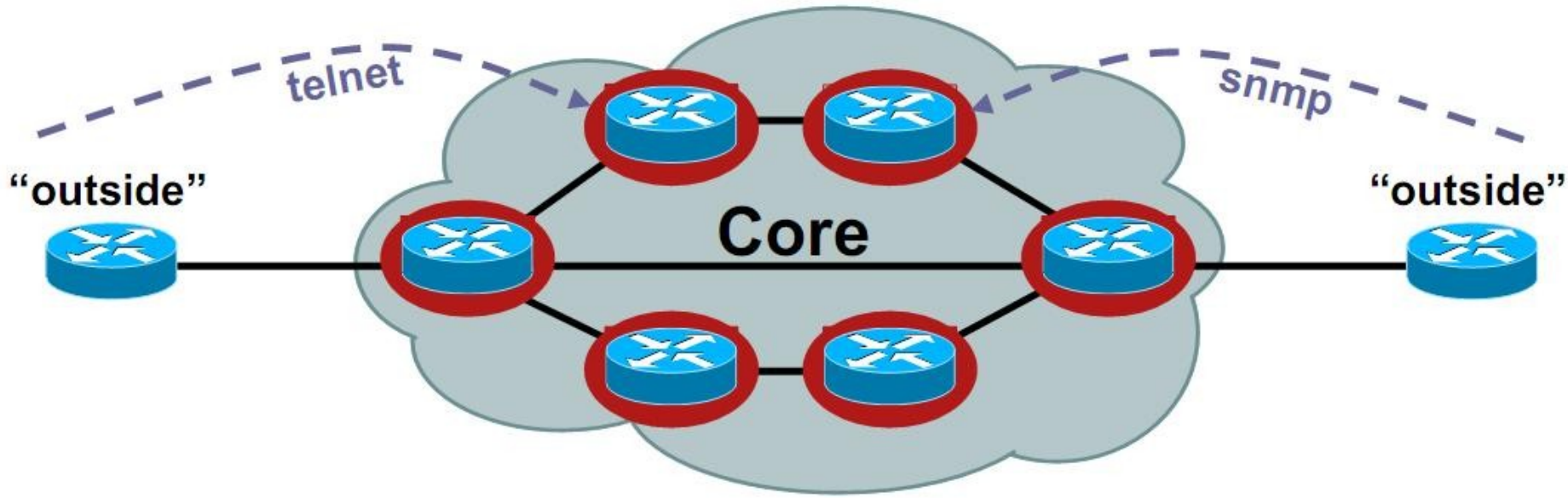
Network Core Protection Best Practices

Thorsten Dahm
t.dahm@resolution.de
10.09.2009

Agenda

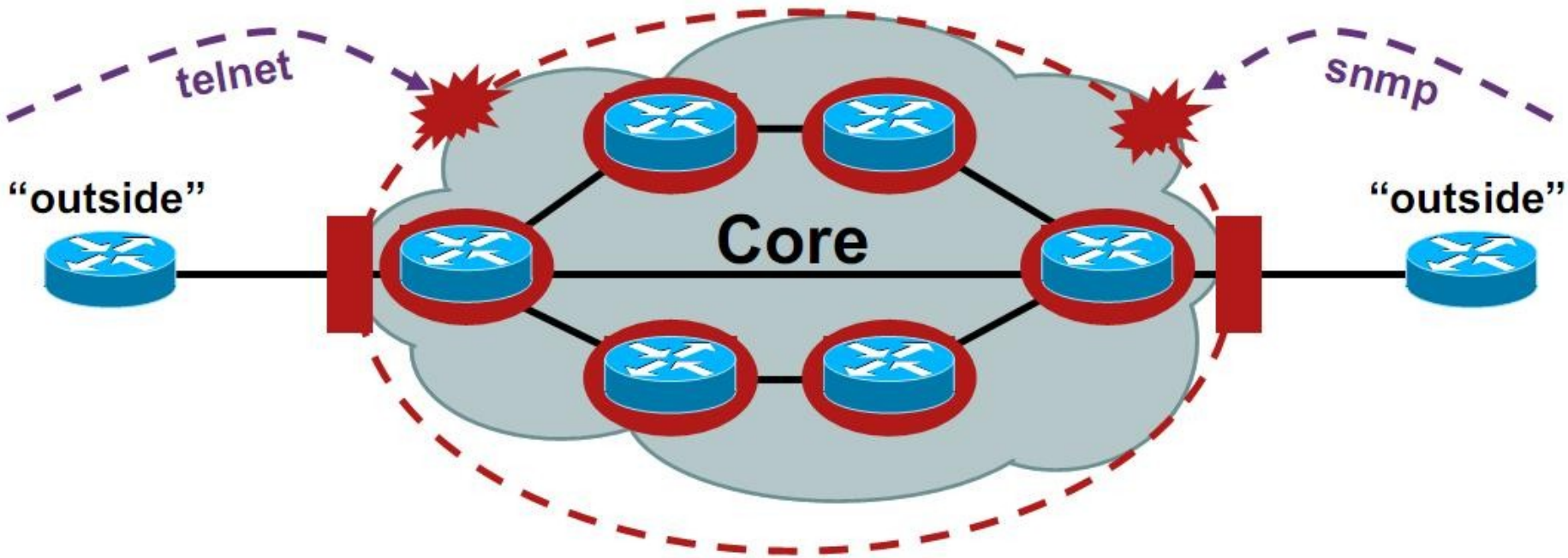
- Schützen der eigenen Infrastruktur
- Überblick über die Probleme
- Methoden zum Schutz der eigenen Infrastruktur:
  - Traditionelle Methoden
  - Schützen der CPU
  - Netzwerk abhärten
- Nicht (!) Teil dieser Präsentation: schützen von Netzwerkverkehr der durch das eigene Netzwerk von Kunde zu Kunde geroutet wird

Traditionelles Netzwerkdesign



- Core router individuell abgesichert
- Jeder Router von außen erreichbar

Netzwerk abhärten



- fernhalten ungewollter IP Pakete vom Core

Die drei Sicherheitscharakteristiken

- Verfügbarkeit
- Vertrauenswürdigkeit
- Integrität

-> Ziel: beibehalten und pflegen

Verfügbarkeit: Schütze die Infrastruktur

- Sicherheit: ein Herzstück von Netzwerken
- Internet: wandel von vertrauenswürdig zu nicht vertrauenswürdig
- Vertraue keinem IP-Paket!
- Vertrauen erarbeiten durch Filtern und Policys
- Fundamental: schützen der eigenen Netzwerk-Infrastruktur
- Sollte Teil jedes Netzwerk Designs sein
- Sicheres und stabiles Netzwerk = Grundlage der Geschäftstatigkeit

Eine angemessene Betrachtungsweise

- Die vorgestellten Möglichkeiten sind nützlich, müssen aber zum eigenen Netzwerk Design passen
- Dürfen die Konnektivität nicht stören/zerstören!
- Nicht alle Features gleichzeitig implementieren!
- Besser: ein Feature welches man verstanden hat
- Was der eigenen Einschätzung nach am wichtigsten ist
- Erstes Ausrollen im realen Netzwerk nur kontrolliert und in einem bestimmten Bereich
- Das gelernte aufschreiben
- Anleitungen erzeugen
- Es kann/darf durchaus 1 Jahr und länger dauern!

Unterscheidung von Attacken

- Intern:
  - menschliche Fehler ("dicke Finger")
  - Interner Angreifer

- Extern:
  - Würmer
  - Packet floods (darauf konzentrieren wir uns im Moment)
  - Sicherheitslücken
  - Eindringversuche
  - Route hijacking
  - Attacken auf Services (DNS, Sprache, Video, ...)

Beispiele: Angriffe auf Router

- data plane:
  - HTTP
  - FTP
- control plane:
  - ARP
  - BGP
  - OSPF
- management plane:
  - Telnet
  - SSH
  - SNMP
  - NTP

Angriffsmöglichkeiten

- Füllen der Queue zwischen den Interfacen und dem RP
- Überlasten der supporting ASIC CPU (zwischen ASIC und RP)
- Überladen der Verbindung zwischen ASIC CPU und RP
- Überladen des Input Buffers des RP
- Überlasten des RP selbst

Absichern von Routern - Best Practices

- Es gibt viele Anleitungen, z. B. von cymru, der NSA, ...
- Die meisten Vorschläge fallen nicht in den Themenbereich dieses Vortrages
- Für jeden Vorschlag gibt es einen Grund (z. B. SSH Version 2)
- Manchmal gibt es auch eine interessante Story hinter den erfolgreichen Angriffen :-)
- Sehen wir uns mal ein paar Features an

Ein paar Beispiele (unvollständig)

- http://www.first.org/resources/guides/
- http://www.sans.org/resources/policies/
- http://www.ietf.org/html.charters/opsec-charter.html
- http://www.nsa.gov/snac/routers/C4-040R-02.pdf
- http://www.cymru.com/Documents/secure-ios-template.html
- http://www.cymru.com/gillsr/documents/junos-template.pdf
- http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080120f48.shtml
- http://www.juniper.net/solutions/literature/app_note/350013.pdf

Absichern von Routern - Traditionelle Methoden (bei Cisco)

- Alle nicht benötigten Dienste abschalten:
  - no service tcp-small-servers / udp-small-servers
  - no cdp run
  - no crypto isakmp enable
- VTY ACLs
- SNMP community ACLs
- SNMP Views
- Abschalten von SNMP rw (oder v3 wenn benötigt)
- Verwerfen von toten TCP sessions die sonst nur ein vty belegen:
  - service tcp-keepalives-in
- QoS an den Netzgrenzen
- Benutzen von "secret" statt "password" bei Passwörtern
- Benutze AAA (Authorization und Accounting nicht vergessen!)
- Logging
- Nicht benötigte Features abschalten:
  - no ip directed-broadcast
  - no ip proxy-arp
  - no ip redirects

Absichern von Routern - Traditionelle Methoden (bei Cisco)

- Validieren der Source-Adressen (RFC2827/BCP38, RFC3704/BCP84):
  - ip verify unicast source reachable-via {any|rx}
  - cable source-verify [dhcp]
  - ip verify source [port-security]
- Source-Routing abschalten:
  - no ip source-route
- Prefix-filter bei eBGP sessions
- BGP max-prefix benutzen
- MD5 bei BGP und dem IGP nutzen
- MD5 bei VRRP/HSRP benutzen
- Hardwareabhängige Schutzmaßnahmen:
  - scheduler allocate (reserviert CPU Zyklen für management)
  - ip icmp rate-limit unreachable
  - Interface null0
    no ip unreachables

Control Plane Policing (CoPP)

- Cisco-Version eines "echten" Loopback-Interfaces
- Nicht nur permit/deny, auch rate-limits verfügbar
- Auf allen Cisco-Plattformen verfügbar
- gleiche Syntax auf allen Plattformen
- CoPP benutzt das "Modular QoS CLI" (MQC) für die QoS policy definition

Control Plane Policing (CoPP)

Die 4 Schritte auf einen Blick:

1. ACLs definieren
   Klassifizieren des Netzwerk traffics

2. class-maps definieren
   Bauen von sogenannten traffic klassen

3. policy-map definieren
   Die QoS policy den einzelnen class-maps zuweisen (police, drop)

4. Die CoPP policy dem control plane “interface” zuweisen

CoPP: ACLs definieren (Beispiel)

- Critical - unbedingt notwendig (z. B. OSPF Hello Pakete)
- Important - tägliche Arbeit (z. B. SSH)
- Normal - gewollt, nicht essentiell notwendig
- Undesirable - “böse” oder “unerwünscht”

- Catch-All - jeglicher andere IP traffic welcher zum RP geschickt wird und bis jetzt nicht identifiziert wurde
- Default - jeglicher übrige nicht-IP traffic welcher zum RP geschickt wird und bis jetzt nicht identifiziert wurde

class-maps definieren

- Erzeuge class-maps um die Klassifizierung zu komplettieren
  - benutze vorher definierte ACLs
- mehrere match Kriterien

  match-any bedeutet jedes Paket, auf welches ein Kriterium zutrifft

  match-all bedeutet das jedes Paket, auf welches alle Kriterien zutreffen
- “undesirable” class sollte der “match-any” Regel folgen

policy-map definieren

- Zuweisung class-maps <-> policy-maps
- Definieren von Aktionen
- Beispiele:
  - für ungewollten Traffic definieren von "drop" (ungeachtet der traffic rate)
  - für kritischen oder wichtigen Traffic definieren von "transmit" (ebenfalls ungeachtet der traffic rate)
  - für catch-all definieren eines rate-limit

- default-class für undefinierten Traffic: konfigurierbar, jedoch nicht löschbar

CoPP - die finale Konfiguration

```
access-list 121 permit tcp host 10.1.1.2 eq bgp host 10.1.1.1 gt 1024
access-list 121 permit tcp host 10.1.1.2 gt 1024 host 10.1.1.1 eq bgp

class-map match-any CoPP-critical
     match access-group 121

policy-map CoPP
 class CoPP-critical
     police 5000000 2500 4470 conform-action transmit  exceed-action
     transmit

Router(config)# control-plane
Router(config-cp)# service-policy [input | output] <policy-map-name>
```

Monitoring CoPP

- show access-list
- show log (wenn das keyword "log" in der ACL benutzt wurde)
- show policy-map control-plane
- SNMP Queries

CoPP Monitoring

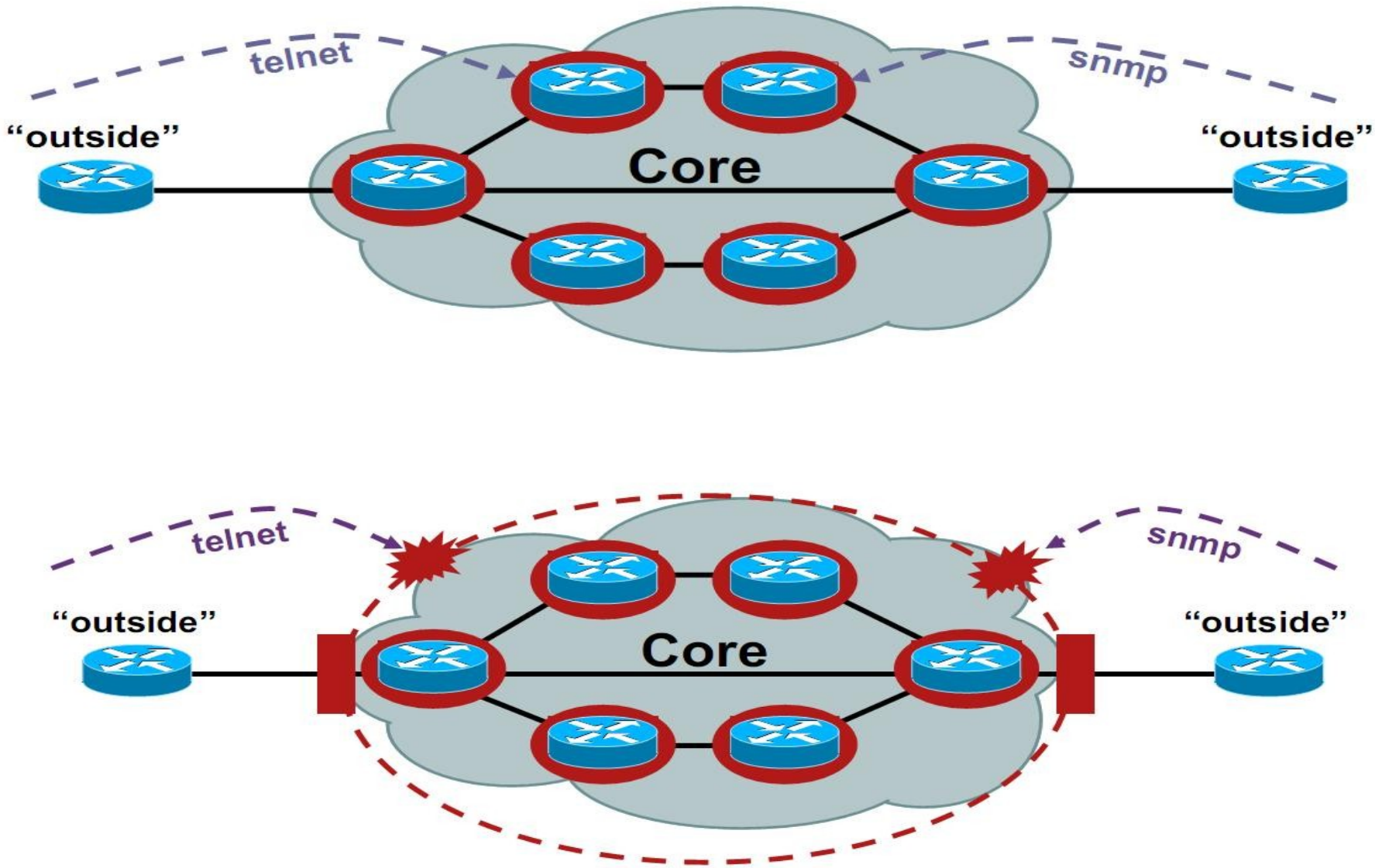
```
Router#show policy-map control-plane input
 Control Plane
   Service-policy input: CoPP
     Class-map: CoPP-critical (match-all)
       16 packets, 2138 bytes
        5 minute offered rate 0 bps, drop rate 0 bps
        Match: access-group 121
        police:
            cir 5000000 bps, bc 2500 bytes
            conformed 16 packets, 2138 bytes; actions:
              transmit
            exceeded 0 packets, 0 bytes; actions:
              transmit
            conformed 0 bps, exceed 0 bps
```

Netzwerk abhärten

- Im Falle einer DoS Attacke ist es bereits zu spät wenn das Paket den Router erreicht
  - CoPP hilft in diesem Falle, löst aber nicht das Problem
  - Besser: verwerfen der unerwünschten Pakete an der Netzwerkgrenze

- Eine Methode um das Problem zu lösen:
  - Infrastruktur ACLs

Vergleich vorher - nachher



Infrastruktur ACLs

- Grundvorraussetzung: Traffic zu den eigenen Core Routern filtern
- Eine Liste erzeugen mit benötigten Protokollen welche von außerhalb des eigenen AS kommen und zu den Core Routern müssen (z. B. eBGP peerings, GRE, IPSec, ...)
- Der (möglichst zusammenhängende) Adress-Block der Core Router ist der zu schützende IP Bereich
  - Summarization hält die ACLs klein
  - schlechte Summary macht ACLs schlechter managebar

Infrastruktur ACLs

- Erlauben nur wirklich benötigte Protokolle und Verbindungen
- Sollten ebenfalls das anti-spoofing Filtern übernehmen:
  - RFC3330 definiert IPv4 Adressen für spezielle Verwendung
  - Verwerfen von eigenen IPs als source von außen
  - Verwerfen von RFC1918 Adressen
  - Verwerfen von Multicast source Adressen (224/4)

Infrastruktur ACLs

RFC 1918 Adressen werden im Internet doch sowieso nicht geroutet, oder doch?

Router#sh ip access <name>
    100 deny ip 10.0.0.0 0.255.255.255 any (12 matches)
    120 deny ip 169.254.0.0 0.0.255.255 any (15 matches)
    130 deny ip 172.16.0.0 0.15.255.255 any (753 matches)
    140 deny ip 192.168.0.0 0.0.255.255 any (24 matches)

Infrastruktur ACLs

- Müssen Transit erlauben
  - IP Verkehr, welcher durch den Core Router weitergeleitet werden soll, muß mit permit ip any any erlaubt werden
- Anwendung eingehend am ingress Interface
- Fragmente von IP Paketen mit destination == Core Router können mittels "fragment" keyword verworfen werden

Schrittweise Implementierung

- Üblicherweise werden nur sehr wenige Protokolle gebraucht
- Noch weniger haben eine Source-IP von außerhalb des eigenen AS
- Die benötigten Zugriffe werden mittels einer ACL definiert
- Diese wird schrittweise ausgerollt und getestet

Definition was erlaubt werden muß

- Jedes IP Paket zum Backbone muß klassifiziert sein
- Hilfe durch NetFlow (export und manuelle Durchsicht)
- "log" keyword (vorsicht, kostet CPU Zyklen)
- Unerwartete Ereignisse sorgfältig untersuchen
- Kein Protokoll / IP Paket erlauben welches man nicht erklären kann!

Implementation im Netzwerk

- Erlaube in der ACL alle vorher identifizierten, notwendigen Zugriffe auf den Backbone Adress-Block
- Verwerfe alle anderen Zugriffe auf den Backbone Adress-Block
- Überwache die counter mit "sh access-list" sowie das Log um evtl. übersehene Zugriffe zu identifizieren
- Letzter Eintrag in der ACL muß "permit ip any any" sein um Transit-Traffic zu erlauben!
- Die ACL bietet nun einen Basis-Schutz der eigenen Backbone Infrastruktur

Die Sicherheit erweitern

- Basisschutz: nur die benötigten Protokolle sind erlaubt
- Nun werden nur eine handvoll Source-IPs für den Zugriff mittels dieser Protokolle erlaubt
- z. B. Bastion-Hosts für SNMP, eBGP peers, Tunnel Endpunkte usw.
- Wenn möglich auch destination IP Filter nutzen

Ein kleines Beispiel...

```
! Deny our internal space as a source of external packets
  access-list 101 deny ip our_CIDR_block any
! Deny src addresses of 0.0.0.0 and 127/8
  access-list 101 deny ip host 0.0.0.0 any
  access-list 101 deny ip 127.0.0.0 0.255.255.255 any
! Deny RFC1918 space from entering our AS
  access-list 101 deny ip 10.0.0.0 0.255.255.255 any
  access-list 101 deny ip 172.16.0.0 0.0.15.255 any
  access-list 101 deny ip 192.168.0.0 0.0.255.255 any
!Permit eBGP from outside
  access-list 101 permit tcp host peerA host peerB eq 179
  access-list 101 permit tcp host peerA eq 179 host peerB
! Deny all other access to infrastructure
  access-list 101 deny ip any core_CIDR_block
! Permit all data plane traffic
  access-list 101 permit ip any any
```

Infrastruktur ACLs - Zusammenfassung

- Infrastruktur ACLs sind sehr wirkungsvoll wenn sie ordendlich aufgebaut sind und überall benutzt werden
- Werden seit Jahren von vielen ISPs benutzt
- Address Summary ist essentiell für einen erfolgreichen Einsatz
- Infrastruktur ACLs haben auch einige Schwächen:
  - Erhöhter administrativer Aufwand für die Verwaltung von ACLs
  - Kollidiert evtl. mit anderen (Kunden-) ACLs

Ende :-)

Fragen?