# IT-Sicherheit

## - Sicherheit vernetzter Systeme -

## Kapitel 6: Netzwerksicherheit



## Inhalt

- Schwächen des IP Protocolls
- IPSec Sicherheitserweiterung des IP-Protokolls
  - Authentication Header (AH)
  - Encapsulation Security Payload (ESP)
  - Anwendungsbeispiele
  - Schlüsselverteilung mit IKE (Internet Key Exchange)
- Firewall Klassen
  - Paketfilter
  - Applikationsfilter
  - Verbindungs-Gateway
- Firewall Architekturen
  - Single Box
  - Screened Host
  - (Multiple) Screened Subnet(s)

# IP: Gefahren und Schwächen

- Vertraulichkeit:
  - Mithören einfach möglich
  - Man-in-the-middle Attack
  - Verkehrsflußanalyse
- Integrität:
  - Veränderung der Daten
  - Session Hijacking
  - Replay Angriffe
- Authentisierung:
  - IP-Spoofing

- Lösung: IPSec (Sicherheitserweiterungen für IP)
  - Integraler Bestandteil von IPv6
  - Als Erweiterungs-Header auch in IPv4 einsetzbar



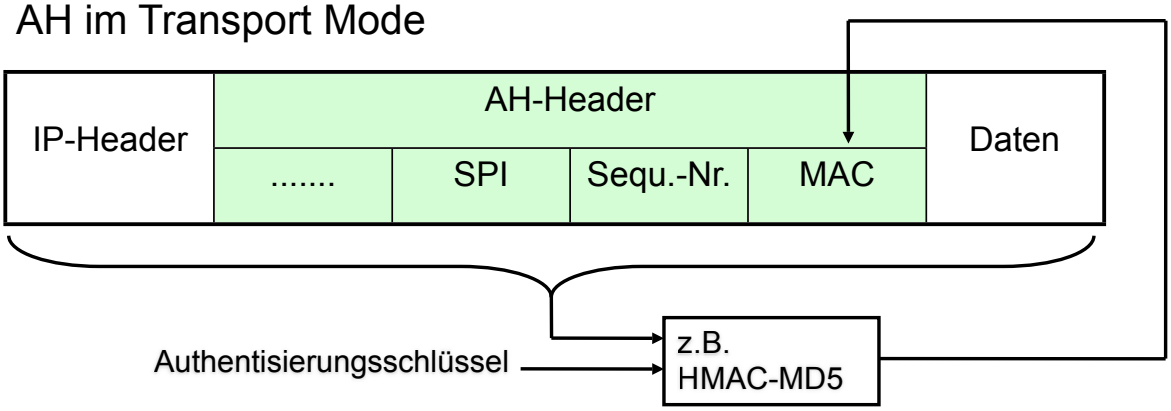
# IPSec Überblick

- IP Authentication Header (AH)
  - Integrität des verbindungslosen Verkehrs
  - Authentisierung des Datenursprungs (genauer des IP Headers)
  - Optional: Anti-Replay Dienst
- IP Encapsulation Security Payload (ESP)
  - Vertraulichkeit (eingeschränkt auch für den Verkehrsfluss)
  - Integrität
  - Authentisierung (der Security Association)
  - Anti-Replay Dienst

- Jeweils zwei verschiedene Betriebsmodi:
  - Transport Mode
  - Tunnel Mode

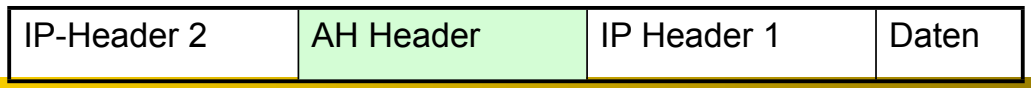
# Authentication Header (AH)

- AH im Transport Mode




  - Integrität durch MAC
  - Authentisierung durch gemeinsamen Schlüssel
  - Anti-Replay durch gesicherte Sequenznummer

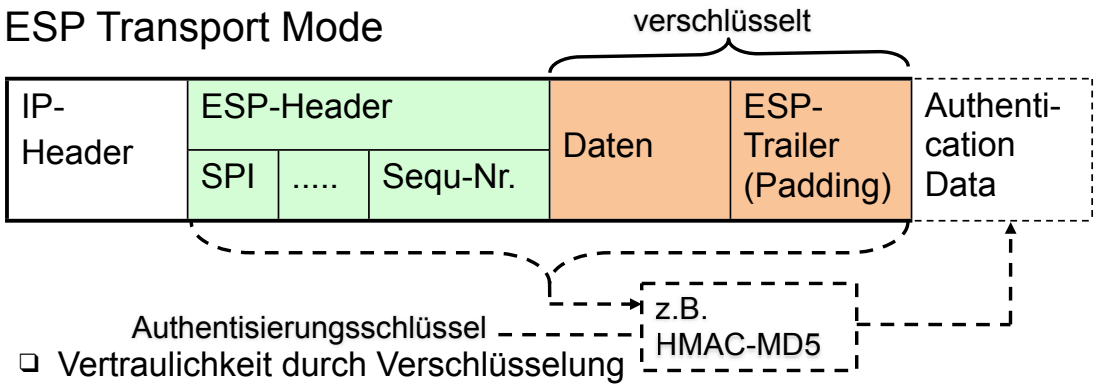
- AH im Tunnel Mode





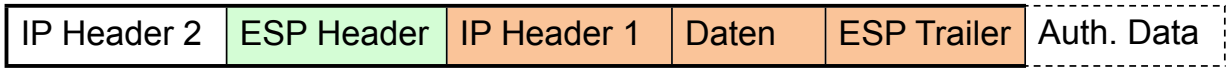

# Encapsulation Security Payload (ESP)

- ESP Transport Mode




  - Vertraulichkeit durch Verschlüsselung
  - Integrität durch MAC (optional)
  - Authentisierung durch HMAC (optional)
  - Anti-Replay durch gesicherte Sequenznummer (optional)

- ESP Tunnel Mode



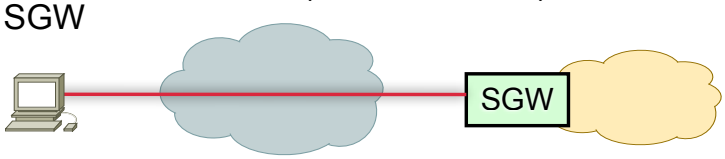

  - Anti-Traffic Analysis durch verschlüsselten IP Header 1

# IPSec Anwendungsszenarien

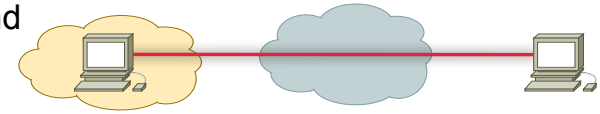
- AH und ESP können kombiniert verwendet werden
- Auch Tunnel und Transport Mode können kombiniert werden

- Mögliche Einsatzszenarien
  - Kopplung von verschiedenen Unternehmensstandorten
    Verbindung von Security Gateway (SGW) zu Security Gateway


  - Telearbeitsplätze; Remote Access („Road Warrior“)
    Endsystem zu SGW

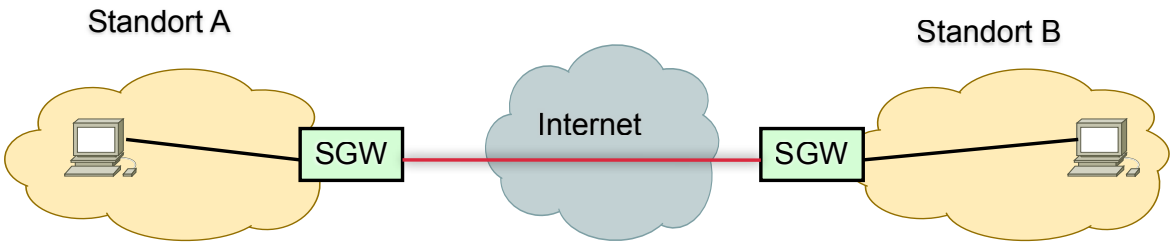



  - End-to-End





# Szenario Standortvernetzung




- Mögliche Anforderungen:
  - Authentisierung SGW-to-SGW oder End-to-End
  - Integritätssicherung SGW-to-SGW oder End-to-End
  - Anti-Replay
  - Vertraulichkeit auch im internen Netz
  - SGW realisiert auch FW Funktionen
  - Verwendung privater IP-Adressen in den Standorten
  - Verschattung interner Netzstrukturen

# Protokollkombinationen

- AH Tunnel Mode am Security Gateway
  - Integritätssicherung
  - Authentisierung SGW to SGW
  - Private Adressen im internen Netz
- ESP Tunnel Mode am Security Gateway
  - Vertraulichkeit (auch der privaten Adressen)
- AH Transport am Endsystem / ESP Transport am SGW
  - Integritätssicherung
  - Authentisierung End to End
  - Vertraulichkeit ab SGW
  - Private Adressen nicht möglich

| IP Header | ESP Header | AH Header | Daten | ESP Trailer |
|---|---|---|---|---|



# Protokollkombinationen (2)

- ESP Transport am Endsystem, AH Transport am SGW
  - Vertraulichkeit End to End
  - Authentisierung SGW to SGW
  - Private Adressen nicht möglich
  - SGW kann nicht mehr filtern (wegen Verschlüsselung)

| IP Header | AH Header | ESP Header | Daten | ESP Trailer |
|---|---|---|---|---|

- AH Transport am Endsystem / ESP Tunnel am SGW
  - Integritätssicherung
  - Authentisierung End to End
  - Vertraulichkeit ab SGW
  - Private Adressen möglich

| IP Header 2 | ESP Header | IP Header 1 | AH Header | Daten | ESP Trailer |
|---|---|---|---|---|---|

# IPSec Security Association (SA)

- Inhalt einer SA
  - IPSec Protocoll Modus (Tunnel oder Transport)
  - Parameter (Algorithmen, Schlüssel, Initialisierungsvektor,...)
  - Lebensdauer der SA
  - Sequenznummernzähler mit –Overflow
  - Anti-Replay-Window
  - .....
- Identifikation einer SA:
  - Security Parameter Index (SPI); 32 Bit Zahl
  - Ziel-Adresse
  - Verwendetes Protocol (AH, ESP)
- D.h. in jede Richtung wird eine eigene SA vereinbart
- Jeder IPSec Teilnehmer hält eine Security Parameter Database (SPD) mit SAs



# IPSec Schlüsselaustausch über IKE

- IKE (Internet Key Exchange)
- Verwendet UDP (Port 500)
- Setzt funktionierende CA voraus
- 2 Phasen
  - Phase 1: Aufbau einer **IKE SA**
    - Main Mode: 6 Nachrichten
    - Aggressive Mode: 3 Nachrichten
  - Phase 2: Aufbau einer **IPSec SA mit Schlüsselaustausch**
    - Quick Mode: 3 Nachrichten
  - Ein Phase 1 Kanal kann für mehrere Phase 2 Exchanges verwendet werden

# Einschub: Diffie-Hellman Schlüsselaustausch

- Ermöglicht den sicheren Austausch eines Schlüssels über einen unsicheren Kanal:
- Primzahl p und eine primitive Wurzel g (mod p) dürfen öffentlich bekannt gemacht werden

- Alice wählt ein x aus [1..p-1]
- Bob wählt ein y aus [1..p-1]
- Alice schickt $A = g^x \bmod p$ an Bob
- Bob schickt $B = g^y \bmod p$

- Beide verwenden den folgenden Schlüssel:

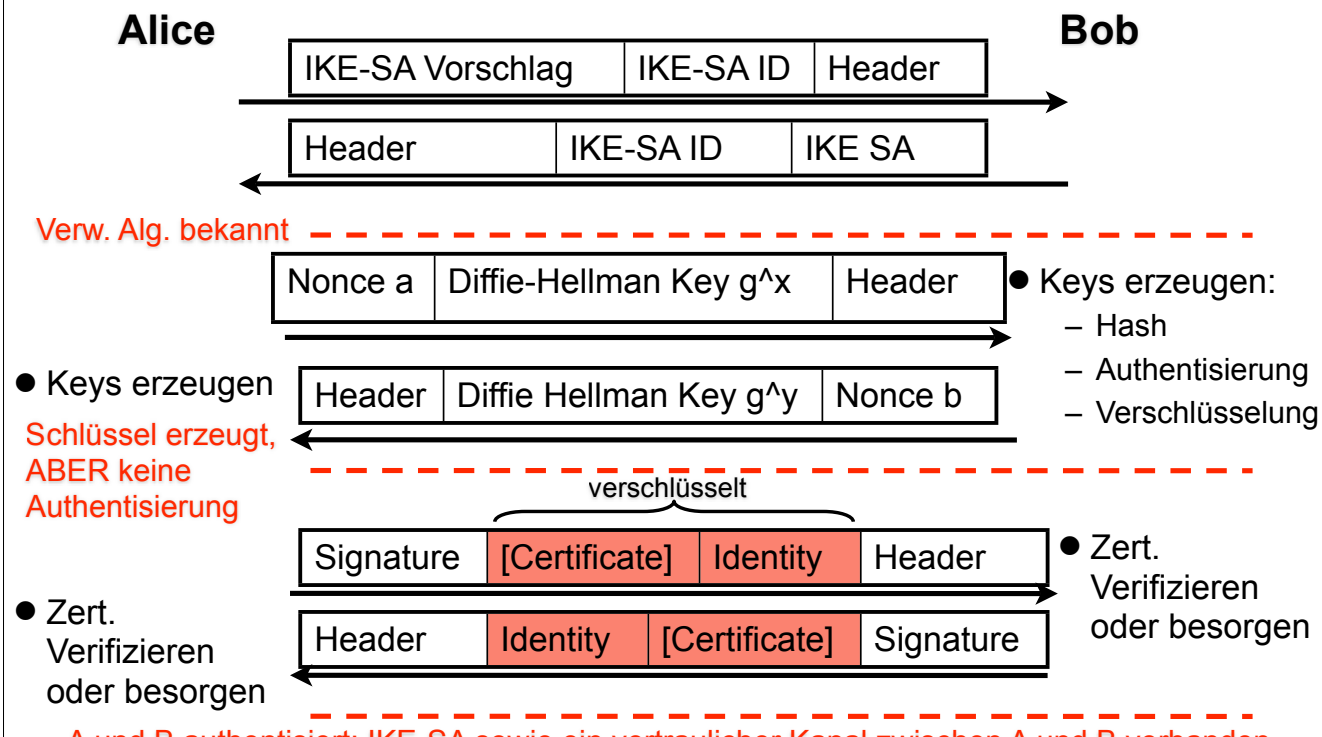
$$Key = A^y = (g^x)^y = g^{xy} = (g^y)^x = B^x \pmod p$$

# IPSec Schlüsselaustausch über IKE

- 2 Phasen
  - Phase 1: Aufbau einer **IKE SA**
    - Main Mode: 6 Nachrichten
    - Aggressive Mode: 3 Nachrichten
  - Phase 2: Aufbau einer **IPSec SA mit Schlüsselaustausch**
    - Quick Mode: 3 Nachrichten
  - Ein Phase 1 Kanal kann für mehrere Phase 2 Exchanges verwendet werden

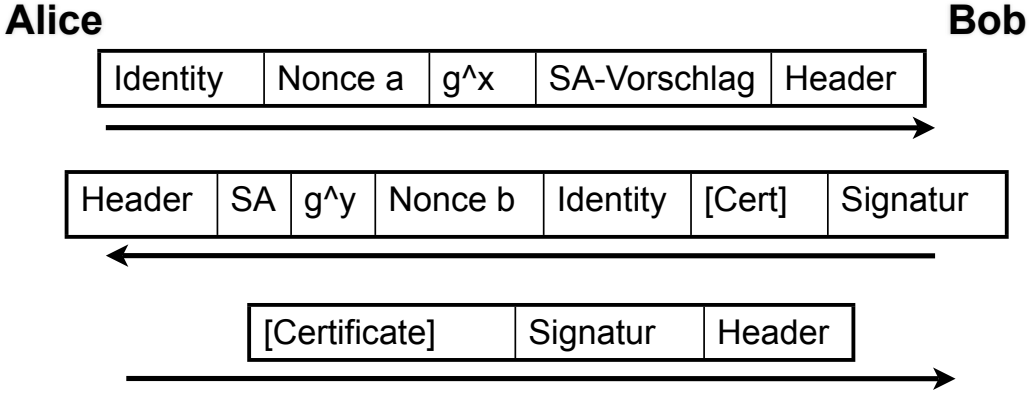


# IKE Phase 1: Main Mode



# IKE Phase 1: Aggressive Mode

- Ziel: Höhere Performance durch Verringerung der Nachrichten




- 3 Nachrichten anstelle von 6
- ABER: keine Vertraulichkeit
  - Preisgabe von Identitäten und damit
  - Verkehrsflussanalyse möglich



# IKE Phase 1: Generierung der Schlüssel

- Erzeugung der Schlüssel für das IKE Protokoll:
  - Master Schlüssel:
    SKEYID = Hash ( Nonce a, Nonce b, g^(xy) )
  - Schlüssel für das Hash Verfahren
    SKEYID_d = Hash (SKEYID, g^(xy), Nonce a, Nonce b, 0 )
  - Authentisierungsschlüssel
    SKEYID_a = Hash (SKEYID, SKEYID_d, g^(xy), Nonce a, Nonce b, 1 )
  - Verschlüsselungsschlüssel:
    SKEYID_e = Hash (SKEYID, SKEYID_a, g^(xy), Nonce a, Nonce b, 2 )

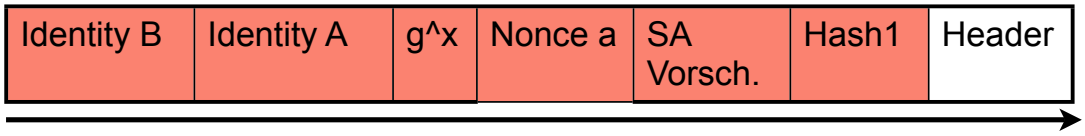
# IPSec Schlüsselaustausch über IKE

- 2 Phasen
  - Phase 1: Aufbau einer **IKE SA**
    - Main Mode: 6 Nachrichten
    - Quick Mode: 3 Nachrichten
  - Phase 2: Aufbau einer **IPSec SA mit Schlüsselaustausch**
    - Quick Mode: 3 Nachrichten
  - Ein Phase 1 Kanal kann für mehrere Phase 2 Exchanges verwendet werden

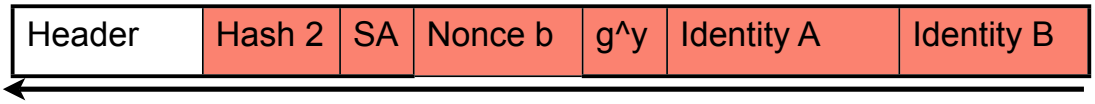


# IKE Phase 2: Quick Mode

- Ziel: Aushandlung einer IPSec SA u. Schlüsselaustausch

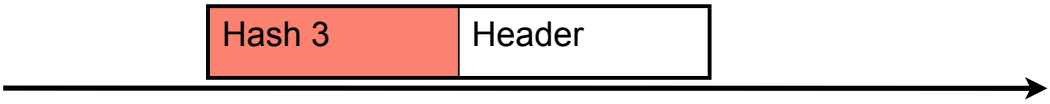



- Hash 1 = Hash ( SKEYID_a, Message-ID, SA, Nonce a, g^x )




- Hash 2 = Hash ( SKEYID_a, Nonce a, Message-ID, SA, Nonce b, g^y )

Schlüssel und Verfahren vereinbart für A -> B und B -> A Kommunikation




- Hash 3 = Hash ( SKEYID_a, , Message-ID, SA, Nonce a, Nonce b )

Acknowledgement

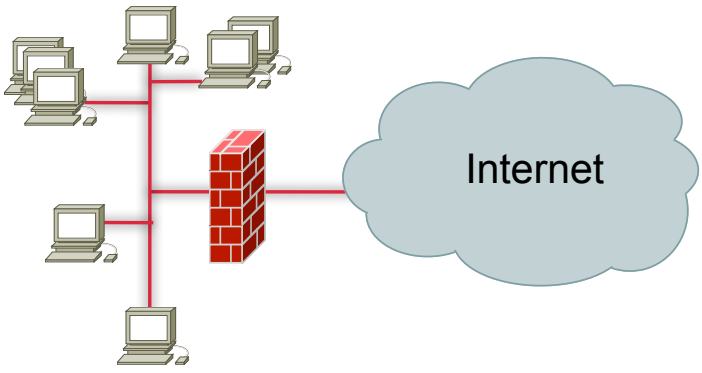
# IKE Phase 2: Generierung der Schlüssel

- Ein Phase 1 Kanal kann für mehrere Phase 2 Aushandlungen verwendet werden, d.h.
  - Parameter der Phase 2 können unabhängig von Phase 1 gewählt werden
  - Z.B. Nonce a der Phase 1 ist nicht dasselbe wie Nonce a der Phase 2
- In Phase 2 können bspw. IPSec Schlüssel vereinbart werden
  - Schlüssel von A nach B:
    KEYMAT_AB = Hash ( SKEYID, g^(xy), Protocol, SPI_B, Nonce a, Nonce b )
  - Schlüssel von B nach A:
    KEYMAT_BA = Hash ( SKEYID, g^(xy), Protocol, SPI_A, Nonce a, Nonce b )



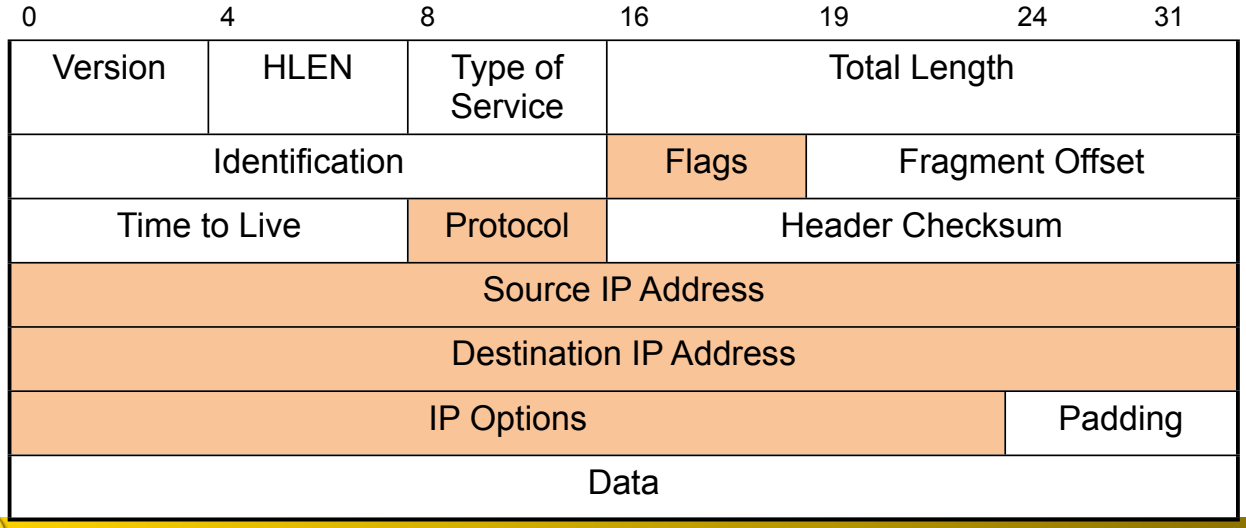
# Firewall-Techniken

- Firewall:
  - Besteht aus einer oder mehreren Hard- und Softwarekomponenten
  - Koppelt zwei Netze als kontrollierter Netzübergang
  - Gesamter Verkehr zwischen den Netzen läuft über die Firewall (FW)
  - Realisiert Sicherheitspolicy bezüglich Zugriff, Authentisierung, Protokollierung, Auditing,....
  - Nur Packete die Policy genügen werden „durchgelassen“
- Klassen:
  - Paketfilter
  - Applikationsfilter (Application Level Gateway)
  - Verbindungs-Gateway (Circuit Level Gateway)
  - Kombinationen daraus



# Paketfilter

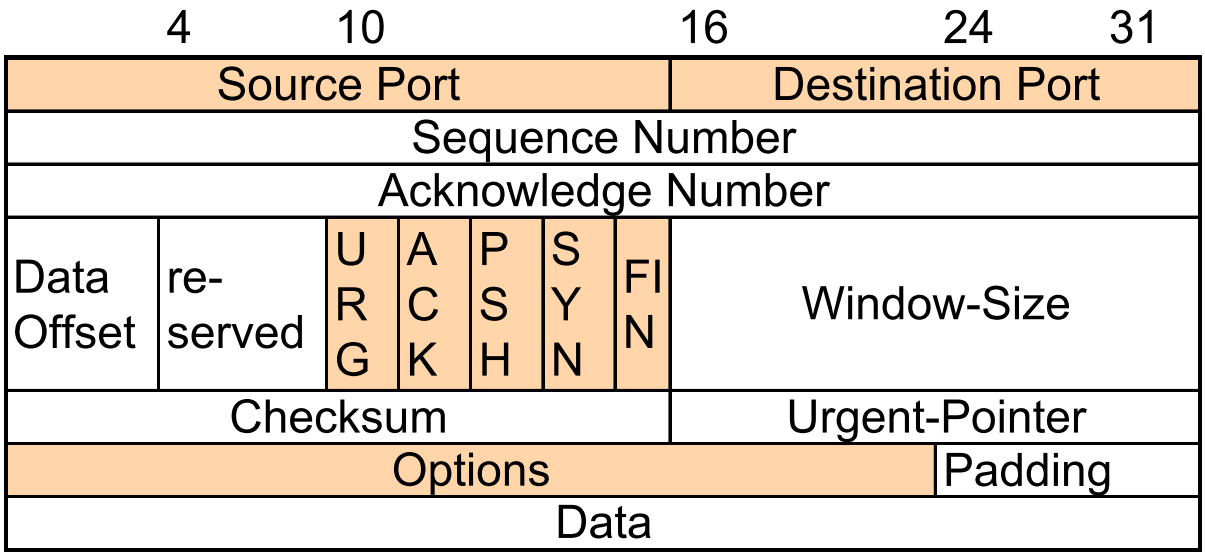
- Filtern auf OSI Schichten 3 und 4
- Filter-Informationen aus den Protokollpaketen
- Im folgenden Beispielhaft TCP / IP
- IP Paket:






# Packetfilter: TCP-Paketformat

- Bei Packetfilter-FW nur Regeln über Felder der Packet-Header möglich!



# FW für TCP/IP : Granularität der Filter

- Schicht 3: wesentliche Filter-Kriterien:
  - Quell-
  - Zieladresse
  - Protokoll der Transportschicht (z.B. TCP, ICMP, UDP,... Vgl. `/etc/protocols`)
- FW auf IP-Basis kann damit Endsysteme filtern (erlauben, verbieten)
- IP-Spoofing u.U. erkennbar, falls:
  - Packet mit interner Quell-Adresse
  - kommt an externem FW-Interface an
- Keine Filterung auf Ebene der Dienste möglich
- Schicht 4: wesentliches Filter-kriterium:
  - Quell- sowie
  - Zielport
  - Flags
- Über Port-Nr. werden Well-Known Services identifiziert;
  z.B. Port 23 = Telnet
  (vgl. `/etc/services`)
- Allerdings nur Konvention; OS setzt diese nicht automatisch durch
- Weitere Konventionen:
  - privilegierte Ports (Ports < = 1023) für Systemdienste
  - Ports > 1023 für jeden nutzbar
- Flags zum Verbindungsauf- und –abbau (vgl. Kap. 3 SYN Flooding)



# Packetfilter: Filterregeln

- Grundsätzliche Ansätze:
  1. Alles was nicht explizit verboten ist, wird erlaubt.
  2. Alles was nicht explizit erlaubt ist, wird verboten.
- Reihenfolge der Regeln wichtig:
  - Regel die zuerst zutrifft wird ausgeführt („feuert“)
  - Daher im Fall 2. letzte Regel immer: alles verbieten
- Statische Paketfilterung
  - Zustandslos
  - Pakete werden unabhängig voneinander gefiltert
- Dynamische Paketfilterung (Statefull Inspection)
  - Zustandsabhängig
  - FW filtert abhängig vom Zustand des Protokoll-Automaten
- Grundsatz: KISS
  Keep it Small and Simple

# Packetfilter-Regeln: Beispiele

- Statischer Paketfilter:
  - Ausgehender Telnet Verkehr erlaubt,
  - Eingehender Telnet Verkehr verboten

| Regel | Source | Destination | Protocol | Source Port | Dest. Port | Flags | Action |
|---|---|---|---|---|---|---|---|
| 1 | <intern> | <extern> | TCP | >1023 | 23 | Any | Accept, Log |
| 2 | <extern> | <intern> | TCP | 23 | >1023 | !SYN | Accept |
| 3 | Any | Any | Any | Any | Any | Any | Drop, Log |

- Dynamischer Packetfilter

| Regel | Source | Destination | Protocol | Source Port | Dest. Port | Action |
|---|---|---|---|---|---|---|
| 1 | <intern> | <extern> | TCP | >1023 | 23 | Accept, Log |
| 2 | Any | Any | Any | Any | Any | Drop, Log |



# Bewertung Packetfilter

- Einfach und preiswert
- Effizient
- Gut mit Router-Funktionalität kombinierbar (Screening Router)

- Integrität der Header Informationen nicht gesichert; alle Felder können relativ einfach gefälscht werden
- Grobgranulare Filterung
- Keine inhaltliche Analyse bei freigegebenen Diensten
- Statische Strategie: Damit Problem bei Diensten, die Ports dynamisch aushandeln (z.B. Videokonferenz-Dienst)
- Abbildungsproblematik: Policy auf konkrete FW-Regeln
- Erstellung einer Filtertabelle nicht triviale Aufgabe
  - Korrektheit ?
  - Vollständigkeit ?
  - Konsistenz ?

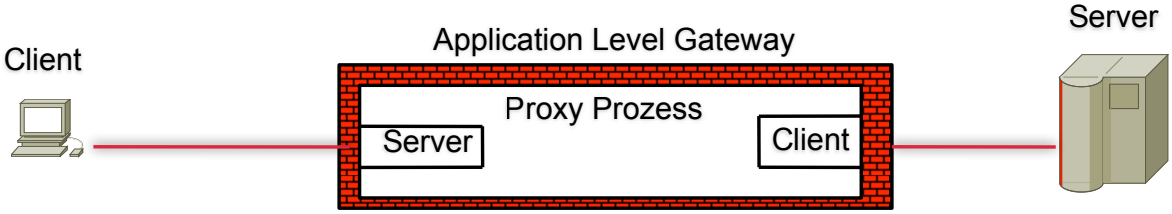
# Weitere Firewall-Techniken auf Schicht 3/4

- Network Adress Translation (NAT)
  - Intern werden andere Adressen („private IP-Adr.“) oder Ports als extern verwendet
  - FW macht Adress/Port-Umsetzung
  - Statisch oder dynamisch
- Masquerading
  - Alle ausgehenden Pakete erhalten Adresse der FW
  - Gesamtes internes Netz wird verborgen
- Anti-Spoofing
  - Binden von FW-Regeln an bestimmte Interfaces (ingress, egress)
  - Wenn an externem Interface ein Packet mit interner Quell-Adresse ankommt muss dieses gefälscht sein



# Applikationsfilter (Application Level Gateway)

- Filtern auf Schicht 7 (Anwendungsschicht)
- Analyse des Anwendungsschichtprotokolls u. d. Protokolldaten
- Für **jeden** Dienst, jedes Protokoll eigener Filter Prozess (auch als **Proxy** bezeichnet) erforderlich
- Interner Client muss sich i.d.R. am Proxy authentisieren
- Proxy trennt Verbindung zwischen Client und Server
- Nach außen erscheint immer nur die Adresse des Application Level Gateways; völlige Entkoppelung von internem und externem Netz
- Kann Zustandsinformationen halten und nutzen



# Proxies

- Für viele Standarddienste verfügbar (z.B. HTTP, Telnet, FTP,..)
- Problematisch für „proprietäre“ Dienste (SAP R3, Lotus Notes,....)
- Beispiel eines HTTP Proxies: Squid
  - Umfangreiche Access Conrol Listen (ACL) möglich:
    - Quell- / Zieladresse
    - Domains
    - Ports
    - Protokolle
    - Protokoll-Primitive (z.B. FTP `put`, HTTP `POST`)
    - Benutzernamen
  - Benutzerauthentisierung am Proxy
  - Zusätzlich Caching-Funktionalität
  - Beispiel:
    - `acl SSL_PORT port 443` (Definition von SSL Ports)
      `acl AUTH proxy_auth REQUIRED` (Benutzerauthentisierung)
      `http_access deny CONNECT !SSL_PORT` (Alle Verbindungen zu einem anderen Port außer SSL verbieten)



# Bewertung Applikationsfilter

- Feingranulare, dienstspezifische Filterung
- Umfangreiche Logging Möglichkeit und damit Accounting
- Zustandsbehaftet
- Inhaltsanalyse (damit z.B. Filterung aktiver Inhalte möglich)
- Benutzerauthentisierung und benutzerabhängige Filterung
- Entkopplung von internem und externem Netz
- Möglichkeit der Erstellung von Nutzungsprofilen

- Möglichkeit der Erstellung von Nutzungsprofilen
- Jeder Dienst braucht eigenen Proxy
- Sicherheit der Proxy Implementierung??
- Problem von Protokollschwächen bleibt bestehen

# Verbindungs-Gateway (Circuit Level Gateway)

- Filtert auf Schicht 7; spezielle Ausprägung des Application Level Gateway
- Circuit Level Gateway stellt generischen Proxy dar
- Nicht auf einzelne Dienste zugeschnitten, allgemeiner „Vermittler“ von TCP/IP Verbindungen
- Trennt Verbindung zwischen Client und Server
- Benutzerauthentisierung am Gateway möglich
- Bsp. `Socks`:
  - Socks-Server filtert den TCP/IP Verkehr
  - Alle Verbindungen der Clients müssen über Socks-Server laufen
  - Daher Modifikation der Clients erforderlich (SOCKSifing)
  - Filtert nach: Quelle, Ziel, Art des Verbindungsaufbaus (z.B. Initiierung oder Antwort), Protokoll, Benutzer



# Bewertung Verbindungs-Gateway

- Anwendungsunabhängige Filterung
- Ein Proxy für alle Dienste
- Umfangreiche Logging Möglichkeit und damit Accounting
- Zustandsbehaftet
- Benutzerauthentisierung und benutzerabhängige Filterung
- Entkopplung von internem und externem Netz
- Möglichkeit der Erstellung von Nutzungsprofilen

- Möglichkeit der Erstellung von Nutzungsprofilen
- I.d.R. keine Filterung nach Dienstprimitiven möglich
- Sicherheit der Proxy Implementierung??
- I.d.R. Modifikation der Clients erforderlich

# Firewall Architekturen

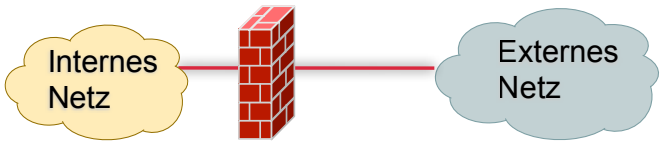
- Kombinationen von FW Komponenten und deren Anordnung wird als FW Architektur bezeichnet

- Single Box Architektur
  - Screening Router
  - Dual Homed Host
- Screened Host
- Screened Subnet
- Multiple Sceened Subnets



# Single Box Architektur

- FW als einziger Übergang ins interne Netz
  - Router (Screening Router) übernimmt FW Funktionalität (i.d.R: Packetfilter)
  - „normaler“ Rechner mit 2 Netzwerk-Interfaces (Dual Homed Host)

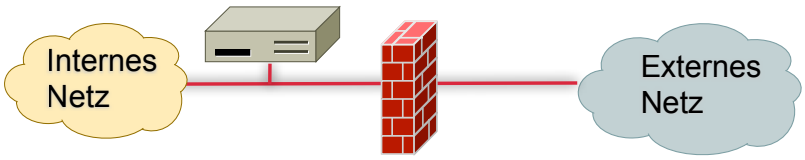



- Billige und einfache Lösung
- Single Point of Administration
- I.d.R. gute Performance (falls nur Packetfilter eingesetzt wird)

- ✶ Wenig flexibel
- ✶ Single Point of Failure

# Screened Host

- FW (**Bastion Host**) liegt im internen Netz (nur 1 Interface)
- Verkehr von außen wird über Screening Router (vor-) gefiltert und i.d.R. zum Bastion Host geleitet
- Bastion Host kann Aplication Level Gateway oder Circuit Level Gateway realisieren

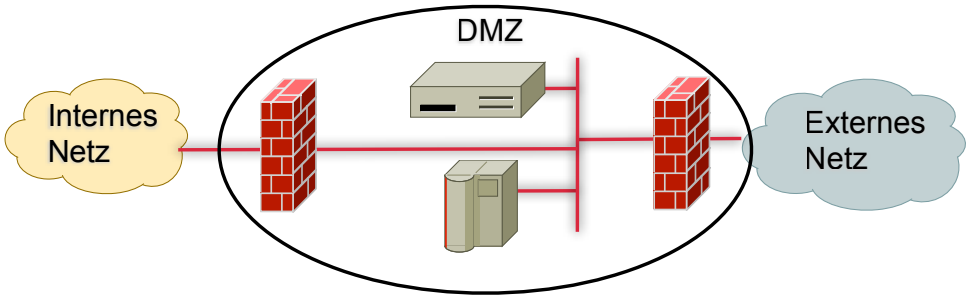



- Trennung von Packet- und Applikationsfilter
- Vorfilterung des externen Verkehrs
- Hohe Flexibilität
- ✶ Packete können immer noch direkt in internes Netz gelangen



# Screened Subnet

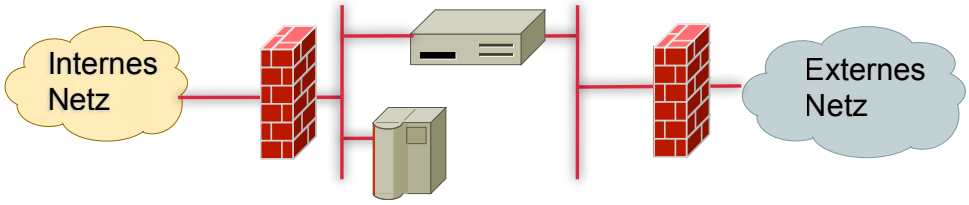
- FW Komponenten liegen in einem eigenen Subnetz (Perimeter Subnet) auch demilitarisierte Zone (DMZ) genannt
- Schutz der DMZ sowohl nach innen als nach außen durch Paketfilter
- Erweiterung der DMZ um dezidierte Server, z.B. WWW



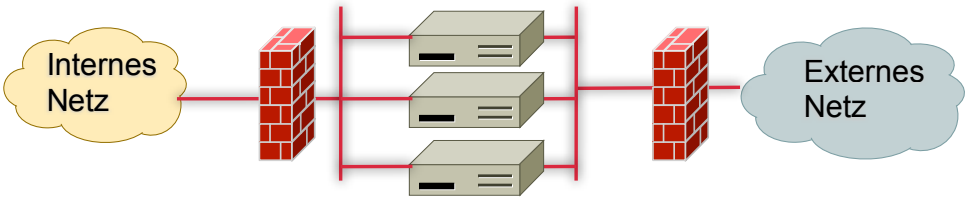

- Keine direkte Verbindung von extern nach intern mehr möglich
- Zusätzlicher Grad an Sicherheit
- Interner Router/FW schützt vor Internet und ggf. vor DMZ

# Multiple Screened Subnet

- Verwendung zweier Perimeter Subnets getrennt durch Dual Homed Host




- Verwendung mehrerer Bastion Hosts (Redundanz)






# Möglichkeiten und Grenzen von Firewall-Arch.

- Abgestufte Sicherheitskontrollen (vom Einfachen zum Komplexen)
- Möglichkeiten effizienter Protokollierung
- Möglichkeiten der Profilbildung

- Problem der Fehlkonfiguration
- Umfangreiche Kenntnisse erforderlich
- Trügerische Sicherheit
- Erheblicher Administrationsaufwand
- Tunnel-Problematik
  - Anwendungsprotokolle werden z.B. über HTTP getunnelt
  - FW kann dies nicht erkennen

# Intrusion Detection Systeme (IDS)

- Intrusion Detection Systeme (IDS)
  - Ziel: „Angriffe“ automatisch erkennen (und ggf. blockieren)
- Intrusion Prevention Systeme (IPS)
  - Ziel: Auf „Angriffe“ reagieren (verhindern / Gegenmaßnahmen ergreifen Gegenangriff durchführen)
- Im folgenden:
  - Host basierte IDS
  - Netz-basierte IDS
  - Angriffs- bzw. Mißbrauchserkennung
  - Beispiele



# Host-basierte IDS (HIDS)

- Ausgeführt auf dem zu überwachenden System
  - Systemüberwachung
  - Applikationsüberwachung
  - Integritätsüberwachung (kryptographische Prüfsummen; Hashing)

\+ Individuell an zu überwachendes System anpassbar

\+ Sehr spezifische Aussagen über erkannte Angriffe möglich

\- Benötigt Ressourcen des zu schützenden Systems

\- HIDS selbst als Angriffsziel

\- Anpassung an individuelles System erforderlich

\- Fällt bei erfolgreichem Angriff mit angegriffenem System aus

- Bsp.:
  - Tripwire
  - AIDE (Advanced Intrusion Detection Environment)

# Netz-basierte IDS (NIDS)

- Eigener Sensor (kein Gastsystem)
- Überwachen den Netzverkehr (Mithören):
  - Eines Rechners
  - Eines (Sub-) Netzes
  - Einer gesamten Domäne
- \+ Ein Sensor für das gesamte Netz
- \+ NIDS kann „unsichtbar“ installiert werden
- \+ NIDS kann ausfallsicher installiert werden
- \+ Verteilte Angriffe erkennbar
- \- Betrieb am Mirror Port von Netzgeräten; Packet-Drop
- \- Überlast des gesamten NIDS möglich
- \- Verschlüsselte Daten und Kanäle
- \- „nur“ Netzauffälligkeiten erkennbar



# Angriffs- bzw. Mißbrauchserkennung

- Signatur-basiert:
  - Pattern-matching & Expertensysteme
  - „typische Angriffsmuster“
  - \+ Zuverlässigkeit
  - \- Nur bekannte Angriffe erkennbar (keine Zero Day Exploits)
  - Bsp.: Snort
- Anomalie-Erkennung:
  - Lernt „Normalverhalten“
  - Abweichung wird als Angriff gedeuted
  - \+ Flexibel bei neuen Angriffen
  - \- Adaptivität bei Netzänderungen
  - \- False Positives & False Negatives
- Integritätsprüfung
  - Berechnung kryptographischer Hashes
  - Speicherung auf WORM und Vergleich
  - \+ Schnell und Zuverlässig
  - \- Aufwand bei gewollter Datenänderung
  - Bsp.: Tripwire
- Kombination der Verfahren

# Bsp.: Tripwire

- www.tripwire.org
- Host-basiertes IDS
- Überwacht anzugebende Dateien und Verzeichnisse
- Erstellt (verschlüsselte) Datenbank mit Prüfsummen:
  - MD5
  - SHA-1
  - HAVAL, ...
- Berechnet regelmäßig Hashes und vergleicht mit gespeicherten
- Probleme:
  - Auswahl schützenswerter Objekte
  - Befugte Änderungen
  - Initialstatus muss sicher sein



# Bsp. Snort


- www.snort.org
- Netzbasiertes IDS
- Signatur-basierte Analyse
  - Regeln in Dateien definiert (*.rules)
  - Alle Regeln mit logischem ODER verknüpft
  - Netzverkehr wird gegen diesen Regelsatz geprüft
- Signatur Beispiel:

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg: "mountd access";)
```

  - alert: Alarm generieren; Packet loggen
  - Protokoll Addresse Port -> (Richtungsoperator) Adresse Port
  - content: (Regel Option) Suche nach Muster in der Payload
    - String und/oder
    - Binärdaten; Hex-Notation; gekennzeichnet durch | |
  - msg: Nachricht in Alarm und Log aufnehmen
  - mehrere Regel Optionen mit logischem UND verknüpft