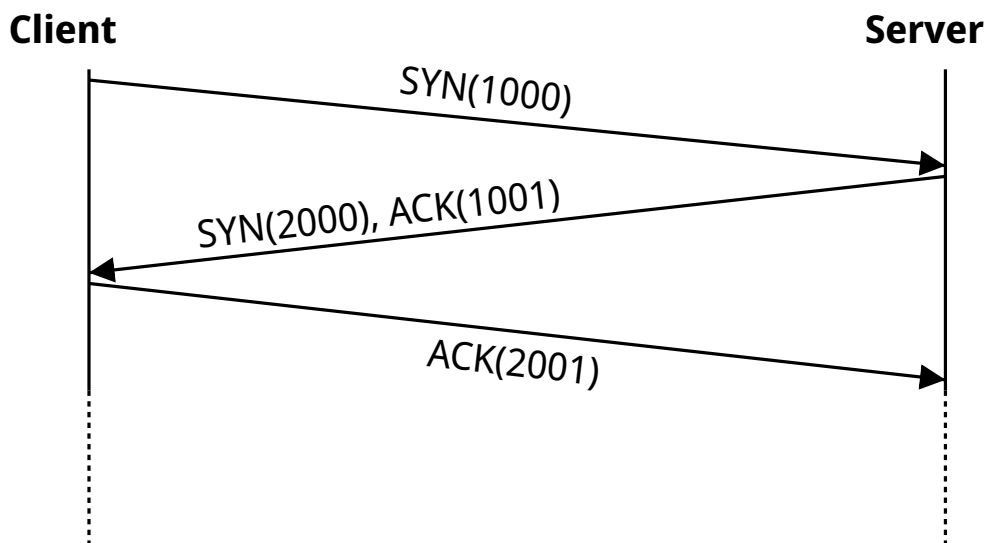
SYN:  *synchronize (session establishment)*  
ACK:  *acknowledge*  
RST:  *reset*

## Verbindungsaufbau - Ablauf:

1. Client sendet SYN Paket mit initialer Sequenznummer (hier) 1000 an den Server.
2. Server sendet ein SYN-ACK Paket mit seiner initialen Sequenznummer (hier) 2000 und ein ACK mit der Sequenznummer 1001 (initiale Sequenznummer des Clients +1) an den Client
3. Client sendet ein ACK Paket mit Sequenznummer 2001 (initiale Sequenznummer des Servers +1) an den Server; danach ist die Verbindung aufgebaut.

Das Betriebssystem sollte die initialen Sequenznummern zufällig wählen, so dass ein Angreifer diese nicht leicht vorhersagen kann. Beide Seiten haben eigene Sequenznummern, die unabhängig voneinander sind.

Bei einer laufenden Verbindung werden die Sequenznummern inkrementiert und es ist nicht (mehr) erkennbar wer die Verbindung aufgebaut hat.



## Ports bei TCP

- Port-Nummern werden für die Kommunikation zwischen zwei Diensten/Prozessen verwendet
- Ports sind 16 Bit Zahlen (0-65535)
- (Unix) Ports < 1024 sind privilegiert (nur root kann diese öffnen)
- einige Port-Nummern sind Standarddiensten zugeordnet

## Port-Nummern einiger Standarddienste [2]

**Ungeschützte Dienste** (Kommunikation findet ohne Verschlüsselung statt.)

Protokoll	Dienst	Portnummer
ftp	Dateitransfer	21
smtp	Simple Mail Transfer Protocol	25
dns	Domain Name System	53
http	Hypertext Transfer Protocol	80
login	Login auf entfernte Rechner	513

**Geschützte Dienste** (Die Kommunikation ist verschlüsselt.)

Protokoll	Dienst	Portnummer
ssh	Secure Shell	22
https	HTTP über Secure Socket Layer	443
smtps	SMTP über Secure Socket Layer	465
imaps	IMAP über Secure Socket Layer	993
pop3s	POP3 über Secure Socket Layer	995

[2] Port numbers assigned by IANA

# Angriffe auf TCP - Motivation

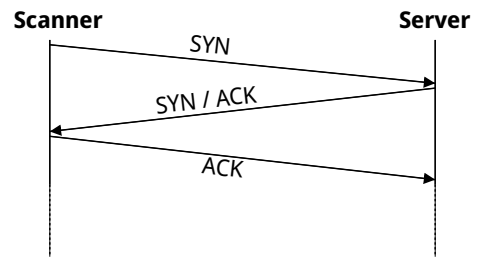
- Netzwerkprogrammierung mit TCP ist relativ komfortabel.
- Viele Dienste sind mit TCP implementiert.  
Insbesondere in der Anfangszeit hatten viele TCP Dienste sowohl technische als auch konzeptionelle Schwachstellen. Einige dieser Schwachstellen sind bis heute nicht behoben.
- Das Auffinden von angreifbaren Diensten kann mit Hilfe von Port Scans systematisch erfolgen.  
Server haben heutzutage im Allgemeinen alle nicht verwendeten Dienste geschlossen.



# Port Scans: TCP Connect Scan

## Vorgehen

Aufbau vollständiger Verbindungen zu allen bzw. zu ausgewählten Ports.



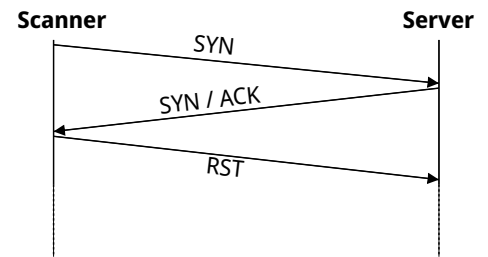
## Bewertung

- simpelster Port Scan
- große Entdeckungsgefahr (Scan selbst ist kein Angriff)
- mögliche Verbesserung: zwischen dem Scannen mehrerer Ports Pausen einstreuen (Wie lange?)

# Port Scans: TCP SYN Scan

## Vorgehen

1. Senden eines TCP-Segments mit gesetztem SYN-Flag an einen Port
2. falls der *Port offen* ist, kommt SYN/ACK zurück danach RST senden
3. falls der *Port nicht offen* ist, kommt RST (oder nichts) zurück



## Bewertung

- kein vollständiger Verbindungsaufbau
- meist nicht protokolliert
- geringe(re) Entdeckungsgefahr

# Port Scans: Stealth Scans

- Vorgehen:** Versenden eines für den Verbindungsaufbau ungültigen TCP-Segments an einen Port:
- NULL-Scan (keine Flags)
  - ACK-Scan (ACK-Flag)
  - FIN-Scan (FIN-Flag)
  - XMAS-Scan (alle Flags)

Laut RFC kommt RST zurück, falls der Port offen ist. (Reaktion ist de-facto aber abhängig vom Betriebssystem und oft kommt keine Antwort zurück.)

## Bewertung

- Zugriff wird meist nicht protokolliert
- Scan bleibt unbemerkt

---

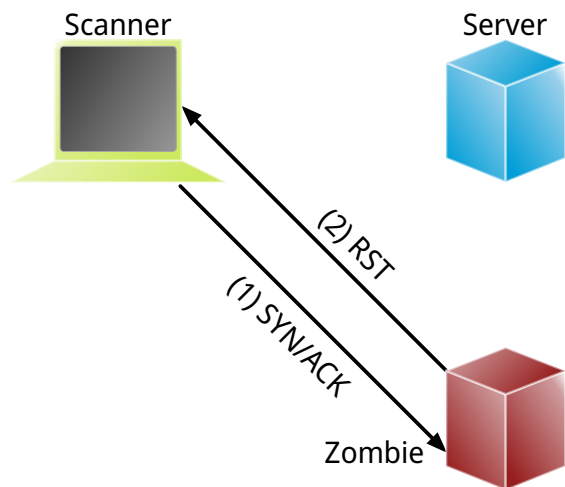
### XMAS-Scan:

Bei diesem Scan sind alle Flags gesetzt; ein XMAS-Scan wird auch als Christmas-Tree-Scan bezeichnet, da das Paket erleuchtet ist wie ein Weihnachtsbaum.

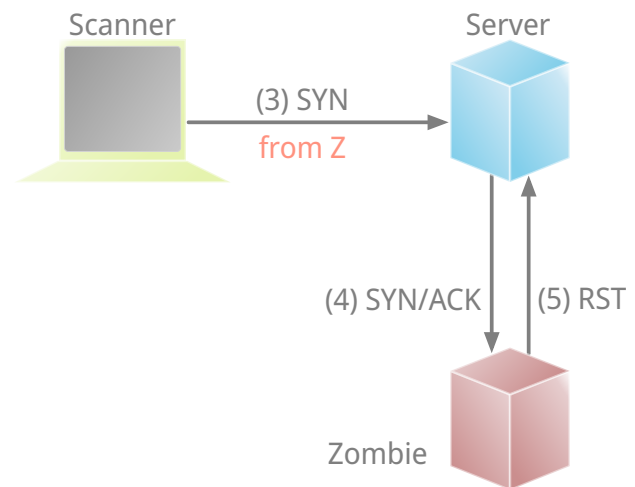
## Port Scans: Idle Scan [3]

Bei allen bisher betrachteten Scans kann der Scanner prinzipiell identifiziert werden. Unter Verwendung eines sog. Zombies geht es auch anders:

Sondiere IP ID des Zombies:



Starte Scan:



**Zombie:** ein Rechner (Computer, Drucker oder anderes IoT Gerät) im Internet *möglichst ohne eigenen Netzverkehr* und mit *altem* Betriebssystem, bei dem die IP ID in vorhersehbarer Weise inkrementiert wird. (Bei modernen Betriebssystemen ist die IP ID zufällig, **konstant** oder sogar `null`.)

**Grundlegende Idee:**

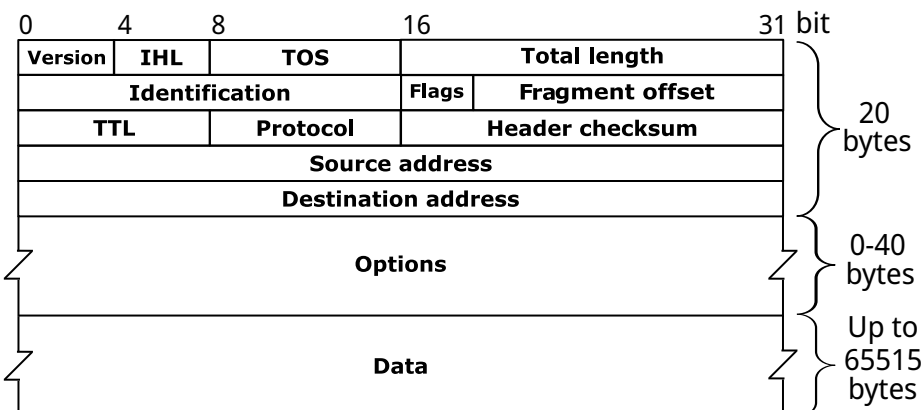
Der Zombie sendet ein RST Paket zurück, da er kein SYN gesendet hat und kein SYN/ACK erwartet. Dadurch erfährt der Angreifer die aktuelle IP ID des Zombies. Über diesen Seitenkanal - d. h. die Veränderung der IP ID des Zombies - kann der Angreifer nun den Zustand des Ports auf dem Zielrechner ermitteln.

### Hinweis

Sollte ein Intrusion Detection System vorhanden sein, so wird dieses den Zombie als Angreifer identifizieren.

## Hintergrund - IP ID

Das Feld *IP Identifikation (IP ID)* dient der Identifizierung einer Gruppe von Fragmenten eines einzelnen IP-Datagramms.

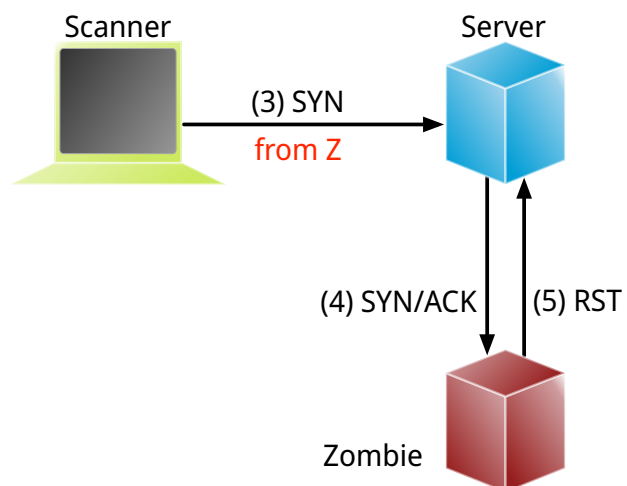


By Michel Bakni - Postel, J. (September 1981) RFC 791, IP Protocol, DARPA Internet Program Protocol Specification, p. 1 DOI: 10.17487/RFC0791., CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=79949694>

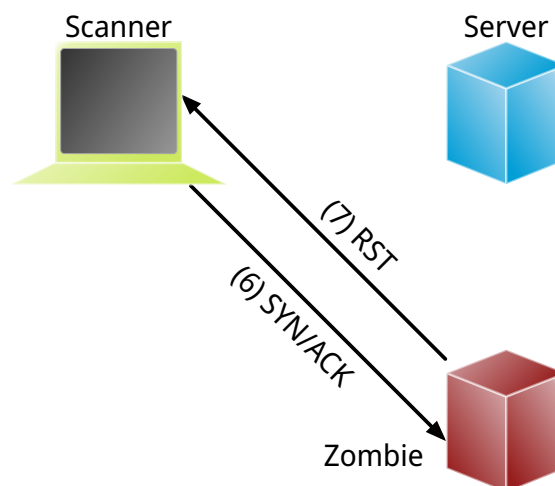


# Port Scans: Idle Scan

Starte Scan:



Sondiere IP ID des Zombies:



# Port Scans: Idle Scan - Zusammenfassung

- Angreifer sendet SYN/ACK Paket an Zombie
- der Zombie antwortet mit RST und enthüllt seine IP ID (🚩 *IP Fragment Identification Number*).
- Angreifer sendet SYN („mit IP vom Zombie“) an Port des Servers:

**[Port offen]** Der Zielrechner antwortet mit SYN/ACK an den Zombie, wenn der Port offen ist. Der Zombie antwortet darauf mit RST an den Server, da er kein SYN gesendet hat und kein SYN/ACK erwartet und *erhöht seine IP ID*.

**[Port geschlossen]** Der Zielrechner antwortet mit RST an den Zombie, wenn der Port geschlossen ist. Dies wird vom Zombie ignoriert.

- Der Angreifer sendet wieder ein SYN/ACK an den Zombie, um die IP ID zu erfahren.

---

Mit einem IDLE Scan kann nicht unterschieden werden, ob der Port geschlossen oder gefiltert ist.

# Port Scans mit nmap

- alle Arten von Port-Scans möglich
- auch OS fingerprinting
- u. U. sogar Ermittlung der Versionsnummern von Diensten

```
$ nmap 192.168.178.121 -Pn
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-14 13:16 PST
Nmap scan report for Michaels-MacBook-Pro (192.168.178.121)
Host is up (0.0056s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
445/tcp   open  microsoft-ds
5000/tcp  open  upnp
7000/tcp  open  afs3-fileserver
```

---

## OS-Fingerprinting

Beim OS-Fingerprinting werden Datenpakete analysiert, die aus einem Netzwerk stammen, um Informationen für spätere Angriffe zu gewinnen. Durch die Erkennung des Betriebssystems, mit dem ein Netzwerk arbeitet, haben Hacker es leichter, Schwachstellen zu finden und auszunutzen. OS-Fingerprinting kann auch Konfigurationsattribute von entfernten Geräten sammeln. Diese Art von Aufklärungsangriff ist in der Regel (einer) der erste(n) Schritt(e).

Es gibt zwei Arten von OS-Fingerprinting: (1) Aktiv und (2) passiv.

1. Bei einem aktiven OS-Fingerprinting-Versuch senden die Angreifer ein Paket an das Zielsystem und warten auf eine Antwort, um den Inhalt des TCP-Pakets zu analysieren.
2. Bei einem passiven Versuch agieren die Angreifer eher als "Schnüffler", der keine absichtlichen Änderungen oder Aktionen im Netzwerk vornimmt. Passives OS-Fingerprinting ist ein unauffälligerer, aber wesentlich langsamerer Prozess.



# Port Knocking

- Ein Knock-Daemon versteckt offene Ports auf dem Server.
- Zugriffe auf alle Ports werden im Log-File protokolliert.
- Knock-Daemon beobachtet das Log-File.
- Erst nach Erkennen einer vordefinierten (Einmal-)Klopfssequenz öffnet der Knock-Daemon den gewünschten Port für diesen Client.
- Client kann nun die Verbindung aufbauen.
- Weiterentwicklung: TCP Stealth
  - In diesem Fall werden offene Ports dadurch versteckt, dass sie nur auf spezielle SYN-Pakete mit bestimmten Sequenznummern reagieren. Die Sequenznummern sind ggf. kryptografisch abgesichert und basieren auf vorher ausgetauschten Schlüsseln.

---

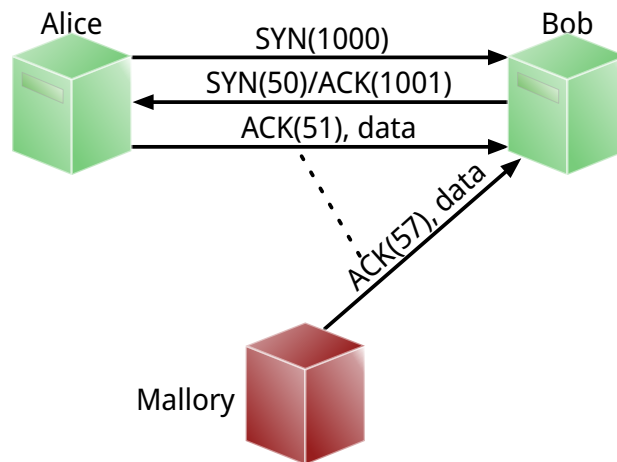
## Weiterführend

Alternativen zu einer Knock-Sequenz ist zum Beispiel, dass der Port nur dann als offen gilt, wenn die IP ID eine bestimmte Sequenznummer aufweist.

M.Krzywinski: Port Knocking: Network Authentication Across Closed Ports in SysAdmin Magazine 12: 12-17. (2003)

# Connection Hijacking

Angreifer übernimmt eine bestehende - zum Beispiel eine bereits durch (Einmal-)Passwort authentifizierte - Verbindung.



---

TCP/IP-Hijacking ist eine Form eines Man-in-the-Middle-Angriffs. Der Angreifer bestimmt erst die IP-Adressen der beiden Sitzungsteilnehmer.

Danach gibt es mehrere Möglichkeiten:

- Der Angreifer schickt ("in einer Pause") ein Paket mit der passenden Sequenznummer an den Server.  
*(Dies kann dann in einem ACK-Storm enden, was ggf. unterbunden werden muss (zum Beispiel durch das Senden eines RSTs), oder ignoriert werden kann.)*
- Der Angreifer macht einen Client mit einem DoS-Angriff unerreichbar, um sich dann mit dem Anderen zu verbinden, indem er die Netzwerk-ID des ausgeschalteten Clients nutzt.

# Denial-of-Service (DoS) Angriffe

Ziel des Angreifers: Lahmlegen eines Dienstes oder des ganzen Systems ...

- durch Ausnutzen von Schwachstellen ( *vulnerabilities*) wie z. B. Buffer Overflows
- durch Generierung von Überlast (Ausschöpfen von RAM, CPU, Netzwerkbandbreite, ...)

## Beispiel

### Ping-of-Death

(Historisch: aus dem Jahr 1997)

Ein `ping` (vgl. Internet Control Message Protocol (ICMP)) verwendet üblicherweise kleine Nachrichten, aber die verwendete Länge ist einstellbar.

Falls die Länge zu groß ist  $\Rightarrow$  Buffer Overflow  $\Rightarrow$  Systemabsturz!

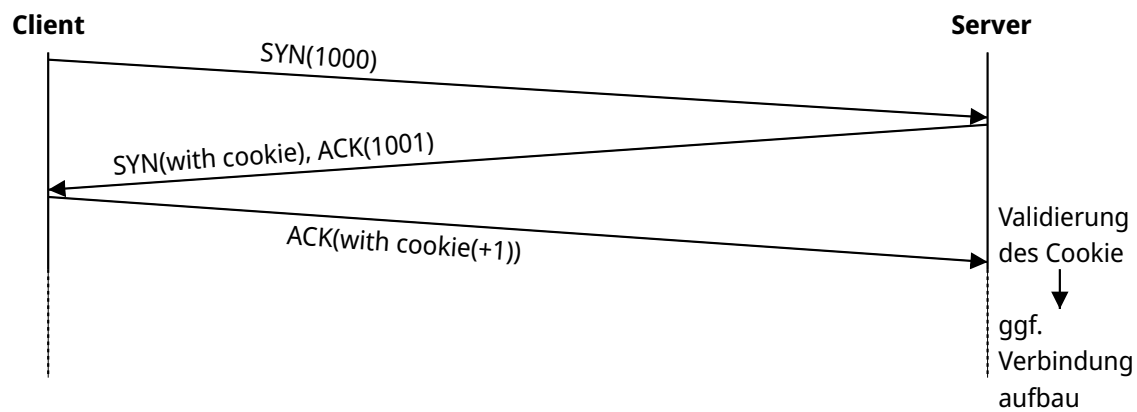
Variante: mittels Fragmentierung ließen sich generell übergroße IP-Pakete ( $>65,536$  Byte) erstellen.

# Denial-of-Service: SYN-flooding Angriff

- Angriff auf Design
- Angreifer sendet eine Verbindungsaufbauanforderung (gesetztes SYN-Flag) an Zielmaschine
- Server generiert eine halboffene TCP-Verbindung
- Angreifer wiederholt in schneller Folge dieses erste Paket zum Verbindungsaufbau
  - ⇒ vollständiges Füllen der internen Systemtabelle
  - ⇒ Anfragen normaler Benutzer werden zurückgewiesen
- Angreifer verwendet i. Allg. IP-Spoofing weswegen Firewalls wirkungslos sind.
- Abwehr: SYN-Cookies

## SYN-Cookies - D.J. Bernstein

SYN-Cookies sind speziell konstruiert initiale Sequenznummern.

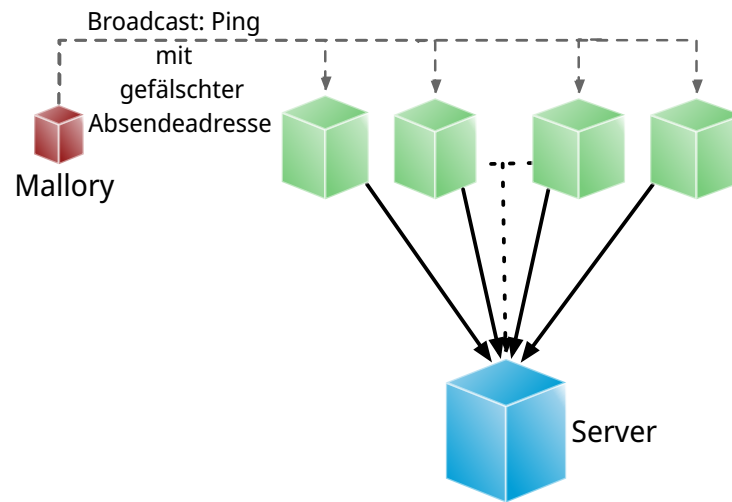


Der Cookie ermöglicht es, dass keine Informationen im Speicher gehalten werden müssen. Der Cookie encodiert die Informationen, die der Server benötigt, um die Verbindung aufzubauen: Client IP, time window, etc.

# Distributed Denial-of-Service (DDoS) Angriff

Opfer wird von sehr vielen Angreifern mit Nachrichten überflutet.

Ein Beispiel: Smurf-Angriff:



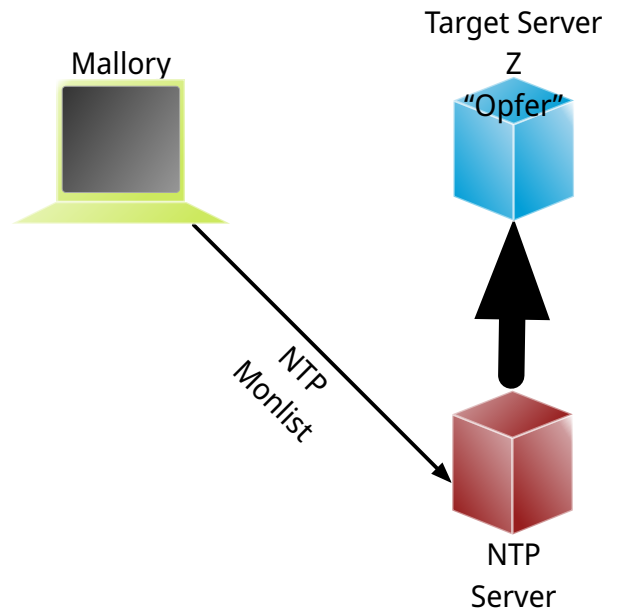
# Distributed Denial-of-Service (DDoS) Angriff

- Bot-Netze (Botnetze) werden verwendet, um DDoS-Angriffe durchzuführen.
- Bot-Netze können viele 10.000 Rechner umfassen.
- IoT Geräte sind besonders beliebt (z. B. IP-Kameras, Smart-TVs, Smart-Home Geräte, ...), da diese oft nicht ausreichend geschützt sind und trotzdem permanent mit dem Internet verbunden sind.
- Beliebte Ziele:
  - Onlinespieleserver
  - Banking-Portale
  - politische Webseiten
- Firewalls und Intrusion Detection Systeme sind meist wirkungslos, da die Angriffe von vielen verschiedenen IP-Adressen kommen.

# Distributed-Reflected-Denial-of-Service Angriff

## Idee eines (DRDoS) Angriffs

- Es wird eine Anfrage an einen Server gesendet, die eine große Antwort auslöst.  
-----  
Z. B. hat(te) der NTP Monlist Befehl eine Antwort, die ca. 200 Fach größer ist als die Anfrage!
- Mittels IP-Spoofing wird die IP-Adresse des Opfers als Absenderadresse verwendet.
- Es werden insbesondere Dienste basierend auf UDP verwendet, da hier keine Verbindung aufgebaut werden muss.



- Nehmen einen signifikanten Teil aller DDoS-Angriffe ein.
- Die Tatsache, dass die Sender legitime Server sind, erschwert die Abwehr.
- Egress Filtering kann helfen, die Verwendung von IP-Spoofing zu verhindern.

-----  
Bereits im Jahr 2018 wurde ein Angriff mit einer Bandbreite von 1,7 TBit/s beobachtet.

**Egress Filtering:** Der Router verwirft alle Pakete, die eine Absenderadresse verwenden, die nicht aus dem eigenen Netzwerk stammt.



## Distributed Denial-of-Service (DDoS) Angriffe - Beispiel

*[...] Google's DDoS Response Team has observed the trend that distributed denial-of-service (DDoS) attacks are **increasing exponentially in size**. Last year, we blocked the largest DDoS attack recorded at the time. This August [2023], we stopped an even larger DDoS attack — 7½ times larger — that also used new techniques to try to disrupt websites and Internet services.*

*This new series of DDoS attacks reached **a peak of 398 million requests per second (rps)**, and relied on a novel HTTP/2 “Rapid Reset” technique based on stream multiplexing that has affected multiple Internet infrastructure companies. By contrast, last year's largest-recorded DDoS attack peaked at 46 million rps.*

# Distributed Denial-of-Service Angriffe - Beispiele

- **TCP Stack Attacks** SYN, FIN, RST, ACK, SYN-ACK, URG-PUSH, other combinations of TCP Flags, slow TCP attacks
- **Application Attacks:** HTTP GET/POST Floods, slow HTTP Attacks, SIP Invite Floods, DNS Attacks, HTTPS Protocol Attacks
- **SSL/TLS Attacks:** Malformed SSL Floods, SSL Renegotiation, SSL Session Floods
- **DNS Cache Poisoning**
- **Reflection Amplification Flood Attacks:** TCP, UDP, ICMP, DNS, mDNS, SSDP, NTP, NetBIOS, RIPv1, rcpbind, SNMP, SQL RS, Chargen, L2TP, Microsoft SQL Resolution Service
- **Fragmentation Attacks:** Teardrop, Targa3, Jolt2, Nestea
- **Vulnerability Attacks**
- **Resource Exhaustion Attacks:** Slowloris, Pyloris, LOIC, etc.
- **Flash Crowd Protection**
- **Attacks on Gaming Protocols**

# Schutz vor DDoS-Angriffen: On-Site Maßnahmen

- Aufrüsten der Ressourcen (z. B. Bandbreite, CPU, RAM, ...)
- Exemplarische Sofortmaßnahmen bei aktivem Angriff:
  - Whitelisting von IP-Adressen von besonders wichtigen Clients
  - Blacklisting von IP-Adressen aus bestimmten Bereichen
  - Captchas
  - Überprüfung der Browser-Echtheit
- Anti-DDos Appliances

## Achtung!

Diese Maßnahmen sind häufig teuer und ggf. begrenzt effektiv; wenn der Angriff die verfügbare Bandbreite übersteigt, sind diese Maßnahmen darüber hinaus wirkungslos.

# Schutz vor DDoS-Angriffen: Off-Site Maßnahmen

- Einbinden des ISP
- Einbinden spezialisierter Dienstleister  
(Im Angriffsfall wird mittels BGP-Rerouting der Traffic an den Dienstleister umgeleitet, der dann die DDos Attacke filtert.)
- Content-Delivery-Networks (CDNs) für statische Inhalte (z. B. Cloudflare, Akamai, ...)
- Distributed Clouds

# Password Sniffing

**In der Anfangszeit:** unverschlüsselte Übertragung von Passwörtern (telnet, ftp, ...)

**In der Übergangszeit:**

Verwendung von Einmal-Passwörtern (S/Key, ...)

**Heute:**

Passwörter werden verschlüsselt übertragen (ssh, https, ...)

Zusätzliche Absicherung durch Zwei-Faktor-Authentifizierung (basierend auf Einmalpassworten: TOTP, ...)

---

Unverschlüsselte Passwörter können leicht mittels eines Sniffers, der den Netzwerkverkehr mitschneidet (z. B. Wireshark), abgefangen werden.

# Einmal-Passwörter

Die Idee ist, dass Passwörter nur genau einmal gültig sind und nicht wiederverwendbar sind.

- Tokens (z. B. RSA SecurID)
- Codebuch: Liste von Einmal-Passwörtern, die das gemeinsame Geheimnis sind.
- S/Key: Passwort „wird mit einem Zähler kombiniert“ und dann gehasht.

# Das S/Key Verfahren

Einmal-Passwort-System nach Codebuch-Verfahren.

## Initialisierung

1. Der Nutzer gibt sein Passwort  $W$  ein; dies ist der geheime Schlüssel.  
(Sollte  $W$  bekannt werden, dann ist die Sicherheit des Verfahrens nicht mehr gewährleistet.)
2. Eine kryptografische Hash-Funktion  $H$  wird  $n$ -mal auf  $W$  angewandt, wodurch eine Hash-Kette von  $n$  einmaligen Passwörtern entsteht.  $H(W), H(H(W)), \dots, H^n(W)$
3. Das initiale Passwort wird verworfen.
4. Der Benutzer erhält die  $n$  Passwörter, die in umgekehrter Reihenfolge ausgedruckt werden:  
 $H^n(W), H^{n-1}(W), \dots, H(H(W)), H(W)$ .
5. Nur das Passwort  $H^n(W)$ , das an erster Stelle der Liste des Benutzers steht, der Wert von  $n$  und ggf. ein Salt, wird auf dem Server gespeichert.

## Anmeldung

Identifiziere das letzte verwendete Passwort  $n$ .

- Der Server fragt den Nutzer nach dem Passwort  $n - 1$  (d. h.  $H^{n-1}(W)$ ) und übermittelt ggf. auch den Salt.
- Der Server hasht das Passwort und vergleicht es dann mit dem gespeicherten Passwort  $H^n(W)$ .
- Ist das Passwort korrekt, dann wird der Nutzer angemeldet und der Server speichert das Passwort  $H^{n-1}(W)$  als neues Passwort  $H^n(W)$  und dekrementiert  $n$ .

---

Im Original basiert S/Key auf der kryptographischen Hashfunktion MD4. Ein Austausch wäre aber selbstverständlich möglich!

Intern verwendet S/KEY 64-bit Zahlen. Für die Benutzbarkeit werden diese Zahlen auf sechs kurze Wörter, von ein bis vier Zeichen, aus einem öffentlich zugänglichen 2048-Wörter-Wörterbuch ( $2048 = 2^{11}$ ) abgebildet. Zum Beispiel wird eine 64-Bit-Zahl auf "ROY HURT SKI FAIL GRIM KNEE" abgebildet.

## HMAC-based one-time password (HOTP)[4]

- ermöglicht die Erzeugung von Einmal-Passwörtern auf Basis eines geheimen Schlüssels und eines Zählers; Parameter:
  - Ein kryptografisches Hash-Verfahren  $H$  (Standard ist SHA-1)
  - einen geheimen Schlüssel  $K$ , der eine beliebige Bytefolge ist
  - Ein Zähler  $C$ , der die Anzahl der Iterationen zählt
  - Länge des Passworts:  $d$  (6-10, Standardwert ist 6, empfohlen werden 6-8)
- Zur Authentifizierung berechnen beide das Einmalpasswort (HOTP) und dann vergleicht der Server den Wert mit dem vom Client übermittelten Wert:

Berechnung aus dem Schlüssel  $K$  und dem Zähler  $C$ :

$$HOTP(K, C) = truncate(HMAC_H(K, C))$$

$$truncate(MAC) = extract31(MAC, MAC[(19 \times 8 + 4) : (19 \times 8 + 7)])$$

$$HOTP\ value = HOTP(K, C) \bmod 10^d \quad (\text{führende Nullen werden nicht abgeschnitten})$$

---

*truncate* verwendet die 4 niederwertigsten Bits des MAC als Byte-Offset  $i$  in den MAC. Der Wert 19 kommt daher, dass ein SHA-1 160 Bit hat und  $160/8 = 20$  Byte.

*extract31* extrahiert 31 Bit aus dem MAC. Das höchstwertig Bit wird (wenn es nicht 0 ist) entsprechend maskiert. Eine Schwäche des Algorithmus ist, dass beide Seiten den Zähler erhöhen müssen und, falls die Zähler aus dem Tritt geraten, ggf. eine Resynchronisation notwendig ist.

---

[4] <https://www.rfc-editor.org/rfc/rfc4226>



## Time-based one-time password (TOTP)[5]

- Erzeugung von zeitlich limitierten Einmal-Passwörtern (z. B. 30 Sekunden)
- Basierend auf einem vorher ausgetauschten geheimen Schlüssel und der aktuellen Zeit  
Z. B. Unix-Zeit in Sekunden (ganzzahlig) und danach gerundet auf 30 Sekunden.
- Es wird das HOTP Verfahren mit der Zeit als Zähler verwendet und entweder SHA-256 oder SHA-512 als Hashverfahren, d. h.  $\text{TOTP value}(K) = \text{HOTP value}(K, C_T)$ , wobei  $T$  die „aktuelle Zeit“ ist.

$$C_T = \lfloor \frac{T - T_0}{T_X} \rfloor$$

$T_X$  ist die Länge eines Zeitintervalls (z. B. 30 Sekunden)

$T$  ist die aktuelle Zeit in Sekunden seit einer bestimmten Epoche

$T_0$  ist bei Verwendung der Unix-Zeit 0

$C_T$  ist somit die Anzahl der Dauern  $T_X$  zwischen  $T_0$  und  $T$

---

Das Verfahren verlangt somit, dass die Uhren von Server und Client (hinreichend) synchronisiert sind.

---

[5] <https://www.rfc-editor.org/rfc/rfc6238>

# Secure Shell (SSH)

## Verschlüsselte Verbindung

SSH ermöglicht die sichere Fernanmeldung von einem Computer bei einem anderen (typischerweise über TCP über Port 22). Es bietet mehrere Optionen für eine starke Authentifizierung und schützt die Sicherheit und Integrität der Kommunikation durch starke Verschlüsselung

## Ablauf

1. Authentisierung des Server-Rechners
2. Authentisierung des Benutzers (bzw. des Clients) mittels
  - a. Passwort
  - b. ~~~hosts-~~Eintrag
  - c. privatem (RSA-)Key (hauptsächlich verwendete Methode)
3. Kommunikation über symmetrisch verschlüsselte Verbindung

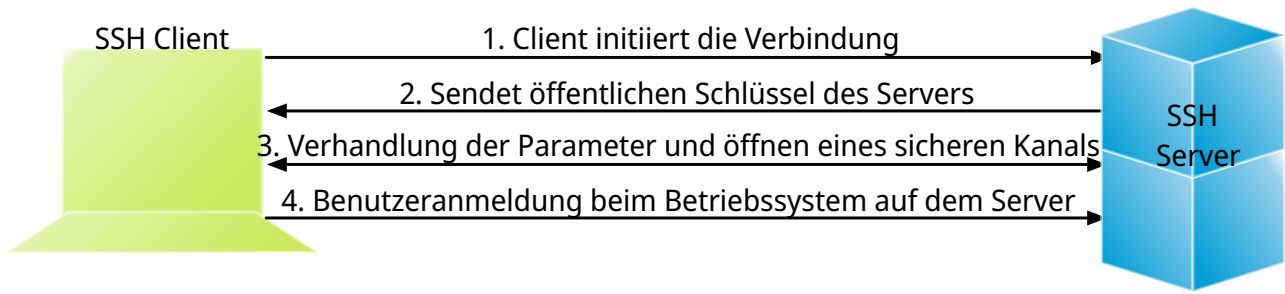
---

Die Authentifizierung mittels eines Schlüsselpaars dient primär der Automatisierung (dann wird auch keine „Schlüsselphrase“ zum Schutz des Passworts verwendet). Auf jeden Fall ist effektives Schlüsselmanagement erforderlich:

*[...] In einigen Fällen haben wir mehrere Millionen SSH-Schlüssel gefunden, die den Zugang zu Produktionsservern in Kundenumgebungen autorisieren, wobei 90 % der Schlüssel tatsächlich ungenutzt sind und für einen Zugang stehen, der zwar bereitgestellt, aber nie gekündigt wurde.*

—SSH.com (Dez. 2023)

# Secure Shell (SSH) - Protokoll



Beide Seiten haben einen Public-private Key Schlüsselpaar zur gegenseitigen Authentifizierung

**User Keys:**

- `Authorized keys` - Serverseitige Datei mit den öffentlichen Schlüsseln der Nutzer
- `Identity keys` - private Schlüssel der Nutzer

**Host keys:** dienen der Authentifizierung von Servern (verhindern Man-in-the-Middle-Angriffe)

**Session Keys:** werden für die symmetrische Verschlüsselung der Daten in einer Verbindung verwendet. Session Keys (🚩 *Sitzungsschlüssel*) werden während des Verbindungsaufbaus ausgehandelt.

---

Im Falle von SSH gibt es kein initiales Vertrauen zwischen Server und Client.

# Secure Shell (SSH) - Verbindungsaufbau - Beispiel

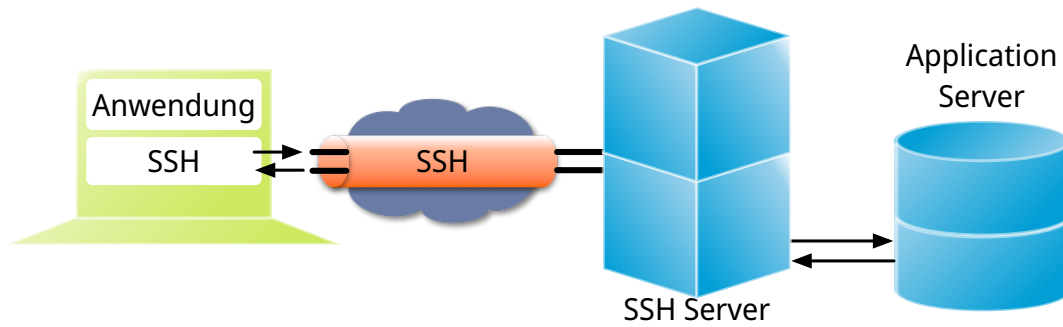
```
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to example.org [1.2.3.4] port 22.
debug1: Connection established.
debug1: identity file /home/user/.ssh/id_rsa type -1
debug1: identity file /home/user/.ssh/id_rsa-cert type -1
debug1: identity file /home/user/.ssh/id_dsa type -1
debug1: identity file /home/user/.ssh/id_dsa-cert type -1
debug1: Remote protocol version 1.99, remote software version OpenSSH_5.8
debug1: match: OpenSSH_5.8 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.5p1 Debian-6
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host 'example.org' is known and matches the RSA host key.
debug1: Found key in /home/user/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: Roaming not allowed by server
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password,keyboard-interactive,hostbased
debug1: Next authentication method: publickey
debug1: Trying private key: /home/user/.ssh/id_rsa
debug1: Trying private key: /home/user/.ssh/id_dsa
debug1: Next authentication method: keyboard-interactive
debug1: Authentications that can continue: publickey,password,keyboard-interactive,hostbased
debug1: Next authentication method: password
user@example.org's password:
debug1: Authentication succeeded (password).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = en_US.UTF-8
```

## Secure Shell (SSH) - Risiken durch mangelnde Schlüssilverwaltung

- Schlüssel werden nicht regelmäßig ausgetauscht
- Schlüssel werden nicht gelöscht, wenn sie nicht mehr benötigt werden
- viele (die meisten) Schlüssel werden nicht verwendet
- Es ist oft nicht bekannt, wer Zugriff auf welche Schlüssel hat(te)
- Es ist nicht bekannt, welche Schlüssel auf welche Systeme Zugriff haben
- Malware kann SSH-Schlüssel stehlen
- SSH Keys können ggf. privilegierten Zugriff gewähren
- SSH Keys können benutzt werden, wenn um Backdoors zu verstecken
- Server keys erlauben ggf. Man-in-the-Middle-Angriffe

# SSH Tunneling

- ermöglicht die Übertragung beliebiger Netzwerkdaten über eine verschlüsselte SSH-Verbindung. z. B.
  - um ältere Anwendungen zu verschlüsseln.
  - um VPNs (Virtual Private Networks) zu implementieren.
  - um über Firewalls hinweg auf Intranetdienste zuzugreifen.
- ermöglicht auch Port-forwarding  
(Lokale Ports werden auf entfernten Rechner weitergeleitet.)



## SSH und „Back-tunneling“

- Der Angreifer richtet einen Server außerhalb des Zielnetzwerks ein
- Nach Infiltration des Zielsystems verbindet der Angreifer sich von innen mit dem externen SSH-Server.
- Diese SSH-Verbindung wird so eingerichtet, dass eine TCP-Port-Weiterleitung von einem Port auf dem externen Server zu einem SSH-Port auf einem Server im internen Netzwerk möglich ist.
- Die meisten Firewalls bieten wenig bis gar keinen Schutz dagegen.

---

Es ist in diesem Fall besonders interessant für den Angreifer den SSH Server zum Beispiel bei einem Cloud-Anbieter zu betreiben, welcher von dem Unternehmen standardmäßig verwendet wird (am Anfang steht immer die Aufklärung!). In diesem Fall wird die Firewall keine ausgehenden SSH-Verbindungen dorthin blockieren.

## ***Nearly 11 million SSH servers vulnerable to new Terrapin attacks***

*[...] It [The Terrapin attack] manipulates sequence numbers during the handshake process to compromise the integrity of the SSH channel, particularly when specific encryption modes like ChaCha20-Poly1305 or CBC with Encrypt-then-MAC are used. [...]*

*By Bill Toulas*

*—January 3, 2024 10:06 AM*



# Übung

## 1.1. Port Scans - IDLE Scan

- Warum kann bei einem IDLE Scan nicht festgestellt werden weshalb ein Port geschlossen oder gefiltert ist?
- Welchen Wert hat die IP ID des Zombies, der einem IDLE Scan durchführt, wenn der Zielport offen bzw. geschlossen ist, wenn der Scanner diesen wieder abfragt?

# Übung

## 1.2. S/Key

1. Welche Vorteile bieten Einmalpasswortsysteme gegenüber Systemen mit mehrfach zu verwendenden Passwörtern?
2. Welchen Angriffen sind Einmalpasswortsysteme weiterhin ausgesetzt?
3. Generieren Sie eine Liste von Einmalpasswörtern mit Initialwert  $r = 769$ . Generieren Sie  $H(r)$  bis  $H^6(r)$  wenn die Einwegfunktion hier der Einfachheit halber  $H(x) = x^2 \bmod 1000$  ist.
4. Wie oft kann sich der Benutzer anmelden? Wie sieht seine Liste aus?
5. Welchen Wert speichert der Server vor dem ersten Anmeldevorgang?
6. Spielen Sie zwei Anmeldevorgänge durch.
7. Wenn ein Passwort  $H^L(W)$ ,  $1 < L < N$  bekannt ist, welche Auswirkungen hat dies auf die Sicherheit des Verfahrens?

# Übung

## 1.3. HOTP

Gegeben sei der folgende MAC:

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Mac	bc	9f	aa	ae	1e	35	d5	2f	3d	ea	96	51	da	12	cd	36	62	7b	84	03

Berechnen Sie den HOTP Wert für  $d = 6$ .

# Übung

## 14. TOTP

Identifizieren Sie die Vor- und Nachteile von TOTP gegenüber S/Key und fragen Sie sich an welcher Stelle es (aus Sicherheitsperspektive) mögliche Schwächen gibt?

Die Standardzeitspanne ist 30 Sekunden. Welcher Konsequenzen hätte eine deutliche Verlängerung bzw. Verkürzung der Zeitspanne?

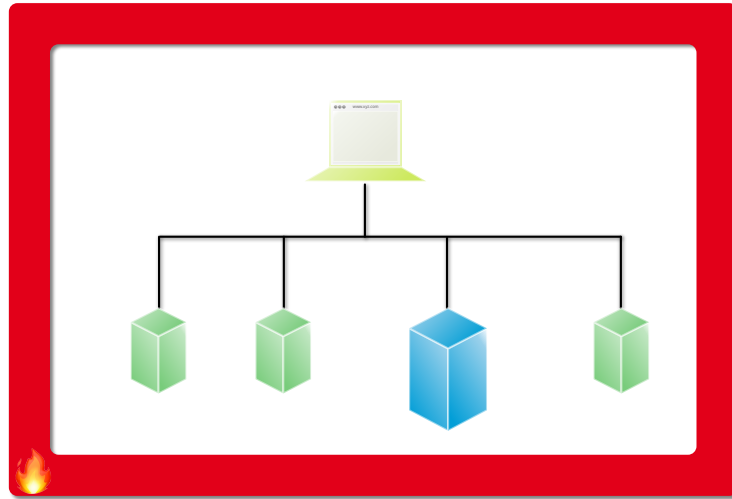
# Übung

## 1.5. DDoS

1. Welches Problem entsteht wenn zum Schutze vor Angriffen auf die Verfügbarkeit die Ressourcen von IT-Systemen und deren Internet-Anbindung erhöht werden?
2. Recherchieren Sie was ein „Low and Slow Angriff“ ist.
3. Wo kann überall „Egress filtering“ statt finden.

## 2. Firewalls

## Unabhängiges Netz - „Ideale Situation“



- Vorteile:**
- keinerlei Angriffsmöglichkeiten von außen
- Nachteile:**
- kein Schutz gegen Insider
  - kein Zugang zum Internet

---

Wie bereits diskutiert gibt es auch Angriffsmuster gegen Air-Gapped-Systeme. Ein Beispiel ist der Stuxnet-Wurm, der sich initial über USB-Sticks verbreitet.

Wenn man kein Zugang zum Internet hat, dann hat man zum Beispiel kein Zugriff auf externe Dienste wie NTP und das Einspielen von Updates ist nur über Umwege möglich.

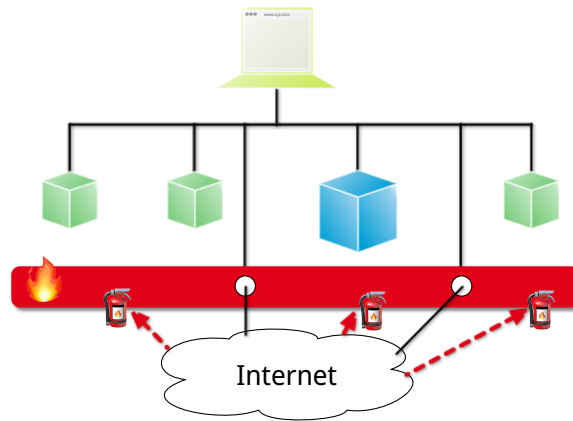
## Von der Notwendigkeit des Schutzes von Rechnern

*[...] Züger und sein Team hätten [...] erst kürzlich ein Experiment durchgeführt, [...]. Sie hätten einen Computer "ohne jeglichen Schutz" mit dem Internet verbunden, um zu sehen, wie lange es dauere, bis er befallen sei. Konkrete Details zur Konfiguration dieses Systems werden zwar nicht genannt, angeblich war der Rechner aber schon nach 20 Minuten infiltriert.*

—**Golem.de 6.2.2024**



## Schutzschicht zwischen internem und externem Netz

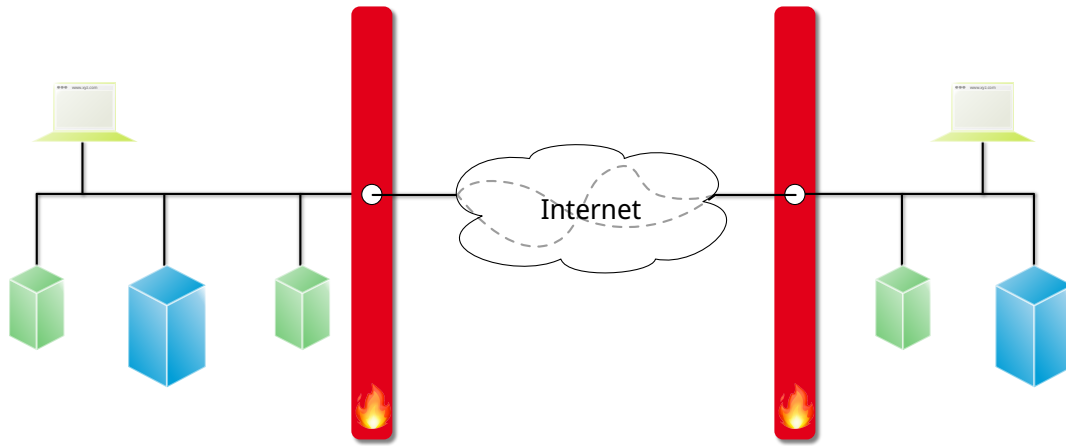


- Kontrolle des Nachrichtenverkehrs durch Filterung
- begrenzte Isolation mit begrenztem Schutz

---

Eine Firewall schafft zwischen verbundenen Netzen Sicherheitsdomänen mit unterschiedlichem Schutzbedarf. Eine wichtige Teilaufgabe ist das Ausarbeiten von Sicherheitsrichtlinien.

## Realisierung von Virtual Private Networks (VPN)

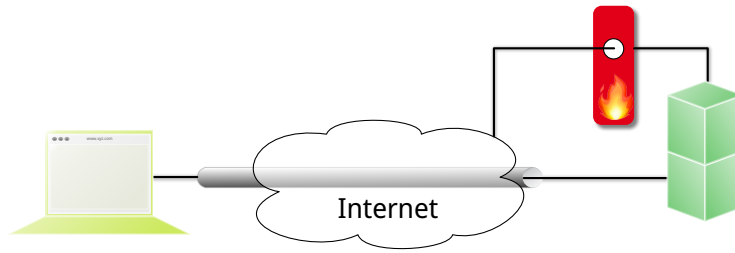


- Aufbau einer scheinbar privaten Verbindung von Firmenteilnetzen über das (öffentliche) Internet.
- Zusätzliche Verbindungsverschlüsselung zwischen den Firewalls.

---

Ziel ist es aktive und passive Angriffe zu unterbinden. Selbst bei verschlüsselten Verbindungen kann die Verkehrsflussanalyse noch Informationen liefern über die Verbindungen liefern.

# Kommerzielle VPNs für Endnutzer



---

## Motivation



- Schutz der Privatsphäre; der ISP kennt nicht mehr die Webseiten, die man aufruft.
- Die IP-Adresse des Nutzers ist den aufgerufenen Webseiten nicht mehr bekannt und kann deswegen der Umgehung von Geo-Blocking dienen.

## Nachteile?

- Vertrauen in den VPN-Anbieter muss vorhanden sein. Insbesondere, beim Einsatz zum Stärken der Privatsphäre, muss der VPN-Anbieter vertrauenswürdig sein und sollte ein so genannter „no-log“ Anbieter sein.
- Es gibt auch (scheinbar kostenlose) VPN-Anbieter, die die Daten der Nutzer dann aber verkaufen (ehemals: **Facebook Onavo**).

# Schutz auf den Schichten des TCP/IP Stacks

Zentraler Schutz des gesamten internen Netzwerks durch:

- Paket Filter ( *Packet Filtering*)
  - Blockieren bestimmter IP-Empfänger-Adressen (extern / intern)
  - Blockieren bestimmter IP-Absender-Adressen (extern / intern)  
(z. B. aus dem Internet mit internen IP-Absender-Adressen)
  - Blockieren bestimmter Dienste; ggf. nur für bestimmte IP-Adressen
- Filter auf Anwendungsebene ( *Application-level Filtering*)
  - inhaltsbezogene Filterung der Verkehrsdaten eines Dienstes  
(z. B. Virenfilter oder Spamfilter)
  - wirkungslos bei verschlüsselten Verkehrsdaten
- Protokollierungsmöglichkeit der Kommunikation von / nach extern

---

Firewalls (alleine) können die Struktur des Netzwerks nicht verbergen.

## DoS Attacke auf Anwendungsebene

### *[...] Angriff auf die Kleinen*


*Waren bei früheren Spamangriffen massenhaft Accounts auf der größten Mastodon-Instanz `mastodon.social` angelegt worden, die dann von dort ihre Inhalte verbreiteten, trifft es nun nicht die größte, sondern die kleinsten. Automatisiert werden dabei Instanzen ausgesucht, auf denen eine Registrierung ohne Überprüfung und sogar ohne ein Captcha möglich ist. Das können etwa solche mit wenigen Accounts sein, die von Enthusiasten etwa für eine Gemeinde betrieben werden. Waren die Verantwortlichen in den vergangenen Tagen nicht aufmerksam, wurden diese Instanzen dann regelrecht überrannt. Die Spam-Accounts verschickten massenhaft Nachrichten mit einem Bild des namensgebenden Frühstücksfleischs und Links zu Discord-Servern, die wohl lahmgelegt werden sollten.*

*—Mastodon: Spamwelle zeigt Schwächen auf [...]*

# Realisierungsmöglichkeiten von Firewalls

- Hardware-Firewall
  - Screening Router
  - Application Gateway (auch Bastion Host)
    - Proxy-Server für bestimmte Dienste
    - Client-Software (HTTP-Browser, telnet, ftp, ...)
    - Server-Software
- Software-Firewall (*Personal Firewall*)

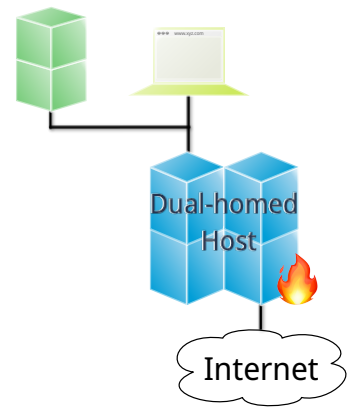
---

Im Falle eines  *Bastion Host*, ist dies der einzige unmittelbar aus dem Internet erreichbare Rechner.

# Dual-Homed Host

## Aufbau

- zwei Netzwerkkarten: ggf. private interne Adressen
- Screening Router & Gate: Packet Filter und Application-Level Filter
- Proxy-Dienste installieren
- Benutzer-Logins von extern



### !! Wichtig

Bei der Konfiguration der Netzwerkkarten gilt:

*IP-Pakete nicht automat. weiterleiten*

# Screening Router

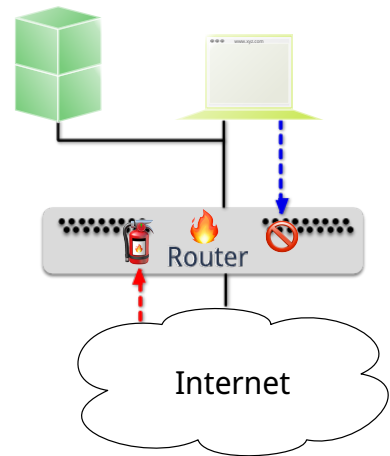
## Aufbau

Programmierbarer Hardwarerouter mit simplen Filterfunktionen:

- nur Paket-Header prüfen
- schnelle Auswertung ermöglicht hohen Durchsatz
- Realisierung eines Packet Filters

## Bewertung

- |                      |                          |
|----------------------|--------------------------|
| ✓ einfach und billig | ! schwer zu testen       |
| ✓ flexibel           | ! Protokollierung        |
|                      | ! Fernwartung            |
|                      | ! keine Inhaltsfilterung |

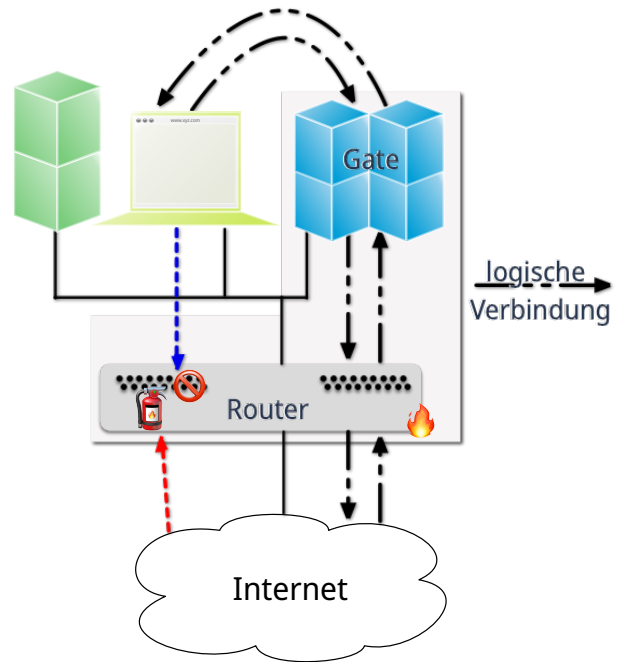





# Screened Host

## Aufbau

- Screening Router blockiert:
  - Pakete von / an interne Rechner (nicht Gate)
  - Source-Routed Pakete
- von extern nur Gate sichtbar
- Pakete von intern nur via Gate
- Gate bietet Proxy-Server (z. B. für E-Mail)



*Source-Routed Pakete* sind Pakete, die den Weg durch das Netzwerk explizit angeben. (*Source-routing* wird auch als *Path Addressing* bezeichnet und wird im Allgemeinen als Sicherheitsproblem angesehen.)

Gibt es für eine bestimmte Anwendung kein Application-level Proxy, dann kann auf einen für TCP/UDP generischen Proxy zurückgegriffen werden. Dieser arbeitet auf dem Session Layer und kann nur die Header-Informationen auswerten. Es handelt sich dann um ein  *Circuit-level Proxy/Gateway*. Im Vergleich zu einem Application-level Proxy ist die Sicherheit geringer, da der Circuit-level Proxy nicht in der Lage ist, die Daten zu interpretieren.

Ein allgemeines Problem ist, dass viele Anwendungen auf generische Protokolle wie HTTP aufsetzen. Weiterhin betreiben einige Anwendungen „Port Hopping“, d. h. sie wechseln den Port wenn der Standardport nicht offen ist.

Eine Anforderung an „Next-generation Firewalls“ ist, dass diese die Analyse von den Daten einer Anwendung unabhängig vom Port und Protokoll ermöglichen.

# Konfiguration eines Gateways

Das Ziel der Konfiguration muss eine minimale angreifbare Oberfläche sein.

- Abschalten aller nicht-benötigten Netzdienste
- Löschen aller nicht benötigter Programme
- Rechte von `/bin/sh` auf 500 setzen
- Rechte aller Systemverzeichnisse auf 711 setzen
- keine regulären Benutzerkennungen
- root-Login mit Einmal-Passwortsystem bzw. 2-Faktor Authentifizierung
- setzen von Platten- und Prozess-Quotas
- volle Protokollierung, möglichst auf Hardcopy-Gerät
- möglichst sichere, stabile und regelmäßig aktualisierte Betriebssystemversion einsetzen

---

Die Rechte von `/bin/sh` auf 500 setzen bedeutet, dass nur der Eigentümer (root) es ausführen kann.

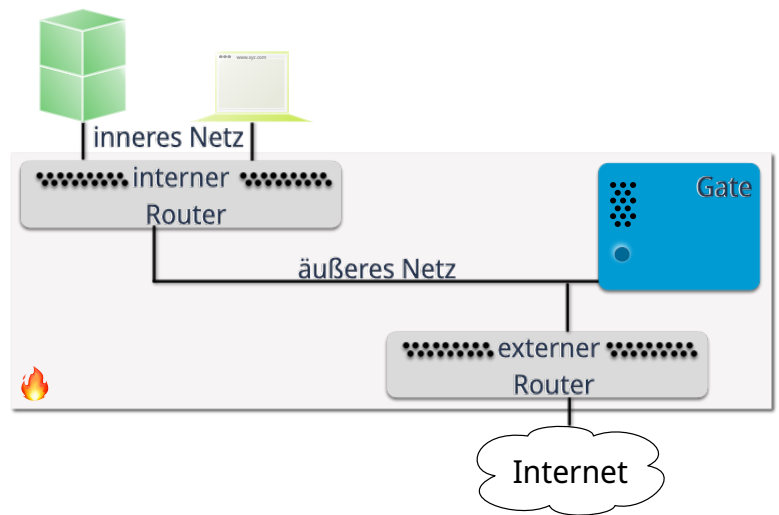
Default:

```
$ ls -al /bin/sh
-rwxr-xr-x 1 root wheel 101232 Oct 1 06:10 /bin/sh
```

# Screened Subnet

## Aufbau

- interner Screening Router als weiterer Schutzwall
  - blockiert Dienste, die nicht einmal bis zum Gate gelangen sollen
  - lässt nur Pakete zum / vom Gate durch
- äußeres Netz realisiert Demilitarisierte Zone (DMZ) für HTTP-Server, Mail-Server, ...



# Intrusion Detection Systeme (IDS)

## Definition

Ein IDS ist ein Gerät (meist ein speziell konfigurierter Rechner), das vielfältige Techniken zur Erkennung von Angriffen anwendet und Angriffe meldet und ggf. abwehrt, in dem (z. B.) die Firewall automatisch umkonfiguriert wird.

## Motivation

- Firewalls alleine sind zu statisch und deswegen häufig nicht ausreichend
- bessere Aufzeichnung und flexiblere Erkennung notwendig
- angepasste Reaktion notwendig

## Umsetzung

An verschiedenen, neuralgischen Stellen werden spezielle Sensoren platziert, die (hier) den Netzwerkverkehr überwachen und verdächtige Aktivitäten melden.

---

Miteinander verwandt bzw. typischerweise in einem Produkt zu finden:

- Intrusion Detection (IDS)
- Intrusion Response (IRS)
- Intrusion Prevention (IPS)


# IDS-Erkennungstechniken

- Signaturerkennung
- statistische Analyse
- Anomalieerkennung

## Probleme

- Fälschlicherweise gemeldete Angriffe (false positives)
- nicht gemeldete Angriffe (false negatives) (insbesondere bei neuartigen Angriffen)
- Echtzeitanforderung, insbesondere bei Hochgeschwindigkeitsnetzen
- Aufzeichnung bei Netzwerken mit Switches ( ⇒ spez. SPAN Port)
- Sensoren sollen unbeobachtbar sein (*stealth*)

---

SPAN (  *Switched Port Analyzer*) Ports sind spezielle Ports auf Switches, die bestimmten Verkehr (z. B. bestimmte Pakete) die über ein Switch gehen, an einen definierten Port weiterleiten können. An diesem Port kann dann eine Analyse des Verkehrs durchgeführt werden / ein Sensor angeschlossen werden.

# Übung

## 2.1. Firewalls

1. Was sind Vorteile eines Dual Homed Host gegenüber einem Paketfilter? Was sind die Nachteile?
2. Benennen Sie zwei konzeptionelle Grenzen von Firewalls. D. h. zwei Szenarien gegen die Firewalls nicht schützen können.
3. Für welche der folgenden Cybersicherheitsstrategien können Firewalls eingesetzt werden:
  1. Angriffe vermeiden
  2. Angriffe erkennen
  3. Angriffe abwehren/Angriffen entgegenwirken
  4. Reaktion auf Angriffe
4. Sie werden beauftragt die Firewall so einzurichten, dass Mails mit Schadsoftware nicht durchgelassen werden. Wie reagieren Sie?