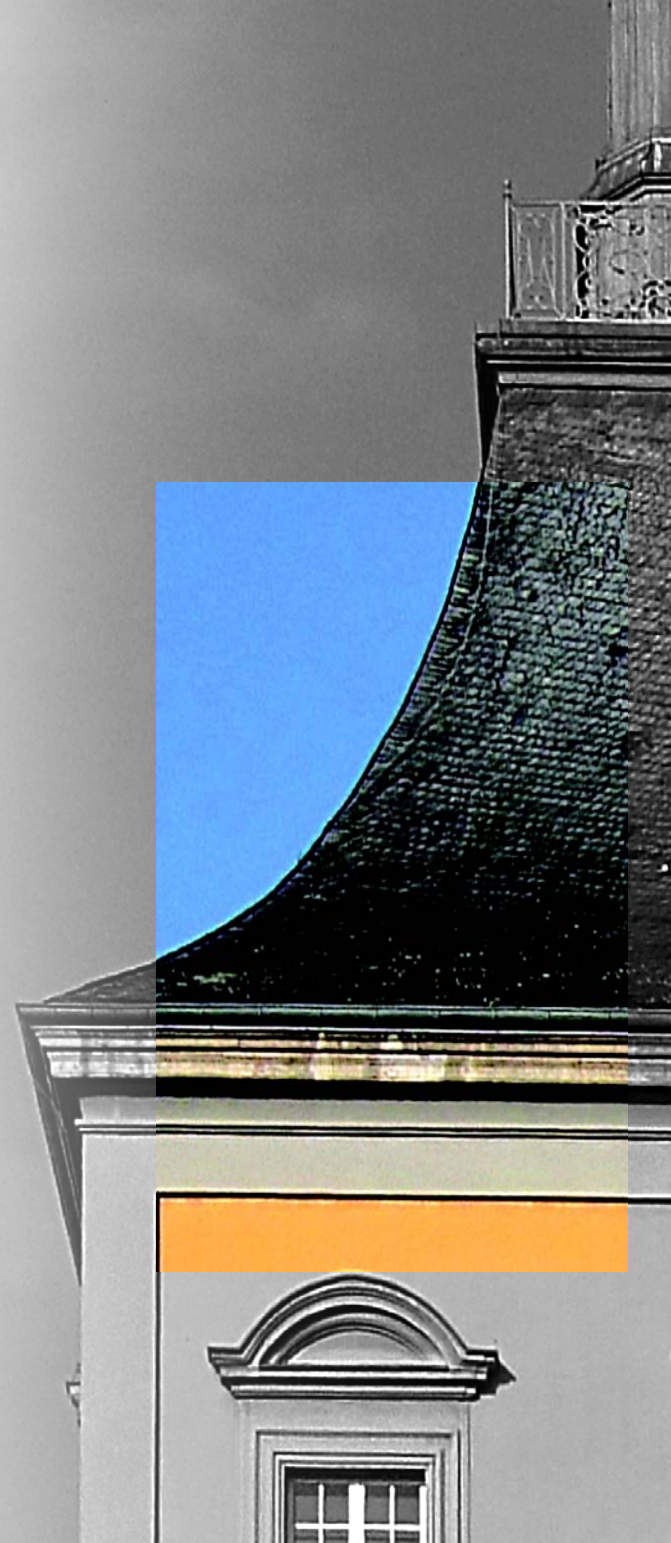
UNIVERSITÄT BONN

VORLESUNG

NETZWERKSICHERHEIT

SOMMERSEMESTER 2023

MO. 14-16 UHR



UNIVERSITÄT BONN

KAPITEL 5

INTERNET PROTOCOL V6 SECURITY ASPECTS

INTERNET PROTOCOL V6

- ICMP?
  - Bekannt aus der letzten Vorlesung
    - IPv6 benötigt zwingend ICMPv6
  - Bekannt von IPv4
    - Blockiere allen IPv4-ICMP-Traffic am Perimeter

INTERNET PROTOCOL V6

- Gefahren durch ICMPv6
  - ICMPv6-Pakete können (Teile der) fehlerverursachenden Pakete beinhalten

    => Covert-Chanel Alarm!
    - Firewalls sollten in Fehlerpakete schauen und Legitimität enthaltener Pakete prüfen

      => Verwendete IP-Adressen im Paket

      => Flussrichtung Fehlermeldung / Paket im Bezug auf IP-Adressen
    - Denial-of-Service gegen Router durch manipulierte Pakete
      - Router bastelt nur noch ICMPv6-Fehlerpakete zusammen
      - Kann verhindert werden durch einen Threshold, wie oft der Router ICMP-Pakete behandeln soll
        - “ipv6 icmp error-interval *milliseconds*”

UNIVERSITÄT BONN

ICMPV6 INTERNER DATENVERKEHR

- Verdächtige ICMPv6-Pakete (Was sollte man filtern?)
  - Ungenutzte ICMPv6-Typen (in produktiven Umgebungen)
    - Unallocated error messages: Type 5–99 and type 102–126
    - Unallocated informational messages: Type 155–199 and type 202–254
    - Experimental messages: Type 100, 101, 200, 201
    - Extension type numbers: Type 127, 255
  - Abhängig von der Funktionalität, die im Netzwerk benötigt wird:.
    - Router Renumbering (ICMPv6 Typ 138)
      - Sollte auch bei internem Datenverkehr überwacht werden
        - Auf keinem Fall aus dem Internet in das lokale Netz erlauben
        - Bei unterschiedlichen Sites (großer Unternehmensnetzwerke) auch nicht von fremden Sites erlauben
    - Node Information Query messages (ICMPv6 Typen 139 und 140)

UNIVERSITÄT BONN

ICMPV6 EXTERNER DATENVERKEHR

- ICMPv6 Echo Requests (Ping) aus fremden Netzen
  - Können zum Ausspionieren der Geräte im lokalen Netzwerk verwendet werden
- Welche ICMPv6-Pakete müssen mindestens erlaubt werden?
  - Type 1: Destination Unreachable
  - Type 2: Packet Too Big—PMTUD
  - Type 3: Time Exceeded
  - Type 4: Parameter Problem
- Was kann erlaubt werden?
  - ICMPv6 Echo Responses (Ping)
  - ICMPv6 Echo Requests (bei Bedarf, evtl. für einzelne Hosts / Server)

ICMPV6 BEST PRACTICES

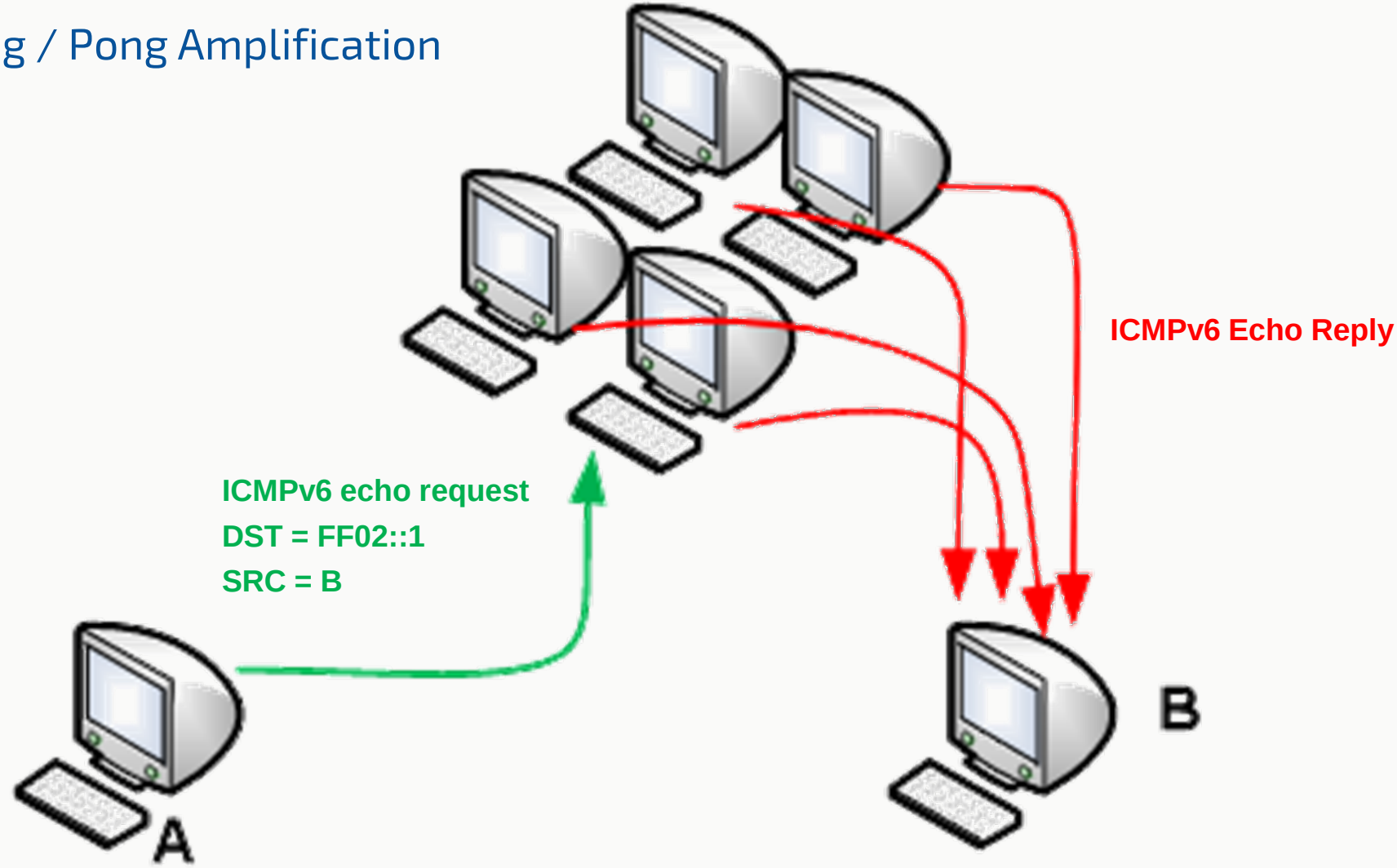
- Firewall / Paketfilter
  - Erlaube nur ICMPv6-Pakettypen, die explizit genutzt / benötigt werden
    - Statt: Verbiete einzelne Typen auf Basis des Missbrauchspotentials
  - RFC 4890: „Recommendations for Filtering ICMPv6 Messages in Firewalls“
    - Enthält ein Bash-Script als Beispiel für ip6tables

IPV6 MULTICAST VERWUNDBARKEITEN

- Reconnaisance
  - Multicastgruppen
    - All Nodes
    - All Routers
    - All DHCPv6 Servers
  - Mögliche Ziele des Angreifers:
    - Zugang zum DHCP-Server erlaubt Blick in die Logs
      - Der Angreifer muss dank Multicast nicht einmal wissen, welche Server DHCP-Server sind!
  - Amplification Attacks mittels Multicast
    - Beispiel: Gespoofte Absenderadresse an Multicast-Gruppe
    - Gut daran: ICMPv6-Error-Nachrichten dürfen nicht als Antwort auf Multicast-Pakete gesendet werden (RFC 2463)!

IPV6 MULTICAST VERWUNDBARKEITEN

- Ping / Pong Amplification



IPV6 MULTICAST VERWUNDBARKEITEN

- Auch möglich: Multicast-Adresse als Absender!
  - Hält alle Hosts gleichermaßen auf Trab!
  - ICMPv6-Error-Nachrichten dürfen nicht an eine multicast-Adresse gesendet werden (RFC 4443)!
  - Lösungen:
    - Verbiete am Perimeter eingehende Pakete mit Multicast-Adressen (global und site-local) und erlaube lediglich link-locale Pakete

UNIVERSITÄT BONN

IPV6 EXTENSION HEADER

- Regeln für Extension Header:
  - Jeder Header (Ausnahme nur für Destination-Options-Header) soll nur einmal in einem Paket enthalten sein
  - Hop-by-Hop-Header sollte der erste Header in der Header-Kette sein
  - Destination-Options-Header soll höchstens zweimal in der Kette sein
    - höchstens einmal vor einem Routing-Header
    - höchstens einmal vor dem Header der darüberliegenden Schicht
  - Der Destination-Options-Header sollte der letzte Header in der Liste sein (vor dem Routing-Header und dem Header der überliegenden Schicht)
  - Der Fragment-Header soll höchstens einmal vorkommen und nicht gemeinsam mit dem Jumbo-Payload Hop-by-Hop-Header

IPV6 EXTENSION HEADER

- RFC 2460 definiert eine Ordnung der IPv6-Header wie folgt:
  - IPv6-Header
  - Hop-by-Hop Options-Header
  - Destination-Options-Header
  - Routing-Header
  - Fragment-Header
  - Authentication-Header
  - Encapsulating-Security-Payload-Header
  - Destination-Options-Header
  - Upper-Layer-Header

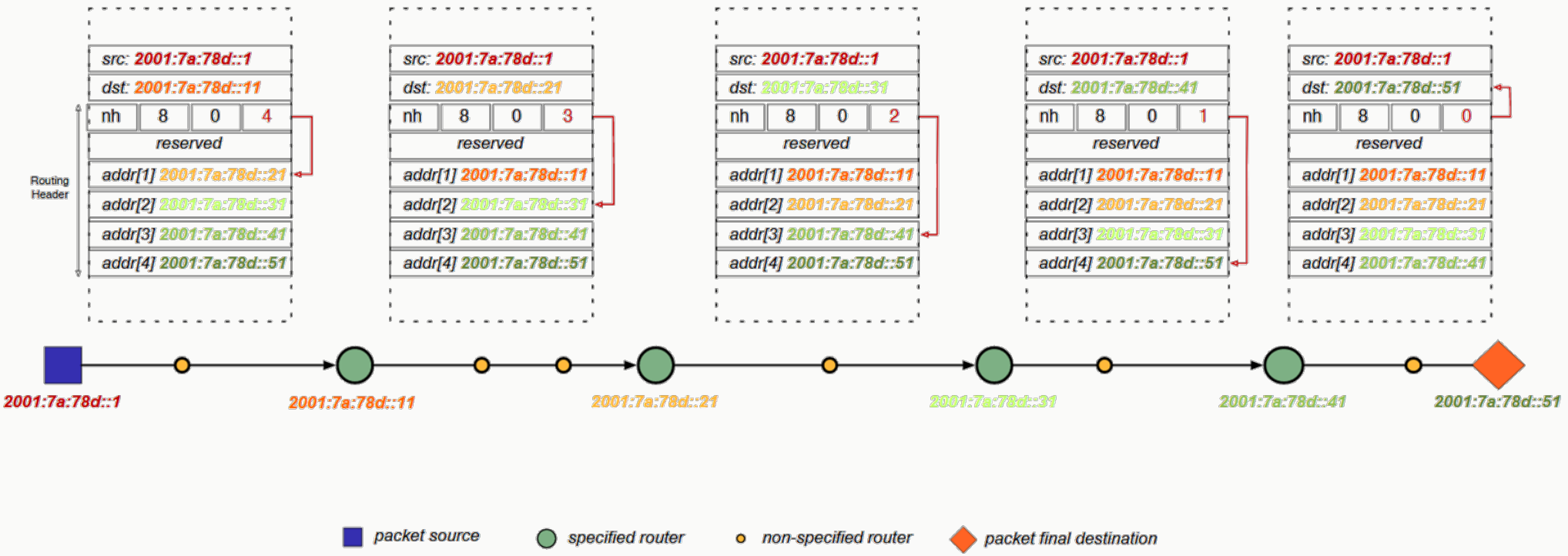
ROUTER ALERT ANGRIFF

- Als Teil der Hop-by-Hop-Header gibt es die Router-Alert-Option
  - Hinweise an den Router: Berücksichtige alle Header des Pakets
- Wird die Option unberechtigter Weise genutzt, wird unangemessener Aufwand beim Router verursacht
  - Router haben nur eine gewisse Anzahl ungenutzter CPU-Zyklen für die Verarbeitung von Ausnahmen
  - Angreifer können durch viele Pakete mit Router-Alert-Option diese Zyklen in Anspruch nehmen und wichtige andere Ausnahmebehandlung unterdrücken / verzögern

ROUTING HEADER

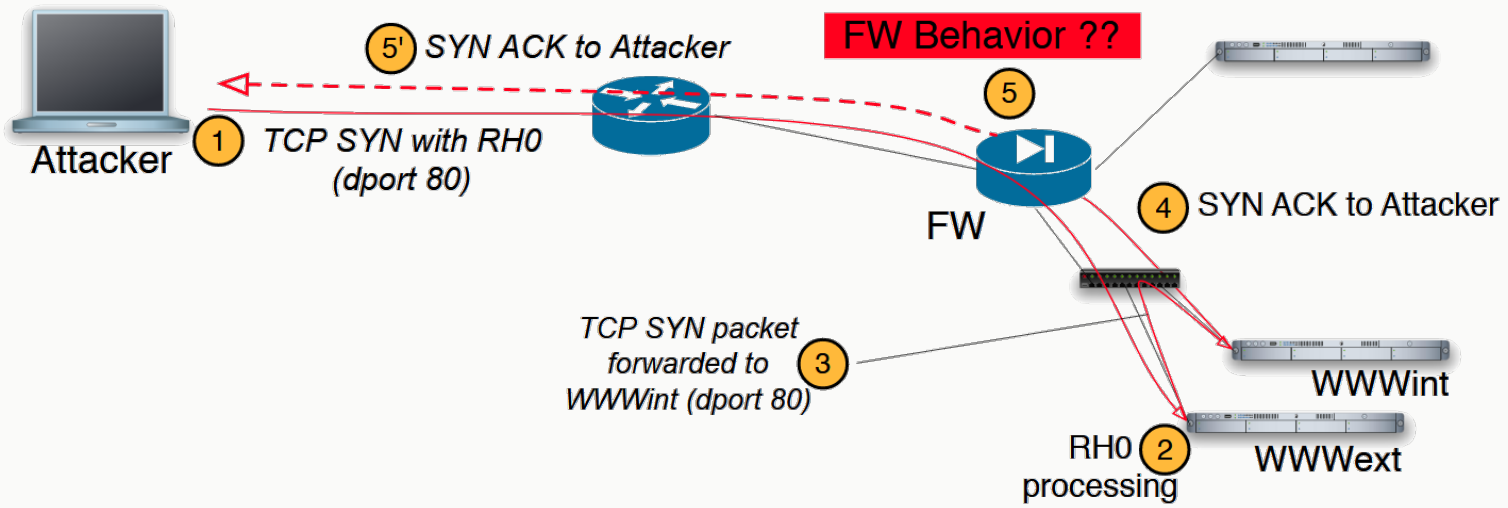
- RFC 2460 definiert, dass Router und Hosts existierende Routing-Header auswerten müssen
  - Es gibt zwei unterschiedliche Typen
    - RH 0: (Source Routing Header) enthält eine Liste von Hops, die ein Paket besuchen soll
    - RH 2: (Mobile IPv6 Header) erlaubt eine Zuordnung mobiler Hosts zu einem Heimnetz
  - Wie funktioniert der RH 0
    - Adresse des nächsten gewählten Hops steht in der Destination-Address des IPv6-Headers
    - Jeder (Zwischen-)Empfänger prüft, ob ein weiterer Hop nach ihm vorgesehen ist
    - Zwischenempfänger ersetzt die Destination-Address des IPv6-Headers mit der Adresse des nächsten Empfängers

ROUTING HEADER TYP 0



ROUTING HEADER

- Angriff mit RH 0 unter Umgehung der Firewall
  - Adresse des nächsten gewählten Hops ist die eines erreichbaren Hosts in einem Netzwerk
  - Next-Hop im Routing-Header zeigt auf einen Rechner, der eigentlich von der Firewall geschützt ist
  - Das Paket erreicht den Zielhost trotz vorgeschalteter Firewall

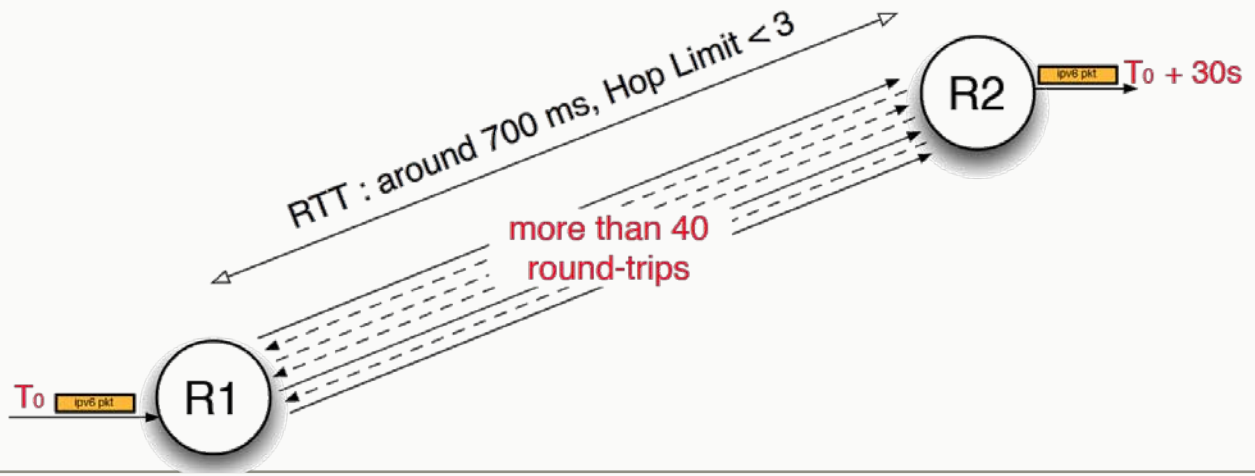


ROUTING HEADER

- Amplification DoS Angriff mit RH 0
  - Erstelle Pakte mit einer Liste zweier sich wiederholender Hops

```
>>> addr1 = '2001:4830:ff:12ea::2'
>>> addr2 = '2001:360:1:10::2'
>>> zz=time.time();                                                    \
    a=sr1(IPv6(dst=addr2, hlim=255)/                                   \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/              \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);       \
    print "%.2f seconds" % (time.time() - zz)

>>>
```



UNIVERSITÄT BONN

IPV6 FRAGMENTATION

- Wie in IPv4: Fragmentierung kann schadhafte Pakete durch eine Firewall schleusen
- Fragmentation DoS
  - Erstelle Pakete mit n Fragmenten
  - Sende für jedes Paket n-1 Fragmente an den Empfänger
  - Empfänger wartet auf n. Fragment, hält andere Fragmente im Speicher
    - Default Timeout für fragmentierte Pakete: 60 Sekunden
  - Ressourcen beim Empfänger werden blockiert

    => kann auch eine Firewall sein, wenn sie einen Datenstrom defragmentiert
- Vorteil von IPv6 (im Vergleich zu IPv4)
  - IPv6-Fragmentierung wird nicht von Routern interpretiert
  - Beschränkung auf Hosts als mögliches Ziel

RECONNAISSANCE IN IPV6-NETZEN

- Reconnaissance als Phase einer Intrusion-Killchain
  - Erstellen eines Überblicks über den Aufbau eines Netzwerks
    - Router
    - DHCP-Server
    - Domänencontroller
    - Webserver
    - Arbeitsplatz-Rechner
    - Drucker
    - …
  - IPv4-Enumeration über Ping, Syn-Scan, … mit OS-Detection
  - Adressraum bei IPv6 ist zu groß, um alle Adressen zu testen

RECONNAISSANCE IN IPV6-NETZEN

- Finden von Hosts mittels Multicast-Adressen
  - Von den vielen Multicast-Adressen eignen sich nur wenige zum Auffinden anderer Hosts im selben Subnetz
    - FF02::1 Link-lokal alle Nodes im Subnetz
    - FF02::2 Link-lokal alle Router im Subnetz
    - FF02::F Link-lokal alle UPnP-Geräte
    - FF02::101 link-lokal alle NTP-Zeitserver
  - Welche Gruppen meist nicht funktionieren
    - FF05::1 Site-lokal alle Nodes (sollte eingehend vom Router geblockt werden)
    - FF05::2 Site-lokal alle Router (sollte eingehend vom Router geblockt werden)

UNIVERSITÄT BONN

OS-DETECTION MIT IPV6-HEADERN

- Hop-Limit in unterschiedlichen Betriebssystemen

System	Value
Windows XP SP3	128
Linux 3.2.0	64
Windows 8	128
OpenBSD 5.4	64
Solaris 11.1	255

OS-DETECTION MIT IPV6-HEADERN

- Next-Header-Wert: ICMP v4 in IPv6-Paket

System	Answer
Windows XP SP3	ICMPv6 Parameter Problem
Linux 3.2.0	ICMPv6 Parameter Problem
Windows 8	ICMPv6 Parameter Problem
OpenBSD 5.4	No Reply
Solaris 11.1	ICMPv6 Parameter Problem

OS-DETECTION MIT IPV6-HEADERN

- Next-Header IPv4

System	Answer
Windows XP SP3	ICMPv6 Parameter Problem
Linux 3.2.0	ICMPv6 Parameter Problem
Windows 8	No reply
OpenBSD 5.4	No reply
Solaris 11.1	ICMPv6 Parameter Problem

UNIVERSITÄT BONN

OS-DETECTION MIT IPV6-HEADERN

- Ungültiger Authentication-Header

System	Answer
Windows XP SP3	No reply
Linux 3.2.0	ICMPv6 Parameter Problem
Windows 8	No reply
OpenBSD 5.4	No reply
Solaris 11.1	No reply

WERKZEUG ZUR PAKETMANIPULATION

- Scapy
  - Werkzeug zum Erzeugen von Netzwerkpaketen
  - Werkzeug zum Sniffen von Antwortpaketen
  - Intuitive Bedienung
    - Python-Shell / IPython
    - Sukzessive Angabe von Headern und Payload

```
>>> IPv6()
<IPv6  |>
>>> IPv6()/TCP()
<IPv6  nh=TCP |<TCP  |>>
>>> IPv6()/TCP()/"GET / HTTP/1.0\r\n\r\n"
<IPv6  nh=TCP |<TCP  |<Raw  load='GET / HTTP/1.0\r\n\r\n' |>>>
```

SCAPY - KOMMANDOÜBERSICHT

- Erstellen des Pakets durch Aneinanderreihung von Protokoll-Header + Payload
  - IP()
  - IP()/ICMP()
  - IPv6(dst="google.com")/TCP()
  - IPv6(dst=DST)/ICMPv6EchoRequest()
  - IPv6(dst=NDST)/IPv6ExtHdrRouting(addresses=[DST])/ICMPv6EchoRequest()
- Senden von Paketen (Unterschied Layer 2 & Layer 3)
  - sendp() / send() - Paket senden
  - srp() / sr() - Paket senden und Antworten empfangen
  - srp1() / sr1() - Paket senden und erste Antwort empfangen
  - srploop() / srloop() - Regelmäßiger Versand eines Pakets
  - srpflood() / srflood() - Fluten des Netzwerks mit einem Paket

SCAPY – ICMPV6 ECHO REQUEST

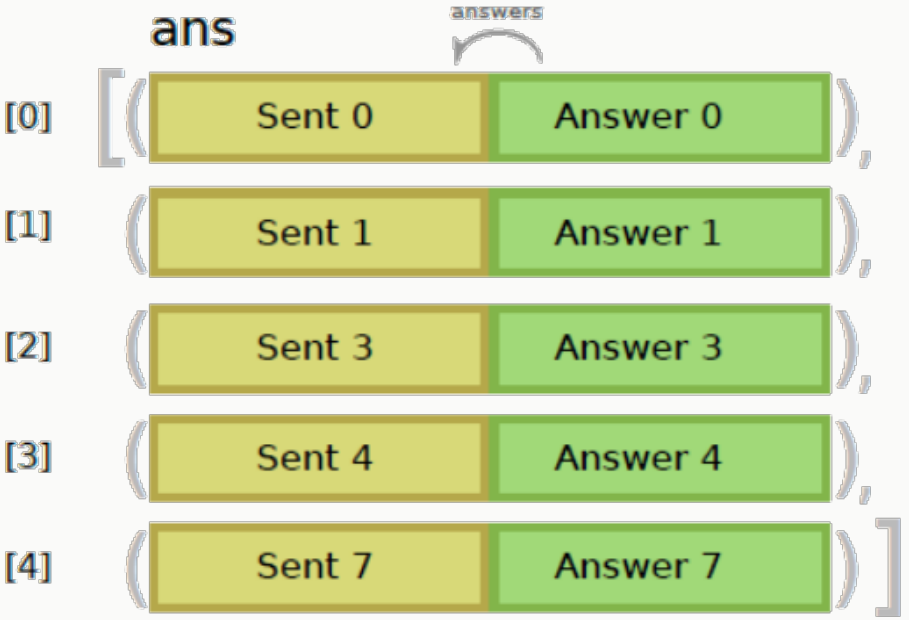
```
>>> p=IPv6(dst="google.com")/ICMPv6EchoRequest()
>>> p.show()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= ICMPv6
  hlim= 64
  src= 2003:f8:5720:2800:8fb2:c43d:e5e8:1244
  dst= Net6('google.com')
###[ ICMPv6 Echo Request ]###
     type= Echo Request
     code= 0
     cksum= None
     id= 0x0
     seq= 0x0
     data= ''
```

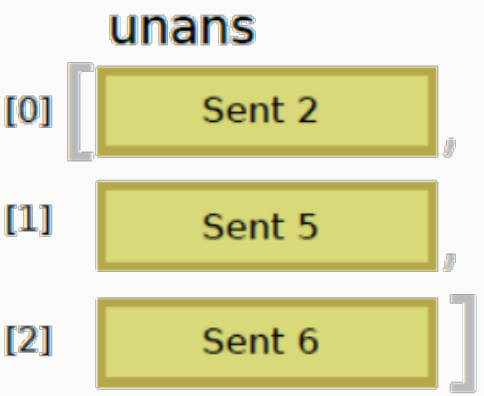
SCAPY – ICMPV6 ECHO RESPONSE

```
>>> sr(p)
Begin emission:
.Finished sending 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
(<Results: TCP:0 UDP:0 ICMP:0 Other:1>, <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> ans,unans = sr(p)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> ans.show()
0000 IPv6 / ICMPv6 Echo Request (id: 0x0 seq: 0x0) ==> IPv6 / ICMPv6 Echo Reply (id: 0x0 seq: 0x0)
```

SCAPY - KOMMANDOÜBERSICHT

>>> ans, unans = sr([p1,...,p8])





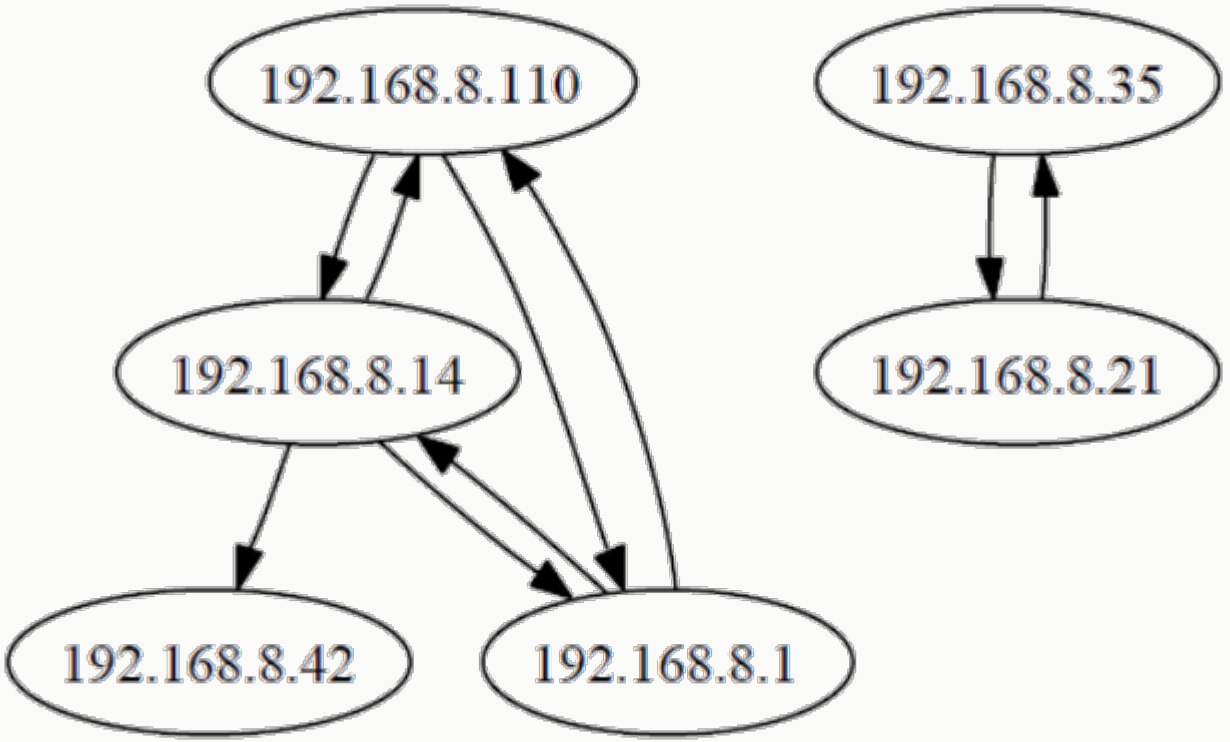

```
>>> ans[2]
(Sent 3, Answer 3)
>>> ans[2][1]
Answer 3
>>> unans[2]
Sent 6
```

SCAPY - KOMMANDOÜBERSICHT

- Higher-Level-Funktionalität
  - traceroute
  - ARPing
- Funktionen zur weiteren Analyse von Paketen
  - plot()
  - pdfdump()
  - conversations()
  - make_table()

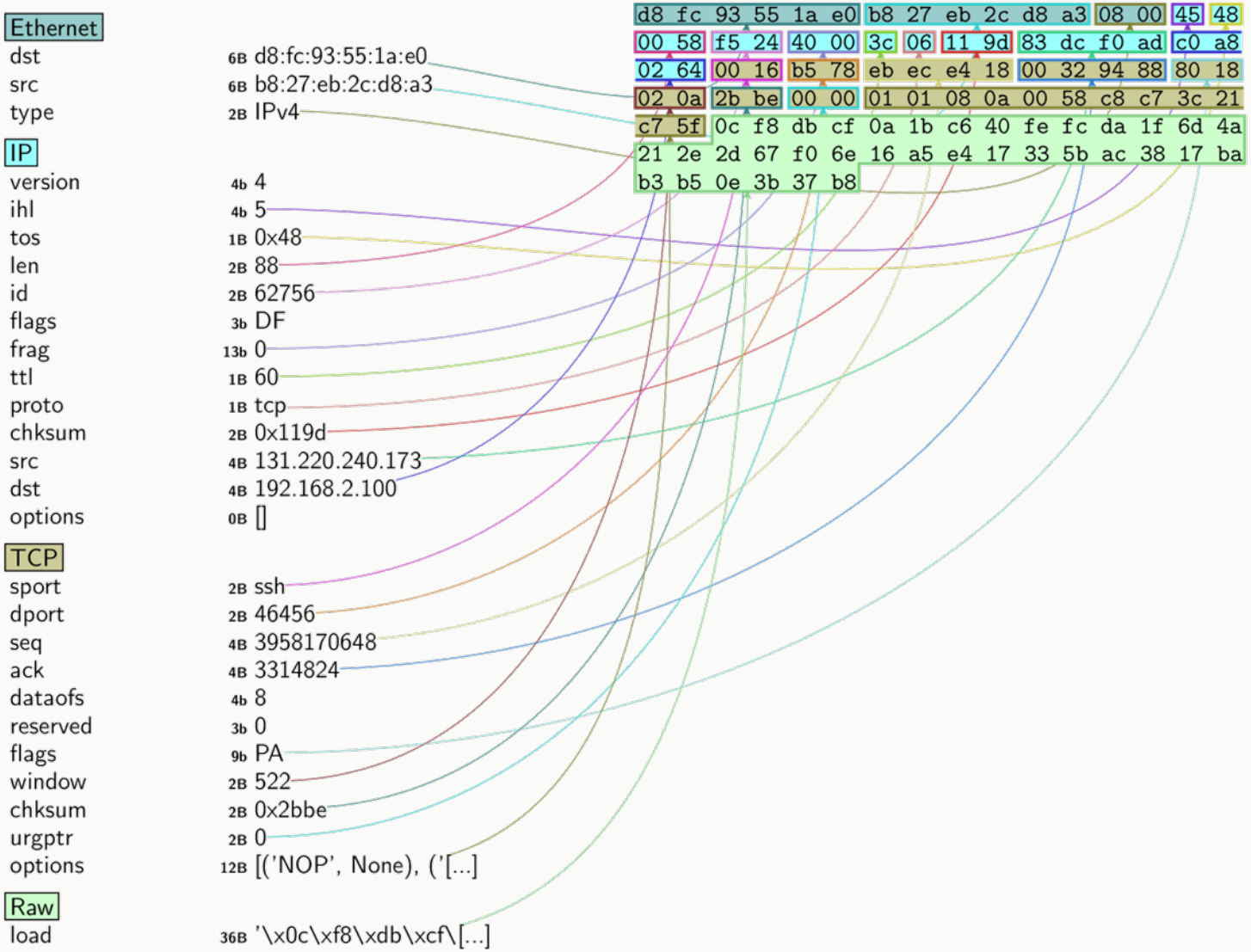
SCAPY - KOMMANDOÜBERSICHT

```
>>> p = sniff(count=30)      // Count-Parameter ist optional
>>> p.conversations()
```



SCAPY - KOMMANDOÜBERSICHT

```
>>> p = sniff(count=1)
>>> p.pdfdump()
```



SCAPY - PAKETSNIFFER

- Mitlesen von Traffic (Sniffen)

```
>>> s=sniff(count=10)
>>> s
<Sniffed: TCP:5 UDP:3 ICMP:0 Other:2>
>>> s.show()
0000 Ether / IP / TCP 10.151.124.2:46456 > 131.220.240.173:ssh A
0001 Ether / IP / UDP 131.220.240.144:54915 > 131.220.240.191:54915 / Raw
0002 Ether / IP / TCP 131.220.240.173:ssh > 10.151.124.2:46456 PA / Raw
0003 Ether / IP / TCP 10.151.124.2:46456 > 131.220.240.173:ssh A
0004 802.3 00:78:88:75:d9:08 > 01:80:c2:00:00:00 / LLC / STP / Raw / Padding
0005 Ether / IP / UDP 131.220.240.144:54915 > 131.220.240.191:54915 / Raw
0006 Ether / ARP who has 131.220.240.142 says 131.220.240.190 / Padding
0007 Ether / IP / TCP 131.220.240.173:ssh > 10.151.124.2:46456 PA / Raw
0008 Ether / IP / TCP 10.151.124.2:46456 > 131.220.240.173:ssh A
0009 Ether / IP / UDP 131.220.240.144:54915 > 131.220.240.191:54915 / Raw
```

UNIVERSITÄT BONN

ENDE

Vielen Dank für die Aufmerksamkeit!

Fragen?

Nächste Vorlesung:

- Montag, 19. Juni 2023

Nächste Übung:

- Dienstag, 13. Juni 2023 – 16 Uhr
- Abgabe des Übungszettels 7 bis morgen – 16 Uhr