BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Methoden und Techniken für den Test von IPv6 Netzwerken

Thomas Scheffler

IPv6 Kongress 2011
Frankfurt/Main, 12./13 Mai 2011

Übersicht

- Prinzipieller Ansätze für Netzwerktests
  - Teststufen
  - Testkriterien
- Ablauf einer Netzwerküberprüfung
  - Wireshark
  - Nmap
  - Scapy
- Kurzvorstellung Scapy-GUI
- Fazit

Prinzipielle Ansätze für Netzwerktests

Teststufen in der Entwicklung und Evaluation eines Netzwerks:

- Komponententests
- Integrationstests
- Systemtests
- Abnahmetests
- ...

Prinzipielle Ansätze für Netzwerktests

Testkriterien:

- Funktionale Tests
- Interoperabilitätstests
- Performance Tests
- Sicherheitstests
- Ermittlung von Fehlerzuständen und -quellen
- ...

Ablauf einer Netzwerksüberprüfung

Ablauf der Überprüfung eines Netzwerks

- **Analyse des Netzwerks**
  - Ermittlung der im Netz bekannten Hostsysteme
  - Ermittlung der im Netz erreichbaren Systeme
  - Ermittlung von aktiver Services und Verbindungen
  - Aufnahme der Netzparameter (Verzögerung, Auslastung, etc.)

- **Analyse der Hostsysteme (Sicherheitstests)**
  - Art und Version des Betriebssystems
  - Art und Version der installierten Dienste

Werkzeuge für die IPv6 Netzwerkanalyse

- **ping6, traceroute6**
- **Wireshark**
- **Nmap**
  - Security Scanner
  - http://nmap.org/
- **THC-IPv6-Toolkit**
  - Sammlung von IPv6 Sicherheitswerkzeugen
  - www.**thc**.org/**thc-ipv6**/
- **Scapy**
  - Python-basiertes Paketcrafting-Tool
  - http://www.secdev.org/projects/scapy/

Scanning eines Netzes mit Nmap

Nmap ist IPv6-fähig mit den folgenden Einschränkungen:

- Nmap kann keine IPv6 Netze scannen – nur einzelne Hosts
- Nmap unterstützt über IPv6 die folgenden Scan-Typen:
  - ‘Connect’ -Scans: **`nmap -6 –sT`** benutzt die Betriebssystemfunktion um eine Verbindung zum Ziel aufzubauen, es können nur die Zustände *open* und *closed* unterschieden werden (default)
  - ‘Ping ‘-Scans: **`nmap -6 –sT`** ermittelt die prinzipielle Erreichbarkeit eines Hosts
  - ‘List ‘-Scans: **`nmap -6 –sL`** ermittelt die im DNS gelisteten Hosts
    List-Scan liefert im lokalen Netz keine neuen Informationen!

Analyse eines Netzes mit Wireshark

Wireshark

- Protokollanalyse
- Welche Protokolle sind in welcher Verteilung im Netz aktiv?
  - Menü: `Statistics > Protocol Hierarchy`

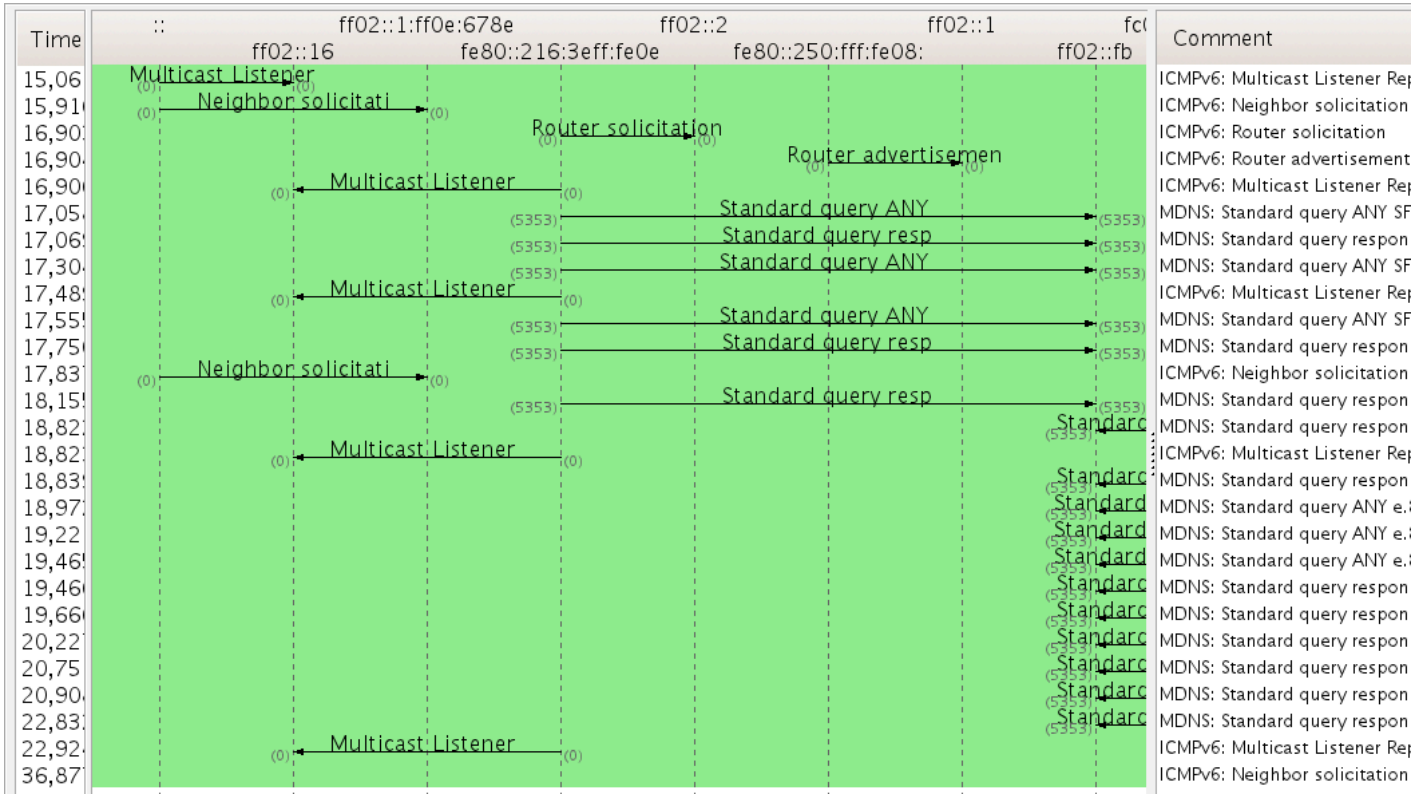
Display filter: ipv6

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100,00 %	27	100,00 %	6079	0,002	0	0	0,000
Ethernet	100,00 %	27	100,00 %	6079	0,002	0	0	0,000
Internet Protocol Version 6	100,00 %	27	100,00 %	6079	0,002	0	0	0,000
Internet Control Message Protocol v6	37,04 %	10	14,81 %	900	0,000	10	900	0,000
User Datagram Protocol	62,96 %	17	85,19 %	5179	0,002	0	0	0,000
Domain Name Service	62,96 %	17	85,19 %	5179	0,002	17	5179	0,002

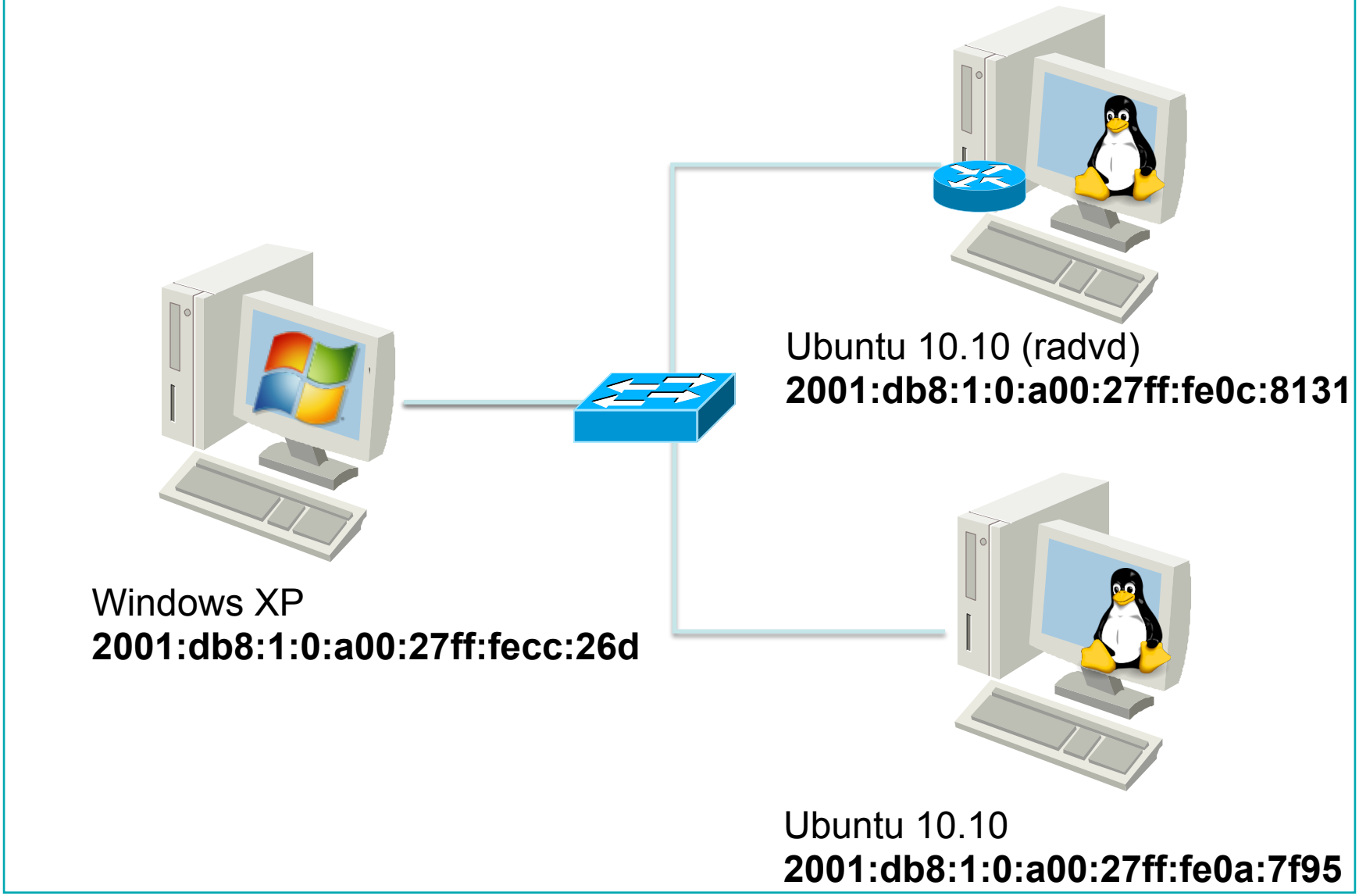
- Welche Endpunkte sind im Netz aktiv?
  - Menü: `Statistics > Endpoint List > IPv6`
- Welche Endpunkte sprechen miteinander?
  - Menü: `Statistics > Conversations List > IPv6`

Analyse eines Netzes mit Wireshark

- Graphische Darstellung der Kommunikationsabläufe im Netz
  - Menü: `Statistics > Flow Graph`



Testnetzwerk



Ermittlung aller IPv6 Hosts im Sub-Netz

Mehrere denkbare Strategien:

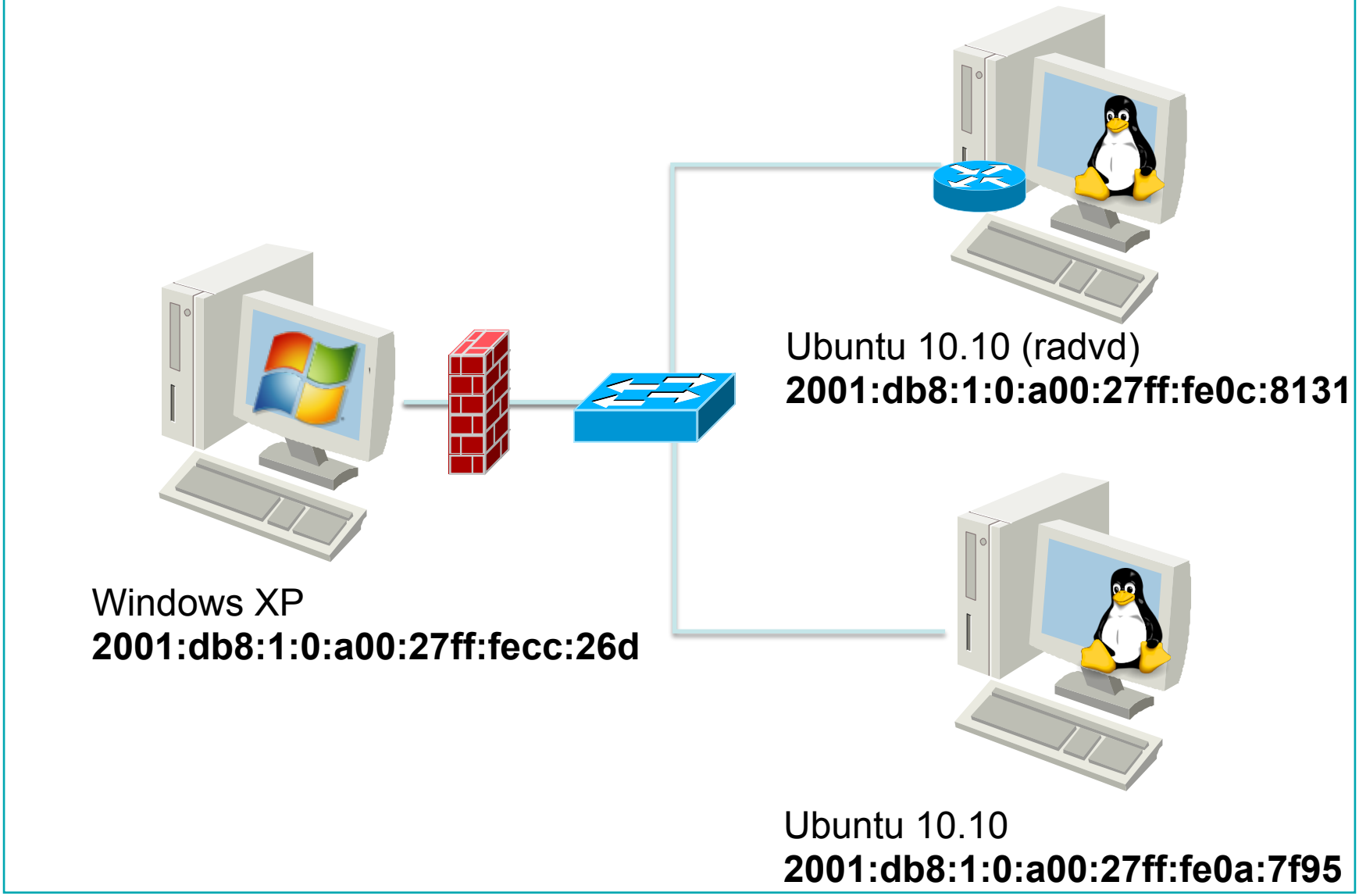
1. Verbreiten eines neuen Präfixes für die IPv6-Address-Autokonfiguration
   - **Idee:**
     - Alle Hosts müssen eine Duplicate Address Detection Nachricht senden, die abgefangen und ausgewertet werden kann.
   - **Nachteile:**
     - Beeinflussung der Netzkonfiguration der Hosts
     - Präfix muss anschließend widerrufen werden

Ermittlung aller IPv6 Hosts im Sub-Netz

Mehrere denkbare Strategien (2):

2. Alive6 (Werkzeug aus dem THC-Toolkit von Marc Heuse)
   - **Idee:**
     - Sendet eine ICMP-EchoRequest Nachricht an die ‚All-Nodes '-Multicastgruppe **FF02::1** (+ einige weitere Tricks)

Testnetzwerk



Ermittlung aller IPv6 Hosts im Sub-Netz

Mehrere denkbare Strategien (2):

2. Alive6 (Werkzeug aus dem THC-Toolkit von Marc Heuse)
   - **Idee:**
     - Sendet eine ICMP-EchoRequest Nachricht an die ‚All-Nodes '-Multicastgruppe **FF02::1**
   - **Nachteile:**
     - Funktioniert nicht, wenn die Firewall ICMP Echo-Requests blockiert (default für Windows XP)
     - RFC 4443 spezifiziert die Antwort auf einen ICMP-EchoRequest auf eine Multicastgruppe als optional (SHOULD), d.h. es ist möglich, dass einzelne Implementierungen diese nicht unterstützen.

Generierung von IPv6-Paketen mit Scapy

Überblick – Scapy

- IPv6-Paket-Generator in Gestalt eines Kommandozeileninterpreters
- Plattformübergreifend lauffähig (Python)
- Programmierkenntnisse sind nicht zwingend nötig, aber vorteilhaft

```
Datei  Bearbeiten  Ansicht  Terminal  Hilfe
tobias@backbuntu:~$ sudo scapy
[sudo] password for tobias:
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
Welcome to Scapy (2.1.0)
>>> sr1(IPv6(dst="fd11:100::1")/ICMPv6EchoRequest())
Begin emission:
.Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
<IPv6  version=6L tc=0L fl=0L plen=8 nh=ICMPv6 hlim=255 src=fd11:100::
e793 |<ICMPv6EchoReply  type=Echo Reply code=0 cksum=0x4aa2 id=0x0 seq
>>>
```

Generieren von IPv6-Paketen - ICMPv6 Ping

- Alle Anweisungen können in eine Kommandozeile geschrieben werden.
- Scapy ergänzt, soweit möglich, nicht angegebene notwendige Paketparameter (z.B. Source-Adresse, Prüfsummen, etc.).
- Direkte Einbettung und Weiterverarbeitung als Python Script ist möglich.

Beispiel - ICMP-EchoRequest (Ping):

```
>>> sr1(IPv6(dst='fd11:100::1')/ICMPv6EchoRequest())
```

Generieren von IPv6-Paketen - ICMPv6 Ping

Etwas genauer erklärt:

- `sr1()` – sendet Pakete auf Layer 3 Ebene

```
>>> help(sr1)
sr1 … send packets at layer 3 and return only the
1st answer
```

- `IPv6()` – erzeugt IPv6-Header, die Zieladresse ist zwingend benötigter Übergabeparameter

- `ICMPv6EchoRequest()` – erzeugt ICMPv6-Paketstruktur bei welcher der ICMP-Typ bereits dem des ICMP *Echo-Requests* entspricht (128)

Paketgenerierung *step-by-step*

- Für komplexe Pakete empfiehlt es sich, die einzelnen Netzwerklayer jeweils nacheinander zu generieren und über Variablen verfügbar zu machen.
  - Komplexe Pakete lassen sich so Schritt-für-Schritt in der Kommandozeile erstellen.
  - Dies ist deutlich übersichtlicher und lässt sich später einfacher anpassen:

```
>>> i=IPv6()
>>> i.dst='fd11:100::1'
>>> q=ICMPv6EchoRequest()
>>> p=(i/q)
>>> sr1(p)
```

Paketgenerierung – Optionen anzeigen

- `ls()` listet alle von Scapy unterstützten Protokoll-Schichten auf.
- Um den ICMP Echo Request weiter zu manipulieren ist es vorteilhaft zu wissen welche Optionen man überhaupt zur Verfügung hat.
  - `ls(ICMPv6EchoRequest)` liefert diese manipulierbaren Felder sowie deren Defaultwerte:

```
>>> ls(ICMPv6EchoRequest)
                      type : ByteEnumField = (128)
                      code : ByteField = (0)
                      cksum : XShortField = (None)
                      id : XShortField = (0)
                      seq : XShortField = (0)
                      data : StrField = ('')
```

Pakete erzeugen - Optionen hinzufügen / ändern

- Echo Requests enthalten für gewöhnlich auch noch Daten und diese kann man nun ganz einfach hinzufügen:

```
>>> q.data='HelloWorldPingData‘
```

- Das Paket muss noch aktualisiert und gesendet werden:

```
>>> p=(i/q)
>>> sr1(p)
```

Scapy - send

- Neben `sr1()` gibt es noch weitere Möglichkeiten ein Paket zu versenden:

```
>>> send(x, inter=0, loop=0, count=None,
verbose=None, *args, **kargs)
```

sendet Pakete, wie `sr1()` auf Layer-3 Ebene, wartet jedoch keine Antwort ab.

- `send()` bietet die Möglichkeit mehrere Pakete automatisch nacheinander zu senden:

```
>>> send(IPv6(dst="fd11:100::1")/
ICMPv6EchoRequest(), inter=3, count=9)
```

Es wird alle 3 Sekunden ein 'Echo Request' gesendet, insgesamt neun mal

Scapy - sendpfast

- Scapy kann Pakete mit höherer Datenrate versenden.
- `sendpfast()` dient dem schnellen Versenden auf der Layer-2 Ebene (benutzt `tcpreplay`).

```
>>> sendpfast(Ether()/IPv6
(dst='fe80::250:fff:fe08:48C0')/ICMPv6EchoRequest
(data='Hallo Welt,*100)*10000, iface='eth0')
```

- Die Ausgabe von Scapy sieht wie folgt aus:

```
>>> sendpfast(Ether()/IPv6(dst='fe80::250:fff:fe08:48c0')/ICMPv6EchoRequest(data
='Hallo Welt'*100)*10000,iface="eth0")
sending out eth0
processing file: /tmp/scapyFiCQq2
Actual: 10000 packets (10620000 bytes) sent in 4.19 seconds
Rated: 2534606.2 bps, 19.34 Mbps, 2386.63 pps
Statistics for network device: eth0
        Attempted packets:         10000
        Successful packets:        10000
        Failed packets:            0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
>>>
```

Pakete kompilieren und anzeigen

`show()` zeigt die gesetzten Optionen des Pakets

```
>>> (IPv6(dst='ff02::1')/ICMPv6EchoRequest()).show()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= ICMPv6
  hlim= 64
  src= fe80::21a:4fff:fe48:e793
  dst= ff02::1
###[ ICMPv6 Echo Request ]###
     type= Echo Request
     code= 0
     cksum= None
     id= 0x0
     seq= 0x0
     data= ''
```

`show2()` kompiliert das Paket und berechnet so auch z.B. Länge der Payload und Prüfsummen

```
>>> (IPv6(dst='ff02::1')/ICMPv6EchoRequest()).show2()
###[ IPv6 ]###
  version= 6L
  tc= 0L
  fl= 0L
  plen= 8
  nh= ICMPv6
  hlim= 64
  src= fe80::21a:4fff:fe48:e793
  dst= ff02::1
###[ ICMPv6 Echo Request ]###
     type= Echo Request
     code= 0
     cksum= 0x4a42
     id= 0x0
     seq= 0x0
     data= ''
```

Ermittlung aktiver Hosts im Netz mit Scapy

Ermittlung aller IPv6 Hosts im Sub-Netz

Mehrere denkbare Strategien (2):

- **Idee:**
  - Sendet eine ICMP-EchoRequest Nachricht an die ‚All-Nodes '-Multicastgruppe **FF02::1**

- **Realisierung mit Scapy:**

```
>>>send(IPv6(dst='ff02::1')/ICMPv6EchoRequest())
```

Ermittlung aller IPv6 Hosts im Sub-Netz

Mehrere denkbare Strategien (3):

3. Nutzung des Multicast Listener Discovery Protokolls zum Aussenden einer allgemeinen Anfrage (General Querry)

- **Idee:**
  - Sendet eine ICMP-Multicast *Listener Querry* Nachricht an die ‚All-Nodes '-Multicastgruppe **FF02::1**
  - Für jede Multicast Gruppe wird ein Multicast *Listener Report* erzeugt ➔ Solicited Nodes Multicast Address
  - Keine weitere Beeinflussung des Netzwerkverkehrs
  - Ermittelt auch Hosts hinter Personal Firewalls
- **Nachteile:**
  - Es besteht die geringe Wahrscheinlichkeit, dass zwei oder mehrere Hosts der gleichen Solicited Nodes Multicast Gruppe angehören (ca. 0,3% für 1000 Hosts in einem Subnetz). Abhilfe schafft ggf. eine mehrfache Generierung der Requests.

Ermittlung aller IPv6 Hosts im Sub-Netz

Mehrere denkbare Strategien (3):

- **Idee:**
  - Sendet eine ICMP-Multicast *Listener Querry* Nachricht an die ‚All-Nodes '-Multicastgruppe **FF02::1**

- **Realisierung mit Scapy:**

```
>>>packet=(IPv6(dst='ff02::1', nh=58)/ICMPv6MLQuery())
>>>send(packet)
```

Ablauf der Überprüfung eines Netzwerks

- **Analyse des Netzwerks**
  - Ermittlung der im Netz bekannten Hostsysteme
  - Ermittlung der im Netz erreichbaren Systeme
  - Ermittlung von aktiver Services und Verbindungen
  - Aufnahme der Netzparameter (Verzögerung, Auslastung, etc.)

- **Analyse der Hostsysteme (Sicherheitstests)**
  - Art und Version des Betriebssystems
  - Art und Version der installierten Dienste
  - Überprüfung der Dienste auf evtl. vorhandene Schwachstellen

Werkzeuge für die Analyse von Hostsystemen

- **Nmap**
  - Security Scanner
  - http://nmap.org/

- **OpenVAS**
  - Freier Schwachststellenscanner
  - http://www.openvas.org/

Scanning eines Hosts mit Nmap

Nmap lässt sich für die Analyse von Hosts im Netzwerk nutzen:

- Ermittlung des Betriebssystems sowie der Versionsstände der installierten Dienste wird über IPv6 nur teilweise unterstützt
  `nmap -6 –sV`

```
thomas@ubuntu: ~
File  Edit  View  Search  Terminal  Help
thomas@ubuntu:~$ sudo nmap -6 -sV fe80::a00:27ff:fe0c:8131%eth0

Starting Nmap 5.21 ( http://nmap.org ) at 2011-04-17 17:48 CEST
Nmap scan report for fe80::a00:27ff:fe0c:8131
Host is up (0.0059s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.5p1 Debian 4ubuntu5 (protocol 2.0)
80/tcp open  http    Apache httpd 2.2.16 ((Ubuntu))
Service Info: OS: Linux

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.78 seconds
thomas@ubuntu:~$
```

OpenVAS Reporting (Ubuntu Host)

Security Issues for Host 2001:db8:1:0:a00:27ff:fe0c:8131

Low general/tcp
NVT: Checks for open tcp ports (OID: 1.3.6.1.4.1.25623.1.0.900239)

```
Open TCP ports are 22
```

Low ssh (22/tcp)
NVT: Services (OID: 1.3.6.1.4.1.25623.1.0.10330)

```
An ssh server is running on this port
```

Low ssh (22/tcp)
NVT: SSH Server type and version (OID: 1.3.6.1.4.1.25623.1.0.10267)

```
Remote SSH version : SSH-2.0-OpenSSH_5.5p1 Debian-4ubuntu5
Remote SSH supported authentication : publickey,password
```

Low ssh (22/tcp)
NVT: SSH Protocol Versions Supported (OID: 1.3.6.1.4.1.25623.1.0.100259)

```
Overview:
The remote SSH Server supports the following SSH Protocol Versions:
1.99
2.0
SSHv2 Fingerprint: db:1b:d1:ad:b1:11:bc:5c:27:a4:9c:27:c6:7e:35:66
Risk factor : None
```

OpenVAS Reporting (WindowsXP Host)

Results per Host

Host 2001:db8:1:0:a00:27ff:fecc:26d

Scanning of this host started at: Fri Apr 15 23:23:54 2011
Number of results: 1

Port Summary for Host 2001:db8:1:0:a00:27ff:fecc:26d

Service (Port)	Threat Level
general/tcp	Low

Security Issues for Host 2001:db8:1:0:a00:27ff:fecc:26d

Low	general/tcp
NVT: Checks for open tcp ports (OID: 1.3.6.1.4.1.25623.1.0.900239)	
`Open TCP ports are 135`	

This file was automatically generated.

Scapy GUI - Kurzvorstellung

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Graphische Oberfläche für Scapy

Motivation

- Einfache Bedienung von Scapy ohne Kommandozeileneingabe
- Flexible, einfache und schnelle Erstellung von IPv6 Paketen für Testzwecke im Netzwerk
- Lernprogramm für IPv6 und Scapy

Optionen

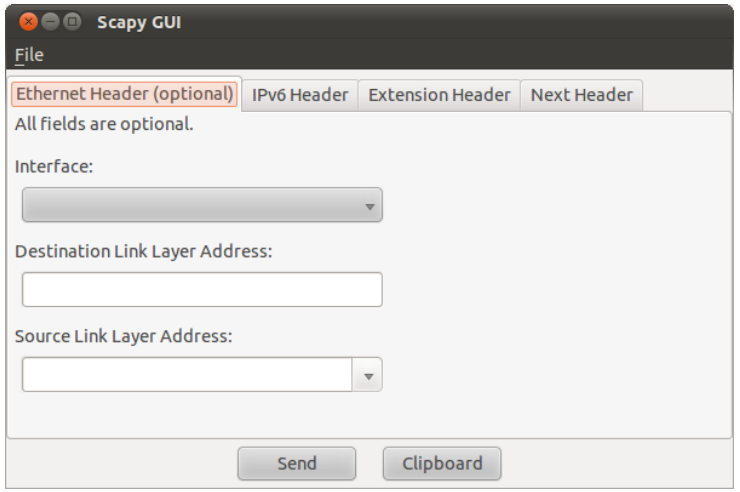
- Einbindungsmöglichkeit von Erweiterungsheadern in einfacher Form
- Auswahl verschiedener Next Header Typen und Payload
- Speichern und Laden von Pcap Dateien
- Senden von Paketen mit dem `sendp()` Befehl
- Speichen eines erstellten Sendecodes in der Zwischenablage

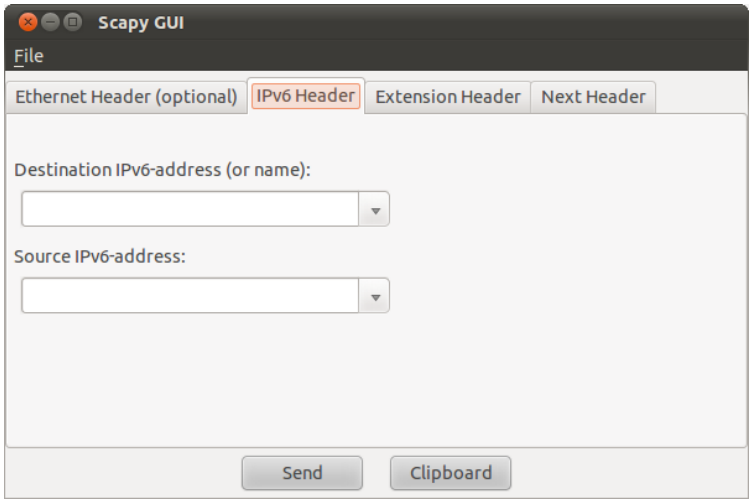
Graphische Oberfläche für Scapy

Aufbau eines Pakets

Ethernet Header (optional):

- Source und Destination Link Layer Adresse
- Interface (notwendig falls mehrere vorhanden sind)





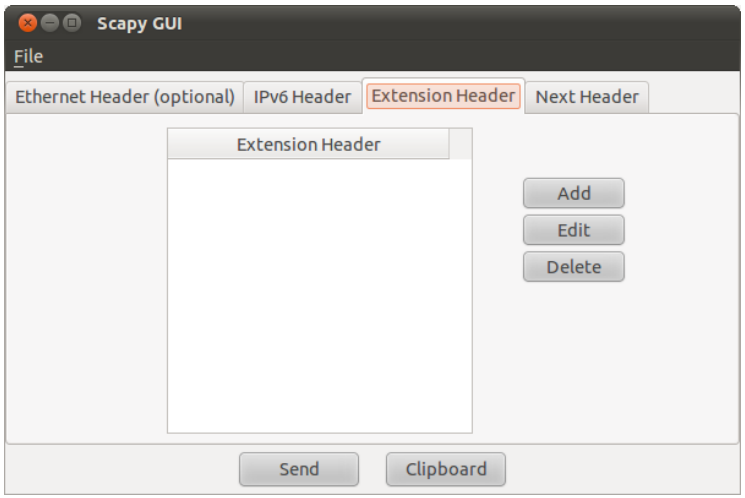
IPv6 Header:

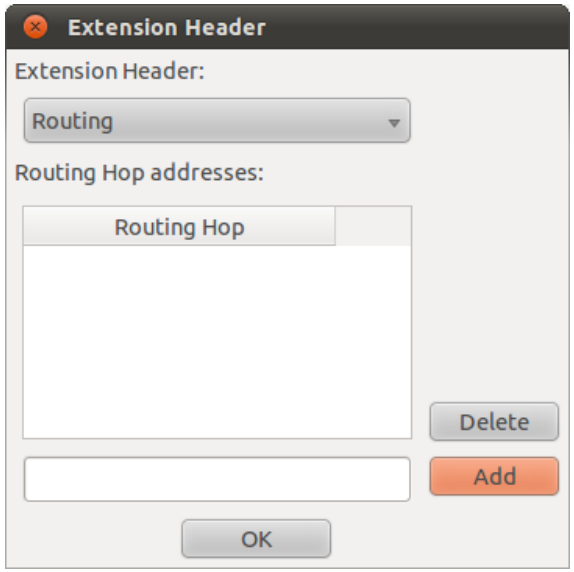
- Source und Destination IPv6 Adresse
- Auswahl von vordefinierten IPv6 Adressen mittels Drop-Down Menü

Graphische Oberfläche für Scapy

Extension Header (optional):

- Auflistung der ausgewählten Extension Header in einer Tabelle
- Bearbeiten und Löschen von vorhandenen Extension Headern



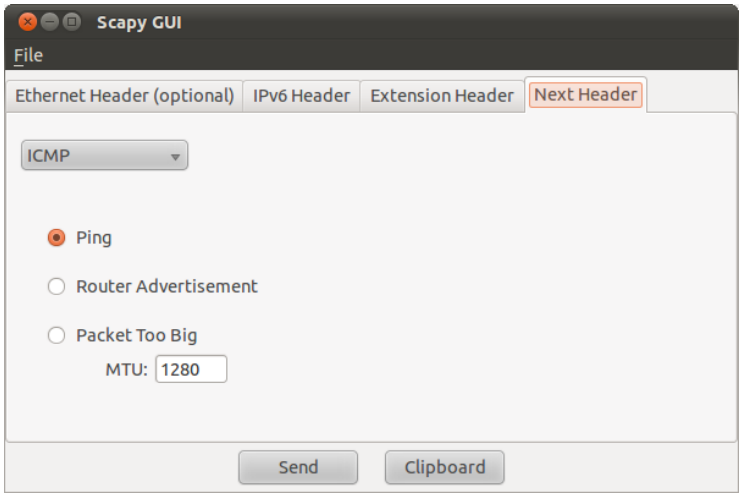


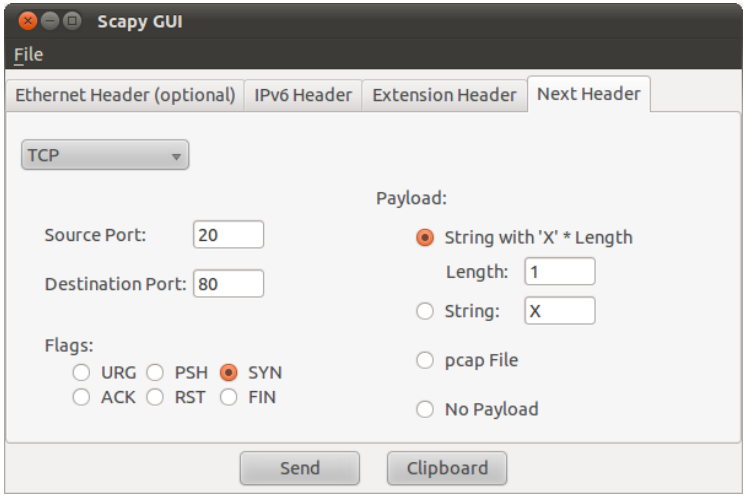
- Auswahlmöglichkeit zwischen vier Extension Header Typen:
  - Destination Options Header
  - Hop-by-Hop Options Header
  - Routing Header
  - Fragment Header

Graphische Oberfläche für Scapy

Next Header:

- Auswahl zwischen vier Next Header Typen
  - ICMP
  - TCP
  - UDP
  - No Next Header

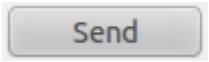


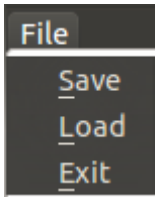


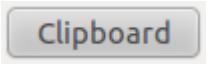
- ICMP mit drei Untertypen
- TCP und UDP mit Payload Option und Angabe von Source- und Destination-Port
- Setzen der Flags bei TCP

Graphische Oberfläche für Scapy

Weiterverarbeitung

- Senden mittels `sendp()` Befehl 

- Speichern der Paketdatenstruktur in einer Pcap-Datei 

- Speichern des Scapycodes in der Zwischenablage 

- Anzeige des Scapycodes in einer Info Message Box

Fazit

Fazit

- Die Verfügbarkeit von Werkzeugen für den Test und die Überprüfungen von IPv6 Netzwerken hat sich verbessert:
  - Funktionale Tests lassen sich auf Basis des Scapy-Paketgenerators leicht selbst durchführen.
  - Sicherheitsüberprüfungen von Hosts und Servern sind mit aktuellen Versionen von OpenVAS möglich.
  - Für Performancetests werden in der Regel spezielle Lastgeneratoren benötig.
- Die Behandlung der Besonderheiten und Eigenarten des IPv6-Protokolls ist in einigen Tools noch nicht vollständig umgesetzt.

Im Rahmen eines durch das BMBF geförderten Verbundprojekts der Beuth-Hochschule, der Uni-Potsdam und der EANTC AG werden derzeit verschiedene Werkzeuge für die Sicherheitsanalyse in IPv6 Netzwerken entwickelt.

Website: http://ipv6-ids.de/

Thomas Scheffler
Fachbereich Elektrotechnik
Beuth-Hochschule für Technik Berlin

Email: scheffler@beuth-hochschule.de
WWW: prof.beuth-hochschule.de/scheffler

Quellen

- http://www.secdev.org/projects/scapy/
(aktuelle Scapy Version)

- SECURITY POWER TOOLS: O'Reilly Media, Inc., 2007
ISBN: 0-596-00963-1, 978-0-596-00963-2

- http://www.secdev.org/projects/scapy/files/scapydoc.pdf

- http://dirk-loss.de/scapy-doc/Scapy.pdf

- http://www.packetlevel.ch/

- www.thc.org/thc-ipv6/