Scapy

On-the-fly Packet Generation

by codemonk@u-sys.org

Overview

- Repetition of network basics
- Python Basics
- Scapy Basics
- Example: SYN Scan
- Hands-on:
  - Traceroute
  - Promiscuous Scan
  - ARP Spoofing

Layers

- OSI Model
- Abstraction layers

(1) Physical
Bit

(2) Data link
Frames with
physical addressing

(3) Network
Packets and routing

(4) Transport
Connections

(5) ... (7) Application

Ethernet

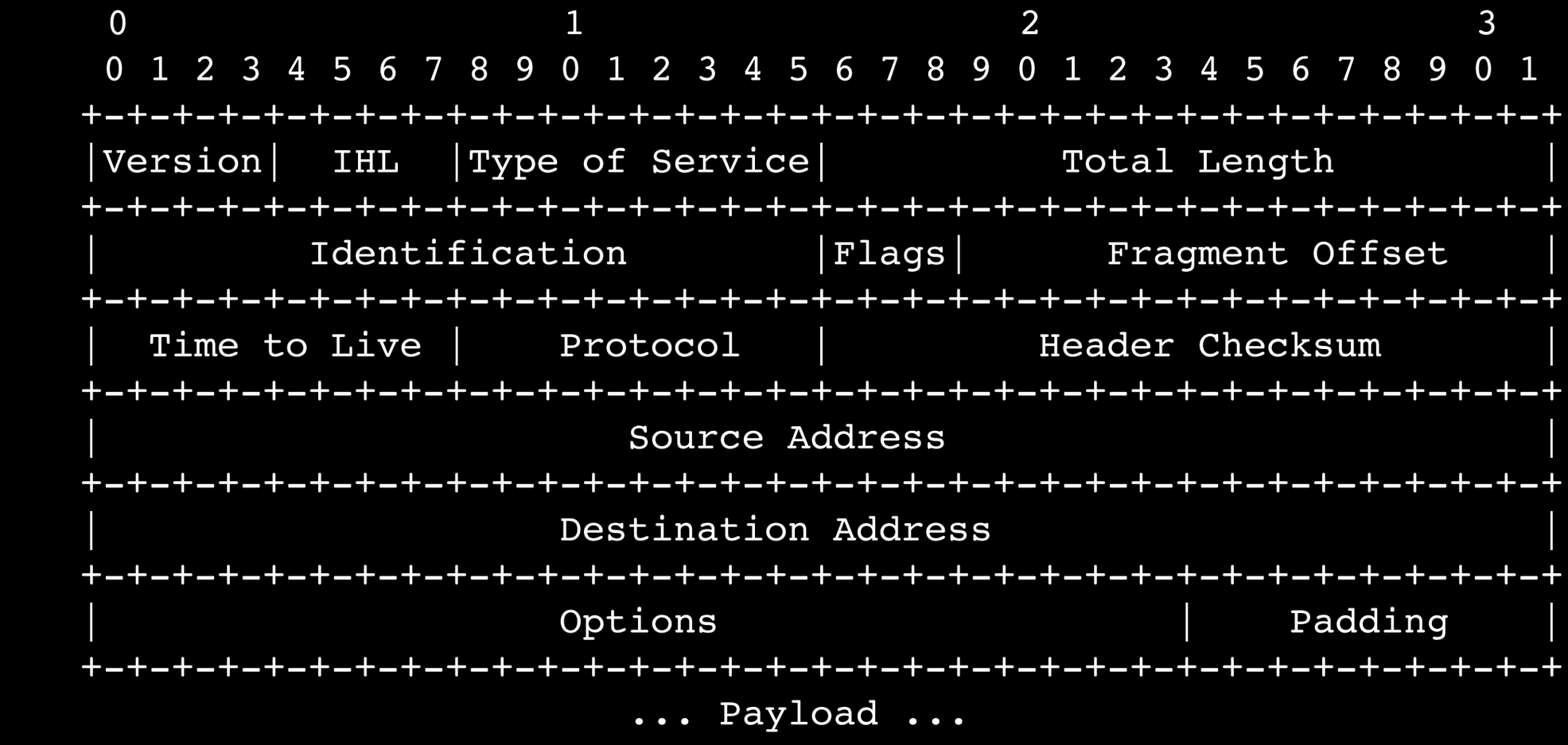
- Layer 2 protocol
- Frames
- Uses hardware/MAC addresses

Internet Protocol

- Layer 3 protocol
- Packets
- Sender and receiver identified by „IP“, e.g.

  123.45.67.89
- „IP“ can have special meaning, like 255.255.255.255 - broadcast

Internet Protocol

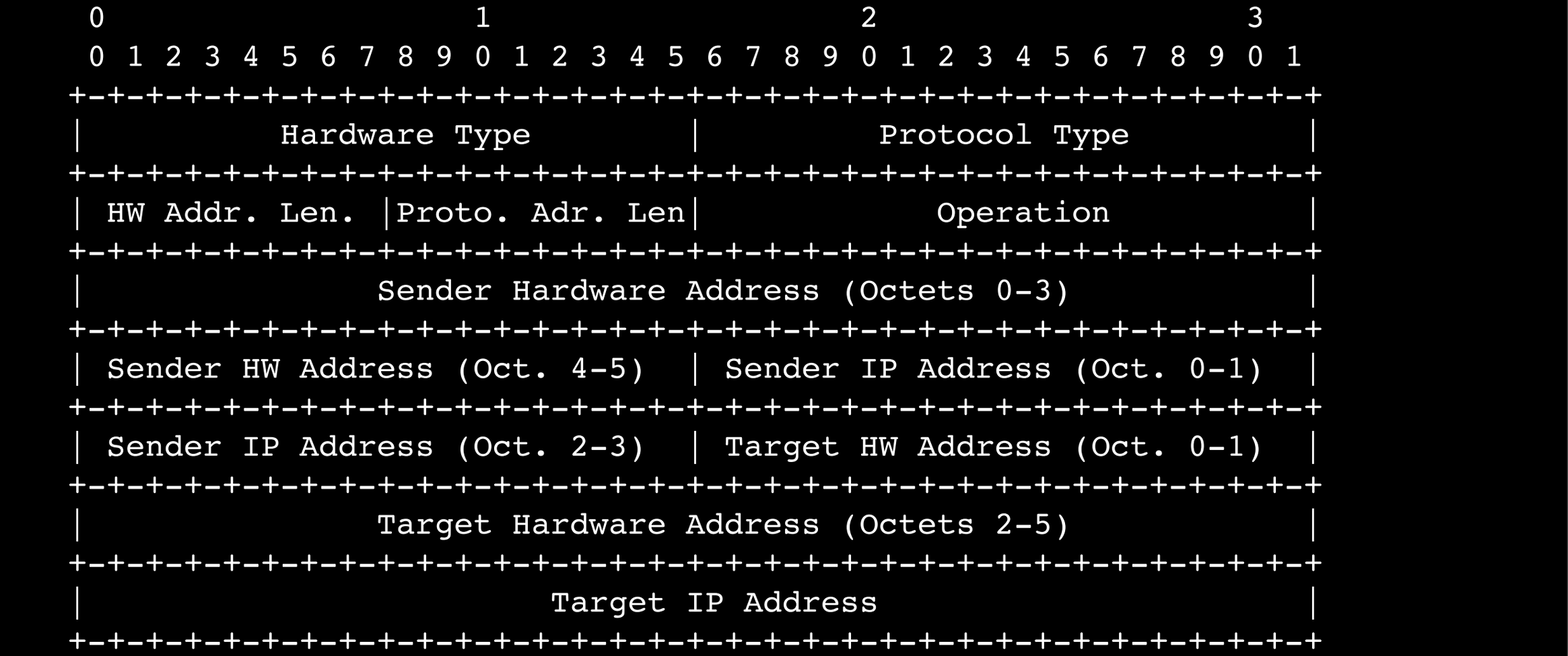


rfc 791

Address Resolution Protocol

- Who-is-who for the local network
- Mapping between layer 2 and 3 addressing

Address Resolution Protocol



rfc 826

Address Resolution Protocol

- Message examples:
- ARP Request or „where is ...?“
  Own IP and MAC are used as sender IP and HW address
  Searched-for IP is target IP
- ARP Reply or „I‘am at ...?“
  Own IP and MAC are used as sender IP and HW address

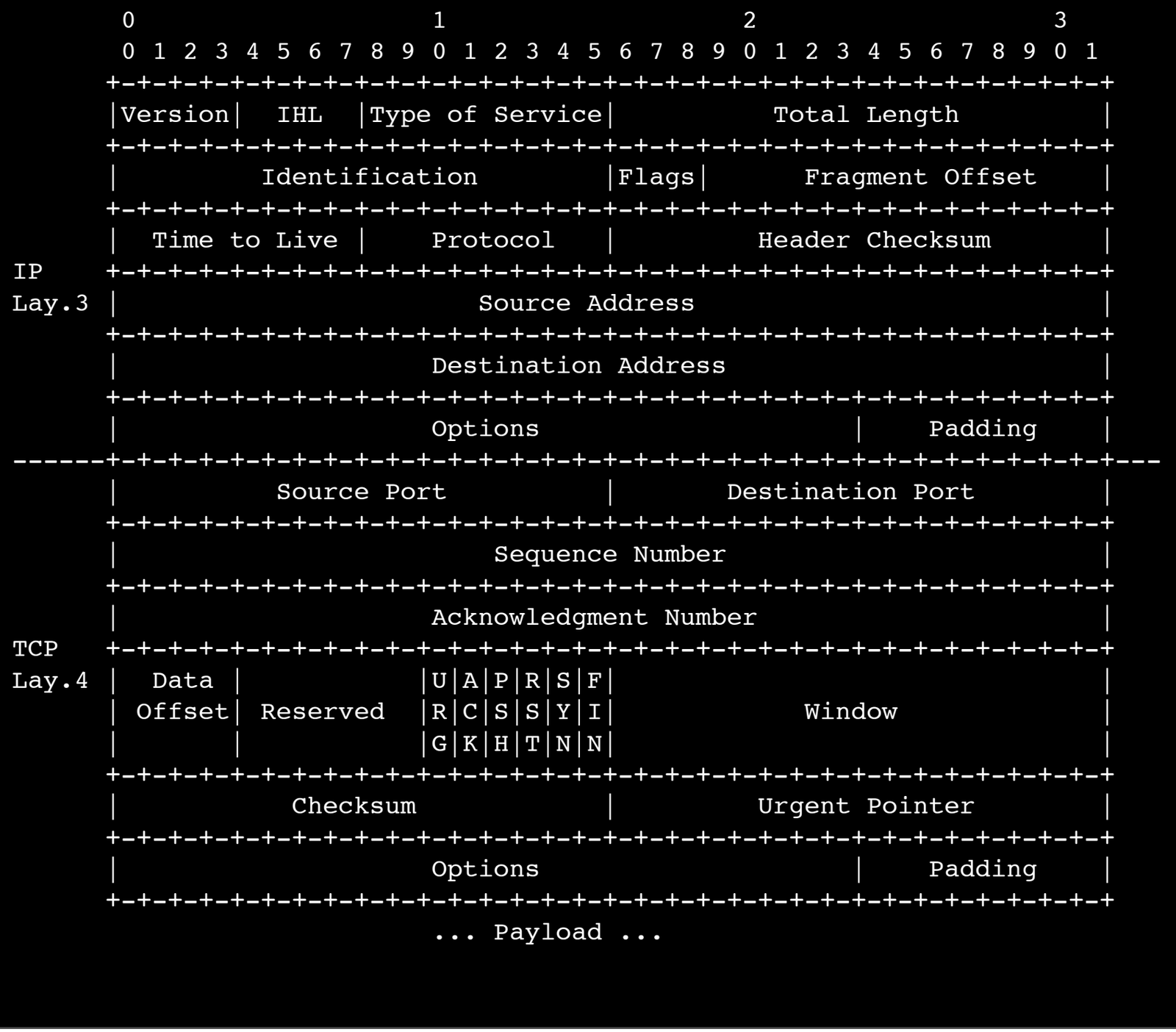
Transmission Control Protocol

- Layer 4 protocol
- Streams/Connections
- Ensures correct transmission or error
- 3-Way-Hand-Shake for connection establishment
- Ports

Transmission Control Protocol

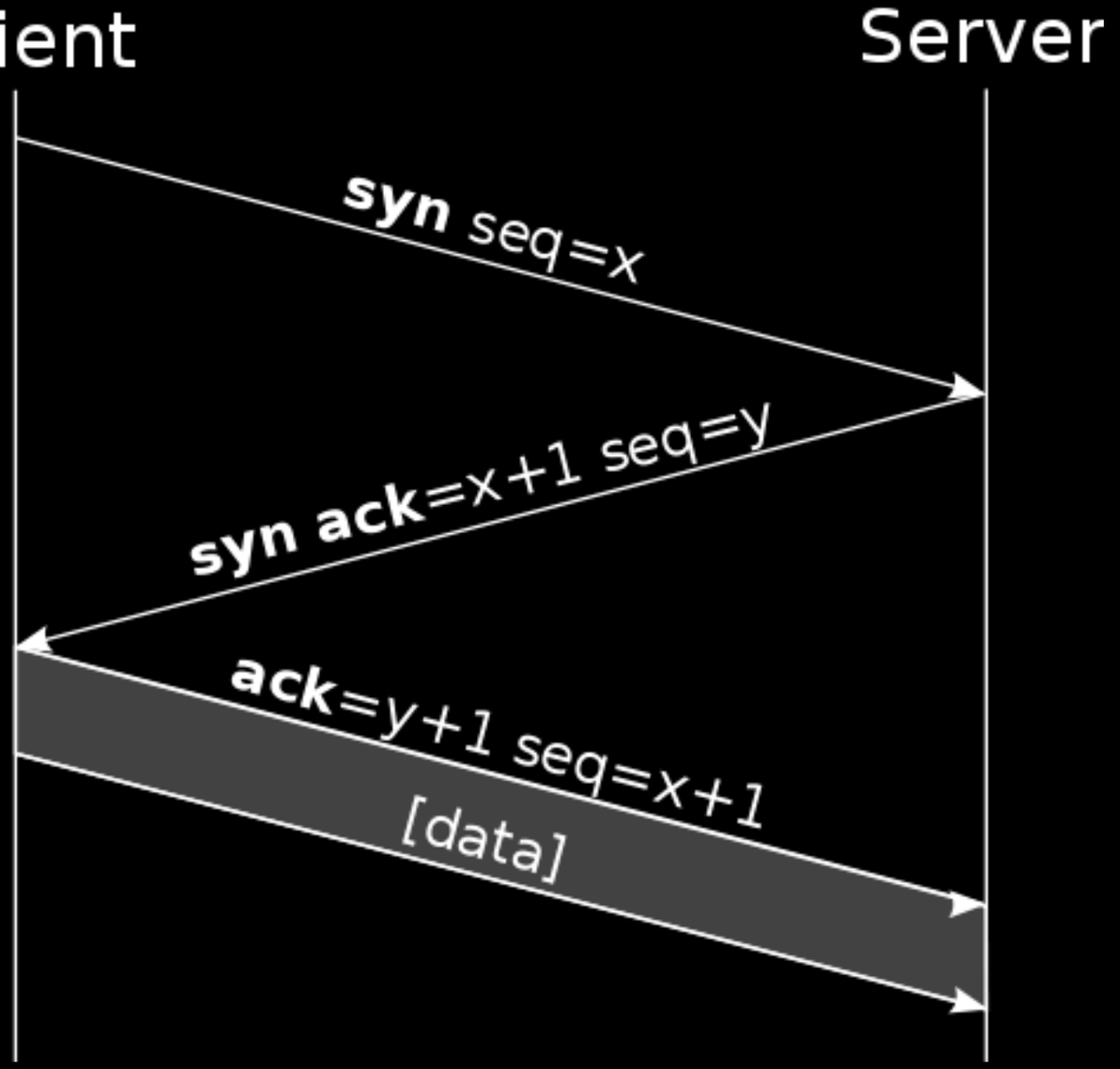
rfc 793

IP + TCP


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
IP
Lay.3
Source Address
Destination Address
Options
Padding
Source Port
Destination Port
Sequence Number
Acknowledgment Number
TCP
Lay.4
Data Offset
Reserved
U R G
A C K
P S H
R S T
S Y N
F I N
Window
Checksum
Urgent Pointer
Options
Padding
... Payload ...

Transmission Control Protocol

- 3-way-hand-shake
- Ensures Client is actually talking to Server
- RST signals a closed port



Python Basics

- Indentation with white spaces (thus no {})
- Dynamic typing
- Automatic memory management
- Interactive shell

Python Basics

```
>>> def add(x,y):
...     return x+y
>>>
>>> a, b = 1234, 5678
>>> c = add(a, b)
>>>
>>> if c > 1000:
...     print 'Big', c
... else:
...     print 'Small', x, y
...
Big 6912
>>>
```

Python Basics

```
>>> li = ['a','b','c']
>>> # Iterate list
... for c in li:
...     print c.upper()
...
A
B
C
>>> # More complex boolean operation:
... if False or len(li):
...     print li[2]
...
c
>>>
```

Python Basics

```
>>> list()
[]
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
>>> # Getting information on available functions and
>>> # objects:
>>> help([])
[...]
>>> help(str)
[...]
>>> help(var)
[...]
>>>
```

Scapy

```
$ sudo scapy
Welcome to Scapy (2.1.0)
>>>
```

<<<Interactive>>>

Workshop

- SYN Scan
- Traceroute
- ARP Spoofing
- Promiscuous Scan
- ICMP Chat

SYN Scan

Simple port scanner

1. Initiate three-way-handshake by sending SYN to „victim“
2. Get responses (if any)
3. All SYN-ACKs are open ports
   All RSTs are closed ports
   No response means down or firewalled

SYN Scan

ans, uans= sr(...) - Layer 3 packet send/receive
ans is a list of send and answered packets
uans are the unanswered packets

SYN flag in IP is just „S“

SYN Scan

```
p = IP(dst="130.83.177.129")/TCP(dport=[22,80], \
    flags="S")

ans, uans = sr(p, timeout=5)

for s,r in ans:
    # response is SYN-ACK?
    if r.flags & 2:
        print "port",r.sport,"is open"
    else:
        print "port",r.sport,"is closed"
```

Traceroute

How do my packets travel the world?

1. Send packets with varying time-to-live, usually „pings“, but TCP SYNs work better
2. Get responses (ICMP time-exceeded)

Traceroute

ans, uans= sr(..., timeout=123)
use timeout to restrict waiting for unanswered packets, in seconds

attr=(0, 90) tells scapy to generate 90 packets with attributes 0 to 89

Traceroute

```
p = IP(dst="www.google.com",ttl=(0,30))/TCP(flags="S")

ans, uans = sr(p, timeout=5)

for s,r in ans:
    print s.ttl, r.src
```

ARP Spoofing

Clients use ARP packets to populate their own ARP table, even if they did not ask for it.

1. Get to-be-spoofed IP
2. Send ARP „is-at“ packets to all (broadcast) with to-be-spoofed IP as source.
3. Route received packets (linux can do that for us)

ARP Spoofing

sendp(...) - Layer 2 packet transmission
attribute inter makes scapy wait between packets and loop sends continuesly

Broadcast MAC: ff:ff:ff:ff:ff:ff

Linux routing can be enabled with:
echo 1 > /proc/sys/net/ipv4/ip_forwarding
(so others can continue using the internet)

ARP Spoofing

```
Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op="is-at",
    hwsrc=own-MAC, psrc=to-be-spoofed-IP)

sendp(p, loop=1, inter=1.5)
```

Promiscuous Scan

Who is sniffing in my LAN?

1. Send packets with invalid destination MAC to a valid IP
2. Check if that IP answered
3. Other methods exists

Promiscuous Scan

ans,uans = srp() - Layer 2 Send and Receive

ans answered packet tuples

uans unanswered packets

Possible invalid MAC: f1:ff:ff:ff:ff:f1

Promiscuous Scan

```
ans,uans = srp(Ether(dst= "f1:ff:ff:ff:ff:f1")/
ARP(pdst=targetnet),timeout=5)

if len(ans) > 0:
    print "The following clients are in promiscuous
mode:"
    for snd,rcv in ans:
        print rcv.src
        if type(rcv.payload) == type(ARP()):
            print rcv.payload.psrc
        else:
            print rcv.payload.src
else:
    print "Found no promiscuous mode enabled clients"
```

ICMP Chat

- Use payload of ICMP packets to transport messages
- use sniff() to receive (see help(sniff))

ICMP Chat

sniff(prn, lfilter)

lfilter will be evaluated with each packet, if it returns True, prn will be called

```
def foo(bar):
    return bar+23
```

foo would be the same as „lambda x: x+23“

ICMP Chat

```
chatpartner = "192.168.214.123"

while 1:
    line = sys.stdin.readline()
    send(IP(dst=chatpartner)/ICMP(type="echo-request")/
(line.strip()),verbose=0)

sniff(prn=lambda x: x.payload.payload.payload.load,
      lfilter=lambda x: x.haslayer(ICMP) and
          x.payload.dst==chatpartner)
```

Food for Thought

- Wired networks are boring? Hack WLAN!
- WLAN is also too boring? Hack GSM!
  - BBosmocom - GSM Stack for python
  - Uses (cheap) mobile phones with custom firmware as radio
- Missing scapy feature: Connection tracking

Questions?

Feedback?

Thank you!